(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 June 2017 (29.06.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/112864 A1

(51) International Patent Classification:
G06F 9/44 (2006.01) *G06Q 99/00* (2006.01)
G06F 15/173 (2006.01)

(21) International Application Number:
PCT/US2016/068296

(22) International Filing Date:
22 December 2016 (22.12.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/271,041 22 December 2015 (22.12.2015) US

(71) Applicant: **OPERA SOLUTIONS U.S.A., LLC** [US/US]; 10 Exchange Place, 11th Floor, Jersey City, NJ 07302 (US).

(72) Inventors: **BAR-OR, Amir**; 232 Old Fram Road, Newton, MA 02459 (US). **LIAO, Yuansong**; 6012 147th Place SE, Bellevue, WA 98006 (US). **SRINIVASAN, Laks**; 5209 Harriet Lane, Bethlehem, PA 18017 (US). **HUANG, Ding**; 350W 43rd Street, Apt. 38C, New York, NY 10036 (US). **WANG, Hong**; Pusan Rd. 930#, Shanghai, 200125 (CN).

(74) Agents: **NIKOLSKY, Mark, E.** et al.; McCarter & English, LLP, Four Gateway Center, 100 Mulberry Street, Newark, NJ 07102 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR RAPID DEVELOPMENT AND DEPLOYMENT OF REUSABLE ANALYTIC CODE FOR USE IN COMPUTERIZED DATA MODELING AND ANALYSIS

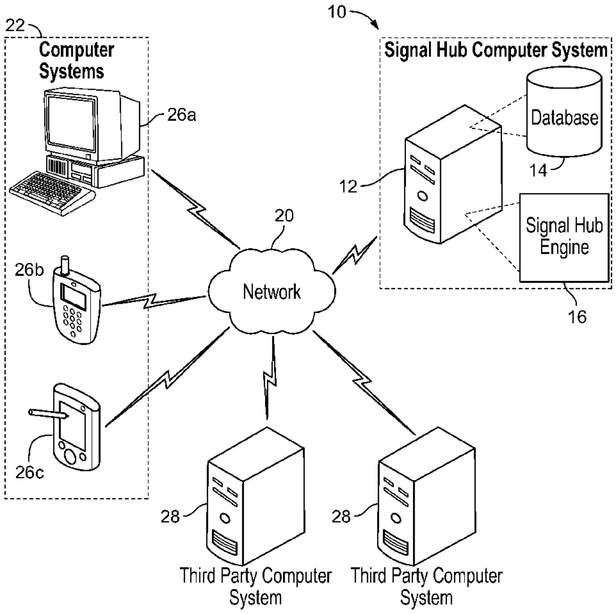

22
Computer Systems
26a
26b
26c
10
Signal Hub Computer System
Database
14
12
Signal Hub Engine
16
20
Network
28
Third Party Computer System
28
Third Party Computer System

FIG. 1

(57) Abstract: A system and method for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis is provided. The system includes a centralized, continually updated environment to capture pre-processing steps used in analyzing big data, such that the complex transformations and calculations become continually fresh and accessible to those investigating business opportunities. The system incorporates deep domain expertise as well as ongoing expertise in data science, big data architecture, and data management processes. In particular, the system allows for rapid development and deployment of analytic code that can easily be re-used in various data analytics applications, and on multiple computer systems.

Published:

— *with international search report (Art. 21(3))*

SYSTEM AND METHOD FOR RAPID DEVELOPMENT AND DEPLOYMENT OF REUSABLE ANALYTIC CODE FOR USE IN COMPUTERIZED DATA MODELING AND ANALYSIS

SPECIFICATION

BACKGROUND

FIELD OF THE DISCLOSURE

The present disclosure relates generally to computer-based tools for developing and deploying analytic computer code. More specifically, the present disclosure relates to a system and method for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis.

RELATED ART

In today's information technology world, there is an increased interest in processing "big" data to develop insights (e.g., better analytical insight, better customer understanding, etc.) and business advantages (e.g., in enterprise analytics, data management processes, etc.). Customers leave an audit trail or digital log of the interactions, purchases, inquiries, and preferences through online interactions with an organization. Discovering and interpreting audit trails within big data provides a significant advantage to companies looking to realize greater value from the data they capture and manage every day. Structured, semi-structured, and unstructured data points are being generated and captured at an ever-increasing pace, thereby forming big data, which is typically defined in terms of velocity, variety, and volume. Big data is fast-flowing, ever-growing, heterogeneous, and has exceedingly noisy input, and as a result transforming data into signals is critical. As more companies (e.g., airlines, telecommunications companies, financial institutions, etc.) focus on real-world use cases, the demand for continually refreshed signals will continue to increase.

Due to the depth and breadth of available data, data science (and data scientists) is required to transform complex data into simple digestible formats for quick interpretation and understanding. Thus, data science, and in particular, the field of data analytics, focuses on transforming big data into business value (e.g., helping companies anticipate customer behaviors and responses). The current analytic approach to capitalize on big data starts with raw data and ends with intelligence, which is then used to solve a particular business need so that data is ultimately translated into value.

However, a data scientist tasked with a well-defined problem (e.g., rank customers by probability of attrition in the next 90 days) is required to expend a significant amount of effort on tedious manual processes (e.g., aggregating, analyzing, cleansing, preparing, and transforming raw data) in order to begin conducting analytics. In such an approach, significant effort is spent on data preparation (e.g., cleaning, linking, processing), and less is spent on analytics (e.g., business intelligence, visualization, machine learning, model building).

Further, usually the intelligence gathered from the data is not shared across the enterprise (e.g., across use cases, business units, etc.) and is specific to solving a particular use case or business scenario. In this approach, whenever a new use case is presented, an entirely new analytics solution needs to be developed, such that there is no reuse of intelligence across different use cases. Each piece of intelligence that is derived from the data is developed from scratch for each use case that requires it, which often means that it’s being recreated multiple times for the same enterprise. There are no natural economies of scale in the process, and there are not enough data scientists to tackle the growing number of business opportunities while relying on such techniques. This can result in inefficiencies and waste, including lengthy use case execution and missed business opportunities.

Currently, to conduct analytics on “big” data, data scientists are often required to develop large quantities of software code. Often, such code is expensive to develop, is highly customized, and is not easily adopted for other uses in the analytics field. Minimizing redundant costs and shortening development cycles requires significantly reducing the amount of time that data scientists spend managing and coordinating raw data. Further, optimizing this work can allow data scientists to improve their effectiveness by honing signals and ultimately improving the foundation that drives faster results and business responsiveness. Thus, there is a need for a system to rapidly develop and deploy analytic code for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis.

SUMMARY

The present disclosure relates to a system and method for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis. The system includes a centralized, continually updated environment to capture pre-processing steps used in analyzing big data, such that the complex transformations and calculations become continually fresh and accessible to those investigating business opportunities. This centralized, continually refreshed system provides a data-centric competitive advantage for users (e.g., to serve customers better, reduce costs, etc.), as it provides the foresight to anticipate future problems and reuses development efforts. The system incorporates deep domain expertise as well as ongoing expertise in data science, big data architecture, and data management processes. In particular, the system allows for rapid development and deployment of analytic code that can easily be re-used in various data analytics applications, and on multiple computer systems.

Benefits of the system include a faster time to value as data scientists can now assemble pre-existing ETL (extract, transform, and load) processes as well as signal generation components to tackle new use cases more quickly. The present disclosure is a technological solution for coding and developing software to extract information for “big data” problems. The system design allows for increased modularity by integrating with various other platforms seamlessly. The system design also incorporates a new technological solution for creating “signals” which allows a user to extract information from “big data” by focusing on high-level issues in obtaining the data the user desires and not having to focus on the low-level minutia of coding big data software as was required by previous systems. The present disclosure allows for reduced software development complexity, quicker software development lifecycle, and reusability of software code.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing features of the disclosure will be apparent from the following Detailed Description, taken in connection with the accompanying drawings, in which:

FIG. 1 is a diagram illustrating hardware and software components of the system;

FIG. 2 is a diagram of a traditional data signal architecture;

FIG. 3 is a diagram of a new data signal architecture provided by the system;

FIGS. 4A-4C are diagrams illustrating the system in greater detail;

FIG. 5 is a screenshot illustrating an integrated development environment generated by the system;

FIG. 6 is a diagram illustrating signal library and potential use cases of the system;

FIGS. 7 is a diagram illustrating analytic model development and deployment carried out by the system;

FIG. 8 is a diagram illustrating hardware and software components of the system in one implementation;

FIGS. 9-10 are diagrams illustrating hardware and software components of the system during development and production;

FIG. 11 is a screenshot illustrating data profiles for each column using the integrated development environment generated by the system;

FIG. 12 is a screenshot illustrating profiling of raw data using the integrated development environment generated by the system;

FIG. 13 is a screenshot illustrating displaying of specific entries within raw data using the integrated development environment generated by the system;

FIG. 14 is a screenshot illustrating aggregating and cleaning of raw data using the integrated development environment generated by the system;

FIG. 15 is a screenshot illustrating managing and confirmation of raw data quality using the integrated development environment generated by the system;

FIG. 16 is a screenshot illustrating auto-generated visualization of a data model created using the integrated development environment;

FIG. 17A is a screenshot illustrating creation of reusable analytic code using the Workbench **500** generated by the system;

FIG. 17B is a screenshot illustrating the graphical user interface generated by the Signal Builder component of the Workbench of the system;

FIG. 18 is a screenshot illustrating a user interface screen generated by the system for visualizing signal paths using the Knowledge Center generated by the system;

FIG. 19 is a screenshot illustrating a user interface screen generated by the system for visualizing a particular signal using the Knowledge Center generated by the system;

FIG. 20A is a screenshot illustrating a user interface screen generated by the system for finding a signal using the Knowledge Center generated by the system;

FIG. 20B is a screenshot illustrating a user interface screen generated by the system for finding a signal using the Knowledge Center **600** generated by the system;

FIGS. 21A-F are screenshots illustrating user interface screens generated by the system for selecting entries with particular signal values using the Knowledge Center generated by the system;

FIG. 22 is a screenshot illustrating a user interface screen generated by the system for visualizing signal parts of a signal using the Knowledge Center generated by the system;

FIG. 23A is a screenshot illustrating a user interface screen generated by the system for visualizing a lineage of a signal using the Knowledge Center generated by the system;

FIG. 23B is a screenshot illustrating a user interface screen generated by the system for displaying signal values, statistics and visualization of signal value distribution;

FIG. 24A is a screenshot illustrating preparation of data to train a model using the integrated development environment generated by the system;

FIG. 24B is a screenshot illustrating a graphical user interface generally by the system of allowing users to select from a variety of model algorithms (e.g., logistic regression, deep autoencoder, etc.);

FIG. 24C is a screenshot illustrating the different parameter experiments users can apply during the model training process;

FIGS. 24D-J are screenshots illustrating the model training process in greater detail;

FIG. 25A is a screenshot illustrating training of a model using the Workbench subsystem of the present disclosure;

FIG. 25B is a screenshot illustrating preparation of data to train a model using the Workbench subsystem of the present disclosure;

FIG. 25C is a screenshot illustrating different data splitting options provided by the Workbench subsystem of the present disclosure**;**

FIG. 26 is another screenshot illustrating loading an external model trained outside of the integrated development environment;

FIG. 27 is a screenshot illustrating scoring a model using the integrated development environment generated by the system;

FIG. 28 is a screenshot illustrating monitoring model performance using the integrated development environment generated by the system;

FIG. 29A is a screenshot illustrating a solution dependency diagram of the integrated development environment generated by the system;

FIG. 29B is a screenshot illustrating a collaborative analytic solution development using the Workbench subsystem of the present disclosure;

FIGS. 29C-29J are screenshots illustrating environment files for enhancing collaboration;

FIGS. 30A-32 are screenshots illustrating the Signal Hub manager generated by the system; and

FIG. 33 is a diagram showing hardware and software components of the system.

DETAILED DESCRIPTION

Disclosed herein is a system and method for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis, as discussed in detail below in connection with **FIGS. 1-33**.

As used herein, the terms "signal" and "signals" refers to the data elements, patterns, and calculations that have, through scientific experimentation, been proven valuable in predicting a particular outcome. Signals can be generated by the system using analytic code that can be rapidly developed, deployed, and reused. Signals carry useful information about behaviors, events, customers, systems, interactions, attributes, and can be used to predict future outcomes. In effect, signals capture underlying drivers and patterns to create useful, accurate inputs that are capable of being processed by a machine into algorithms. High-quality signals are necessary to distill the relationships among all the entities surrounding a problem and across all the attributes (including their time dimension) associated with these entities. For many problems, high-quality signals are as important in generating an accurate prediction as the underlying machine-learning algorithm that acts upon these signals in creating the prescriptive action.

The system of the present disclosure is referred to herein as "Signal Hub." Signal Hub enables transforming data into intelligence as analytic code and then maintaining the intelligence as signals in a computer-based production environment that allows an entire organization to access and exploit the signals for value creation. In a given domain, many signals can be similar and reusable across different use cases and models. This signal-based approach enables data scientists to "write once and reuse everywhere," as opposed to the traditional approach of "write once and reuse never." The system provides signals (and the accompanying analytic code) in the fastest, most cost-effective method available, thereby accelerating the development of data science applications and lowering the cost of internal development cycles. Signal Hub allows ongoing data management tasks to be performed by systems engineers, shifting more mundane tasks away from scarce data scientists.

Signal Hub integrates data from a variety of sources, which enables the process of signal creation and utilization by business users and systems. Signal Hub provides a layer of maintained and refreshed intelligence (e.g., Signals) on top of the raw data that serves as a repository for scientists (e.g., data scientists) and developers (e.g., application developers) to execute analytics. This prevents users from having to go back to the raw data for each new use case, and can instead benefit from existing signals stored in Signal Hub. Signal Hub continually extracts, stores, refreshes, and delivers the signals needed for specific applications, such that application developers and data scientists can work directly with signals rather than raw data. As the number of signals grows, the model development time shrinks. In this "bow tie" architecture, model developers concentrate on creating the best predictive models with expedited time to value for analytics. Signal Hub is highly scalable in terms of processing large amounts of data as well as supporting the implementation of a myriad of use cases. Signal Hub could be enterprise-grade, which means that in addition to supporting industry-standard scalability and security features, it is easy to integrate with existing systems and workflows. Signal Hub can also have a data flow engine that is flexible to allow processing of different computing environments, languages, and frameworks. A multi target system data flow compiler can generate code to deploy on different target data flow engines utilizing different computer environments, languages, and frameworks. For applications with hard return on investment (ROI) metrics (e.g., churn reduction), faster time to value can equate to millions of dollars earned. Additionally, the system could lower development costs as data science project timelines potentially shrink, such as from 1 year to 3 months (e.g., a 75% improvement). Shorter development cycles and lower development costs could result in increased accessibility of data science to more parts of the business. Further, the system could reduce the total costs of ownership (TCO) for big data analytics.

FIG. 1 is a diagram illustrating hardware and software components of the system. The system **10** includes a computer system **12** (e.g., a server) having a database **14** stored therein and a Signal Hub engine **16**. The computer system **12** could be any suitable computer server or cluster of servers (e.g., a server with an INTEL microprocessor, multiple processors, multiple processing cores, etc.) running any suitable operating system (e.g., Windows by Microsoft, Linux, Hadoop, etc.). The database **14** could be stored on the computer system **12**, or located externally therefrom (e.g., in a separate database server in communication with the system **10**).

The system **10** could be web-based and remotely accessible such that the system **10** communicates through a network **20** with one or more of a variety of computer systems **22** (e.g., personal computer system **26a**, a smart cellular telephone **26b**, a tablet computer **26c**, or other devices). Network communication could be over the Internet using standard TCP/IP communications protocols (e.g., hypertext transfer protocol (HTTP), secure HTTP (HTTPS), file transfer protocol (FTP), electronic data interchange (EDI), etc.), through a private network connection (e.g., wide-area network (WAN) connection, emails, electronic data interchange (EDI) messages, extensible markup language (XML) messages, file transfer protocol (FTP) file transfers, etc.), or any other suitable wired or wireless electronic communications format. Further, the system **10** could be in communication through a network **20** with one or more third party servers **28**. These servers **28** could be disparate "compute" servers on which analytics could be performed (e.g., Hadoop, etc.). The Hadoop system can manage resources (e.g., split workload and/or automatically optimize how and where computation is performed). For example, the system could be fully or partially executed on Hadoop, a cloud-based implementation, or a stand-alone implementation on a single computer. More specifically, for example, system development could be executed on a laptop, and production could be on Hadoop, where Hadoop could be hosted in a data center.

FIGS. 2-3 are diagrams comparing traditional signal architecture **40** and new data signal architecture **48** provided by the system. As shown, in the traditional signal architecture **40** (e.g., the spaghetti architecture), for every new use case **46**, raw data **42** is transformed through processing steps **44**, even if that raw data **42** had been previously transformed for a different use case **46**. More specifically, a data element **42** must be processed for use in a first use case **46**, and that same data element must be processed again for use in a second use case **46**. In particular, the analytic code written to perform the processing steps **44** cannot be easily re-used. Comparatively, in the new data signal architecture **48** (e.g., the bowtie architecture) of the present disclosure, raw data **50** is transformed into descriptive and predictive signals **52** only once. Advantageously, the analytic code generated by the system for each signal **52** can be rapidly developed, deployed, and re-used with many of the use cases **54**.

Signals are key ingredients to solving an array of problems, including classification, regression, clustering (segmentation), forecasting, natural language processing, intelligent data design, simulation, incomplete data, anomaly detection, collaborative filtering, optimization, etc. Signals can be descriptive, predictive, or a combination thereof. For instance, Signal Hub can identify high-yield customers who have a high propensity to buy a discounted ticket to destinations that are increasing in popularity. Descriptive signals are those which use data to evaluate past behavior. Predictive signals are those which use data to predict future behavior. Signals become more powerful when the same data is examined over a (larger) period of time, rather than just an instance.

Descriptive signals could include purchase history, usage patterns, service disruptions, browsing history, time-series analysis, etc. As an example, an airline trying to improve customer satisfaction may want to know about the flying experiences of its customers, and it may be important to find out if a specific customer had his/her last flight cancelled. This is a descriptive signal that relies on flight information as it relates to customers. In this example, a new signal can be created to look at the total number of flight cancelations a given customer experienced over the previous twelve months. Signals can measure levels of satisfaction by taking into account how many times a customer was, for instance, delayed or upgraded in the last twelve months.

Descriptive signals can also look across different data domains to find information that can be used to create attractive business deals and/or to link events over time. For example, a signal may identify a partner hotel a customer tends to stay with so that a combined discounted deal (e.g., including the airline and the same hotel brand) can be offered to encourage the customer to continue flying with the same airline. This also allows for airlines to benefit from and leverage the customer’s satisfaction level with the specific hotel partner. In this way, raw input data is consolidated across industries to create a specific relationship with a particular customer. Further, a flight cancelation followed by a hotel stay could indicate that the customer got to the destination but with a different airline or a different mode of transportation.

Predictive signals allow for an enterprise to determine what a customer will do next or how a customer will respond to a given event and then plan appropriately. Predictive signals could include customer fading, cross-sell/up-sell, propensity to buy, price sensitivity, offer personalization, etc. A predictive signal is usually created with a use case in mind. For example, a predictive signal could cluster customers that tend to fly on red-eye flights, or compute the propensity level a customer has for buying a business class upgrade.

Signals can be categorized into classes including sentiment signals, behavior signals, event/anomaly signals, membership/cluster signals, and correlation signals. Sentiment signals capture the collective prevailing attitude about an entity (e.g., consumer, company, market, country, etc.) given a context. Typically, sentiment signals have discrete states, such as positive, neutral, or negative (e.g., current sentiment on X corporate bonds is positive.). Behavior signals capture an underlying fundamental behavioral pattern for a given entity or a given dataset (e.g., aggregate money flow into ETFs, number of “30 days past due” in last year for a credit card account, propensity to buy a given product, etc.). These signals are most often a time series and depend on the type of behavior being tracked and assessed. Event/Anomaly signals are discrete in nature and are used to trigger certain actions or alerts when a certain threshold condition is met (e.g., ATM withdrawal that exceeds three times the daily average, bond rating downgrade by a rating agency), etc. Membership/Cluster signals designate where an entity belongs, given a dimension. For example, gaming establishments create clusters of their customers based on spending (e.g., high rollers, casual gamers, etc.), or wealth management firms can create clusters of their customers based on monthly portfolio turnover (e.g., frequent traders, buy and hold, etc.). Correlation signals continuously measure the correlation of various entities and their attributes throughout a time series of values between 0 and 1 (e.g., correlation of stock prices within a sector, unemployment and retail sales, interest rates and GDP, home prices and interest rates, etc.).

Signals have attributes based on their representation in time or frequency domains. In a time domain, a Signal can be continuous (e.g., output from a blood pressure monitor) or discrete (e.g., daily market close values of the Dow Jones Index). Within the frequency domain, signals can be defined as high or low frequency (e.g., asset allocation trends of a brokerage account can be measured every 15 minutes, daily, and monthly). Depending on the frequency of measurement, a signal derived from the underlying data can be fast-moving or slow-moving.

Signals are organized into signal sets that describe (e.g., relate to) specific business domains (e.g. customer management). Signal sets are industry-specific and cover domains including customer management, operations, fraud and risk management, maintenance, network optimization, digital marketing, etc. Signal Sets could be dynamic (e.g., continually updated as source data is refreshed), flexible (e.g., adaptable for expanding parameters and targets), and scalable (e.g., repeatable across multiple use cases and applications).

FIGS. 4A-4B are diagrams illustrating the system in greater detail. The main components of Signal Hub **60** include an integrated development environment (Workbench) **62**, Knowledge Center (KC) **64**, and Signal Hub Manager ("SHM") **65**, and Signal Hub Server **66**. The Workbench **62** is an integrated software-based productivity tool for data scientists and developers, offering analytic functionalities and approaches for the making of a complete analytic solution, from data to intelligence to value. The Workbench **62** enables scientists to more effectively transform data to intelligence through the creation of signals. Additionally, the Workbench **62** allows data scientists to rapidly develop and deploy reusable analytic code for conducting analytics on various (often, disparate) data sources, on numerous computer platforms. The Knowledge Center **64** is a centralized place for institutional intelligence and memory and facilitates the transformation of intelligence to value through the exploration and consumption of signals. The Knowledge Center **64** enables the management and reuse of signals, which leads to scalability and increased productivity. The Signal Hub manager **65** provides a management and monitoring console for analytic operational stewards (e.g., IT, business, science, etc.). The Signal Hub manager **65** facilitates understanding and managing the production quality and computing resources with alert system. Additionally, the Signal Hub manager **65** provides role-based access control for all Signal Hub platform components to increase network security in an efficient and reliable way. The Signal Hub Server **66** executes analytics by running the analytic code developed in the Workbench **62** and producing the Signal output. The Signal Hub Server **66** provides fast, flexible and scalable processing of data, code, and artifacts (e.g., in Hadoop via a data-flow execution engine; Spark Integration). The Signal Hub Server **66** is responsible for the end-to-end processing of data and its refinement into signals, as well as enabling users to solve problems across industries and domains (e.g., making Signal Hub a horizontal platform).

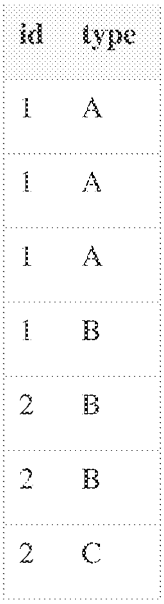
The platform architecture provides great deployment flexibility. It can be implemented on a single server as a single process (e.g., a laptop), or it can run on a large-scale Hadoop cluster with distributed processing, without modifying any code. It could also be implemented on a standalone computer. This allows scientists to develop code on their laptops and then move it into a Hadoop cluster to process large volumes of data. The

Signal Hub Server architecture addresses the industry need for large-scale production-ready analytics, a need that popular tools such as SAS and R cannot fulfill even today, as their basic architecture is fundamentally main memory–limited.

Signal Hub components include signal sets, ETL processing, dataflow engine, signal-generating components (e.g., signal-generation processes), APIs, centralized security, model execution, and model monitoring. The more use cases that are executed using Signal Hub **60**, the less time it takes to actually implement them over time because the answers to a problem may already exist inside Signal Hub **60** after a few rounds of signal creation and use case implementation. Signals are hierarchical, such that within Signal Hub **60**, a signal array might include simple signals that can be used by themselves to predict behavior (e.g., customer behavior powering a recommendation) and/or can be used as inputs into more sophisticated predictive models. These models, in turn, could generate second-order, highly refined signals, which could serve as inputs to business-process decision points.

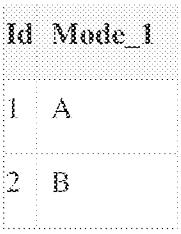
The design of the system and Signal Hub 60 allows users to use a single simple expression that represents multiple expressions of different levels of data aggregations. For example, suppose there is a dataset with various IDs. Each ID could be associated with an ID type which could also be associated with an occurrence of an event. One level of aggregation could be to determine for each ID and each ID type, the number of occurrence of an event. A second level of aggregation could be to determine for each ID, what is the most common type of ID based on the number of occurrence of an event. The system of the present disclosure allows this determination based on multiple layers of aggregation to be based on a single scalar expression and returning one expected output at one time. For example, using the code category_histogram(col), the system will create a categorical histogram for a given column, with each unique value in the column being considered a category. Using the code "mode(histogram, n = 1)," allows the system to return the category with the highest number of entries. If n > 1, retrieve the n'th most common value (2nd, 3rd...); if n < 0, retrieve the least common value (n = -1); and second least common (n = -2) etc. In the event several keys have equal frequencies, the smallest (if keys are numerical) or earliest (if keys are alphabetical) are returned. The following an example of a sample input and output based on the foregoing example.

Input:



id	type
1	A
1	A
1	A
1	B
2	B
2	B
2	C

Output:



Id	Mode_1
1	A
2	B

FIG. 4C is a screenshot of an event pattern matching feature of the system of the present disclosure. The system allows users to determine whether a specified sequence of events occurred in the data and then submit a query to retrieve information about the matched data. For example, in **FIG. 4C**, for the raw input data shown, a user can (1) define an event; (2) create a pattern matcher; and (3) query the pattern matcher to return the output as shown. As can be seen, a user can easily define with a regular expression an occurrence of a specified event such as "service fixed after call." Once the pattern matches algorithm is executed, a signal is extracted in the output showing the pattern occurrence.

FIG. 5 is a screenshot illustrating an Workbench **70** generated by the system. The Workbench **70** (along with the Knowledge Center) enables users to interact with the functionality and capabilities of the Signal Hub system via a graphical user interface (GUI). The Workbench **70** is an environment to develop end-to-end analytic solutions (e.g., a development environment for analytics) including reusable and easily developed analytic code. It offers all the necessary functionality for aggregating of the entire analytic modeling process, from data to signals. It provides an environment for the coding and development of data schemas, data quality management processes (e.g. missing value imputation and outlier detection), collections (e.g., the gathering of raw data files with the same data schema), views (e.g., logic to create a new relational dataset from other views or collections), descriptive and predictive signals, model validation and visualization (e.g., measuring of model performance through ROC (receiver operator characteristic), KS (Kolmogorov-Smirnov), Lorenz curves, etc.), visualization and maintenance of staging, input, output data models, etc. The Workbench **70** facilitates data ingestion and manipulating, as well as enabling data scientists to extract intelligence and value from data through signals (e.g., analytics through signal creation and computation).

The user interface of the Workbench could include components such as a tree view **72**, an analytic code development window **74**, and a supplementary display portion **76**. The tree view **72** displays each collection of raw data files (e.g., indicated by “Col” **73a**) as well as logical data views (e.g., indicated by “Vw” **73b**), as well as third-party code called as user defined functions if any (e.g., python, R, etc.). The analytic code development window **74** has a plurality of tabs including Design **78**, Run **80**, and Results **82**. The Design tab **78** provides a space where analytic code can be written by the developer. The Run tab **80** allows the developer to run the code and generate signal sets. Finally, the Results tab **82** allows the developer to view the data produced by the operations defined in the Run tab **80**.

The supplementary display portion **76** could include additional information including schemas **84** and dependencies **86**. Identifying, extracting, and calculating signals at scale from noisy big data requires a set of predefined signal schema and a variety of algorithms. A signal schema is a specific type of template used to transform data into signals. Different types of schema may be used, depending on the nature of the data, the domain, and/or the business environment. Initial signal discovery could fall into one or more of a variety of problem classes (e.g., regression classification, clustering, forecasting, optimization, simulation, sparse data inference, anomaly detection, natural language processing, intelligent data design, etc.). Solving these problem classes could require one or more of a variety of modeling techniques and/or algorithms (e.g., ARMA, CART, CIR++, compression nets, decision trees, discrete time survival analysis, D-Optimality, ensemble model, Gaussian mixture model, genetic algorithm, gradient boosted trees, hierarchical clustering, kalman filter, k-means, KNN, linear regression, logistic regression, Monte Carlo Simulation, Multinomial logistic regression, neural networks, optimization (LP, IP, NLP), poisson mixture model, Restricted Boltzmann Machine, Sensitivity trees, SVD, A-SVD, SVD++, SVM, projection on latent structures, spectral graph theory, etc.).

Advantageously, the Workbench **70** provides access to pre-defined libraries of such algorithms, so that they can be easily accessed and included in analytic code being generated. The user then can re-use analytic code in connection with various data analytics projects. Both data models and schemas can be developed within the Workbench **70** or imported from popular third-party data modeling tools (e.g., CA Erwin). The data models and schemas are stored along with the code and can be governed and maintained using modern software lifecycle tools. Typically, at the beginning of a Signal Hub project, the Workbench **70** is used by data scientists for profiling and schema discovery of unfamiliar data sources. Signal Hub provides tools that can discover schema (e.g., data types and column names) from a flat file or a database table. It also has built-in profiling tools, which automatically compute various statistics on each column of the data such as missing values, distribution parameters, frequent items, and more. These built-in tools accelerate the initial data load and quality checks.

Once data is loaded and discovered, it needs to be transformed from its raw form into a standard representation that will be used to feed the signals in the signal layer. Using the Workbench **70**, data scientists can build workflows composed of “views” that transform the data and apply data quality checks and statistical measures. The Signal Hub platform can continuously execute these views as new data appears, thus keeping the signals up to date.

The dependencies tab **86** could display a dependency diagram (e.g., a graph) of all the activities comprising the analytic project, as discussed below in more detail. A bottom bar **88** could include compiler information, such as the number of errors and warnings encountered while processing views and signal sets.

FIG. 6 is a diagram **90** illustrating use cases (e.g., outputs, signals, etc.) of the system. There could be multiple signal libraries, each with subcategories for better navigation and signal searching. For example, as shown, the Signal Hub could include a Customer Management signal library **92**. Within the Customer Management Signal

Library **92** are subcategories for Flight **94**, Frequent Flyer Program **96**, Partner **98**, and Ancillary **99**. The Flight subcategory **94** could include, for example, "Signal 345. Number of times customer was seated in middle seat in the past 6 months," "Signal 785. Number of trips customer has made on a weekend day in past 1 year," "Signal 956. Number of flights customer with < 45 mins between connections," "Signal 1099. Indicates a customer has been delayed more than 45 minutes in last 3 trips," "Signal 1286. Number of involuntary cancellations experienced by the customer in past 1 year," etc. The Frequent Flyer Program subcategory **96** could include, for example, "Signal 1478. % of CSat surveys taken out of total flights customer has flown in past 1 month," "Signal 1678. Number of complimentary upgrades a member received in past 6 months," "Signal 2006. Ratio of mileage earned to mileage used by a member in past 1 year," "Signal 2014. Average # of days before departure when an upgrade request is made by member," "Signal 2020. Number upgrades redeemed using mileage in past 1 year," etc. The Partner subcategory **98** could include, for example, "Signal 563. Mileage earned using Cable Company (TM) in past 1 month," "Signal 734. Number of partners with whom that customer has engaged in the past 6 months," "Signal 737. Mileage earned via Rental Car in past 1 yr," "Signal 1729. Number of emails received about Luxury Hotel in the past 3 months," "Signal 1993. Number of times customer booked hotel with Airlines' partner without booking associated flight in the past 1 year," etc. The Ancillary subcategory **99** could include, for example, "Signal 328. Number of times customer has had baggage misplaced in past 3 months," "Signal 1875. Total amount spent on check bags in past 1 month," "Signal 1675. Number of times wifi was unavailable on customer's flight," "Signal 1274. Number of emails received pertaining to bags in last 1 year," "Signal 1564. Number of times customer has purchased duty free on board," etc.

FIG. 7 is a diagram illustrating analytic model development and deployment carried out by the system. In step **202**, a user defines a business requirement (e.g., business opportunity, business problem) needing analyzing. In step **204**, one or more analytics requirements are defined. In step **214**, the user searches for signals, and if an appropriate signal is found, the user selects the signal. If a signal is not found, then in step **212**, the user creates one or more signals by identifying the aggregated and cleansed data to base the signal on. After the signal is created the process then proceeds to step **214**. If the raw data is not available to create the signal in step **212**, then in step **208** the user obtains the raw data, and in step **210**, the data is aggregated and cleansed, and then the process proceeds to step **212**. It is noted that the system of the present disclosure facilitates skipping steps **208**-**212** (unlike the traditional approach which must proceed through such steps for every new business requirement).

Once the signals are selected, then in step **216**, solutions and models are developed based on the signals selected. In step **218**, results are evaluated and if necessary, signals (e.g., created and/or selected) and/or solutions/models are revised accordingly. Then in step **220**, the solutions/models are deployed. In step **222**, results are monitored and feedback gathered to incorporate back into the signals and/or solutions/models.

FIG. 8 is a diagram **250** illustrating hardware and software components of the system in one implementation. Other implementations could be implemented. The workflow includes model-building tools **252**, Hadoop/YARN and Signal Hub processing steps **254**, and Hadoop Data Lake (Hadoop Distributed file system (HDFS) and HIVE) databases **256**.

The Signal Hub Server is able to perform large-scale processing of terabytes of data across thousands of Signals. It follows a data-flow architecture for processing on a Hadoop cluster (e.g., Hadoop 2.0). Hadoop 2.0 introduced YARN (a large-scale, distributed operating system for big data applications), which allows many different data processing frameworks to coexist and establishes a strong ecosystem for innovating technologies. With YARN, Signal Hub Server solutions are native certified Hadoop applications that can be managed and administered alongside other applications. Signal Hub users can leverage their investment in Hadoop technologies and IT skills and run Signal Hub side-by-side with their current Hadoop applications.

Raw data is stored in the raw data database **258** of the Hadoop Data Lake **256**. In step **260**, Hadoop/Yarn and Signal Hub **254** process the raw data **258** with ETL (extract, transform, and load) modules, data quality management modules, and standardization modules. The results of step **260** are then stored in a staging database **262** of the Hadoop Data Lake. In step **260**, Hadoop/Yarn and Signal Hub **254** process the staging data **262** with signal calculation modules, data distribution modules, and sampling modules. The results of step **264** are then stored in the Signals and Model Input database **266**. In step **268**, the model development and validation module **268** of the model building tools **252** processes the signals and model input data **266**. The results of step **268** are then stored in the model information and parameters database **270**. In step **272**, the model execution module **272** of the Hadoop/Yarn and Signal Hub **254** processes signals and model input data **266** and/or model information and parameters data **270**. The results of step **272** are then stored in the model output database **274**. In step **276**, the Hadoop/Yarn and Signal Hub **254** processes the model output data **274** with a business rules execution output transformation for business intelligence and case management user interface. The results of step **276** are then stored in the final output database **278**. Enterprise applications **280** and business intelligence systems **282** access the final output data **278**, and can provide feedback to the system which could be integrated into the raw data **258**, the staging data **262**, and/or the signals and model input **266**.

The Signal Hub Server automates the processing of inputs to outputs. Because of its data flow architecture, it has a speed advantage. The Signal Hub Server has multiple capabilities to automate server management. It can detect data changes within raw file collections and then trigger a chain of processing jobs to update existing Signals with the relevant data changes without transactional system support.

FIGS. 9-10 are diagrams illustrating hardware and software components of the system during development and production. More specifically, **FIG. 9** is a diagram **300** illustrating hardware and software components of the system during development and production. Source data **302** is in electrical communication with Signal Hub **304**. Signal Hub **304** comprises a Workbench **306**, and a Knowledge Center **308**. Signal Hub **304** could also include a server in electronic communication with the Workbench **306** and the Knowledge Center **308**, such as via Signal Hub manager **312**. Signal Hub further comprises infrastructure **314** (e.g., Hadoop, YARN, etc.) and hosting options **316**, such as Client, Opera, and Virtual Cloud (e.g., AWS).

Signal Hub **304** allows companies to absorb information from various data sources **302** to be able to address many types of problems. More specifically, Signal Hub **304** can ingest both internal and external data as well as structured and unstructured data. As part of the Hadoop ecosystem, the Signal Hub Server can be used together with tools such as Sqoop or Flume to digest data after it arrives in the Hadoop system. Alternatively, the Signal Hub Server can directly access any JDBC (Java Database Connectivity) compliant database or import various data formats transferred (via FTP, SFTP, etc.) from source systems.

Signal Hub **304** can incorporate existing code **318** coded in various (often non-compatible) languages (e.g., Python, R, Unix Shell, etc.), called from the Signal Hub platform as user defined functions. Signal hub **304** can further communicate with modeling tools **320** (e.g., SAS, SPSS, etc.), such as via flat file, PMML (Predictive Model Markup Language), etc. The PMML format is a file format describing a trained model. A model developed in SAS, R, SPSS, or other tools can be consumed and run within Signal Hub **304** via the PMML standard. Advantageously, such a solution allows existing analytic code that may be written in various, non-compatible languages (e.g., SAS, SPSS, Python, R, etc.) to be seamlessly converted and integrated for use together within the system, without requiring that the existing code be re-written. Additionally, Signal Hub **304** can create tests and reports as needed. Through the Workbench, descriptive signals can be exported into a flat file for the training of predictive models outside Signal Hub **304**. When the model is ready, it can then be brought back to Signal Hub **304** via the PMML standard. This feature is very useful if a specific machine-learning technique is not yet part of the model repertoire available in Signal Hub **304**. It also allows Signal Hub **304** to ingest models created by clients in third-party analytic tools (including R, SAS, SPSS). The use of PMML allows Signal Hub users to benefit from a high level of interoperability among systems where models built in any PMML-compliant analytics environment can be easily consumed. In other words, because the system can automatically convert existing (legacy) analytic code modules/libraries into a common format that can be executed by the system (e.g., by automatically converting such libraries into PMML-compliant libraries that are compatible with other similarly compliant libraries), the system thus permits easy integration and re-use of legacy analytic code, interoperably with other modules throughout the system.

Signal Hub **304** integrates seamlessly with a variety of front-end systems **322** (e.g., use-case specific apps, business intelligence, customer relationship management (CRM) system, content management system, campaign execution engine, etc.). More specifically, Signal Hub **304** can communicate with front end systems **322** via a staging database (e.g., MySQL, HIVE, Pig, etc.). Signals are easily fed into visualization tools (e.g. Pentaho, Tableau), CRM systems, and campaign execution engines (e.g. Hubspot, ExactTarget). Data is transferred in batches, written to a special data landing zone, or accessed on-demand via APIs (application programming interfaces). Signal Hub **304** could also integrate with existing analytic tools, pre-existing code, and models. Client code can be loaded as an external library and executed within the server. All of this ensures that existing client investments in analytics can be reused with no need for recoding.

The Workbench **306** could include a workflow to process signals that includes loading **330**, data ingestion and preparation **332**, descriptive signal generation **336**, use case building **338**, and sending **340**. In the loading step **330**, source data is loaded into the Workbench **306** in any of a variety of formats (e.g., SFTP, JDBC, Sqoop, Flume, etc.). In the data ingestion and preparation step **332**, the Workbench **306** provides the ability to process a variety of big data (e.g., internal, external, structured, unstructured, etc.) in a variety of ways (e.g., delta processing, profiling, visualizations, ETL, DQM, workflow management, etc.). In the descriptive signal generation step **334**, a variety of descriptive signals could be generated (e.g., mathematical transformations, time series, distributions, pattern detection, etc.). In the predictive signal generation step **336**, a variety of predictive signals could be generated (e.g., linear regression, logistic regression, decision tree, Naïve Bayes, PCA, SVM, deep autoencoder, etc.). In the use case building step **338**, uses cases could be created (e.g., reporting, rules engine, workflow creator, visualizations, etc.). In the sending step **340**, the Workbench **306** electronically transmits the output to downstream connectors (e.g., APIs, SQL, batch file transfer, etc.).

FIG. 10 is a diagram **350** illustrating hardware and software components of the system during production. As discussed in **FIG. 9**, Signal Hub includes an Workbench **352**, a Knowledge Center **354**, and a Signal Hub Manager **356**. The Workbench **352** could communicate with an execution layer **360** via a compiler **358**. The Knowledge Center **354** and Signal Hub manager **356** could directly communicate with the execution layer **360**. The execution layer **360** could include a workflow server **362**, a plurality of flexible data flow engines **364**, and an operational graph database **366**. Signal Hub further comprises infrastructure **366** (e.g., Hadoop, YARN, etc.) and hosting options **370**, such as Client, Opera, and Virtual Private Cloud (e.g., AWS, Amazon, etc.). The plurality of flexible data flow engines **364** can have the latest cutting-edge technology.

FIGS. 11-17 are screenshots illustrating use of the Signal Hub platform to create descriptive signals. The Workbench user interface **500** includes a tree view **502** and an analytic code development window **504**. The Workbench provides direct access to the Signal API, which speeds up development and simplifies (e.g., reduce errors in) signal creation (e.g., descriptive signals). The Signal API provides an ever-growing set of mathematical transformations that will allow for the creation of powerful descriptive signals, along with a syntax that is clear, concise, and expressive. Signal API allows scientists to veer away from the implementation details and focus solely on data analysis, thus maximizing productivity and code reuse. For example, the Signal API allows for easy implementation of complex pattern-matching signals. For example, for the telecom industry, one pattern could be a sequence of events in the data that are relevant for measuring attrition, such as a widespread service disruption followed by one or more customer complaints followed by restored service. The Signal API also provides a direct link between the Workbench and the Knowledge Center. Users can add metatags and descriptions to signals directly in Signal API code (which is reusable analytic code). These tags and taxonomy information are then used by the Knowledge Center to enable signal search and reuse, which greatly enhances productivity.

As for predictive signals, training and testing of models can easily be done in the Workbench through its intuitive and interactive user interface. Current techniques available for modeling and dimensionality reduction include SVMs, k-means, decision trees, association rules, linear and logistic regression, neural networks, RBM (machine-learning technique), PCA, and Deep AutoEncoder (machine-learning technique) which allows data scientists to train and score deep-learning nets. Some of these advanced machine-learning techniques (e.g., Deep AutoEncoder and RBM) project data from a high-dimensional space into a lower-dimensional one. These techniques are then used together with clustering algorithms to understand customer behavior.

FIG. 11 is a screenshot illustrating data profiles for each column (e.g., number of unique, number of missing, average, max, min, etc.) using the Workbench **500** generated by the system. As described above, the Workbench user interface could include sets of components including a tree view **502**, an analytic code development window **504**, and a supplementary display portion **506**. The analytic code development window **504** includes a design tab **508**, which provides a user with the ability to choose a format, name, file pattern, schema, header, and/or field separator. Signal Hub supports various input file formats including delimited, fixed width, JDBX, xml, excel, log file, etc. A user can load data from various data sources. More specifically, parameterized definitions allow a user to load data from a laptop, cluster, and/or client database system. The supplementary display portion **506** includes a YAML tab **510**, a Schema tab **512**, and a dependencies tab **514**. The YAML tab **510** includes a synchronized editor so that a user can develop the code in a graphical way or in a plain text format, where these two formats are easily synchronized.

FIG. 12 is a screenshot illustrating profiling of raw data using the Workbench **500** generated by the system. The analytic code development window **504** includes a design tab **508**, a run tab **520**, and a results tab **522**. The design tab **508** is activated, and within the design tab **508** are a plurality of other tabs. More specifically, the design tab **508** includes a transformations tab **524**, a measures tab **526**, a models tab **528**, a persistence tab **530**, a meta tab **532**, and a graphs tab **534**. The measures tab **526** is activated, thereby allowing a user to add a measure from a profiling library, such as from a drop down menu. The profiling library offers data profiling tools to help a user understand the data. For example, profiling measures could include basicStats, contingency Table, edd (Enhanced Data Dictionary), group, histogram, monotonic, percentiles, woe, etc. The edd is a data profiling capability which analyzes content of data sources.

FIG. 13 is a screenshot illustrating displaying of specific entries within raw data using the Workbench **500** generated by the system. The analytic code development window **504** includes a table **540** showing specific data entries for the measure “edd”, as well as a plurality of columns pertaining to various types of information for each data entry. More specifically, the table **540** includes columns directed to obs, name, type, nmiss, pctMissing, unique, stdDev, mean_or_top1, min_or_top2, etc. The table **540** includes detailed data statistics including number of records, missing rate, unique values, percentile distribution, etc.

FIG. 14 is a screenshot illustrating aggregating and cleaning of raw data using the Workbench **500** generated by the system. As shown, the analytic code development window **504** has the transformations tab **524** activated. The transformation tab **524** is directed to the transformation library which allows users to do various data aggregation and cleaning work before using data. In the transformations tab **524**, the user can add one or more transformations, such as cubePercentile, dedup, derive, filter, group, join, limitRows, logRows, lookup, etc. **FIG. 15** is a screenshot illustrating managing and confirmation of raw data quality using the Workbench **500** generated by the system. As shown, the analytic code development window **504** has the transformations tab **524**

activated. A user can gather more information about each transformation, such as shown for Data Quality. The data quality management uses a series of checks which contains a predicate, an action, and an optional list of fields to control and manage the data quality.

FIG. 16 is a screenshot illustrating auto-generated visualization of a data model created using the Workbench **500**. This visualization could be automatically generated from YAML code (e.g., the code that reads and does initial linking and joining of data). As shown, analytic code development window **504** allows a user to view relations and interactions between various data elements. The data model organizes data elements into fact and dimension tables and standardizes how the data elements relate to one another. This could be automatically generated in Signal Hub after loading the data. **FIG. 17A** is a screenshot illustrating creation of reusable analytic code using the Workbench **500** generated by the system. As shown, the analytic code development window **504** includes many lines of code that incorporate and utilize the raw data previously selected and prepared. The Signal API could be scalable and easy to use (e.g., for loop signals, peer comparison signals, etc.). Further, Signal Hub could provide signal management by using @tag and @doc to specify signal metadata and description, which can be automatically extracted and displayed in the Knowledge Center. **FIG. 17B** is a screenshot illustrating the graphical user interface of Signal API in Workbench. Similar to excel, users can select from a function list **524** and a column list **526** to create new signals with a description **528** and example code provided at the bottom. Users can use Signal API either in a plain text format or in a graphical way, where these two formats are easily synchronized.

FIGS. 18-23 are screenshots illustrating user interface screens generated by the system using the Knowledge Center **600** to find and use a signal. As an integral part of Signal Hub, the Knowledge Center could be used as an interactive signal management system to enable model developers and business users to easily find, understand, and reuse signals that already exist in the signal library inside Signal Hub. The Knowledge Center allows for the intelligence (e.g., signals) to be accessed and explored across use cases and teams throughout the enterprise. Whenever a new use case needs to be implemented, the Knowledge Center enables relevant signals to be reused so that their intrinsic value naturally flows toward the making of a new analytic solution that drives business value.

Multiple features of the Knowledge Center facilitate accessing and consuming intelligence. The first is its filtering and searching capabilities. When signals are created, they are tagged based on metadata and organized around a taxonomy. The Knowledge Center empowers business users to explore the signals through multiple filtering and searching mechanisms.

Key components of the metadata in each signal include the business description, which explains what the signal is (e.g., number of times a customer sat in the middle seat on a long-haul flight in the past three years). Another key component of the metadata in each signal is the taxonomy, which shows each signal's classification based on its subject, object, relationship, time window, and business attributes (e.g., subject = customer, object = flight, relationship = count, time window = single period, and business attributes = long haul and middle seat).

The Knowledge Center facilitates exploring and identifying signals based on this metadata when executing use cases by using filtering and free-text searching. The Knowledge Center also allows for a complete visualization of all the elements involved in the analytical solution. Users can visualize how data sources connect to models through a variety of descriptive signals, which are grouped into Signal Sets depending on a pre-specified and domain-driven taxonomy. The same interface also allows users to drill into specific signals. Visualization tools can also allow a user to visualize end-to-end analytics solution components from the data, to the signal and finally to the use-cases. The system can automatically detect the high level lineage between the data, signal and use-cases when hovering over specific items. The system can also allow a user to further drill down specific data, signal and use-cases by predefined metadata which can also allow a user to view the high level lineage as well.

FIG. 18 is a screenshot illustrating a user interface screen generated by the system for visualizing signal paths using the Knowledge Center **600** generated by the system. As shown, the Signal Hub platform **600** includes a side menu **602** which allows a user to filter signals, such as by entering a search description into a search bar, or by browsing through various categories (e.g., business attribute, window, subject, object, relationship, category, etc.). The Signal Hub platform **600** further includes a main view portion **604**. The main view portion **604** diagrammatically displays data sources **606** (e.g., business inputs), descriptive signals **608** (e.g., grouped and organized by metadata), and predictive signals **610**. The descriptive signals **608** includes a wheel of tabs indicating categories to browse in searching for a particular signal. For example, the categories could include route, flight, hotel, etc. Once a particular category is selected in the descriptive signals **608**, the center of the descriptive signals **608** displays information about that particular category. For example, when "route" is chosen, the system indicates to the user that there are 23 related terms, 4 signal sets, and 536 signals.

The Signal Hub platform **600** also displays all the data sources that are fed into the signals of the category chosen. For example, for the "route" category, the data sources include event mater, customer, clickthrough, hierarchy, car destination, ticket coupon, non-flight delivery item, booking master, holiday hotel destination, customer, ancillary master, customer membership, ref table: station pair, table: city word cloud, web session level, ref table: city info, ref table: country code, web master, redemption flight items, email notification, gold guest list, table: station pair info, customer account tcns, service recovery master, etc. A user can then choose one or more of these data sources to further filter the signals (and/or to navigate to those data sources for additional information).

The Signal Hub platform **600** also displays all the models that utilize the signals of the category chosen. For example, for the "route" category, the predictive signals within that category include hotel propensity, destination propensity, pay-for-seat propensity, upgrade propensity, etc. A user can then choose one or more of these predictive signals.

FIG. 19 is a screenshot illustrating a user interface screen generated by the system for visualizing a particular signal using the Knowledge Center **600** generated by the system. As shown, the particular descriptive signal "bkg_avg_mis_gh_re_v_1y_per_dest" at an individual level, the data sources **606** that feed into that signal include "ancillary master," "booking master," and "ref table: station pair," and the predictive signals that use that descriptive signal include "hotel propensity," "pay-for-seat-propensity," and "destination propensity."

FIG. 20A is a screenshot illustrating a user interface screen generated by the system for finding a signal using the knowledge center **600** generated by the system. The main view portion **604** includes a signal table listing all existing signals with summary information (e.g., loaded 100 of 2851 signals) for browsing signals and their related information. The table includes the signal name, signal description, signal tags, signal set, signal type (e.g., Common:Real, Common:Long, etc.), and function. The signal description is an easy to understand business description (e.g., average number of passengers per trip customer travelled with). A user could also conduct a free text search to identify a signal description that contains a specific word (e.g., hotel signals). Further, a metadata filter could identify signals that fit within certain metadata criteria (e.g., signals that calculate an average). **FIG. 20B** is a screenshot illustrating a user interface screen generated by the system for finding a signal using the knowledge center **600** generated by the system. Users are first asked to select a pre-defined signal subject from "Search Signal" dropdown list to start the signal search process. The main view portion **604** includes a signal table listing all existing signals with summary information (e.g., filtered conditions applied; loaded 100 of 2851 signals) for browsing signals and their related information. The table includes the signal signal description, signal type (e.g., Real, Long, etc.), update time, refresh frequency, etc. The signal description is an easy to understand business description (e.g., average number of passengers per trip customer travelled with). A user could also define search columns (e.g., description) and conduct a free text search within the search columns that contains a specific word (e.g., hotel signals). Further, a metadata filter could identify signals that fit within certain metadata criteria as shown in the left side panel (e.g., signals that calculate an average).

FIG. 21A is a screenshot illustrating a user interface screen generated by the system for selecting entries (e.g., customers) with particular signal values using the Knowledge Center **600** generated by the system. Users are also able to apply business rules to signals to filter the data and target subsections of the population. For example, the user may want to identify all customers with a propensity to churn that is greater than 0.7 and those who have had two or more friends churn in the last two weeks. This is particularly important as it enables business users to build sophisticated prescriptive models allowing true democratization of big data analytics across the enterprise. More specifically, a user can select signals to limit the table to only signals necessary to execute the specific use case (e.g., Signal: "cmcnt_trp_oper_led_abdn"). The table **618** also provides for the ability to apply rules to filter the table to include only data that fits within the thresholds (e.g., customers with a hotel propensity score > 0.3). For example, the table **618** includes the columns "matched_party_id" **620**, "cmcnt_trp_oper_led_abdn" **622**, "cmbin_sum_seg_tvl_rev_p1y" **624**, "cmavg_mins_dly_p3m" **626**, "SILENT_ATTRITION" **628**. A user can narrow the search for a signal by indicating requirements for each column. For example, a user can request to see all signals that have a cmbin_sum_seg_tvl_rev_p1y of ="g. 5000-10000" and a cmavg_mins_dly_p3mof >5. A user can also apply more complex transformation on top the signals with standard SQL query language. Further, as shown in **FIG. 21B**, the Signal Hub platform **600** can schedule the business report at regular basis (e.g., daily, weekly, monthly, ect) using a reporting tool **630** to gain recurring insights or export the filtered data to external systems (e.g., CSV file into client's campaign execution engine). The system of the present disclosure can also include a reporting tool implemented in a Hadoop environment. The user can generate a report and query various reports. Further, the user can query a single signal table and view the result in real-time. Still further, the reporting tool can include a query code and a data table fully listed out in the same page so users are able to switch between different steps easily and view the result for previous step.

FIG. 21C is a screenshot illustrating a user interface screen generated by the system for displaying dashboard created using the Knowledge Center **600** generated by the system. A user is able to create various type of graphs (e.g. line chart, pie chart, scattered 3D chart, heat map, etc) in the Knowledge Center and populate dashboard with graphs created in certain layout. Dashboard will get refreshed automatically as the backend data get refreshed. A user can also export the dashboard to external system. **FIG. 21D** is a screenshot illustrating a user interface screen generated by the system for exploring data dictionary created using the Knowledge Center **600** generated by the system. A user is able to learn all the data input tables used the solution, with name, description, metadata, columns, and refresh rate information for each data input table. A user can also further explore individual data input table and learn the meaning of each column in the table. The Signal Hub platform collects and centralizes all the siloed (stored) data knowledge together via data dictionary and makes it accessible and reusable for all the users. **FIG. 21E** is a screenshot illustrating a user interface screen generated by the system for exploring models created using the Knowledge Center **600** generated by the system. A user is able to learn all the models created in the solution and explore individual model in depth. The Signal Hub platform can display model description, metadata, input signal, output column, etc. all in one centralized page for each model. **FIG. 21E** also illustrates a user interface screen generated by the system for commenting signals using the Knowledge Center **600** generated by the system. Users can comment on a signal via Knowledge Center user interface directly to express interest on a signal, propose potential use case for the signal, or validate the signal value. The Signal Hub platform allows users to interact with each other and exchange ideas. **FIG. 21F** is a screenshot generated by the system which illustrates the charts that could be generated by the system. The charts could be a representation of a signal or multiple signals. The types of charts could include, but is not limited to, bar charts, line charts, density charts, pie charts, bar graphs, or any other chart known to those of ordinary skill in the art. Further, as shown, multiple charts could be included in the dashboard for comparing and viewing different charts simultaneously.

FIG. 22 is a screenshot illustrating a user interface screen generated by the system for visualizing signal parts of a signal using the Knowledge Center **600** generated by the system. Shown is a table showing various signals of a signal set. Users can isolate exactly which columns in the raw data or other signals were combined to create the signal of interest. The Signal Hub platform **600** can display the top level diagram **650**, the definition level diagram **652**, the predecessors **654**, raw data **656**, consumers **658**, definition **660**, schema **62**, and metadata **664** and stats. The predecessors tab is used to understand the raw data columns and signals that are used to create a specific signal (e.g., txh_mst_rx_cnt_txn_onl) and can be used to track the detailed signal calculation step by step. When the predecessors tab is selected the resulting table can have one or more columns. For example, the table could include a column **670** of names of the signals within the signal set (e.g., within signal set "signals.signals_pos_txn_mst_04_app"), as well as the formula **672**, and what the signal is defined in **674**.

FIG. 23A is a screenshot illustrating a user interface screen generated by the system for visualizing a lineage of a signal using the Knowledge Center **600** generated by the system. The lineage is used to understand the transformation from raw data to descriptive signals and predictive signals (e.g., how is the number of trips required to move to the next loyalty tier signal generated and which models consume it). As shown, when the definition level diagram button **652** is activated, the Signal Hub platform **600** displays the lineage of a particular signal, which includes what data is being pulled, and what models the signal is being used in. Once a signal of interest is identified, users can gain a deeper understanding of the signal by exploring its lineage from the raw data through all transformations, providing insight into how a particular Signal was created and what the value truly represents. They can identify which signals, if any, consume the signal of interest and view the code that was used to define it. **FIG. 23B** is a screenshot illustrating a user interface screen generated by the system for displaying signal values stats and visualization of signal value distribution. Both features provide a better understanding of signals, helps scientists determine what codes need to be evoked in the production system to calculate the signal, and makes signal management easier and faster. The Knowledge Center contains visualization capabilities to allow users to explore the values of signals directly in the Signal Hub platform **600**.

FIGS. 24-29 are screenshots illustrating using the Workbench **700** generated by the system to create predictive signals (models) with Analytic Wizard module. Analytic Wizard streamlines model development process with predefined steps and parameter presets. More specifically, **FIG. 24A** is a screenshot illustrating preparation of data to train a model using the Workbench **700** generated by the system. As shown, the Workbench **700** includes a tree view **702**, and an analytic code development window **704** which includes a design tab **708**, run tab **710**, and results tab **712**. The design tab **708** is activated, and within the design tab **708** are a plurality of other tabs. More specifically, the design tab **708** includes a transformations tab **714**, a measures tab **716**, a models tab **718**, a persistence tab **720**, a meta tab **722**, and a graphs tab **724**. Signal Hub offers several ways to split train and test data for model development purposes. The supplementary display portion **706** includes a YAML tab **726**, a schema tab **728**, and a dependencies tab **730**. Signal Hub performs missing value imputation, normalization, and other necessary signal treatment before training the model, as shown in the supplementary display portion **706**. Once a model has been selected, more information regarding the model is easily accessible, such as the description and model path. A user can also train an external model using any desired analytic tool. As long as the model output conforms to a standard pmml format, the Signal Hub platform can incorporate the model result and do the scoring later. **FIG. 24B** is a screenshot illustrating an alternative embodiment as to how users can select from a variety of model algorithms (e.g., logistic regression, deep autoencoder, etc.). As shown, the Workbench **700** can include a tab **703** for displaying a variety of signals. The Workbench **700** can include a selection means **732** for selecting a model algorithm. The selection means **732** can be a drop down menu or similar means known to those of ordinary skill in the art. **FIG. 24C** is a screenshot illustrating the different parameter experiments users can apply during the model training process. Signal Hub also allows user to configure execution of models with parameter pre-sets that optimize speed or optimize accuracy as execution steps. **FIG. 24D** is a screenshot illustrating how data preparation can be handled during the model training process. For example, missing values can be replaced with a median value. Furthermore, a normalization method can be applied to the data training. **FIG. 24E** is a screenshot illustrating how dummy variables can be introduced to facilitate the model training process. **FIG. 24F** is a screenshot illustrating the dimensional reduction that can be applied to the model training process. For example, a variance threshold can be introduced and the number of dimensions can be specified to further improve the model training accuracy. **FIG. 24G** is a screenshot illustrating the data splitting aspect of the model training process. For example, a splitting method can be chosen such as cross-fold validation or any other data splitting method known to those of ordinary skill in the art. Furthermore, the number of folds, seed, percent of validation, and the stratified field can be specified. **FIG. 24H** is a screenshot illustrating the measure tab which allows graph names to be specified along with sampling percentages. The measure tab further allows the corresponding measures to be selected**.** **FIG. 24I** is a screenshot illustrating the process tab which allows the user to create a library for the wizard output. In particular, a search path, library and comments can be inputted to the system. **FIG. 24J** is a screenshot of the result tab showing the output of the model training to the user. The foregoing steps of training a predictive model can be done over a Hadoop cluster using dataflow operations.

FIG. 25A is a screenshot illustrating training a model using the Workbench **700** generated by the system. Signal Hub could include prebuilt models that a user can train (e.g., logistic regression, deep autoencoder, etc.). As shown, the models tab **718** is selected, and a user can add one or more models, such as "binarize," "decision tree," "deepAutoencoder," "externalModel," "frequentItems," "gmm," "kmeans," "linearRegression," and "logisticRegression." A user can train an external model using any desired analytic tool. As long as the model output conforms to a standard pmml format, the Signal Hub platform can incorporate the model result and do the scoring. Under the models tab **718**, once a model has been selected, more information regarding the model is easily accessible, such as the description and model path. **FIG. 25B** is a screenshot illustrating preparation of data to train a model using the Workbench generated by the system. The Workbench **700** can include a data preparation tab **734**. Signal Hub can perform in the data preparation tab **734** missing value imputation **736**, normalization **738**, and other necessary signal treatment **740** before training the model. **FIG. 25C** is a screenshot illustrating different data splitting options provided by Workbench **700**. The Workbench **700** can include a data splitting tab **740** for allowing input of the number of folds **741**, number of seeds **742**, percent of validation **743** and stratified input **744**. **FIG. 26** is a screenshot illustrating loading an external model trained outside of the integrated development environment.

FIG. 27 is a screenshot illustrating scoring a model using the Workbench **700** generated by the system. Signal Hub prebuilt a number of model scorers that can perform end to end analytic development activities. **FIG. 28** is a screenshot illustrating monitoring model performance using the Workbench **700** generated by the system. Signal Hub offers various monitoring matrices to measure the model performance (e.g., ROC, KS, Lorenz, etc.). As shown, any of a variety of measures can be used to monitor and score the model. For example, monitoring measures could include "captureRate," "categoricalWoe," "conditionIndex," "confusionMatrix," "informatonValue," "kolmogorovSmirnov," "Lorenz," "roc," etc.

FIG. 29A is a screenshot illustrating a solution dependency diagram **750** of the Workbench **700** generated by the system. The diagram **750** illustrates various modules for each portion of the analytics development lifecycle. For example, the diagram illustrates raw data modules **760**, aggregate and cleanse data modules **762**, create descriptive signals modules **764**, select descriptive signal modules **766** (which is also the develop solutions/models module **766**), and evaluate model results modules **768**.

FIG. 29B is a screenshot illustrating a collaborative analytic solution development using the Workbench generated by the system. The system of the present disclosure allows users to collaborate on large software projects for code development. In addition to code development, developers can also develop and collaborate on data assets. Besides stand-alone development mode, Signal Hub Workbench can also be connected with version control system (eg: SVN, etc) in the backend to support collaborative development. Users can create individual workspace and submit changes from Workbench user interface directly. Centralized Workbench also enables users to learn the different activity streams happening in the solution. Files that are being worked on by other developers would show up as locked by the system automatically to avoid conflicts. Locked files will become unlocked after developer submitting the changes or a solution manager forces to break the lock and all the developers would get a workspace update notification automatically. The system of the present disclosure can implement isolation requirements to further facilitate collaboration. For example, the system can isolate upstream code and data changes. If a developer is reading the results of a view or signal set, she expects them not to change without her knowledge. If a change has been made to the view, either because the underlying data has changed or because the code has changed, it should not automatically affect her work until she decides to integrate the updates into her work stream. Additionally, the system can protect a developer's code and data from the other developers' activities. Further, the system can also allow a user to decide when to make their work public. A user has the ability to develop new code without worrying about affecting the work of those downstream. When the work is completed, the user can then "release" her version of the code and data. Users will see this released version and chose whether they'd like to upgrade their view to read it.

The system can further facilitate collaboration by allowing a single library to be developed by a single developer at one point in time which will reduce code merging issues. Furthermore, the system can use source control to make code modifications. A user can update when she wants to receive changes from her team members, and commit when she wants them to be able to see other developers' changes. Each developer at a point in time can be responsible for specific views and their data assets. The owner of the view can be responsible for creating new versions of their data while other developers can only read the data that has been made public to them. Ownership can change between developers or even to a common shared user. A dedicated workspace can be created in the shared cluster which can be read-only for other developers and only the owner of the workspace can write and update data. When new code and data is developed, the developer can commit the changes to the source control and publish the new data in the cluster to the other developers. This allows the other developers to see the code changes and determine if they would like to integrate it with their current work.

FIG. 29C is a screenshot of a common environment file that contains code and library output paths to grant every developer access to the code and data of every developer, regardless of where the data resides. The definitions in the file can be referenced with a qualified name instead of a filepath. This allows an easy move from one workspace to another without changing the code by making a small change to the common environment file. **FIG. 29D** is a screenshot of a separate personal environment file for a user working on a subset of a project. The file begins by inheriting the common project environment file “env_project.yaml.” Thus, all of the parameters set in the general environment file will also apply if you run with the personal file, such as “env_myusername.yaml.” Any parameters that are also defined in the personal file, in this case “etlVersion” will be over-ridden. So if the workflows are run with “env_project.yaml,” the “etlVersion” will be 1.4. If the workflows are run with “env_myusername.yaml,” then “etlVersion” will be 1.5. With either environment file, “importVersion” will be 1.1. **FIGS. 29E-29I** are screenshots of environment files having multiple output paths. The system can also allow users to have multiple output paths for the data views using the “libraryOutputPaths” parameter in the environment file. These paths can be specified as a map between a library and a file path in which to place the data of that library. For a shared Hadoop cluster, the file path can point to a folder on HDFS. The default data location can still be decided using the “dataOutputPath” if the library is not mapped to any new location. Using this map, each library can be assigned to a unique data location. The project owner can therefore, map each library to a directory that is owned by a given developer. This can further allow data view abstraction modes for maintaining fast incremental data updates without underlying filesystem support for the data update

FIG. 29J is a screenshot of code for data versioning. Data versioning is the ability to store different generations of data, and allow other collaborators to decide which version to use. To achieve this, users can version their data using the label properties of views. There are two ways of doing this: one in the view itself, the other in the common environment file. The code is shown in **FIG. 29J**. Every time the view is executed, the version of the view data can be determined by the label. If a new label is used, a new folder can be created with a new version of the view's data. The granularity of this versioning is up the user; she can choose to assign the version number to just one view or to some subset, depending on the needs of the project. Every time the user wants to publish to her team members a new version of “myView, the user can increment “myView_LatestVersion” in the common environment file. This change can indicate either a code update or a data update. Additionally, the user may add a comment to the environment file giving information about this latest version, including when it was updated, what the changes were, etc. The user can then commit the common environment file with the rest of her code changes. With this information, users of the view further downstream can choose whether they'd like to upgrade to the latest version or continue using an earlier version. If the downstream users would always get the latest version, they can use the same variable "myView_LatestVersion" in the label parameter of the "readView" for "myView." Since they share the same common environment file, the latest value will be used when a user updates her code from system. If the user wants to stay with an existing version, the user can override the version in their private environment file to a specific version. Once a version is "released," the permissions on that directory can be changed to make it non-writable even for the developer herself, so that it is not accidentally overwritten. This can allow users to set different version numbers and "libraryOutputPaths." For example, the project-level environment file (the one users are using by default) can have the latest release version for a given view. The user developing it can have a private environment file with a later version. The user can do this by including the same version parameter in her file and running the view with her private environment file. This can allow the user to develop new versions while others are reading the older stable version.

In most cases, users can "own" a piece of code, either independently or as a team. They can be responsible for updating and testing the code, upgrading the inputs to their code, and releasing versions to be consumed by other users downstream. Thus, if the team maintaining a given set of code needs an input upgraded, they can contact the team responsible for that code and request the relevant changes and new release. If the team upstream is not able to help, the user can change the "libraryOutputPaths" for the necessary code to a directory in which they have permissions. It involves no code changes past the small change in the environment file. If the upstream team is able to help, they can make the release. This allows collaboration with minimum disruption.

FIGS. 30-32 are screenshots illustrating the Signal Hub manager **800** generated by the system to manage user access to overall Signal Hub platform and analytic operation process.. The Signal Hub manager **800** provides a management and monitoring console for analytic operational stewards (e.g., IT, business, science, etc.). The Signal Hub manager **800** facilitates understanding and managing the production quality and computing resources.

FIG. 30A is a screenshot of the Signal Hub manager **800** generated by the system. The Signal Hub manager **800** facilitates easy viewing and management of signals, signal sets, and models. The management console allows for the creation of custom dashboards and charting, and the ability to drill into real time data and real time charting for a continuous process. As shown, the Signal Hub manager **800** includes a diagram view. In this view, the Signal Hub manager **800** could include a data flow diagram **802** showing the general data flow of raw data to signals to models. Further, the Signal Hub manager **800** could include a chart area **804** providing a variety of information about the data, signals, signal sets, and models. For example, the chart area **804** could provide one or more tabs related to performance, invocation history, data result, and configuration. The data result tab could include information such as data, data quality, measure, PMML, and graphs. The Signal Hub manager **800** could also include additional information as illustrated in window **806**, such as performance charts and heat maps. The chart area allows a user to drill down on every workflow to easily understand the processing of all views involved in the execution of a use case.

FIG. 30B is a screenshot for user access management of the Signal Hub manager **800** generated by the system. The Signal Hub manager **800** provides role-based access control for all Signal Hub platform components to increase network security in an efficient and reliable way. As shown, users are assigned to different groups and different groups are authorized with different permissions including admin, access, operate, develop and email. Besides global permission management, Signal Hub platform also allows admin user to manage authentication and authorization on solution basis.

FIG. 30C is a screenshot for overall Signal Hub platform usage tracking of the Signal Hub manager **800** generated by the system. As shown, a user is able to download the usage report from Signal Hub manager user interface to track how other user are using different Signal Hub platform components by detailed event (e.g. login, entering Knowledge Center, create a report, create a dashboard, etc.) and conduct further analysis on top of it.

FIG. 31A-B are screenshots for alerts system of the Signal Hub manager **800** generated by the system. Based on monitor system stats, a user can set up alerts at different level including system level alert, workflow level alert and view level alert. Signal Hub platform also allows user to set up different types of alert (eg: resource usage, execution time, signal value drift, etc), define threshold and trigger recovery behaviors (eg: email notification, fail job, roll back job) automatically. The alert feature enables users to better track solution status from both operational and analytic perspectives and greatly improves solution operation efficiency. **FIG. 31A** is another screenshot of the Signal Hub manager **800** generated by the system. The Signal Hub manager **800** includes a table view. In this view, the Signal Hub manager includes a data flow table of information regarding the general data flow of raw data to signals to models. The data flow table includes view name, label, status, last run, invocation number (e.g., success number, failure number), data quality (e.g., treated number, rejected number), timestamp of last failure, current wait time, average wait time, average rows per second, average time to completion, update (e.g., input record number, output record number), historical (input record number, output record number), etc. Similar to the diagram view discussed above, the table view could also include a chart area. For example, the chart area **804** could provide one or more tabs related to performance, invocation history, data result, and configuration. The invocation history tab could include invocation, status, result, elapsed time, wait time, rows per second, time to completion, update (e.g., input record number, output record number), and historical (e.g., input record number, output record number). **FIG. 31B** is a screenshot illustrating overall Signal Hub platform usage tracking of the Signal Hub manager **800** and alert functionality generated by the system. As shown, a user is able to download the usage report from Signal Hub manager user interface to track how other user are using different Signal Hub platform components by detailed event (e.g. login, entering Knowledge Center, create a report, create a dashboard, etc.) and conduct further analysis on top of it.

FIG. 32 is another screenshot of the Signal Hub manager **800** generated by the system. More specifically, shown is the monitor system of the Signal Hub manager **800**. This facilitates easy monitoring of all analytic processes from a single dashboard. The current activities window **810** has a table which includes solution names, workflow names, status, last run, success number, failure number, timestamp of last failure, and average elapsed time. The top storage consumers window **812** has a table which includes solution names, views, volume, last read, last write, number of variants, number of labels. The top run time consumers window **814** has a table which includes solution names, views, run time, number parallel, elapsed time, requested memory, and number of containers. A user is also able to drill down to a specific solution, workflow, or view to learn about their operational status.

FIG. 33 is a diagram showing hardware and software components of the system **100**. The system **100** comprises a processing server **102** which could include a storage device **104**, a network interface **108**, a communications bus **110**, a central processing unit (CPU) (microprocessor) **112**, a random access memory (RAM) **114**, and one or more input devices **116**, such as a keyboard, mouse, etc. The server **102** could also include a display (e.g., liquid crystal display (LCD), cathode ray tube (CRT), etc.). The storage device **104** could comprise any suitable, computer-readable storage medium such as disk, non-volatile memory (e.g., read-only memory (ROM), eraseable programmable ROM (EPROM), electrically-eraseable programmable ROM (EEPROM), flash memory, field-programmable gate array (FPGA), etc.). The server **102** could be a networked computer system, a personal computer, a smart phone, tablet computer etc. It is noted that the server **102** need not be a networked server, and indeed, could be a stand-alone computer system.

The functionality provided by the present disclosure could be provided by a Signal Hub program/engine **106**, which could be embodied as computer-readable program code stored on the storage device **104** and executed by the CPU **112** using any suitable, high or low level computing language, such as Python, Java, C, C++, C#, .NET, MATLAB, etc. The network interface **108** could include an Ethernet network interface device, a wireless network interface device, or any other suitable device which permits the server **102** to communicate via the network. The CPU **112** could include any suitable single- or multiple-core microprocessor of any suitable architecture that is capable of implementing and running the signal hub program **106** (e.g., Intel processor). The random access memory **114** could include any suitable, high-speed, random access memory typical of most modern computers, such as dynamic RAM (DRAM), etc.

Having thus described the system and method in detail, it is to be understood that the foregoing description is not intended to limit the spirit or scope thereof. It will be understood that the embodiments of the present disclosure described herein are merely exemplary and that a person skilled in the art may make any variations and modification without departing from the spirit and scope of the disclosure. All such variations and modifications, including those discussed above, are intended to be included within the scope of the disclosure.

CLAIMS

What is claimed is:

1. A system for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis comprising:

a computer system having stored thereon and executing a signal hub engine;

a graphical, interactive workbench displayed on a display of the computer system and accessible by a user of the computer system;

a plurality of data sources in communication with the signal hub engine of the computer system, the signal hub engine integrating each of the plurality of data sources for access within the interactive workbench;

an analytic code development window displayed in the interactive workbench, the analytic code development window providing access to the user to a plurality of modular analytic code libraries available on the system, wherein the user can rapidly write customized analytic code in the code development window using one or more of the plurality of modular analytic code libraries;

an execution environment within the interactive workbench, the execution environment allowing the user to execute the customized analytic code in order to generate a desired signal from at least one of the plurality of data sources;

a signal management tool for automatically extracting metadata relating to the desired signal and allowing the user to retrieve a value of the desired signal and develop an interactive report relating to the value of the desired signal;

wherein the customized analytic code can be identified by a label, stored, and then rapidly retrieved and executed for subsequent data modeling and analysis using the label; and

wherein the execution environment can automatically monitor the desired signal and can take a recovery action for the customized analytic code based on a predefined threshold.

2. The system of Claim 1, wherein the signal hub engine is implemented in a Hadoop environment to allow data view abstraction modes for maintaining fast incremental data updates without underlying filesystem support for the data updates.

3. The system of Claim 1, further comprising the step of implementing the analytic code as a layered model, where a reusable signal layer resides between raw data inputs and

use cases, which allows the system to process multiple use cases simultaneously.

4. The system of Claim 1, wherein the desired signal comprises of descriptive signals and predictive signals.

5. The system of Claim 1, wherein the signal management tool can generate a dependency diagram for tracking a data element as it is transformed in the analytic life cycle.

6. The system of Claim 1, wherein the signal management tool can categorize a plurality of signals based on taxonomies and allowing the user to search for the plurality of signals based on the taxonomies.

7. The system of Claim 1, further comprising a multi target system data flow compiler that can generate code to deploy on a plurality of target data flow engines utilizing different computer environments, languages, and frameworks.

8. The system of Claim 1, wherein the signal hub engine automatically detects changes from the plurality of data sources and triggers a chain of processing jobs to update the desired signal with relevant data changes without transactional system support.

9. The system of Claim 1, wherein a predictive signal or a model algorithm is trained at scale with predefined model development steps and parameter pre-sets over a Hadoop cluster using dataflow operations.

10. The system of Claim 1, wherein the desired signal can be created by reusing at least one other previously created signal and analytic code.

11. The system of Claim 1, wherein a descriptive signal can be extracted from a pattern occurrence based on an occurrence of a specific event sequence over a time period with an event pattern matcher algorithm.

12. The system of Claim 1, wherein the graphical interactive workbench facilitates a collaborative development environment.

13. The system of Claim 12, wherein the graphical interactive workbench supports data versioning by using data label features and a plurality of configuration files to allow the user to publish and use the latest version of the analytic code.

14. The system of Claim 12, wherein the graphical interactive workbench facilitates a collaborative environment by seamless integration with a code version control tool and by isolating the user's workspace from previous versions of the analytic code so that the user does not encounter interruptions from new versions of the analytic code.

15. The system of Claim 12, wherein the graphical interactive workbench facilitates a collaborative development environment by having user role and task management assignments.

16. The system of Claim 1 wherein the signal management tool allows visualization of a data element, a plurality of signals, and a plurality of use-cases.

17. The system of Claim 16 wherein the signal management tool allows the user to detect higher levels of lineage between the data element, the plurality of signals, and the plurality of use-cases.

18. The system of Claim 16 wherein the signal management tool allows the user to view lower levels of lineage between the data element, the plurality of signals, and the plurality of use-cases.

19. The system of Claim 1, wherein the signal management tool comprises of a reporting tool in a Hadoop environment for allowing the user to define a query for a single signal table and view a result in real-time.

20. The system of Claim 19, wherein the reporting tool includes a query code and a data table listed in the same page so users are able to switch between different steps and view the result for the previous step.

21. A computer-implemented method for rapid development and deployment of reusable analytic code for use in computerized data modeling and analysis comprising:

providing a computer system having stored thereon and executing a signal hub engine;

providing a graphical, interactive workbench displayed on a display of the computer system and accessible by a user of the computer system;

providing a plurality of data sources in communication with the signal hub engine of the computer system, the signal hub engine integrating each of the plurality of data sources for access within the interactive workbench;

providing an analytic code development window displayed in the interactive workbench, the analytic code development window providing access to the user to a plurality of modular analytic code libraries available on the system, wherein the user can rapidly write customized analytic code in the code development window using one or more of the plurality of modular analytic code libraries; and

providing an execution environment within the interactive workbench, the execution environment allowing the user to execute the customized analytic code in order to generate a desired signal from at least one of the plurality of data sources,

identifying the customized analytic code by a label, storing, and then rapidly retrieving and executing for subsequent data modeling and analysis using the label,

providing a signal management tool for extracting metadata automatically relating to the desired signal and allowing the user to retrieve a value of the desired signal and develop an interactive report relating to the value of the desired signal, and

monitoring the desired signal and taking a recovery action for the customized analytic code based on a predefined threshold.

22. The method of Claim 21, further comprising the step of implementing the signal hub engine in a Hadoop environment to allow data view abstraction modes for maintaining fast incremental data updates without underlying filesystem support for the data updates.

23. The method of Claim 21, further comprising the step of implementing the analytic code as a layered model, where a reusable signal layer resides between raw data inputs and use cases, which allows the system to process multiple use cases simultaneously.

24. The method of Claim 21, further comprising the step of providing descriptive signals and predictive signals.

25. The method of Claim 21, further comprising the step of generating a dependency diagram for tracking a data element as it is transformed in the analytic life cycle.

26. The method of Claim 21, further comprising the step of categorizing a plurality of signals based on taxonomies and allowing the user to search for the plurality of signals based on the taxonomies.

27. The method of Claim 21, further comprising the step of utilizing a multi target system data flow compiler that can generate code to deploy on a plurality of target data flow engines utilizing different computer environments, languages, and frameworks.

28. The method of Claim 21, further comprising the step of detecting changes automatically from the plurality of data sources and triggering a chain of processing jobs to update the desired signal with relevant data changes without transactional system support.

29. The method of Claim 21, further comprising the step of allowing the desired signal to be created by reusing at least one other previously created signal and analytic code.

30. The method of Claim 21, further comprising the step of extracting a descriptive signal from a pattern occurrence based on an occurrence of a specific event sequence over

a time period with an event pattern matcher algorithm.

31. The method of Claim 21, further comprising the step of training a predictive model or model algorithm at scale with predefined model development steps and parameter presets over a Hadoop cluster using dataflow operations.

32. The method of Claim 21, further comprising the step of facilitating a collaborative development environment in the graphical interactive workbench.

33. The method of Claim 32, further comprising the step of supporting data versioning by using data label features and a plurality of configuration files for allowing the user to publish and use the latest version of the analytic code.

34. The method of Claim 32, further comprising the step of facilitating a collaborative environment by seamless integration with a code version control tool and isolating the user's workspace from previous versions of the analytic code so that the user does not encounter interruptions from new versions of the analytic code.

35. The method of Claim 32, further comprising the step of facilitating a collaborative development environment by having user role and task management assignments.

36. The method of Claim 21, further comprising the step of allowing visualization of a data element, a plurality of signals, and a plurality of use-cases by using the signal management tool.

37. The method of Claim 36, further comprising the step of allowing the user to detect the higher levels of lineage between the data element, the plurality of signals, and the plurality of use-cases.

38. The method of Claim 36, further comprising the step of allowing the user to view lower levels of the data element, the plurality of signals, and the plurality of use-cases.

39. The method of Claim 21, further comprising the step of providing a reporting tool in a Hadoop environment for allowing the user to define a query for a single signal table and view a result in real-time.

40. The method of Claim 39, further comprising the step of including in the reporting tool a query code and a data table listed in the same page so users are able to switch between different steps and view the result for the previous step.

41. A non-transitory computer-readable medium having computer-readable instructions stored thereon which, when executed by a computer system, cause the computer system to perform the steps of:

providing a computer system having stored thereon and executing a signal hub engine;

providing a graphical, interactive workbench displayed on a display of the computer system and accessible by a user of the computer system;

providing a plurality of data sources in communication with the signal hub engine of the computer system, the signal hub engine integrating each of the plurality of data sources for access within the interactive workbench;

providing an analytic code development window displayed in the interactive workbench, the analytic code development window providing access to the user to a plurality of modular analytic code libraries available on the system, wherein the user can rapidly write customized analytic code in the code development window using one or more of the plurality of modular analytic code libraries; and

providing an execution environment within the interactive workbench, the execution environment allowing the user to execute the customized analytic code in order to generate a desired signal from at least one of the plurality of data sources,

identifying the customized analytic code by a label, storing, and then rapidly retrieving and executing for subsequent data modeling and analysis using the label,

providing a signal management tool for extracting metadata automatically relating to the desired signal and allowing the user to retrieve a value of the desired signal and develop an interactive report relating to the value of the desired signal, and

monitoring the desired signal and taking a recovery action for the customized analytic code based on a predefined threshold.

42. The computer-readable medium of Claim 41, further comprising the step of implementing the signal hub engine in a Hadoop environment to allow data view abstraction modes for maintaining fast incremental data updates without underlying filesystem support for the data updates.

43. The computer-readable medium of Claim 41, further comprising the step of implementing the analytic code as a layered model, where a reusable signal layer resides between raw data inputs and use cases, which allows the system to process multiple use cases simultaneously.

44. The computer-readable medium of Claim 41, further comprising the step of providing descriptive signals and predictive signals.

45. The computer-readable medium of Claim 41, further comprising the step of generating a dependency diagram for tracking a data element as it is transformed in the analytic life cycle.

46. The computer-readable medium of Claim 41, further comprising the step of categorizing a plurality of signals based on taxonomies and allowing the at least one user to search for the plurality of signals based on the taxonomies.

47. The computer-readable medium of Claim 41, further comprising the step of utilizing a multi target system data flow compiler that can generate code to deploy on a plurality of target data flow engines utilizing different computer environments, languages, and frameworks.

48. The computer-readable medium of Claim 41, further comprising the step of detecting changes automatically from the plurality of data sources and triggering a chain of processing jobs to update the desired signal with relevant data changes without transactional system support.

49. The computer-readable medium of Claim 41, further comprising the step of allowing the desired signal to be created by reusing at least one other previously created signal and analytic code.

50. The computer-readable medium of Claim 41, further comprising the step of extracting a descriptive signal from a pattern occurrence based on an occurrence of a specific event sequence over a time period with an event pattern matcher algorithm.

51. The computer-readable medium of Claim 41, further comprising the step of training a predictive model or model algorithm at scale with a predefined model development steps and parameter pre-sets over a Hadoop cluster using dataflow operations.

52. The computer-readable medium of Claim 41, further comprising the step of facilitating a collaborative development environment in the graphical interactive workbench.

53. The computer-readable medium of Claim 52, further comprising the step of supporting data versioning by using data label features and a plurality of configuration files for allowing the user to publish and use the latest version of the analytic code.

54. The computer-readable medium of Claim 52, further comprising the step of facilitating a collaborative environment by seamless integration with a code version control tool and isolating the user’s workspace from previous versions of the analytic code so that

the user does not encounter interruptions from new versions of the analytic code.

55. The computer-readable medium of Claim 52, further comprising the step of facilitating a collaborative development environment by having user role and task management assignments.

56. The computer-readable medium of Claim 41, further comprising the step of allowing visualization of a data element, a plurality of signals, and a plurality of use-cases by using the signal management tool.

57. The computer-readable medium of Claim 56, further comprising the step of allowing the user to detect the higher levels of lineage between the data element, the plurality of signals, and the plurality of use-cases.

58. The computer-readable medium of Claim 56, further comprising the step of allowing the user to view lower levels of the data element, the plurality of signals, and the plurality of use-cases.

59. The computer-readable medium of Claim 41, further comprising the step of providing a reporting tool in a Hadoop environment for allowing the user to define a query for a single signal table and view a result in real-time.

60. The computer-readable medium of Claim 59, further comprising the step of including in the reporting tool a query code and a data table listed in the same page so users are able to switch between different steps and view the result for the previous step.

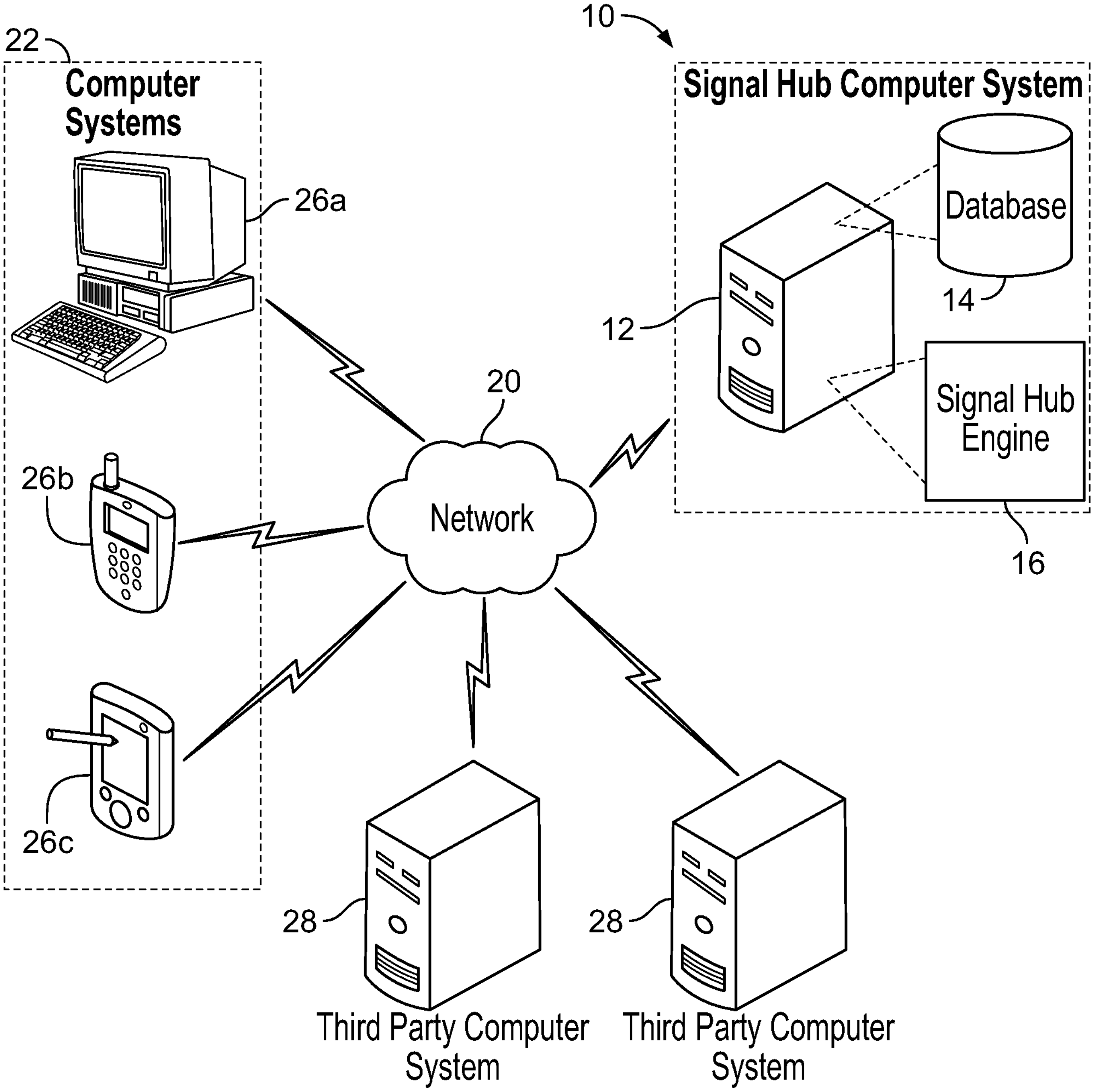

22
Computer Systems
26a
26b
26c
10
Signal Hub Computer System
Database
14
12
Signal Hub Engine
16
20
Network
28
Third Party Computer System
28
Third Party Computer System

FIG. 1

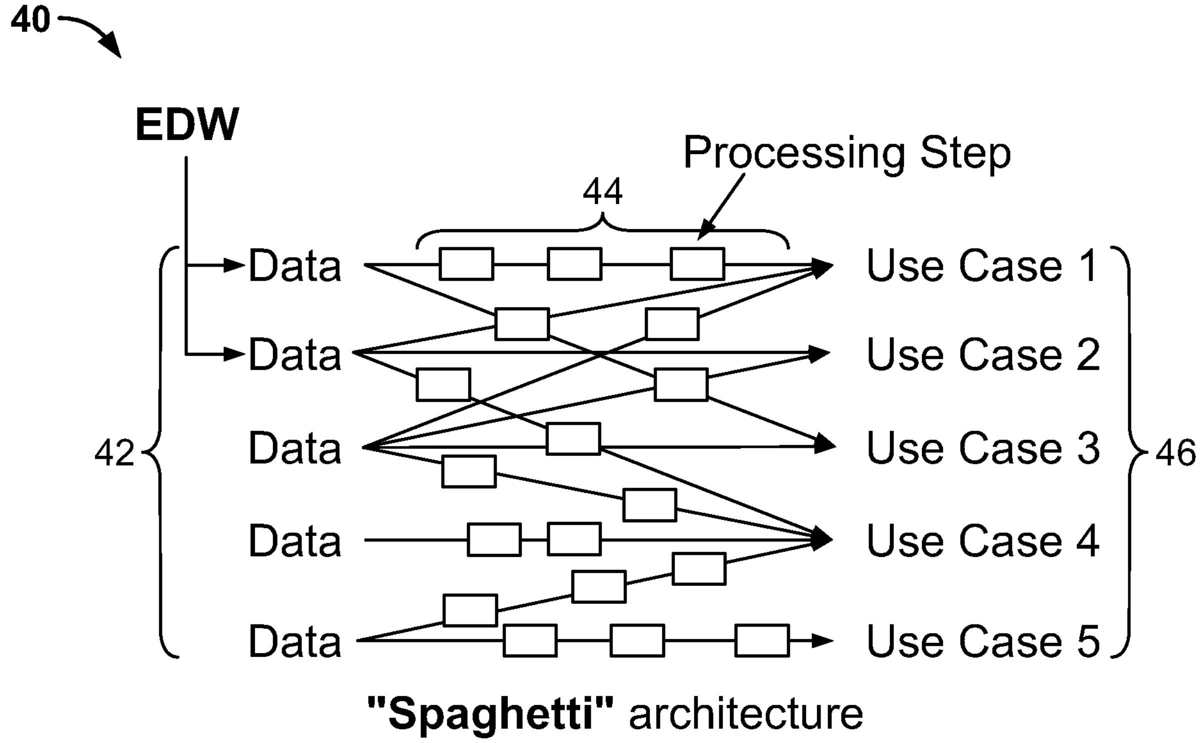

40
EDW
Processing Step
44
Data
Data
Data
Data
Data
42
Use Case 1
Use Case 2
Use Case 3
Use Case 4
Use Case 5
46
"Spaghetti" architecture

FIG. 2

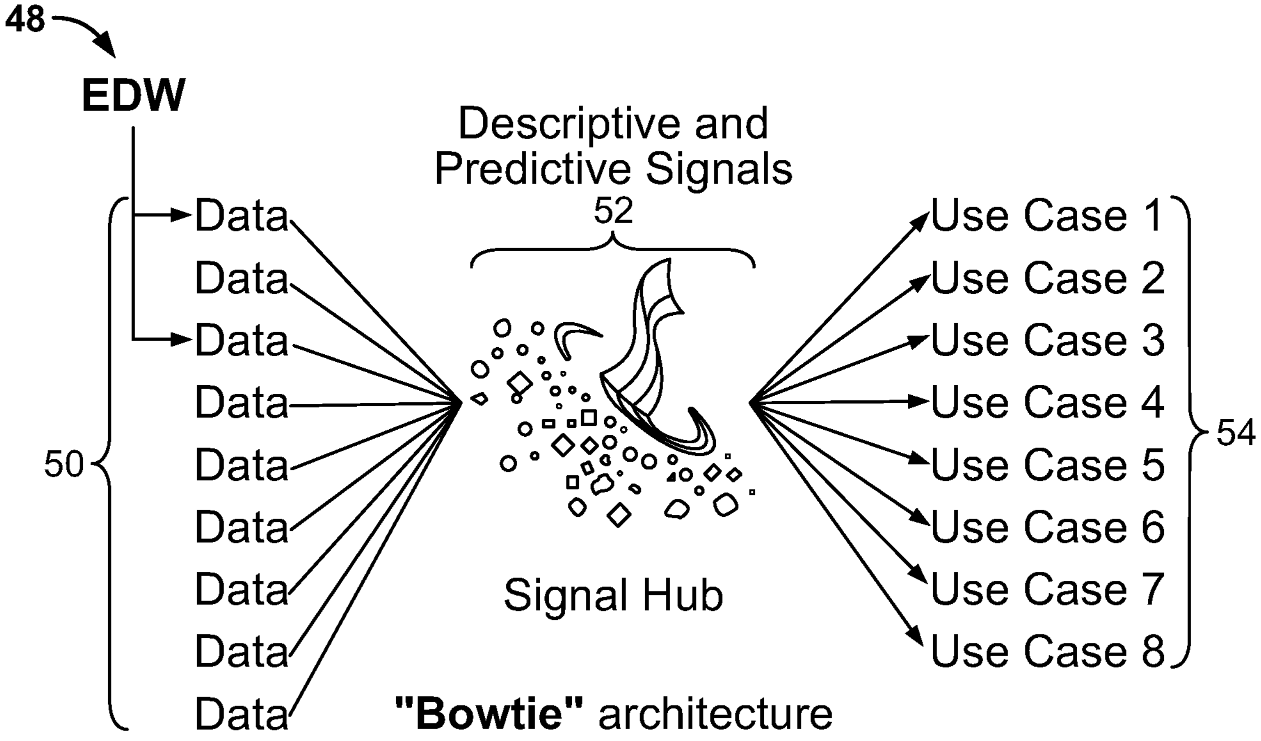

48
EDW
Descriptive and Predictive Signals
52
Data
Data
Data
Data
Data
Data
Data
Data
Data
50
Signal Hub
Use Case 1
Use Case 2
Use Case 3
Use Case 4
Use Case 5
Use Case 6
Use Case 7
Use Case 8
54
"Bowtie" architecture

FIG. 3

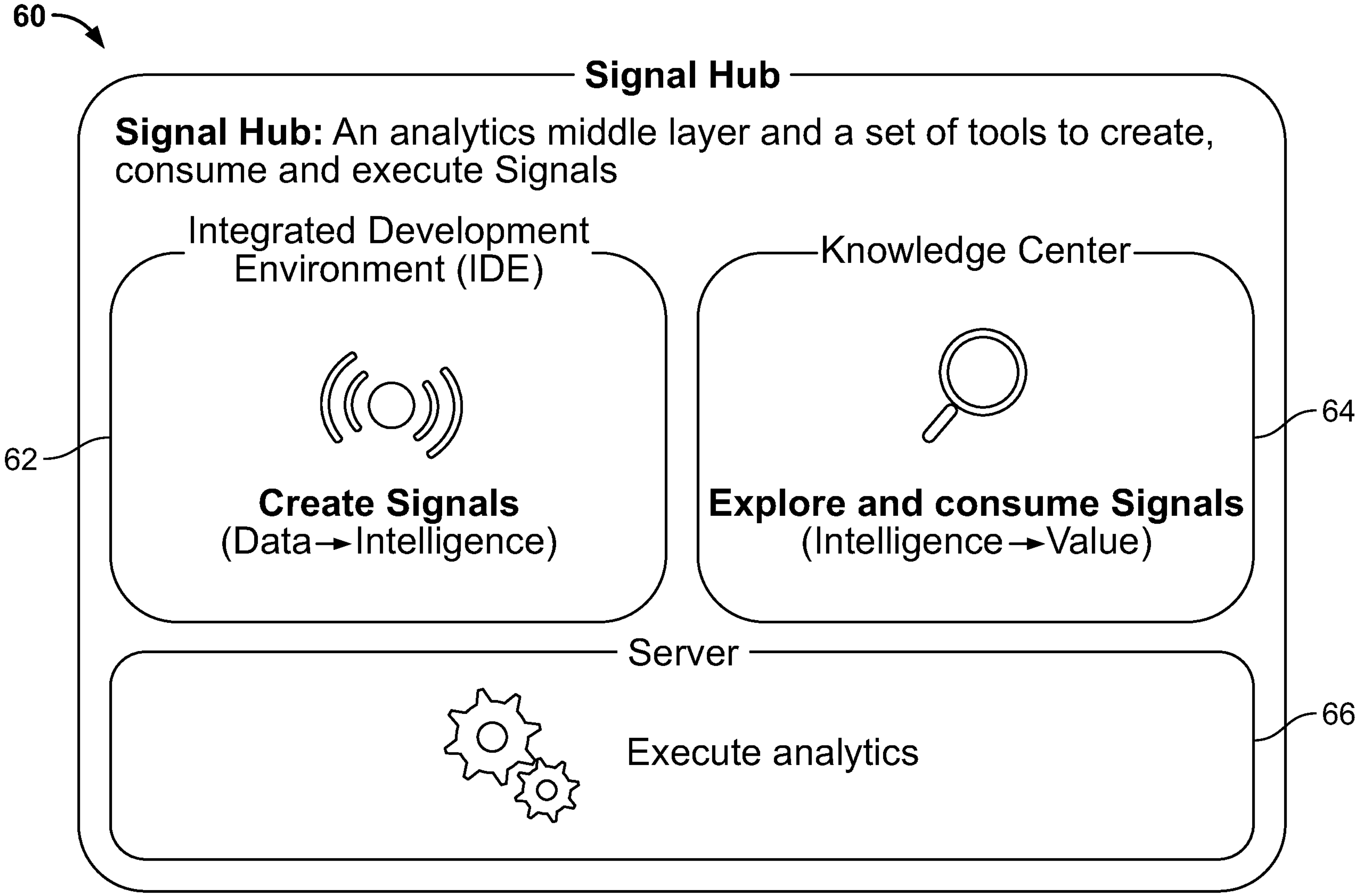

60
Signal Hub
Signal Hub: An analytics middle layer and a set of tools to create, consume and execute Signals
Integrated Development Environment (IDE)
Create Signals
(Data→Intelligence)
62
Knowledge Center
Explore and consume Signals
(Intelligence→Value)
64
Server
Execute analytics
66

FIG. 4A

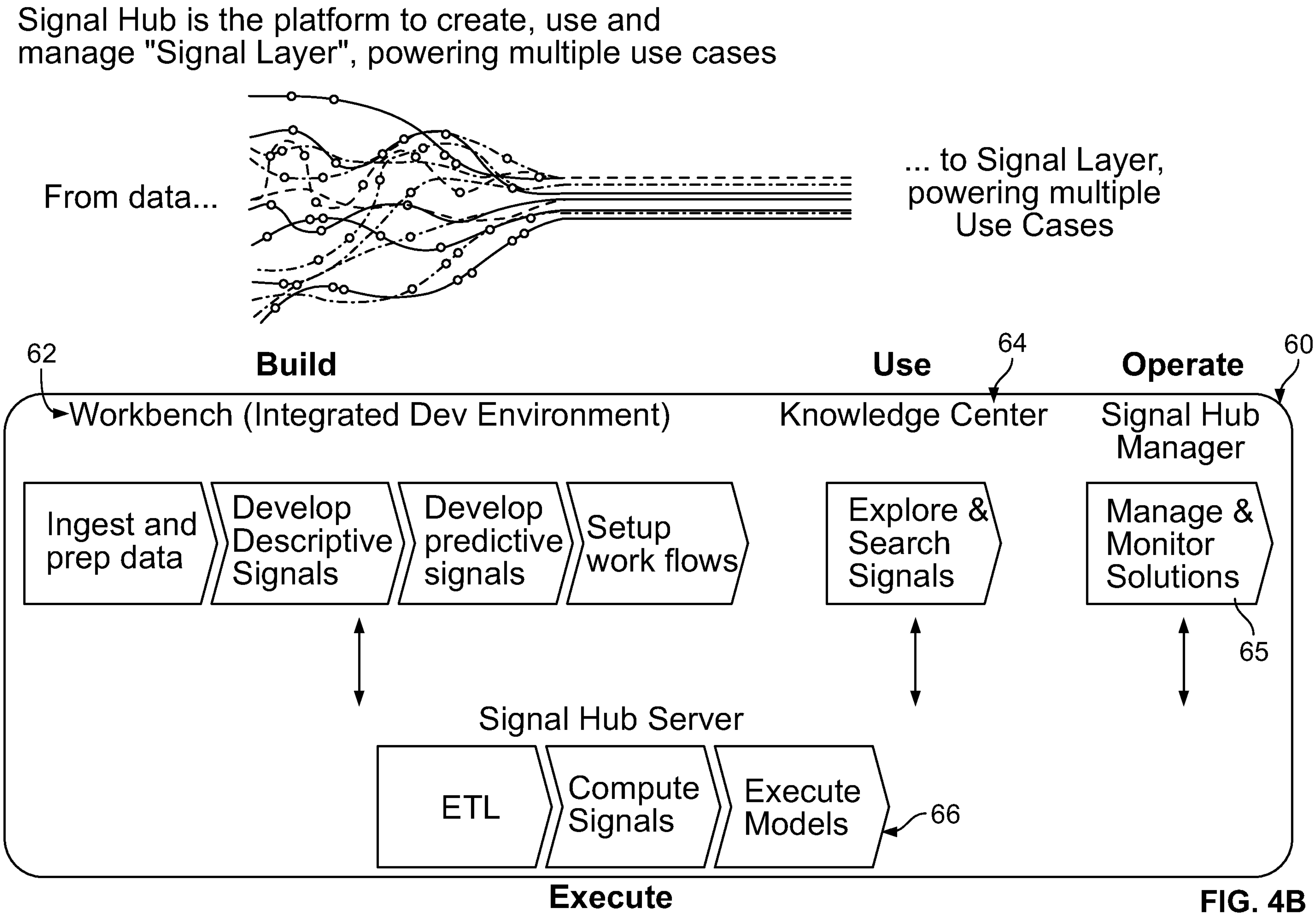
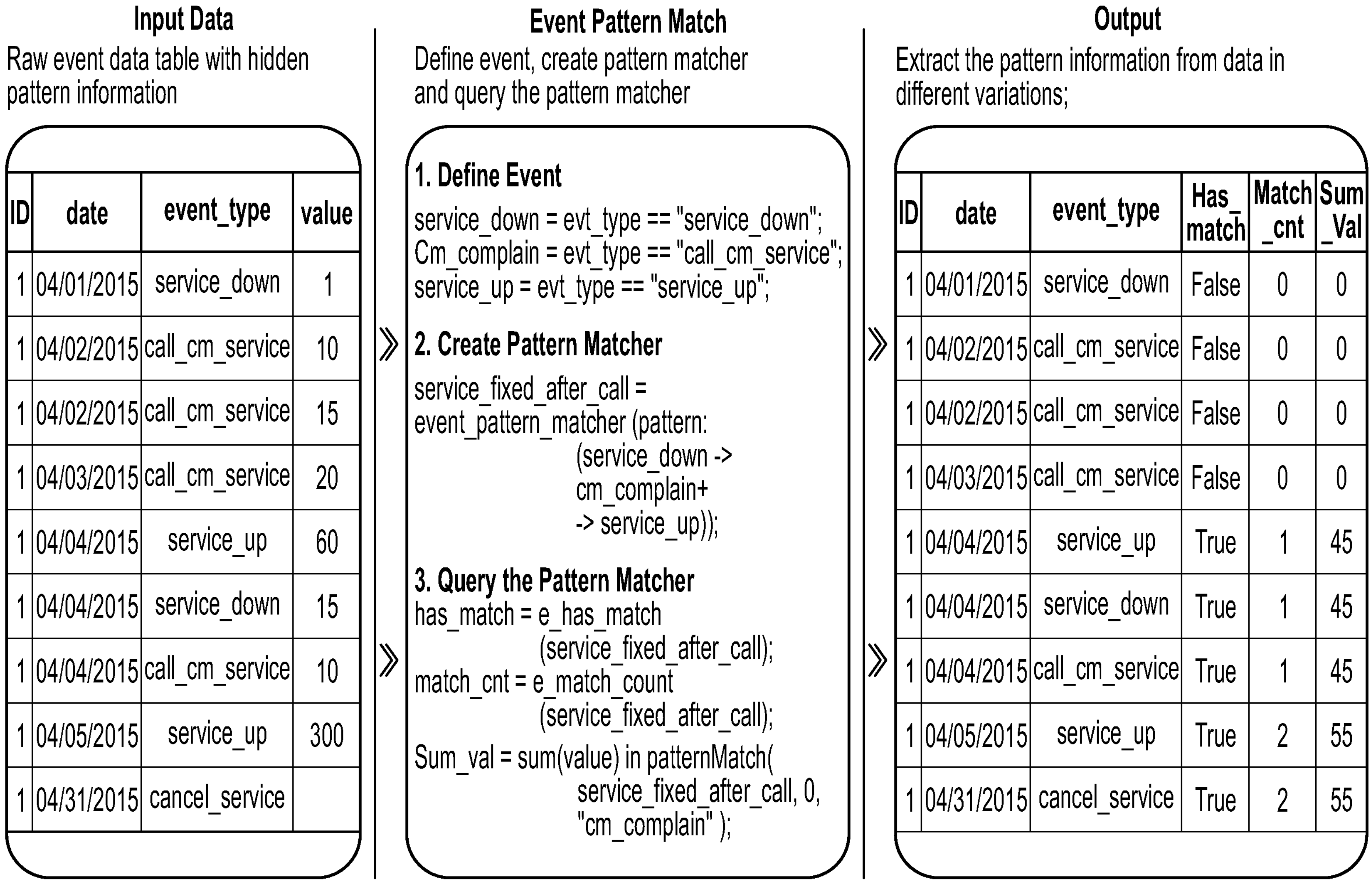

Signal Hub is the platform to create, use and manage "Signal Layer", powering multiple use cases
From data...
... to Signal Layer, powering multiple Use Cases
Build
Use
Operate
60
62
64
Workbench (Integrated Dev Environment)
Knowledge Center
Signal Hub Manager
Ingest and prep data
Develop Descriptive Signals
Develop predictive signals
Setup work flows
Explore & Search Signals
Manage & Monitor Solutions
65
Signal Hub Server
ETL
Compute Signals
Execute Models
66
Execute

FIG. 4B



Input Data

Raw event data table with hidden pattern information

ID	date	event_type	value
1	04/01/2015	service_down	1
1	04/02/2015	call_cm_service	10
1	04/02/2015	call_cm_service	15
1	04/03/2015	call_cm_service	20
1	04/04/2015	service_up	60
1	04/04/2015	service_down	15
1	04/04/2015	call_cm_service	10
1	04/05/2015	service_up	300
1	04/31/2015	cancel_service	

Event Pattern Match

Define event, create pattern matcher and query the pattern matcher

1. Define Event

```
service_down = evt_type == "service_down";
Cm_complain = evt_type == "call_cm_service";
service_up = evt_type == "service_up";
```

2. Create Pattern Matcher

```
service_fixed_after_call =
event_pattern_matcher (pattern:
                    (service_down ->
                    cm_complain+
                    -> service_up));
```

3. Query the Pattern Matcher

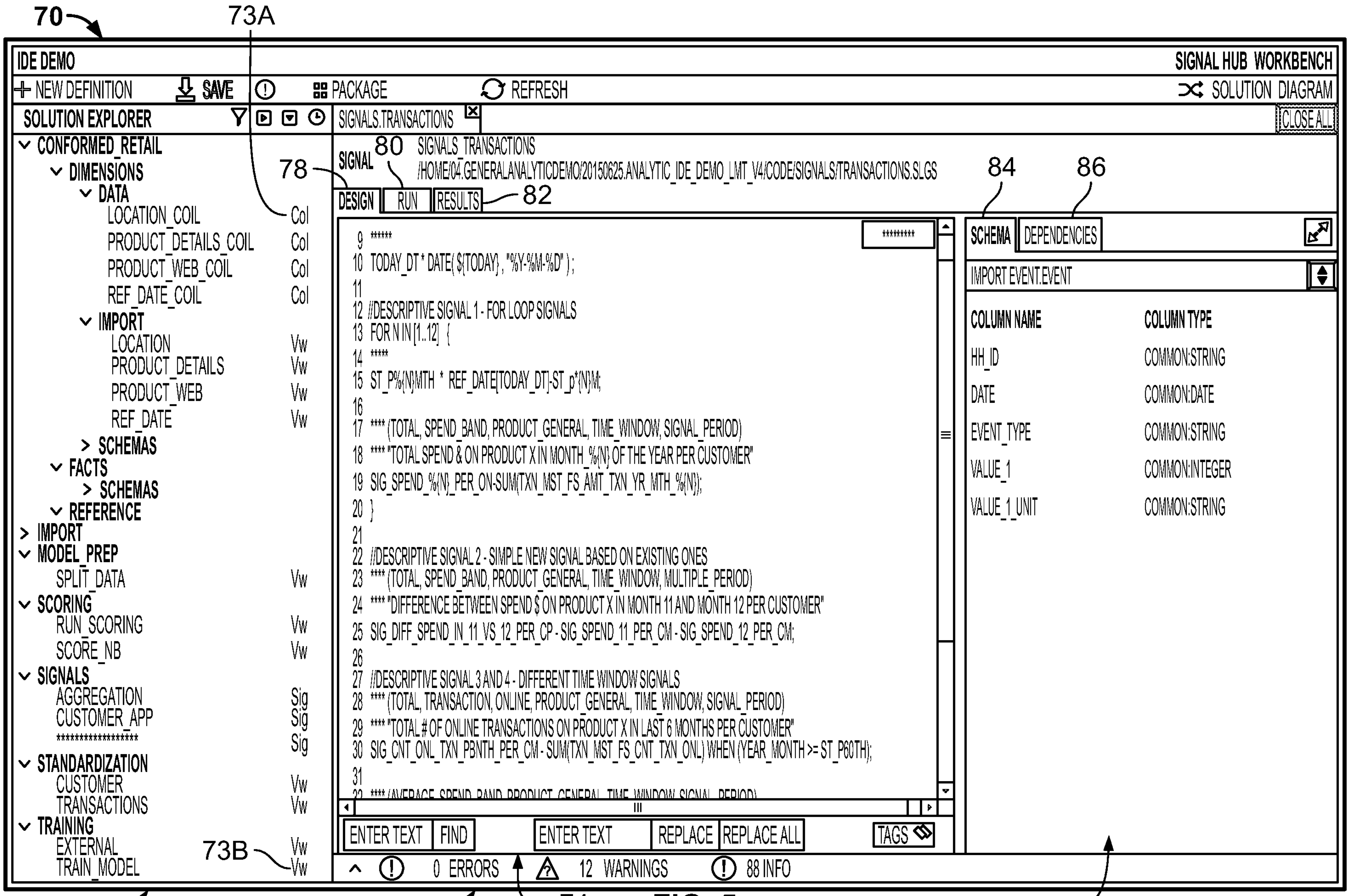
```
has_match = e_has_match
               (service_fixed_after_call);
match_cnt = e_match_count
               (service_fixed_after_call);
Sum_val = sum(value) in patternMatch(
                    service_fixed_after_call, 0,
                    "cm_complain" );
```

Output

Extract the pattern information from data in different variations;

ID	date	event_type	Has_ match	Match _cnt	Sum _Val
1	04/01/2015	service_down	False	0	0
1	04/02/2015	call_cm_service	False	0	0
1	04/02/2015	call_cm_service	False	0	0
1	04/03/2015	call_cm_service	False	0	0
1	04/04/2015	service_up	True	1	45
1	04/04/2015	service_down	True	1	45
1	04/04/2015	call_cm_service	True	1	45
1	04/05/2015	service_up	True	2	55
1	04/31/2015	cancel_service	True	2	55

FIG. 4C



```
CONFORMED_RETAIL
  DIMENSIONS
    DATA
      LOCATION_COIL           Col
      PRODUCT_DETAILS_COIL    Col
      PRODUCT_WEB_COIL        Col
      REF_DATE_COIL           Col
    IMPORT
      LOCATION                Vw
      PRODUCT_DETAILS         Vw
      PRODUCT_WEB             Vw
      REF_DATE                Vw
    SCHEMAS
  FACTS
    SCHEMAS
  REFERENCE
IMPORT
MODEL_PREP
  SPLIT_DATA                  Vw
SCORING
  RUN_SCORING                 Vw
  SCORE_NB                    Vw
SIGNALS
  AGGREGATION                 Sig
  CUSTOMER_APP                Sig
  *****************           Sig
STANDARDIZATION
  CUSTOMER                    Vw
  TRANSACTIONS                Vw
TRAINING
  EXTERNAL                    Vw
  TRAIN_MODEL                 Vw
```

SIGNAL SIGNALS_TRANSACTIONS
/HOME/04.GENERALANALYTICDEMO/20150625.ANALYTIC_IDE_DEMO_LMT_V4/CODE/SIGNALS/TRANSACTIONS.SLGS

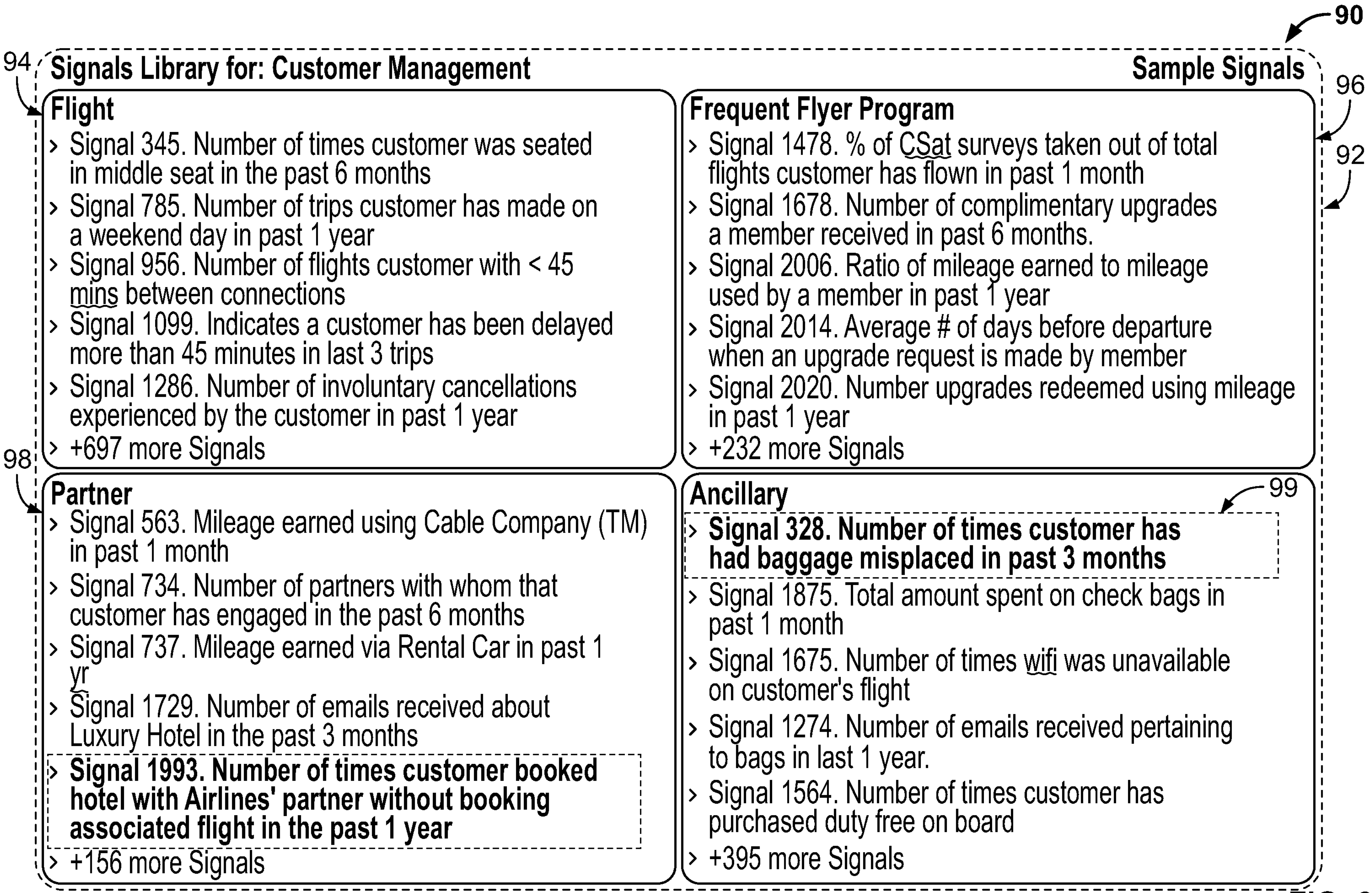
```
*****
TODAY_DT * DATE( ${TODAY} , "%Y-%M-%D" ) ;

//DESCRIPTIVE SIGNAL 1 - FOR LOOP SIGNALS
FOR N IN [1..12]  {
*****
ST_P%{N}MTH  *  REF_DATE[TODAY_DT]-ST_p*{N}M;

**** (TOTAL, SPEND_BAND, PRODUCT_GENERAL, TIME_WINDOW, SIGNAL_PERIOD)
**** "TOTAL SPEND & ON PRODUCT X IN MONTH_%{N} OF THE YEAR PER CUSTOMER"
SIG_SPEND_%{N}_PER_ON-SUM(TXN_MST_FS_AMT_TXN_YR_MTH_%{N});
}

//DESCRIPTIVE SIGNAL 2 - SIMPLE NEW SIGNAL BASED ON EXISTING ONES
**** (TOTAL, SPEND_BAND, PRODUCT_GENERAL, TIME_WINDOW, MULTIPLE_PERIOD)
**** "DIFFERENCE BETWEEN SPEND $ ON PRODUCT X IN MONTH 11 AND MONTH 12 PER CUSTOMER"
SIG_DIFF_SPEND_IN_11_VS_12_PER_CP - SIG_SPEND_11_PER_CM - SIG_SPEND_12_PER_CM;

//DESCRIPTIVE SIGNAL 3 AND 4 - DIFFERENT TIME WINDOW SIGNALS
**** (TOTAL, TRANSACTION, ONLINE, PRODUCT_GENERAL, TIME_WINDOW, SIGNAL_PERIOD)
**** "TOTAL # OF ONLINE TRANSACTIONS ON PRODUCT X IN LAST 6 MONTHS PER CUSTOMER"
SIG_CNT_ONL_TXN_PBNTH_PER_CM - SUM(TXN_MST_FS_CNT_TXN_ONL) WHEN (YEAR_MONTH >= ST_P60TH);

**** (AVERAGE_SPEND_BAND_PRODUCT_GENERAL_TIME_WINDOW_SIGNAL_PERIOD)
```

****

IMPORT EVENT.EVENT

COLUMN NAME	COLUMN TYPE
HH_ID	COMMON:STRING
DATE	COMMON:DATE
EVENT_TYPE	COMMON:STRING
VALUE_1	COMMON:INTEGER
VALUE_1_UNIT	COMMON:STRING

FIG. 5



Signals Library for: Customer Management

Sample Signals

Flight

- Signal 345. Number of times customer was seated in middle seat in the past 6 months
- Signal 785. Number of trips customer has made on a weekend day in past 1 year
- Signal 956. Number of flights customer with < 45 mins between connections
- Signal 1099. Indicates a customer has been delayed more than 45 minutes in last 3 trips
- Signal 1286. Number of involuntary cancellations experienced by the customer in past 1 year
- +697 more Signals

Frequent Flyer Program

- Signal 1478. % of CSat surveys taken out of total flights customer has flown in past 1 month
- Signal 1678. Number of complimentary upgrades a member received in past 6 months.
- Signal 2006. Ratio of mileage earned to mileage used by a member in past 1 year
- Signal 2014. Average # of days before departure when an upgrade request is made by member
- Signal 2020. Number upgrades redeemed using mileage in past 1 year
- +232 more Signals

Partner

- Signal 563. Mileage earned using Cable Company (TM) in past 1 month
- Signal 734. Number of partners with whom that customer has engaged in the past 6 months
- Signal 737. Mileage earned via Rental Car in past 1 yr
- Signal 1729. Number of emails received about Luxury Hotel in the past 3 months
- **Signal 1993. Number of times customer booked hotel with Airlines' partner without booking associated flight in the past 1 year**
- +156 more Signals

Ancillary

- **Signal 328. Number of times customer has had baggage misplaced in past 3 months**
- Signal 1875. Total amount spent on check bags in past 1 month
- Signal 1675. Number of times wifi was unavailable on customer's flight
- Signal 1274. Number of emails received pertaining to bags in last 1 year.
- Signal 1564. Number of times customer has purchased duty free on board
- +395 more Signals

FIG. 6

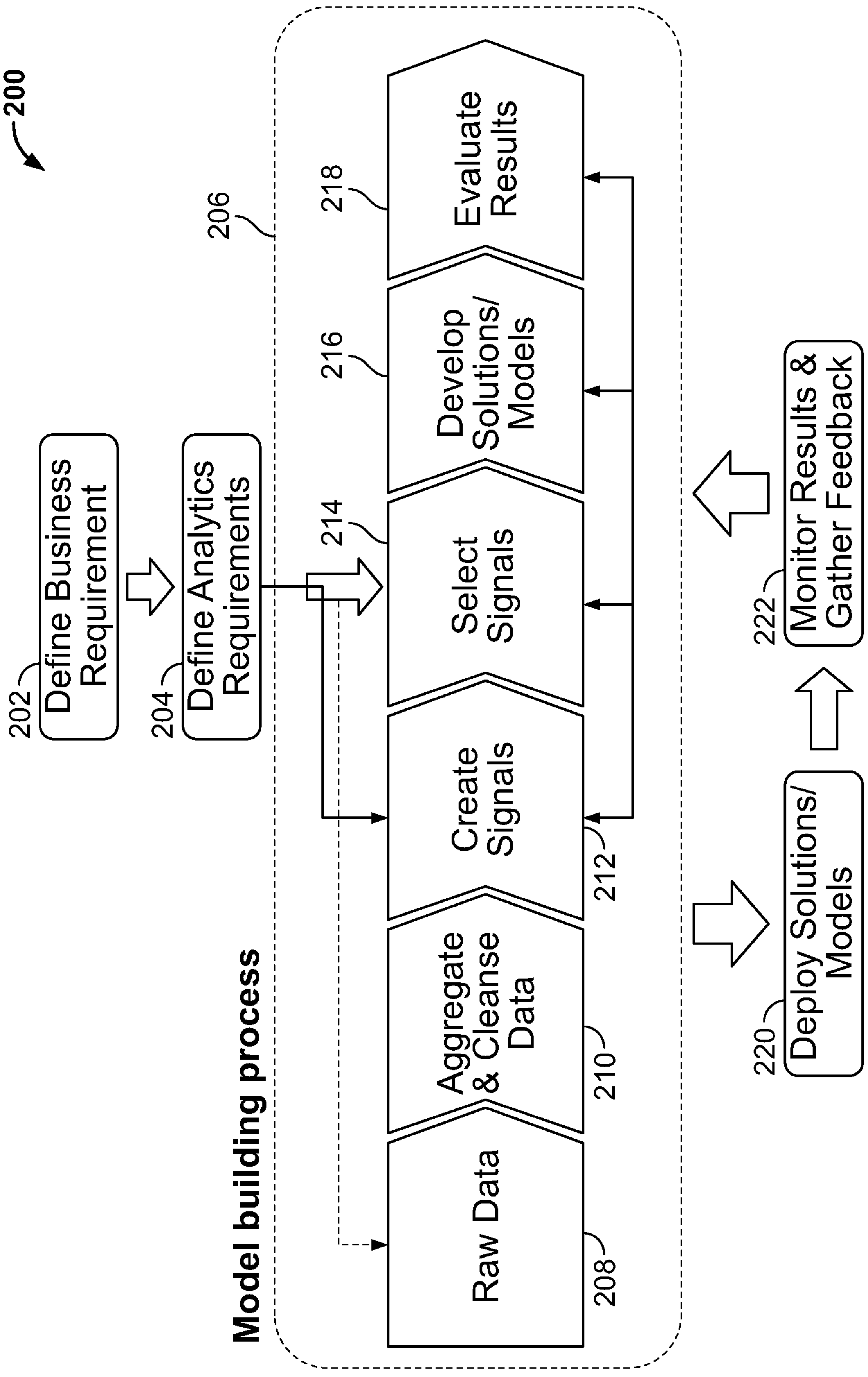

200
202
Define Business Requirement
204
Define Analytics Requirements
206
Model building process
208
Raw Data
210
Aggregate & Cleanse Data
212
Create Signals
214
Select Signals
216
Develop Solutions/ Models
218
Evaluate Results
220
Deploy Solutions/ Models
222
Monitor Results & Gather Feedback

FIG. 7

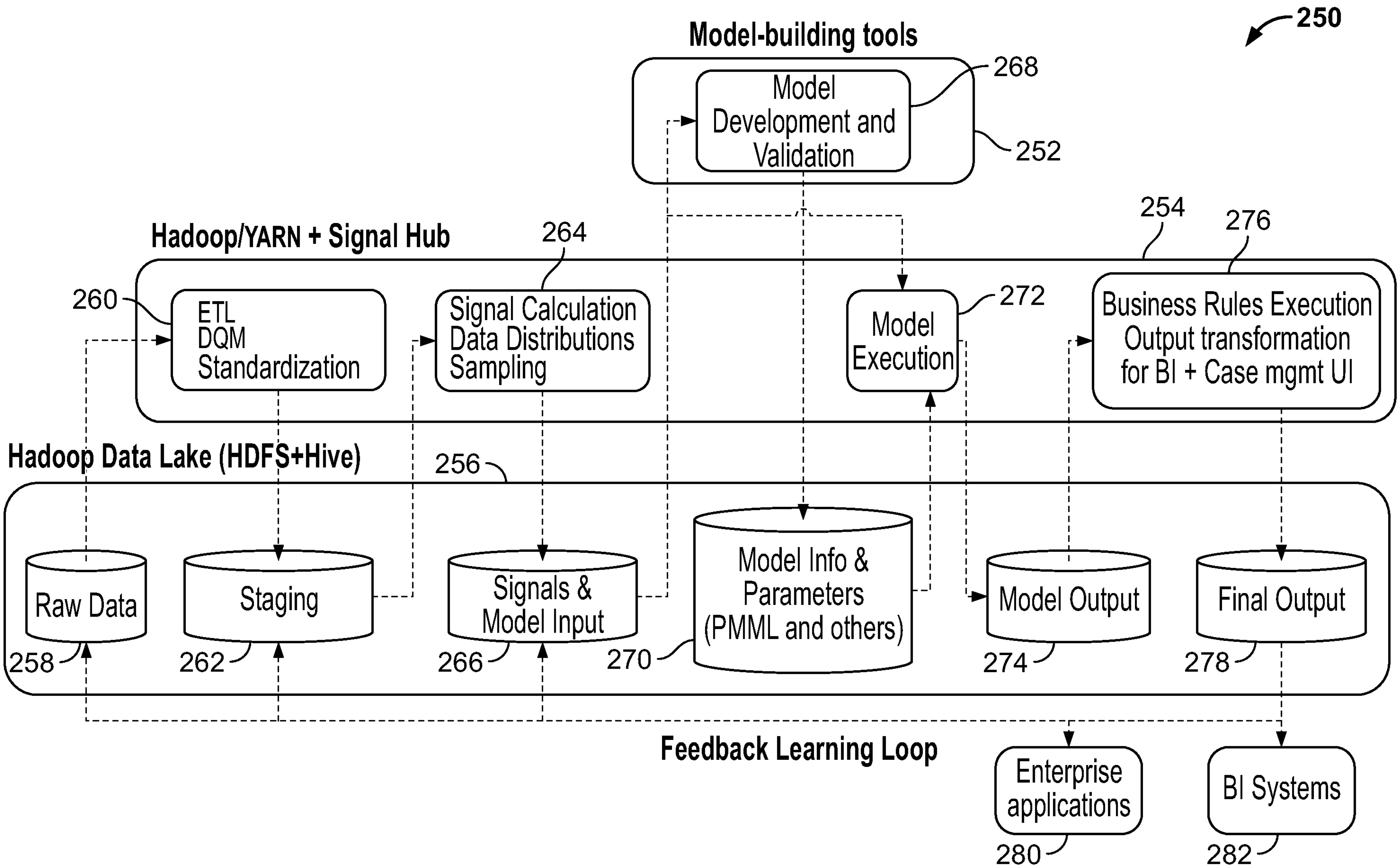

250
Model-building tools
Model Development and Validation
268
252
Hadoop/YARN + Signal Hub
264
254
276
260
ETL
DQM
Standardization
Signal Calculation
Data Distributions
Sampling
Model Execution
272
Business Rules Execution
Output transformation
for BI + Case mgmt UI
Hadoop Data Lake (HDFS+Hive)
256
Raw Data
Staging
Signals & Model Input
Model Info & Parameters (PMML and others)
Model Output
Final Output
258
262
266
270
274
278
Feedback Learning Loop
Enterprise applications
BI Systems
280
282

FIG. 8

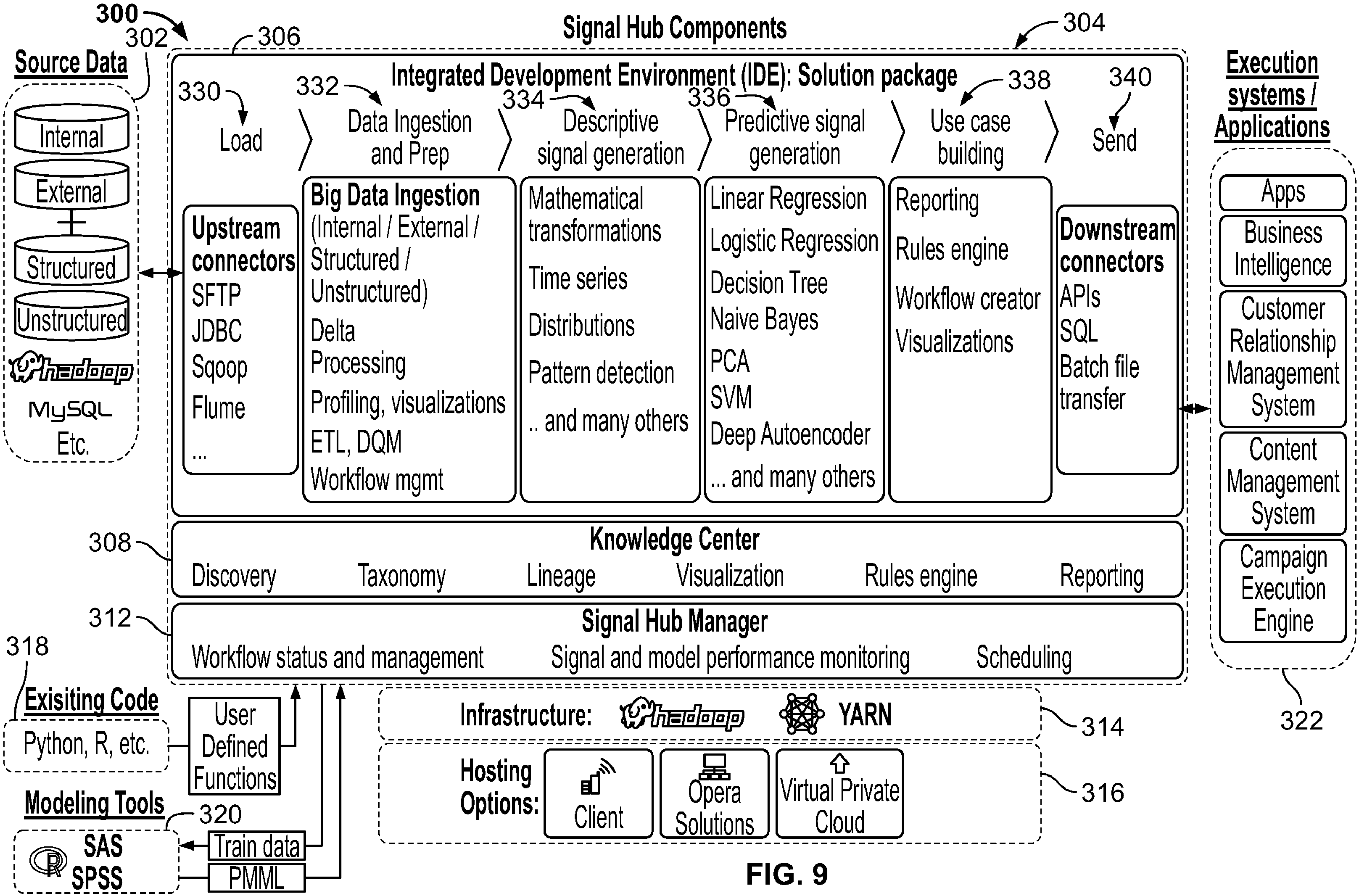

300
302
Source Data
Internal
External
Structured
Unstructured
hadoop
MySQL
Etc.
306
Signal Hub Components
304
Integrated Development Environment (IDE): Solution package
330
Load
332
Data Ingestion and Prep
334
Descriptive signal generation
336
Predictive signal generation
338
Use case building
340
Send
Upstream connectors
SFTP
JDBC
Sqoop
Flume
...
Big Data Ingestion
(Internal / External / Structured / Unstructured)
Delta
Processing
Profiling, visualizations
ETL, DQM
Workflow mgmt
Mathematical transformations
Time series
Distributions
Pattern detection
.. and many others
Linear Regression
Logistic Regression
Decision Tree
Naive Bayes
PCA
SVM
Deep Autoencoder
... and many others
Reporting
Rules engine
Workflow creator
Visualizations
Downstream connectors
APIs
SQL
Batch file transfer
Execution systems / Applications
Apps
Business Intelligence
Customer Relationship Management System
Content Management System
Campaign Execution Engine
322
308
Knowledge Center
Discovery
Taxonomy
Lineage
Visualization
Rules engine
Reporting
312
Signal Hub Manager
Workflow status and management
Signal and model performance monitoring
Scheduling
318
Exisiting Code
Python, R, etc.
User Defined Functions
Modeling Tools
320
SAS
SPSS
Train data
PMML
Infrastructure:
hadoop
YARN
314
Hosting Options:
Client
Opera Solutions
Virtual Private Cloud
316

FIG. 9

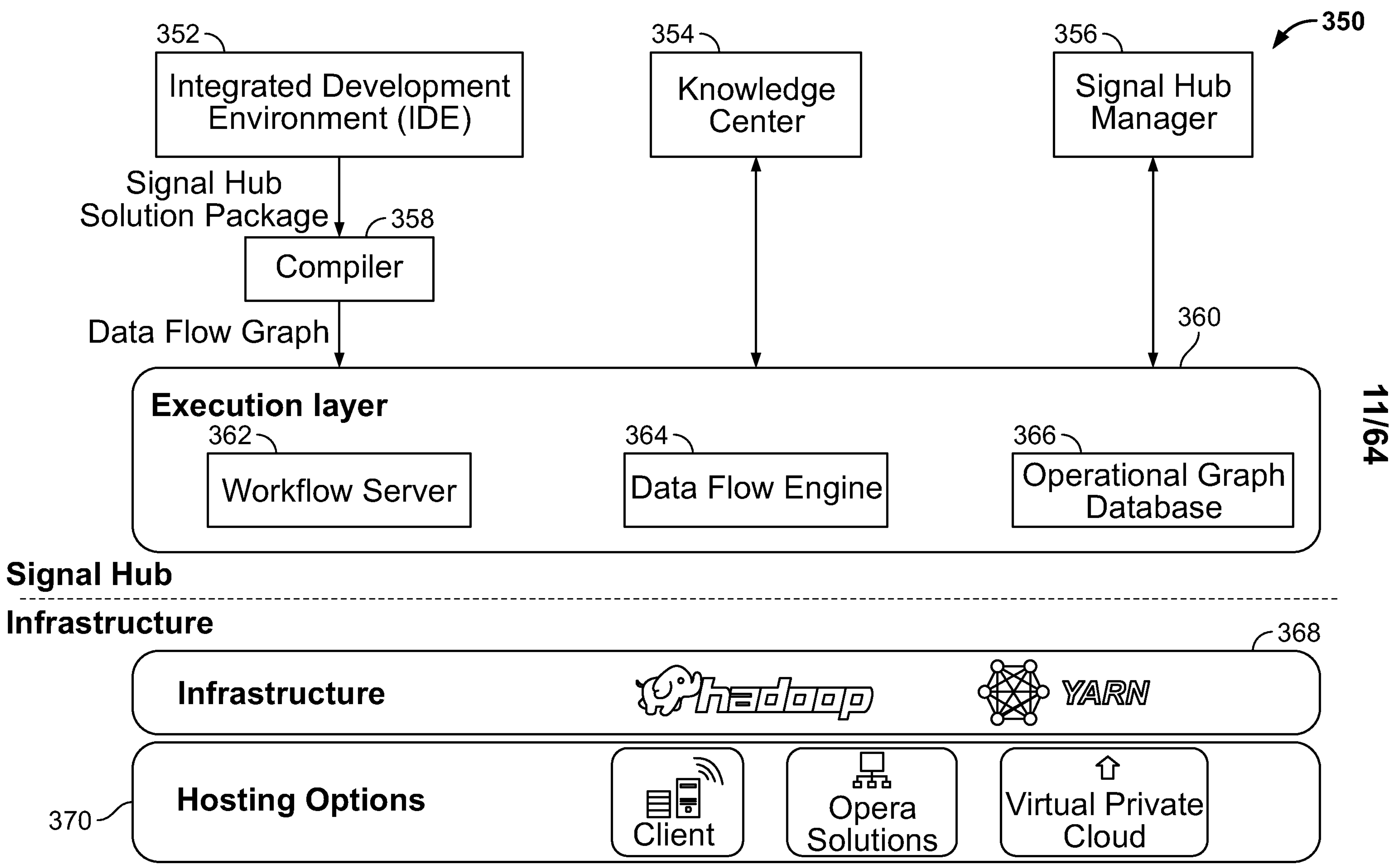
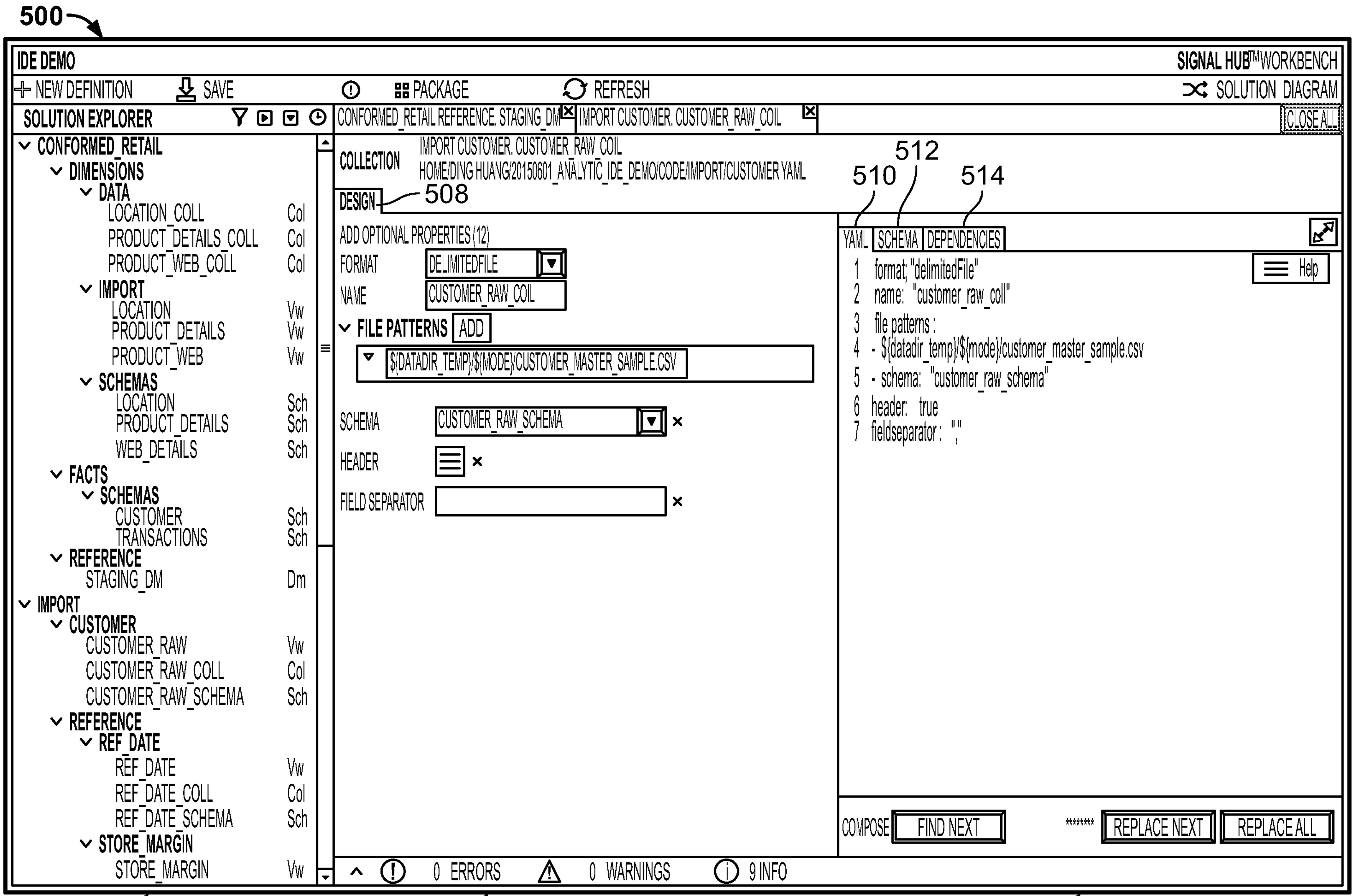

350
352
Integrated Development Environment (IDE)
354
Knowledge Center
356
Signal Hub Manager
Signal Hub Solution Package
358
Compiler
Data Flow Graph
360
Execution layer
362
Workflow Server
364
Data Flow Engine
366
Operational Graph Database
Signal Hub
Infrastructure
368
Infrastructure
hadoop
YARN
370
Hosting Options
Client
Opera Solutions
Virtual Private Cloud

FIG. 10



COLLECTION IMPORT CUSTOMER. CUSTOMER_RAW_COIL
HOME/DING HUANG/20150601_ANALYTIC_IDE_DEMO/CODE/IMPORT/CUSTOMER YAML

FORMAT: DELIMITEDFILE

NAME: CUSTOMER_RAW_COIL

FILE PATTERNS: ${DATADIR_TEMP}/${MODE}/CUSTOMER_MASTER_SAMPLE.CSV

SCHEMA: CUSTOMER_RAW_SCHEMA

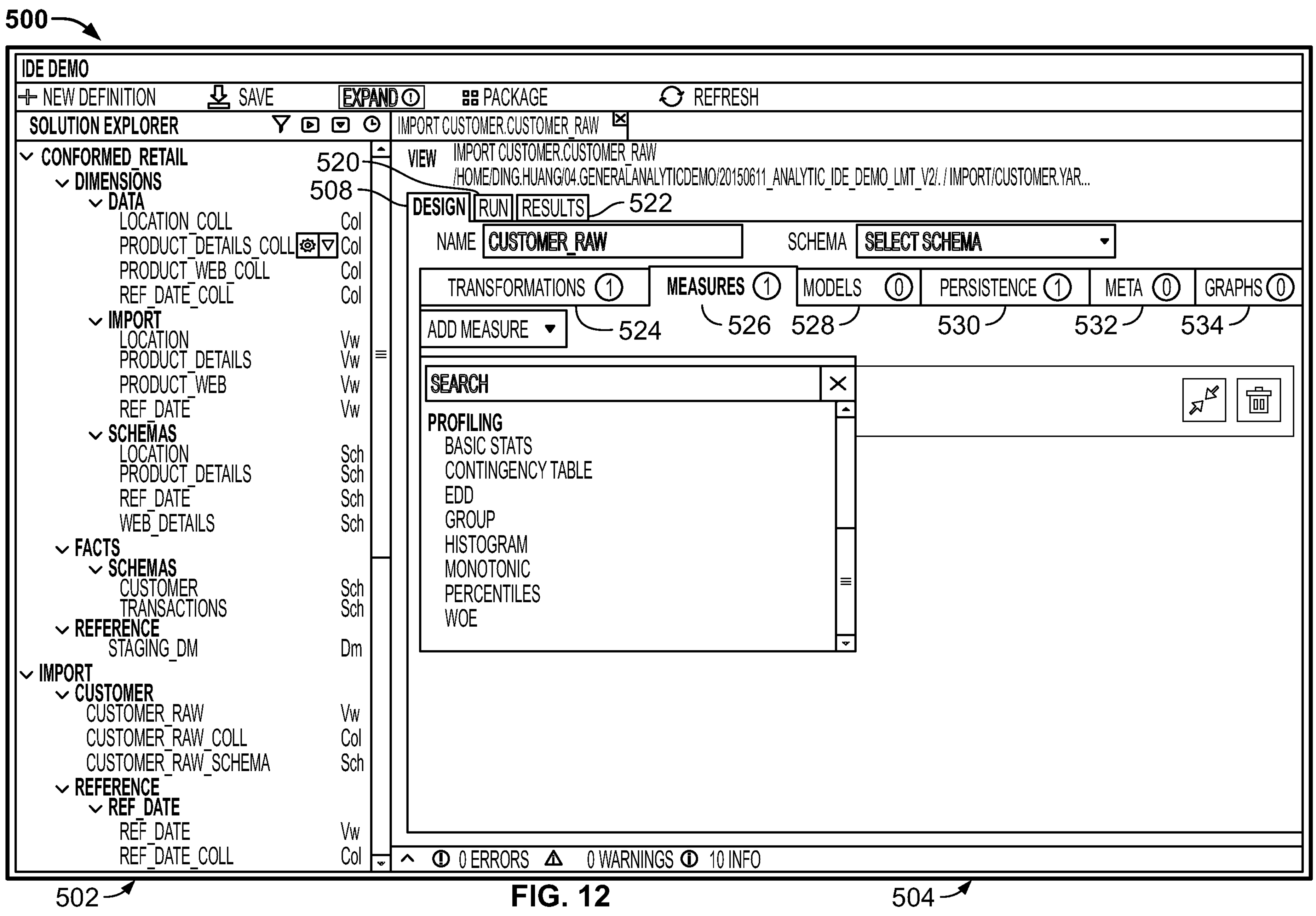
HEADER: ☰

FIELD SEPARATOR:

```
format; "delimitedFile"
name:  "customer_raw_coll"
file patterns :
- ${datadir_temp}/${mode}/customer_master_sample.csv
- schema:  "customer_raw_schema"
header:  true
fieldseparator :  ","
```

502 504 506

FIG. 11



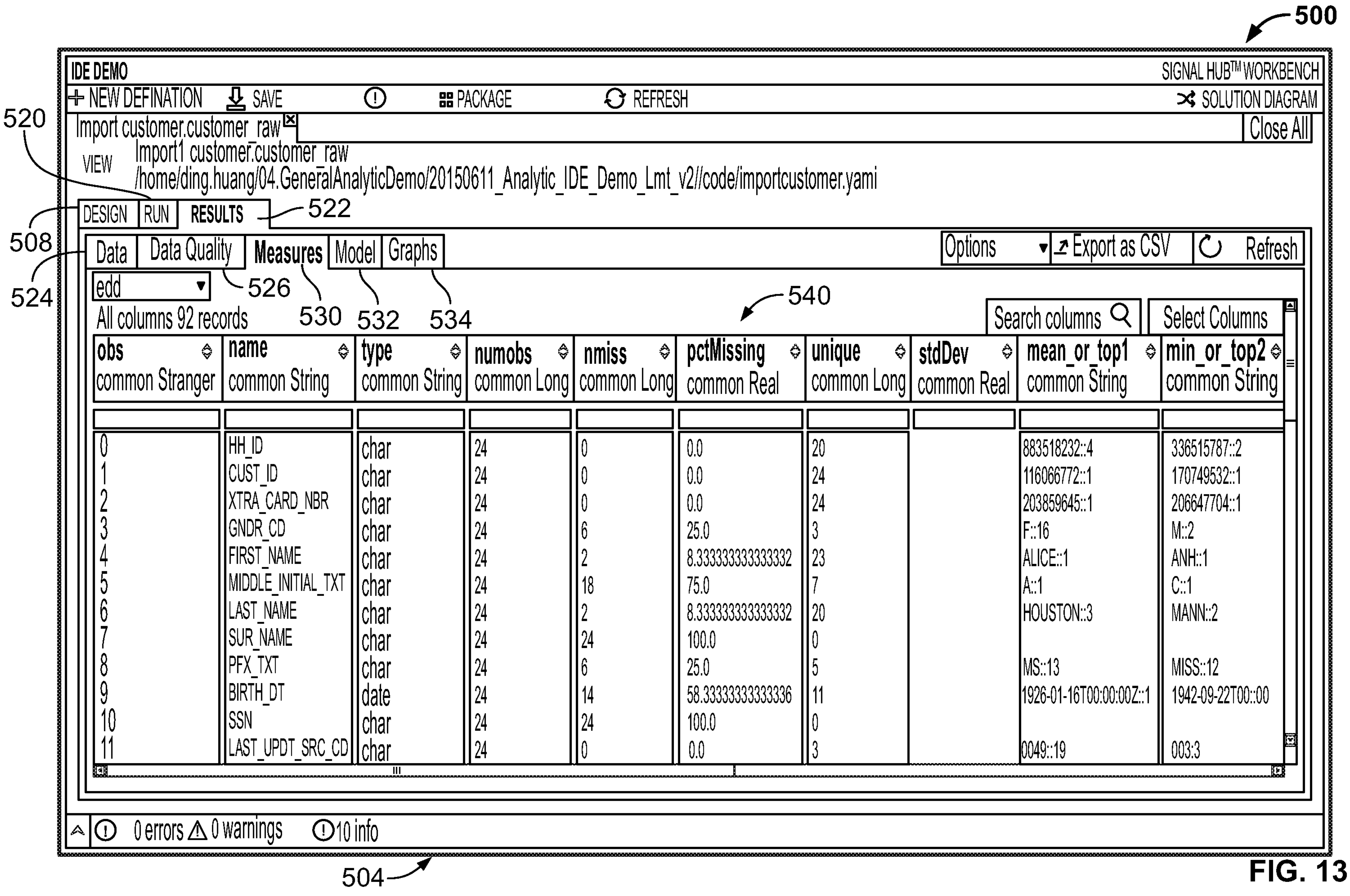
VIEW IMPORT CUSTOMER.CUSTOMER_RAW
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/. / IMPORT/CUSTOMER.YAR...

NAME CUSTOMER_RAW

SCHEMA SELECT SCHEMA

0 ERRORS 0 WARNINGS 10 INFO

FIG. 12



Import1 customer.customer_raw
/home/ding.huang/04.GeneralAnalyticDemo/20150611_Analytic_IDE_Demo_Lmt_v2//code/importcustomer.yami

All columns 92 records

obs common Stranger	name common String	type common String	numobs common Long	nmiss common Long	pctMissing common Real	unique common Long	stdDev common Real	mean_or_top1 common String	min_or_top2 common String
0	HH_ID	char	24	0	0.0	20		883518232::4	336515787::2
1	CUST_ID	char	24	0	0.0	24		116066772::1	170749532::1
2	XTRA_CARD_NBR	char	24	0	0.0	24		203859645::1	206647704::1
3	GNDR_CD	char	24	6	25.0	3		F::16	M::2
4	FIRST_NAME	char	24	2	8.333333333333332	23		ALICE::1	ANH::1
5	MIDDLE_INITIAL_TXT	char	24	18	75.0	7		A::1	C::1
6	LAST_NAME	char	24	2	8.333333333333332	20		HOUSTON::3	MANN::2
7	SUR_NAME	char	24	24	100.0	0			
8	PFX_TXT	char	24	6	25.0	5		MS::13	MISS::12
9	BIRTH_DT	date	24	14	58.33333333333336	11		1926-01-16T00:00:00Z::1	1942-09-22T00::00
10	SSN	char	24	24	100.0	0			
11	LAST_UPDT_SRC_CD	char	24	0	0.0	3		0049::19	003:3

FIG. 13

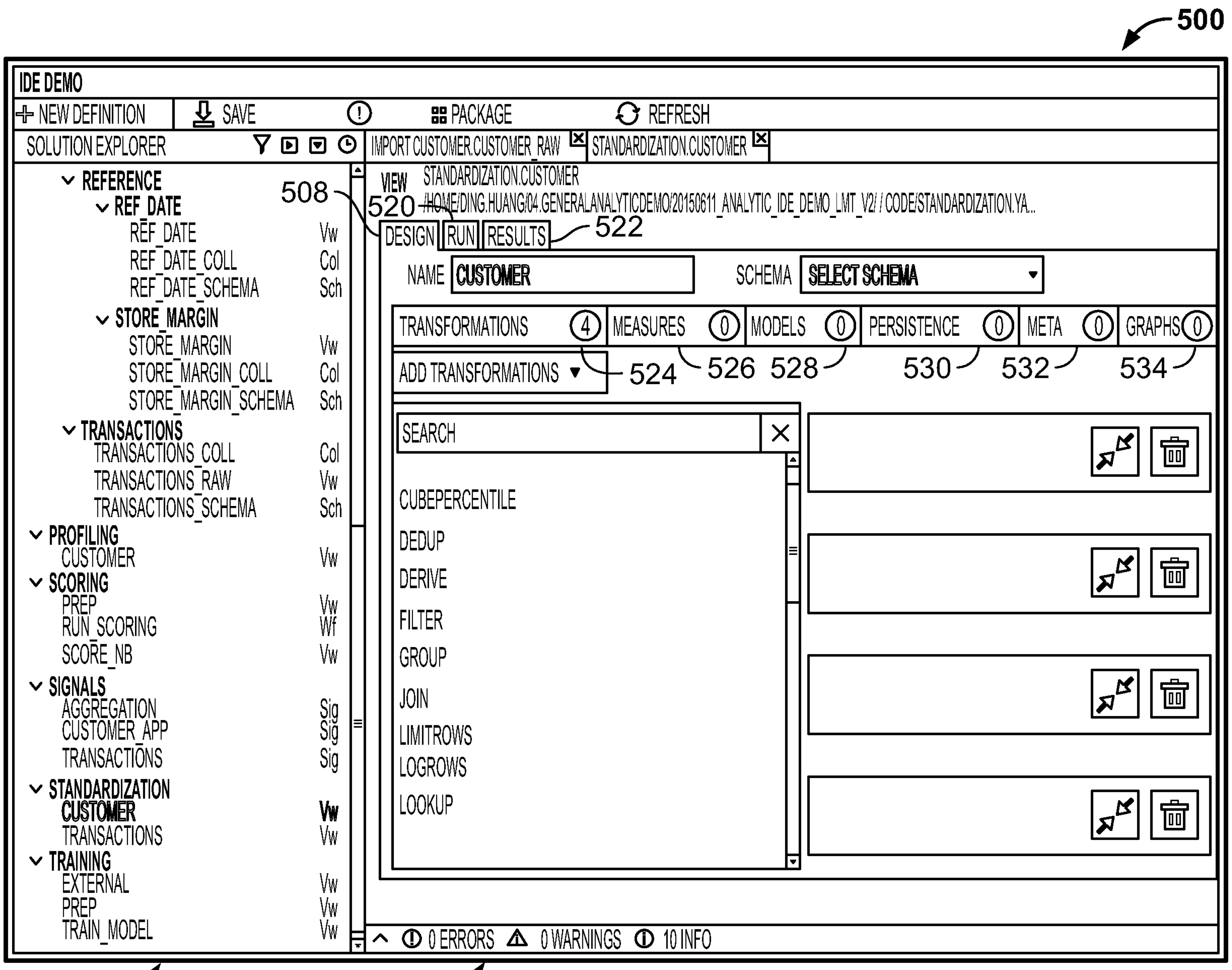
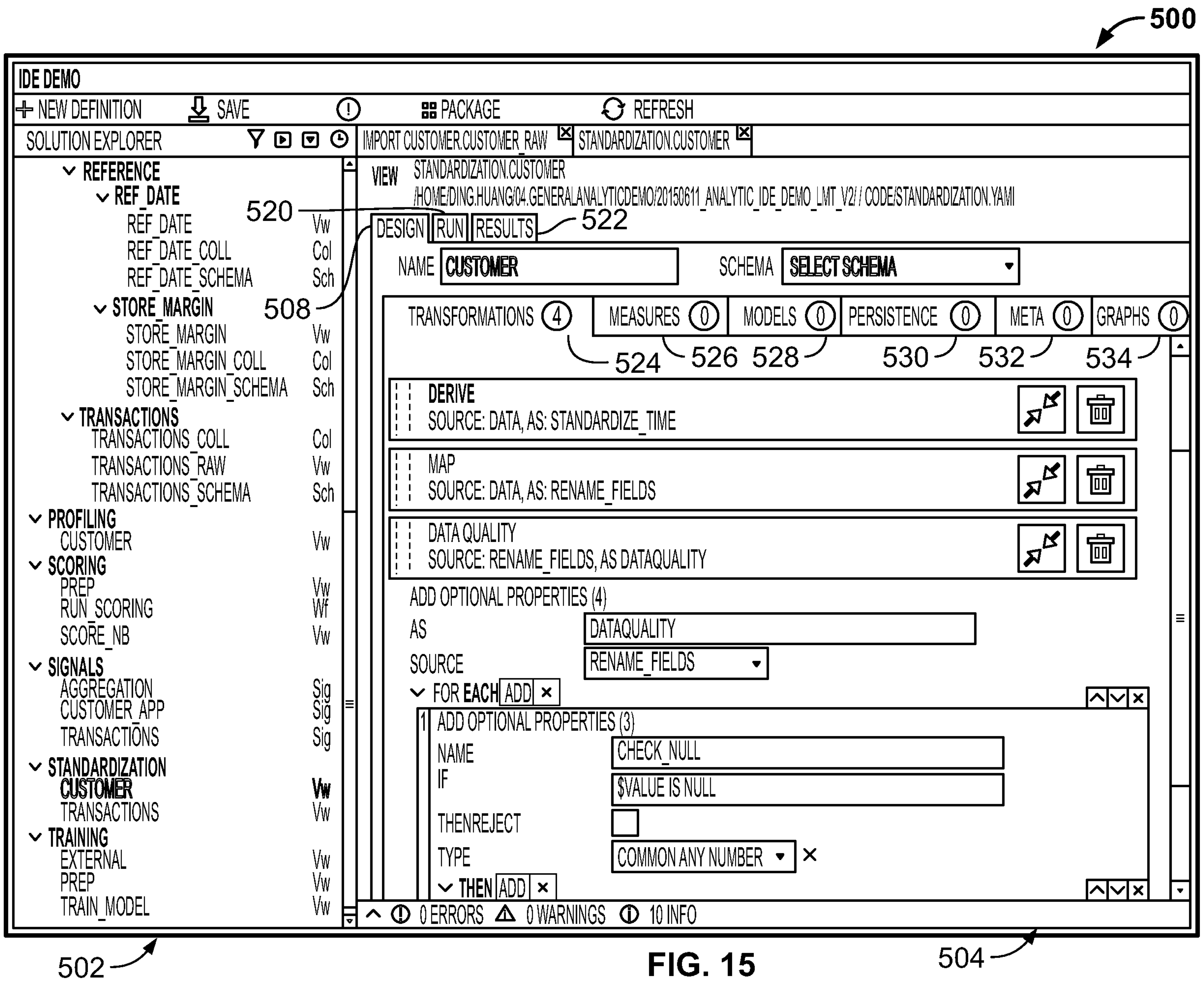

500
IDE DEMO
NEW DEFINITION
SAVE
PACKAGE
REFRESH
SOLUTION EXPLORER
REFERENCE
REF_DATE
REF_DATE Vw
REF_DATE_COLL Col
REF_DATE_SCHEMA Sch
STORE_MARGIN
STORE_MARGIN Vw
STORE_MARGIN_COLL Col
STORE_MARGIN_SCHEMA Sch
TRANSACTIONS
TRANSACTIONS_COLL Col
TRANSACTIONS_RAW Vw
TRANSACTIONS_SCHEMA Sch
PROFILING
CUSTOMER Vw
SCORING
PREP Vw
RUN_SCORING Wf
SCORE_NB Vw
SIGNALS
AGGREGATION Sig
CUSTOMER_APP Sig
TRANSACTIONS Sig
STANDARDIZATION
CUSTOMER Vw
TRANSACTIONS Vw
TRAINING
EXTERNAL Vw
PREP Vw
TRAIN_MODEL Vw
508
IMPORT CUSTOMER.CUSTOMER_RAW
STANDARDIZATION.CUSTOMER
VIEW STANDARDIZATION.CUSTOMER
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/ / CODE/STANDARDIZATION.YA...
520
DESIGN
RUN
RESULTS
522
NAME CUSTOMER
SCHEMA SELECT SCHEMA
TRANSFORMATIONS 4
MEASURES 0
MODELS 0
PERSISTENCE 0
META 0
GRAPHS 0
ADD TRANSFORMATIONS
524
526
528
530
532
534
SEARCH
CUBEPERCENTILE
DEDUP
DERIVE
FILTER
GROUP
JOIN
LIMITROWS
LOGROWS
LOOKUP
0 ERRORS
0 WARNINGS
10 INFO
502
504

FIG. 14



VIEW STANDARDIZATION.CUSTOMER
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/ / CODE/STANDARDIZATION.YAMI

NAME CUSTOMER SCHEMA SELECT SCHEMA

REFERENCE	
REF_DATE	
REF_DATE	Vw
REF_DATE_COLL	Col
REF_DATE_SCHEMA	Sch
STORE_MARGIN	
STORE_MARGIN	Vw
STORE_MARGIN_COLL	Col
STORE_MARGIN_SCHEMA	Sch
TRANSACTIONS	
TRANSACTIONS_COLL	Col
TRANSACTIONS_RAW	Vw
TRANSACTIONS_SCHEMA	Sch
PROFILING	
CUSTOMER	Vw
SCORING	
PREP	Vw
RUN_SCORING	Wf
SCORE_NB	Vw
SIGNALS	
AGGREGATION	Sig
CUSTOMER_APP	Sig
TRANSACTIONS	Sig
STANDARDIZATION	
CUSTOMER	**Vw**
TRANSACTIONS	Vw
TRAINING	
EXTERNAL	Vw
PREP	Vw
TRAIN_MODEL	Vw

DERIVE
SOURCE: DATA, AS: STANDARDIZE_TIME

MAP
SOURCE: DATA, AS: RENAME_FIELDS

DATA QUALITY
SOURCE: RENAME_FIELDS, AS DATAQUALITY

ADD OPTIONAL PROPERTIES (4)

AS DATAQUALITY

SOURCE RENAME_FIELDS

FOR EACH

1 ADD OPTIONAL PROPERTIES (3)

NAME CHECK_NULL

IF $VALUE IS NULL

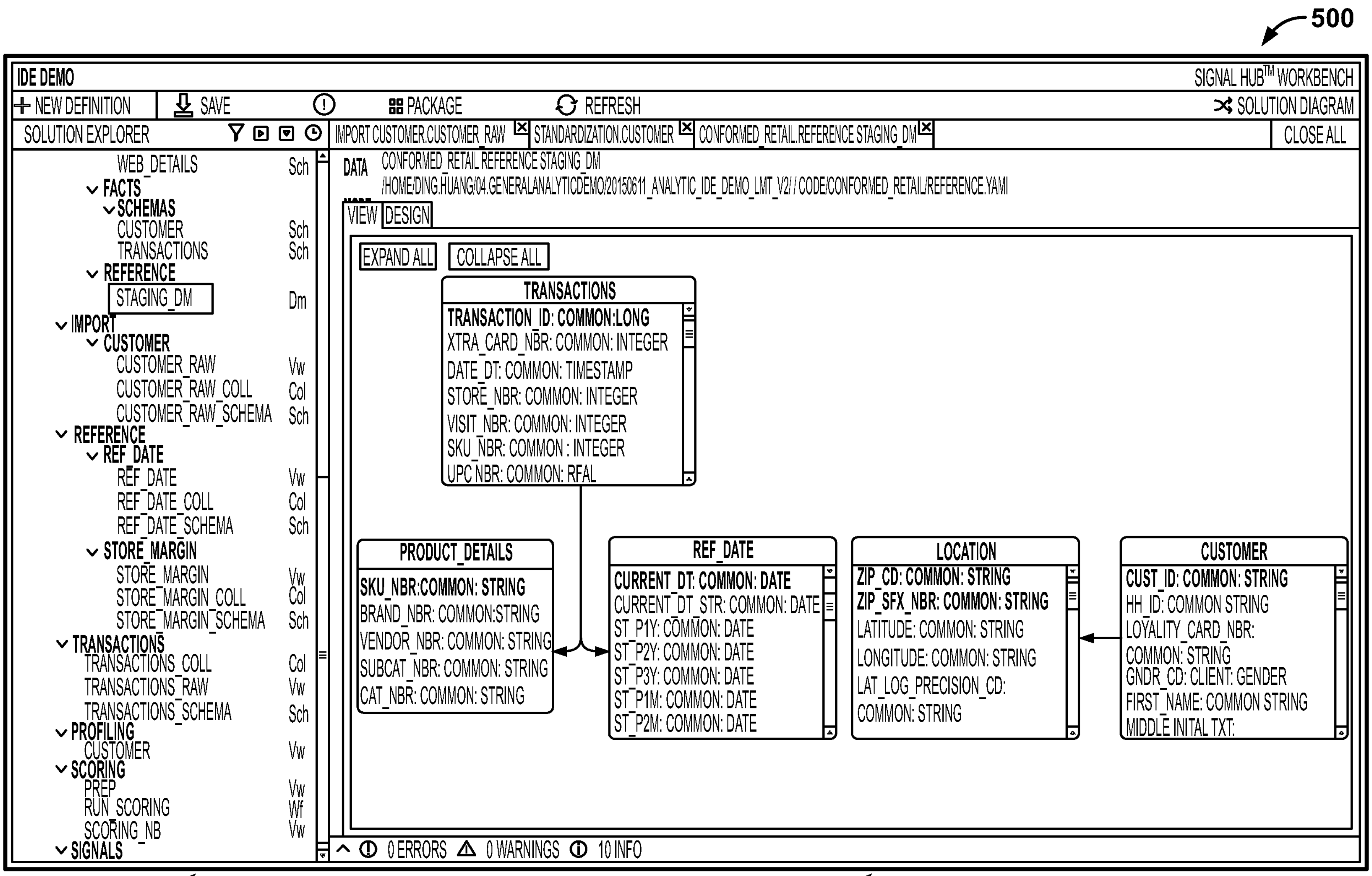
THENREJECT □

TYPE COMMON ANY NUMBER

THEN

0 ERRORS 0 WARNINGS 10 INFO

FIG. 15



WEB_DETAILS	Sch
∨ **FACTS**	
∨ **SCHEMAS**	
CUSTOMER	Sch
TRANSACTIONS	Sch
∨ **REFERENCE**	
STAGING_DM	Dm
∨ **IMPORT**	
∨ **CUSTOMER**	
CUSTOMER_RAW	Vw
CUSTOMER_RAW_COLL	Col
CUSTOMER_RAW_SCHEMA	Sch
∨ **REFERENCE**	
∨ **REF_DATE**	
REF_DATE	Vw
REF_DATE_COLL	Col
REF_DATE_SCHEMA	Sch
∨ **STORE_MARGIN**	
STORE_MARGIN	Vw
STORE_MARGIN_COLL	Col
STORE_MARGIN_SCHEMA	Sch
∨ **TRANSACTIONS**	
TRANSACTIONS_COLL	Col
TRANSACTIONS_RAW	Vw
TRANSACTIONS_SCHEMA	Sch
∨ **PROFILING**	
CUSTOMER	Vw
∨ **SCORING**	
PREP	Vw
RUN_SCORING	Wf
SCORING_NB	Vw
∨ **SIGNALS**	

DATA CONFORMED_RETAIL.REFERENCE STAGING_DM
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/ / CODE/CONFORMED_RETAIL/REFERENCE.YAMl

TRANSACTIONS
TRANSACTION_ID: COMMON:LONG
XTRA_CARD_NBR: COMMON: INTEGER
DATE_DT: COMMON: TIMESTAMP
STORE_NBR: COMMON: INTEGER
VISIT_NBR: COMMON: INTEGER
SKU_NBR: COMMON : INTEGER
UPC NBR: COMMON: RFAL

PRODUCT_DETAILS
SKU_NBR:COMMON: STRING
BRAND_NBR: COMMON:STRING
VENDOR_NBR: COMMON: STRING
SUBCAT_NBR: COMMON: STRING
CAT_NBR: COMMON: STRING

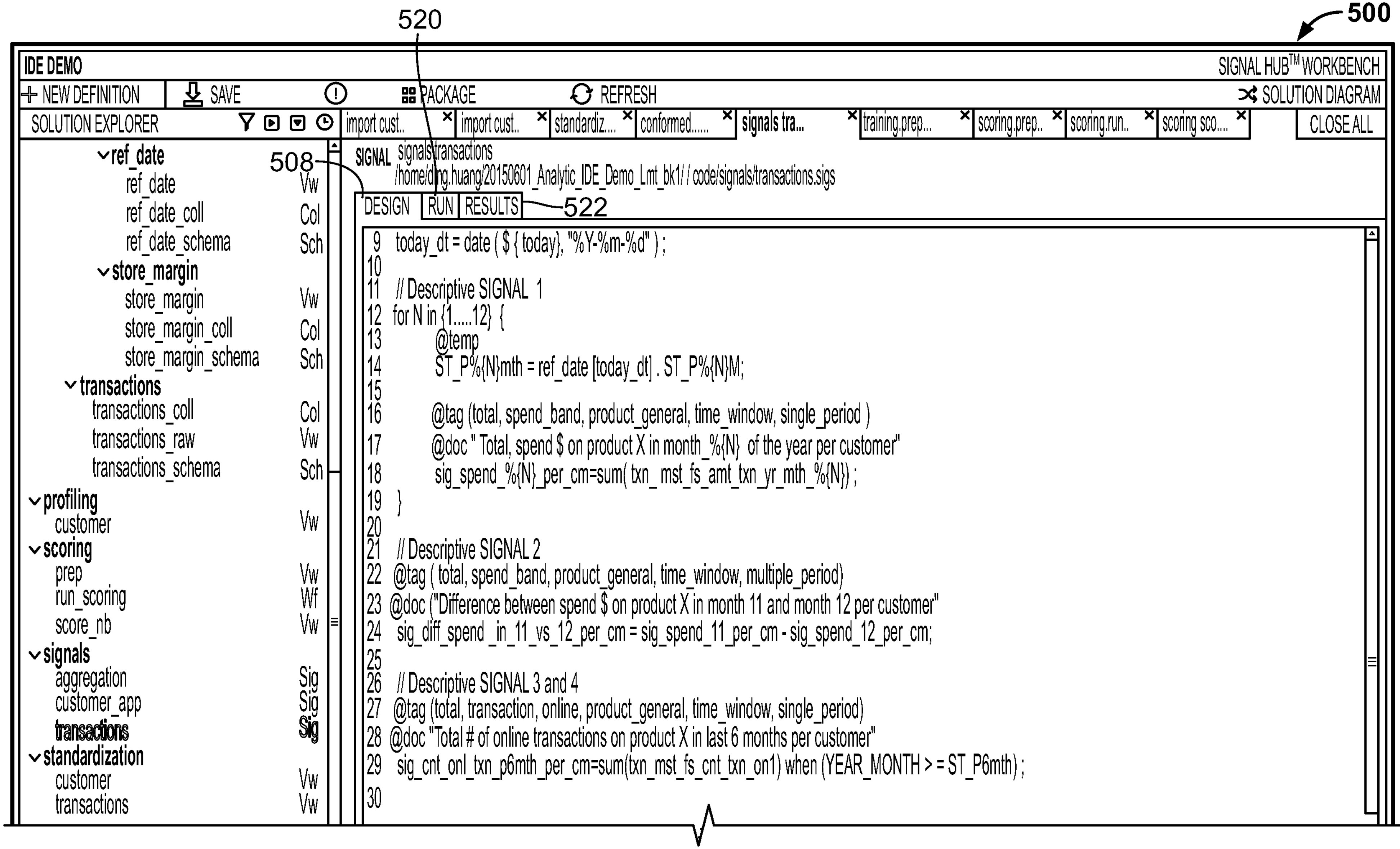
REF_DATE
CURRENT_DT: COMMON: DATE
CURRENT_DT_STR: COMMON: DATE
ST_P1Y: COMMON: DATE
ST_P2Y: COMMON: DATE
ST_P3Y: COMMON: DATE
ST_P1M: COMMON: DATE
ST_P2M: COMMON: DATE

LOCATION
ZIP_CD: COMMON: STRING
ZIP_SFX_NBR: COMMON: STRING
LATITUDE: COMMON: STRING
LONGITUDE: COMMON: STRING
LAT_LOG_PRECISION_CD:
COMMON: STRING

CUSTOMER
CUST_ID: COMMON: STRING
HH_ID: COMMON STRING
LOYALITY_CARD_NBR:
COMMON: STRING
GNDR_CD: CLIENT: GENDER
FIRST_NAME: COMMON STRING
MIDDLE INITAL TXT:

0 ERRORS 0 WARNINGS 10 INFO

FIG. 16

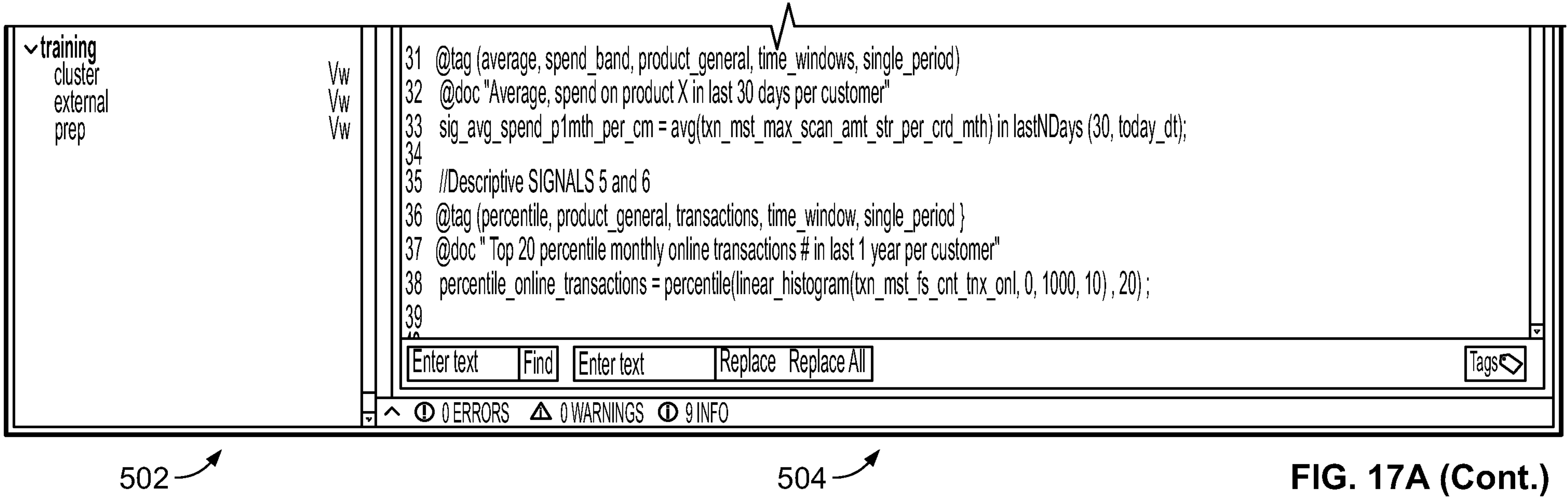


ref_date
ref_date Vw
ref_date_coll Col
ref_date_schema Sch
store_margin
store_margin Vw
store_margin_coll Col
store_margin_schema Sch
transactions
transactions_coll Col
transactions_raw Vw
transactions_schema Sch
profiling
customer Vw
scoring
prep Vw
run_scoring Wf
score_nb Vw
signals
aggregation Sig
customer_app Sig
transactions Sig
standardization
customer Vw
transactions Vw

SIGNAL signals\transactions
/home/ding.huang/20150601_Analytic_IDE_Demo_Lmt_bk1/ / code/signals/transactions.sigs

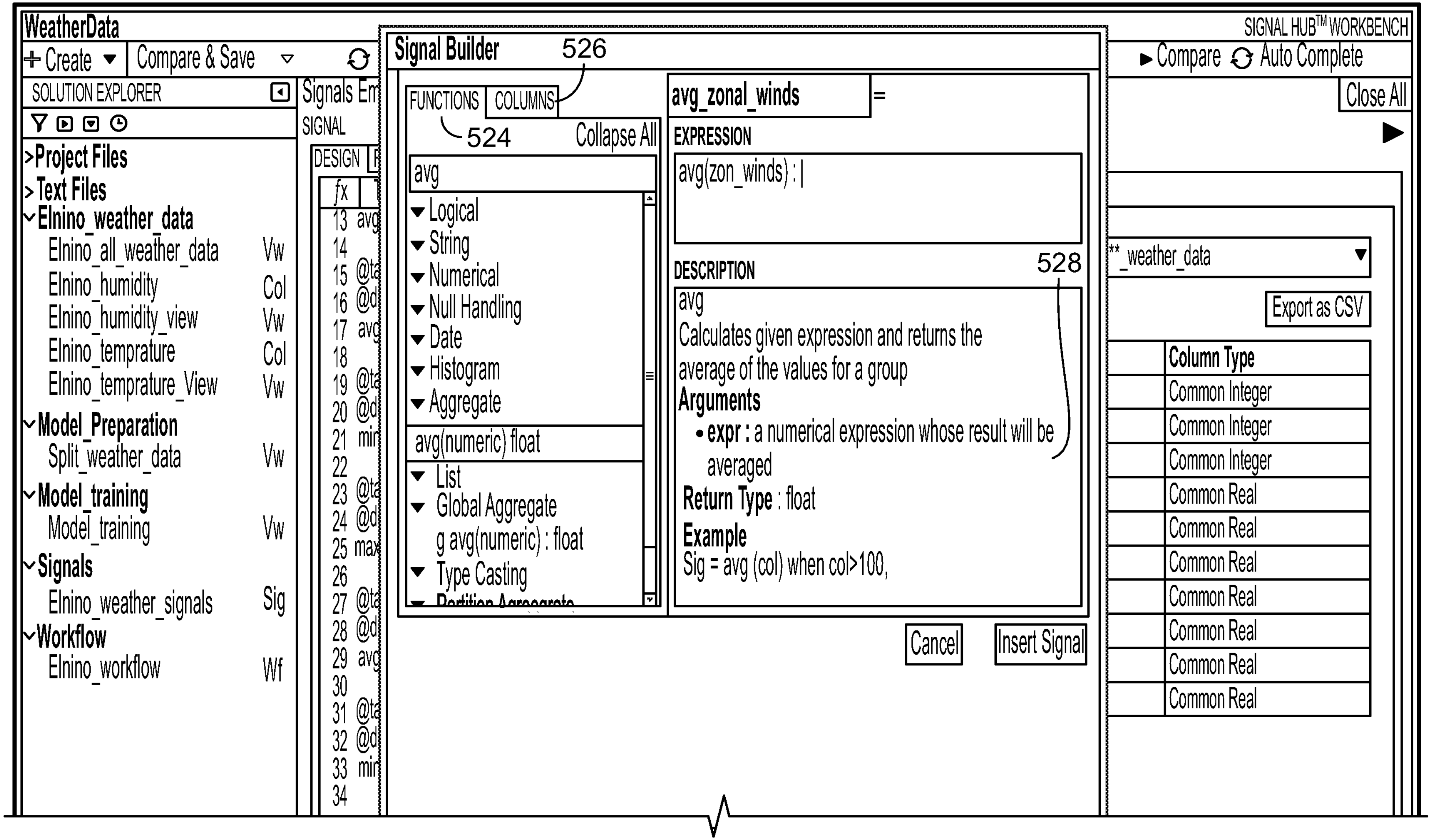
```
today_dt = date ( $ { today}, "%Y-%m-%d" ) ;

// Descriptive SIGNAL 1
for N in {1.....12} {
        @temp
        ST_P%{N}mth = ref_date [today_dt] . ST_P%{N}M;

       @tag (total, spend_band, product_general, time_window, single_period )
       @doc " Total, spend $ on product X in month_%{N}  of the year per customer"
        sig_spend_%{N}_per_cm=sum( txn_ mst_fs_amt_txn_yr_mth_%{N}) ;
}

// Descriptive SIGNAL 2
@tag ( total, spend_band, product_general, time_window, multiple_period)
@doc ("Difference between spend $ on product X in month 11 and month 12 per customer"
 sig_diff_spend _in_11_vs_12_per_cm = sig_spend_11_per_cm - sig_spend_12_per_cm;

// Descriptive SIGNAL 3 and 4
@tag (total, transaction, online, product_general, time_window, single_period)
@doc "Total # of online transactions on product X in last 6 months per customer"
 sig_cnt_onl_txn_p6mth_per_cm=sum(txn_mst_fs_cnt_txn_on1) when (YEAR_MONTH > = ST_P6mth) ;

```

FIG. 17A



```
@tag (average, spend_band, product_general, time_windows, single_period)
 @doc "Average, spend on product X in last 30 days per customer"
 sig_avg_spend_p1mth_per_cm = avg(txn_mst_max_scan_amt_str_per_crd_mth) in lastNDays (30, today_dt);

 //Descriptive SIGNALS 5 and 6
@tag (percentile, product_general, transactions, time_window, single_period }
@doc " Top 20 percentile monthly online transactions # in last 1 year per customer"
 percentile_online_transactions = percentile(linear_histogram(txn_mst_fs_cnt_tnx_onl, 0, 1000, 10) , 20) ;

```

FIG. 17A (Cont.)



avg_zonal_winds =

EXPRESSION

avg(zon_winds) : |

DESCRIPTION

avg

Calculates given expression and returns the average of the values for a group

Arguments

- **expr :** a numerical expression whose result will be averaged

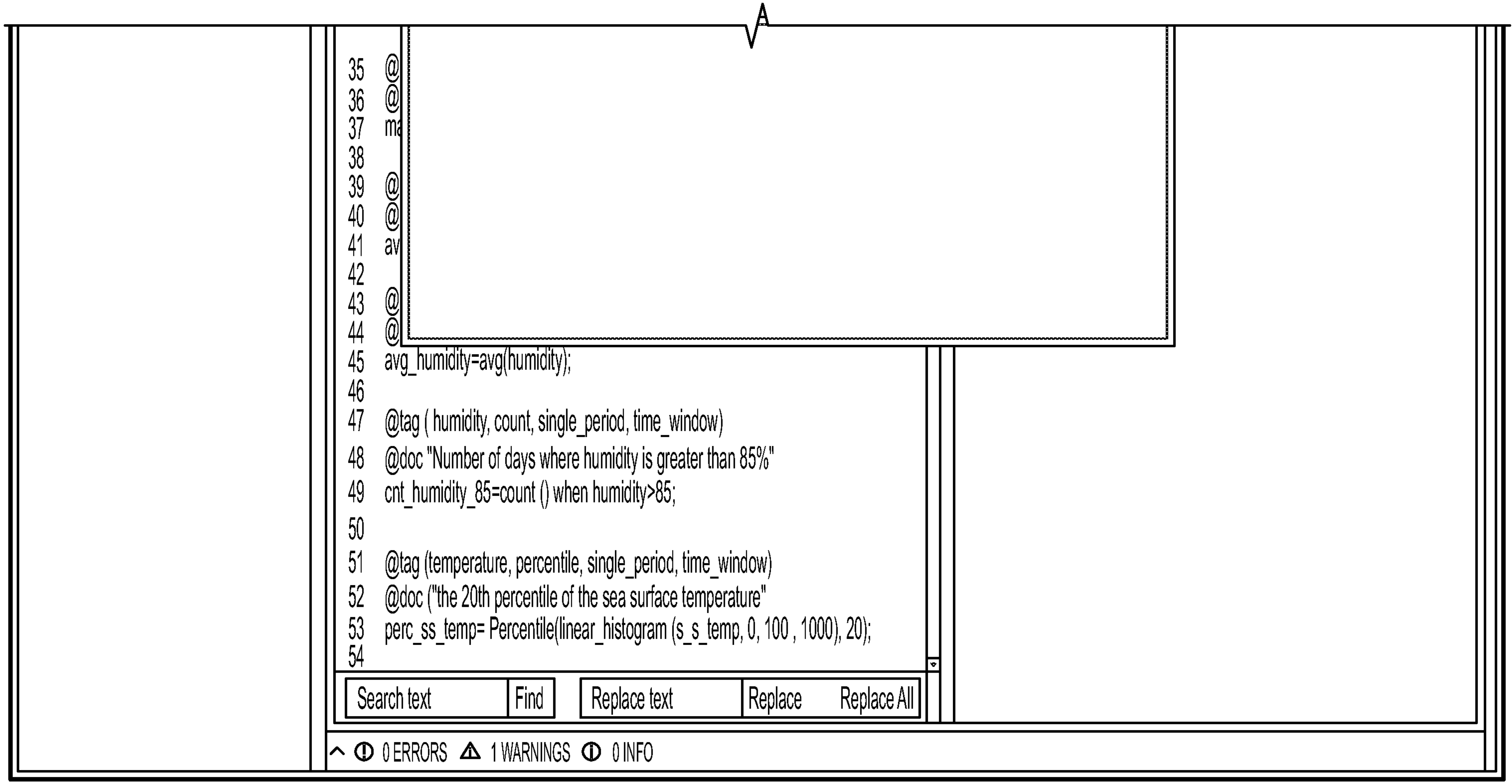
Return Type : float

Example

Sig = avg (col) when col>100,

Column Type
Common Integer
Common Integer
Common Integer
Common Real
Common Real
Common Real
Common Real
Common Real
Common Real
Common Real

FIG. 17B



```
@
@
ma

@
@
av

@
@
avg_humidity=avg(humidity);

@tag ( humidity, count, single_period, time_window)
@doc "Number of days where humidity is greater than 85%"
cnt_humidity_85=count () when humidity>85;

@tag (temperature, percentile, single_period, time_window)
@doc ("the 20th percentile of the sea surface temperature"
perc_ss_temp= Percentile(linear_histogram (s_s_temp, 0, 100 , 1000), 20);

```

0 ERRORS 1 WARNINGS 0 INFO

FIG. 17B (Cont.)

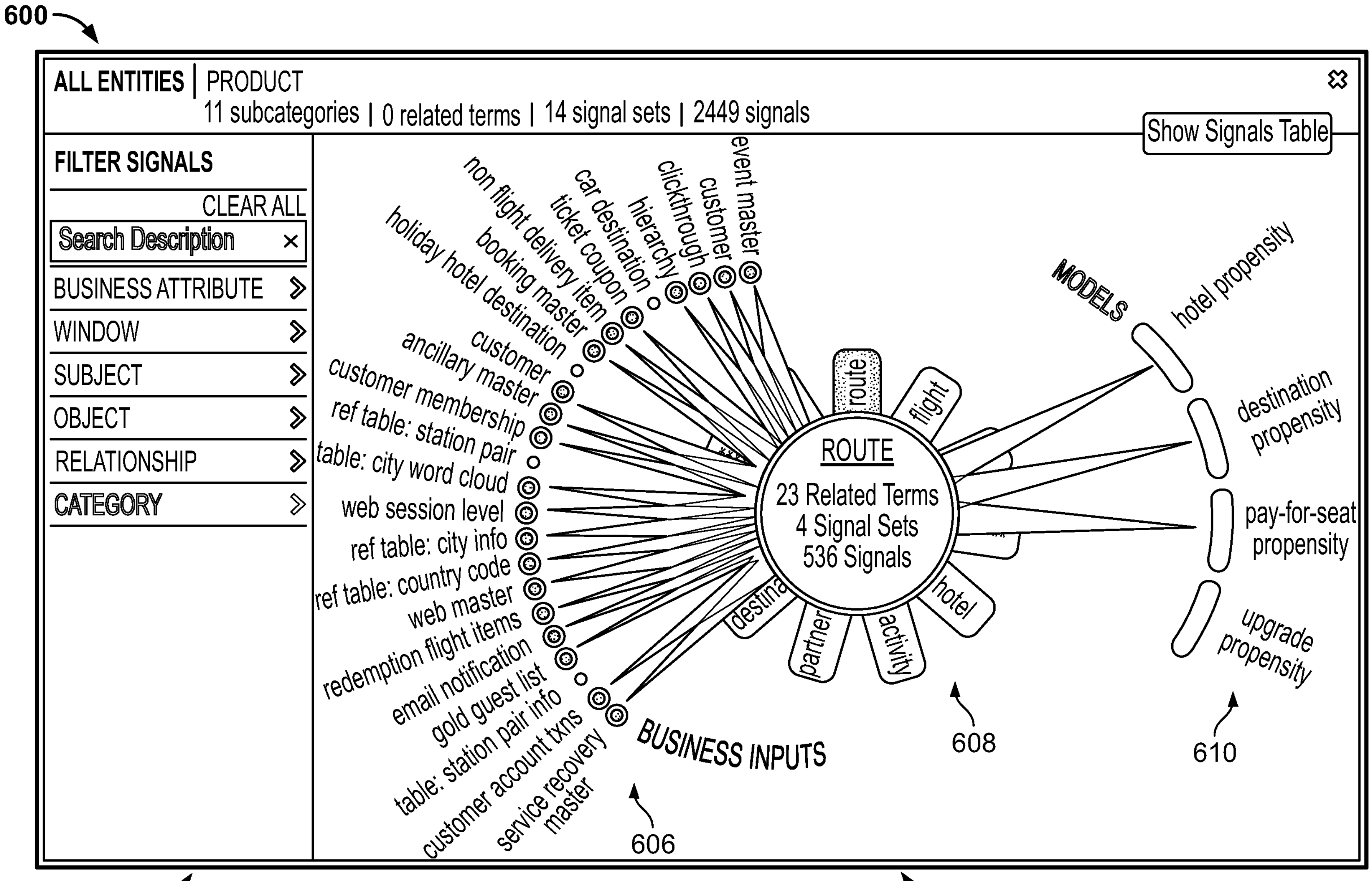
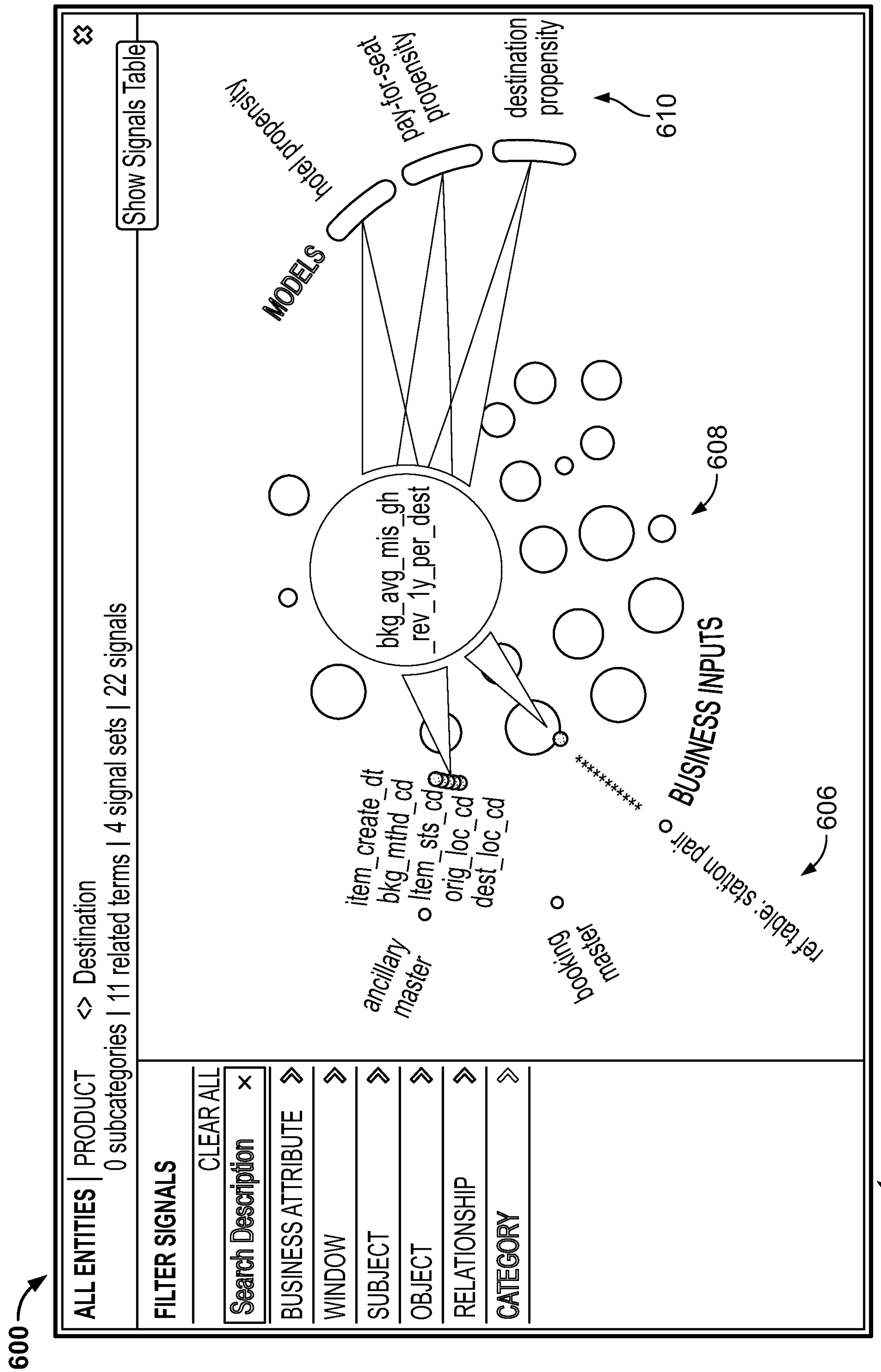
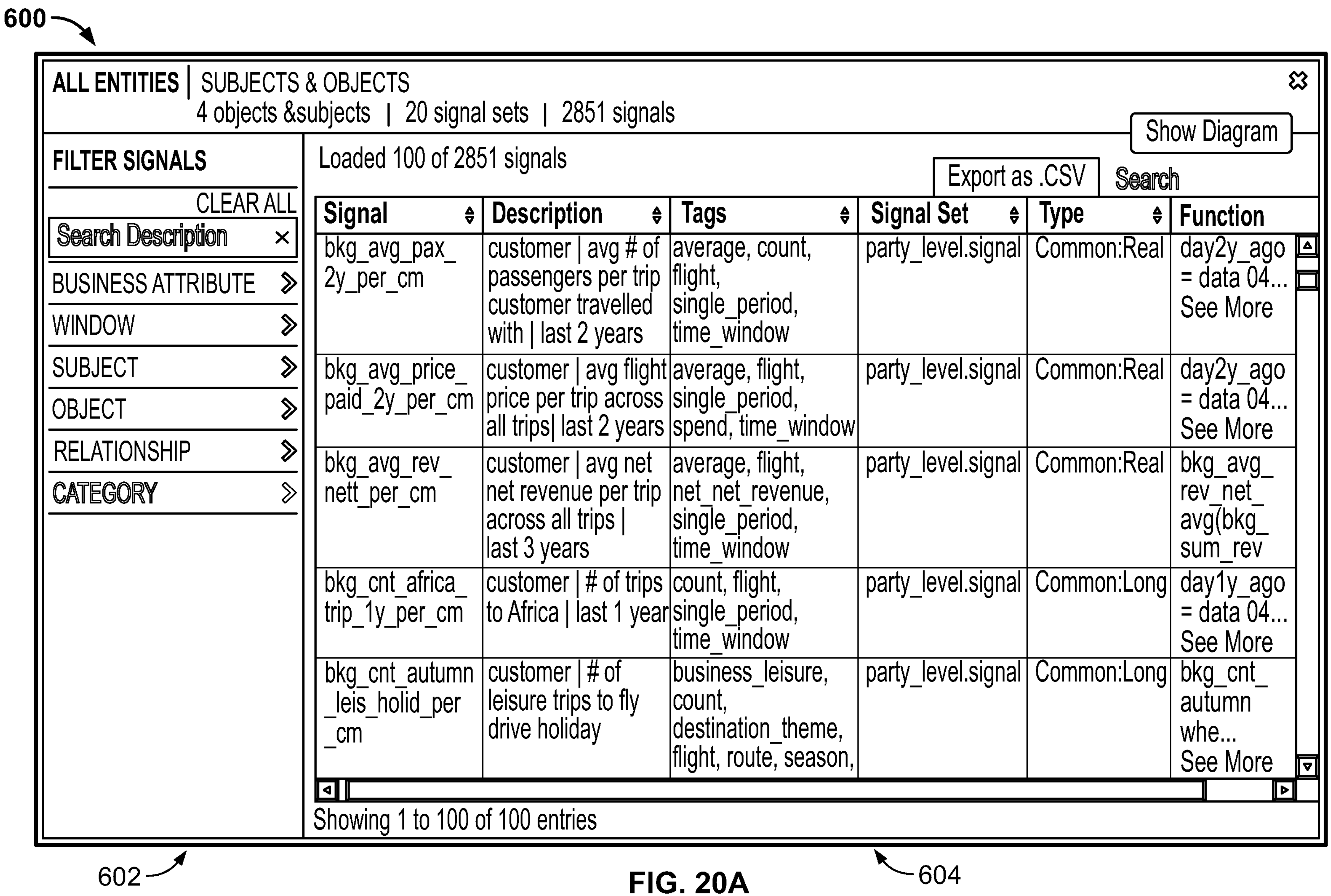

600
ALL ENTITIES | PRODUCT
11 subcategories | 0 related terms | 14 signal sets | 2449 signals
Show Signals Table
FILTER SIGNALS
CLEAR ALL
Search Description ×
BUSINESS ATTRIBUTE
WINDOW
SUBJECT
OBJECT
RELATIONSHIP
CATEGORY
event master
customer
clickthrough
hierarchy
car destination
ticket coupon
non flight delivery item
booking master
holiday hotel destination
customer
ancillary master
customer membership
ref table: station pair
table: city word cloud
web session level
ref table: city info
ref table: country code
web master
redemption flight items
email notification
gold guest list
table: station pair info
customer account txns
service recovery master
BUSINESS INPUTS
route
flight
ROUTE
23 Related Terms
4 Signal Sets
536 Signals
hotel
activity
partner
destina
MODELS
hotel propensity
destination propensity
pay-for-seat propensity
upgrade propensity
606
608
610
602
604

FIG. 18



600
ALL ENTITIES | PRODUCT <> Destination
0 subcategories | 11 related terms | 4 signal sets | 22 signals
Show Signals Table
FILTER SIGNALS
CLEAR ALL
Search Description ×
BUSINESS ATTRIBUTE
WINDOW
SUBJECT
OBJECT
RELATIONSHIP
CATEGORY
MODELS
hotel propensity
pay-for-seat propensity
destination propensity
610
bkg_avg_mis_gh _rev_1y_per_dest
ancillary master
item_create_dt
bkg_mthd_cd
Item_sts_cd
orig_loc_cd
dest_loc_cd
booking master
****
BUSINESS INPUTS
608
ref table: station pair
606
602

FIG. 19



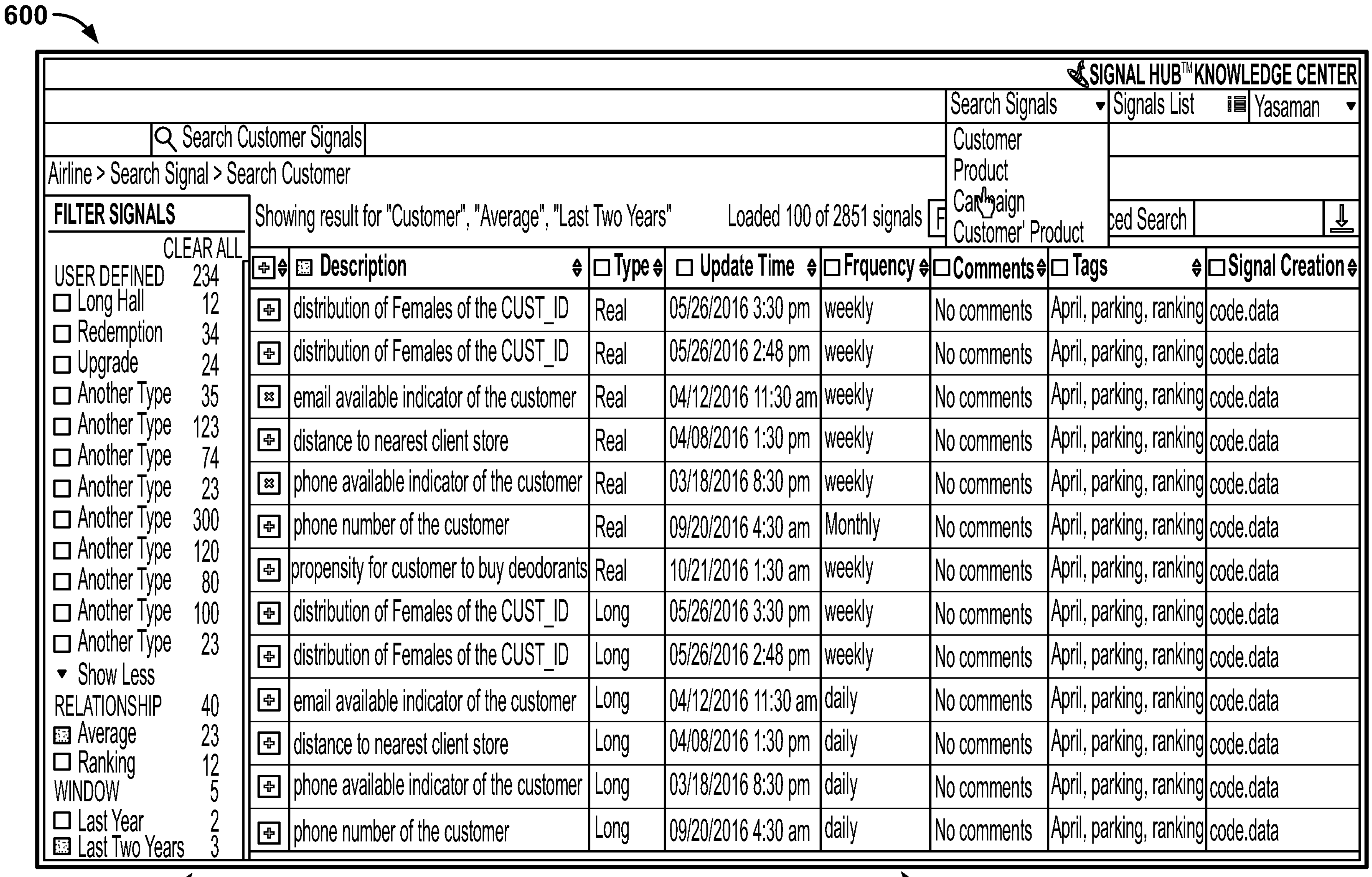
4 objects &subjects | 20 signal sets | 2851 signals

Loaded 100 of 2851 signals

Signal	Description	Tags	Signal Set	Type	Function
bkg_avg_pax_ 2y_per_cm	customer | avg # of passengers per trip customer travelled with | last 2 years	average, count, flight, single_period, time_window	party_level.signal	Common:Real	day2y_ago = data 04... See More
bkg_avg_price_ paid_2y_per_cm	customer | avg flight price per trip across all trips| last 2 years	average, flight, single_period, spend, time_window	party_level.signal	Common:Real	day2y_ago = data 04... See More
bkg_avg_rev_ nett_per_cm	customer | avg net net revenue per trip across all trips | last 3 years	average, flight, net_net_revenue, single_period, time_window	party_level.signal	Common:Real	bkg_avg_ rev_net_ avg(bkg_ sum_rev
bkg_cnt_africa_ trip_1y_per_cm	customer | # of trips to Africa | last 1 year	count, flight, single_period, time_window	party_level.signal	Common:Long	day1y_ago = data 04... See More
bkg_cnt_autumn _leis_holid_per _cm	customer | # of leisure trips to fly drive holiday	business_leisure, count, destination_theme, flight, route, season,	party_level.signal	Common:Long	bkg_cnt_ autumn whe... See More

Showing 1 to 100 of 100 entries

FIG. 20A



Airline > Search Signal > Search Customer

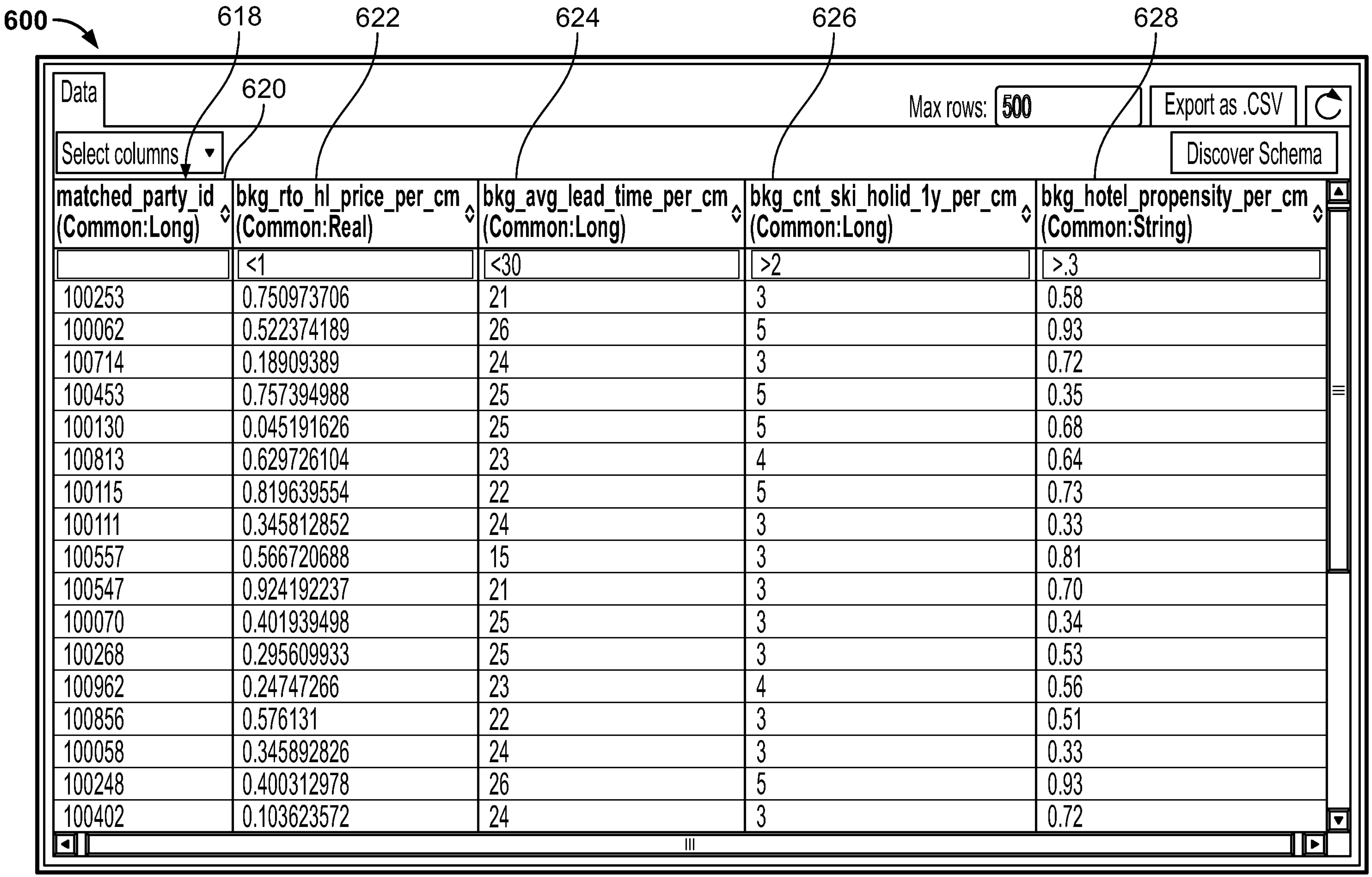
FILTER SIGNALS

USER DEFINED	234
□ Long Hall	12
□ Redemption	34
□ Upgrade	24
□ Another Type	35
□ Another Type	123
□ Another Type	74
□ Another Type	23
□ Another Type	300
□ Another Type	120
□ Another Type	80
□ Another Type	100
□ Another Type	23
▾ Show Less	
RELATIONSHIP	40
▨ Average	23
□ Ranking	12
WINDOW	5
□ Last Year	2
▨ Last Two Years	3

Showing result for "Customer", "Average", "Last Two Years" Loaded 100 of 2851 signals

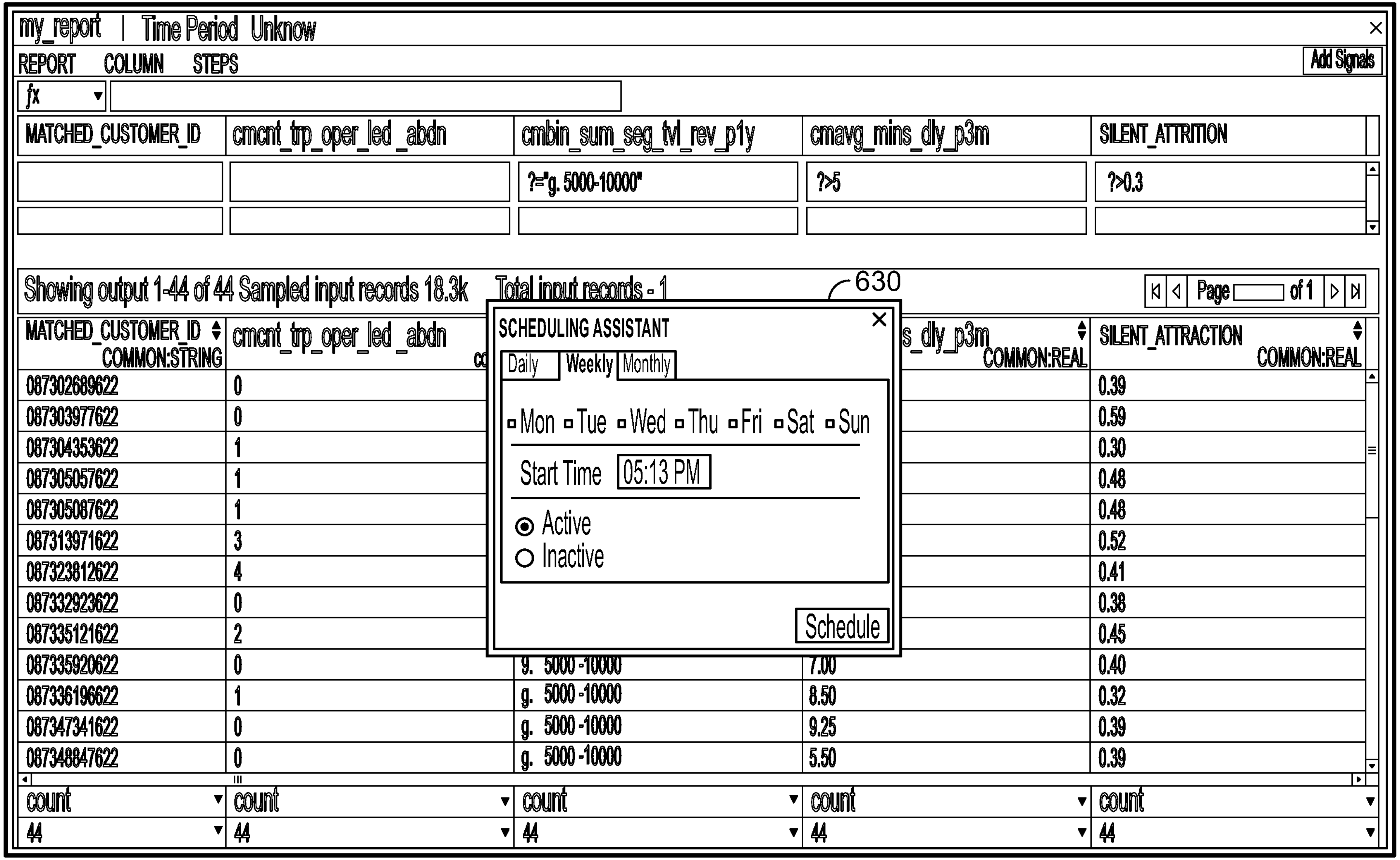
⊞	▨ Description	□ Type	□ Update Time	□ Frquency	□ Comments	□ Tags	□ Signal Creation
⊞	distribution of Females of the CUST_ID	Real	05/26/2016 3:30 pm	weekly	No comments	April, parking, ranking	code.data
⊞	distribution of Females of the CUST_ID	Real	05/26/2016 2:48 pm	weekly	No comments	April, parking, ranking	code.data
⊠	email available indicator of the customer	Real	04/12/2016 11:30 am	weekly	No comments	April, parking, ranking	code.data
⊞	distance to nearest client store	Real	04/08/2016 1:30 pm	weekly	No comments	April, parking, ranking	code.data
⊠	phone available indicator of the customer	Real	03/18/2016 8:30 pm	weekly	No comments	April, parking, ranking	code.data
⊞	phone number of the customer	Real	09/20/2016 4:30 am	Monthly	No comments	April, parking, ranking	code.data
⊞	propensity for customer to buy deodorants	Real	10/21/2016 1:30 am	weekly	No comments	April, parking, ranking	code.data
⊞	distribution of Females of the CUST_ID	Long	05/26/2016 3:30 pm	weekly	No comments	April, parking, ranking	code.data
⊞	distribution of Females of the CUST_ID	Long	05/26/2016 2:48 pm	weekly	No comments	April, parking, ranking	code.data
⊞	email available indicator of the customer	Long	04/12/2016 11:30 am	daily	No comments	April, parking, ranking	code.data
⊞	distance to nearest client store	Long	04/08/2016 1:30 pm	daily	No comments	April, parking, ranking	code.data
⊞	phone available indicator of the customer	Long	03/18/2016 8:30 pm	daily	No comments	April, parking, ranking	code.data
⊞	phone number of the customer	Long	09/20/2016 4:30 am	daily	No comments	April, parking, ranking	code.data

FIG. 20B



matched_party_id (Common:Long)	bkg_rto_hl_price_per_cm (Common:Real)	bkg_avg_lead_time_per_cm (Common:Long)	bkg_cnt_ski_holid_1y_per_cm (Common:Long)	bkg_hotel_propensity_per_cm (Common:String)
	<1	<30	>2	>.3
100253	0.750973706	21	3	0.58
100062	0.522374189	26	5	0.93
100714	0.18909389	24	3	0.72
100453	0.757394988	25	5	0.35
100130	0.045191626	25	5	0.68
100813	0.629726104	23	4	0.64
100115	0.819639554	22	5	0.73
100111	0.345812852	24	3	0.33
100557	0.566720688	15	3	0.81
100547	0.924192237	21	3	0.70
100070	0.401939498	25	3	0.34
100268	0.295609933	25	3	0.53
100962	0.24747266	23	4	0.56
100856	0.576131	22	3	0.51
100058	0.345892826	24	3	0.33
100248	0.400312978	26	5	0.93
100402	0.103623572	24	3	0.72

FIG. 21A



my_report | Time Period Unknow

REPORT COLUMN STEPS

MATCHED_CUSTOMER_ID	cmcnt_trp_oper_led _abdn	cmbin_sum_seg_tvl_rev_p1y	cmavg_mins_dly_p3m	SILENT_ATTRITION
		?="g. 5000-10000"	?>5	?>0.3

Showing output 1-44 of 44 Sampled input records 18.3k Total input records - 1

MATCHED_CUSTOMER_ID COMMON:STRING	cmcnt_trp_oper_led _abdn co		s_dly_p3m COMMON:REAL	SILENT_ATTRACTION COMMON:REAL
087302689622	0			0.39
087303977622	0			0.59
087304353622	1			0.30
087305057622	1			0.48
087305087622	1			0.48
087313971622	3			0.52
087323812622	4			0.41
087332923622	0			0.38
087335121622	2			0.45
087335920622	0	g. 5000 -10000	7.00	0.40
087336196622	1	g. 5000 -10000	8.50	0.32
087347341622	0	g. 5000 -10000	9.25	0.39
087348847622	0	g. 5000 -10000	5.50	0.39
count ▾	count ▾	count ▾	count ▾	count ▾
44 ▾	44 ▾	44 ▾	44 ▾	44 ▾

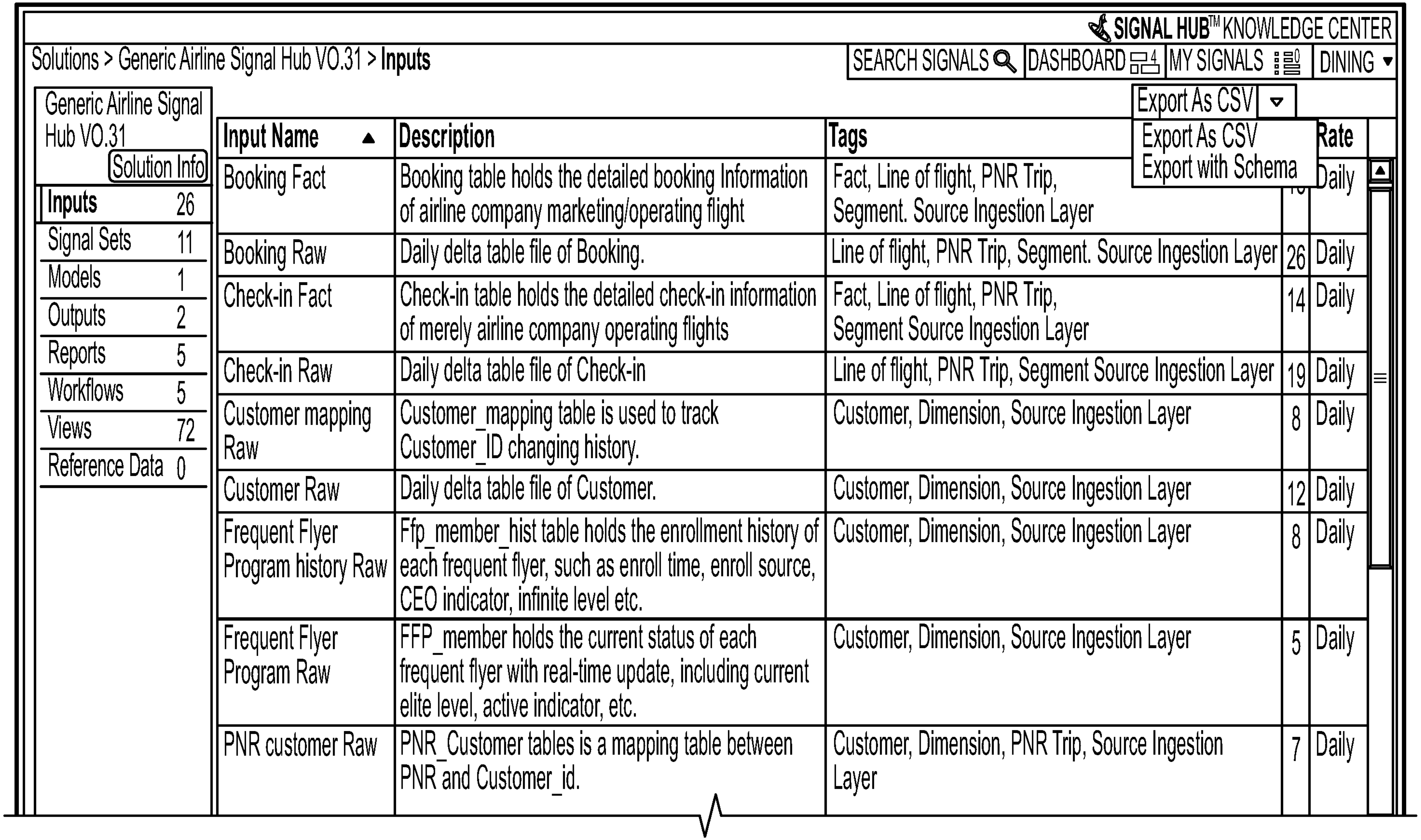
SCHEDULING ASSISTANT

□ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

Start Time 05:13 PM

● Active
○ Inactive

FIG. 21B



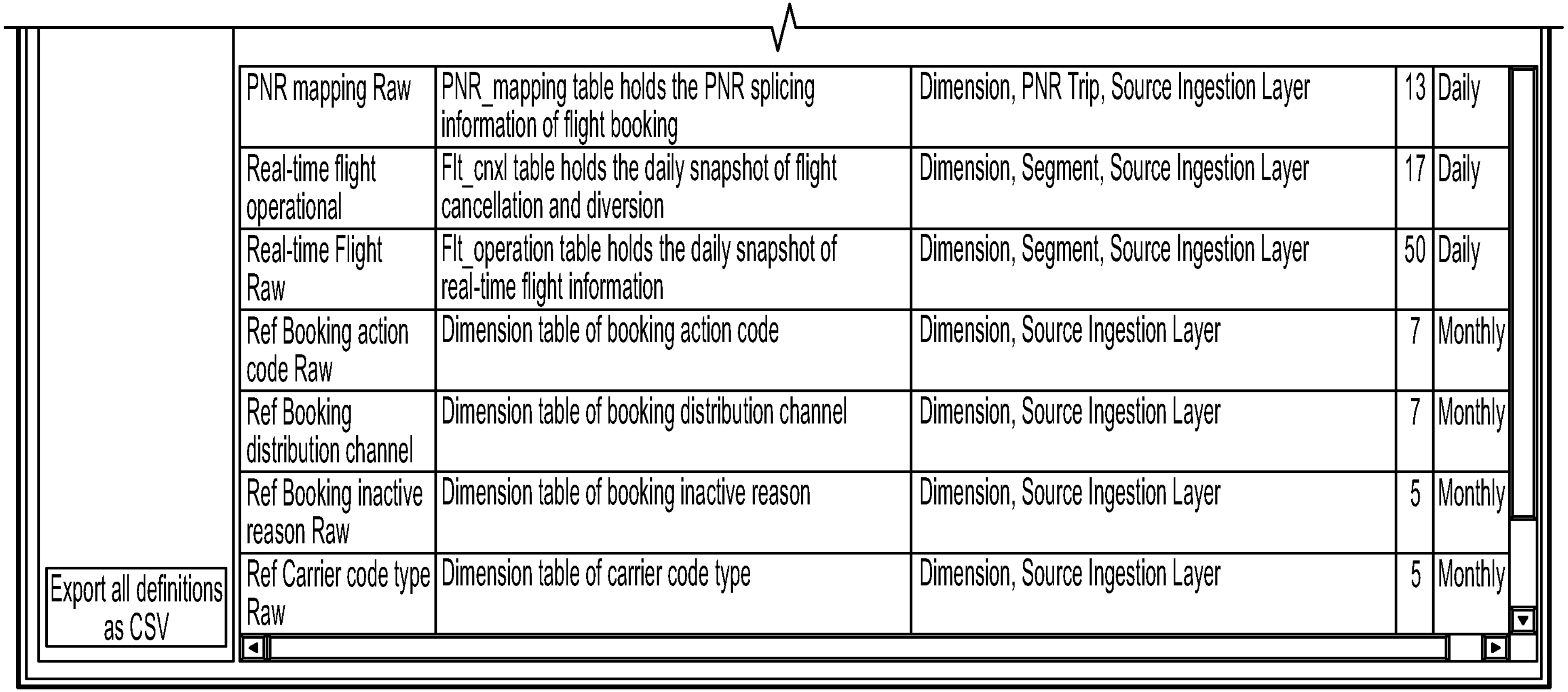
Solutions > Generic Airline Signal Hub VO.31 > **Inputs**

Generic Airline Signal Hub VO.31

Inputs	26
Signal Sets	11
Models	1
Outputs	2
Reports	5
Workflows	5
Views	72
Reference Data	0

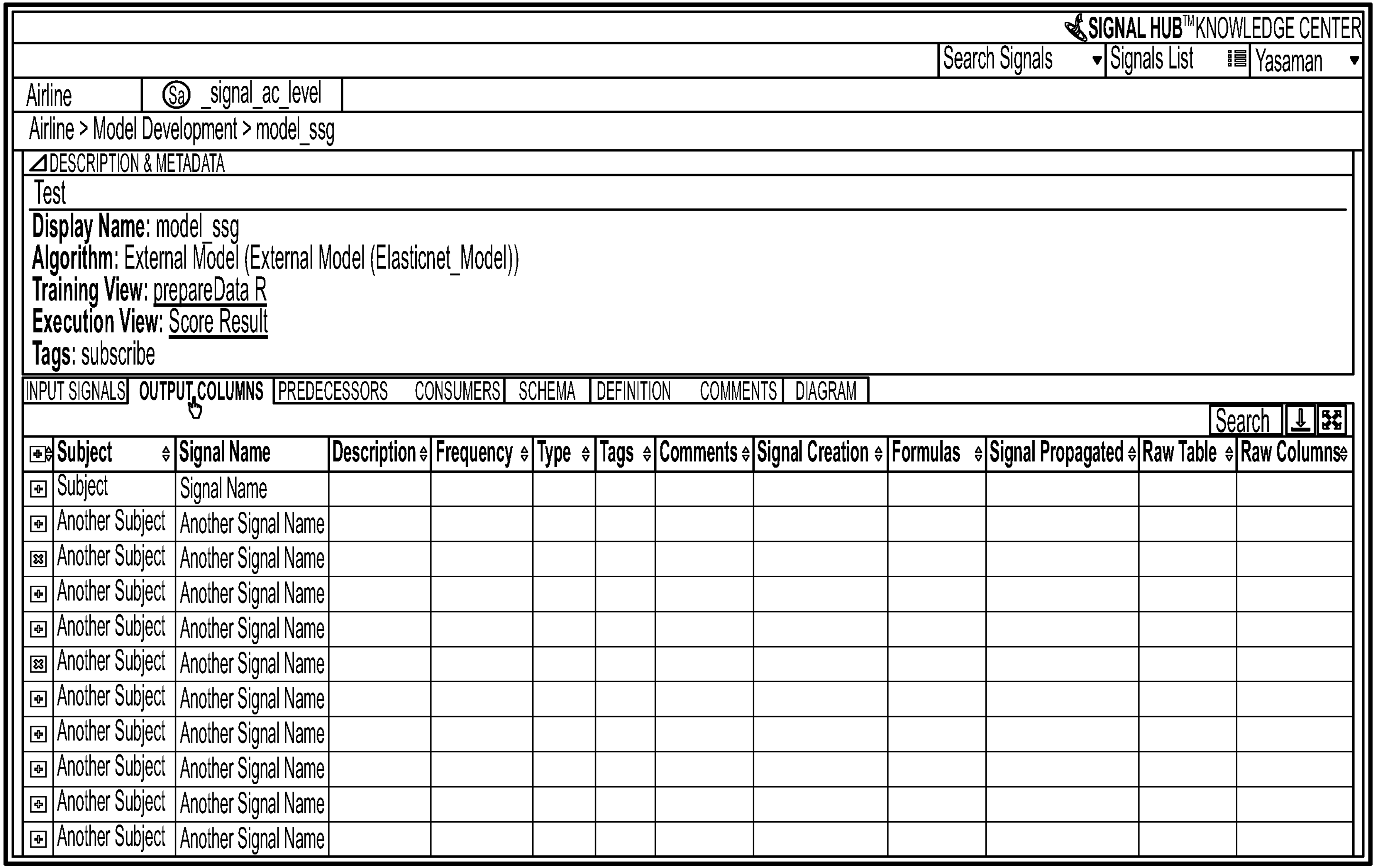
Input Name ▲	Description	Tags		Rate
Booking Fact	Booking table holds the detailed booking Information of airline company marketing/operating flight	Fact, Line of flight, PNR Trip, Segment. Source Ingestion Layer	[illegible]	Daily
Booking Raw	Daily delta table file of Booking.	Line of flight, PNR Trip, Segment. Source Ingestion Layer	26	Daily
Check-in Fact	Check-in table holds the detailed check-in information of merely airline company operating flights	Fact, Line of flight, PNR Trip, Segment Source Ingestion Layer	14	Daily
Check-in Raw	Daily delta table file of Check-in	Line of flight, PNR Trip, Segment Source Ingestion Layer	19	Daily
Customer mapping Raw	Customer_mapping table is used to track Customer_ID changing history.	Customer, Dimension, Source Ingestion Layer	8	Daily
Customer Raw	Daily delta table file of Customer.	Customer, Dimension, Source Ingestion Layer	12	Daily
Frequent Flyer Program history Raw	Ffp_member_hist table holds the enrollment history of each frequent flyer, such as enroll time, enroll source, CEO indicator, infinite level etc.	Customer, Dimension, Source Ingestion Layer	8	Daily
Frequent Flyer Program Raw	FFP_member holds the current status of each frequent flyer with real-time update, including current elite level, active indicator, etc.	Customer, Dimension, Source Ingestion Layer	5	Daily
PNR customer Raw	PNR_Customer tables is a mapping table between PNR and Customer_id.	Customer, Dimension, PNR Trip, Source Ingestion Layer	7	Daily

FIG. 21C



PNR mapping Raw	PNR_mapping table holds the PNR splicing information of flight booking	Dimension, PNR Trip, Source Ingestion Layer	13	Daily
Real-time flight operational	Flt_cnxl table holds the daily snapshot of flight cancellation and diversion	Dimension, Segment, Source Ingestion Layer	17	Daily
Real-time Flight Raw	Flt_operation table holds the daily snapshot of real-time flight information	Dimension, Segment, Source Ingestion Layer	50	Daily
Ref Booking action code Raw	Dimension table of booking action code	Dimension, Source Ingestion Layer	7	Monthly
Ref Booking distribution channel	Dimension table of booking distribution channel	Dimension, Source Ingestion Layer	7	Monthly
Ref Booking inactive reason Raw	Dimension table of booking inactive reason	Dimension, Source Ingestion Layer	5	Monthly
Ref Carrier code type Raw	Dimension table of carrier code type	Dimension, Source Ingestion Layer	5	Monthly

FIG. 21C (Cont.)



Airline > Model Development > model_ssg

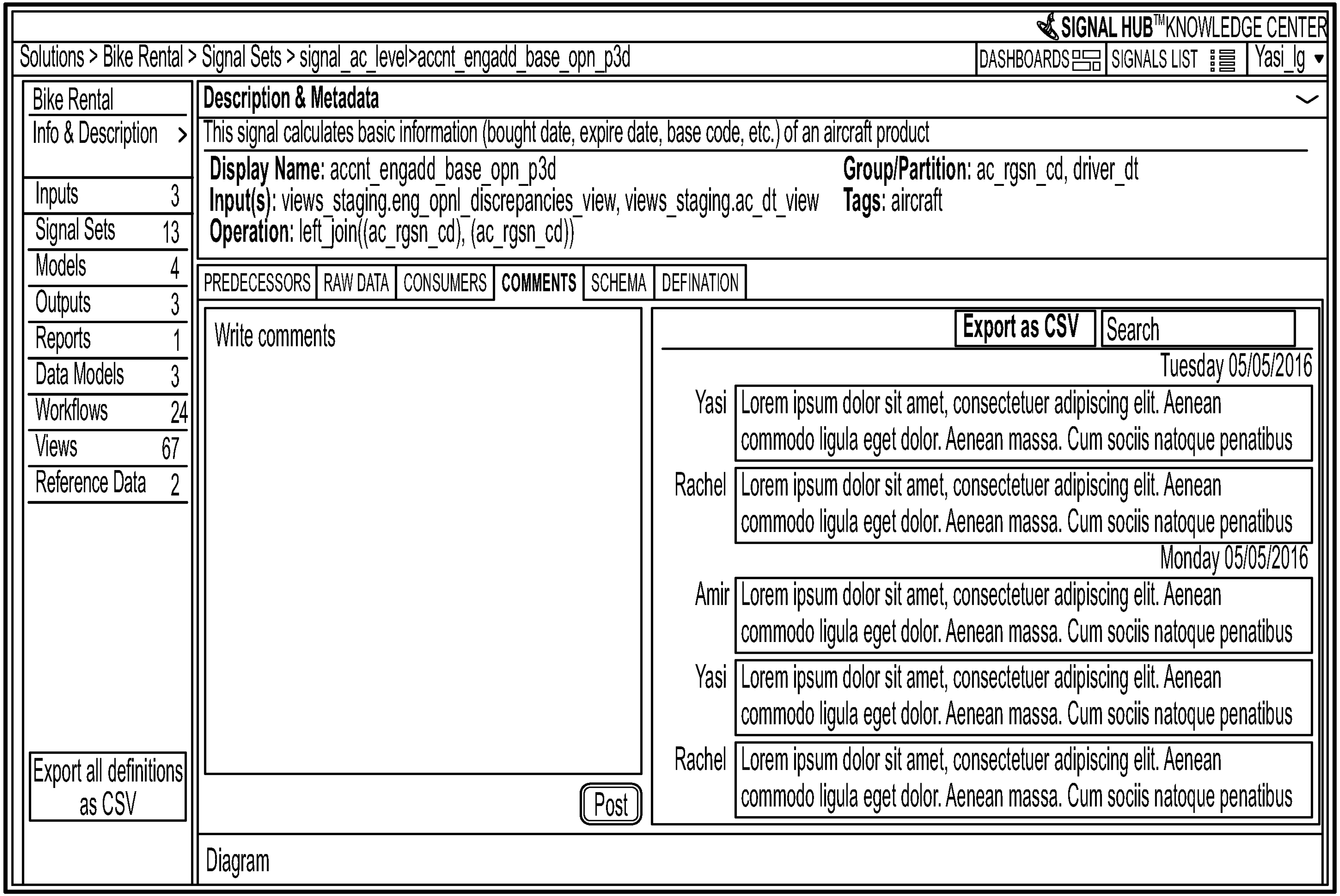
DESCRIPTION & METADATA

Test

Display Name: model_ssg
Algorithm: External Model (External Model (Elasticnet_Model))
Training View: prepareData R
Execution View: Score Result
Tags: subscribe

⊞	**Subject**	**Signal Name**	**Description**	**Frequency**	**Type**	**Tags**	**Comments**	**Signal Creation**	**Formulas**	**Signal Propagated**	**Raw Table**	**Raw Columns**
⊞	Subject	Signal Name										
⊞	Another Subject	Another Signal Name										
☒	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
☒	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										
⊞	Another Subject	Another Signal Name										

FIG. 21D



Description & Metadata

This signal calculates basic information (bought date, expire date, base code, etc.) of an aircraft product

Display Name: accnt_engadd_base_opn_p3d
Input(s): views_staging.eng_opnl_discrepancies_view, views_staging.ac_dt_view
Operation: left_join((ac_rgsn_cd), (ac_rgsn_cd))
Group/Partition: ac_rgsn_cd, driver_dt
Tags: aircraft

Tuesday 05/05/2016

Yasi: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus

Rachel: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus

Monday 05/05/2016

Amir: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus

Yasi: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus

Rachel: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus

FIG. 21E

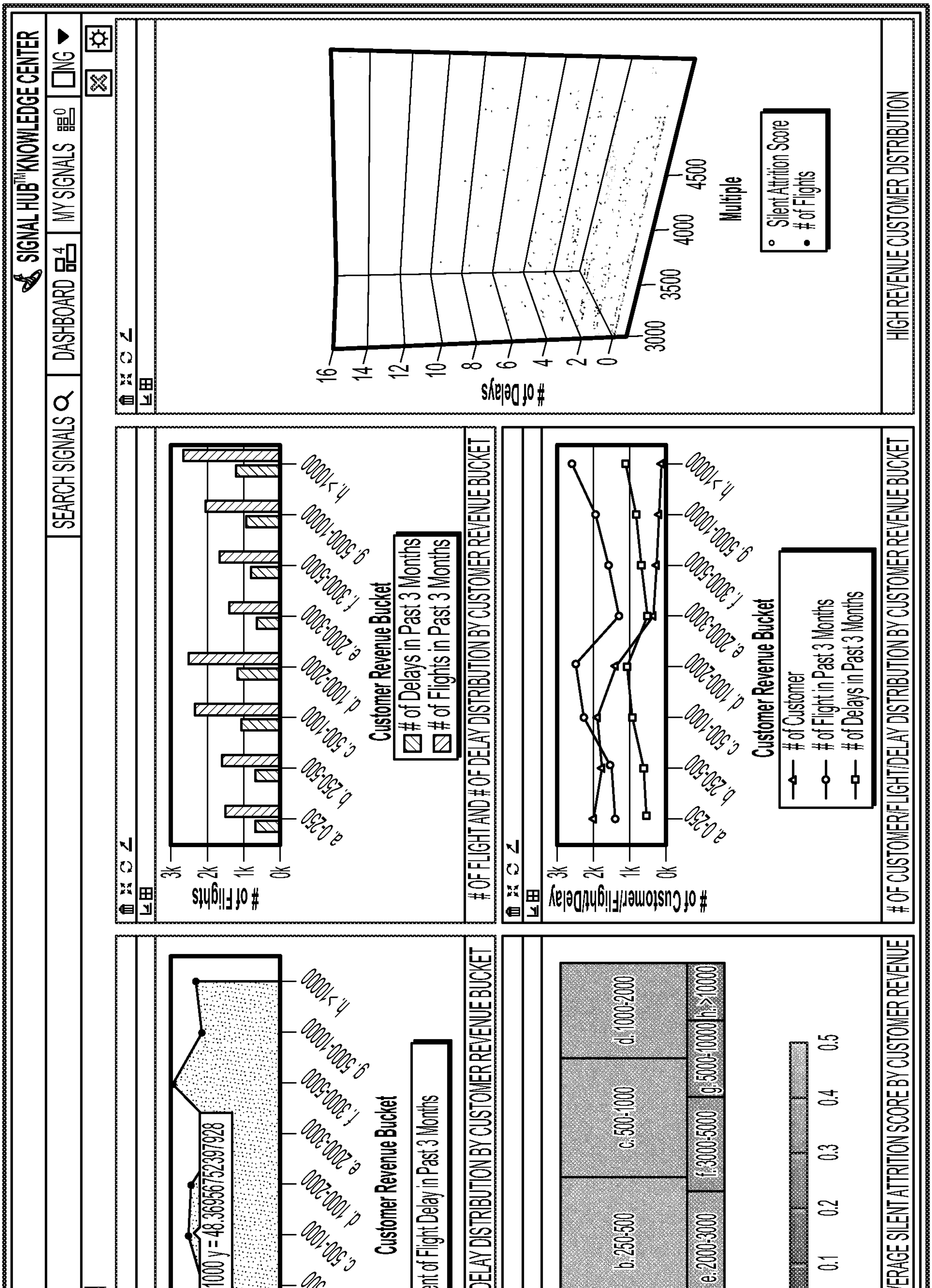
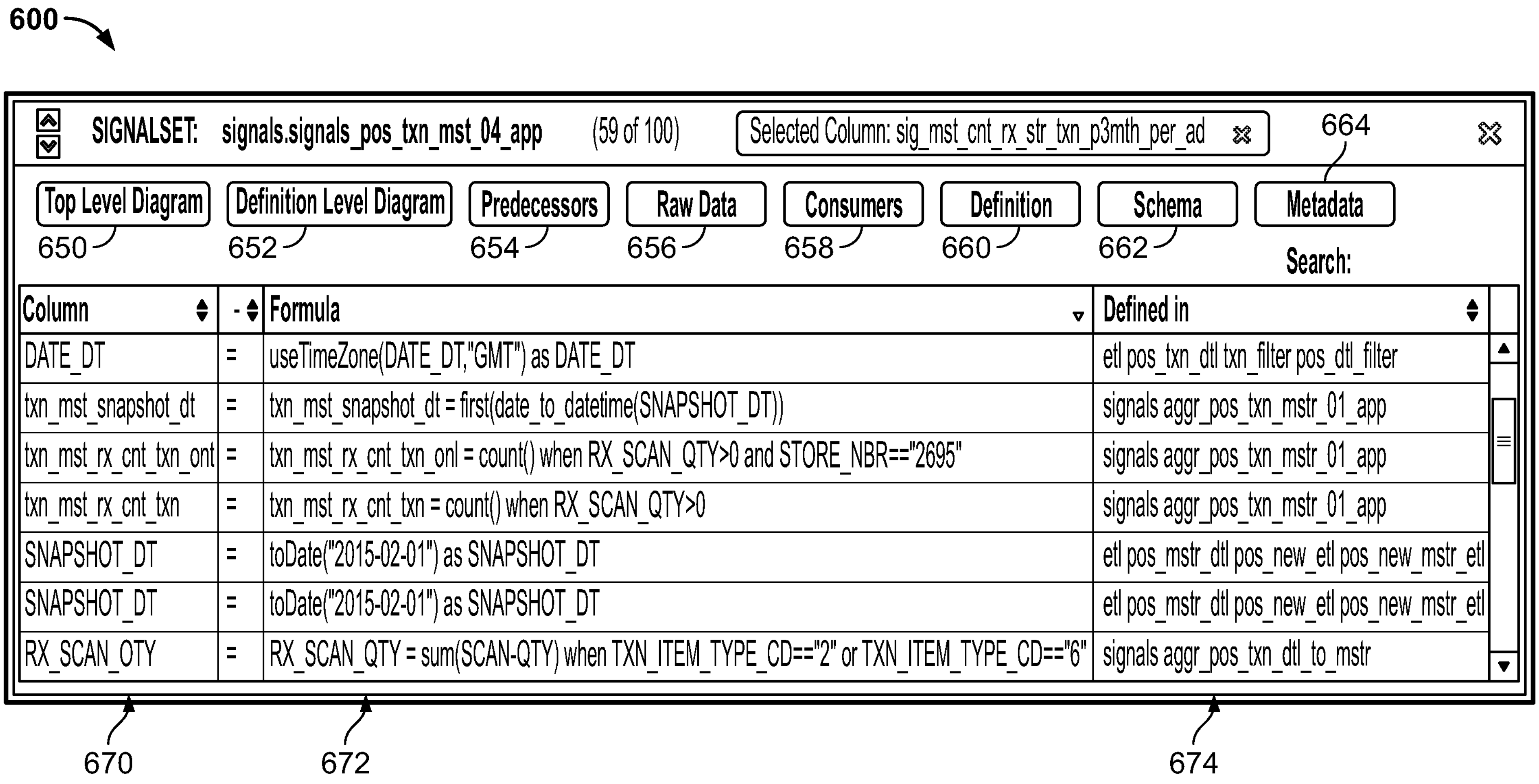

SIGNAL HUB™ KNOWLEDGE CENTER
SEARCH SIGNALS
DASHBOARD 4
MY SIGNALS
NG
LargeAirline_Dashboard
50
45
40
****
x: c. 500-1000 y = 48.36956752397928
a. 0-250
b. 250-500
c. 500-1000
d. 1000-2000
e. 2000-3000
f. 3000-5000
g. 5000-10000
h. >10000
Customer Revenue Bucket
Percent of Flight Delay in Past 3 Months
PERCENT OF FLIGHT DELAY DISTRIBUTION BY CUSTOMER REVENUE BUCKET
# of Flights
3k
2k
1k
0k
Customer Revenue Bucket
# of Delays in Past 3 Months
# of Flights in Past 3 Months
# OF FLIGHT AND # OF DELAY DISTRIBUTION BY CUSTOMER REVENUE BUCKET
# of Delays
16
14
12
10
8
6
4
2
0
3000
3500
4000
4500
Multiple
Silent Attrition Score
# of Flights
HIGH REVENUE CUSTOMER DISTRIBUTION
a. 0-250
b. 250-500
c. 500-1000
d. 1000-2000
e. 2000-3000
f. 3000-5000
g. 5000-10000
h. >10000
0
0.1
0.2
0.3
0.4
0.5
# OF CUSTOMER AND AVERAGE SILENT ATTRITION SCORE BY CUSTOMER REVENUE
# of Customer/Flight/Delay
3k
2k
1k
0k
Customer Revenue Bucket
# of Customer
# of Flight in Past 3 Months
# of Delays in Past 3 Months
# OF CUSTOMER/FLIGHT/DELAY DISTRIBUTION BY CUSTOMER REVENUE BUCKET

FIG. 21F



SIGNALSET: signals.signals_pos_txn_mst_04_app (59 of 100) Selected Column: sig_mst_cnt_rx_str_txn_p3mth_per_ad

Column	-	Formula	Defined in
DATE_DT	=	useTimeZone(DATE_DT,"GMT") as DATE_DT	etl pos_txn_dtl txn_filter pos_dtl_filter
txn_mst_snapshot_dt	=	txn_mst_snapshot_dt = first(date_to_datetime(SNAPSHOT_DT))	signals aggr_pos_txn_mstr_01_app
txn_mst_rx_cnt_txn_ont	=	txn_mst_rx_cnt_txn_onl = count() when RX_SCAN_QTY>0 and STORE_NBR=="2695"	signals aggr_pos_txn_mstr_01_app
txn_mst_rx_cnt_txn	=	txn_mst_rx_cnt_txn = count() when RX_SCAN_QTY>0	signals aggr_pos_txn_mstr_01_app
SNAPSHOT_DT	=	toDate("2015-02-01") as SNAPSHOT_DT	etl pos_mstr_dtl pos_new_etl pos_new_mstr_etl
SNAPSHOT_DT	=	toDate("2015-02-01") as SNAPSHOT_DT	etl pos_mstr_dtl pos_new_etl pos_new_mstr_etl
RX_SCAN_OTY	=	RX_SCAN_QTY = sum(SCAN-QTY) when TXN_ITEM_TYPE_CD=="2" or TXN_ITEM_TYPE_CD=="6"	signals aggr_pos_txn_dtl_to_mstr

FIG. 22

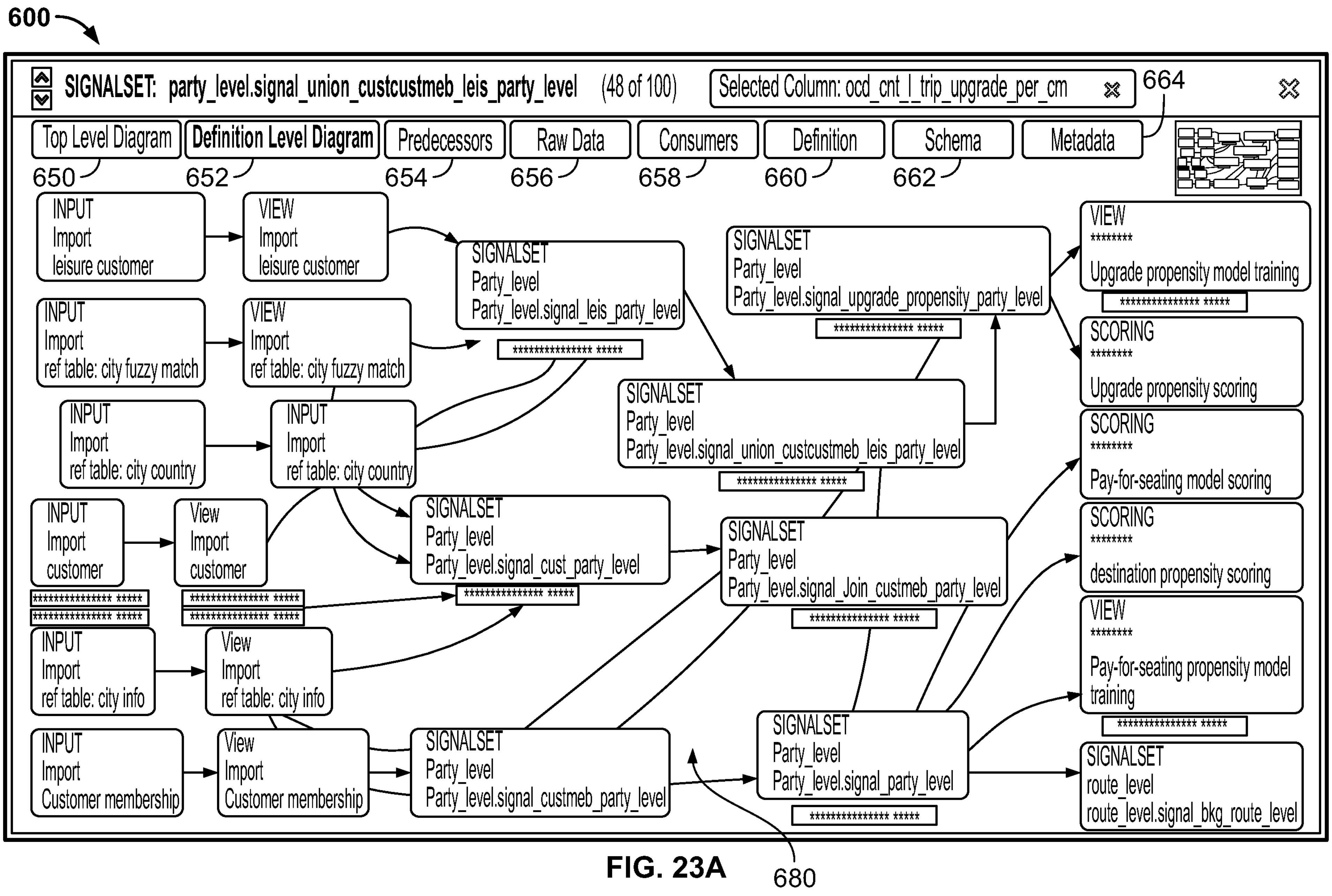
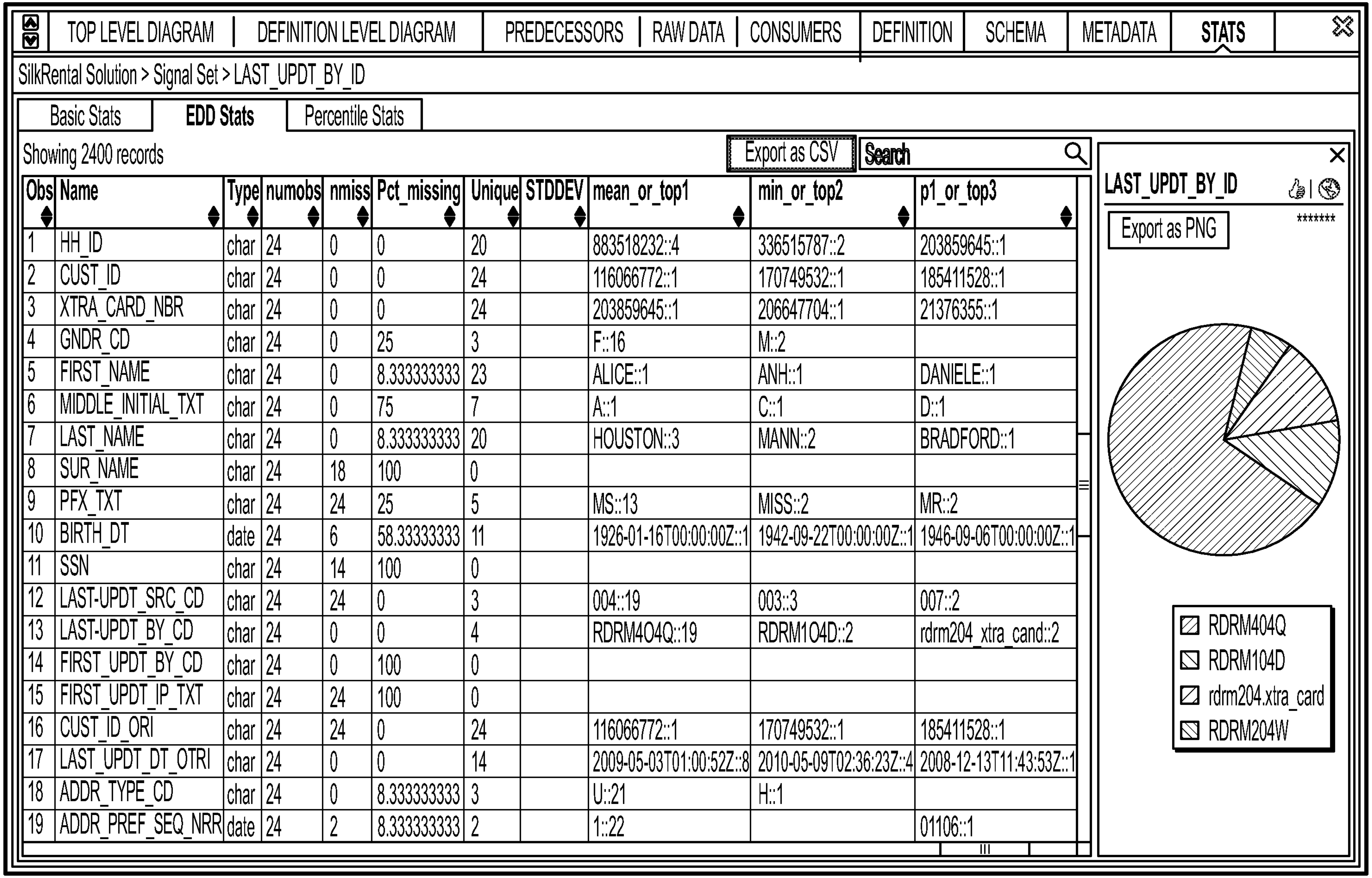
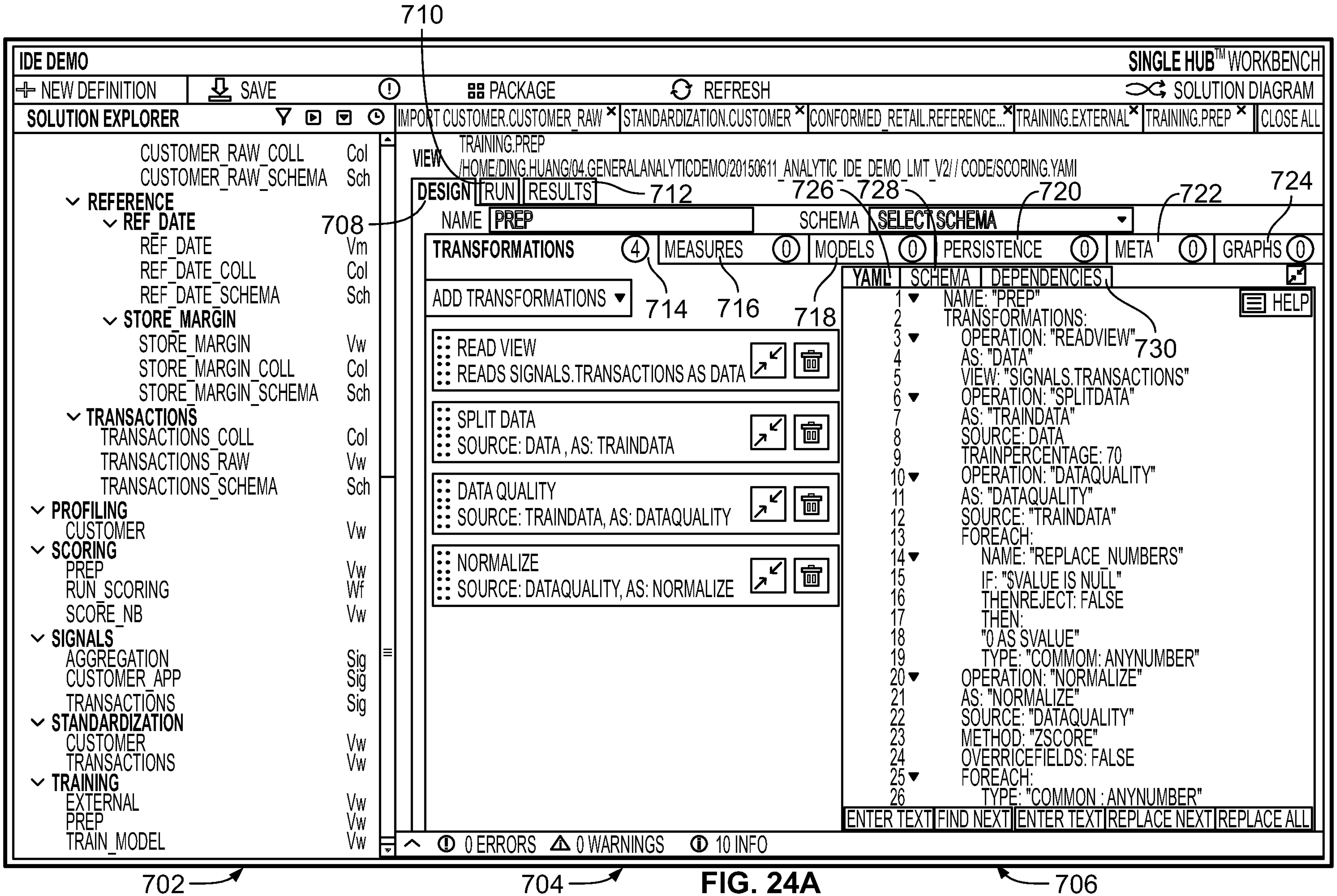

600
SIGNALSET: party_level.signal_union_custcustmeb_leis_party_level (48 of 100)
Selected Column: ocd_cnt_l_trip_upgrade_per_cm
664
Top Level Diagram
Definition Level Diagram
Predecessors
Raw Data
Consumers
Definition
Schema
Metadata
650
652
654
656
658
660
662
INPUT Import leisure customer
VIEW Import leisure customer
INPUT Import ref table: city fuzzy match
VIEW Import ref table: city fuzzy match
INPUT Import ref table: city country
INPUT Import ref table: city country
INPUT Import customer
View Import customer
INPUT Import ref table: city info
View Import ref table: city info
INPUT Import Customer membership
View Import Customer membership
SIGNALSET Party_level Party_level.signal_leis_party_level
SIGNALSET Party_level Party_level.signal_cust_party_level
SIGNALSET Party_level Party_level.signal_custmeb_party_level
SIGNALSET Party_level Party_level.signal_union_custcustmeb_leis_party_level
SIGNALSET Party_level Party_level.signal_Join_custmeb_party_level
SIGNALSET Party_level Party_level.signal_party_level
SIGNALSET Party_level Party_level.signal_upgrade_propensity_party_level
VIEW ******** Upgrade propensity model training
SCORING ******** Upgrade propensity scoring
SCORING ******** Pay-for-seating model scoring
SCORING ******** destination propensity scoring
VIEW ******** Pay-for-seating propensity model training
SIGNALSET route_level route_level.signal_bkg_route_level
*************** *****
680

FIG. 23A



Obs	Name	Type	numobs	nmiss	Pct_missing	Unique	STDDEV	mean_or_top1	min_or_top2	p1_or_top3
1	HH_ID	char	24	0	0	20		883518232::4	336515787::2	203859645::1
2	CUST_ID	char	24	0	0	24		116066772::1	170749532::1	185411528::1
3	XTRA_CARD_NBR	char	24	0	0	24		203859645::1	206647704::1	21376355::1
4	GNDR_CD	char	24	0	25	3		F::16	M::2	
5	FIRST_NAME	char	24	0	8.333333333	23		ALICE::1	ANH::1	DANIELE::1
6	MIDDLE_INITIAL_TXT	char	24	0	75	7		A::1	C::1	D::1
7	LAST_NAME	char	24	0	8.333333333	20		HOUSTON::3	MANN::2	BRADFORD::1
8	SUR_NAME	char	24	18	100	0				
9	PFX_TXT	char	24	24	25	5		MS::13	MISS::2	MR::2
10	BIRTH_DT	date	24	6	58.33333333	11		1926-01-16T00:00:00Z::1	1942-09-22T00:00:00Z::1	1946-09-06T00:00:00Z::1
11	SSN	char	24	14	100	0				
12	LAST-UPDT_SRC_CD	char	24	24	0	3		004::19	003::3	007::2
13	LAST-UPDT_BY_CD	char	24	0	0	4		RDRM4O4Q::19	RDRM1O4D::2	rdrm204_xtra_cand::2
14	FIRST_UPDT_BY_CD	char	24	0	100	0				
15	FIRST_UPDT_IP_TXT	char	24	24	100	0				
16	CUST_ID_ORI	char	24	24	0	24		116066772::1	170749532::1	185411528::1
17	LAST_UPDT_DT_OTRI	char	24	0	0	14		2009-05-03T01:00:52Z::8	2010-05-09T02:36:23Z::4	2008-12-13T11:43:53Z::1
18	ADDR_TYPE_CD	char	24	0	8.333333333	3		U::21	H::1	
19	ADDR_PREF_SEQ_NRR	date	24	2	8.333333333	2		1::22		01106::1

FIG. 23B



IDE DEMO | SINGLE HUB™ WORKBENCH

+ NEW DEFINITION | SAVE | PACKAGE | REFRESH | SOLUTION DIAGRAM

SOLUTION EXPLORER

CUSTOMER_RAW_COLL	Col
CUSTOMER_RAW_SCHEMA	Sch
⌄ REFERENCE	
⌄ REF_DATE	
REF_DATE	Vm
REF_DATE_COLL	Col
REF_DATE_SCHEMA	Sch
⌄ STORE_MARGIN	
STORE_MARGIN	Vw
STORE_MARGIN_COLL	Col
STORE_MARGIN_SCHEMA	Sch
⌄ TRANSACTIONS	
TRANSACTIONS_COLL	Col
TRANSACTIONS_RAW	Vw
TRANSACTIONS_SCHEMA	Sch
⌄ PROFILING	
CUSTOMER	Vw
⌄ SCORING	
PREP	Vw
RUN_SCORING	Wf
SCORE_NB	Vw
⌄ SIGNALS	
AGGREGATION	Sig
CUSTOMER_APP	Sig
TRANSACTIONS	Sig
⌄ STANDARDIZATION	
CUSTOMER	Vw
TRANSACTIONS	Vw
⌄ TRAINING	
EXTERNAL	Vw
PREP	Vw
TRAIN_MODEL	Vw

IMPORT CUSTOMER.CUSTOMER_RAW × | STANDARDIZATION.CUSTOMER × | CONFORMED_RETAIL.REFERENCE... × | TRAINING.EXTERNAL × | TRAINING.PREP × | CLOSE ALL

VIEW TRAINING.PREP
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/ / CODE/SCORING.YAMI

DESIGN | RUN | RESULTS

NAME: PREP SCHEMA: SELECT SCHEMA ▾

TRANSFORMATIONS (4) | MEASURES (0) | MODELS (0) | PERSISTENCE (0) | META (0) | GRAPHS (0)

ADD TRANSFORMATIONS ▾

READ VIEW
READS SIGNALS.TRANSACTIONS AS DATA

SPLIT DATA
SOURCE: DATA , AS: TRAINDATA

DATA QUALITY
SOURCE: TRAINDATA, AS: DATAQUALITY

NORMALIZE
SOURCE: DATAQUALITY, AS: NORMALIZE

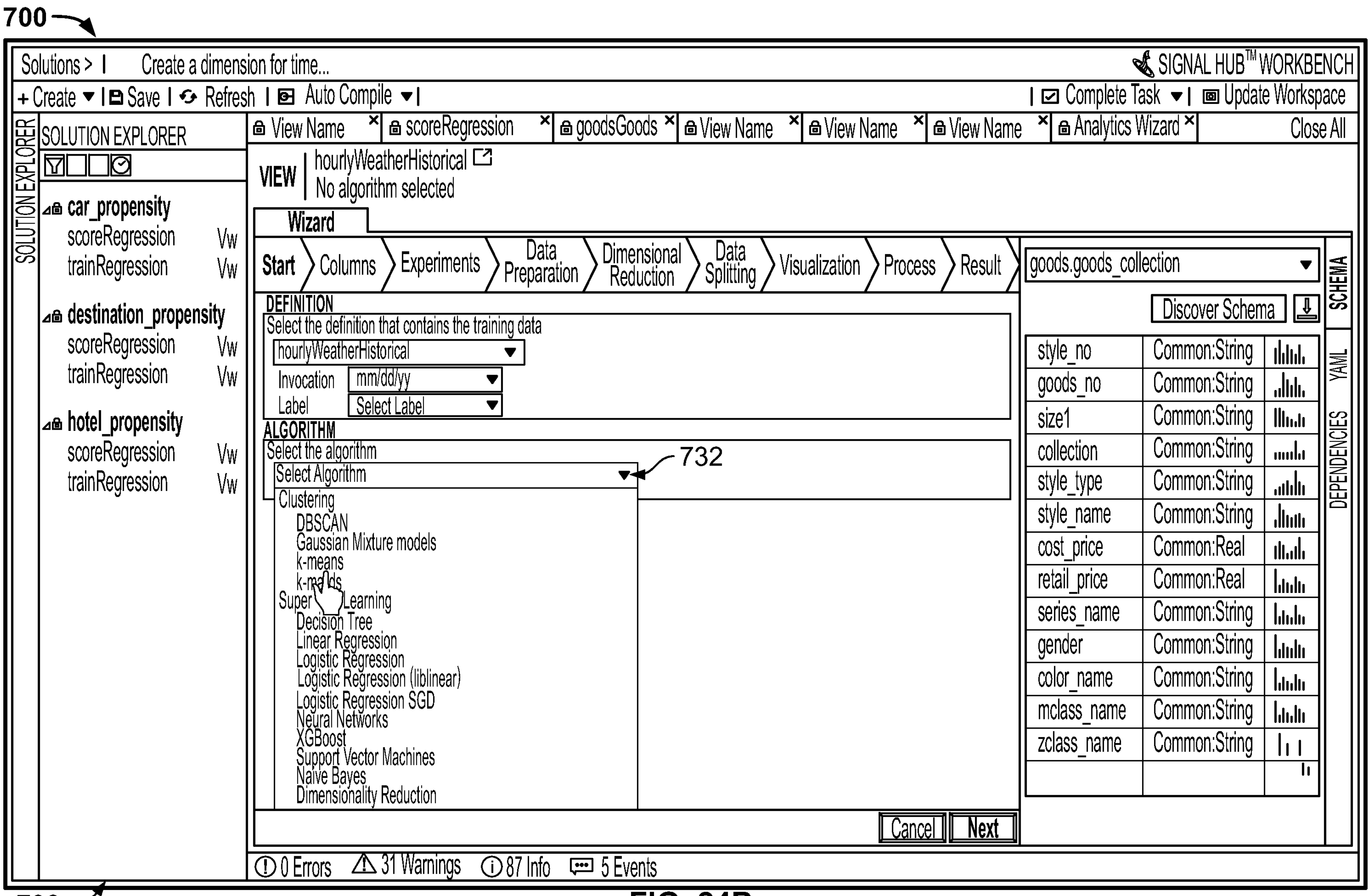
YAML | SCHEMA | DEPENDENCIES | HELP

```
▾ NAME: "PREP"
  TRANSFORMATIONS:
▾   OPERATION: "READVIEW"
    AS: "DATA"
    VIEW: "SIGNALS.TRANSACTIONS"
▾   OPERATION: "SPLITDATA"
    AS: "TRAINDATA"
    SOURCE: DATA
    TRAINPERCENTAGE: 70
▾   OPERATION: "DATAQUALITY"
    AS: "DATAQUALITY"
    SOURCE: "TRAINDATA"
    FOREACH:
▾     NAME: "REPLACE_NUMBERS"
      IF: "$VALUE IS NULL"
      THENREJECT: FALSE
      THEN:
      "0 AS SVALUE"
      TYPE: "COMMOM: ANYNUMBER"
▾   OPERATION: "NORMALIZE"
    AS: "NORMALIZE"
    SOURCE: "DATAQUALITY"
    METHOD: "ZSCORE"
    OVERRICEFIELDS: FALSE
▾   FOREACH:
      TYPE: "COMMON : ANYNUMBER"
```

ENTER TEXT | FIND NEXT | ENTER TEXT | REPLACE NEXT | REPLACE ALL

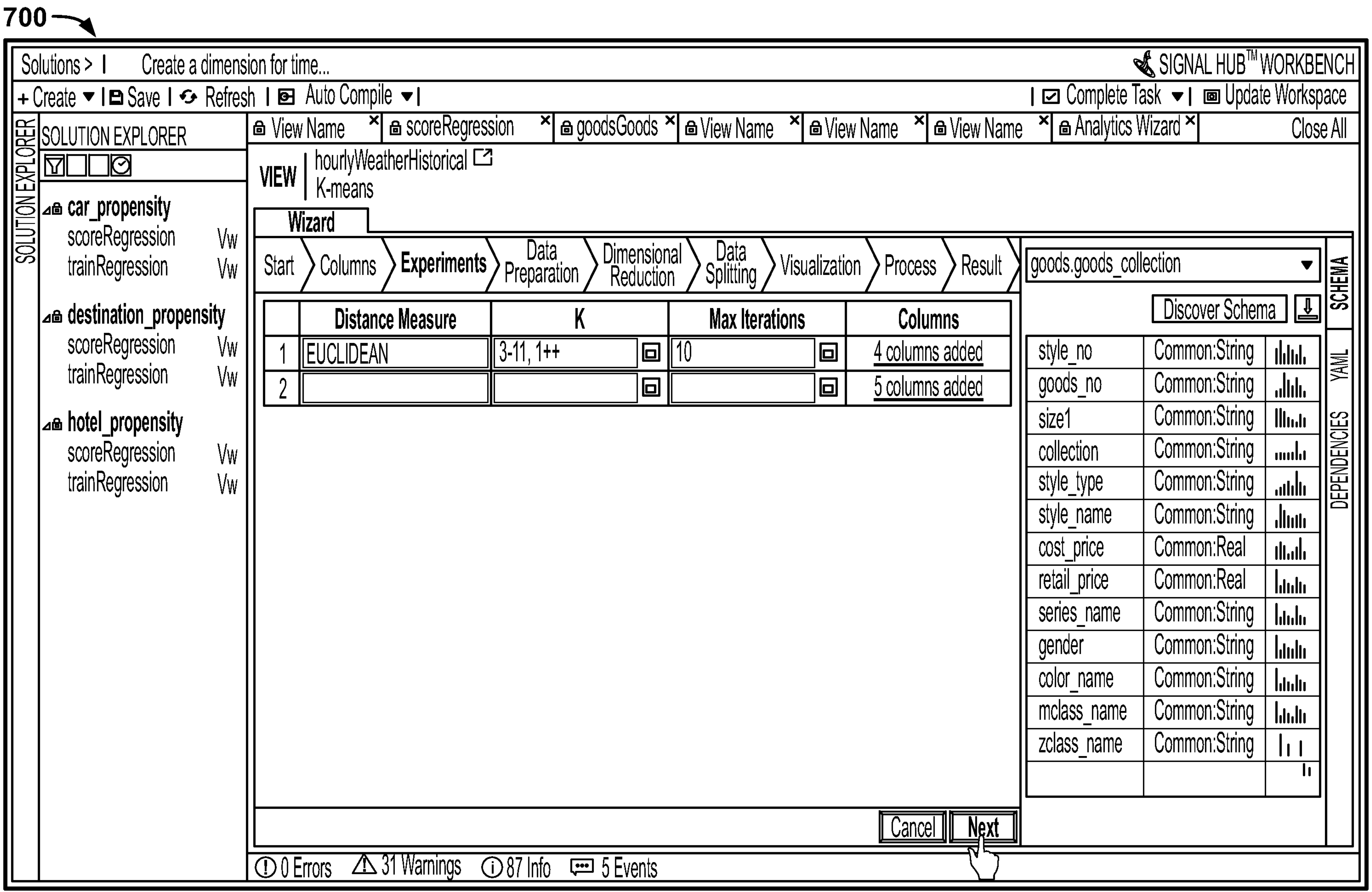
0 ERRORS | 0 WARNINGS | 10 INFO

FIG. 24A



style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

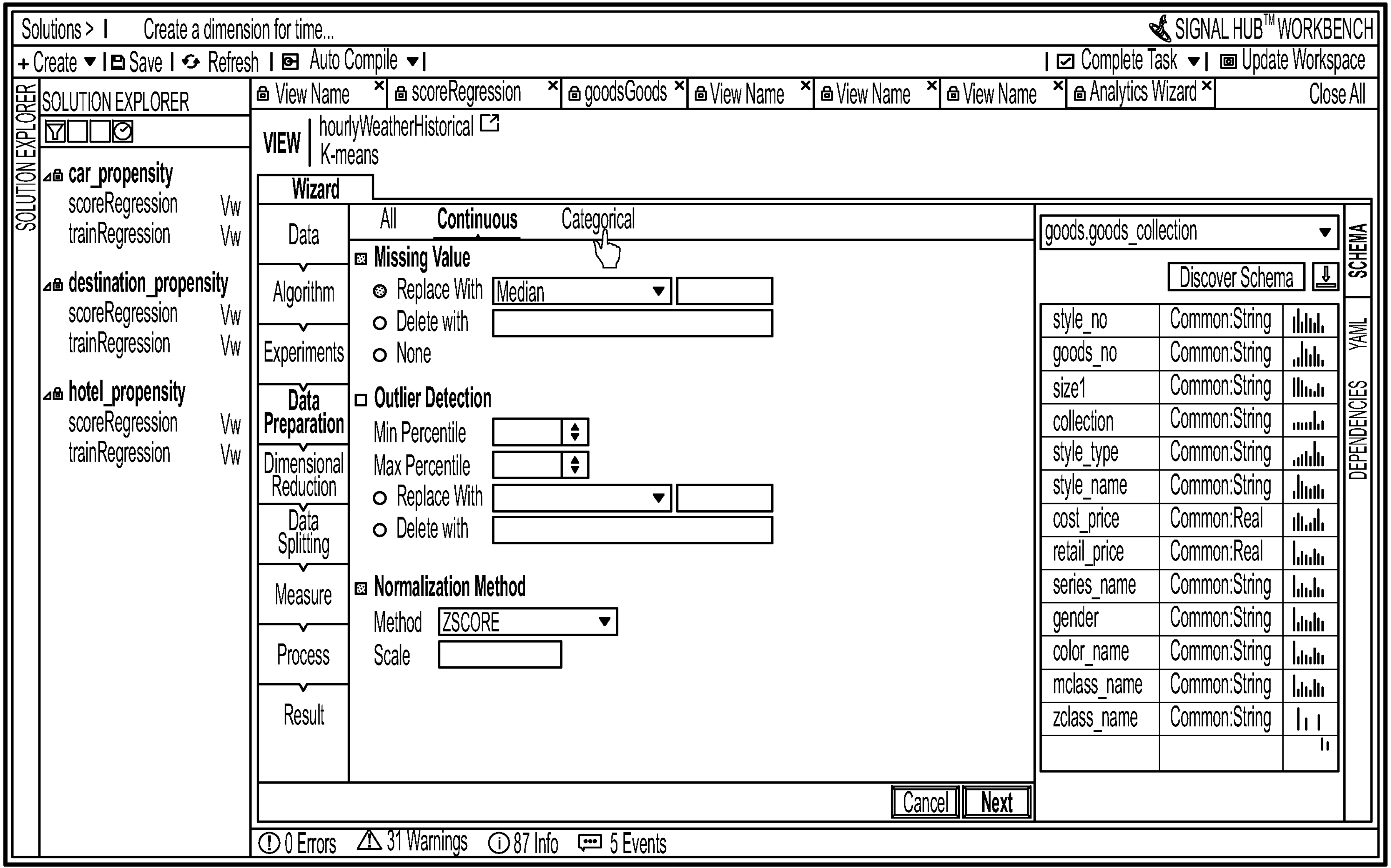
FIG. 24B



	Distance Measure	K	Max Iterations	Columns
1	EUCLIDEAN	3-11, 1++	10	4 columns added
2				5 columns added

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24C



☒ Missing Value

● Replace With Median ▾

○ Delete with

○ None

□ Outlier Detection

Min Percentile

Max Percentile

○ Replace With

○ Delete with

☒ Normalization Method

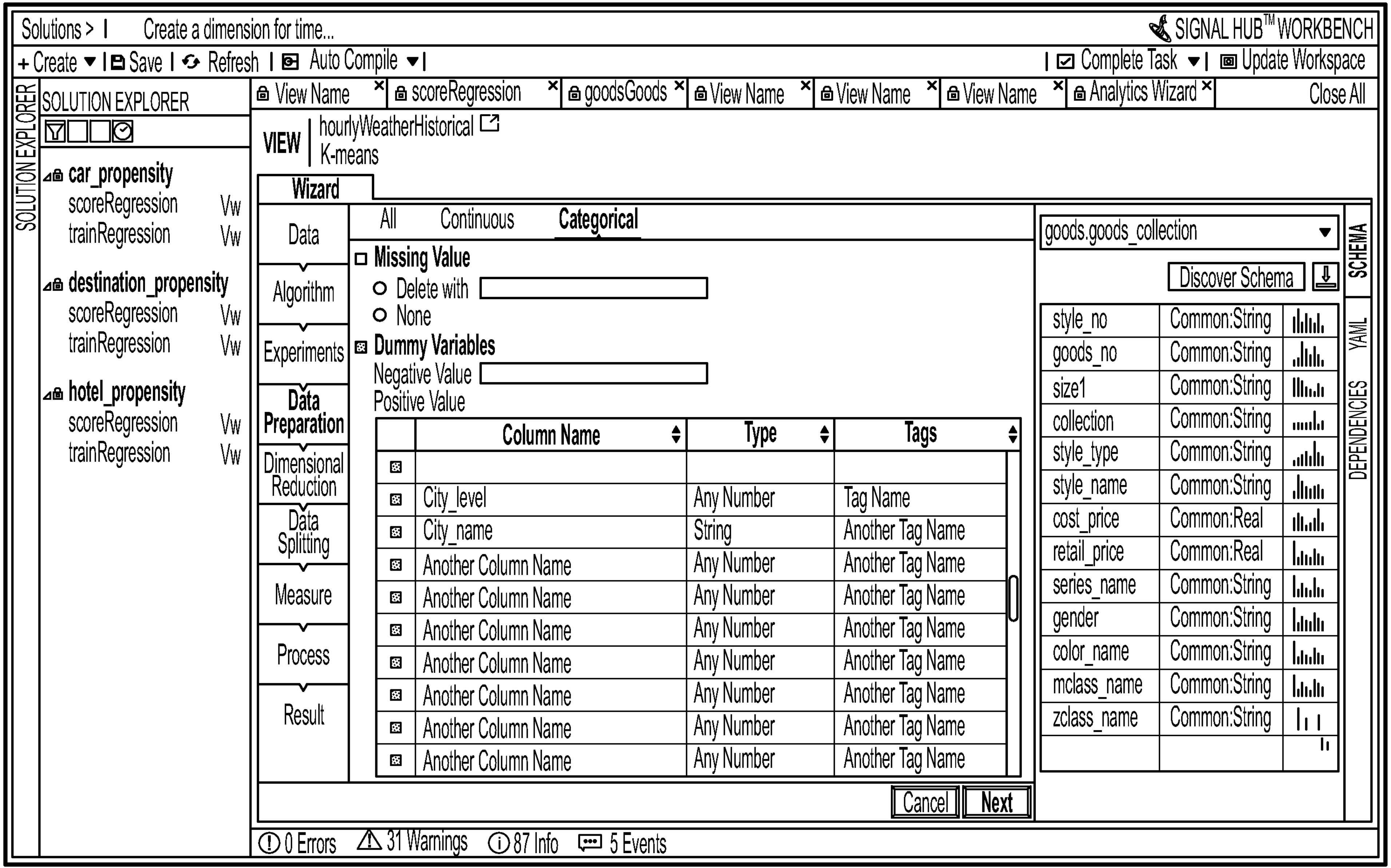
Method ZSCORE ▾

Scale

goods.goods_collection ▾

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24D



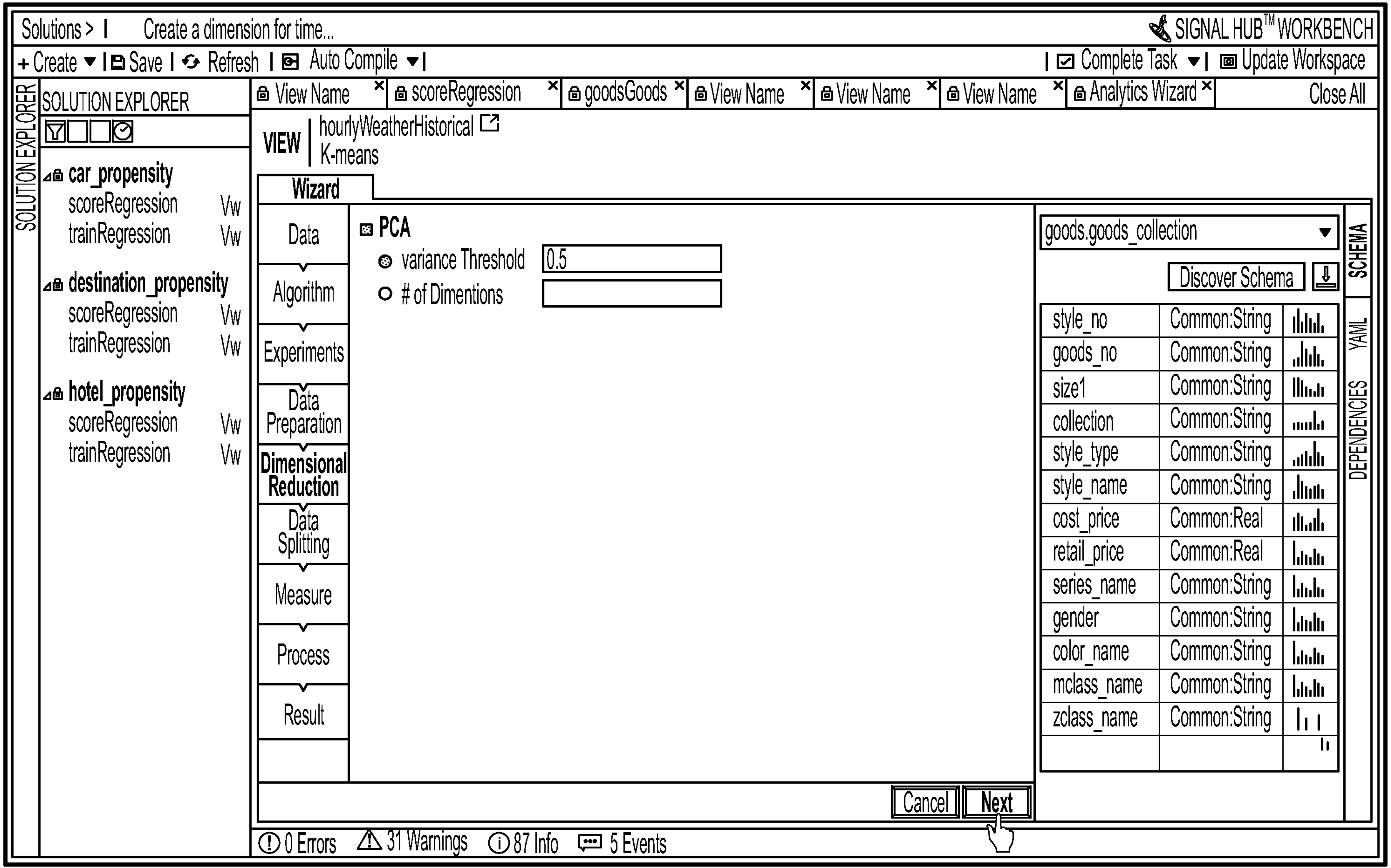
□ Missing Value
○ Delete with
○ None
☒ Dummy Variables
Negative Value
Positive Value

	Column Name	Type	Tags
☒			
☒	City_level	Any Number	Tag Name
☒	City_name	String	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name
☒	Another Column Name	Any Number	Another Tag Name

goods.goods_collection ▼

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24E



☒ PCA

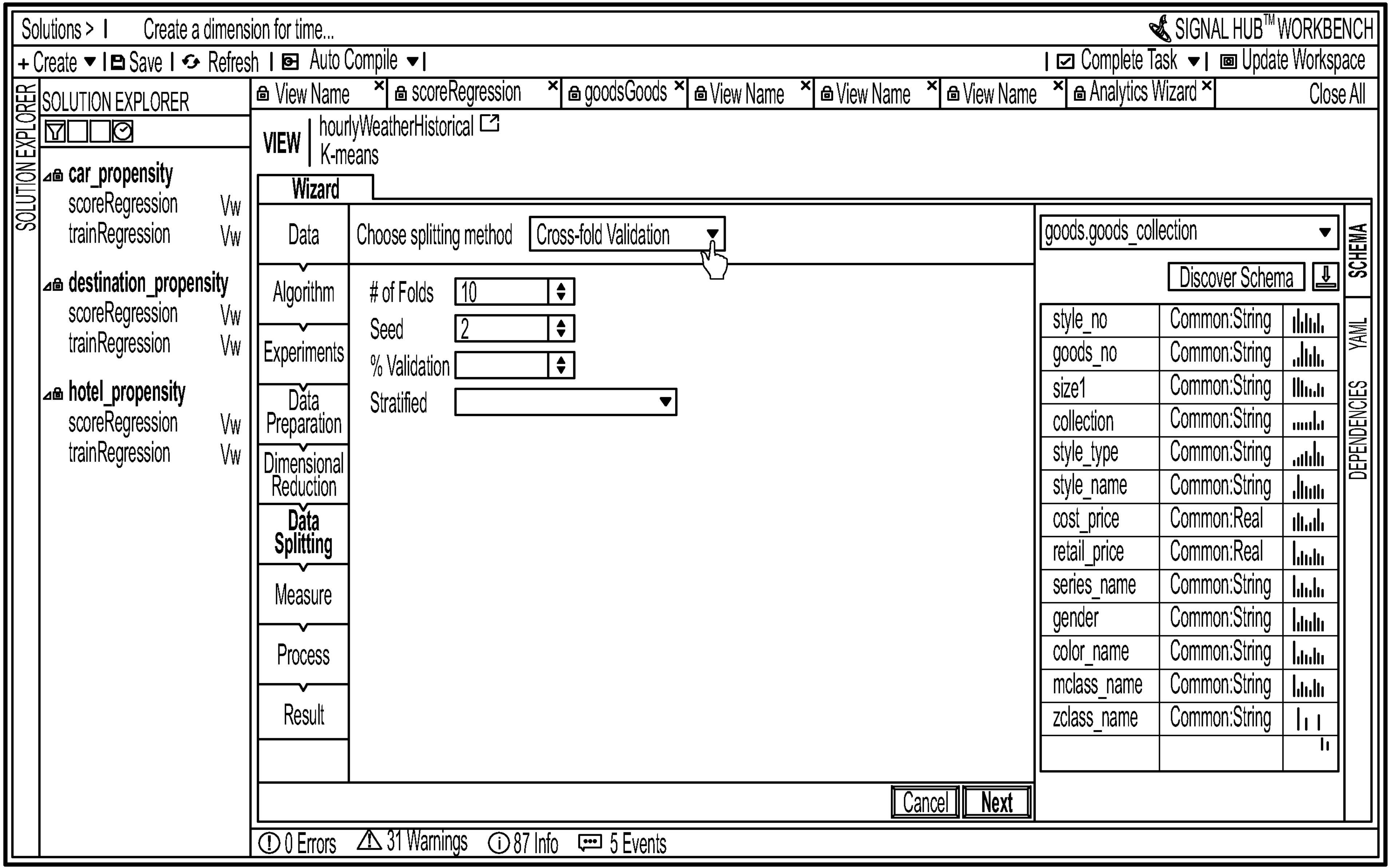
● variance Threshold 0.5

○ # of Dimentions

goods.goods_collection ▾

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24F



Choose splitting method: Cross-fold Validation ▾

# of Folds: 10

Seed: 2

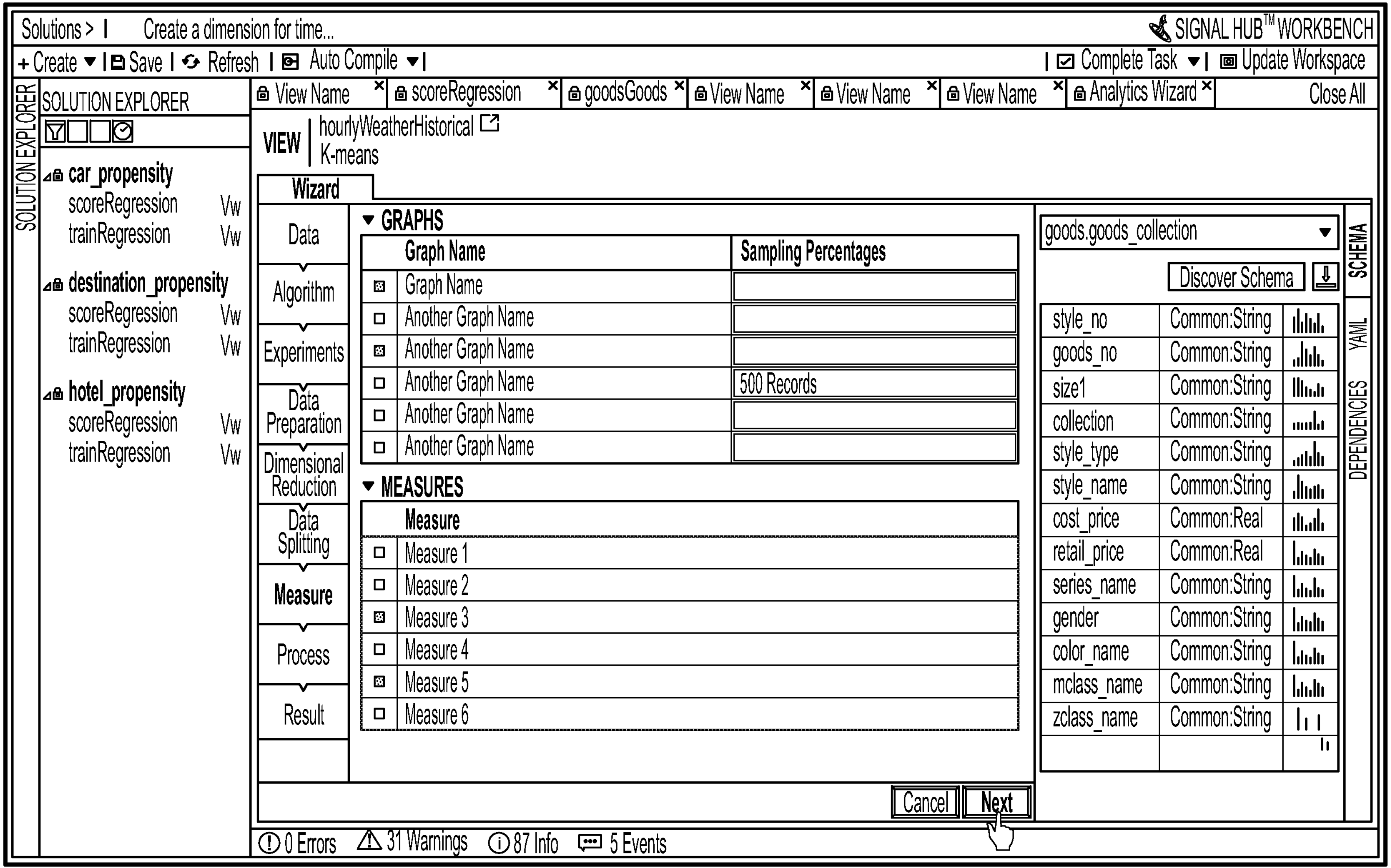
% Validation:

Stratified: ▾

goods.goods_collection ▾

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24G

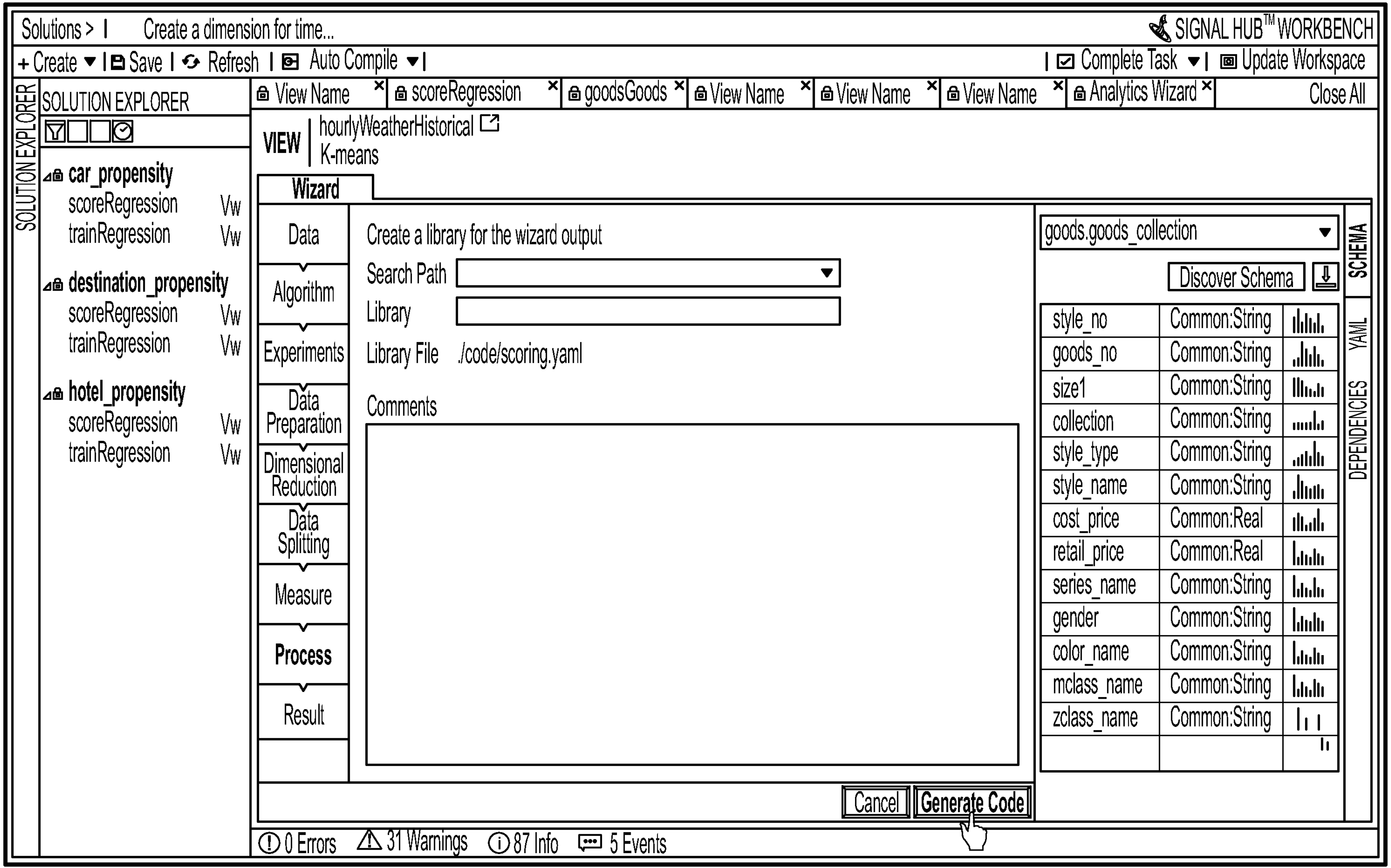


	Graph Name	Sampling Percentages
☒	Graph Name	
□	Another Graph Name	
☒	Another Graph Name	
□	Another Graph Name	500 Records
□	Another Graph Name	
□	Another Graph Name	

	Measure
□	Measure 1
□	Measure 2
☒	Measure 3
□	Measure 4
☒	Measure 5
□	Measure 6

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24H



Create a library for the wizard output

Search Path ▾

Library

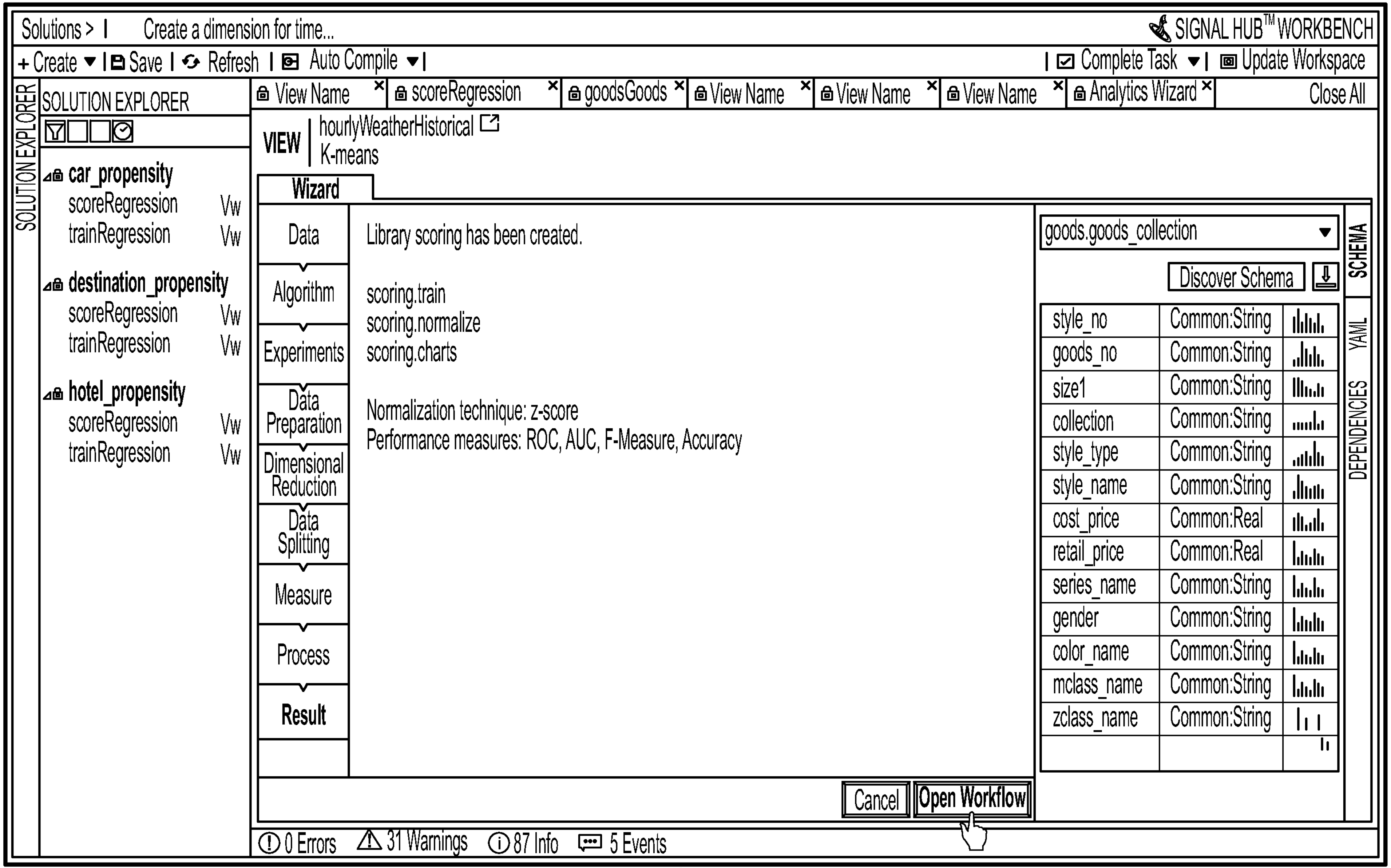
Library File ./code/scoring.yaml

Comments

goods.goods_collection ▾

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

FIG. 24I



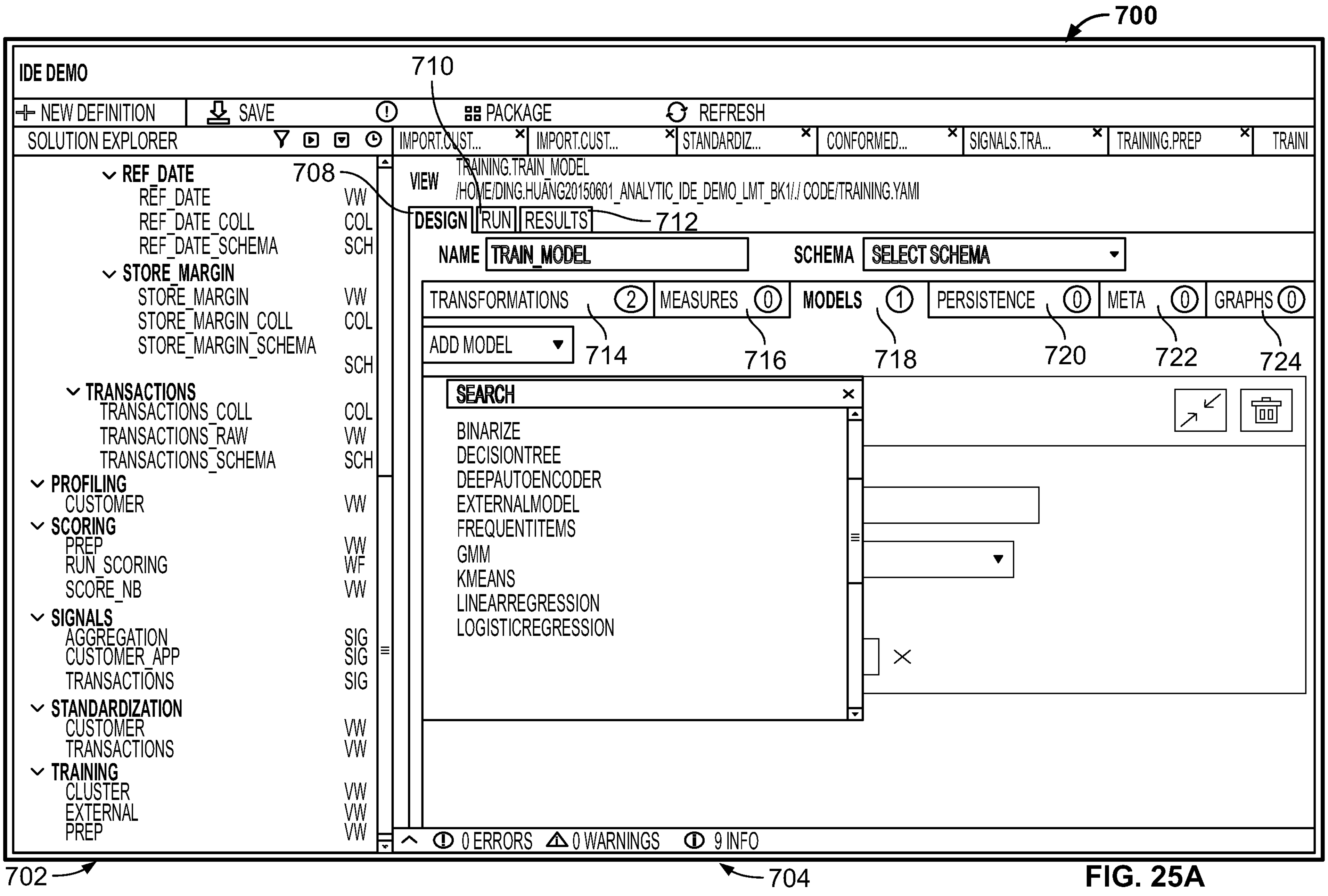
Library scoring has been created.

scoring.train
scoring.normalize
scoring.charts

Normalization technique: z-score
Performance measures: ROC, AUC, F-Measure, Accuracy

style_no	Common:String
goods_no	Common:String
size1	Common:String
collection	Common:String
style_type	Common:String
style_name	Common:String
cost_price	Common:Real
retail_price	Common:Real
series_name	Common:String
gender	Common:String
color_name	Common:String
mclass_name	Common:String
zclass_name	Common:String

FIG. 24J



IDE DEMO

NEW DEFINITION | SAVE | PACKAGE | REFRESH

SOLUTION EXPLORER

IMPORT.CUST... | IMPORT.CUST... | STANDARDIZ... | CONFORMED... | SIGNALS.TRA... | TRAINING.PREP | TRAINI

REF_DATE	
REF_DATE	VW
REF_DATE_COLL	COL
REF_DATE_SCHEMA	SCH
STORE_MARGIN	
STORE_MARGIN	VW
STORE_MARGIN_COLL	COL
STORE_MARGIN_SCHEMA	SCH
TRANSACTIONS	
TRANSACTIONS_COLL	COL
TRANSACTIONS_RAW	VW
TRANSACTIONS_SCHEMA	SCH
PROFILING	
CUSTOMER	VW
SCORING	
PREP	VW
RUN_SCORING	WF
SCORE_NB	VW
SIGNALS	
AGGREGATION	SIG
CUSTOMER_APP	SIG
TRANSACTIONS	SIG
STANDARDIZATION	
CUSTOMER	VW
TRANSACTIONS	VW
TRAINING	
CLUSTER	VW
EXTERNAL	VW
PREP	VW

VIEW TRAINING.TRAIN_MODEL
/HOME/DING.HUANG20150601_ANALYTIC_IDE_DEMO_LMT_BK1/./ CODE/TRAINING.YAMI

DESIGN | RUN | RESULTS

NAME TRAIN_MODEL

SCHEMA SELECT SCHEMA

TRANSFORMATIONS 2 | MEASURES 0 | MODELS 1 | PERSISTENCE 0 | META 0 | GRAPHS 0

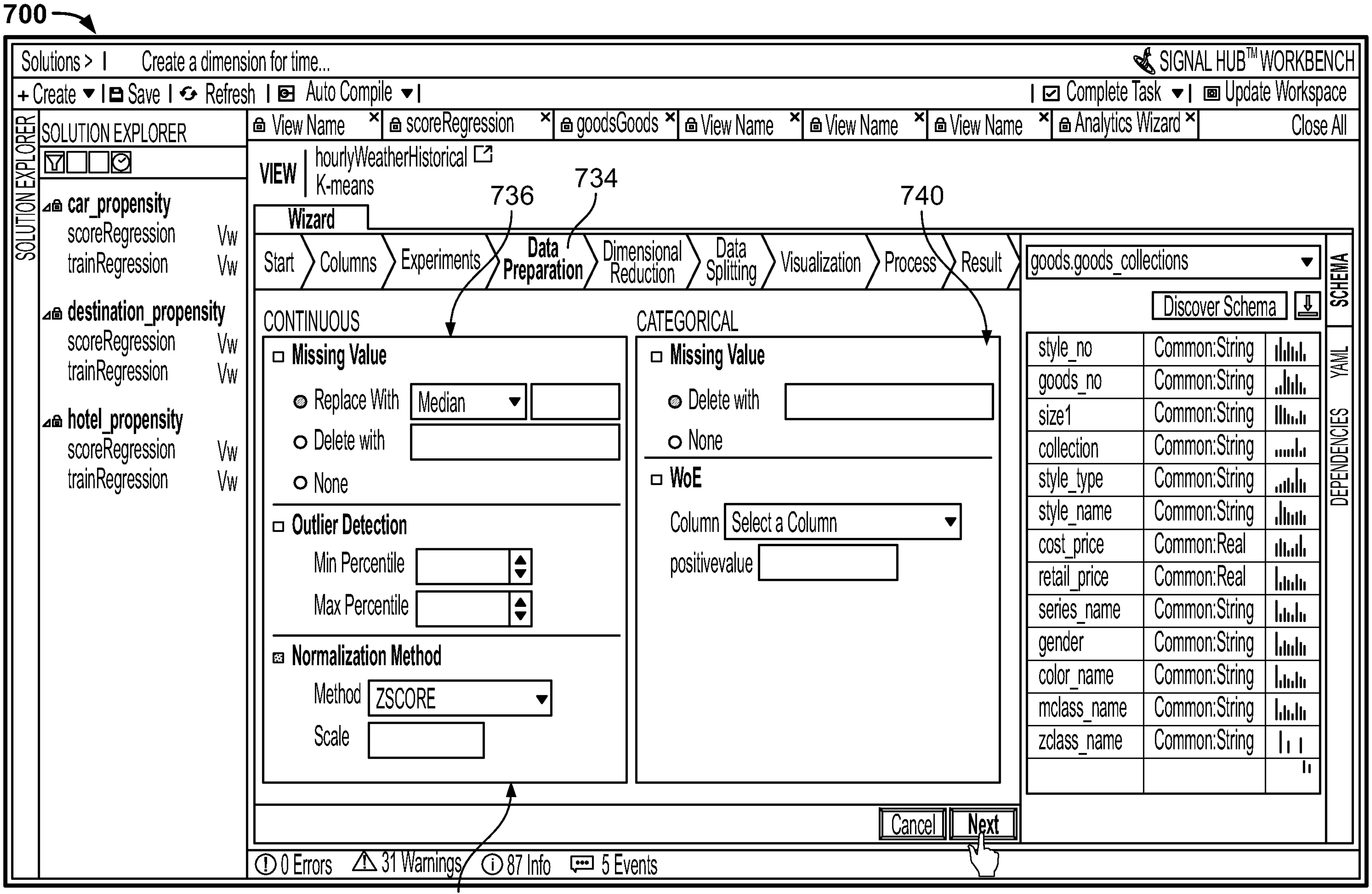
ADD MODEL

SEARCH

BINARIZE
DECISIONTREE
DEEPAUTOENCODER
EXTERNALMODEL
FREQUENTITEMS
GMM
KMEANS
LINEARREGRESSION
LOGISTICREGRESSION

0 ERRORS 0 WARNINGS 9 INFO

FIG. 25A



Solutions > | Create a dimension for time...

SIGNAL HUB™ WORKBENCH

+ Create ▾ | Save | Refresh | Auto Compile ▾ | | ☑ Complete Task ▾ | Update Workspace

View Name × | scoreRegression × | goodsGoods × | View Name × | View Name × | View Name × | Analytics Wizard × | Close All

SOLUTION EXPLORER

- car_propensity
  - scoreRegression Vw
  - trainRegression Vw
- destination_propensity
  - scoreRegression Vw
  - trainRegression Vw
- hotel_propensity
  - scoreRegression Vw
  - trainRegression Vw

VIEW | hourlyWeatherHistorical K-means

Wizard

Start > Columns > Experiments > **Data Preparation** > Dimensional Reduction > Data Splitting > Visualization > Process > Result

CONTINUOUS

□ **Missing Value**

● Replace With: Median ▾

○ Delete with

○ None

□ **Outlier Detection**

Min Percentile

Max Percentile

☒ **Normalization Method**

Method: ZSCORE ▾

Scale

CATEGORICAL

□ **Missing Value**

● Delete with

○ None

□ **WoE**

Column: Select a Column ▾

positivevalue

Cancel | Next

goods.goods_collections ▾

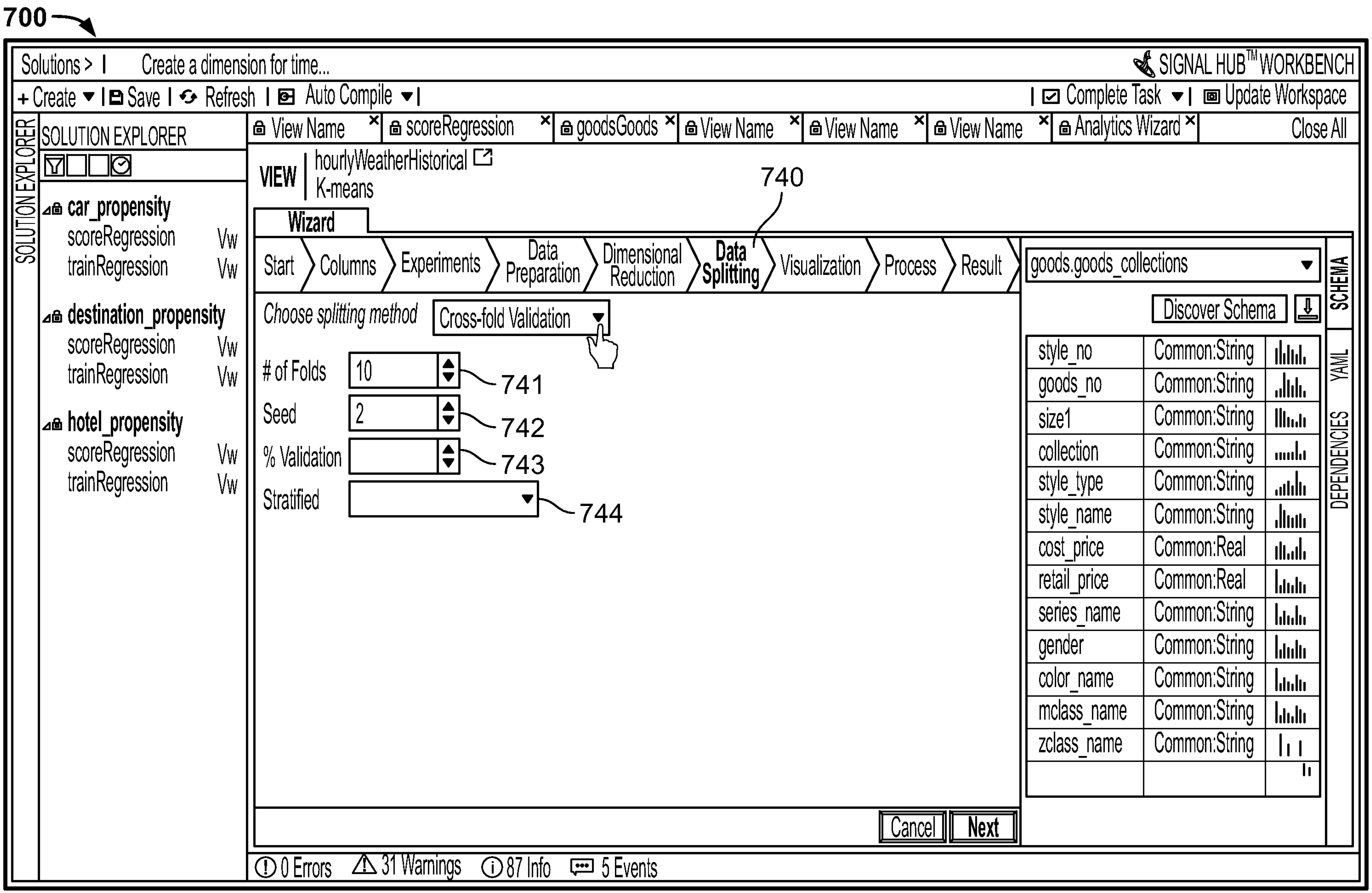
Discover Schema

style_no	Common:String	
goods_no	Common:String	
size1	Common:String	
collection	Common:String	
style_type	Common:String	
style_name	Common:String	
cost_price	Common:Real	
retail_price	Common:Real	
series_name	Common:String	
gender	Common:String	
color_name	Common:String	
mclass_name	Common:String	
zclass_name	Common:String	

SCHEMA | YAML | DEPENDENCIES

ⓘ 0 Errors ⚠ 31 Warnings ⓘ 87 Info 5 Events

FIG. 25B



Choose splitting method Cross-fold Validation ▾

# of Folds 10

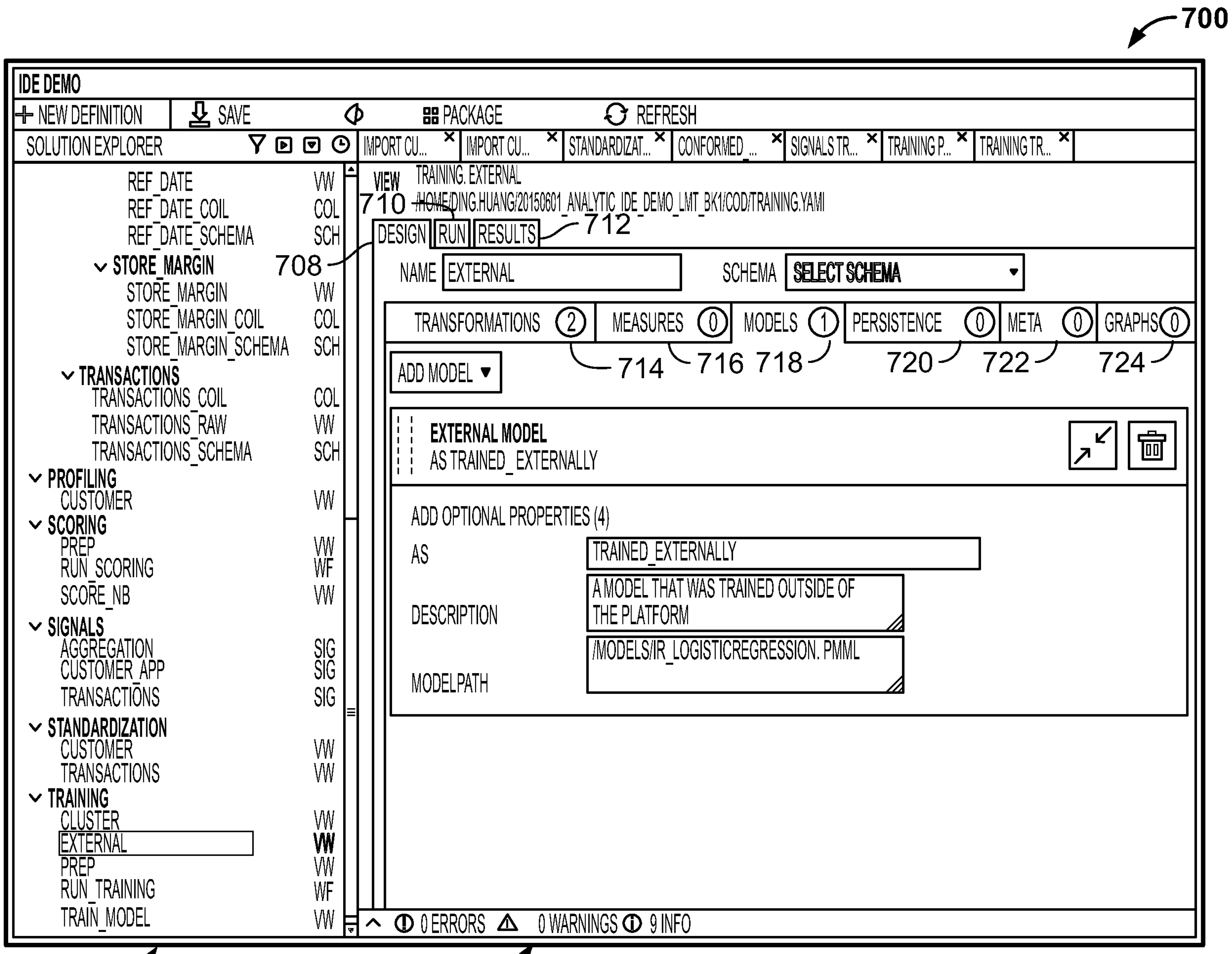
Seed 2

% Validation

Stratified ▾

style_no	Common:String
goods_no	Common:String
size1	Common:String
collection	Common:String
style_type	Common:String
style_name	Common:String
cost_price	Common:Real
retail_price	Common:Real
series_name	Common:String
gender	Common:String
color_name	Common:String
mclass_name	Common:String
zclass_name	Common:String

FIG. 25C



TRAINING. EXTERNAL
~~/HOME~~/DING.HUANG/20150601_ANALYTIC_IDE_DEMO_LMT_BK1/COD/TRAINING.YAMl

NAME EXTERNAL

SCHEMA SELECT SCHEMA

EXTERNAL MODEL
AS TRAINED_EXTERNALLY

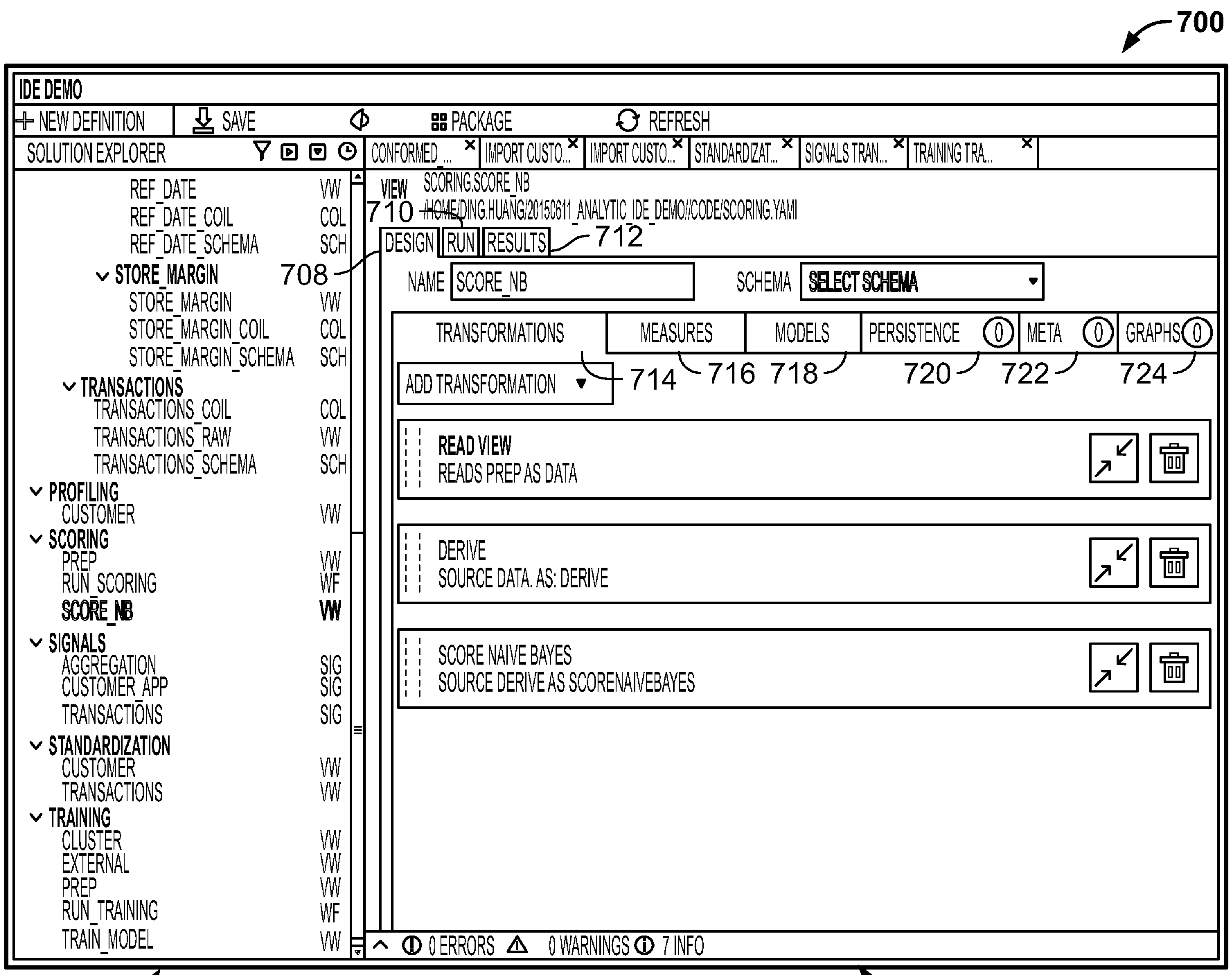
ADD OPTIONAL PROPERTIES (4)

AS TRAINED_EXTERNALLY

DESCRIPTION A MODEL THAT WAS TRAINED OUTSIDE OF THE PLATFORM

MODELPATH /MODELS/IR_LOGISTICREGRESSION. PMML

FIG. 26



REF_DATE	VW
REF_DATE_COIL	COL
REF_DATE_SCHEMA	SCH
STORE_MARGIN	
STORE_MARGIN	VW
STORE_MARGIN_COIL	COL
STORE_MARGIN_SCHEMA	SCH
TRANSACTIONS	
TRANSACTIONS_COIL	COL
TRANSACTIONS_RAW	VW
TRANSACTIONS_SCHEMA	SCH
PROFILING	
CUSTOMER	VW
SCORING	
PREP	VW
RUN_SCORING	WF
SCORE_NB	VW
SIGNALS	
AGGREGATION	SIG
CUSTOMER_APP	SIG
TRANSACTIONS	SIG
STANDARDIZATION	
CUSTOMER	VW
TRANSACTIONS	VW
TRAINING	
CLUSTER	VW
EXTERNAL	VW
PREP	VW
RUN_TRAINING	WF
TRAIN_MODEL	VW

VIEW SCORING.SCORE_NB
/HOME/DING.HUANG/20150611_ANALYTIC_IDE_DEMO//CODE/SCORING.YAMI

NAME SCORE_NB SCHEMA SELECT SCHEMA

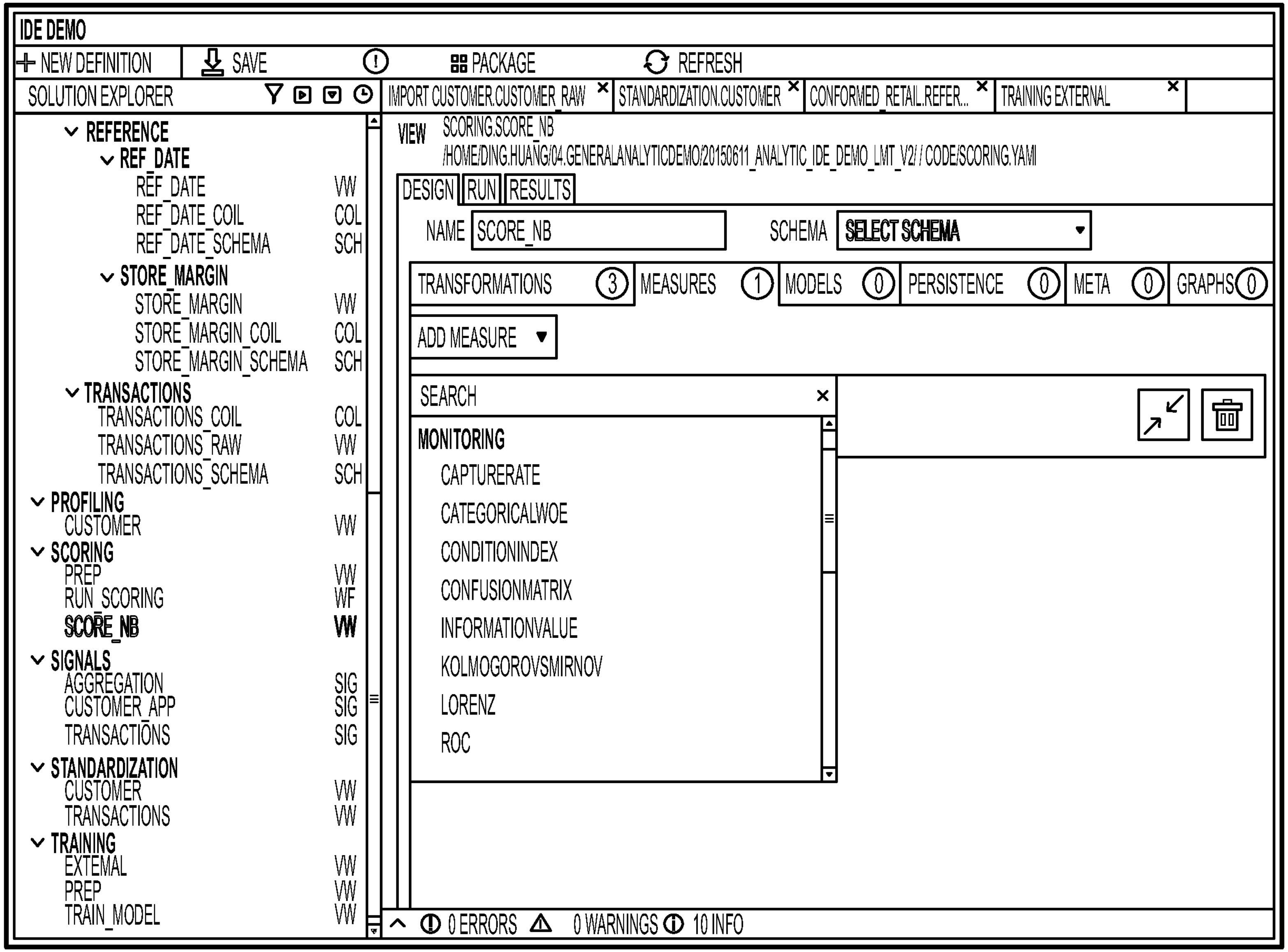
READ VIEW
READS PREP AS DATA

DERIVE
SOURCE DATA. AS: DERIVE

SCORE NAIVE BAYES
SOURCE DERIVE AS SCORENAIVEBAYES

0 ERRORS 0 WARNINGS 7 INFO

FIG. 27



IDE DEMO

NEW DEFINITION
SAVE
PACKAGE
REFRESH
SOLUTION EXPLORER
IMPORT CUSTOMER.CUSTOMER_RAW ×
STANDARDIZATION.CUSTOMER ×
CONFORMED_RETAIL.REFER... ×
TRAINING EXTERNAL ×

REFERENCE	
REF_DATE	
REF_DATE	VW
REF_DATE_COIL	COL
REF_DATE_SCHEMA	SCH
STORE_MARGIN	
STORE_MARGIN	VW
STORE_MARGIN_COIL	COL
STORE_MARGIN_SCHEMA	SCH
TRANSACTIONS	
TRANSACTIONS_COIL	COL
TRANSACTIONS_RAW	VW
TRANSACTIONS_SCHEMA	SCH
PROFILING	
CUSTOMER	VW
SCORING	
PREP	VW
RUN_SCORING	WF
SCORE_NB	**VW**
SIGNALS	
AGGREGATION	SIG
CUSTOMER_APP	SIG
TRANSACTIONS	SIG
STANDARDIZATION	
CUSTOMER	VW
TRANSACTIONS	VW
TRAINING	
EXTEMAL	VW
PREP	VW
TRAIN_MODEL	VW

VIEW SCORING.SCORE_NB
/HOME/DING.HUANG/04.GENERALANALYTICDEMO/20150611_ANALYTIC_IDE_DEMO_LMT_V2/ / CODE/SCORING.YAMI

NAME SCORE_NB
SCHEMA SELECT SCHEMA ▾

MONITORING
CAPTURERATE
CATEGORICALWOE
CONDITIONINDEX
CONFUSIONMATRIX
INFORMATIONVALUE
KOLMOGOROVSMIRNOV
LORENZ
ROC

0 ERRORS 0 WARNINGS 10 INFO

FIG. 28

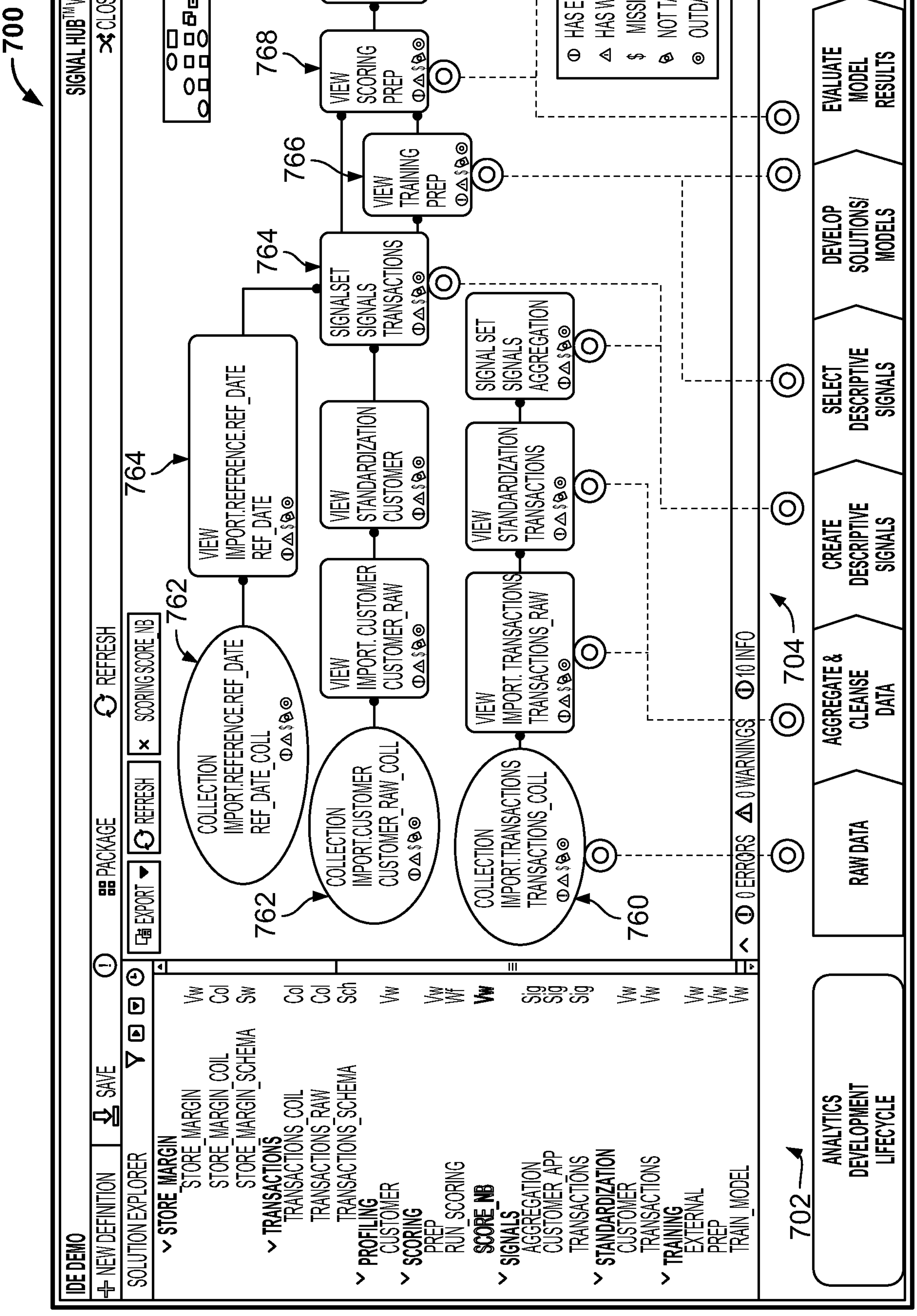
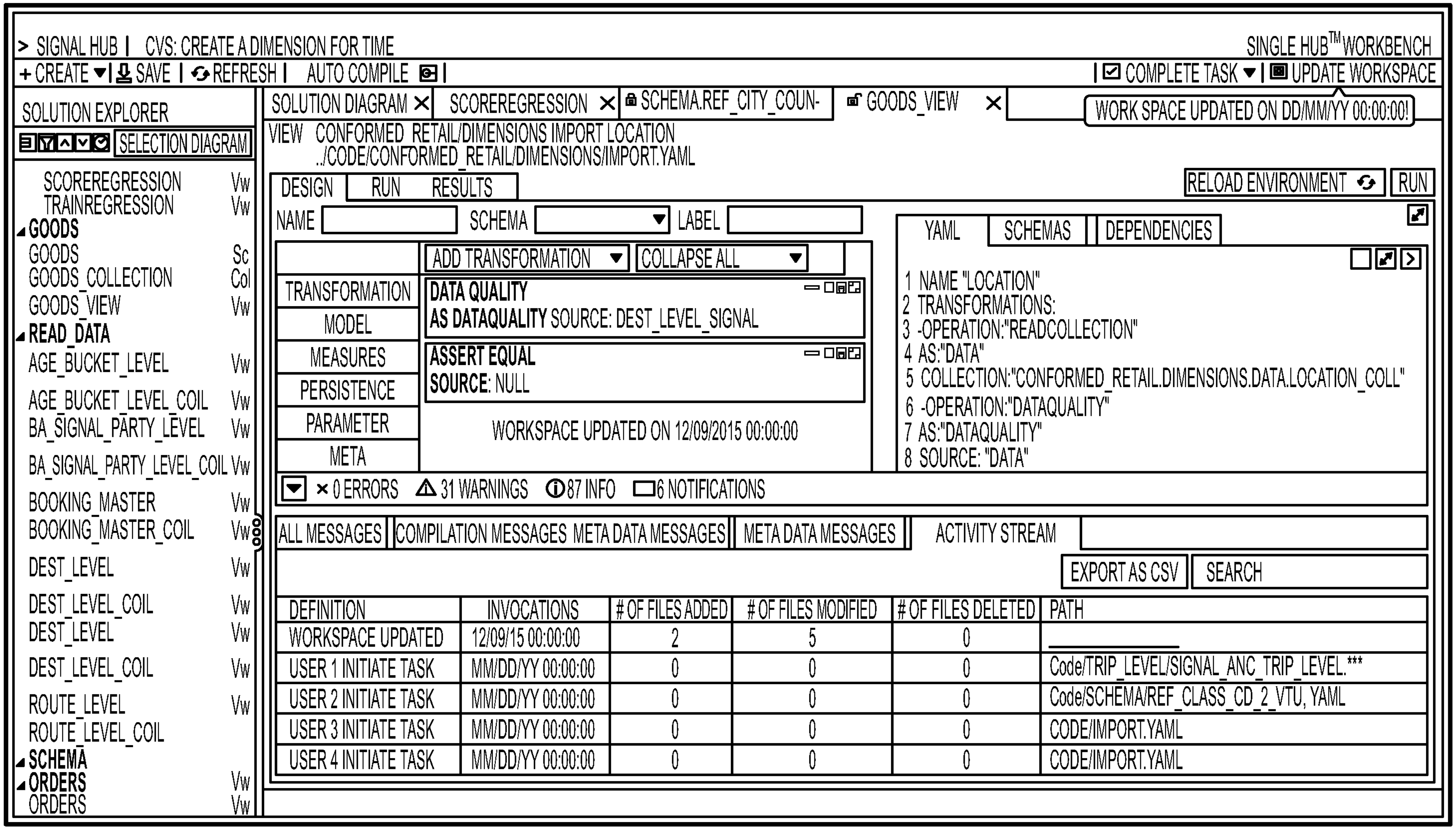

700
IDE DEMO
SIGNAL HUB™ WORKBENCH
+ NEW DEFINITION
SAVE
!
PACKAGE
REFRESH
CLOSE DIAGRAM
SOLUTION EXPLORER
STORE_MARGIN
STORE_MARGIN Vw
STORE_MARGIN_COLL Col
STORE_MARGIN_SCHEMA Sw
TRANSACTIONS
TRANSACTIONS_COLL Col
TRANSACTIONS_RAW Col
TRANSACTIONS_SCHEMA Sch
PROFILING
CUSTOMER Vw
SCORING
PREP Vw
RUN_SCORING Wf
SCORE_NB Vw
SIGNALS
AGGREGATION Sig
CUSTOMER_APP Sig
TRANSACTIONS Sig
STANDARDIZATION
CUSTOMER Vw
TRANSACTIONS Vw
TRAINING
EXTERNAL Vw
PREP Vw
TRAIN_MODEL Vw
EXPORT ▾
REFRESH
× SCORING.SCORE_NB
762
COLLECTION IMPORT.REFERENCE.REF_DATE REF_DATE_COLL
764
VIEW IMPORT.REFERENCE.REF_DATE REF_DATE
762
COLLECTION IMPORT.CUSTOMER CUSTOMER_RAW_COLL
VIEW IMPORT.CUSTOMER CUSTOMER_RAW
VIEW STANDARDIZATION CUSTOMER
760
COLLECTION IMPORT.TRANSACTIONS TRANSACTIONS_COLL
VIEW IMPORT.TRANSACTIONS TRANSACTIONS_RAW
VIEW STANDARDIZATION TRANSACTIONS
SIGNAL SET SIGNALS AGGREGATION
764
SIGNALSET SIGNALS TRANSACTIONS
766
VIEW TRAINING PREP
768
VIEW SCORING PREP
768
VIEW SCORING SCORE_NB
HAS ERRORS
HAS WARNINGS
MISSING SCHEMA
NOT TAGGED
OUTDATED
0 ERRORS 0 WARNINGS 10 INFO
704
702
ANALYTICS DEVELOPMENT LIFECYCLE
RAW DATA
AGGREGATE & CLEANSE DATA
CREATE DESCRIPTIVE SIGNALS
SELECT DESCRIPTIVE SIGNALS
DEVELOP SOLUTIONS/ MODELS
EVALUATE MODEL RESULTS

FIG. 29A



> SIGNAL HUB | CVS: CREATE A DIMENSION FOR TIME

SINGLE HUB™ WORKBENCH

+ CREATE ▾ | SAVE | REFRESH | AUTO COMPILE | ☑ COMPLETE TASK ▾ | UPDATE WORKSPACE

SOLUTION DIAGRAM ✕ | SCOREREGRESSION ✕ | SCHEMA.REF_CITY_COUN- | GOODS_VIEW ✕

WORK SPACE UPDATED ON DD/MM/YY 00:00:00!

SOLUTION EXPLORER

SELECTION DIAGRAM

SCOREREGRESSION	Vw
TRAINREGRESSION	Vw
◢ **GOODS**	
GOODS	Sc
GOODS_COLLECTION	Col
GOODS_VIEW	Vw
◢ **READ_DATA**	
AGE_BUCKET_LEVEL	Vw
AGE_BUCKET_LEVEL_COIL	Vw
BA_SIGNAL_PARTY_LEVEL	Vw
BA_SIGNAL_PARTY_LEVEL_COIL	Vw
BOOKING_MASTER	Vw
BOOKING_MASTER_COIL	Vw
DEST_LEVEL	Vw
DEST_LEVEL_COIL	Vw
DEST_LEVEL	Vw
DEST_LEVEL_COIL	Vw
ROUTE_LEVEL	Vw
ROUTE_LEVEL_COIL	
◢ **SCHEMA**	
◢ **ORDERS**	Vw
ORDERS	Vw

VIEW CONFORMED_RETAIL/DIMENSIONS IMPORT LOCATION
../CODE/CONFORMED_RETAIL/DIMENSIONS/IMPORT.YAML

DESIGN | RUN | RESULTS

RELOAD ENVIRONMENT | RUN

NAME | SCHEMA ▾ | LABEL

TRANSFORMATION | MODEL | MEASURES | PERSISTENCE | PARAMETER | META

ADD TRANSFORMATION ▾ | COLLAPSE ALL ▾

DATA QUALITY
AS DATAQUALITY SOURCE: DEST_LEVEL_SIGNAL

ASSERT EQUAL
SOURCE: NULL

WORKSPACE UPDATED ON 12/09/2015 00:00:00

YAML | SCHEMAS | DEPENDENCIES

```
NAME "LOCATION"
TRANSFORMATIONS:
-OPERATION:"READCOLLECTION"
AS:"DATA"
COLLECTION:"CONFORMED_RETAIL.DIMENSIONS.DATA.LOCATION_COLL"
-OPERATION:"DATAQUALITY"
AS:"DATAQUALITY"
SOURCE: "DATA"
```

✕ 0 ERRORS ⚠ 31 WARNINGS ⓘ 87 INFO ▭ 6 NOTIFICATIONS

ALL MESSAGES | COMPILATION MESSAGES META DATA MESSAGES | META DATA MESSAGES | ACTIVITY STREAM

EXPORT AS CSV | SEARCH

DEFINITION	INVOCATIONS	# OF FILES ADDED	# OF FILES MODIFIED	# OF FILES DELETED	PATH
WORKSPACE UPDATED	12/09/15 00:00:00	2	5	0	________
USER 1 INITIATE TASK	MM/DD/YY 00:00:00	0	0	0	Code/TRIP_LEVEL/SIGNAL_ANC_TRIP_LEVEL.***
USER 2 INITIATE TASK	MM/DD/YY 00:00:00	0	0	0	Code/SCHEMA/REF_CLASS_CD_2_VTU, YAML
USER 3 INITIATE TASK	MM/DD/YY 00:00:00	0	0	0	CODE/IMPORT.YAML
USER 4 INITIATE TASK	MM/DD/YY 00:00:00	0	0	0	CODE/IMPORT.YAML

FIG. 29B

```
fileSearchPaths:
- code

libraryOutputPaths:
- import: hdfs://172.30.255.255:8020/projects/StoreSales/import
- etl: hdfs://172.30.255.255:8020/projects/StoreSales/etl

dataOutputPath : gen/default
ontologyPath: ./ontology.yaml


parameters:
  dataDir: ./data
  importVersion: 1.1
  etlVersion: 1.4
```

FIG. 29C

```
inherit: env_project.yaml
parameters:
  etlversion: 1.5 # still testing with data from 4/16/2015
```

FIG. 29D

```
libraryOutputPaths:
  import: hdfs://172.30.255.255:8020/projects/StoreSales/import
  etl: hdfs://172.30.255.255:8020/projects/StoreSales/etl
```

FIG. 29E

```
libraryOutputPaths:
  import.customers:
hdfs://172.30.255.255:8020/projects/StoreSales/import_customers
  import.stores: hdfs://172.30.255.255:8020/projects/StoreSales/import_stores
  etl: hdfs://172.30.255.255:8020/projects/StoreSales/etl
```

FIG. 29F

```
libraryOutputPaths:
  import.customers:
hdfs://172.30.255.255:8020/projects/StoreSales/import_customers
  import.stores: hdfs://172.30.255.255:8020/projects/StoreSales/import_stores
  etl: hdfs://172.30.255.255:8020/projects/StoreSales/etl
```

FIG. 29G

```
libraryOutputPaths:
  import.customers:
hdfs://172.30.255.255:8020/projects/StoreSales/import_customers
  import.stores: hdfs://172.30.255.255:8020/projects/StoreSales/import_stores
  etl.makeMaster:
hdfs://172.30.255.255:8020/projects/StoreSales/etl_makeMaster
  etl.dataQuality:
hdfs://172.30.255.255:8020/projects/StoreSales/etl_dataQuality
```

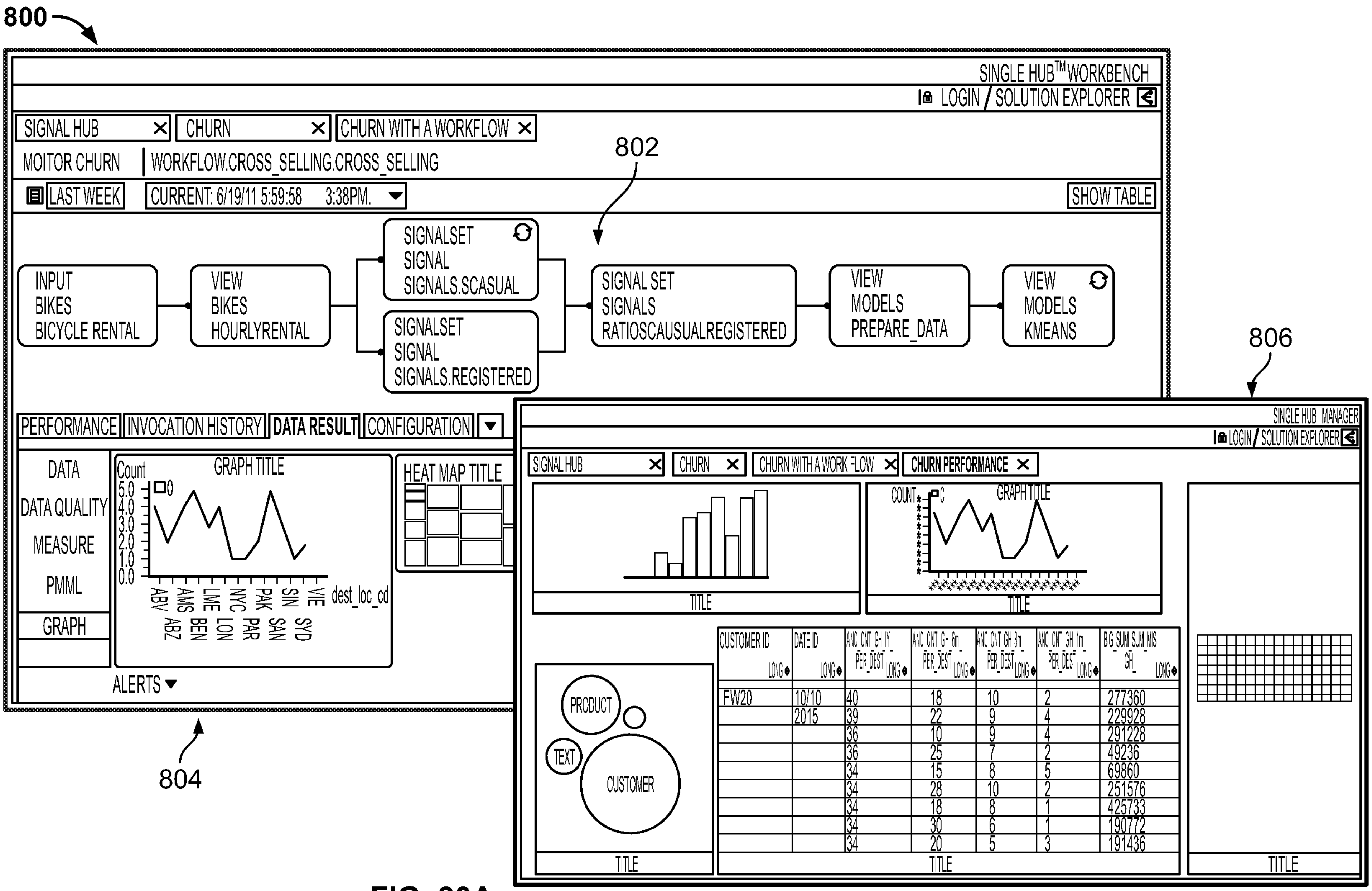
FIG. 29H

```
libraryOutputPaths:
  import.customers.addresses:
hdfs://172.30.255.255:8020/projects/StoreSales/import_customers.addresses
  import.stores: hdfs://172.30.255.255:8020/projects/StoreSales/import_stores
  etl.makeMaster:
hdfs://172.30.255.255:8020/projects/StoreSales/etl_makeMaster
  etl.dataQuality:
hdfs://172.30.255.255:8020/projects/StoreSales/etl_dataQuality
```

FIG. 29I

```
- name: myView
  transformations:
  ...
  ...
  persist:
    label: ${myView_LatestVersion}
```

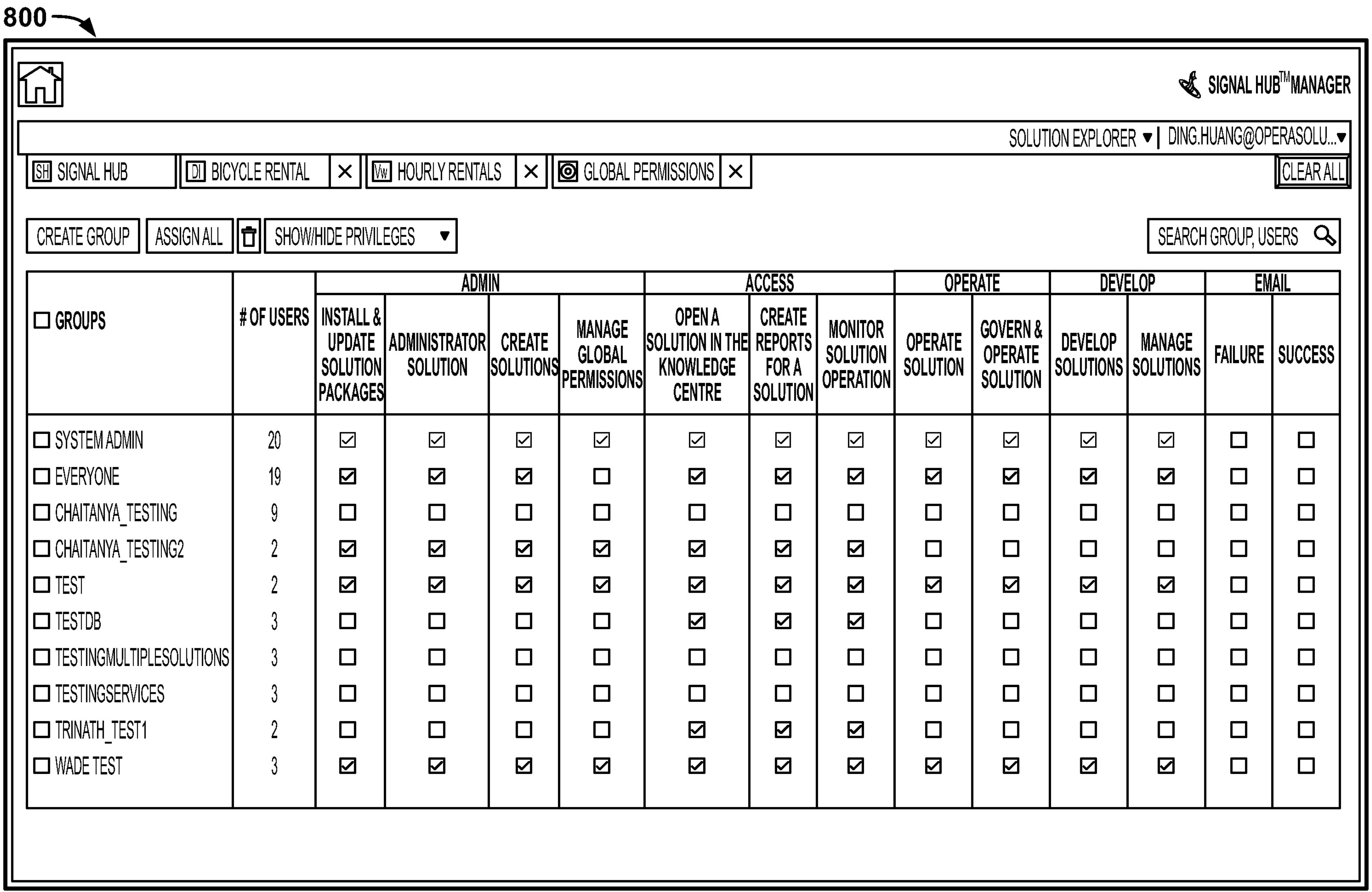
FIG. 29J



CUSTOMER ID LONG ◆	DATE ID LONG ◆	ANC_CNT_GH_IY_PER_DEST LONG ◆	ANC_CNT_GH_6m_PER_DEST LONG ◆	ANC_CNT_GH_3m_PER_DEST LONG ◆	ANC_CNT_GH_1m_PER_DEST LONG ◆	BIG_SUM_SUM_MIS_GH_ LONG ◆
FW20	10/10	40	18	10	2	277360
	2015	39	22	9	4	229928
		36	10	9	4	291228
		36	25	7	2	49236
		34	15	8	5	69860
		34	28	10	2	251576
		34	18	8	1	425733
		34	30	6	1	190772
		34	20	5	3	191436

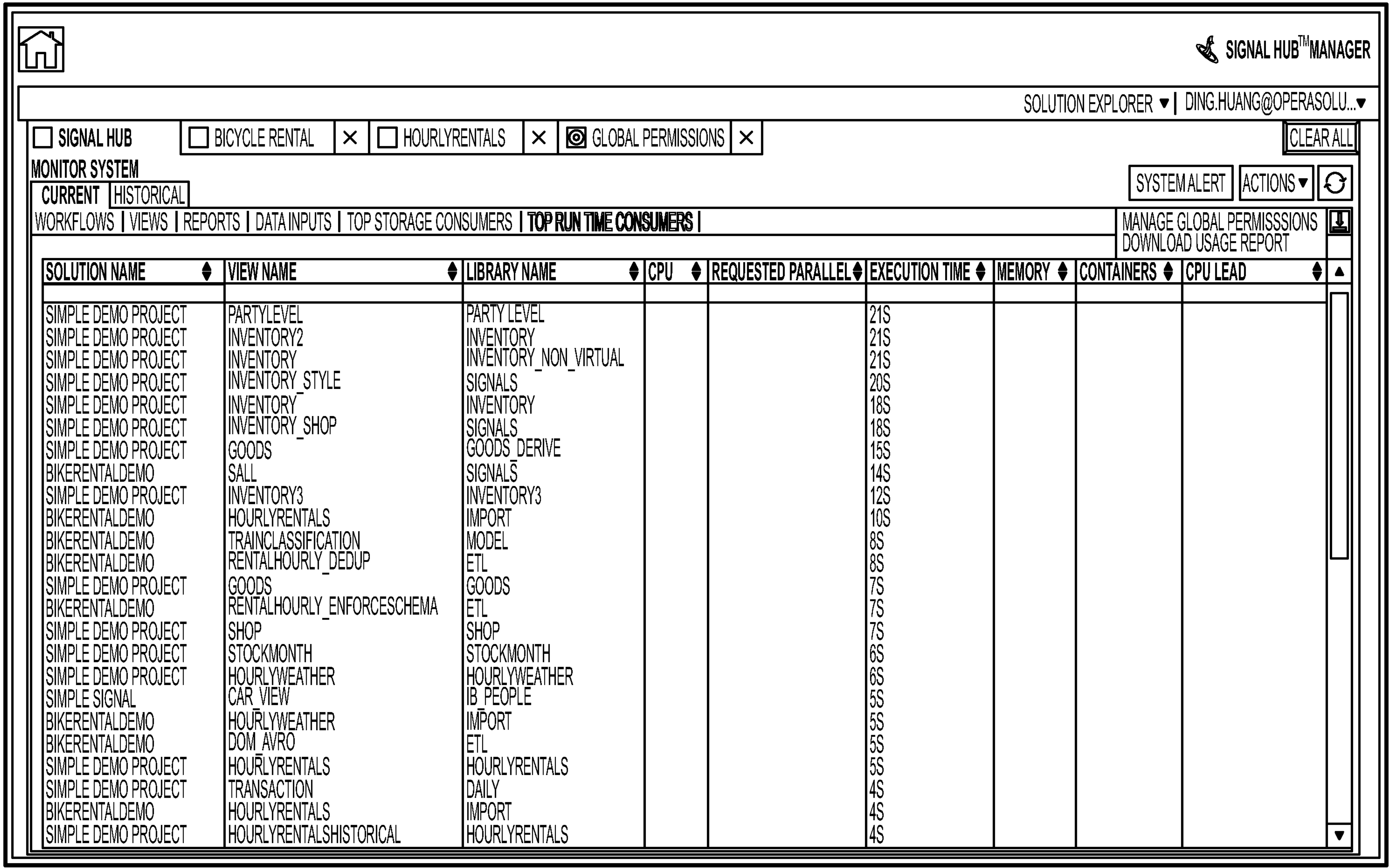
TITLE

FIG. 30A



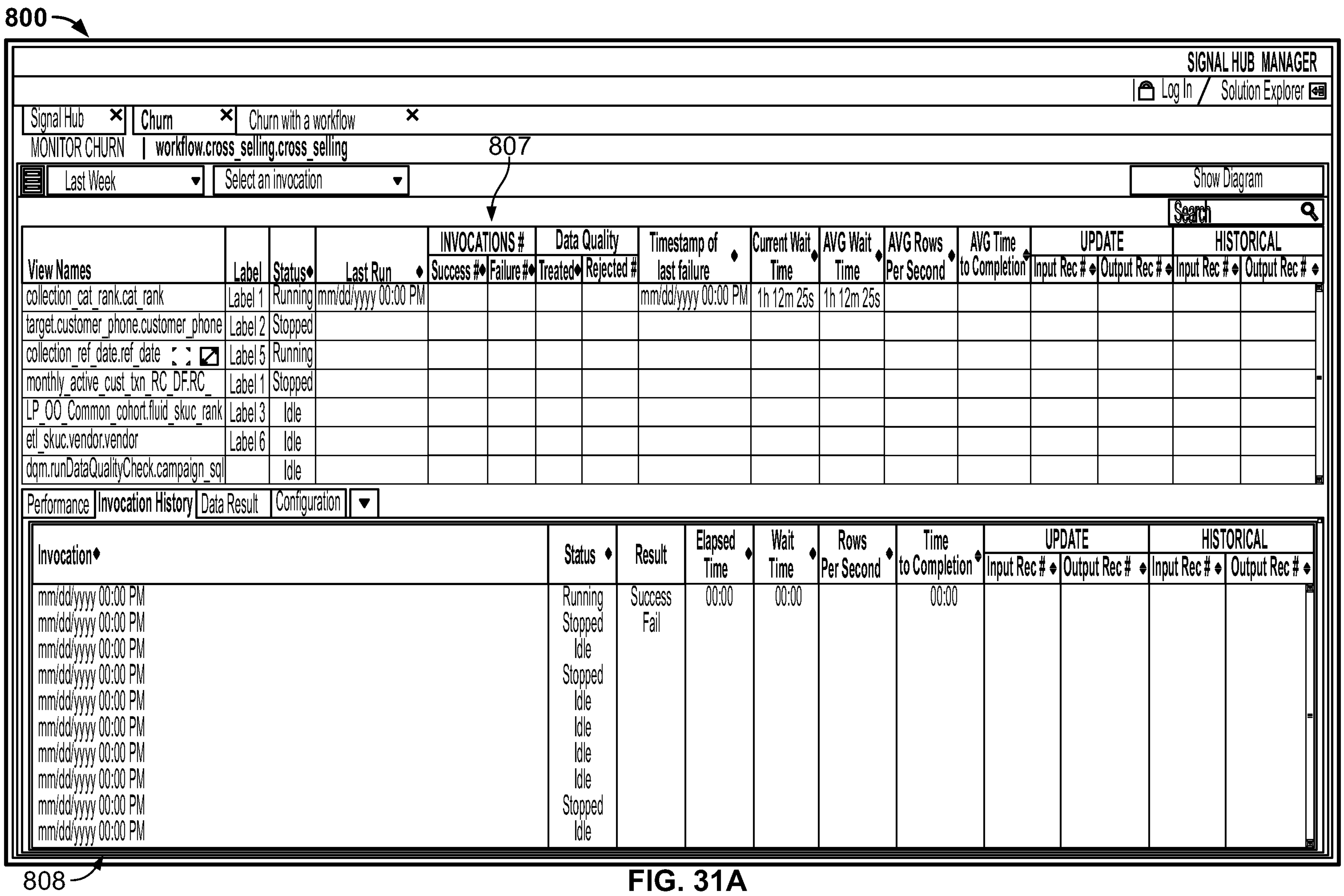
□ GROUPS	# OF USERS	ADMIN				ACCESS			OPERATE		DEVELOP		EMAIL	
		INSTALL & UPDATE SOLUTION PACKAGES	ADMINISTRATOR SOLUTION	CREATE SOLUTIONS	MANAGE GLOBAL PERMISSIONS	OPEN A SOLUTION IN THE KNOWLEDGE CENTRE	CREATE REPORTS FOR A SOLUTION	MONITOR SOLUTION OPERATION	OPERATE SOLUTION	GOVERN & OPERATE SOLUTION	DEVELOP SOLUTIONS	MANAGE SOLUTIONS	FAILURE	SUCCESS
□ SYSTEM ADMIN	20	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□
□ EVERYONE	19	☑	☑	☑	□	☑	☑	☑	☑	☑	☑	☑	□	□
□ CHAITANYA_TESTING	9	□	□	□	□	□	□	□	□	□	□	□	□	□
□ CHAITANYA_TESTING2	2	☑	☑	☑	☑	☑	☑	☑	□	□	□	□	□	□
□ TEST	2	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□
□ TESTDB	3	□	□	□	□	☑	☑	☑	□	□	□	□	□	□
□ TESTINGMULTIPLESOLUTIONS	3	□	□	□	□	□	□	□	□	□	□	□	□	□
□ TESTINGSERVICES	3	□	□	□	□	□	□	□	□	□	□	□	□	□
□ TRINATH_TEST1	2	□	□	□	□	☑	☑	☑	□	□	□	□	□	□
□ WADE TEST	3	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	□	□

FIG. 30B



SOLUTION NAME ⬍	VIEW NAME ⬍	LIBRARY NAME ⬍	CPU ⬍	REQUESTED PARALLEL ⬍	EXECUTION TIME ⬍	MEMORY ⬍	CONTAINERS ⬍	CPU LEAD ⬍
SIMPLE DEMO PROJECT	PARTYLEVEL	PARTY LEVEL			21S			
SIMPLE DEMO PROJECT	INVENTORY2	INVENTORY			21S			
SIMPLE DEMO PROJECT	INVENTORY	INVENTORY_NON_VIRTUAL			21S			
SIMPLE DEMO PROJECT	INVENTORY_STYLE	SIGNALS			20S			
SIMPLE DEMO PROJECT	INVENTORY	INVENTORY			18S			
SIMPLE DEMO PROJECT	INVENTORY_SHOP	SIGNALS			18S			
SIMPLE DEMO PROJECT	GOODS	GOODS_DERIVE			15S			
BIKERENTALDEMO	SALL	SIGNALS			14S			
SIMPLE DEMO PROJECT	INVENTORY3	INVENTORY3			12S			
BIKERENTALDEMO	HOURLYRENTALS	IMPORT			10S			
BIKERENTALDEMO	TRAINCLASSIFICATION	MODEL			8S			
BIKERENTALDEMO	RENTALHOURLY_DEDUP	ETL			8S			
SIMPLE DEMO PROJECT	GOODS	GOODS			7S			
BIKERENTALDEMO	RENTALHOURLY_ENFORCESCHEMA	ETL			7S			
SIMPLE DEMO PROJECT	SHOP	SHOP			7S			
SIMPLE DEMO PROJECT	STOCKMONTH	STOCKMONTH			6S			
SIMPLE DEMO PROJECT	HOURLYWEATHER	HOURLYWEATHER			6S			
SIMPLE SIGNAL	CAR_VIEW	IB_PEOPLE			5S			
BIKERENTALDEMO	HOURLYWEATHER	IMPORT			5S			
BIKERENTALDEMO	DOM_AVRO	ETL			5S			
SIMPLE DEMO PROJECT	HOURLYRENTALS	HOURLYRENTALS			5S			
SIMPLE DEMO PROJECT	TRANSACTION	DAILY			4S			
BIKERENTALDEMO	HOURLYRENTALS	IMPORT			4S			
SIMPLE DEMO PROJECT	HOURLYRENTALSHISTORICAL	HOURLYRENTALS			4S			

FIG. 30C

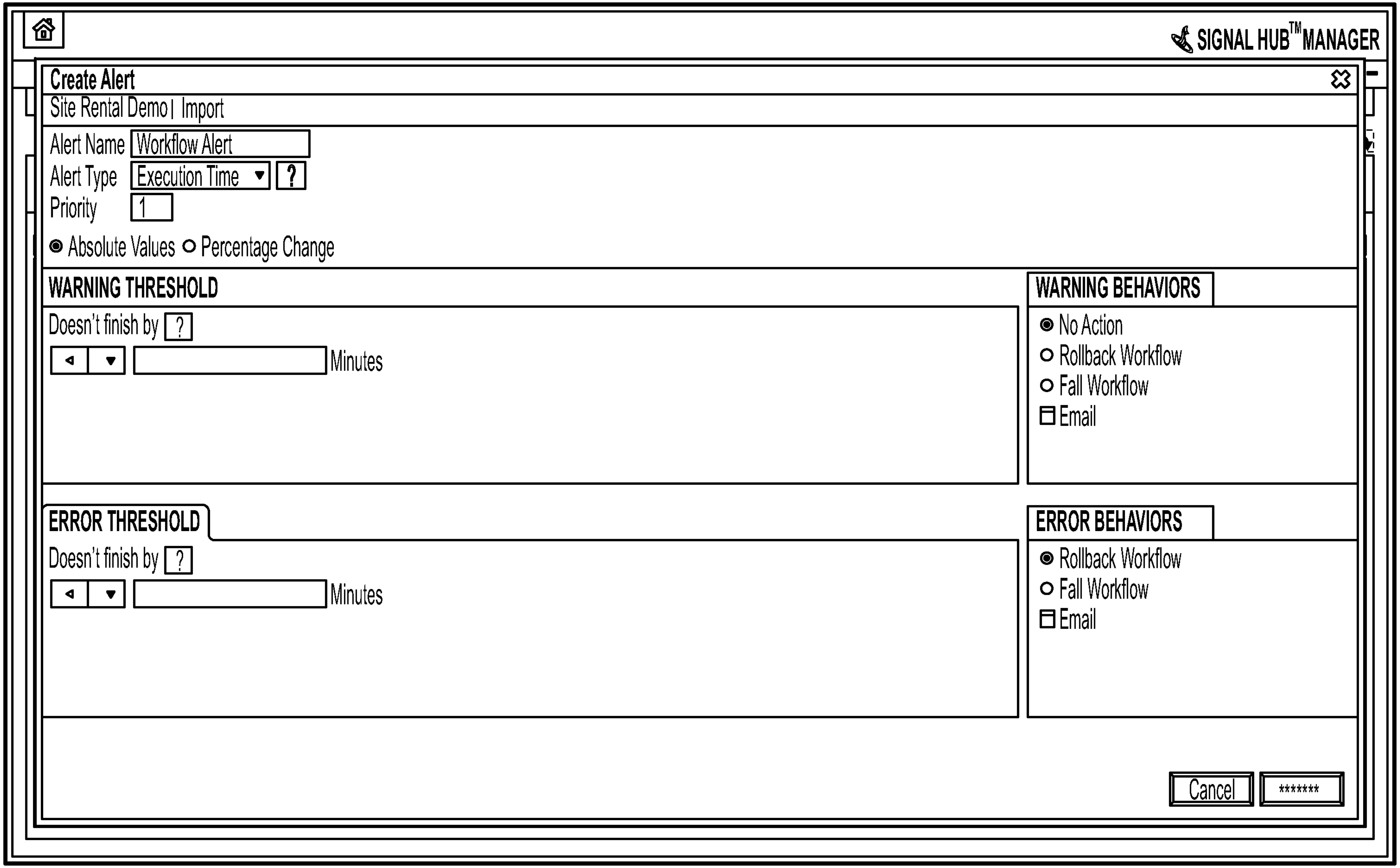


MONITOR CHURN | workflow.cross_selling.cross_selling

View Names	Label	Status◆	Last Run ◆	INVOCATIONS #		Data Quality		Timestamp of last failure ◆	Current Wait Time ◆	AVG Wait Time ◆	AVG Rows Per Second ◆	AVG Time to Completion ◆	UPDATE		HISTORICAL	
				Success #◆	Failure #◆	Treated◆	Rejected #						Input Rec # ◆	Output Rec # ◆	Input Rec # ◆	Output Rec # ◆
collection_cat_rank.cat_rank	Label 1	Running	mm/dd/yyyy 00:00 PM					mm/dd/yyyy 00:00 PM	1h 12m 25s	1h 12m 25s						
target.customer_phone.customer_phone	Label 2	Stopped														
collection_ref_date.ref_date	Label 5	Running														
monthly_active_cust_txn_RC_DF.RC_	Label 1	Stopped														
LP_OO_Common_cohort.fluid_skuc_rank	Label 3	Idle														
etl_skuc.vendor.vendor	Label 6	Idle														
dqm.runDataQualityCheck.campaign_sql		Idle														

Invocation◆	Status ◆	Result	Elapsed Time ◆	Wait Time ◆	Rows Per Second ◆	Time to Completion ◆	UPDATE		HISTORICAL	
							Input Rec # ◆	Output Rec # ◆	Input Rec # ◆	Output Rec # ◆
mm/dd/yyyy 00:00 PM	Running	Success	00:00	00:00		00:00				
mm/dd/yyyy 00:00 PM	Stopped	Fail								
mm/dd/yyyy 00:00 PM	Idle									
mm/dd/yyyy 00:00 PM	Stopped									
mm/dd/yyyy 00:00 PM	Idle									
mm/dd/yyyy 00:00 PM	Idle									
mm/dd/yyyy 00:00 PM	Idle									
mm/dd/yyyy 00:00 PM	Idle									
mm/dd/yyyy 00:00 PM	Stopped									
mm/dd/yyyy 00:00 PM	Idle									

FIG. 31A



Create Alert

Site Rental Demo | Import

Alert Name: Workflow Alert

Alert Type: Execution Time ▼ ?

Priority: 1

● Absolute Values ○ Percentage Change

WARNING THRESHOLD

Doesn't finish by ?

◄ ▼ Minutes

WARNING BEHAVIORS

● No Action
○ Rollback Workflow
○ Fall Workflow
□ Email

ERROR THRESHOLD

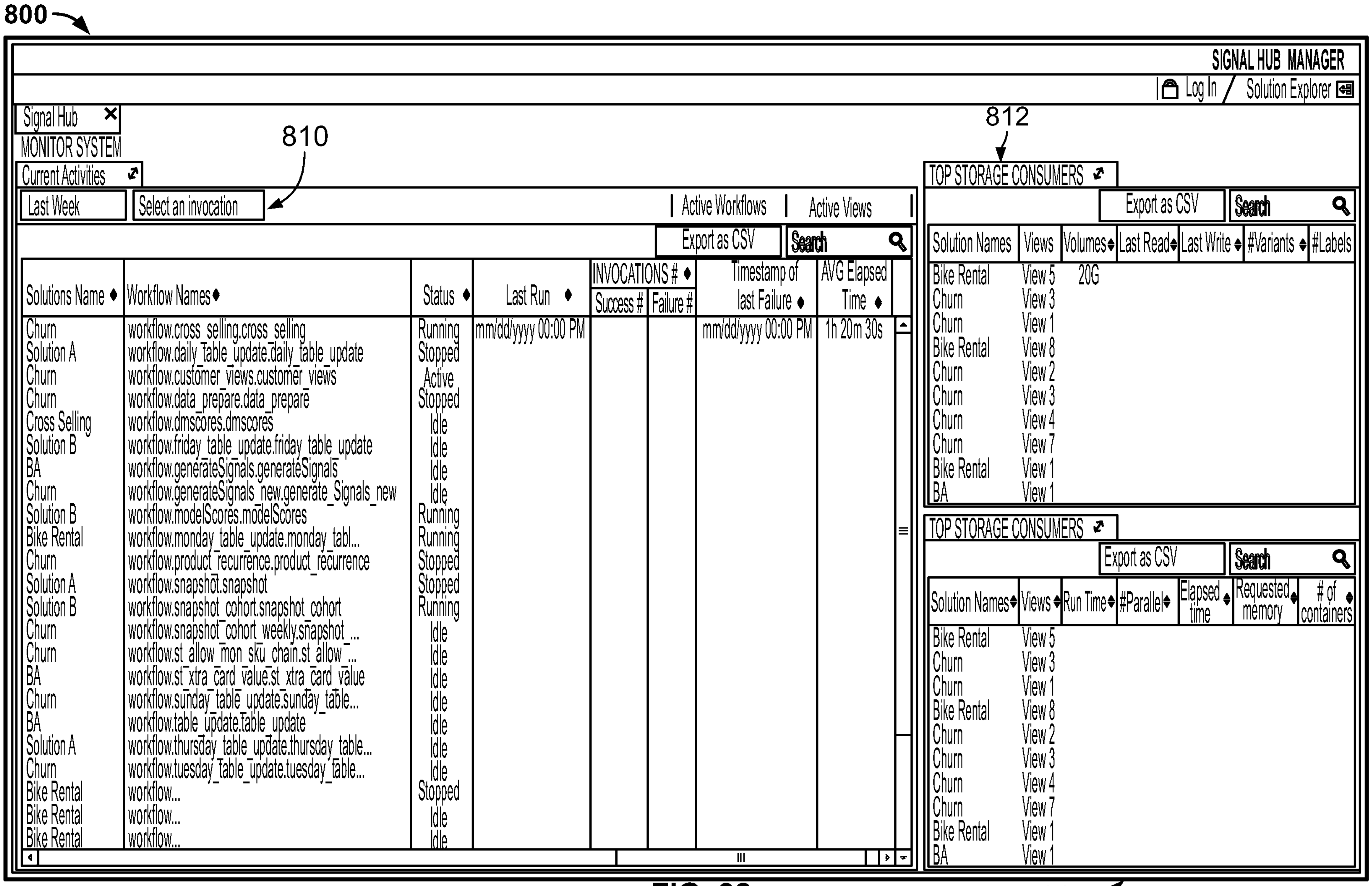
Doesn't finish by ?

◄ ▼ Minutes

ERROR BEHAVIORS

● Rollback Workflow
○ Fall Workflow
□ Email

FIG. 31B



Solutions Name ♦	Workflow Names♦	Status ♦	Last Run ♦	INVOCATIONS # ♦		Timestamp of last Failure ♦	AVG Elapsed Time ♦
				Success #	Failure #		
Churn	workflow.cross_selling.cross_selling	Running	mm/dd/yyyy 00:00 PM			mm/dd/yyyy 00:00 PM	1h 20m 30s
Solution A	workflow.daily_Table_update.daily_table_update	Stopped					
Churn	workflow.customer_views.customer_views	Active					
Churn	workflow.data_prepare.data_prepare	Stopped					
Cross Selling	workflow.dmscores.dmscores	Idle					
Solution B	workflow.friday_table_update.friday_table_update	Idle					
BA	workflow.generateSignals.generateSignals	Idle					
Churn	workflow.generateSignals_new.generate_Signals_new	Idle					
Solution B	workflow.modelScores.modelScores	Running					
Bike Rental	workflow.monday_table_update.monday_tabl...	Running					
Churn	workflow.product_recurrence.product_recurrence	Stopped					
Solution A	workflow.snapshot.snapshot	Stopped					
Solution B	workflow.snapshot_cohort.snapshot_cohort	Running					
Churn	workflow.snapshot_cohort_weekly.snapshot_...	Idle					
Churn	workflow.st_allow_mon_sku_chain.st_allow_...	Idle					
BA	workflow.st_xtra_card_value.st_xtra_card_value	Idle					
Churn	workflow.sunday_table_update.sunday_table...	Idle					
BA	workflow.table_update.table_update	Idle					
Solution A	workflow.thursday_table_update.thursday_table...	Idle					
Churn	workflow.tuesday_Table_update.tuesday_table...	Idle					
Bike Rental	workflow...	Stopped					
Bike Rental	workflow...	Idle					
Bike Rental	workflow...	Idle					

Solution Names	Views	Volumes♦	Last Read♦	Last Write ♦	#Variants ♦	#Labels
Bike Rental	View 5	20G				
Churn	View 3					
Churn	View 1					
Bike Rental	View 8					
Churn	View 2					
Churn	View 3					
Churn	View 4					
Churn	View 7					
Bike Rental	View 1					
BA	View 1					

Solution Names♦	Views ♦	Run Time♦	#Parallel♦	Elapsed time ♦	Requested memory ♦	# of containers ♦
Bike Rental	View 5					
Churn	View 3					
Churn	View 1					
Bike Rental	View 8					
Churn	View 2					
Churn	View 3					
Churn	View 4					
Churn	View 7					
Bike Rental	View 1					
BA	View 1					

FIG. 32

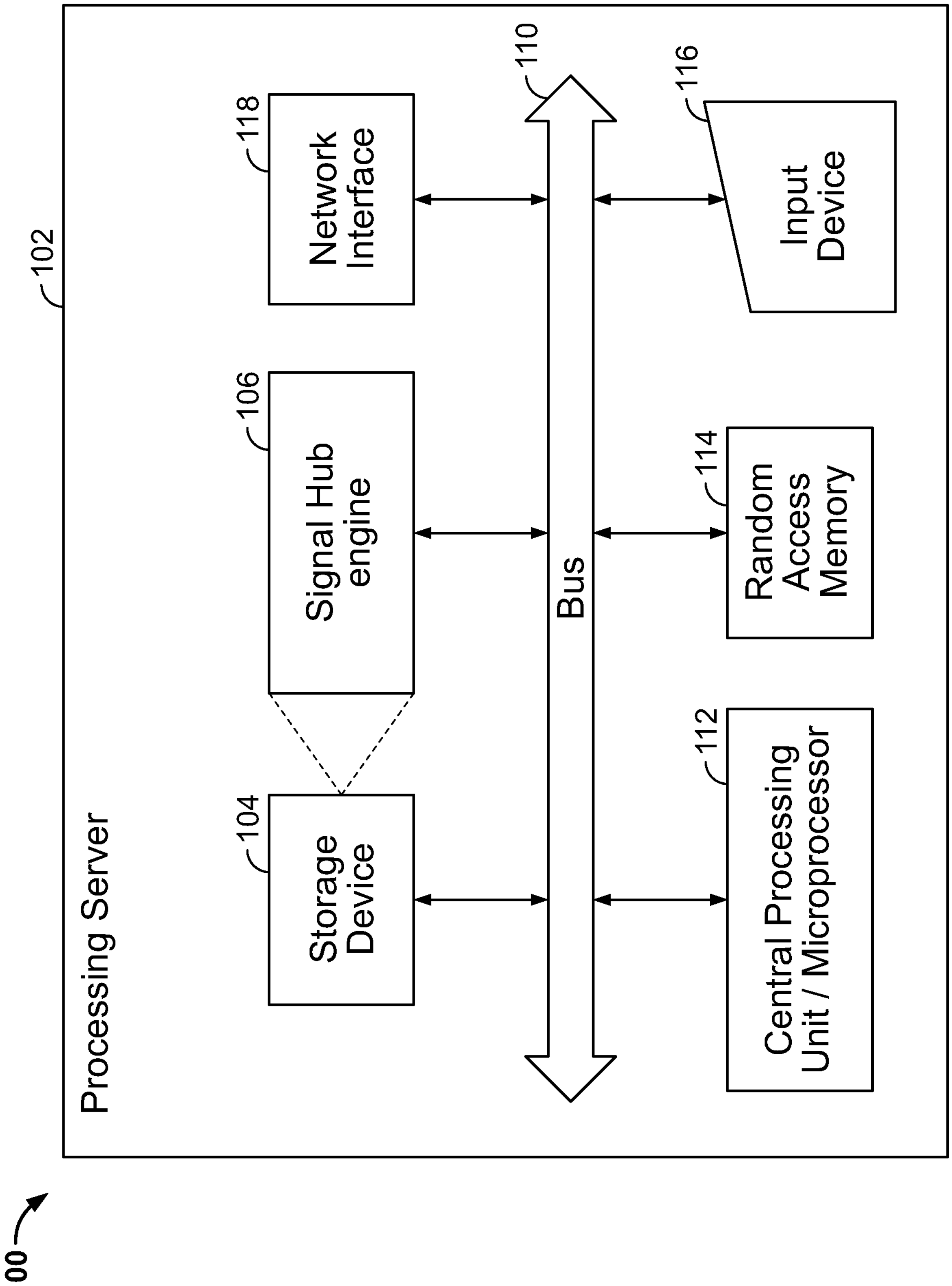

100
102
Processing Server
104
Storage Device
106
Signal Hub engine
118
Network Interface
110
Bus
112
Central Processing Unit / Microprocessor
114
Random Access Memory
116
Input Device

FIG. 33

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/068296

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06F 9/44; G06F 15/173; G06Q 99/00 (2016.01)

CPC - G06F 8/10; G06Q 10/06; H04L 67/02; H04L 67/125; H04L 67/34 (2016.08)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 709/223; 709/224; 717/104 (keyword delimited)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2013/0091170 A1 (ZHANG et al) 11 April 2013 (11.04.2013) entire document	1-60
A	US 2012/0185913 A1 (MARTINEZ et al) 19 July 2012 (19.07.2012) entire document	1-60
A	US 2003/0200532 A1 (GENSEL) 23 October 2003 (23.10.2003) entire document	1-60
A	US 2007/0156430 A1 (KAETKER et al) 05 July 2007 (05.07.2007) entire document	1-60
A	US 2003/0204487 A1 (SSSV et al) 30 October 2003 (30.10.2003) entire document	1-60

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
15 February 2017

Date of mailing of the international search report
10 MAR 2017

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents
P.O. Box 1450, Alexandria, VA 22313-1450
Facsimile No. 571-273-8300

Authorized officer
Blaine R. Copenheaver
PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774