Course 7 - Getting Started with Data Warehousing and BI Analytics

Data Warehouse, Data Marts and Data Lakes

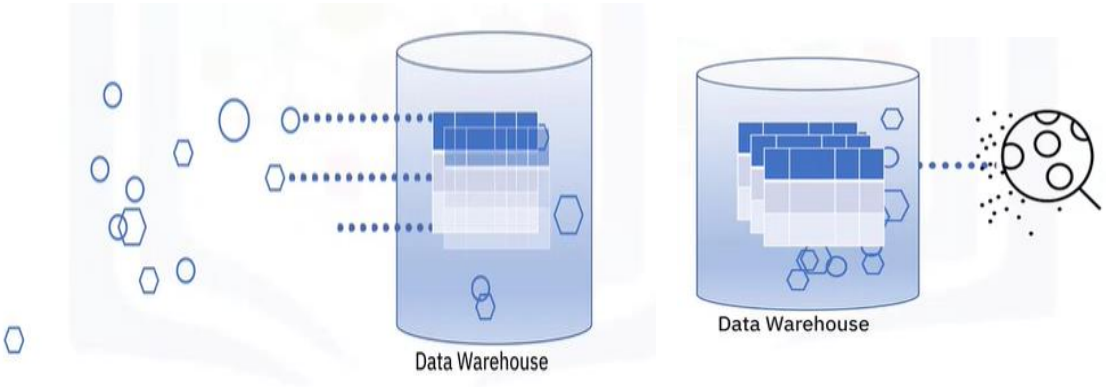
Week 1

Module 1: - An Introduction to Data Warehouses, Data Marts and Data Lakes

1] Data Warehouse Overview

What is a data warehouse?

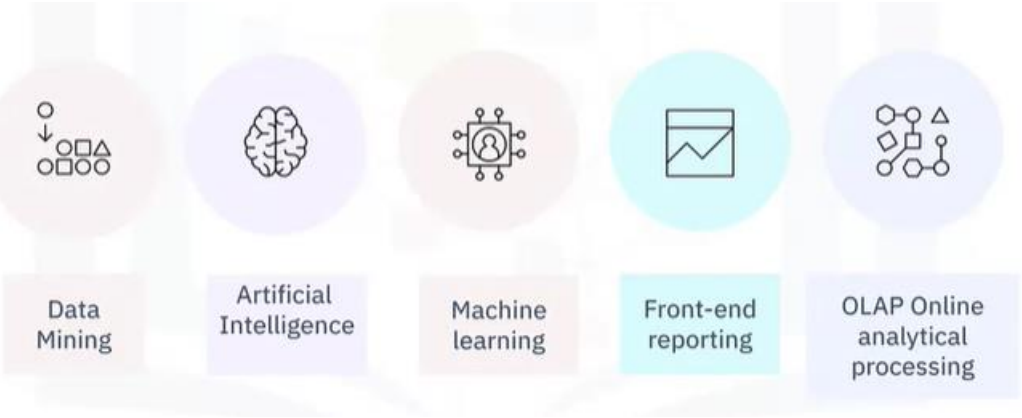
- A data warehouse is a system that aggregates data from one or more sources into a single consistent data store to support data analytics.



Data Warehouse analytics –

Data warehouse systems support:

- Data warehouse systems support data mining, including the application of artificial intelligence and machine learning.
- Data transformation during the ETL process speeds front-end reporting, delivering critical information fast.
- Data warehouses enable online analytical processing, known as OLAP, which provides fast, flexible, multidimensional data analysis for business intelligence and decision support applications.



Where are data warehouse hosted?

The beginning –

- Traditional data warehouses hosted on-premises within enterprise datacentres initially hosted on mainframes, and then on Unix, Windows and Linux systems

2000s –

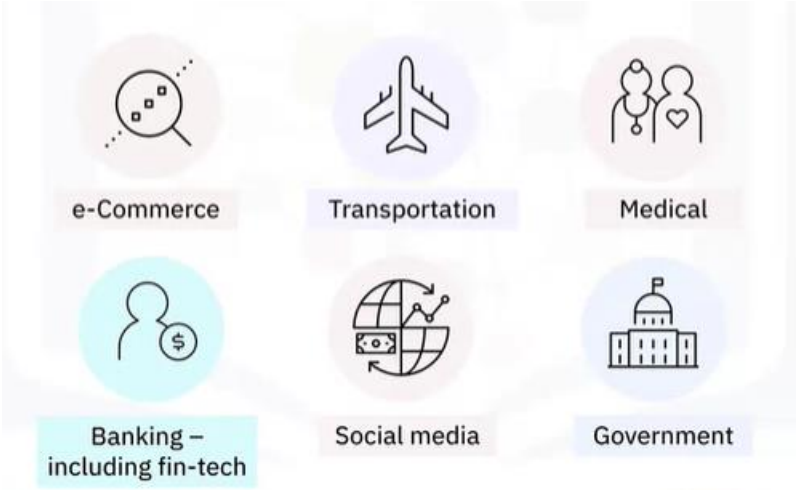
- Growth of larger data volumes, emergence of data warehousing appliances (DWAs), consisting of specialized hardware with pre-integrated software

2010-present –

- Adoption of Cloud Data Warehouses, providing eliminating hardware purchases, as a scalable service pay-as-you-go service.

Who uses data warehouse?

Partially every industry –



Retail and e-commerce –

- Analyze and report on sales performance
- Create machine learning assisted shopping recommendations

Healthcare –

- Apply AI to patient data to assist with diagnosis and treatment

Transportation –

- Optimizes roues, travel times, equipment needs and staffing requirements

Banking and fin-tech –

- Evaluate risks, detect fraud and cross-sell services

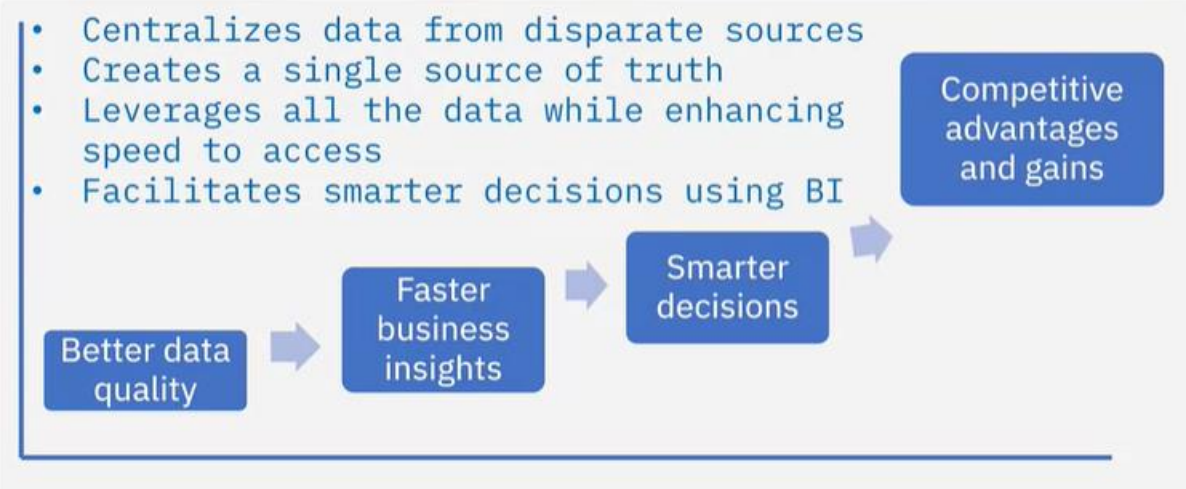
Social media –

- Measure customer sentiment and project product sales

Governments –

- Analyze and evaluate citizen-focused programs and assist with policy change decisions.

Benefits of a data warehouse –



Summary: -

- A data warehouse is a system that aggregates data from one or more sources into a single consistent data store to support data analytics.
- Data warehouses support data mining, AI and machine learning, OLAP, and front-end reporting.

- And finally, data warehouses and BI help organizations improve data quality, speed business insights, and improve decision-making, all of which can result in competitive gains.

2] Popular Data Warehouse Systems

Categorizing data warehouse systems –

- Most data warehouse systems are supported via one or more of three platforms.
- First are appliances, which are pre-integrated bundles of hardware and software that provide high performance for workloads and low maintenance overhead.
- Other vendors support cloud deployments only, offering the benefits of cloud scalability and pay-per-use economics, and in many cases, deliver their data warehouses as fully managed services.
- Some warehouse offerings have traditionally been available as software installed only within on-premises environments, but in recent years, most of these vendors now offer cloud-deployed data warehouse systems.

Vendors – appliance offerings: -

Oracle Exadata –

- Deployable on premises and Oracle Public Cloud
- Includes built-in algorithms
- Runs all type of workloads

IBM Netezza –

- Deploys on IBM Cloud, Amazon Web Services, and Microsoft Azure
- Deploys on private clouds using the IBM Cloud Pak for Data System
- Powers data science and machine learning

Vendors – cloud only: -

Amazon Redshift –

- Uses AWS-specific hardware and proprietary software
- Performs data compression and encryption
- Supports machine learning
- Applies graph optimization algorithms that automatically organize and store data

Snowflake –

- Flexible, multicloud analytics solution
- GDPR and CCPA data privacy compliance
- Always-on encryption of data in transit and at rest
- FedRAMP Moderate authorized

Google BigQuery –

- Flexible, multicloud data warehouse solution
- Reported uptime of 99.99%
- Delivers sub-second query response times from any business intelligence tool
- Petabyte, real-time analytics with high availability and massive concurrency

Vendors – on premises and cloud: -

Microsoft Azure Synapse Analytics –

- Offers code-free visual ETL/ELT processes to easily ingest data from more than 95 negative connectors
- Supports data lake and data warehouse use cases
- Supports T-SQL, Python, Scala, Spark SQL and .NET for both serverless and dedicated resources.

Teradata Vantage –

- Multicloud data platform for enterprise analytics that unifies everything
- Supports mixed workloads with high query concurrency using workload management and adaptive optimization
- Provides a single point of contact for operational task services

IBM Db2 Warehouse –

- Widely recognized for its scalability, MPP capabilities, petaflop speeds
- Rich security features with 99.99% service uptime
- Designed as a containerized, scale-out data warehousing solution
- Move workloads with minimal or no changes required

Vertica –

- Multicloud support for AWS, Google, Microsoft Azure, and on-premises Linux hardware
- Fast multi-GB data transfer rates
- Scalable, elastic compute and storage
- Eon Mode provides notable fault tolerance for volatile cloud environments

Oracle Autonomous Data Warehouse –

- Available in both Oracle Public Cloud and on premises
- Supports multimodel data and multiple workloads
- Built to eliminate manual data management
- Autonomously secures data and performs threat detection

Summary: -

- Data warehouse systems can include appliances, exist on-premises, exist in the cloud, or use a combination of these deployment options.
- Popular data warehouse vendors include Oracle, Teradata, Vertica, Google, IBM, Microsoft, Snowflake, Amazon, and others.

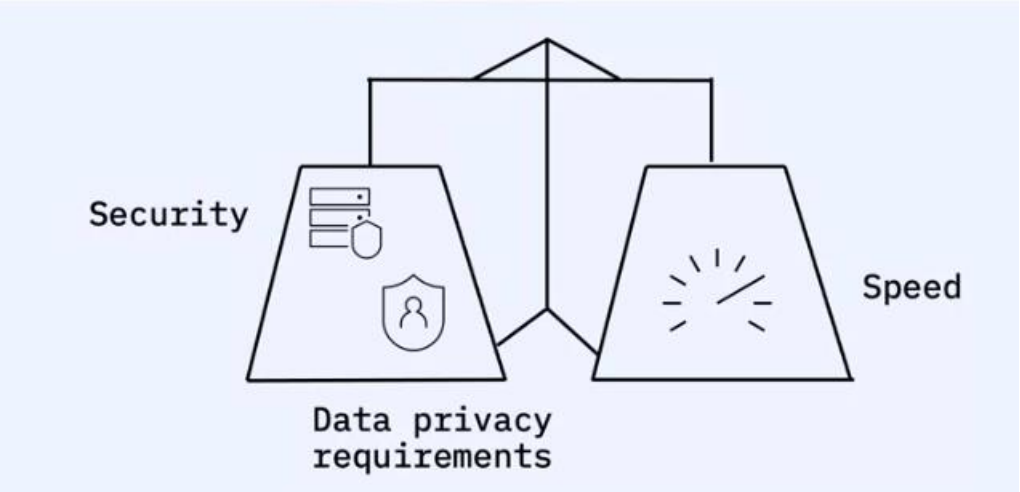
3] Selecting a Data Warehouse System

Evaluating data warehouse systems –

Features and Capabilities: –

Location –

- On-premise
- On-application
- On-cloud



To select a location, organizations must balance multiple demands related to data ingestion, storage, and access. For some organizations, securing their data is their highest priority, requiring a mandatory on-premises solution. Multi-location businesses that grapple with data privacy requirements such as CCPA or GDPR need on-premises or geo-specific data warehouse locations. Every organization balances security and data privacy requirements with the need for speed that delivers critical, profit-producing business insights.

- Architecture and structure
  - Vendor-specific architecture
  - Multicloud installation
  - Scalable
  - Supported data types
  - Batch and streaming data capable

Ease of implementation –

- Governance
- Data migration
- Transformation capabilities
- Optimization
- User management
- Notifications and reports

Ease of use and skills –

- Staff implementation skills
- Vendor or implementation partner's data warehouse implementation skills
- Technical and engineering staff front-end and administration skills for querying, reporting, and visualization tools.

Support –

Does the vendor offer:

- Consolidated support benefits?
- Service-level agreements?
- Convenient support contact methods?
- Self-service solutions and a rich user community?

Costs –

Total cost of ownership (TCO):

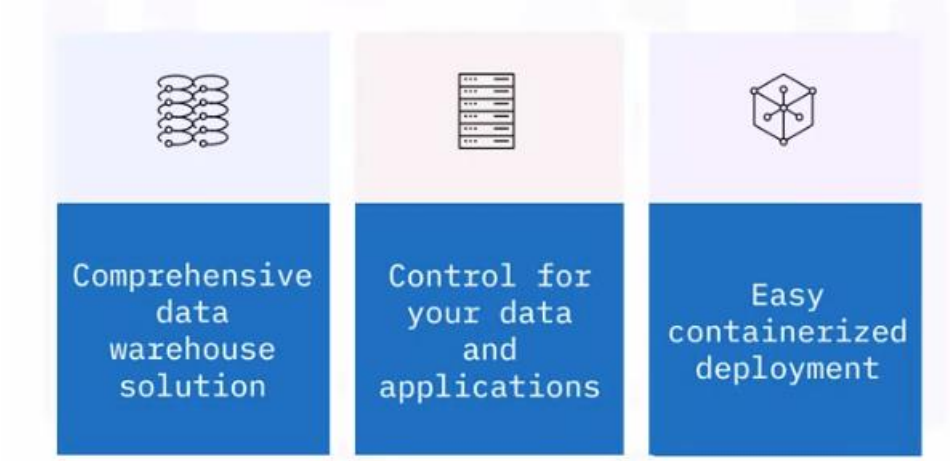
- Infrastructure
- Software licensing / cloud services
- Data migration and integration
- Administration
- Support and maintenance

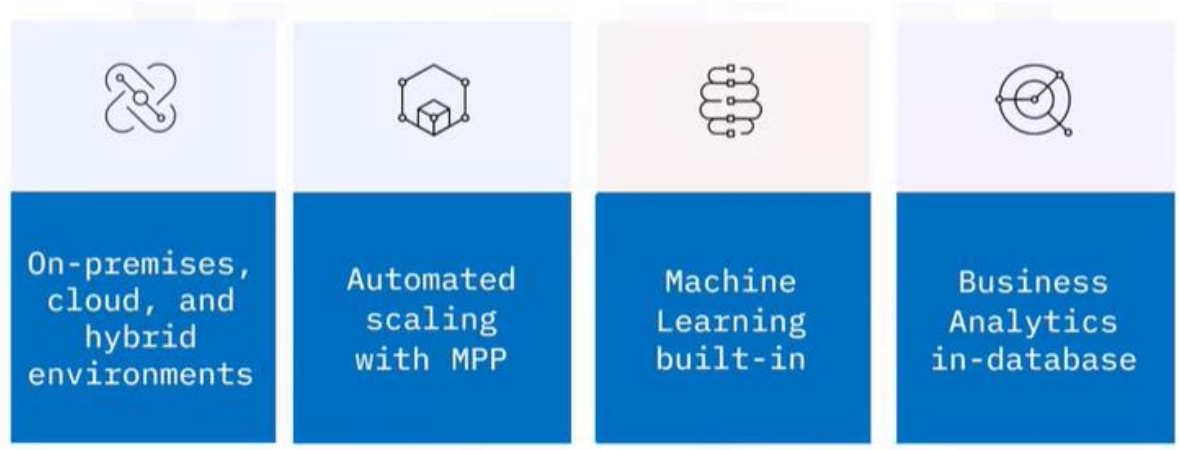
Summary: -

- Businesses evaluate data warehouse systems based on features and capabilities, compatibility and implementation, ease of use and required skills, support quality and availability, and multiple cost considerations.
- An organization might need a traditional on-premises installation to adhere to data security and privacy requirements.
- Public cloud sites offer organizations the benefits of economies of scale including powerful compute power and scalable storage resulting in flexible price-for-performance options.
- And, when selecting a data warehouse system consider the total cost of ownership including infrastructure, compute and storage, data migration, administration, and data maintenance costs in your calculations.

4] IBM Db2 Warehouse (Optional)

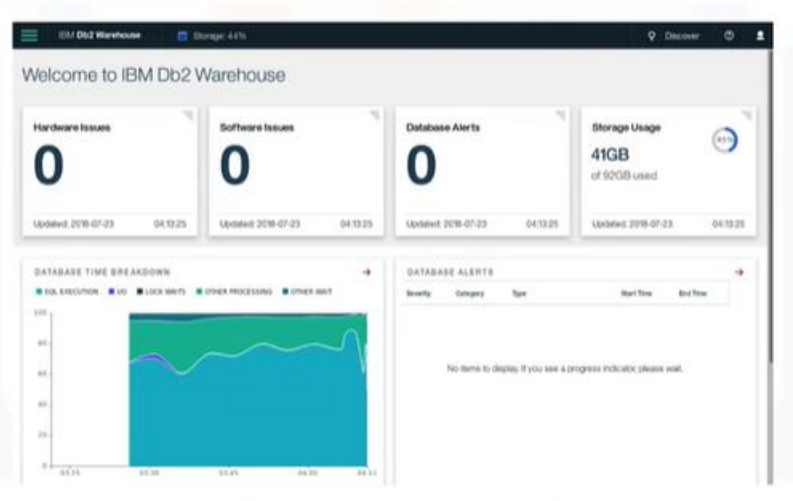
IBM Db2 Warehouse features –



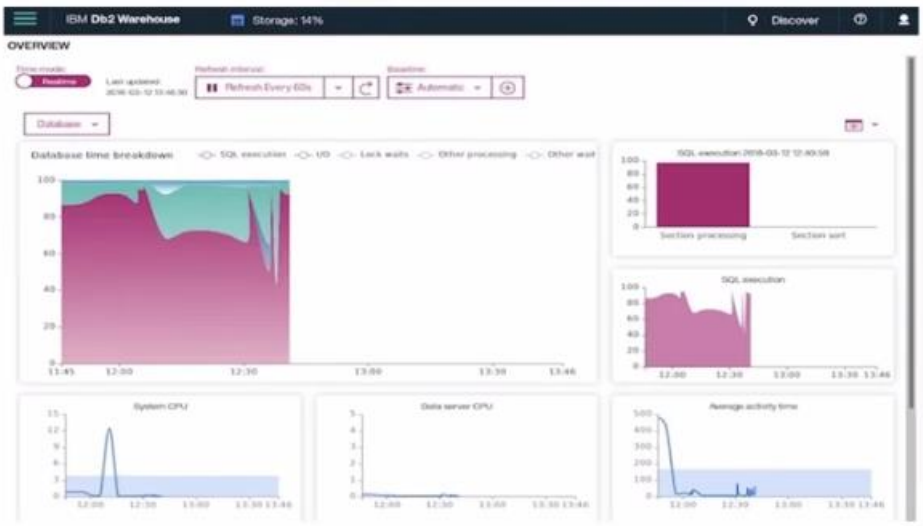


Performance & diagnostics widgets –

- Hardware and software issue counts, database alerts, and the amount of allotted storage used. You can also view a breakdown of how much time is spent in different states, such as waiting for locks and time to execute SQL queries, and a table of details regarding any database alert events.



- There are many other widgets available, such as system and data server CPU utilization history, and others.



IBM Db2 Warehouse use cases –

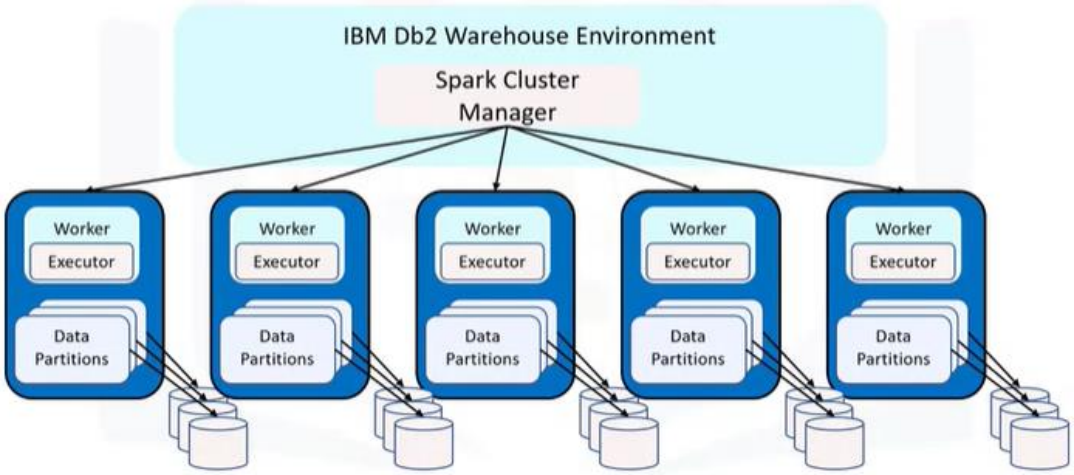
- High-availability requirements
- Cloud, on-premises, or hybrid hosting
- Consolidation and integration of data silos
- Accelerated development of data marts
- Management of sensitive or regulated data

- Storage of cold SQL data

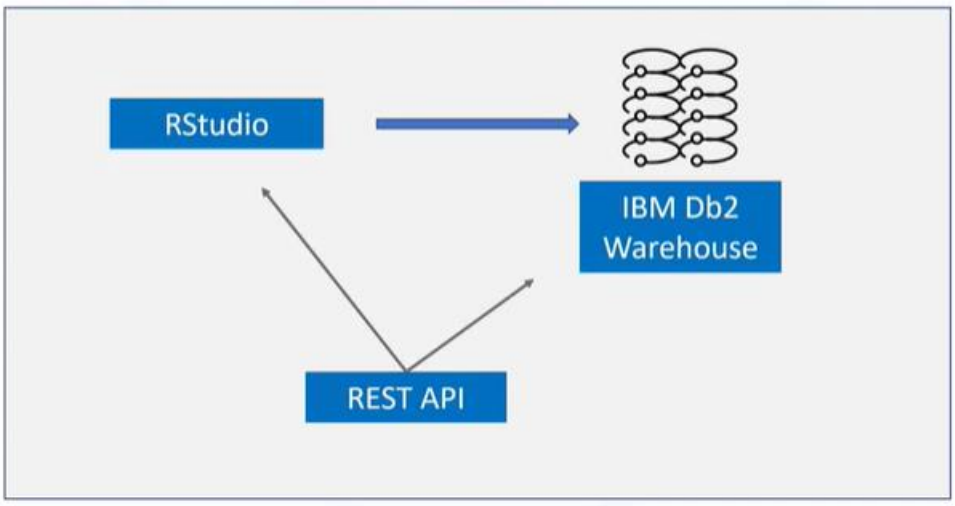
Integrations and plug-in support –

IBM Db2 Warehouse client and plugin support:

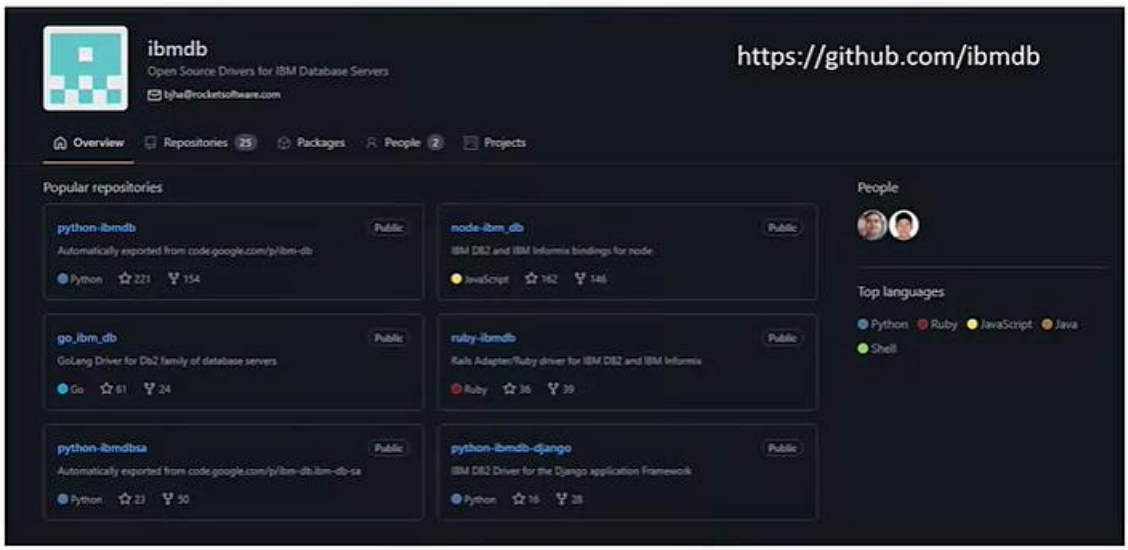
Integrations with Apache Spark –



Integrations with RStudio –



Open-source drivers on GitHub –



Summary: -

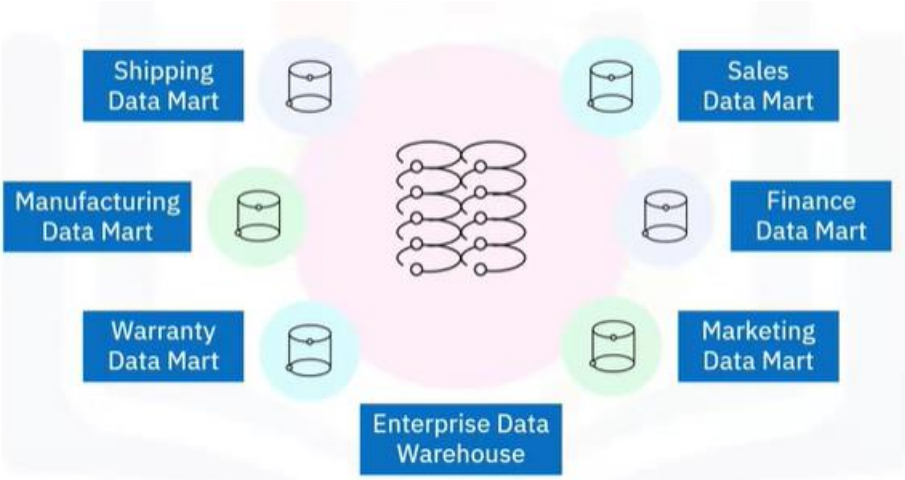
- IBM Db2 Warehouse is a cloud-ready, highly flexible data warehouse platform.
- Key features of IBM Db2 Warehouse include speed, scalability, automated schema generation, and built-in machine learning.
- Use cases include data integration and rapid development of data marts.
- IBM Db2 Warehouse integrates with JDBC, Apache Spark, Python, and R Studio.

5] Data Marts Overview

What is a data mart?

- A data mart is an isolated part of the larger enterprise data warehouse that is specifically built to serve a particular business function, purpose, or community of users.

For example, the sales and finance departments in a company may have access to dedicated data marts that supply the data required for their quarterly sales reports and projections. The marketing team may use data marts to analyze customer behavior data, and the shipping, manufacturing and warranty departments may have their own data marts.



What are data mart used for?

Data marts are used to:

- Provide support for tactical decision-making
- Help end users focus only on relevant data
- Save time otherwise spent searching the data warehouse for answers

Data mart structure –

- Relational database
- Star or snowflake schema
- Central fact table of business metrics
- Surrounded by associated dimension tables

Data repository comparisons -

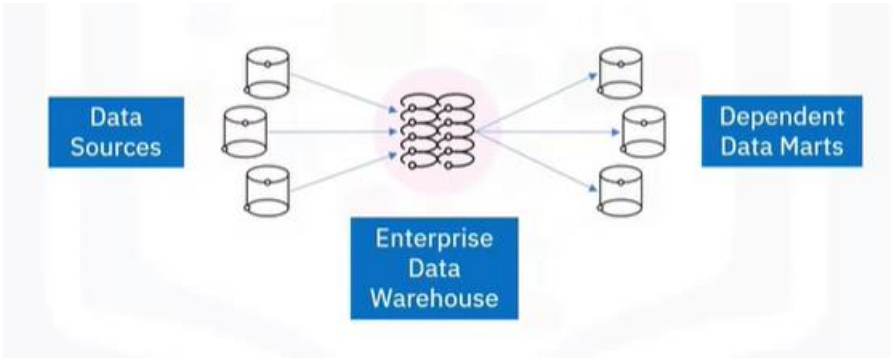
Data Marts	Databases
OLAP systems – read intensive	OLTP systems – write intensive
Use Txn DBs or warehouses as data sources	Use operational applications as sources of data
Contain clean, validated analytical data	Contain raw, unprocessed transactional data
Accumulate history for trend analysis	May not always store history

Data Marts	Data Warehouses
Small data warehouses with tactical scope	Large repositories with broad, strategic scope
Lean and fast	Large and slow

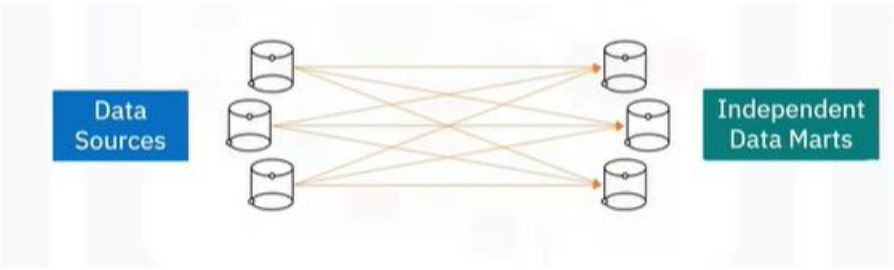
Types of data marts: –

Dependent, independent, and hybrid data marts

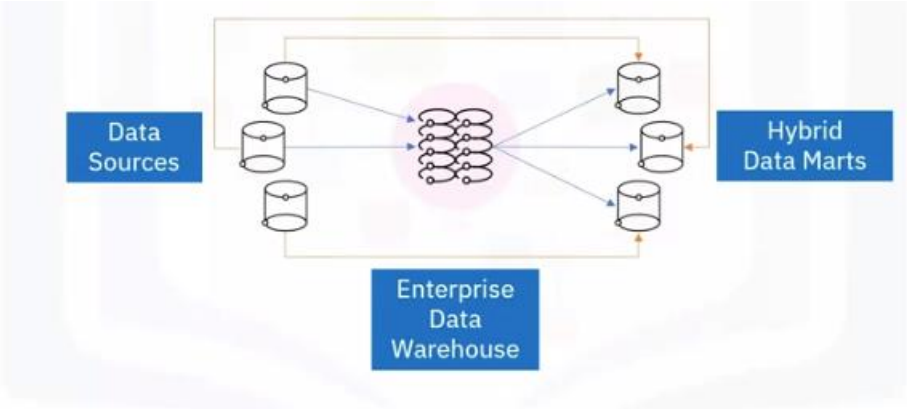
- The difference between these three kinds of data marts depends on their relationship with the data warehouse and the sources used for supplying each of them with data.
- Dependent data marts draw data from the enterprise data warehouse



- Independent data marts bypass the data warehouse and are created directly from sources, which may include internal operational systems or external data from vendors or other sources outside the enterprise.

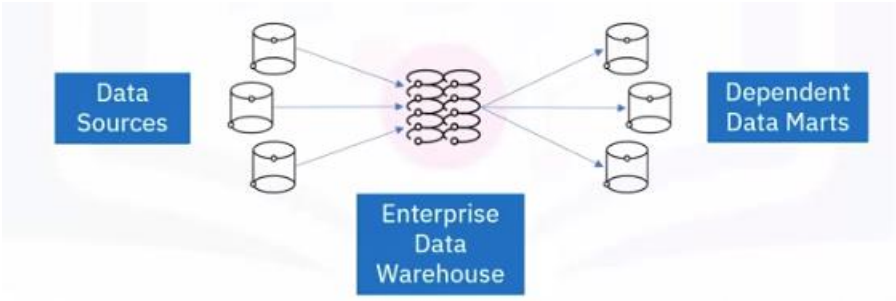


- Hybrid data marts only depend partially on the enterprise data warehouse. They combine inputs from data warehouses with data from operational systems and other systems external to the warehouse.



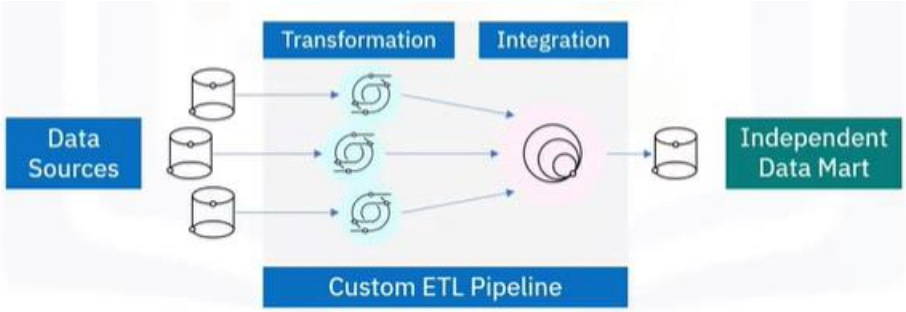
Dependent data marts –

- Dependent data marts offer analytical capabilities within a restricted area of the enterprise data warehouse.
- Inherit security from the EDW
- Use cleaned and transformed data
- Have simpler data pipelines



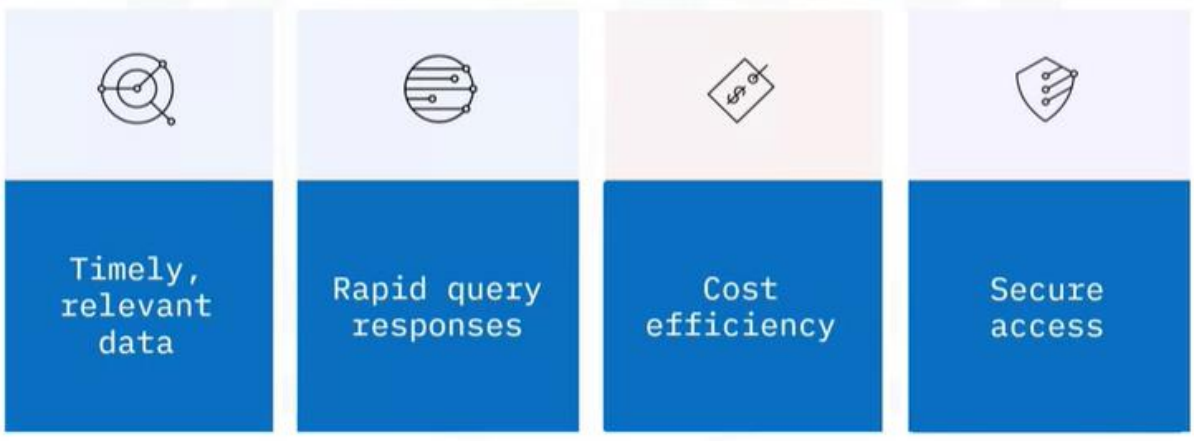
Independent data marts –

- Require custom ETL data pipelines
- May require additional security measures



Data mart purpose –

The purpose of a data mart is to provide:



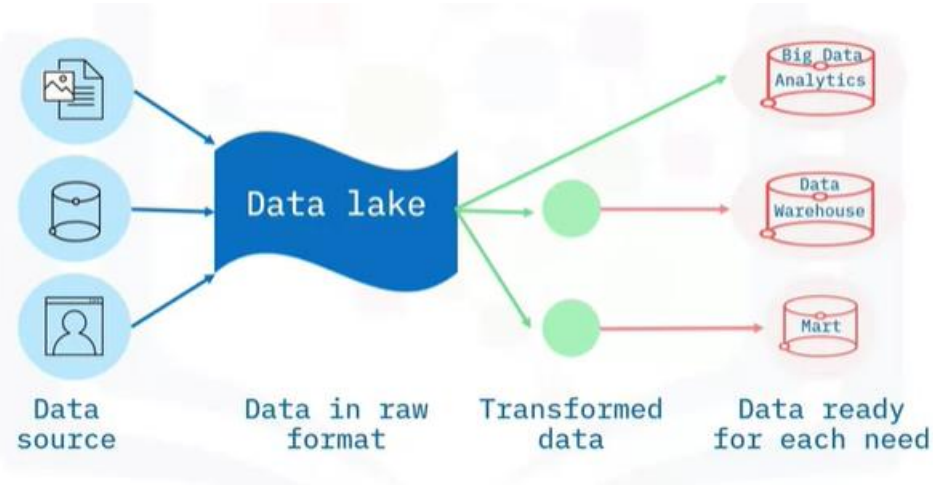
Summary: -

- Is an isolated part of the larger enterprise data warehouse that is specifically built to serve a particular business function, purpose, or community of users.
- Is designed to provide specific, timely, and rapid support for making tactical decisions.
- Typically has a star or snowflake schema. You also learned that unlike an OLTP database, an OLAP data mart stores clean and validated data and accumulates historical data.
- And, you can categorize data marts in terms of their dependence on the enterprise data warehouse.
- A data mart can be completely dependent on the data warehouse, a completely independent, standalone, mini data warehouse, or a hybrid of the two.

6] Data Lakes Overview

What is a data lake?

- A data lake is a storage repository that can store large amounts of structured, semi-structured, and unstructured data in their native format, classified and tagged with metadata.
- While a data warehouse stores data processed for a specific need, a data lake is a pool of raw data where each data element is given a unique identifier and is tagged with metatags for further use.
- You would opt for a data lake if you generate, or have access to, large amounts of data on an ongoing basis but don't want to be restricted to specific or pre-defined use cases.
- Data lakes are sometimes also used as a staging area for transforming data prior to loading into a data warehouse or a data mart.



- Store large amounts of structured, semi-structured, and unstructured data in their native format
- Data can be loaded without defining the structure or schema of data
- Use cases do not need to be known in advance
- Exist as a repository of raw data straight from the source
- A reference architecture that combines multiple technologies
- Can be deployed using
    - Cloud-object storage
    - Large-scale distributed systems
    - Relational database management systems
    - NoSQL data repositories

Data lake benefits –

- Handles all types of data
    - Unstructured
    - Semi-structured
    - Structured
- Scalable storage capacity
- Saves time that would have been used to define structures, create schemas, and transform data
- Can quickly repurpose data for a wide range of use cases

Vendors for data lakes –

- Amazon
- Cloudera
- Google
- IBM
- Informatica
- Microsoft
- Oracle
- SAS
- Snowflake
- Teradata
- Zaloni

Data lakes versus data warehouses –

Parameters	Data Lake	Data Warehouse
Data	Data is loaded in its raw and unstructured form	Data has been processed prior to loading
Schema	No need to define schema prior to loading	Schema designed prior to loading
Data Quality	• Any data that might not might not be curated • Data is agile and might not comply with governance guidelines	Data is cured and follows data governance practices
Users	Data scientists, data developers, machine learning engineers	• Business analysts • Data analysts

Summary: -

- A data lake is a storage repository that can store large amounts of structured, semi-structured, and unstructured data in their raw or native format, classified and tagged with metadata.
- You do not need to define the structure and schema of data before loading into the data lake.
- Data lakes offer several benefits, such as storage for all types of data, scalable storage capacity, time savings, and flexible data reuse.
- Finally, you learned that data lakes can be used as a kind of self-serve staging area for a variety of use cases, including machine learning development and advanced analytics.

Week 2

Designing, Modelling and Implementing Data Warehousing

Module 1: - Designing, Modelling and Implementing Data Warehousing

1] Overview of Data Warehouse Architectures

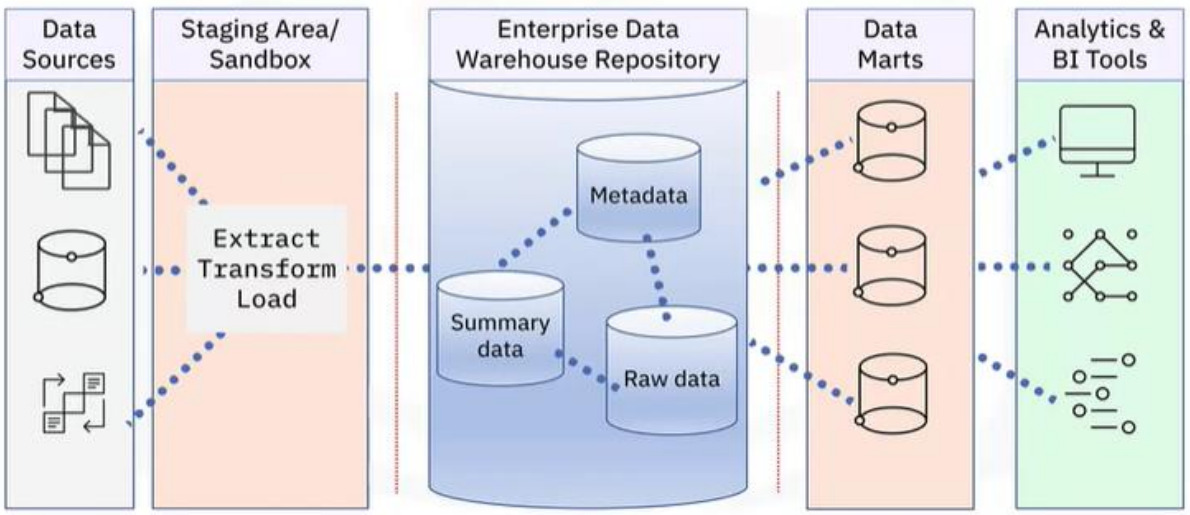
Data warehouse architecture –

Data warehouse architecture details depend on use cases:

- Report generation and dashboarding
- Exploratory data analysis
- Automation and machine learning
- Self-service analytics

General EDW architecture –

- Data sources, such as flat files, databases, and existing operational systems, an ETL layer for extracting, transforming, and loading data, optional staging and sandbox areas for holding data and developing workflows, an enterprise data warehouse repository, sometimes, data marts, which are known as a "hub and spoke" architecture when multiple data marts are involved, and an analytics layer and business intelligence tools.
- Data warehouses also enforce security for incoming data and data passing through to further stages and users throughout the network.



EDW reference architectures –

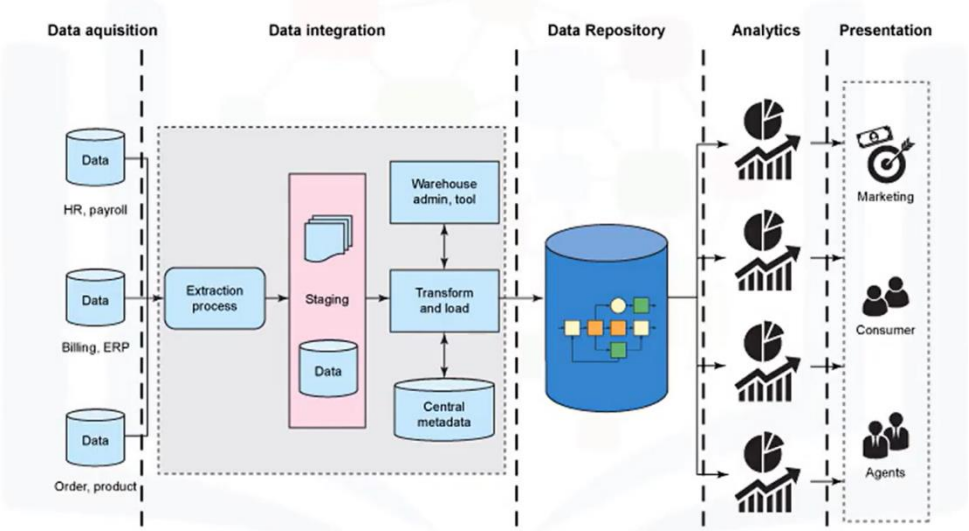
Vendor specific reference architectures:

Enterprise data warehouse vendors often create proprietary reference architecture and implement template data warehousing solutions that are variations on this general architectural model. A data warehousing platform is a complex environment with lots of moving parts. Thus, interoperability among components is vital. Vendor-specific reference architecture typically incorporates tools and products from the vendor's ecosystem that work well together.

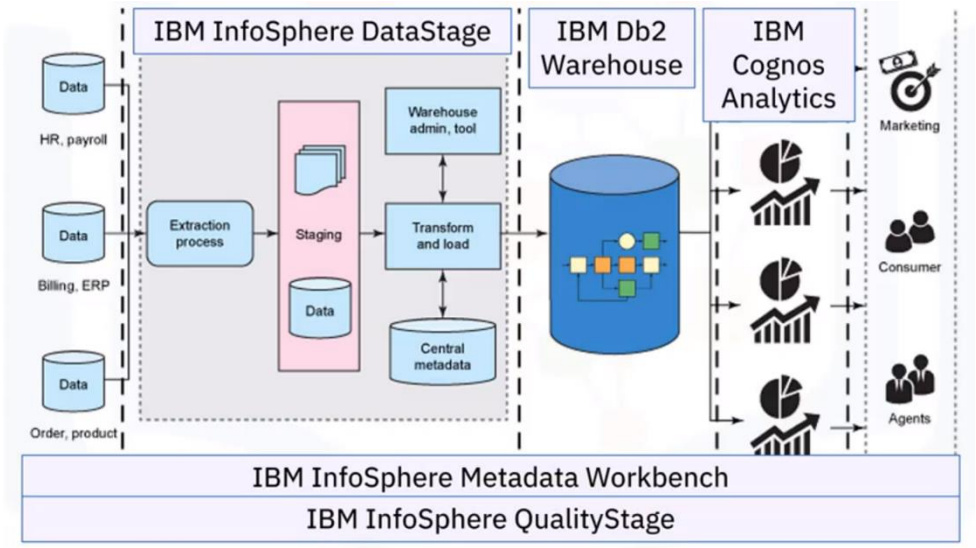
- Adaption of general model
- Interoperability
- Tool integrations tested

IBM reference architecture –

- Each layer of the architecture performs a specific function:
  - The data acquisition layer consists of components to acquire raw data from source systems, such as human resources, finance, and billing departments.
  - The data integration layer, essentially a staging area, has components for extracting the data, transforming it, and loading it into the data repository layer. It also houses administration tools and central metadata.
  - The data repository layer stores the integrated data, typically employing a relational model.
  - The analytics layer often stores data in a cube format to make it easier for users to analyze it.
  - And, the final presentation layer incorporates applications that provide access for different sets of users, such as marketing analysts, users, and agents. Applications consume the data through web pages and portals defined in the reporting tool or through web services.



- IBM reference architecture is supported and extended using several products from the IBM InfoSphere suite.
- IBM InfoSphere DataStage is a scalable ETL platform that delivers near real-time integration of all data types, on-premises, and in cloud environments.
- IBM InfoSphere MetaData Workbench provides end-to-end data flow reporting and impacts analysis of information assets in an environment that allows organizations to share easily, locate, and retrieve information from these systems. Use the built-in data flow reporting capabilities to monitor how IBM InfoSphere DataStage moves and transforms your data.
- IBM InfoSphere QualityStage, designed to support your data quality and information governance initiatives, enables you to investigate, cleanse, and manage your data.
- This solution helps you create and maintain consistent views of key entities, including customers, vendors, locations, and products.
- IBM Db2 Warehouse is a family of highly performant, scalable, and reliable data management products that manage both structured and unstructured data across on-premises and cloud environments.
- And finally, IBM Cognos Analytics is an advanced business intelligence platform that generates reports, scoreboards, and dashboards, performs exploratory data analysis, and even curates and joins your data using multiple sources.



Summary: -

- An architectural model for a general data warehousing platform includes data sources, ETL pipelines, optional staging and sandbox areas, an enterprise data warehouse repository, optional data marts, and analytics and business intelligence tools.

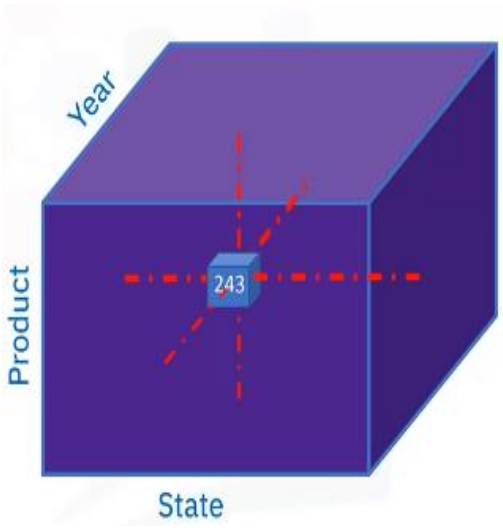
- Companies can modify general enterprise data warehouse architecture to suit their analytics requirements.
- Vendors offer proprietary reference architecture based on the general model, which they test for interoperability among components.
- An IBM enterprise data warehouse solution combines InfoSphere with Db2 Warehouse and Cognos Analytics.

2] Cubes, Rollups, and Materialized Views and Tables

What is a data cube?

- Example: Sales OLAP cube
- Coordinates = dimensions
- Cells = facts

- The coordinates of the cube are defined by a set of dimensions, which are selected from the star schema.
- In this illustration, we are only showing three dimensions, but data cubes can have many dimensions. We have the Product categories corresponding to the items sold, the State or Province the items were sold from, and the Year these products were sold in. The cells of the cube are defined by a fact of interest from the schema, which could be something like "total sales in thousands of dollars." Here the "243" indicates "243 thousand dollars" for some given Product, State, and Year combination.



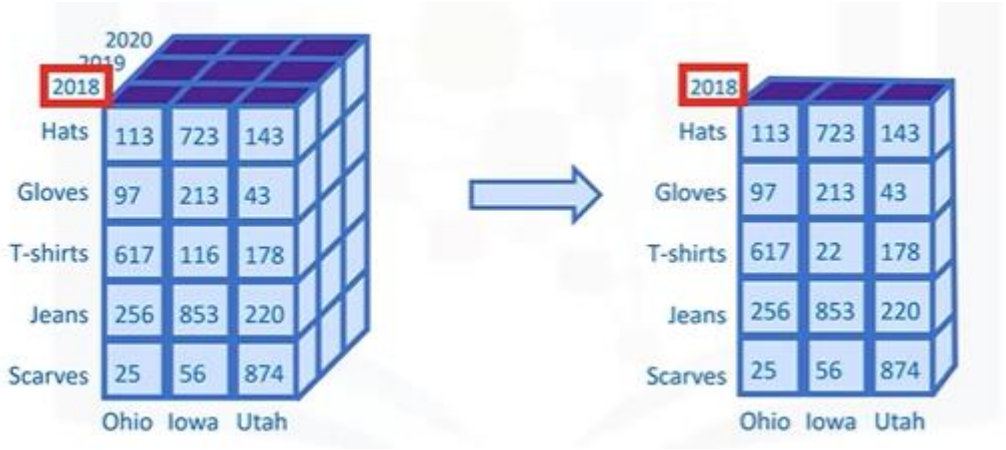
- Cube Operations:
  - Slicing
  - Dicing
  - Drilling up and down
  - Pivoting
  - Rolling up

Slicing data cubes –

- Slicing a data cube involves selecting a single member from a dimension, which yields a data cube that has one dimension less than the original.

For example, you can slice this sales cube by selecting only the year 2018 from the year dimension, allowing you to analyze sales totals for all sales states and all products for the year 2018.

Slicing reduces cube dimension by 1:

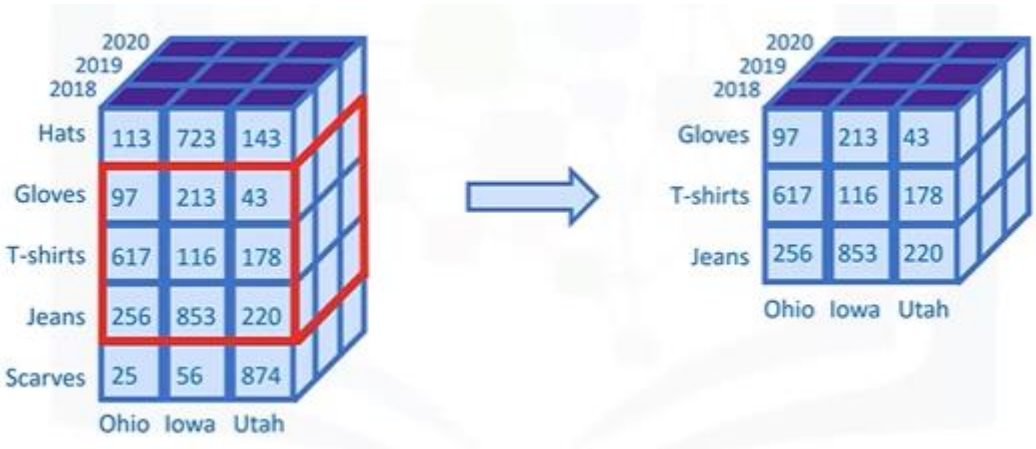


Dicing data cubes –

- Dicing a cube involves selecting a subset of values from a dimension, effectively shrinking it.

For example, you can dice this sales cube by selecting only “Gloves”, “T-shirts”, and “Jeans” from the Product-Type dimension, allowing you to restrict your view to just those product types.

Dicing shrinks a dimension:

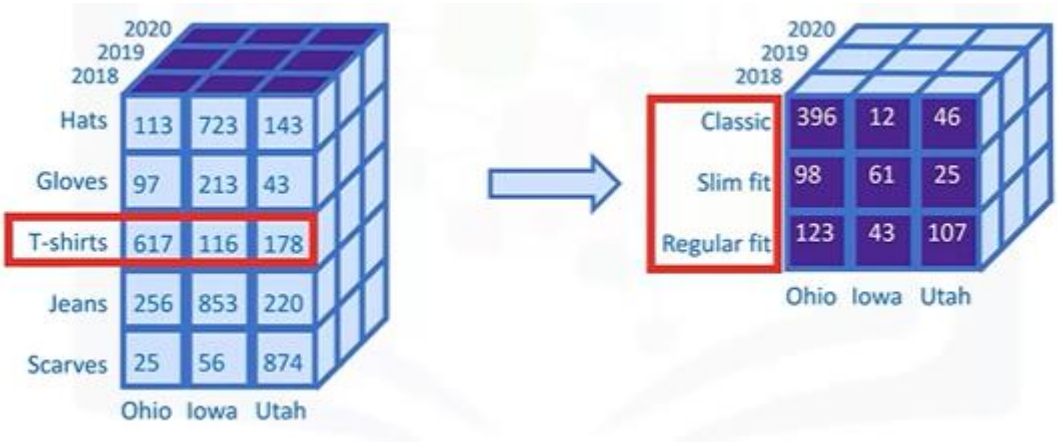


Drilling up or down in data cubes –

- In snowflake schema, you will find hierarchies, or subcategories within some of your dimensions that you can drill into.
- Drilling up is just the reverse process, which would take you back to the original data cube.

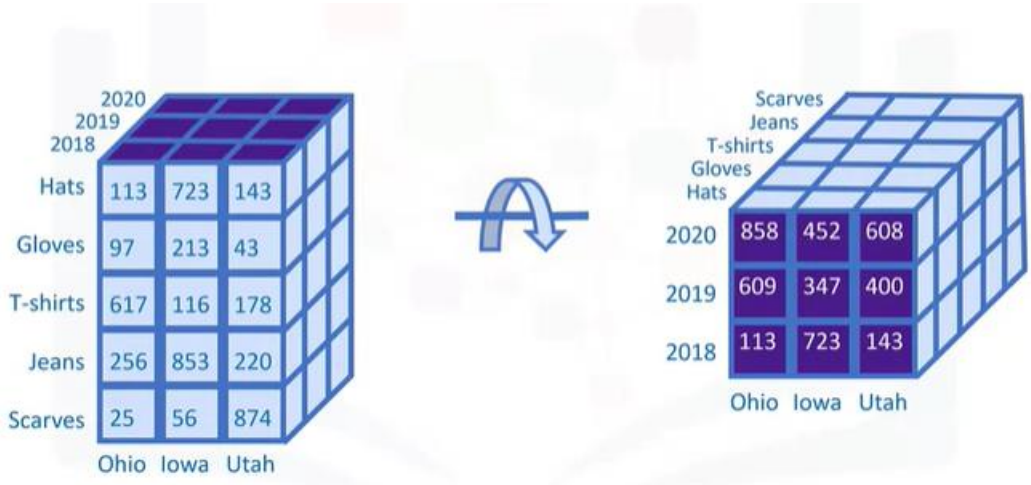
Thus, for example, you can "drill down" into a particular member of the "Product category" dimension, such as "T-shirts," resulting in this view, which may include more specific "product groups" such as "Classic," "Slim fit," and "Regular fit."

Drilling into subcategories within a dimension:



Pivoting data cubes –

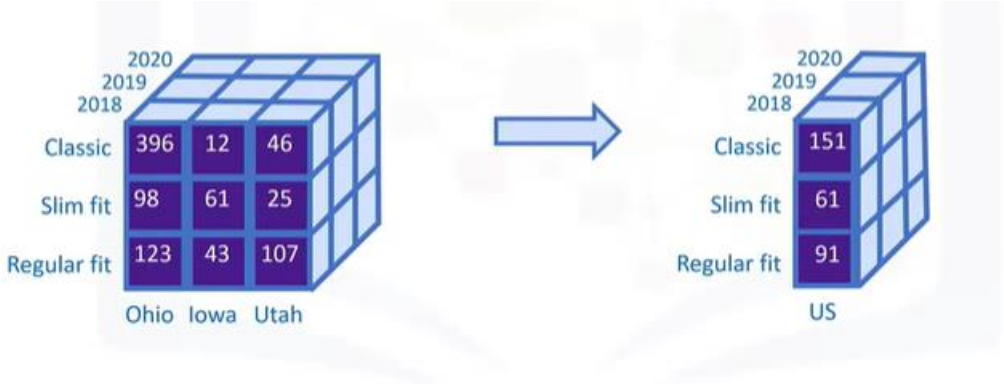
- Pivoting data cubes is straightforward. It involves a rotation of the data cube.
- In this case, the year and product dimensions have been interchanged, while the State dimension has been fixed "as is."
- Pivoting doesn't change its information content; it just changes the point of view you may choose to analyze it from.



Rolling up in data cubes –

- Roll up = summarize a dimension
- Aggregate using COUNT, MIN, MAX, SUM, AVERAGE

For example, you could calculate the average selling price of Classic, Slim fit, and Regular fit T-shirts by summing horizontally over the three US states and dividing by three.



<u>Materialized views</u> –

- A "snapshot" containing results of a query.
- Used to replicate data in a staging database, or
- Precompute expensive queries for a data warehouse
- Automatically keep query results synced to database
- Safely work without affecting source database
- Can be set up for different refresh options:
  - Never – populated on creation only
  - Upon request – manually or scheduled
  - Immediately -automatically, after every statement

Materialized views in Oracle –

Create a materialized view in Oracle:

```
CREATE MATERIALIZED VIEW MY_MAT_VIEW
REFRESH FAST START WITH SYSDATE
NEXT SYSDATE + 1
AS SELECT * FROM <my_table_name>;
```

Materialized views in PostgreSQL –

Create a materialized view in PostgreSQL:

```
CREATE MATERIALIZED VIEW MY_MAT_VIEW
[ WITH (storage_parameter [= value] [, ... ]) ]
 [ TABLESPACE tablespace_name ]
  AS SELECT * FROM <table_name>;

REFRESH MATERIALIZED VIEW MY_MAT_VIEW;
```

Materialized views in Db2 –

In Db2, materialized views are called MQTs

```
create table emp as (select e.empno, e.firstnme, e.lastname, e.phoneno, d.deptno,
  substr(d.deptname, 1, 12) as department, d.mgrno from employee e, department d
    where e.workdept = d.deptno)
      data initially deferred refresh immediate

immediate checked not incremental
```

```
select * from emp

EMPNO  FIRSTNME     LASTNAME        PHONENO DEPTNO DEPARTMENT   MGRNO
------ ------------ --------------- ------- ------ ------------ ------
000010 CHRISTINE    HAAS            3978    A00    SPIFFY COMPU 000010
000020 MICHAEL      THOMPSON        3476    B01    PLANNING     000020
000030 SALLY        KWAN            4738    C01    INFORMATION  000030
000050 JOHN         GEYER           6789    E01    SUPPORT SERV 000050
000060 IRVING       STERN           6423    D11    MANUFACTURIN 000060

5 record(s) selected.  connect reset
```

Summary: -

- A data cube represents a star or snowflake schema's dimensions as coordinates, plus a fact from the schema to populate its cells with values.
- Many operations can be applied to data cubes, such as: drilling down into hierarchical dimensions, slicing, dicing, and rolling up.
- Materialized views can be used to replicate data or to precompute expensive queries.
- And finally, modern enterprise data warehouse tools, such Oracle and Db2, allow you to automatically keep your material views up-to-date.

Grouping Sets in SQL

The GROUPING SETS clause is used in conjunction with the GROUP BY clause to allow you to easily summarize data by aggregating a fact over as many dimensions as you like.

SQL GROUP BY clause

Recall that the SQL GROUP BY clause allows you to summarize an aggregation such as SUM or AVG over the distinct members, or groups, of a categorical variable or dimension.

You can extend the functionality of the GROUP BY clause using SQL clauses such as CUBE and ROLLUP to select multiple dimensions and create multi-dimensional summaries. These two clauses also generate grand totals, like a report you might see in a spreadsheet application

or an accounting style sheet. Just like CUBE and ROLLUP, the SQL GROUPING SETS clause allows you to aggregate data over multiple dimensions but does not generate grand totals.

Examples

Let's start with an example of a regular GROUP BY aggregation and then compare the result to that of using the GROUPING SETS clause. We'll use data from a fictional company called Shiny Auto Sales. The schema for the company's warehouse is displayed in the entity-relationship diagram in Figure 1.

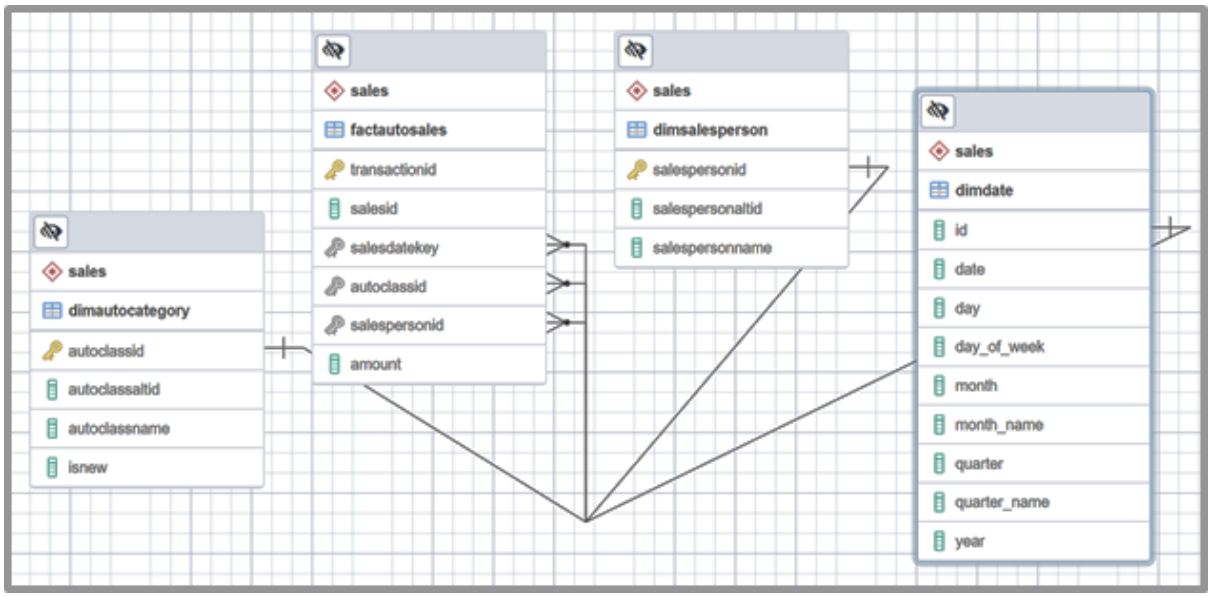


Fig. 1. Entity-relationship diagram for a "sales" star schema based on the fictional "Shiny Auto Sales" company.

We'll work with a convenient materialized view of a completely denormalized fact table from the sales star schema, called DNsales, which looks like the following:

```
SELECT * FROM DNsales LIMIT 5;

    date    | autoclassname | isnew |   salespersonname    |   amount
------------+---------------+-------+----------------------+------------
 2021-01-04 | 4 Door Sedan  | t     | Jake Salesbouroughs | $42,000.00
 2021-01-04 | Truck         | t     | Gocart Joe          | $17,680.00
 2021-01-05 | Truck         | t     | Jake Salesbouroughs | $37,100.00
 2021-01-05 | Midsize SUV   | t     | Cadillac Jack       | $26,500.00
 2021-01-05 | Truck         | f     | Jane Honda          |  $8,200.00
(5 rows)
```

This DNsales table was created by joining all the dimension tables to the central fact table and selecting only the columns which are displayed. Each record in DNsales contains details for an individual sales transaction.

Example 1

Consider the following SQL code which invokes GROUP BY on the auto class dimension to summarize total sales of new autos by auto class.

```
SELECT
    autoclassname,
    SUM(amount)
FROM
    DNsales
WHERE
    isNew=True
GROUP BY
    autoclassname
```

The result looks like this:

```
 autoclassname |     sum
---------------+-------------
 Compact SUV   | $168,598.00
 Truck         |  $96,879.00
 Midsize SUV   |  $58,599.00
 4 Door Sedan  |  $42,000.00
```

Example 2

Now suppose you want to generate a similar view, but you also want to include the total sales by salesperson. You can use the GROUPING SETS clause to access both the auto class and salesperson dimensions in the same query. Here is the SQL code you can use to summarize total sales of new autos, both by auto class and by salesperson, all in one expression:

```
SELECT
    autoclassname,
    salespersonname,
    SUM(amount)
FROM
    DNsales
WHERE
    isNew=True
GROUP BY
GROUPING SETS(autoclassname,
              salespersonname)
```

Here is the query result. Notice that the first four rows are identical to the result of Example 1, while the next 5 rows are what you would get
by substituting salespersonname for autoclassname in Example 1.

```
 autoclassname |   salespersonname    |     sum
---------------+----------------------+-------------
 Compact SUV   |                      | $168,598.00
 Truck         |                      |  $96,879.00
 Midsize SUV   |                      |  $58,599.00
 4 Door Sedan  |                      |  $42,000.00
               | Happy Dollarmaker    |  $74,500.00
               | Jake Salesbouroughs  | $121,199.00
               | Cadillac Jack        |  $26,500.00
               | William Jeepman      |  $51,999.00
               | Gocart Joe           |  $91,878.00
```

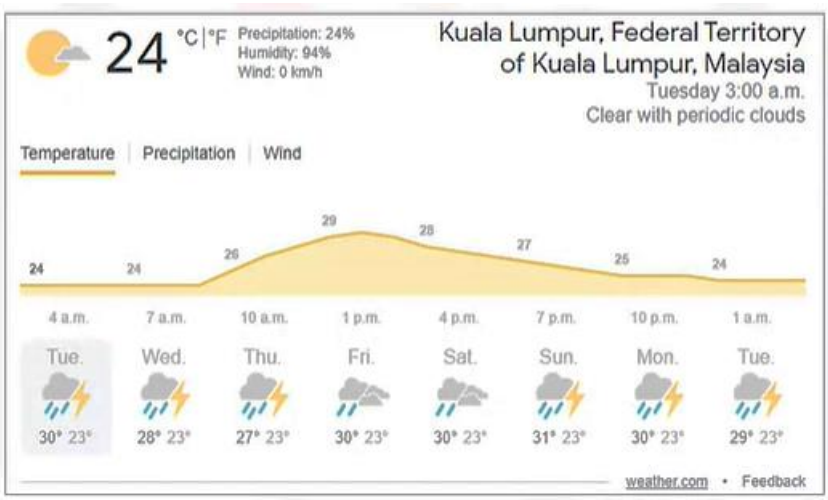
Essentially, applying GROUPING SETS to the two dimensions, salespersonname and autoclassname, provides the same result that you would get by appending the two individual results of applying GROUP BY to each dimension separately as in Example 1.

3] Facts and Dimensional Modelling

Facts and dimensions –

- Data can be categorized as facts and dimensions
- Facts are usually measured quantities, such as temperature, number of sales, or mm of rainfall
- Facts can also be qualitative
- Dimensions are attributes relating to facts
- Dimensions provide context to facts

For example, a temperature such as “24 degrees celsius,” all by itself, is not meaningful information. Let’s look at a familiar example. Here we have a weather report obtained by googling “weather in Kuala Lumpur.” The facts here are things like the temperature, humidity, probability of precipitation, and wind speed. Notice that these are all quantities. But there are other facts here, such as this icon, which means “partly cloudy,” or the statement “Clear with periodic clouds,” and these other icons which indicate forecasted conditions, such as thundershowers. All of these non-numeric facts are examples of qualitative facts. To make sense of these facts, we need to provide context. Context is provided by dimensions, which includes things like the location, “Kuala Lumpur, Malaysia” and the day and time, “Tuesday at 3 a.m.” The statement “24°C in Kuala Lumpur, Malaysia on Tuesday, August 17th at 3:00 a.m.” means something.

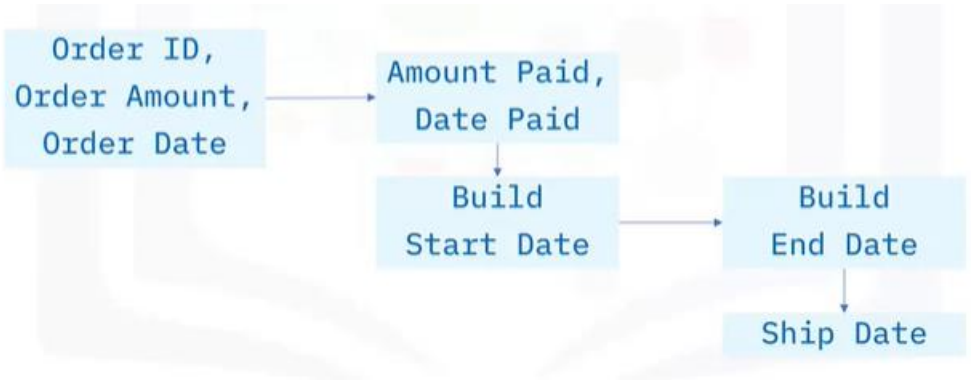


Facts tables –

- Facts of a business process, plus
- Foreign keys to dimension tables
- Dollar amounts for sales transactions
- Can contain detail level facts, or
- Facts that have been aggregated
- Summary tables contain aggregated facts
    - "Quarterly Sales" summary table, with
    - "store_id" as foreign key

Accumulating snapshot fact tables –

- Used to record events during a well-defined business process



Dimensions –

- Dimensions categorize facts
- Called categorial variables in stats and machine learning
- Used to answer business questions
- Used for filtering, grouping, and labelling

    - People, product, and place names, and date or timestamps
- Dimension table stores dimensions of a fact
    - Joined to fact table via foreign key

Dimension table examples –

Product tables:

- Make, model, color, size

Employee tables:

- Name, title, department

Temporal tables:

- Date/time at granularity of recorded events

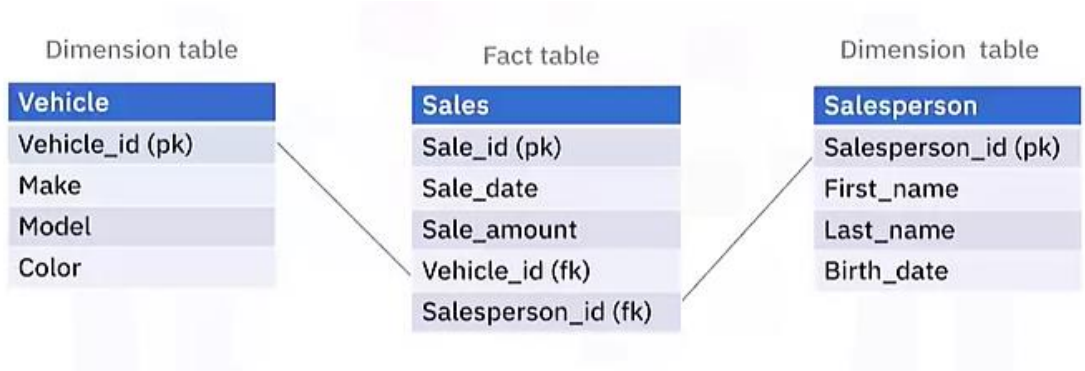
Geography tables:

- Country, state, city, region

Example schema with fact and dimension tables –

- Each fact table typically has multiple dimension tables related to it.

We could have a fact table for recording sales at a car dealership This table would store facts about each car sale such as the Sale date, and Sale amount as well as a primary key "Sale ID." For each sales transaction, we also need to record its dimensions like the Vehicle sold and the Salesperson who sold it. The attributes of these dimensions, such as the vehicle Make and Model or the Salesperson's First and Last name are stored in separate Vehicle and Salesperson tables However, we link them with the Sales table by recording the "Vehicle ID" and "Salesperson ID" as foreign keys in the Fact table. This way we typically end up with multiple dimension tables for each fact table.



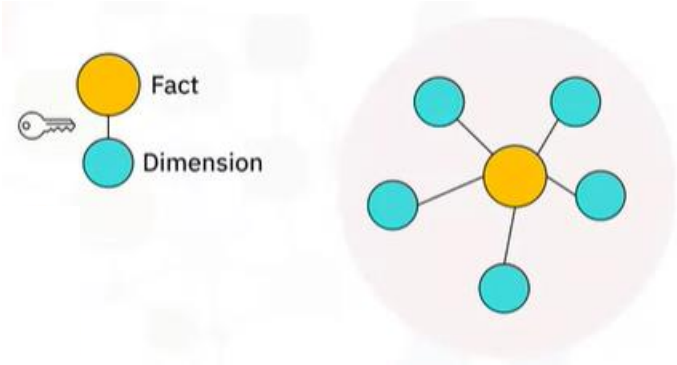
Summary: -

- Business data falls into two categories: Facts and dimensions.
- Facts are usually numerical measures of business processes, such as sale amounts in dollars.
- Dimensions such as ‘sold by’ and ‘store ID’ categorize facts, such as who sold the product, when, and which store it was sold from.
- Dimensions are categorical variables that provide context for facts and are used for filtering, grouping, and labeling.
- Facts and dimension tables are linked together by foreign and primary keys.

4] Data Modelling using Star and Snowflake Schemas

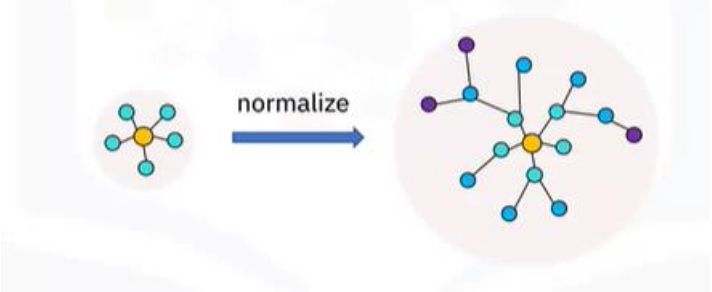
Star schemas –

- Keys connect facts with dimensions
- Dimensions ‘radiate’ from a central fact.
- Graph whose edges are relations between facts and dimensions



Snowflake schema –

- Snowflake schemas are normalized star schemas
- Dimension tables split into child tables
- Snowflakes don’t need to be fully normalized



Modelling with a star schema –

Design consideration –

1. Select a business process
2. Choose level of details
3. Identify the dimensions
4. Identify the facts

Data warehouse architecture: -

A2Z Discount Warehouse – Data Ops Engineer

For example, that you are a data engineer helping to lay out the data ops for a new store called "A to Z Discount Warehouse."

- **POS Transaction –**

  They would like you to develop a data plan to capture everyday POS, or point-of-sales transactions that happen at the till, where customers have their items scanned and pay for them. Thus, "point-of-sale transactions" is the business process that you want to model.

- **Individual line items –**

  The finest granularity you can expect to capture from POS transactions comes from the individual line items, which is included in the detailed information you can see on a typical store receipt. This is precisely what "A to Z" is interested in capturing.

- **Date, time, store, product, cashier –**

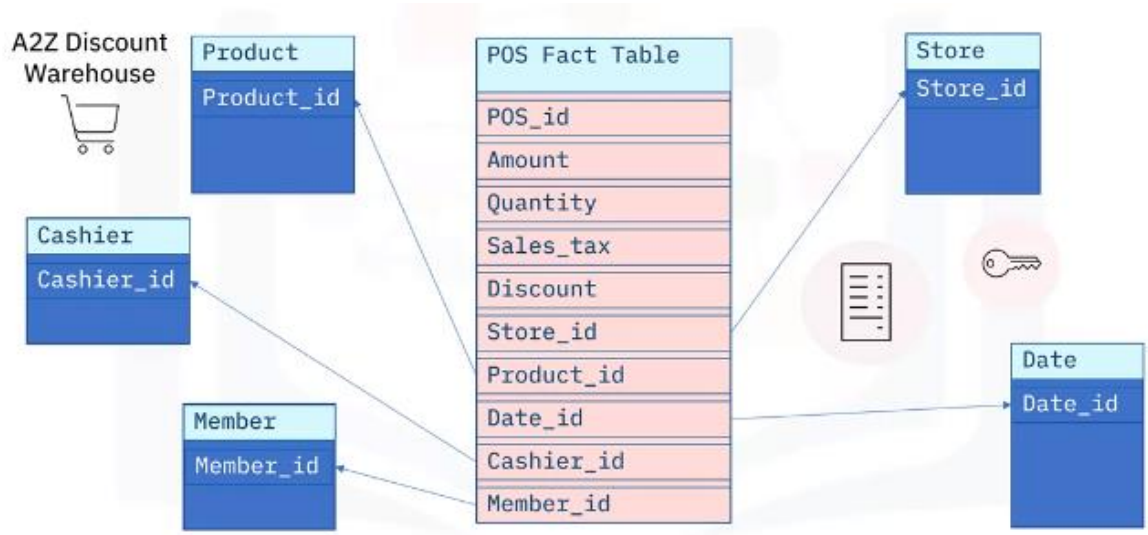
  The next step in the process is to identify the dimensions. These include attributes such as the date and time of the purchase, the store name, the products purchased, and the cashier who processed the items. You might add other dimensions, like "payment method," whether the line item is a return or a purchase, and perhaps a "customer membership number."

- **Amount, quantity, discount, sales tax –**

  Now it's time to consider the facts. Thus, you identify facts such as the amount for each item's price, the quantity of each product sold, any discounts applied to the sale, and the sales tax applied. Other facts to consider include environmental fees, or deposit fees for returnable containers.

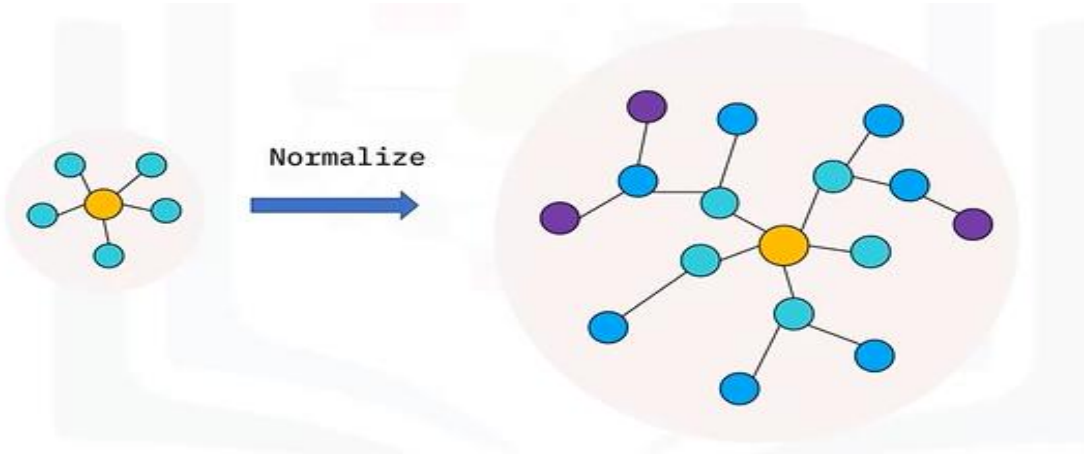
Point-of-sale star –

At the center of your star schema sits a "point-of-sales fact table," which contains a unique ID, called "P O S ID," for each line item in the transaction, plus the following facts, or measures: the amount of the transaction in dollars, the quantity, or number of items involved, the sales tax, and any discount applied. There may be other facts to include, but these can be added later as you discover them. Each line item from a sales transaction has many dimensions associated with it. You include them as foreign keys in your fact table, or as links to the primary keys of your dimension tables. For example, the name of the

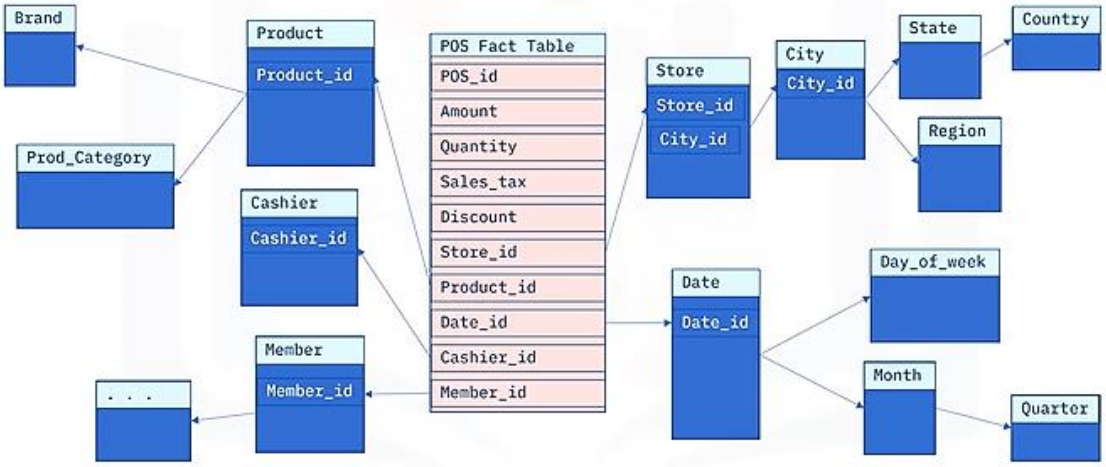


store at which the item was sold is kept in a dimension table called "store," which is identified in the fact-table by the value of the foreign "Store ID" key, which is the primary key for the Store table. Product information is stored in the Product table, which is uniquely identified by the "ProductID" key. Similarly, the date of the transaction is keyed by the "Date ID," which cashier entered the transaction is keyed by the "Cashier ID," and which member was involved is indicated by the "Member ID."

From star schema to snowflake schema –



Point-of-sale snowflake –



Summary: -

- Facts and dimension tables, together with foreign and primary keys, are used to form star and snowflake modelling schemas.
- Design considerations for data modelling with star schema include identifying a business process, its granularity, and its facts and dimensions.
- Snowflake schemas can be described as normalized star schemas.
- Normalization involves separating dimension tables into individual tables defined by levels or hierarchies of the parent dimension and reduces storage footprint.

Reading: Data Warehousing using Star and Snowflake schemas

Why do we use these schemas, and how do they differ?

Star schemas are optimized for reads and are widely used for designing data marts, whereas snowflake schemas are optimized for writes and are widely used for transactional data warehousing. A star schema is a special case of a snowflake schema in which all hierarchical dimensions have been denormalized or flattened.

Attribute	Star schema	Snowflake schema
Read speed	Fast	Moderate
Write speed	Moderate	Fast
Storage space	Moderate to high	Low to moderate
Data integrity risk	Low to moderate	Low

Query complexity	Simple to moderate	Moderate to complex
Schema complexity	Simple to moderate	Moderate to complex
Dimension hierarchies	Denormalized single tables	Normalized over multiple tables
Joins per dimension hierarchy	One	One per level
Ideal use	OLAP systems, Data Marts	OLTP systems

Table 1. A comparison of star and snowflake schema attributes.

Normalization reduces redundancy

Both star and snowflake schemas benefit from the application of normalization. "Normalization reduces redundancy" is an idiom that points to a key advantage leveraged by both schemas. Normalizing a table means to create, for each dimension:

1. A surrogate key to replace the natural key, that is, the unique values of the given column, and
2. A lookup table to store the surrogate and natural key pairs.

Each surrogate key's values are repeated exactly as many times within the normalized table as the natural key was before moving the natural key to its new lookup table. Thus, you did nothing to reduce the redundancy of the original table.

However, dimensions typically contain groups of items that appear frequently, such as a "city name" or "product category". Since you only need one instance from each group to build your lookup table, your lookup table will have many fewer rows than your fact table. If there are child dimensions involved, then the lookup table may still have some redundancy in the child dimension columns. In other words, if you have a hierarchical dimension, such as "Country", "State", and "City", you can repeat the process on each level to further reduce the redundancy. Notice that further normalizing your hierarchical dimensions has no effect on the size or content of your fact table - star and snowflake schema data models share identical fact tables.

Normalization reduces data size

When you normalize a table, you typically reduce its data size, because in the process you likely replace expensive data types, such as strings, with much smaller integer types. But to preserve the information content, you also need to create a new lookup table that contains the original objects. The question is, does this new table use less storage than the savings you just gained in the normalized table? For small data, this question is probably not worth considering, but for big data, or just data that is growing rapidly, the answer is yes, it is inevitable. Indeed, your fact table will grow much more quickly than your dimension tables, so normalizing your fact table, at least to the minimum degree of a star schema is likely warranted. Now the question is about which is better – star or snowflake?

Comparing benefits: snowflake vs. star data warehouses

The snowflake, being completely normalized, offers the least redundancy and the smallest storage footprint. If the data ever changes, this minimal redundancy means the snowflaked data needs to be changed in fewer places than would be required for a star schema. In other words, writes are faster, and changes are easier to implement. However, due to the additional joins required in querying the data, the snowflake design can have an adverse impact on read speeds. By denormalizing to a star schema, you can boost your query efficiency. You can also choose a middle path in designing your data warehouse. You could opt for a partially normalized schema. You could deploy a snowflake schema as your basis and create views or even materialized views of denormalized data. You could for example simulate a star schema on top of a snowflake schema. At the cost of some additional complexity, you can select from the best of both worlds to craft an optimal solution to meet your requirements.

Practical differences

Most queries you apply to the dataset, regardless of your schema choice, go through the fact table. Your fact table serves as a portal to your dimension tables. The main practical difference between star and snowflake schema from the perspective of an analyst has to do with querying the data. You need more joins for a snowflake schema to gain access to the deeper levels of the hierarchical dimensions, which can reduce query performance over a star schema. Thus, data analysts and data scientists tend to prefer the simpler star schema. Snowflake schemas are generally good for designing data warehouses and in particular, transaction processing systems, while star schemas are better for serving data marts, or data warehouses that have simple fact-dimension relationships. For example, suppose you have point-of-sale records accumulating in an Online Transaction Processing System (OLTP) which are copied as a daily batch ETL process to one or more Online Analytics Processing (OLAP) systems where subsequent analysis of large volumes of historical data is carried out. The OLTP source might use a snowflake schema to optimize performance for frequent writes, while the OLAP system uses a star schema to optimize for frequent reads. The ETL pipeline that moves the data between systems includes a denormalization step which collapses each hierarchy of dimension tables into a unified parent dimension table.

Too much of a good thing?

There is always a tradeoff between storage and compute that should factor into your data warehouse design choices. For example, do your end-users or applications need to have precomputed, stored dimensions such as ‘day of week’, ‘month of year’, or ‘quarter’ of the year? Columns or tables which are rarely required are occupying otherwise usable disk space. It might be better to compute such dimensions within your SQL statements only when they are needed. For example, given a star schema with a date dimension table, you could apply the SQL ‘MONTH’ function as MONTH(dim_date.date_column) on demand instead of joining the precomputed month column from the MONTH table in a snowflake schema.

Scenario

Suppose you are handed a small sample of data from a very large dataset in the form of a table by your client who would like you to take a look at the data and consider potential schemas for a data warehouse based on the sample. Putting aside gathering specific requirements for the moment, you start by exploring the table and find that there are exactly two types of columns in the dataset - facts and dimensions. There are no foreign keys although there is an index. You think of this table as being a completely denormalized, or flattened dataset.
You also notice that amongst the dimensions are columns with relatively expensive data types in terms of storage size, such as strings for names of people and places.
At this stage you already know you could equally well apply either a star or snowflake schema to the dataset, thereby normalizing to the degree you wish. Whether you choose star or snowflake, the total data size of the central fact table will be dramatically reduced. This is because instead of using dimensions directly in the main fact table, you use surrogate keys, which are typically integers; and you move the natural dimensions to their own tables or hierarchy of tables which are referenced by the surrogate keys. Even a 32-bit integer is small compared to say a 10-character string (8 X 10 = 80 bits).

5] Staging Areas for Data Warehouses

Data warehouse staging areas –

A staging area is:

- Intermediate storage for ETL processing
- Bridge between data sources and the target system
- Sometimes transient
- Sometimes held for archiving or troubleshooting
- Used to optimize and monitor ETL jobs

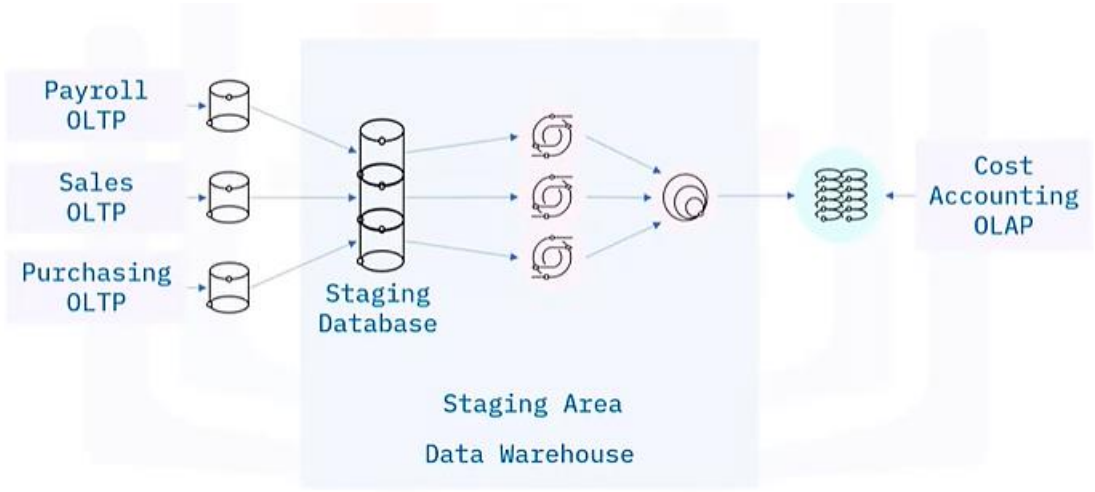
Staging can be implemented as:

- Flat files + Python
- SQL Tables + Db2
- Database + Cognos

DW staging area example –

Imagine the enterprise would like to create a dedicated “Cost Accounting” Online Analytical Processing system. The required data is managed in separate Online Transaction Processing Systems within the enterprise, from the Payroll, Sales, and Purchasing departments. From these siloed systems, the data is extracted to individual Staging Tables, which are created in the Staging Database. Data from these tables is then

transformed in the Staging Area using SQL to conform it to the requirements of the Cost Accounting system. The conformed tables can now be integrated, or joined, into a single table. The final phase is the loading phase, where the data is loaded into the target cost-accounting system.



Functions of a staging area –

Some typical ones include:

- **Integration:** Indeed, one of the primary functions performed by a staging area is consolidation of data from multiple source systems**.**
- **Change detection:** Staging areas can be set up to manage extraction of new and modified data as needed.
- **Scheduling:** Individual tasks within an ETL workflow can be scheduled to run in a specific sequence, concurrently, and at certain times.
- **Data cleansing and validation:** For example, you can handle missing values and duplicated records.
- **Aggregating data:** You can use the staging area to summarize data. For example, daily sales data can be aggregated into weekly, monthly, or annual averages, prior to loading into a reporting system.
- **Normalizing data:** To enforce consistency of data types, or names of categories such as country and state codes in place of mixed naming conventions such as “Mont,” “MA,” or “Montana.”

Why use a staging area?

- Separate location where data from source systems is extracted to
- Decouples operations such as validation, cleansing and other processes
- Minimizes corruption risk

- Simplifies ETL workflows
- Simplifies recovery

Summary: -

- A staging area acts as a bridge between data sources and the target system and are mainly used to integrate disparate data sources in data warehouses.
- Staging areas can be implemented quite simply as a set of flat files in a directory and managed with scripts, or as tables in a database.
- Staging areas decouple data processing from the source systems and thus help minimize risk of data corruption.
- Although they are often transient, staging areas can be held for archiving or troubleshooting purposes.

6] Verify Data quality

What is data quality verification?

Data verification includes checking your data for:

- **Accuracy:** Is your data correct?
- **Completeness:** Is there missing data?
- **Consistency:** Are fields consistently entered?
- **Currency:** Is your data up to date?

Data Verification is about:

- Managing data quality
- Enhancing data reliability

High-quality data enables:

- Successful integration of data
- Integrity of complex relationships

Data verification provides you with:

- A complete, connected view of your organization
- Data ready for advanced analysis
- Statistical modeling and machine learning
- Boosts confidence in your insights and decision-making

<u>Data quality concerns</u> –

Accuracy –

- Source and destination records match
- Duplicated records
- Typos
  - Out-of-range values
  - Spelling errors
- Mass misalignment
  - Misinterpretation of a comma as a separator

Completeness –

- Locating missing values
  - Void or null fields
  - Placeholders like “999” or “-1”
- Locating missing data records due to system failures

Consistency –

- Non-conformance to standard terms
- Date formatting
- YMD and MDY
- Inconsistent data entry
- “Mr. John Doe” and “John Doe”
- Inconsistent units
  - Metric and imperial
  - Dollars and thousands of dollars

Currency –

- Avoiding outdated information
  - Update addresses: Check against a ‘change of address’ database
  - Manage name changes

Managing data quality –

Resolving bad data is an iterative process

First, you’ll implement rules to detect bad data. Then you’ll apply those rules to capture and quarantine any bad data. You might need to report any bad data and share the findings with the appropriate domain experts. You and your team can investigate the root cause of each problem, searching for clues upstream in the data lineage. Once you diagnose each problem, you can begin correcting the issues.

Example:

Scenario: Data from specific data sources consistently has the following data quality issues:

- Missing data (NULL values)
- Out-of-range values
- Duplicate values
- Invalid values

An action plan:

Here's how an organization might manage and resolve these issues:

1. Write SQL queries to detect and test for these conditions
2. Create rules for correcting and managing those conditions
3. Create a script that runs the data validation SQL queries every night
4. Automate the script created in step 3 that performs detection and correction processes
5. Review the automated report and address issues that could not be resolved automatically

Data quality solutions –

- IBM InfoSphere Server, & IBM Data Refinery
- Informatica Data Quality
- SAP Data Quality Management
- SAS Data Quality
- Talend Open Studio for Data Quality
- Precisely Spectrum Quality

- Microsoft Data Quality Services
- Oracle Enterprise Data Quality
- OpenRefine

IBM InfoSphere –

Enables you to:

- Continuously monitor data quality
- Clean data on an ongoing basis
- Turn your data into trusted information

End-to-end data quality tools to:

- Understand your data and its relationships
- Monitor and analyze data quality continuously
- Clean, standardize, and match data
- Maintain data lineage

Summary: -

- Data verification includes checking your data for accuracy, completeness, consistency, and currency.
- Data verification is about managing data quality, enhancing data reliability, and maximizing data value.
- Determining how to resolve and prevent bad data can be a complex and iterative process.
- Enterprise-grade tools such as “IBM InfoSphere Information Server for Data Quality” can help you perform data verification in a unified environment.

7] Populating a Data Warehouse

Loading frequency –

Populating the warehouse is an ongoing process:

- Initial load + periodic incremental loads
- Full refreshes due to schema changes or catastrophic failure are rare
- Fact tables need more frequent updating than dimension tables
- City and store lists change slowly, unlike sales transactions

Typical ways of loading data –

- Many tools can help you automate the ongoing loading process
- Db2 has a Load utility that is faster than inserting a row at a time
- You can automate loading as part of your ETL pipeline using tools like Apache Airflow and Kafka
- You can use tools like Bash, Python, and SQL, to build your data pipeline and schedule it with cron
- InfoSphere DataStage allows you to compile and run jobs to load your data

Populating your data warehouse –

Preparation:

- Your schema has been modeled
- Your data has been staged
- You have verified the data quality

Setup and initial load:

- Instantiate the data warehouse and its schema
- Create production tables
- Establish relations between tables
- Load your transformed and cleaned data

Ongoing loads:

- You can automate incremental loads using a script as part of your ETL pipeline
- For example, append data daily or weekly
- Incremental loading requires you to have a method to correctly detect new and changed data

Change detection –

- Normally, you detect changes in the source system
- Many RDBMSs have change tracking to identify new, changed, or deleted records
- You might have timestamps identifying when data was first written and when it was modified
- Otherwise, you may need to use brute-force comparison to your staged data

Periodic maintenance –

- Perform monthly or yearly data archiving
- Script both the deletion of older data and its archiving to slower, less costly storage

Example: auto sales data

Manually populating 'sales' star schema DW:

Here's a sample of some auto sales transaction data from a fictional company called Shiny Auto Sales. You can see several foreign key columns, such as "sales ID," which is a sequential key identifying the sales invoice number, "emp no," which is the employee number, and "class ID," which encodes the type of car sold, such as "small SUV." Each of these keys represents a dimension that points to a corresponding dimension table in the star schema. The "date" column is a dimension that indicates the sale date. The "amount" column is the sales amount, which happens to be the fact of interest. his table is already close to the form of a fact table. The only exception is the date column, which is not yet represented by a foreign "date ID" key.

	sales_id	amount	empl_no	class_id	date
0	1629	42000.00	642	30	2021-01-04
1	1630	17680.00	617	70	2021-01-04
2	1631	37100.00	642	70	2021-01-05
3	1632	26500.00	680	60	2021-01-05
4	1633	8200.00	707	71	2021-01-05
5	1634	42099.00	642	40	2021-01-05
6	1635	12099.00	720	61	2021-01-06
7	1636	51999.00	607	40	2021-01-06

Create a dimension table –

Create a salesperson dimension table:

```
CREATE TABLE sales.DimSalesPerson (
      SalesPersonID SERIAL primary key,
      SalesPersonAltID varchar(10) not null,
      SalesPersonName varchar(50)
);
```

Populate the dimension table –

```
INSERT INTO sales.DimSalesPerson(SalesPersonAltID,
                                 SalesPersonName)
values
( 617, 'Gocart Joe'),
( 642, 'Jake Salesbouroughs'),
( 680, 'Cadillac Jack'),
( 707, 'Jane Honda'),
( 720, 'Kayla Kaycar'),
( 607, 'William Jeepman'),
( 609, 'Happy Dollarmaker'),
( 711, 'Sally Caraway');
```

View the salesperson table –

```
SELECT * FROM sales.DimSalesPerson LIMIT 5;

salespersonid   |  salespersonaltid |      salespersonname
----------------+-------------------+----------------------
            1   |               617 |           Gocart Joe
            2   |               642 |   Jake Salesboroughs
            3   |               680 |        Cadillac Jack
            4   |               707 |           Jane Honda
            5   |               720 |         Kayla Kaycar
```

Create the sales fact table –

Create an auto sales fact table:

```
CREATE TABLE sales.FactAutoSales(
      TransactionID bigserial primary key,
      SalesID int not null,
      SalesDateKey int,
      AutoClassID int not null,
      SalesPersonID int not null,
      Amount money
);
```

Set up table relations –

Create relations between fact and auto class tables:

```
ALTER TABLE sales.FactAutoSales
ADD CONSTRAINT
KV_AutoClassID FOREIGN KEY (AutoClassID)
REFERENCES
sales.DimAutoCategory(AutoClassID);
```

Populate the sales fact table –

Populate the Auto Sales fact table:

```
INSERT INTO sales.FactAutoSales(
      SalesID,
      Amount,
      SalesPersonID,
      AutoClassID,
      SalesDateKey
) values

(1629, 42000.00, 2,  1, 4),
(1630, 17680.00, 1,  2, 4),
(1631, 37100.00, 2,  2, 5),
```

View the sales fact table –

```
SELECT * sales.FactAutoSales LIMIT 5;
```

transactionid	salesid	salesdatekey	autoclassid	salespersonid	amount
52	1629	4	1	2	$42,000.00
53	1630	4	2	1	$17,680.00
54	1631	5	2	2	$37,100.00
55	1632	5	3	3	$26,500.00
56	1633	5	4	4	$8,200.00

Summary: -

- Populating an EDW includes creation of production tables and their relations and loading of clean data into tables
- Loading the EDW is an ongoing process beginning with an initial load, followed by periodic incremental loads
- Fact tables require frequent updating while dimension tables don't change often
- Incremental loading and periodic maintenance can be scripted and scheduled as part of your ETL pipeline

8] Querying the Data

CUBE, ROLLUP, materialized views –

- CUBE and ROLLUP provide summary reports
- Easier to implement than alternative SQL queries
- Materialized views are stored tables
- Useful for reducing load of complex, frequently requested views on large data sets
- Querying materialized views can be much faster than querying the underlying tables
- Combining cubes or rollups with materialized views can enhance performance

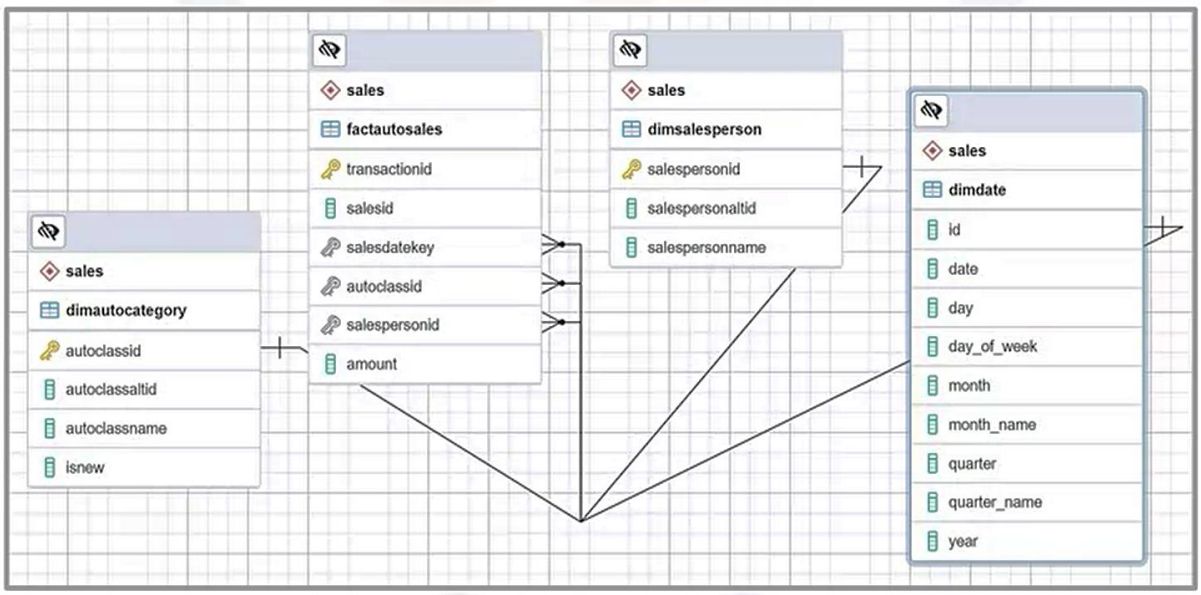
Scenario –

Consider the following scenario:

- Create live summary tables for reporting sales by salesperson and automobile type for ShinyAutoSales
- Start by understanding the sales star schema "sasDW" in their data warehouse
- Then explore relevant ShinyAutoSales data by querying tables in the "sales" star schema

- After exploring the schema, you decide to create a materialized view as a staging table
- You'll get the data you need and minimize your impact on the database
- You can refresh the data incrementally during off-peak hours

ShinyAutoSales – sales ERD



View the sales fact table –

```
sasDW=# SELECT * FROM sales.factautosales LIMIT 10;
 transactionid | salesid | salesdatekey | autoclassid | salespersonid |   amount
---------------+---------+--------------+-------------+---------------+-----------
            52 |    1629 |            4 |           1 |             2 | $42,000.00
            53 |    1630 |            4 |           2 |             1 | $17,680.00
            54 |    1631 |            5 |           2 |             2 | $37,100.00
            55 |    1632 |            5 |           3 |             3 | $26,500.00
            56 |    1633 |            5 |           4 |             4 |  $8,200.00
            57 |    1634 |            5 |           5 |             2 | $42,099.00
            58 |    1635 |            6 |           6 |             5 | $12,099.00
            59 |    1636 |            6 |           5 |             6 | $51,999.00
            60 |    1637 |            7 |           2 |             1 | $42,099.00
            61 |    1638 |            7 |           3 |             1 | $32,099.00
(10 rows)
```

View the auto category table –

```
sasDW=# SELECT * FROM sales.dimautocategory LIMIT 10;
 autoclassid | autoclassaltid | autoclassname | isnew
-------------+----------------+---------------+-------
           1 | 30             | 4 Door Sedan  | t
           2 | 70             | Truck         | t
           3 | 60             | Midsize SUV   | t
           4 | 71             | Truck         | f
           5 | 40             | Compact SUV   | t
           6 | 61             | Midsize SUV   | f
           7 | 41             | Compact SUV   | f
           8 | 31             | 4 Door Sedan  | f
(8 rows)
```

View the salesperson table –

```
sasDW=# SELECT * FROM sales.dimsalesperson LIMIT 10;
 salespersonid | salespersonaltid |   salespersonname
---------------+------------------+---------------------
             1 | 617              | Gocart Joe
             2 | 642              | Jake Salesbouroughs
             3 | 680              | Cadillac Jack
             4 | 707              | Jane Honda
             5 | 720              | Kayla Kaycar
             6 | 607              | William Jeepman
             7 | 609              | Happy Dollarmaker
             8 | 711              | Sally Caraway
(8 rows)
```

View the date table –

```
sasDW=# SELECT * FROM sales.dimdate LIMIT 8;
 id |    date    | day | day_of_week | month | month_name | quarter | quarter_name | year
----+------------+-----+-------------+-------+------------+---------+--------------+------
  1 | 2021-01-01 |   1 | Fri         |     1 | Jan        |       1 | Q1           | 2021
  2 | 2021-01-02 |   2 | Sat         |     1 | Jan        |       1 | Q1           | 2021
  3 | 2021-01-03 |   3 | Sun         |     1 | Jan        |       1 | Q1           | 2021
  4 | 2021-01-04 |   4 | Mon         |     1 | Jan        |       1 | Q1           | 2021
  5 | 2021-01-05 |   5 | Tue         |     1 | Jan        |       1 | Q1           | 2021
  6 | 2021-01-06 |   6 | Wed         |     1 | Jan        |       1 | Q1           | 2021
  7 | 2021-01-07 |   7 | Thu         |     1 | Jan        |       1 | Q1           | 2021
  8 | 2021-01-08 |   8 | Fri         |     1 | Jan        |       1 | Q1           | 2021
(8 rows)
```

Denormalized view –

Selecting human-interpretable columns

```
sasDW-# SELECT
sasDW-#     D.date,
sasDW-#     C.autoclassname,
sasDW-#     C.isnew,
sasDW-#     SP.salespersonname,
sasDW-#     F.amount
sasDW-#
sasDW-# FROM sales.factautosales F
sasDW-#
sasDW-# INNER JOIN sales.dimdate D          ON (F.salesdatekey = D.id)
sasDW-# INNER JOIN sales.dimautocategory C  ON (F.autoclassid = C.autoclassid)
sasDW-# INNER JOIN sales.dimsalesperson SP  ON (F.salespersonid = SP.salespersonid)
sasDW-# ;
SELECT 17
```

Denormalized, materialized view –

```
sasDW=# CREATE MATERIALIZED VIEW  DNsales AS
sasDW-#
sasDW-# SELECT
sasDW-#     D.date,
sasDW-#     C.autoclassname,
sasDW-#     C.isnew,
sasDW-#     SP.salespersonname,
sasDW-#     F.amount
sasDW-#
sasDW-# FROM sales.factautosales F
sasDW-#
sasDW-# INNER JOIN sales.dimdate D          ON (F.salesdatekey = D.id)
sasDW-# INNER JOIN sales.dimautocategory C  ON (F.autoclassid = C.autoclassid)
sasDW-# INNER JOIN sales.dimsalesperson SP  ON (F.salespersonid = SP.salespersonid)
sasDW-# ;
SELECT 17
```

DNSales materialized view –

```
sasDW=# SELECT * FROM DNsales LIMIT 10;
    date    | autoclassname | isnew |   salespersonname    |   amount
------------+---------------+-------+----------------------+------------
 2021-01-04 | 4 Door Sedan  | t     | Jake Salesbouroughs  | $42,000.00
 2021-01-04 | Truck         | t     | Gocart Joe           | $17,680.00
 2021-01-05 | Truck         | t     | Jake Salesbouroughs  | $37,100.00
 2021-01-05 | Midsize SUV   | t     | Cadillac Jack        | $26,500.00
 2021-01-05 | Truck         | f     | Jane Honda           |  $8,200.00
 2021-01-05 | Compact SUV   | t     | Jake Salesbouroughs  | $42,099.00
 2021-01-06 | Midsize SUV   | f     | Kayla Kaycar         | $12,099.00
 2021-01-06 | Compact SUV   | t     | William Jeepman      | $51,999.00
 2021-01-07 | Truck         | t     | Gocart Joe           | $42,099.00
 2021-01-07 | Midsize SUV   | t     | Gocart Joe           | $32,099.00
(10 rows)
```

Applying CUBE to your materialized view –

```
sasDW=# SELECT
sasDW-#     autoclassname,
sasDW-#     salespersonname,
sasDW-#     SUM(amount)
sasDW-# FROM
sasDW-#     DNsales
sasDW-# WHERE
sasDW-#     isNew=True
sasDW-# GROUP BY
sasDW-#     CUBE (
sasDW(#         autoclassname,
sasDW(#         salespersonname
sasDW(#     );
```

autoclassname	salespersonname	sum
		$366,076.00
4 Door Sedan	Jake Salesbouroughs	$42,000.00
Truck	Gocart Joe	$59,779.00
Truck	Jake Salesbouroughs	$37,100.00
Compact SUV	Jake Salesbouroughs	$42,099.00
Midsize SUV	Gocart Joe	$32,099.00
Compact SUV	William Jeepman	$51,999.00
Compact SUV	Happy Dollarmaker	$74,500.00
Midsize SUV	Cadillac Jack	$26,500.00
Compact SUV		$168,598.00
Truck		$96,879.00
Midsize SUV		$58,599.00
4 Door Sedan		$42,000.00
	Happy Dollarmaker	$74,500.00
	Jake Salesbouroughs	$121,199.00
	Cadillac Jack	$26,500.00
	William Jeepman	$51,999.00
	Gocart Joe	$91,878.00

(18 rows)

Applying ROLLUP to your materialized view –

```
sasDW=# SELECT
sasDW-#     autoclassname,
sasDW-#     salespersonname,
sasDW-#     SUM(amount)
sasDW-# FROM
sasDW-#     DNsales
sasDW-# WHERE
sasDW-#     isNew=True
sasDW-# GROUP BY
sasDW-#     ROLLUP (
sasDW(#         autoclassname,
sasDW(#         salespersonname
sasDW(#     );
```

autoclassname	salespersonname	sum
		$366,076.00
4 Door Sedan	Jake Salesbouroughs	$42,000.00
Truck	Gocart Joe	$59,779.00
Truck	Jake Salesbouroughs	$37,100.00
Compact SUV	Jake Salesbouroughs	$42,099.00
Midsize SUV	Gocart Joe	$32,099.00
Compact SUV	William Jeepman	$51,999.00
Compact SUV	Happy Dollarmaker	$74,500.00
Midsize SUV	Cadillac Jack	$26,500.00
Compact SUV		$168,598.00
Truck		$96,879.00
Midsize SUV		$58,599.00
4 Door Sedan		$42,000.00

(13 rows)

Summary: -

- CUBE and ROLLUP summaries on materialized views provide powerful capabilities for quickly querying and analyzing data in data warehouses.
- CUBE and ROLLUP operations generate the kinds of summaries grouped by dimensions that management often requests.
- You can denormalize star schemas using joins to bring together human-interpretable facts and dimensions in a single materialized view.
- You can create staging tables from materialized views, which you can incrementally refresh during off-peak hours.

Week 3

Data Warehouse Analytics

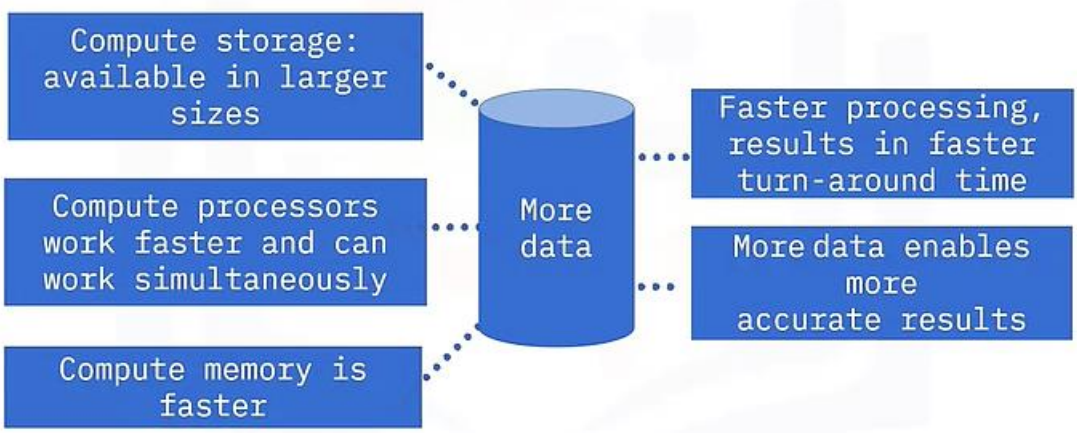
Module 1: - Data Warehouse Analytics

1] Introduction to Analytics and Business Intelligence (BI) Tools

Analytics defined –

Analytics is the methodical compilation and dissection of data, statistics and operations research to build models that enable better decisions

Tools-revolutionizing analytics –



Analytics in action and results –

Built-in expertise provides insightful outcomes in almost real-time

- Descriptive outcomes: What happened in the past
- Prescriptive outcomes: What can happen in the future
- Predictive outcomes: What should happen in the future

Business Intelligence (BI) tools –

- Enable data preparation, mining, management, and visualization
- Focus on the “what” and the “why”
- Apply statistics and operations research
- Harness the power of descriptive, diagnostic, predictive and prescriptive analytics

- Dashboarding, reporting and self-serve analytics
- Transform data into opportunity

Tools for Business Intelligence –

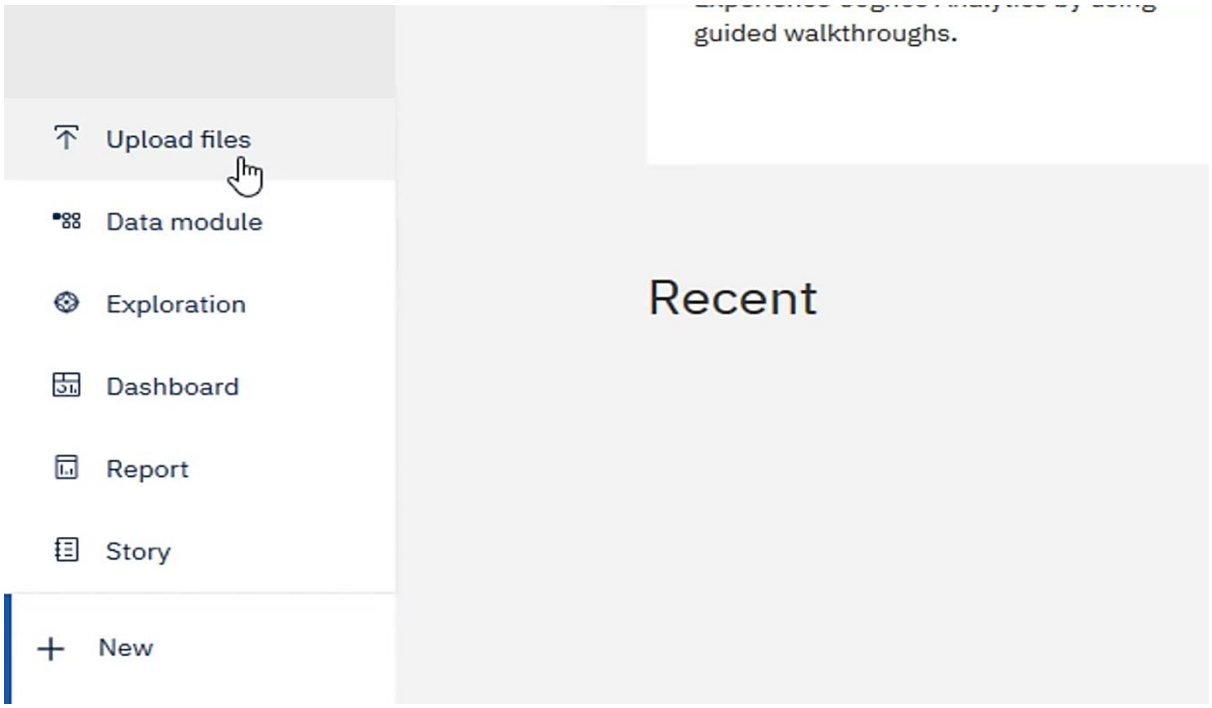
- IBM Cognos Analytics – integrates Watson Analytics leverages IBM AI and natural language processing
- Microsoft Power BI – known for its security and suite of applications
- Tableau – provides powerful visualizations
- Oracle Analytics Cloud – known for conversational analytics
- SAP Business Objects – recognized for smarter analytics
- Tibaco Spotfire – provides AI-infused visual analytics and custom analytics app creation, offers scalability for small and large organizations.

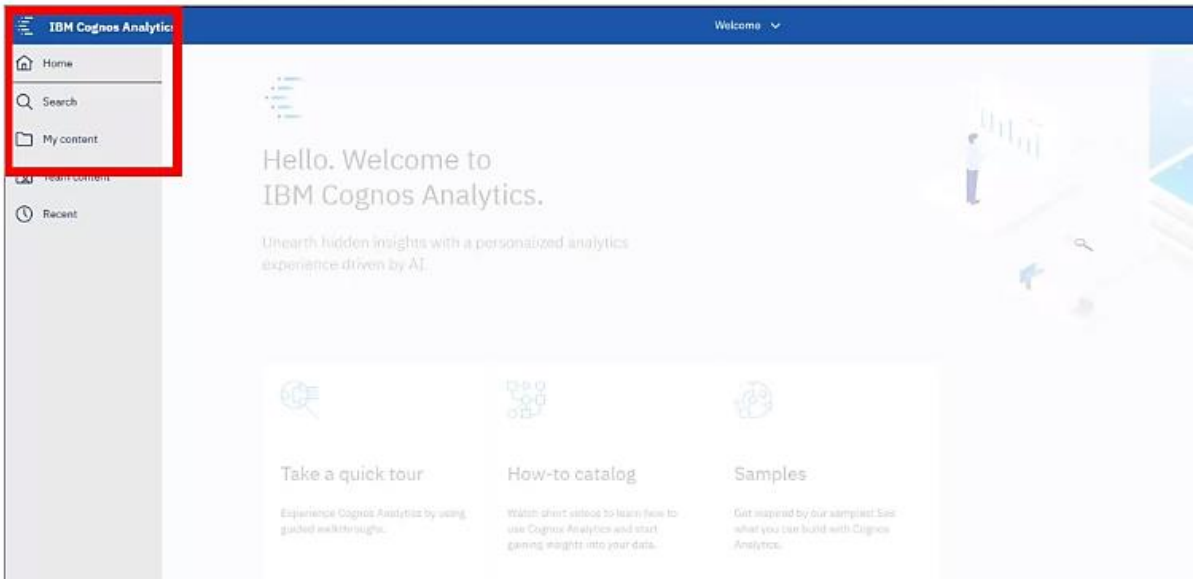
IBM Cognos Analytics –

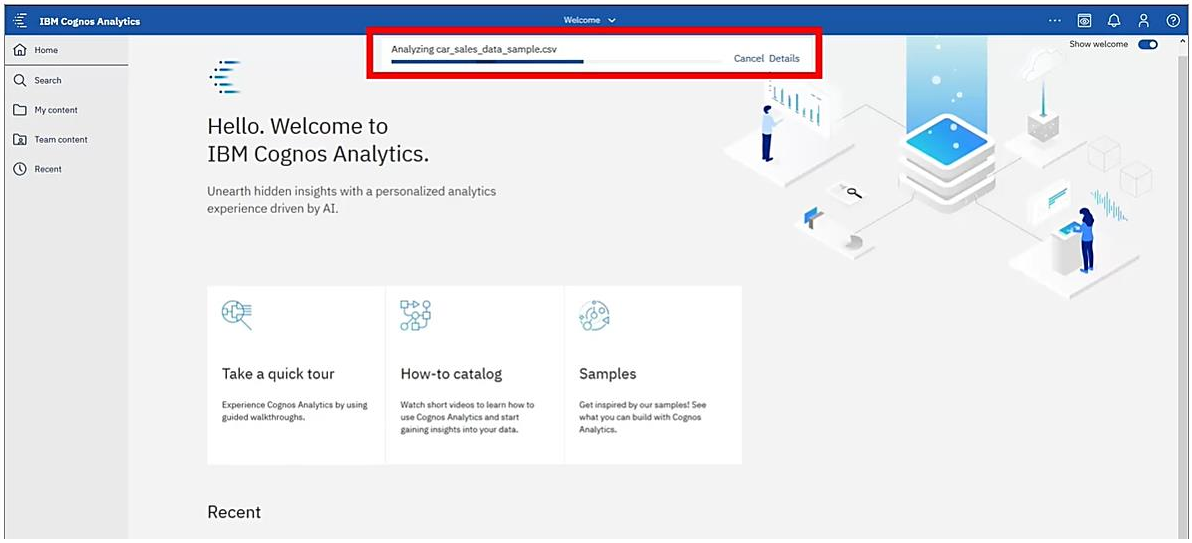
Summary: -

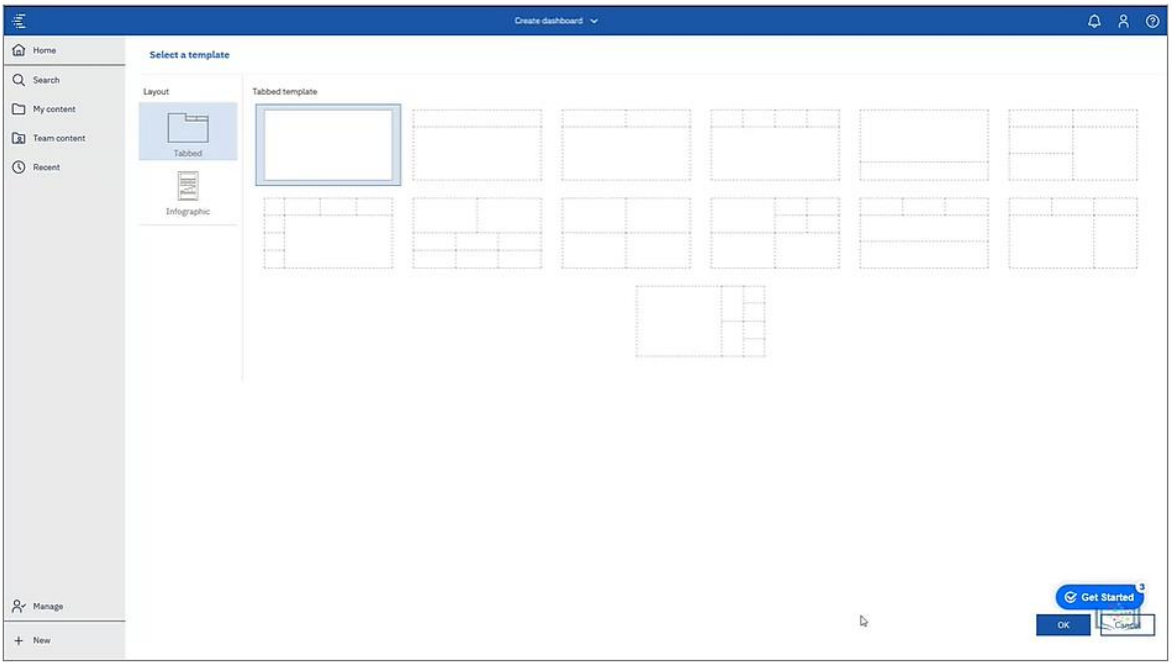
- Analytics is the building models with data to make better decisions
- Business Intelligence is a technology that enables data preparation, data mining, data management, and data visualization
- The software market offers many Business Intelligence tools
- IBM Cognos Analytics is one of the top BI solutions available based on AI.

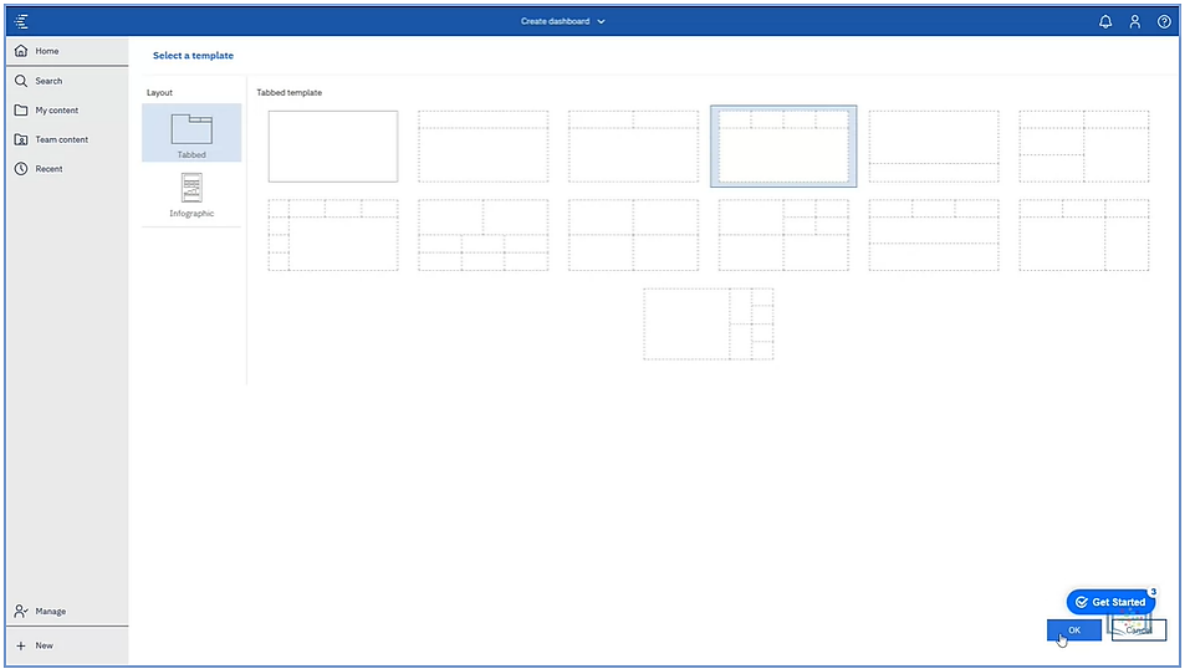
2] Navigating in Cognos Analytics

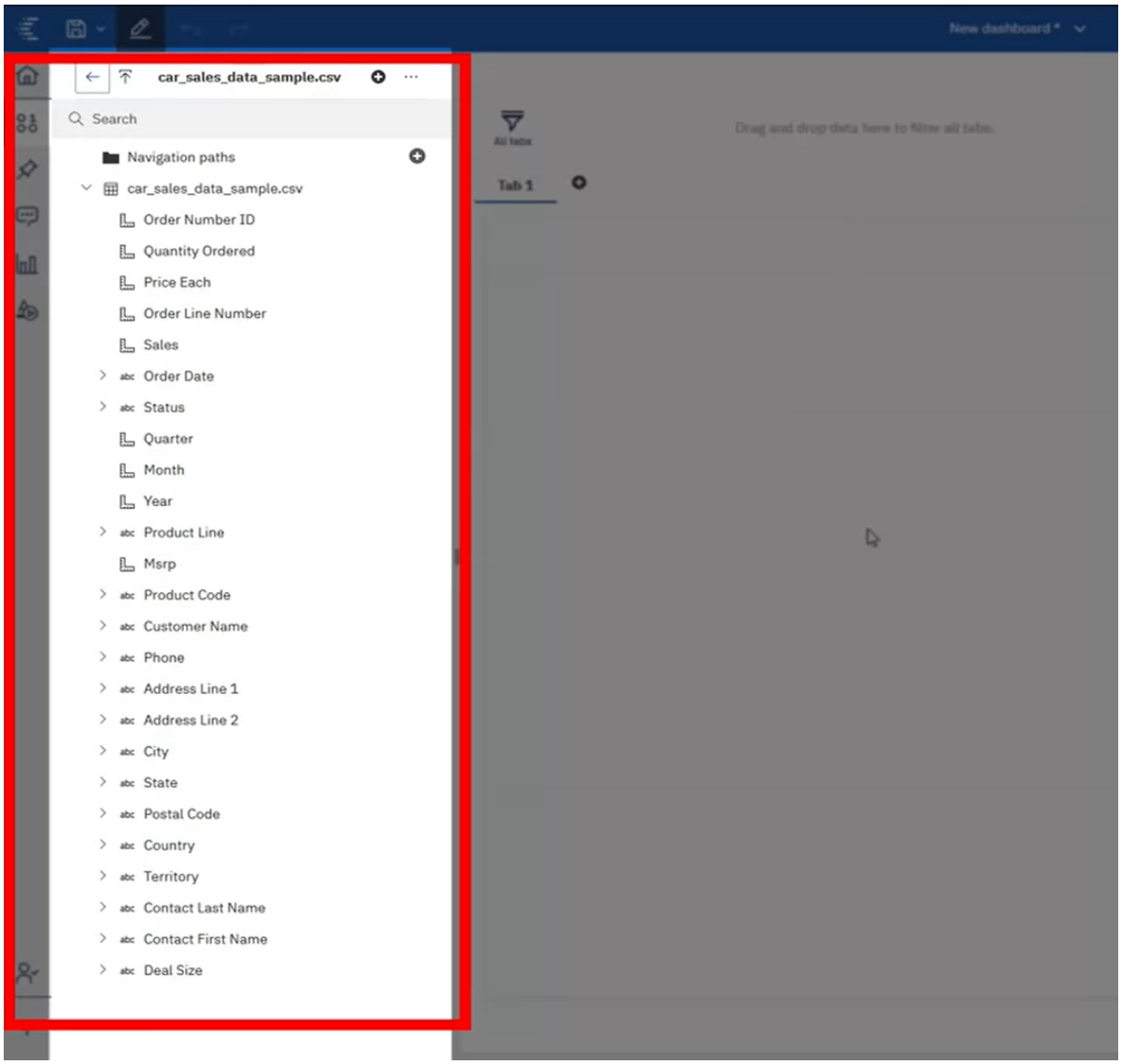


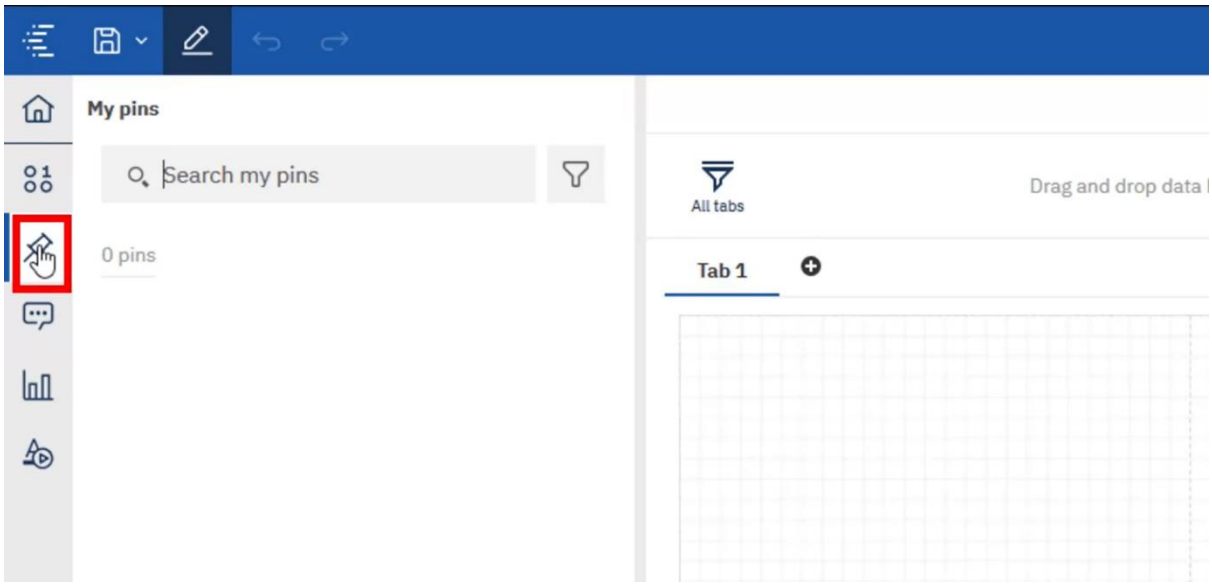


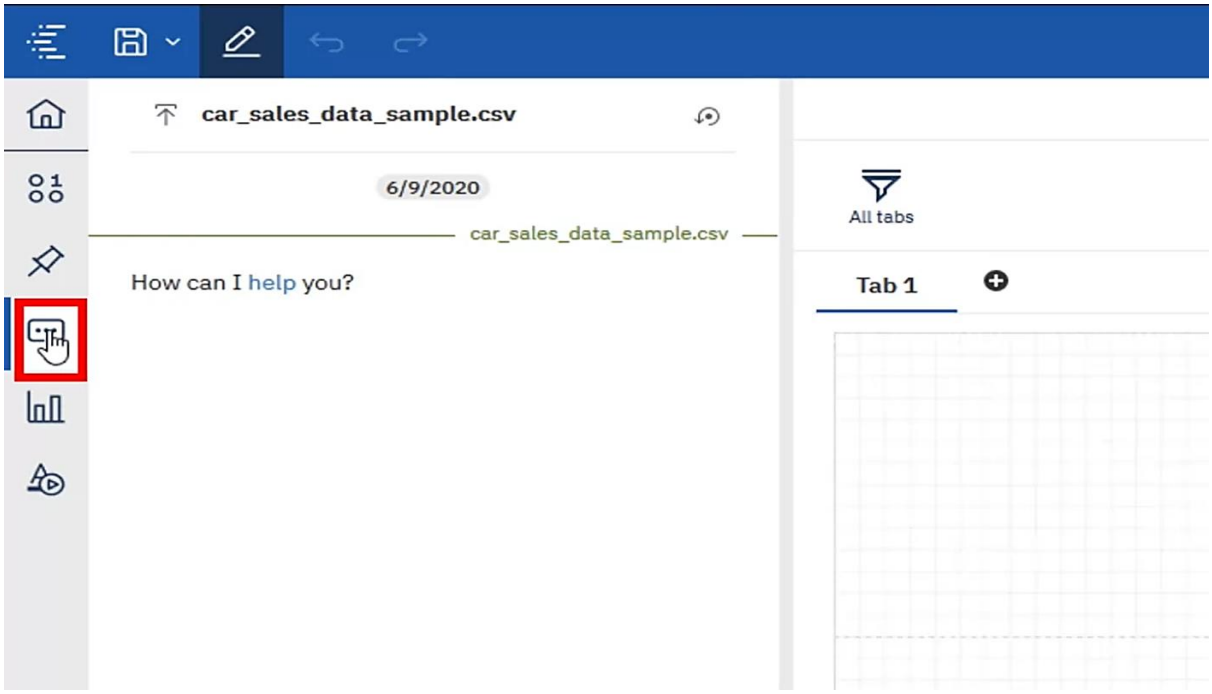

IBM Cognos Analytics
Welcome
Home
Search
My content
Team content
Recent
Analyzing car_sales_data_sample.csv
Cancel Details
Show welcome
Hello. Welcome to
IBM Cognos Analytics.
Unearth hidden insights with a personalized analytics experience driven by AI.
Take a quick tour
Experience Cognos Analytics by using guided walkthroughs.
How-to catalog
Watch short videos to learn how to use Cognos Analytics and start gaining insights into your data.
Samples
Get inspired by our samples! See what you can build with Cognos Analytics.
Recent

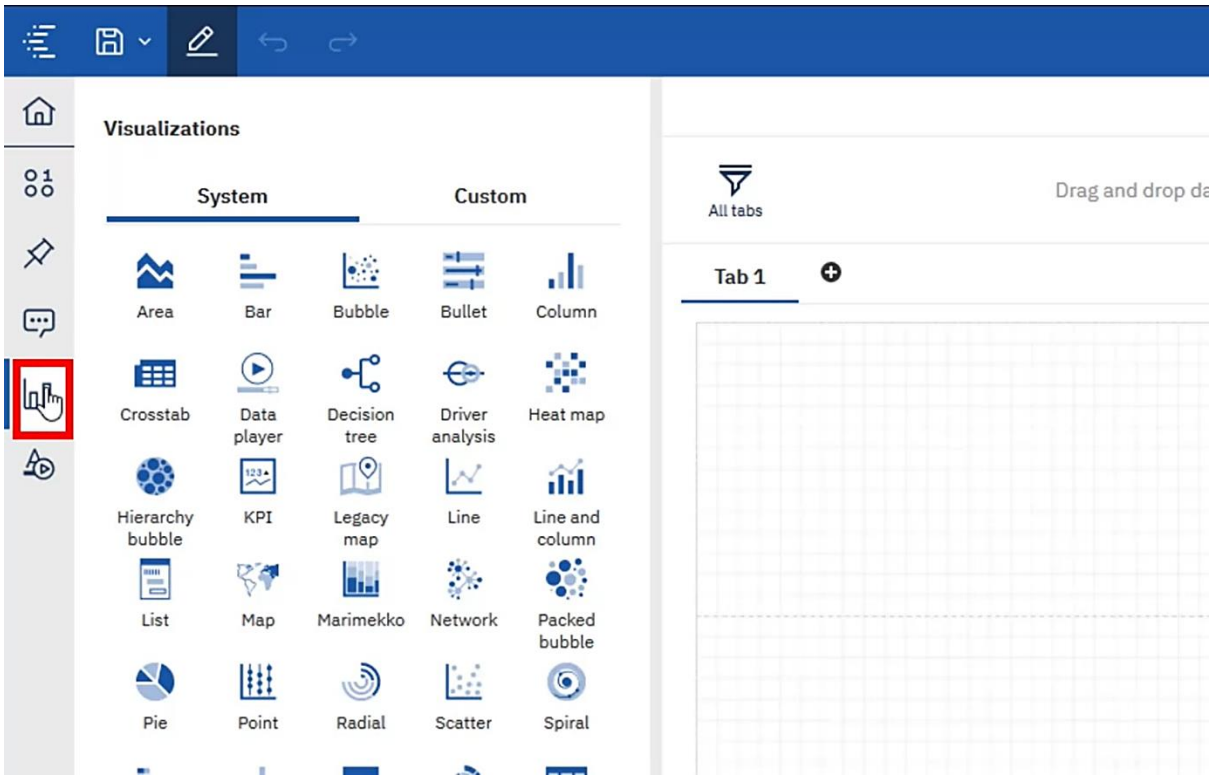

Create dashboard
Home
Search
My content
Team content
Recent
Manage
New
Select a template
Layout
Tabbed
Infographic
Tabbed template
Get Started
OK

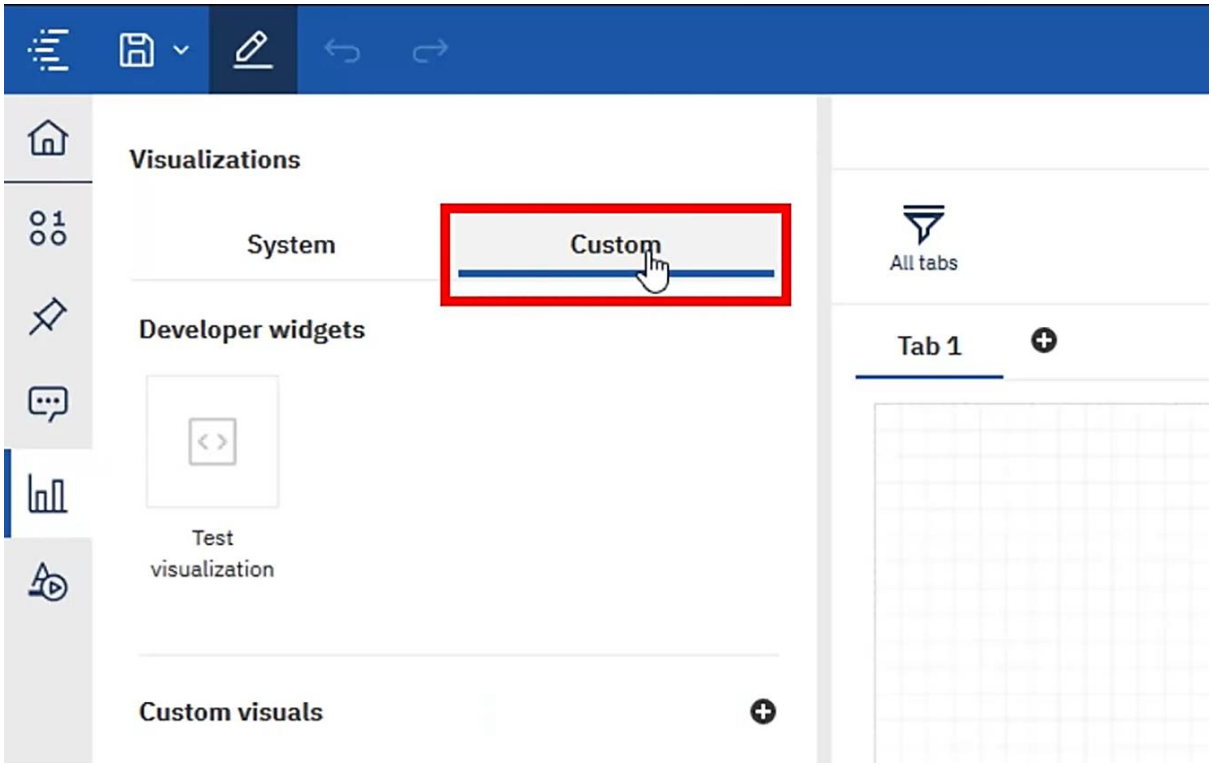

Create dashboard
Home
Search
My content
Team content
Recent
Manage
New
Select a template
Layout
Tabbed
Infographic
Tabbed template
Get Started
OK

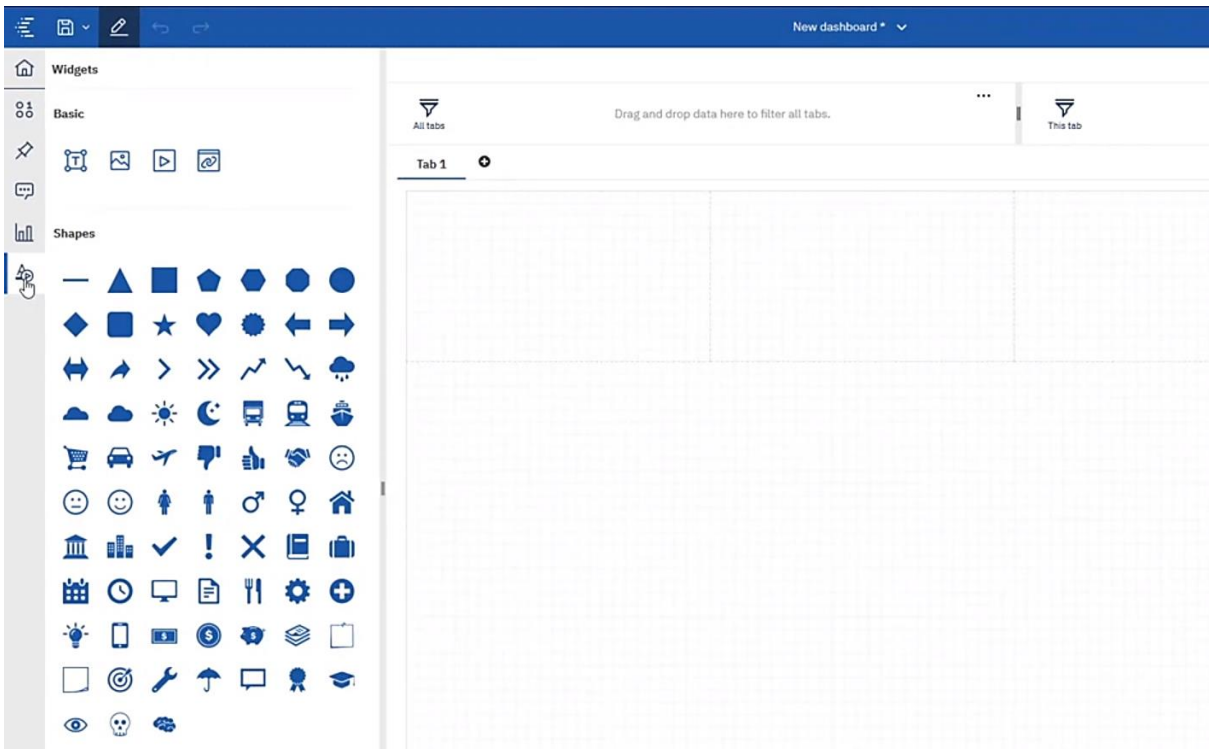

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
Contact First Name
Deal Size
All tabs
Tab 1
Drag and drop data here to filter all tabs.


My pins
Search my pins
0 pins
All tabs
Drag and drop data
Tab 1

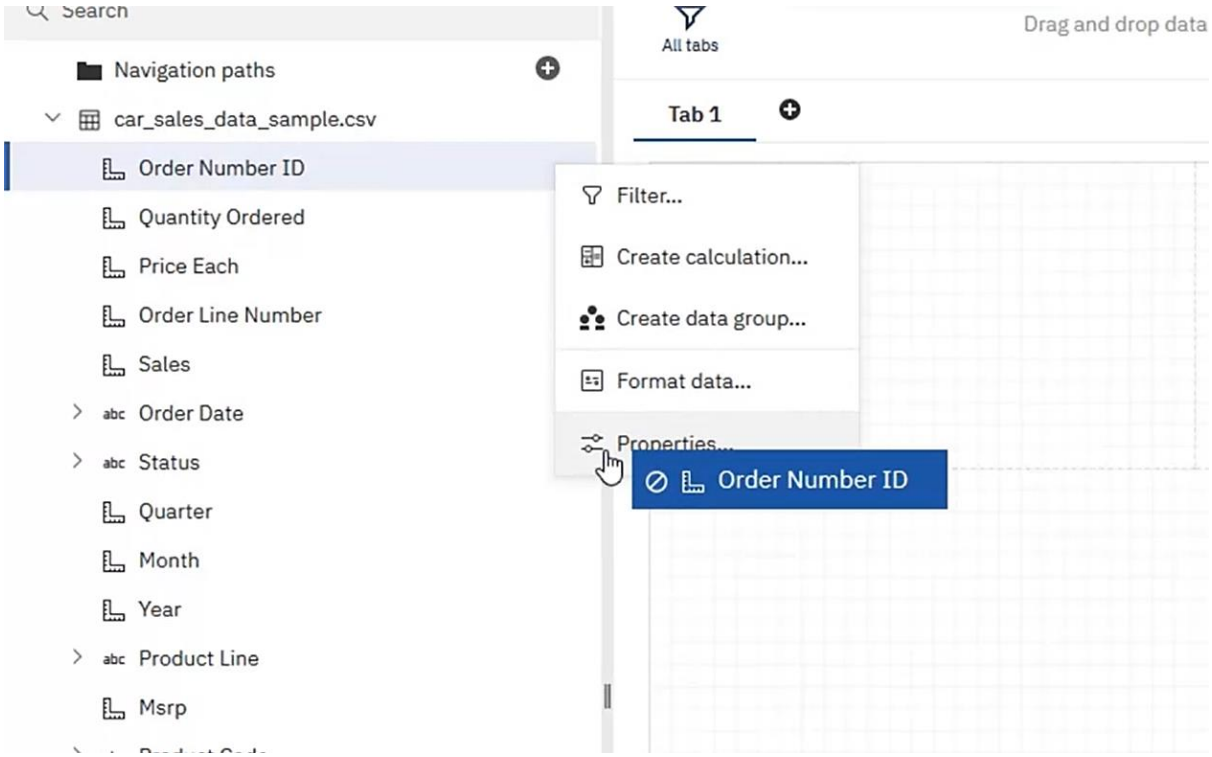

car_sales_data_sample.csv
6/9/2020
car_sales_data_sample.csv
How can I help you?
All tabs
Tab 1

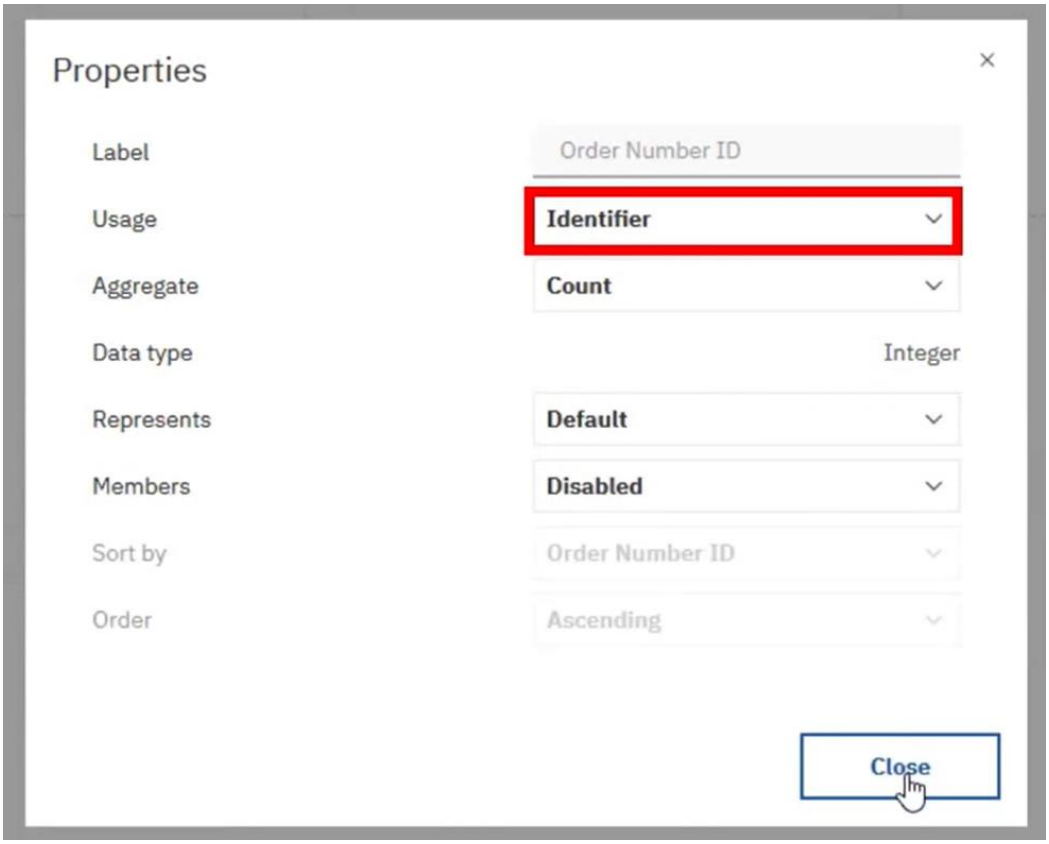

Visualizations
System
Custom
Area
Bar
Bubble
Bullet
Column
Crosstab
Data player
Decision tree
Driver analysis
Heat map
Hierarchy bubble
KPI
Legacy map
Line
Line and column
List
Map
Marimekko
Network
Packed bubble
Pie
Point
Radial
Scatter
Spiral
All tabs
Drag and drop da
Tab 1

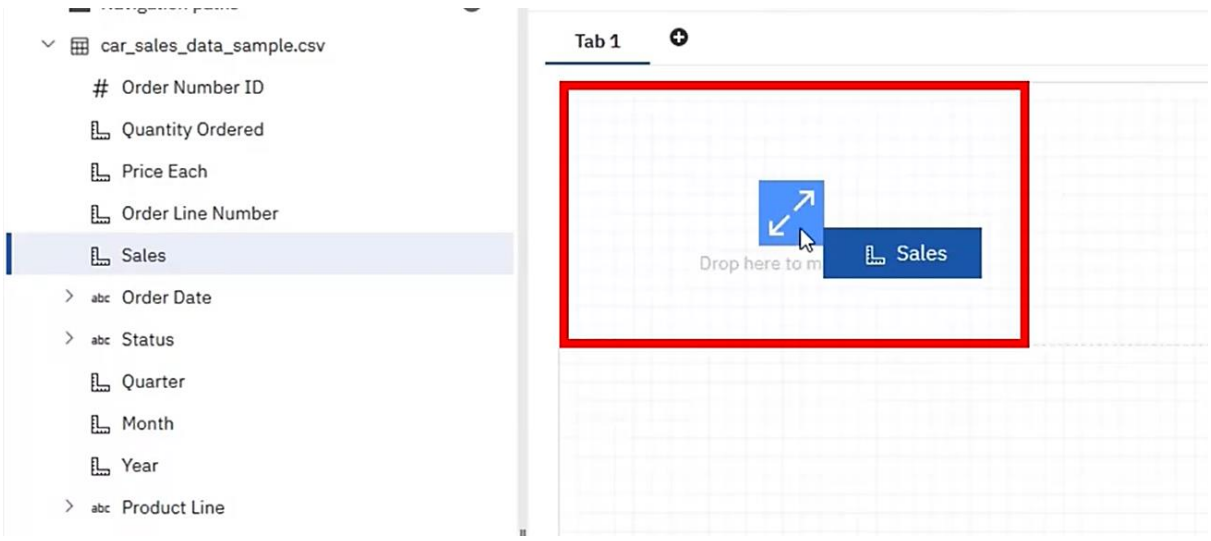

Visualizations
System
Custom
Developer widgets
Test visualization
Custom visuals
All tabs
Tab 1

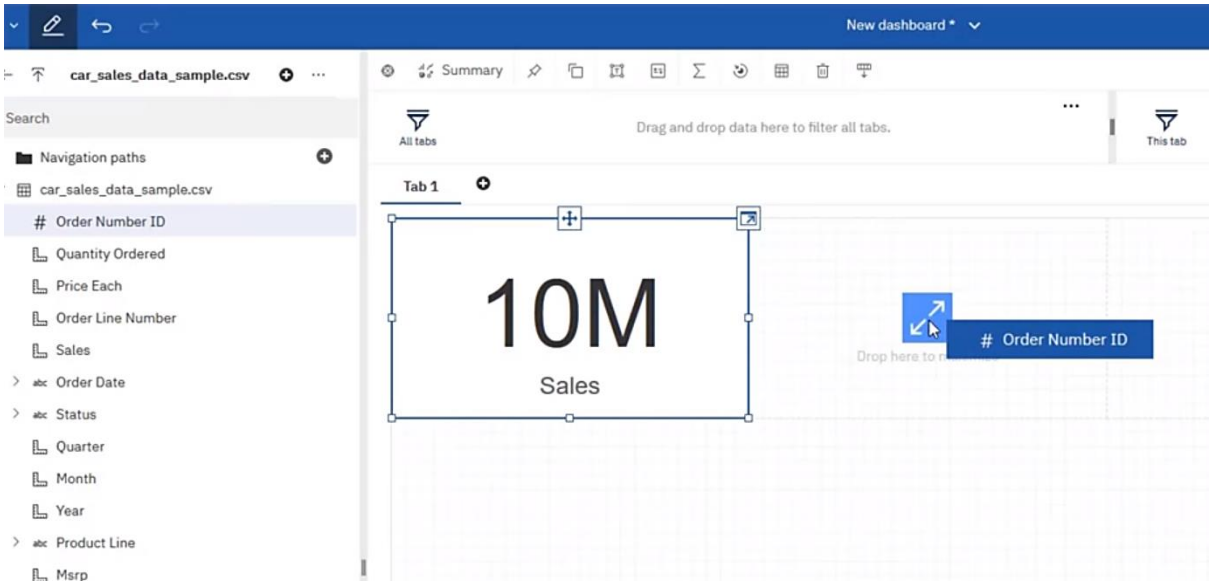


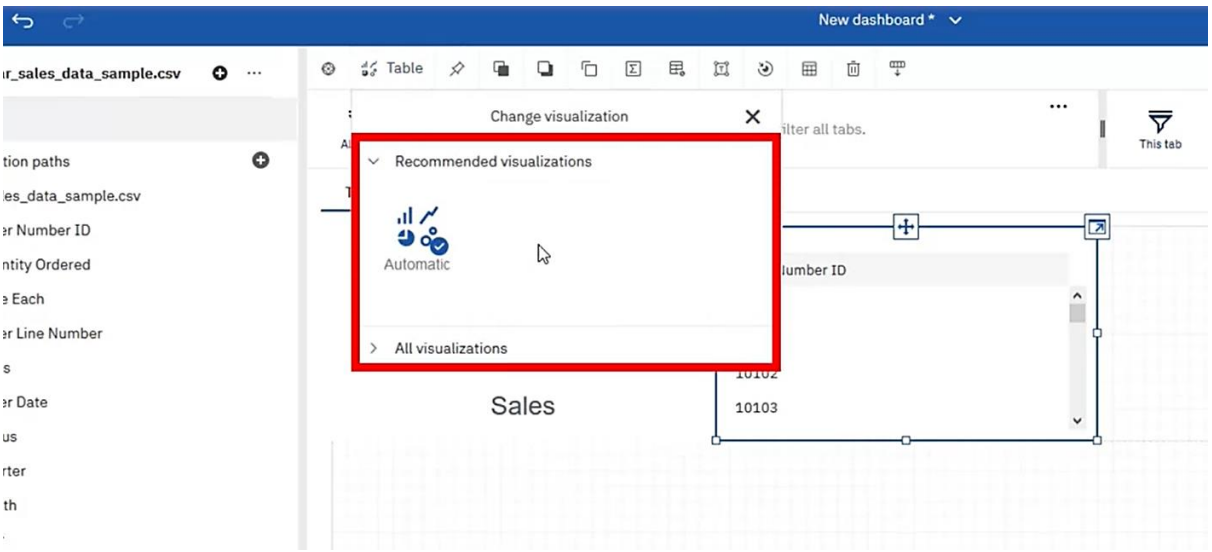
3] Creating a Simple Dashboard in Cognos

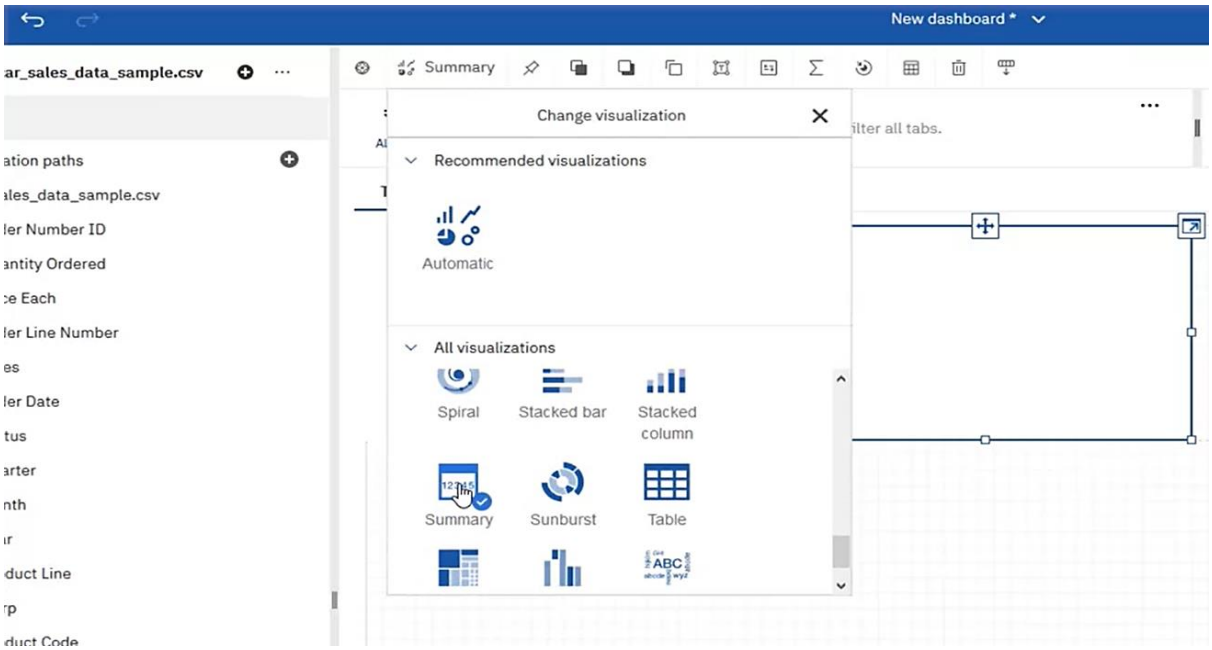


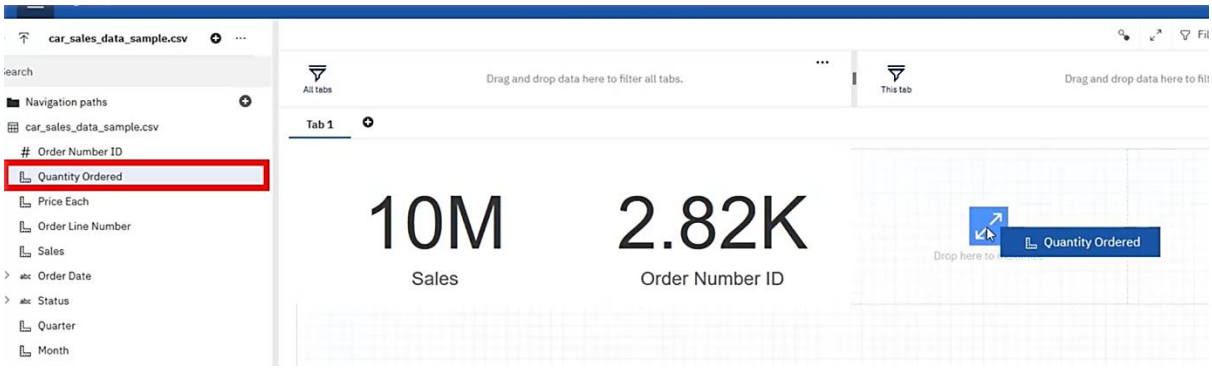

Properties
Label
Order Number ID
Usage
Identifier
Aggregate
Count
Data type
Integer
Represents
Default
Members
Disabled
Sort by
Order Number ID
Order
Ascending
Close

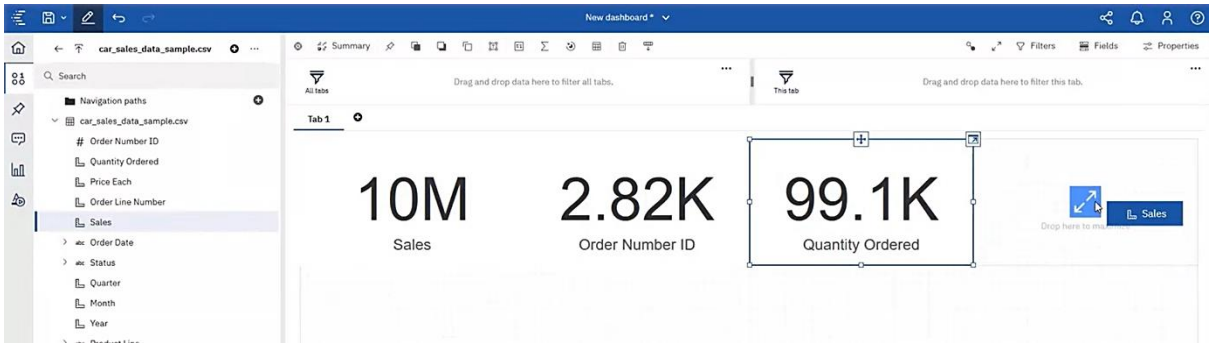

car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Tab 1
Drop here to m
Sales

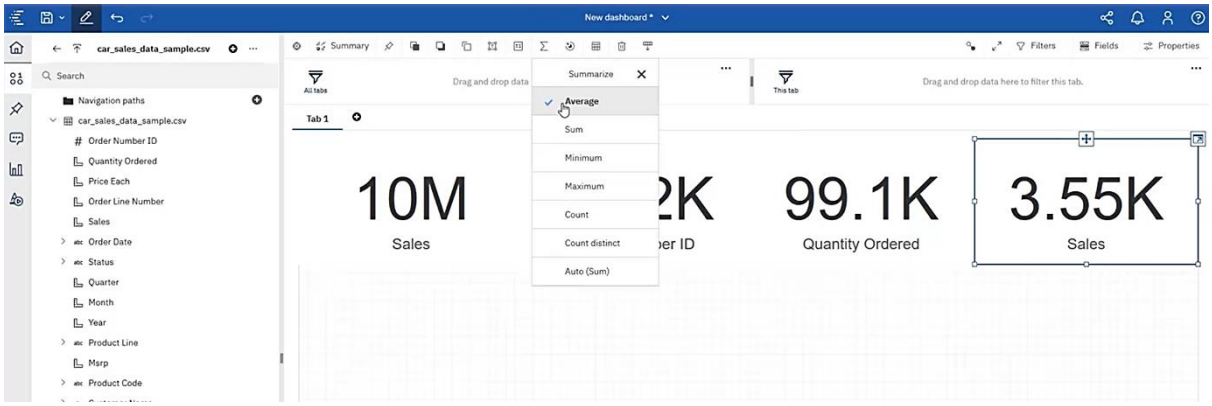

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Summary
All tabs
Drag and drop data here to filter all tabs.
This tab
Tab 1
10M
Sales
Order Number ID

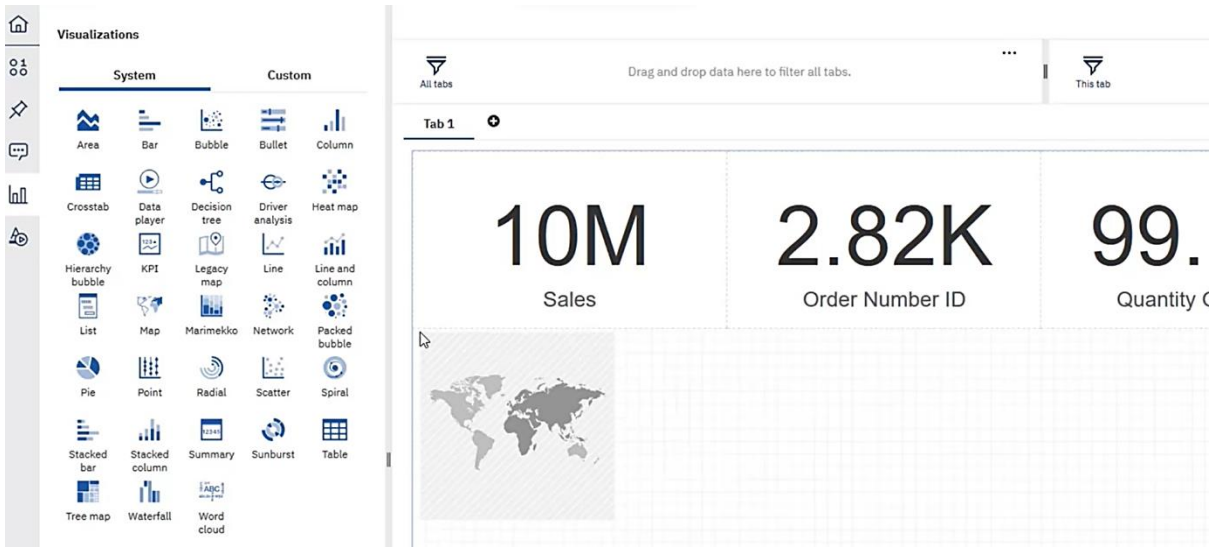

New dashboard *
Table
Change visualization
Recommended visualizations
Automatic
All visualizations
This tab
Sales
10103

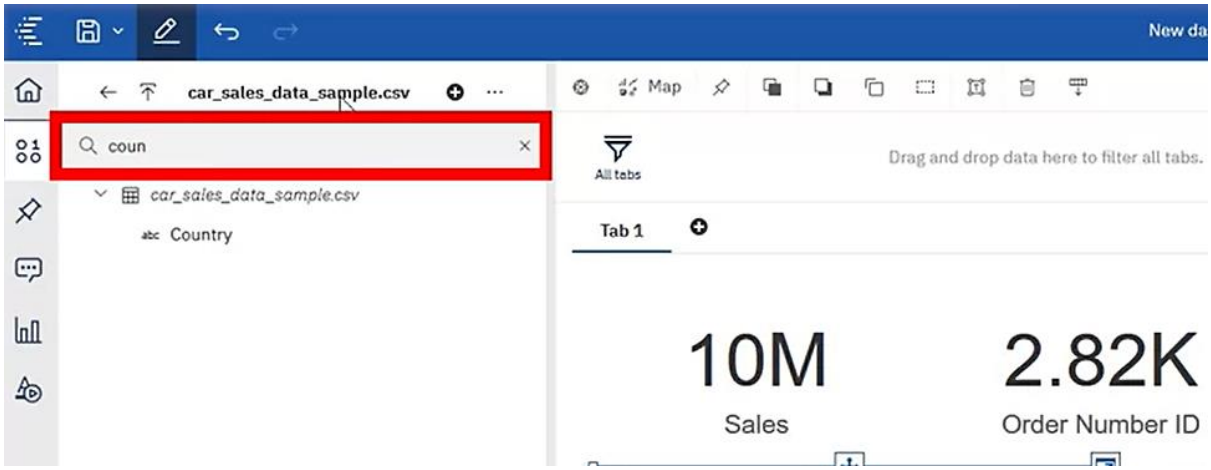

New dashboard *
Summary
Change visualization
Recommended visualizations
Automatic
All visualizations
Spiral
Stacked bar
Stacked column
Summary
Sunburst
Table

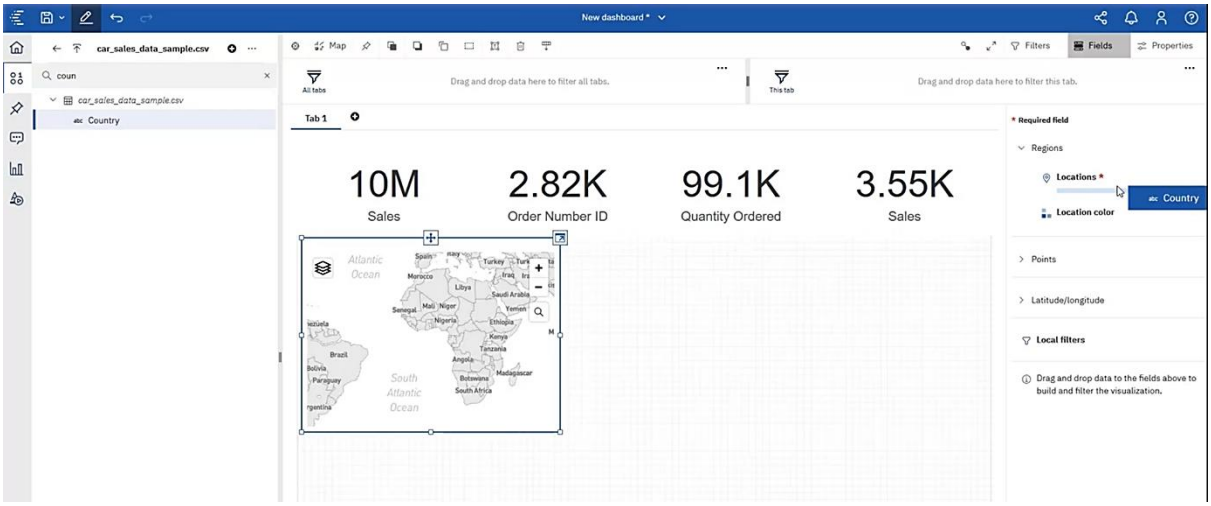

car_sales_data_sample.csv
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
All tabs
Drag and drop data here to filter all tabs.
This tab
Tab 1
10M
Sales
2.82K
Order Number ID
Quantity Ordered

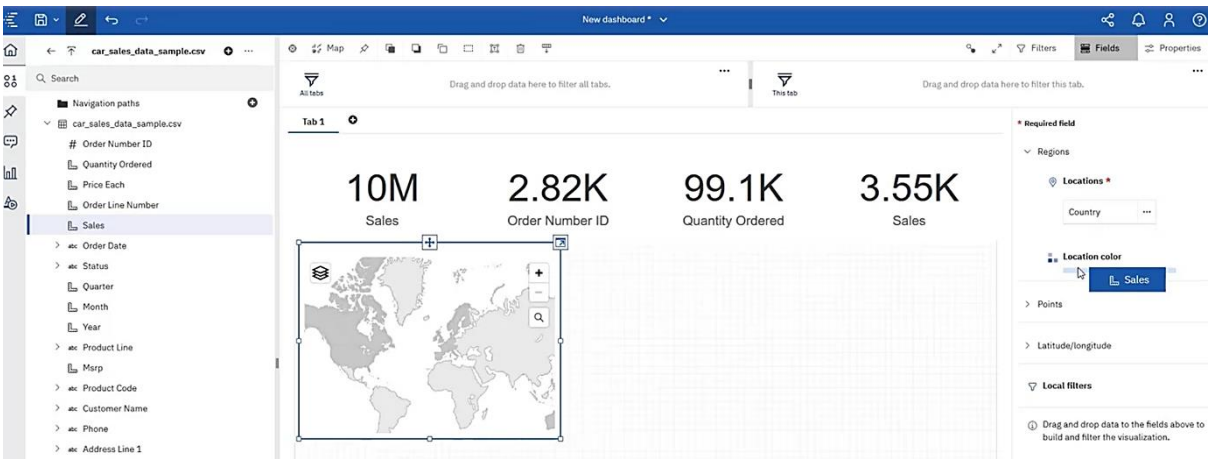

New dashboard *
car_sales_data_sample.csv
Summary
Filters
Fields
Properties
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
Sales

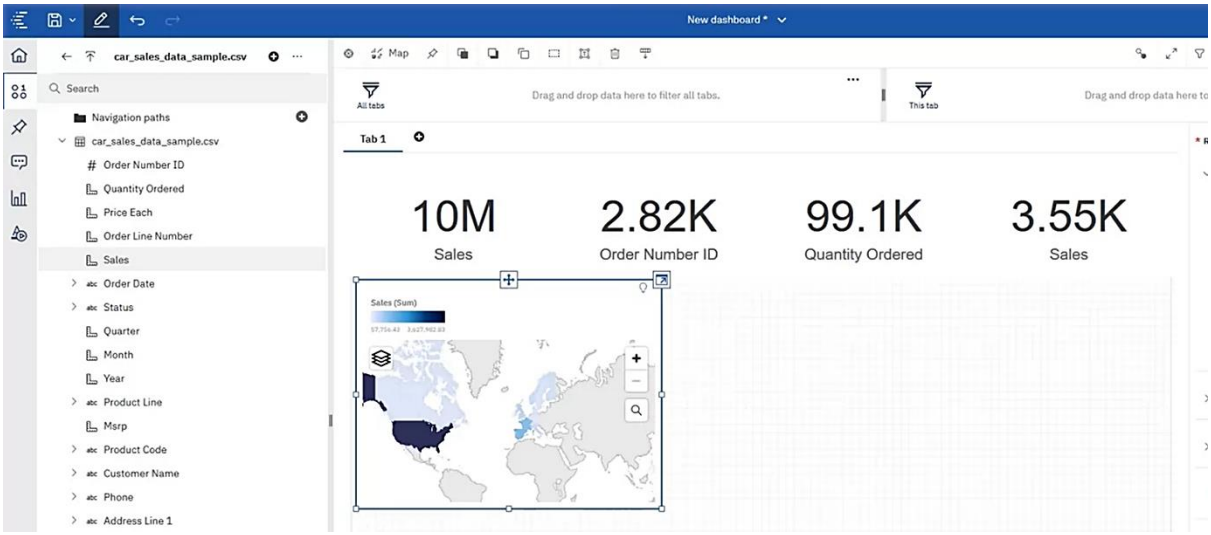

New dashboard *
car_sales_data_sample.csv
Summary
Filters
Fields
Properties
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Summarize
Average
Sum
Minimum
Maximum
Count
Count distinct
Auto (Sum)
Tab 1
10M
Sales
99.1K
Quantity Ordered
3.55K
Sales

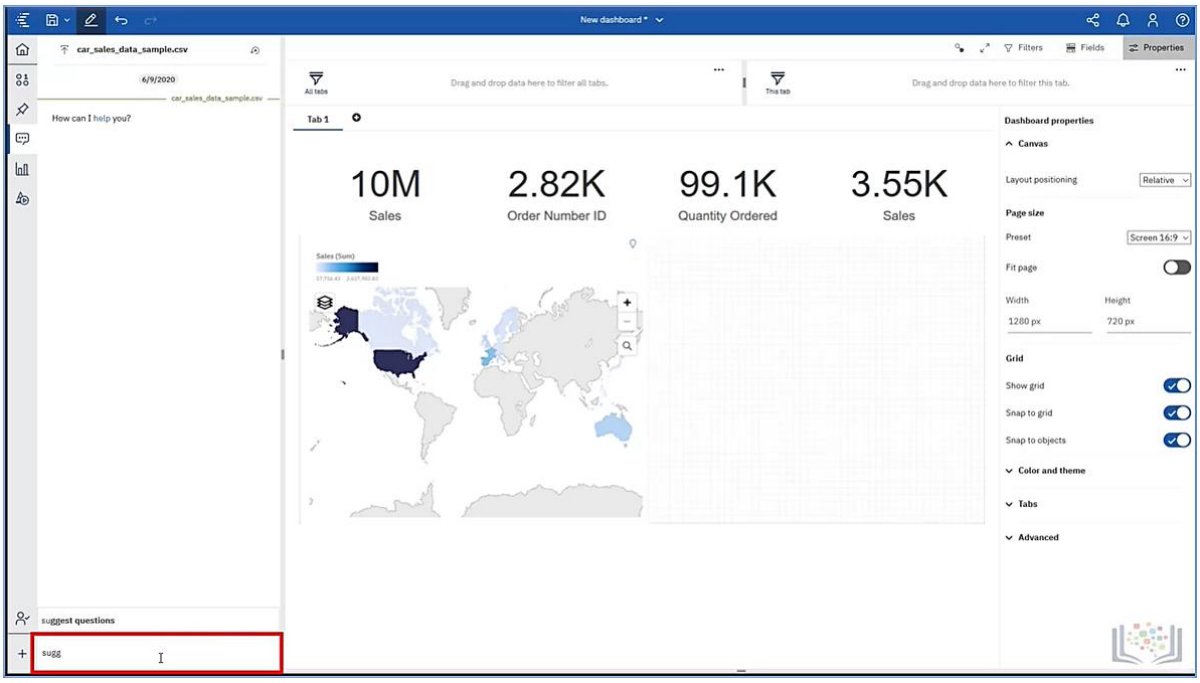

Visualizations
System
Custom
Area
Bar
Bubble
Bullet
Column
Crosstab
Data player
Decision tree
Driver analysis
Heat map
Hierarchy bubble
KPI
Legacy map
Line
Line and column
List
Map
Marimekko
Network
Packed bubble
Pie
Point
Radial
Scatter
Spiral
Stacked bar
Stacked column
Summary
Sunburst
Table
Tree map
Waterfall
Word cloud
All tabs
Drag and drop data here to filter all tabs.
This tab
Tab 1
10M
Sales
2.82K
Order Number ID
99.
Quantity


car_sales_data_sample.csv
coun
car_sales_data_sample.csv
Country
Map
All tabs
Drag and drop data here to filter all tabs.
Tab 1
10M
Sales
2.82K
Order Number ID

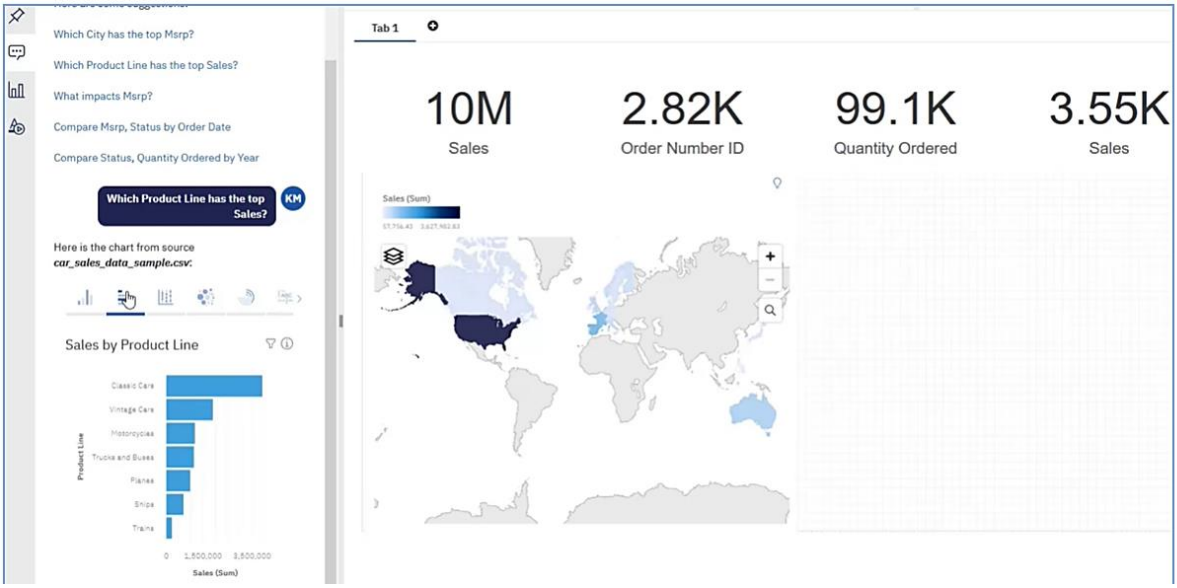

New dashboard *
car_sales_data_sample.csv
Map
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
* Required field
Regions
Locations *
Country
Location color
Points
Latitude/longitude
Local filters
Drag and drop data to the fields above to build and filter the visualization.

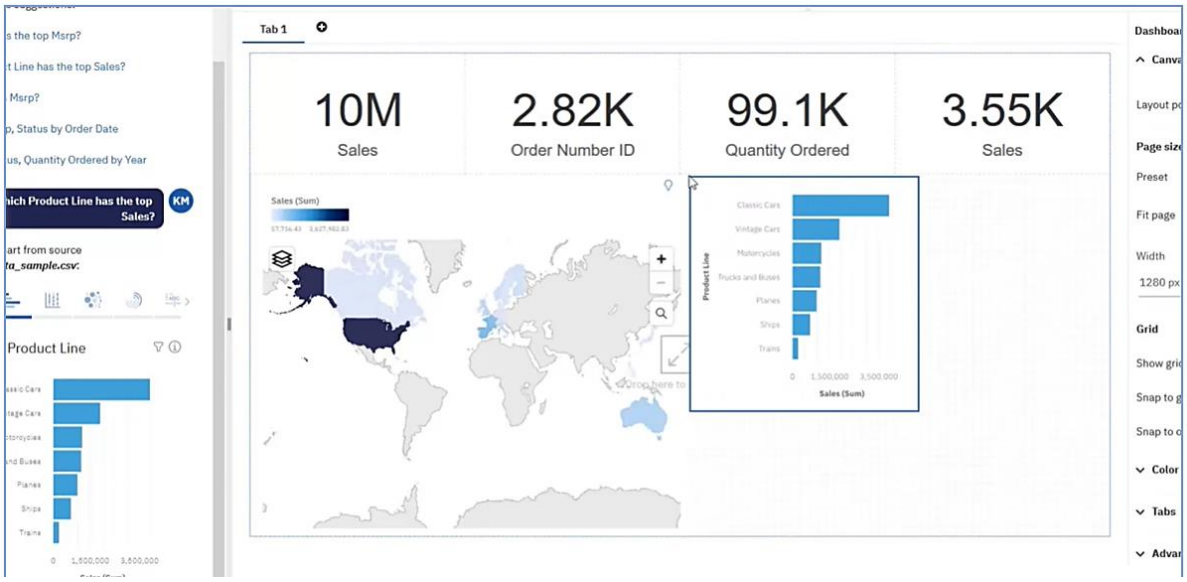

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Map
Filters
Fields
Properties
Drag and drop data here to filter all tabs.
Drag and drop data here to filter this tab.
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
* Required field
Regions
Locations *
Country
Location color
Sales
Points
Latitude/longitude
Local filters
Drag and drop data to the fields above to build and filter the visualization.

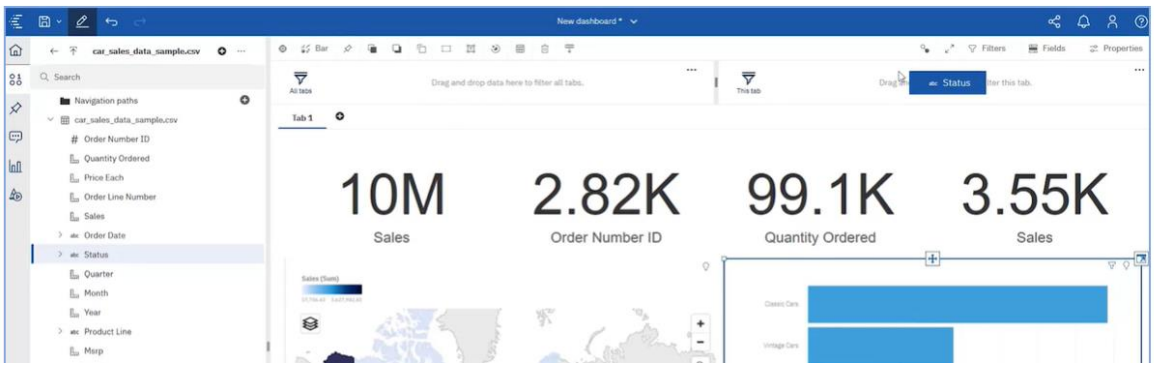

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
car_sales_data_sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Map
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
Sales (Sum)

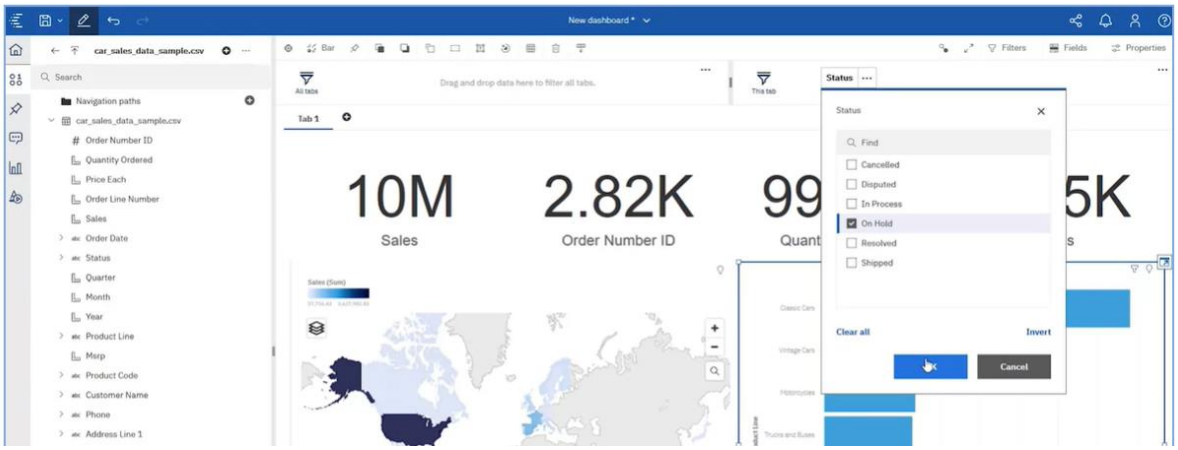

New dashboard *
car_sales_data_sample.csv
6/9/2020
How can I help you?
suggest questions
sugg
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
Sales (Sum)
Dashboard properties
Canvas
Layout positioning
Relative
Page size
Preset
Screen 16:9
Fit page
Width
1280 px
Height
720 px
Grid
Show grid
Snap to grid
Snap to objects
Color and theme
Tabs
Advanced

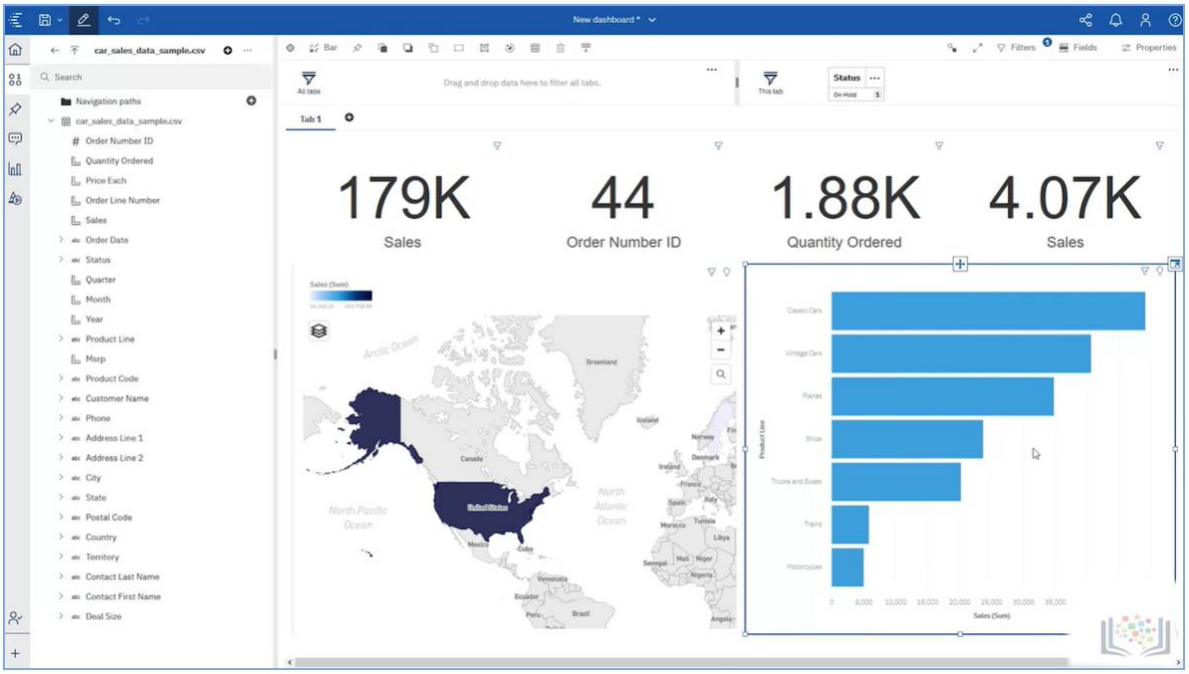

How can I help you?
suggest questions
KM
Here are some suggestions:
Which City has the top Msrp?
Which Product Line has the top Sales?
What impacts Msrp?
Compare Msrp, Status by Order Date
Compare Status, Quantity Ordered by Year
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
Sales (Sum)


Which City has the top Msrp?
Which Product Line has the top Sales?
What impacts Msrp?
Compare Msrp, Status by Order Date
Compare Status, Quantity Ordered by Year
Which Product Line has the top Sales?
KM
Here is the chart from source
car_sales_data_sample.csv:
Sales by Product Line
Classic Cars
Vintage Cars
Motorcycles
Trucks and Buses
Planes
Ships
Trains
Product Line
Sales (Sum)
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales

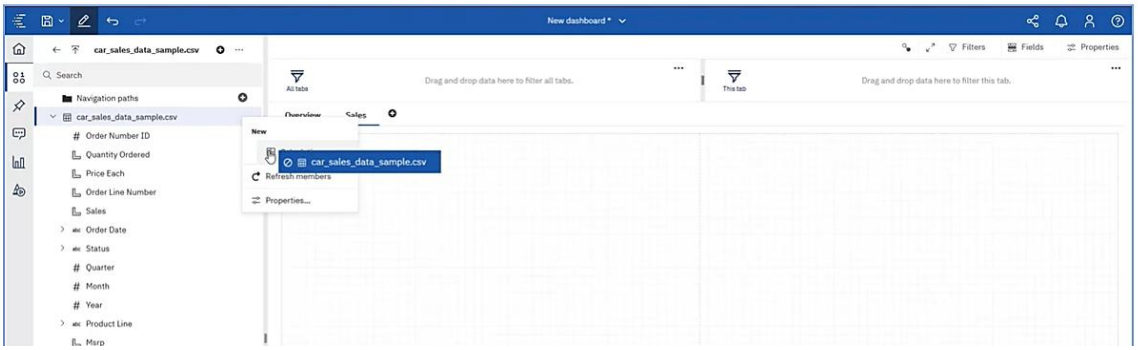

Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales
Product Line
Sales (Sum)
Dashboard
Layout
Page size
Preset
Fit page
Width
1280 px
Grid
Color
Tabs

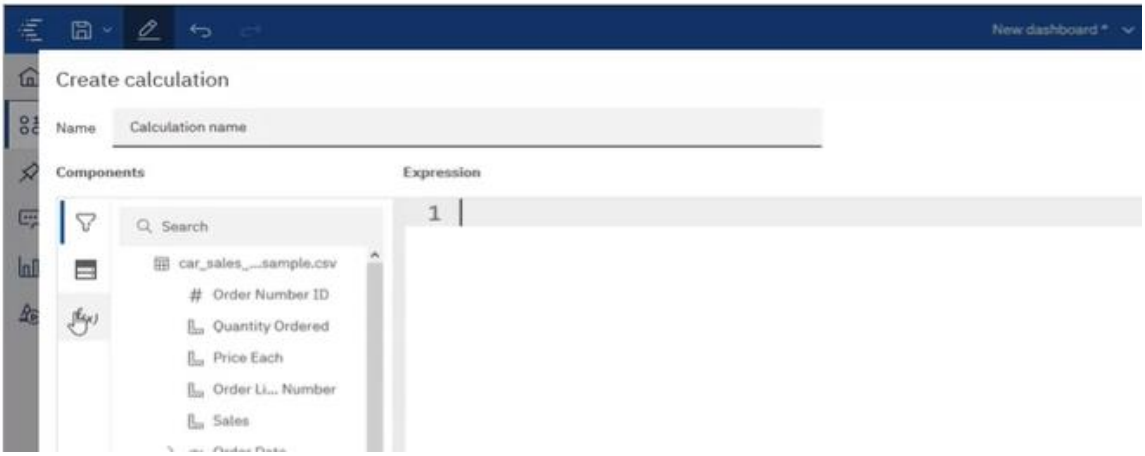

New dashboard *
car_sales_data_sample.csv
Navigation paths
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Filters
Fields
Properties
Tab 1
10M
Sales
2.82K
Order Number ID
99.1K
Quantity Ordered
3.55K
Sales

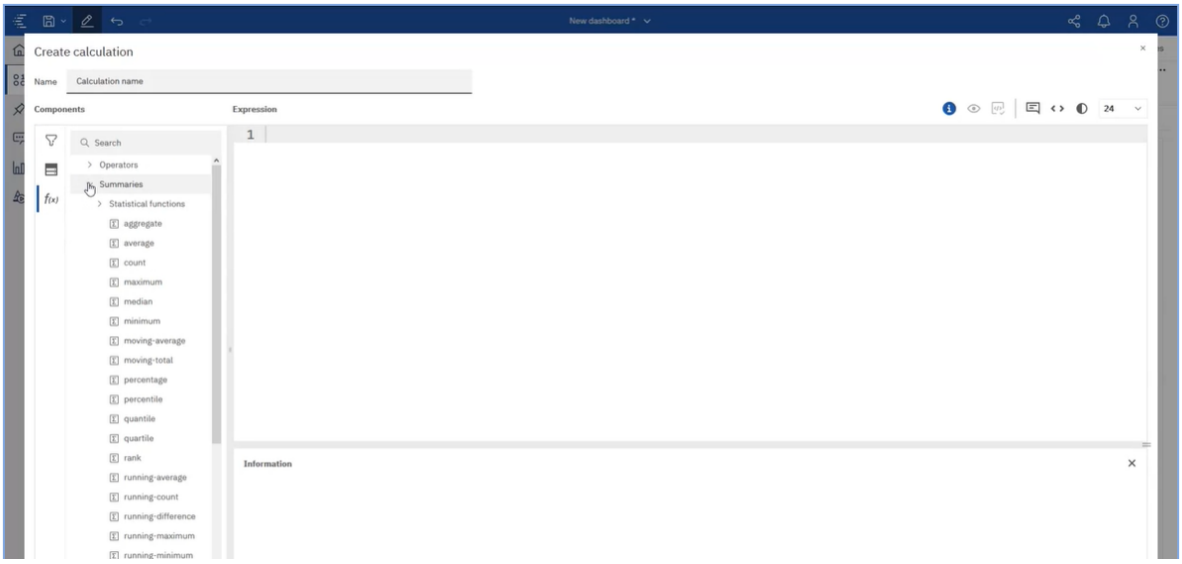


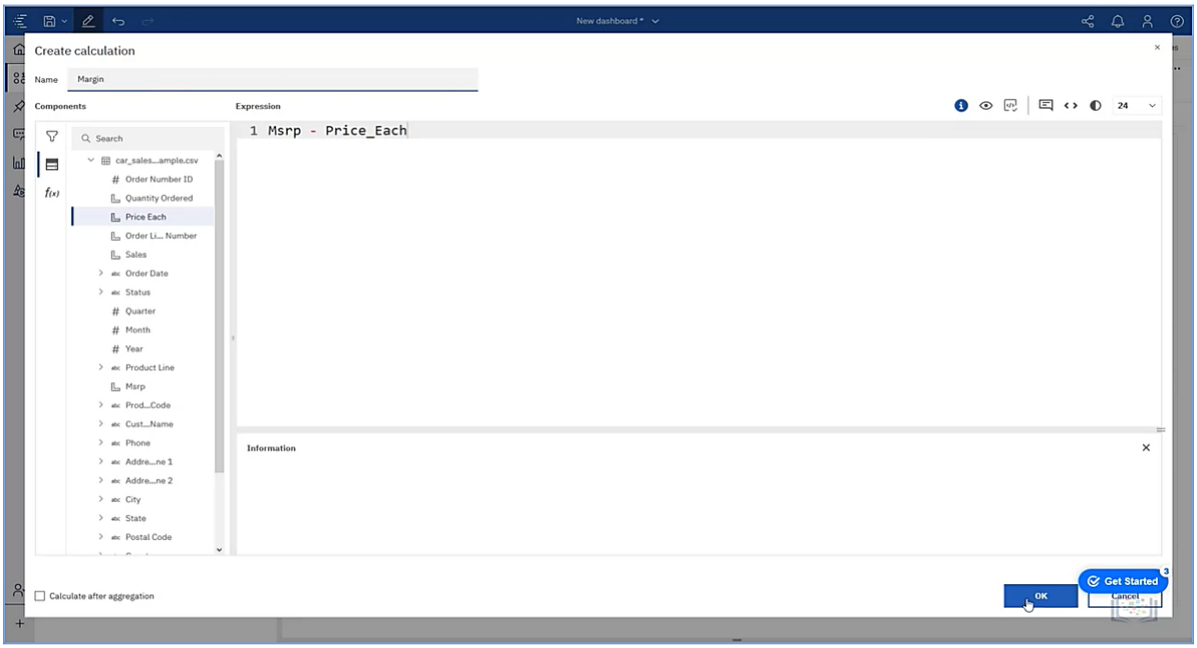


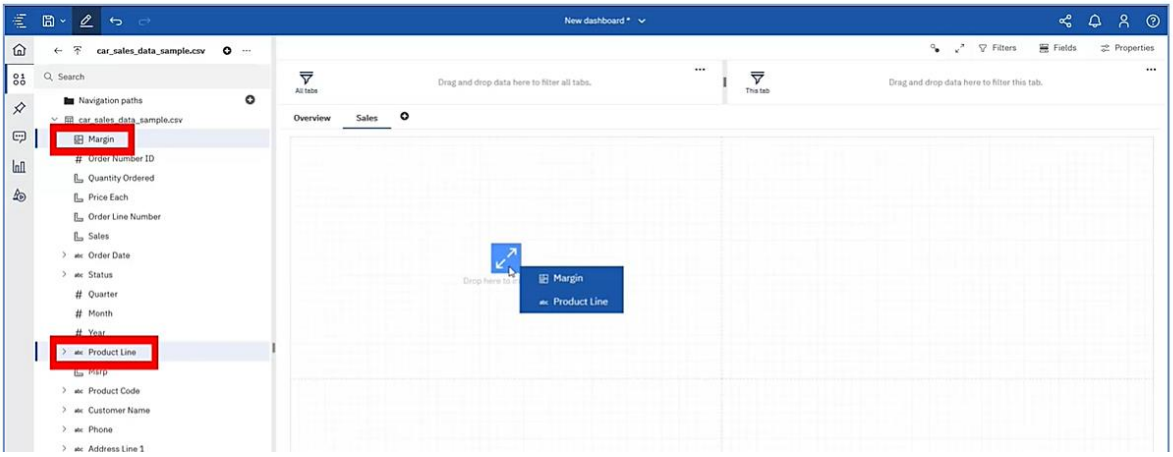
4] Advanced Capabilities in Cognos Analytics Dashboards

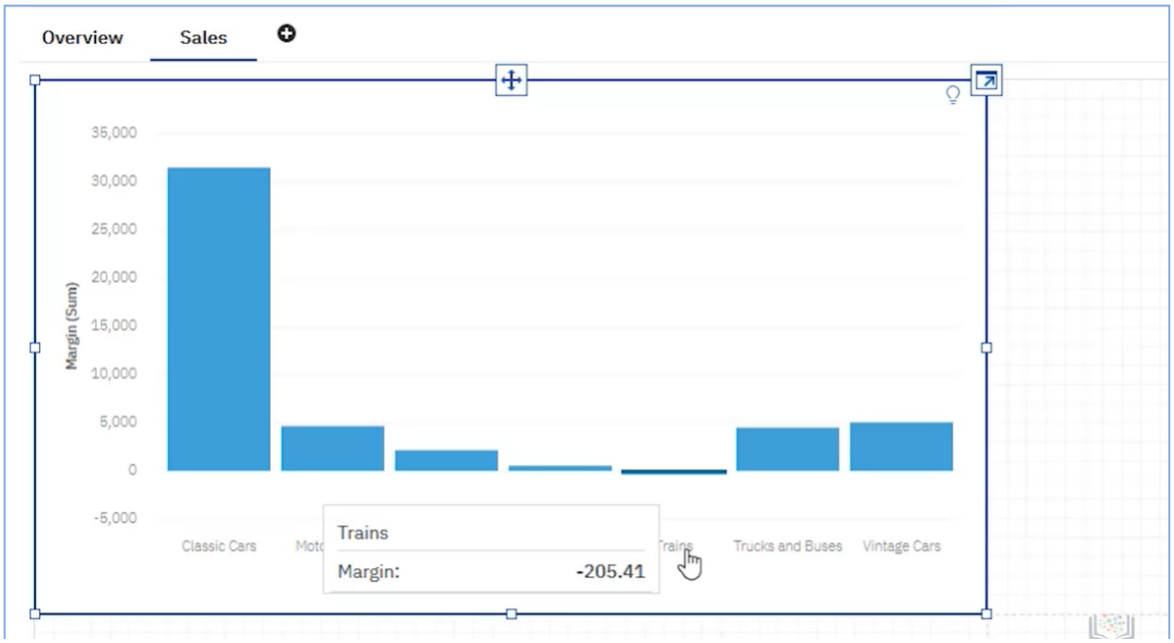


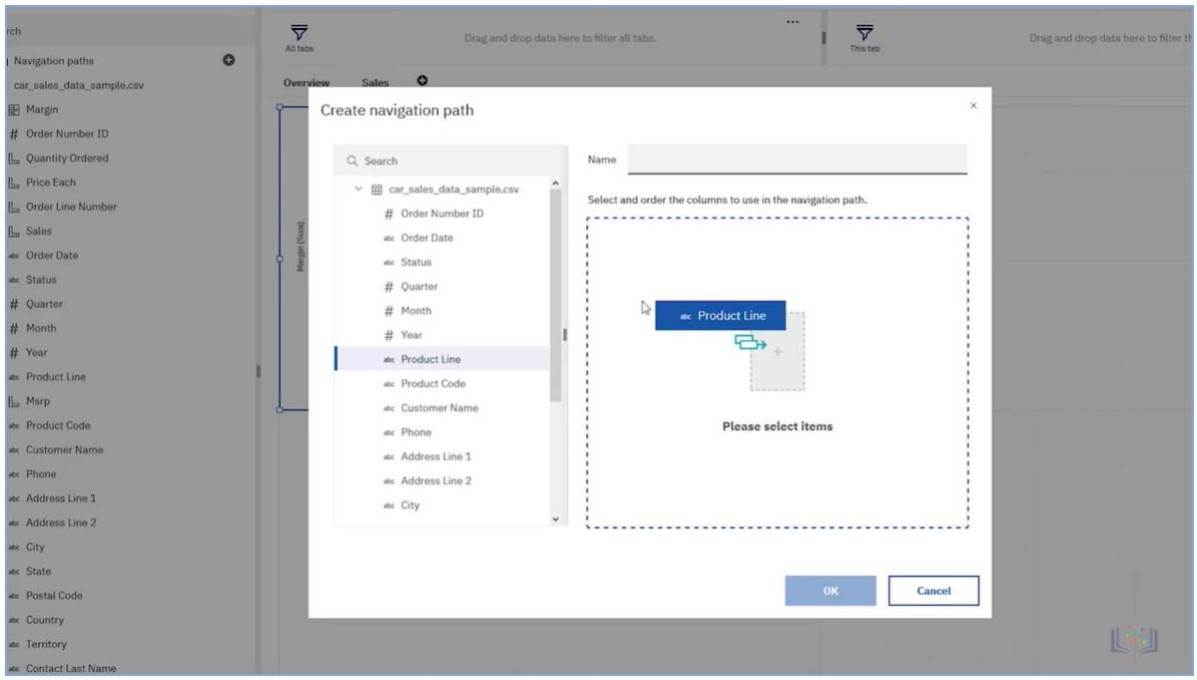

New dashboard *
Create calculation
Name
Calculation name
Components
Expression
1
Search
car_sales_....sample.csv
Order Number ID
Quantity Ordered
Price Each
Order Li... Number
Sales

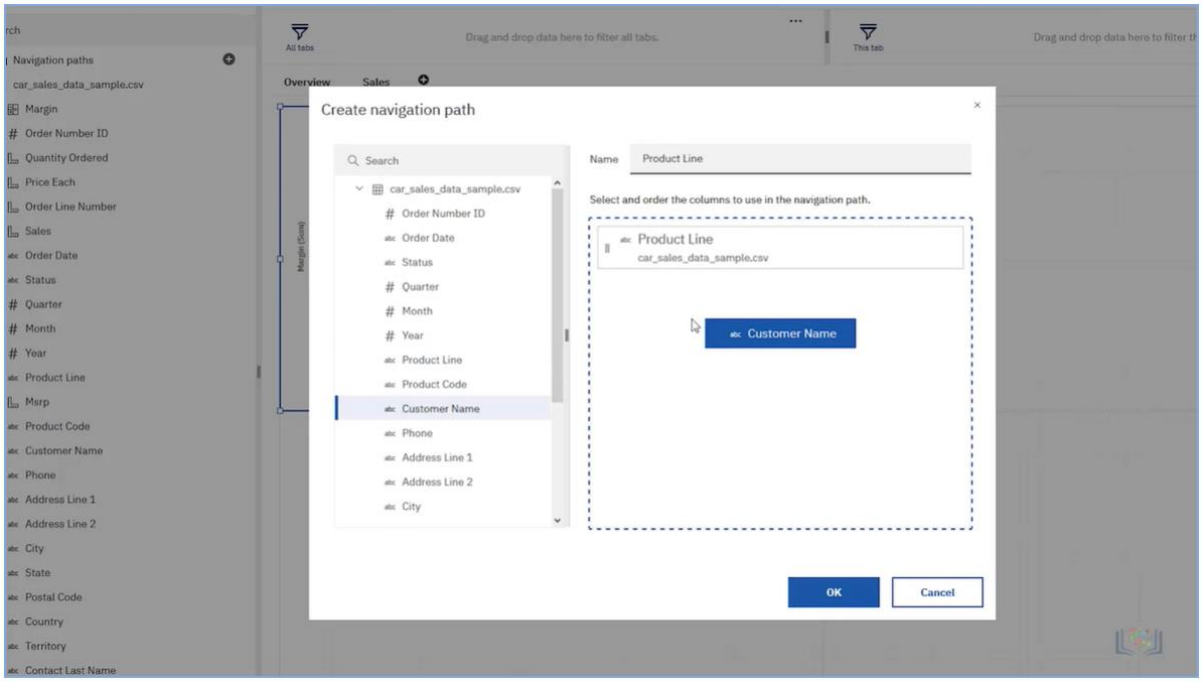

New dashboard *
Create calculation
Name
Calculation name
Components
Expression
1
Search
Operators
Summaries
Statistical functions
aggregate
average
count
maximum
median
minimum
moving-average
moving-total
percentage
percentile
quantile
quartile
rank
running-average
running-count
running-difference
running-maximum
running-minimum
Information

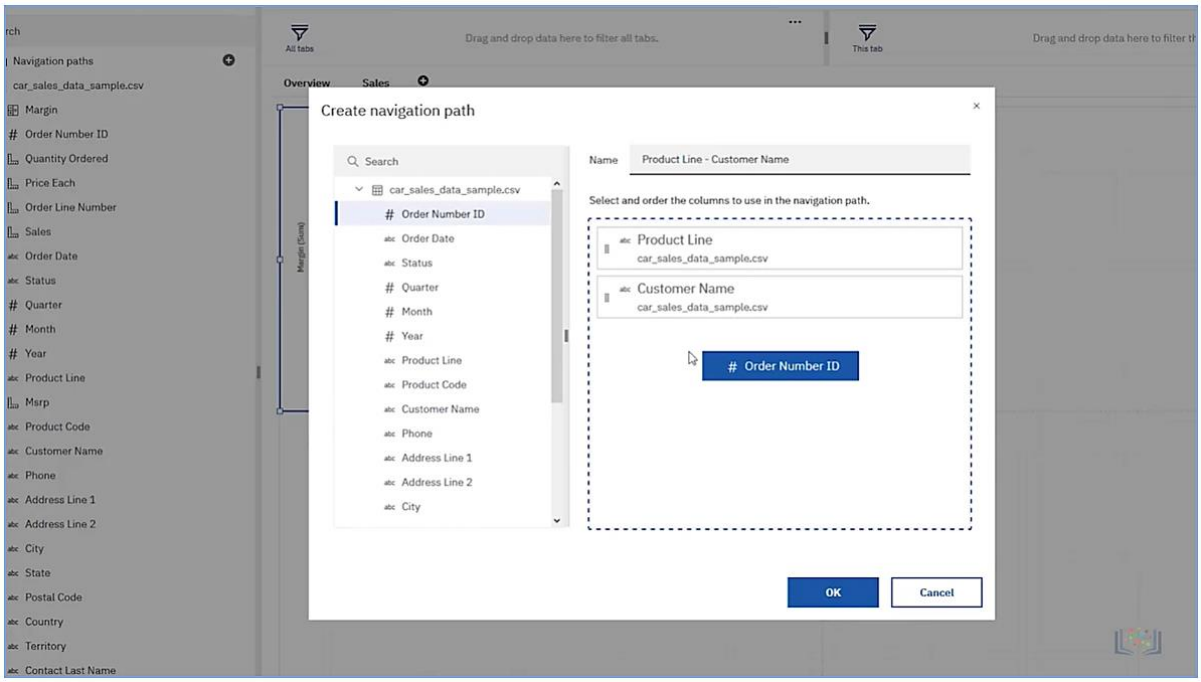

New dashboard *
Create calculation
Name
Margin
Components
Expression
1 Msrp - Price_Each
Search
car_sales...ample.csv
Order Number ID
Quantity Ordered
Price Each
Order Li... Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Prod...Code
Cust...Name
Phone
Addre...ne 1
Addre...ne 2
City
State
Postal Code
Information
Calculate after aggregation
Get Started
OK
Cancel

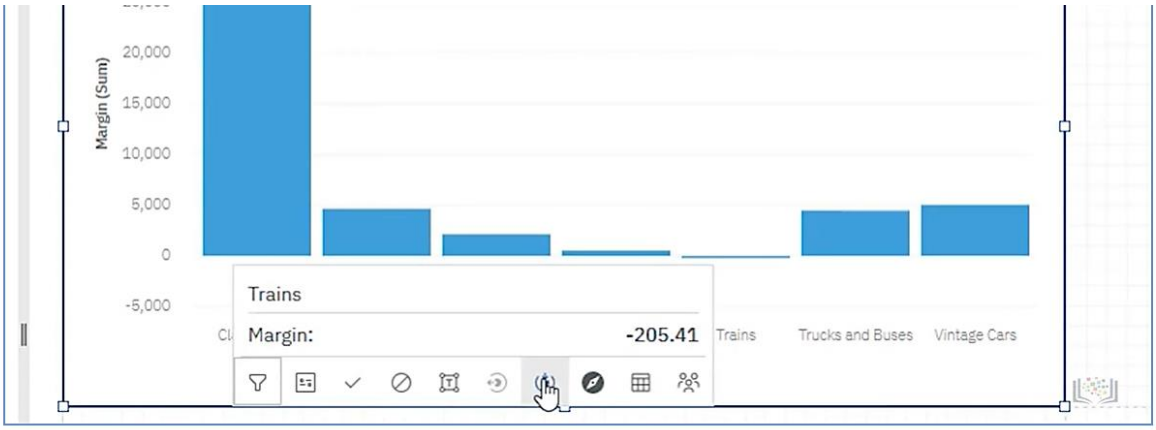

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Filters
Fields
Properties
Drag and drop data here to filter all tabs.
Drag and drop data here to filter this tab.
Overview
Sales
Margin
Product Line

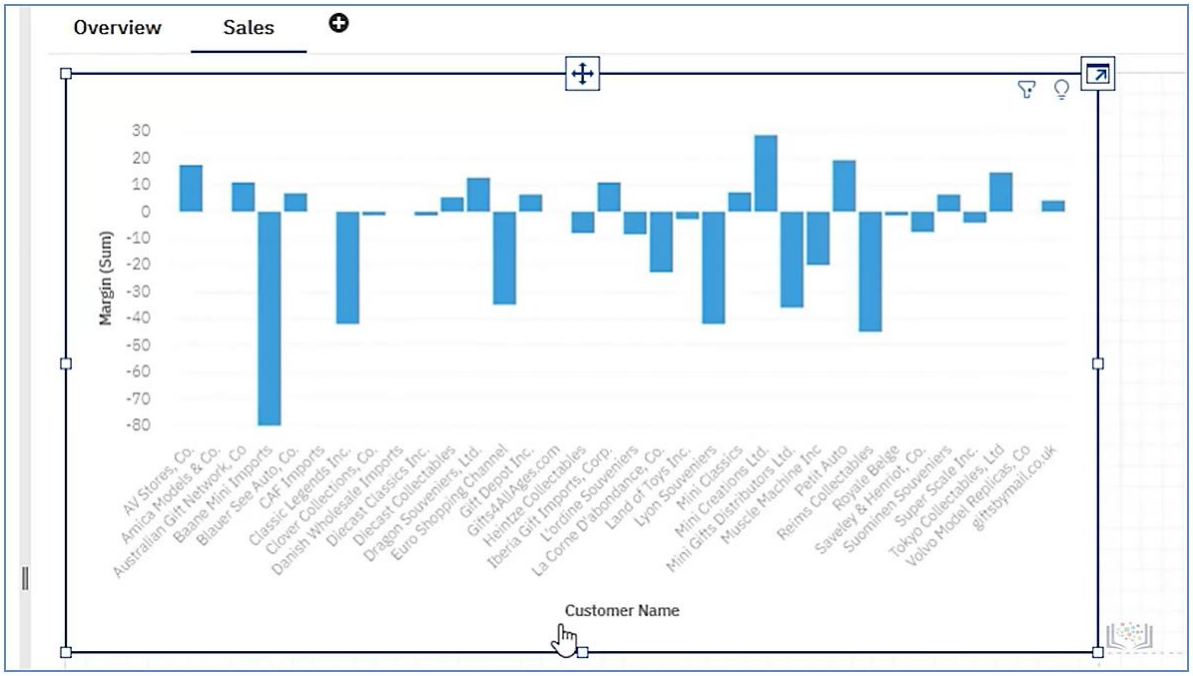

Overview
Sales
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
-5,000
Margin (Sum)
Classic Cars
Trains
Trucks and Buses
Vintage Cars
Trains
Margin: -205.41


Navigation paths
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
Drag and drop data here to filter all tabs.
Overview
Sales
Create navigation path
Search
Name
Select and order the columns to use in the navigation path.
Product Line
Please select items
OK
Cancel


Navigation paths
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter th
Overview
Sales
Create navigation path
Search
Name
Product Line
Select and order the columns to use in the navigation path.
Product Line
car_sales_data_sample.csv
Customer Name
OK
Cancel

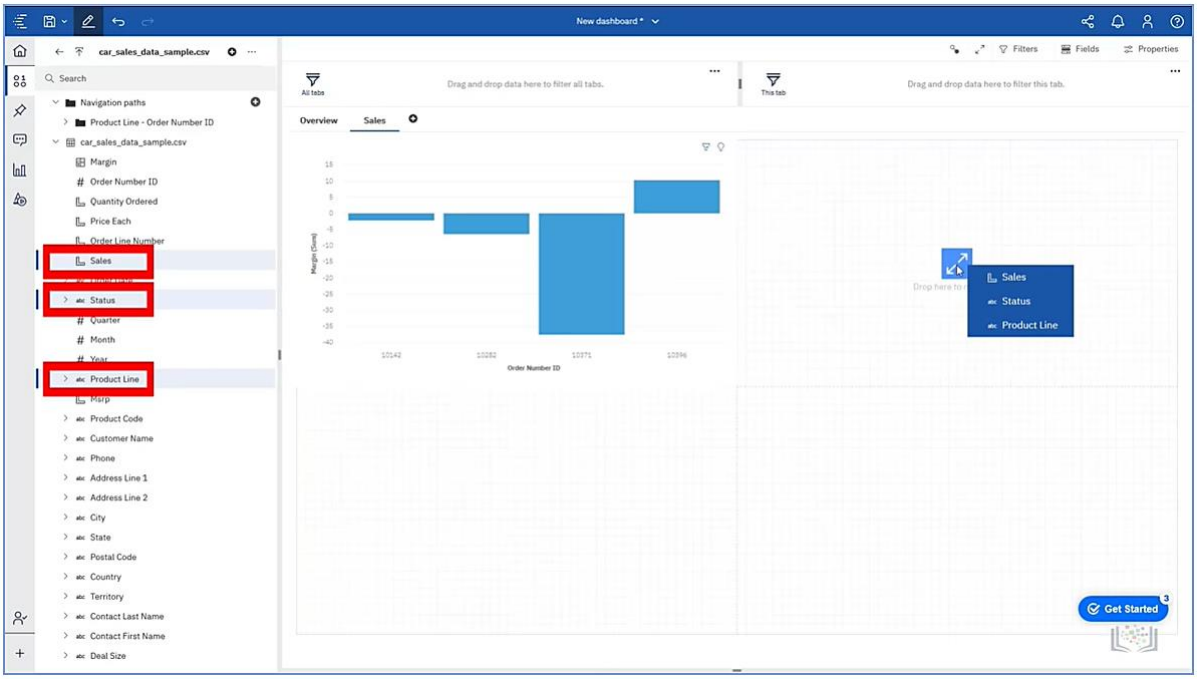

Navigation paths
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter th
Overview
Sales
Create navigation path
Search
Name
Product Line - Customer Name
Select and order the columns to use in the navigation path.
Product Line
car_sales_data_sample.csv
Customer Name
car_sales_data_sample.csv
Order Number ID
OK
Cancel

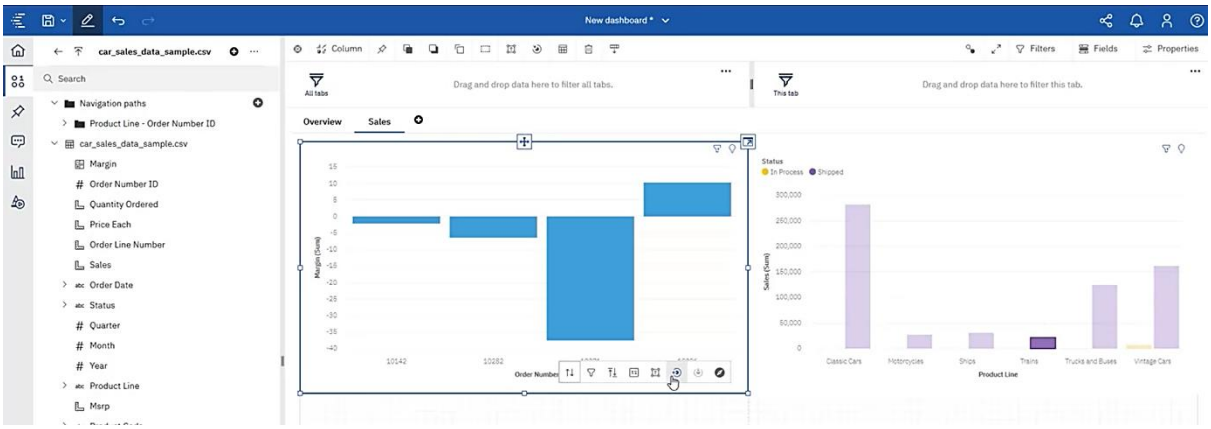

Margin (Sum)
20,000
15,000
10,000
5,000
0
-5,000
Trains
Margin:
-205.41
Trains
Trucks and Buses
Vintage Cars

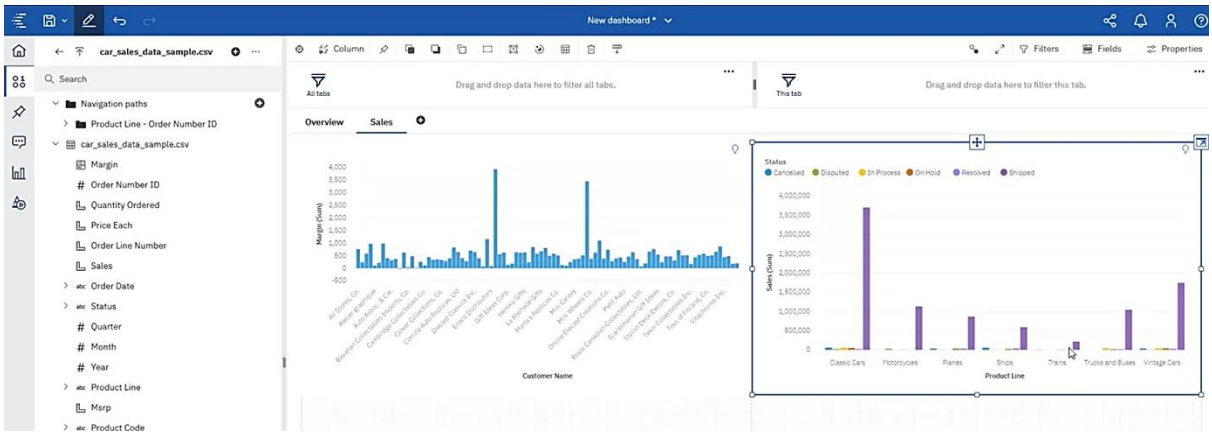

Overview
Sales
Margin (Sum)
Customer Name


Mini Gifts Distributors Ltd.
Margin: -35.9
Customer Name

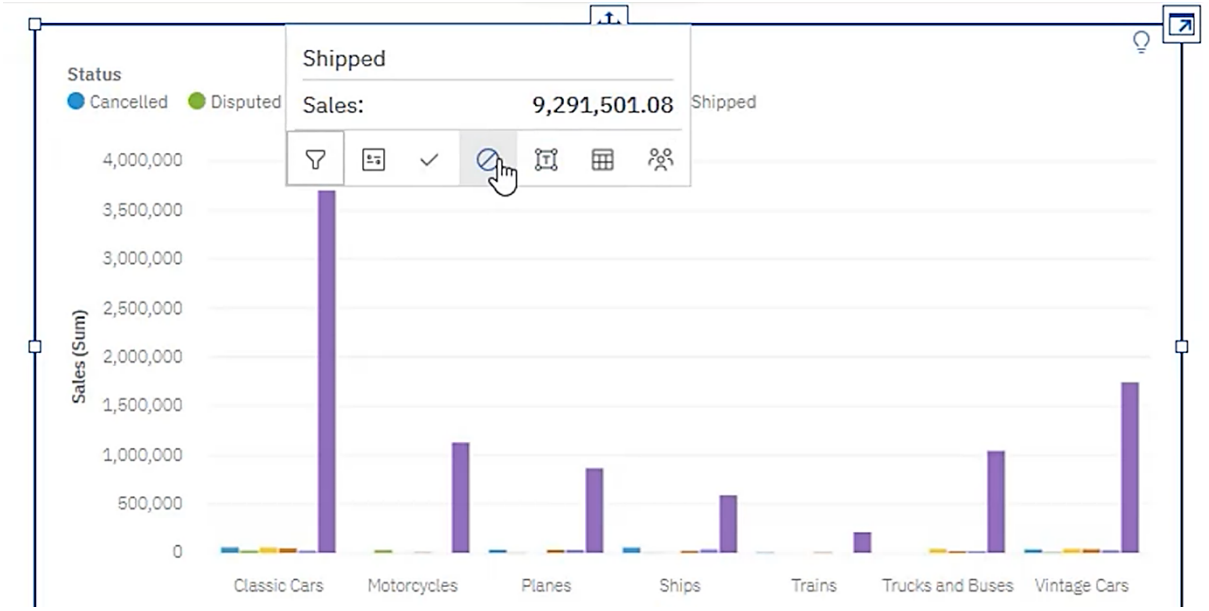

Overview
Sales
10396
Margin: 10.25
Margin (Sum)
10142
10282
10371
10396
Order Number ID

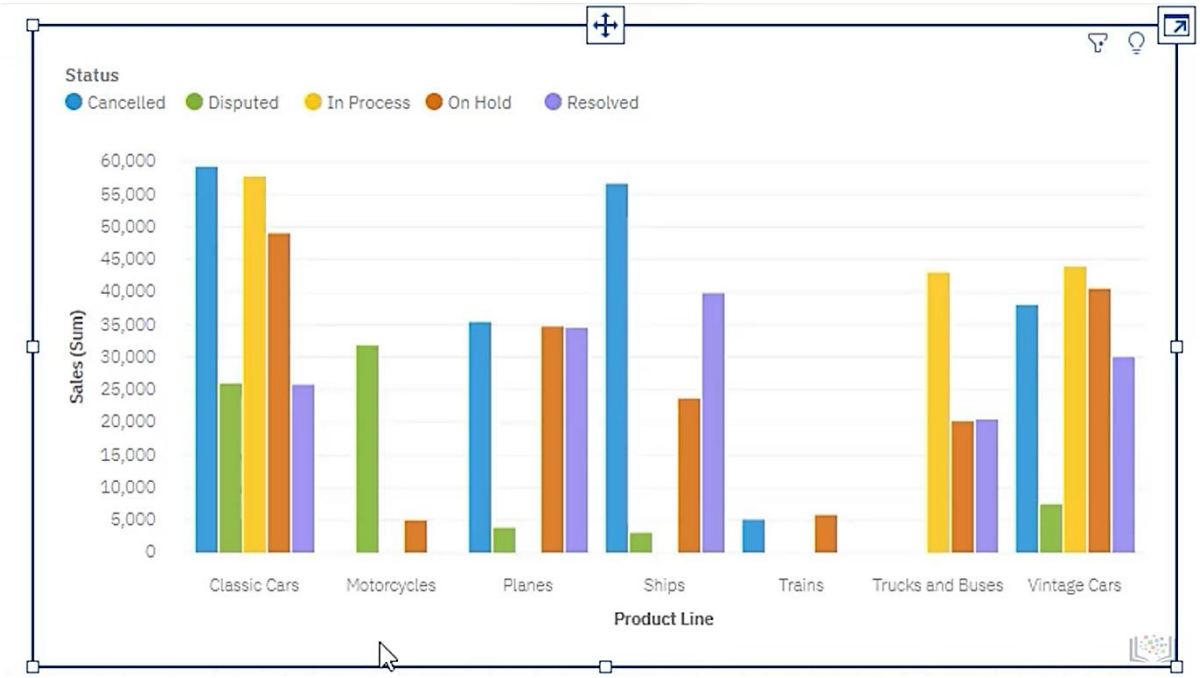

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
Product Line - Order Number ID
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
Contact First Name
Deal Size
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Overview
Sales
Margin (Sum)
Order Number ID
Sales
Status
Product Line
Get Started

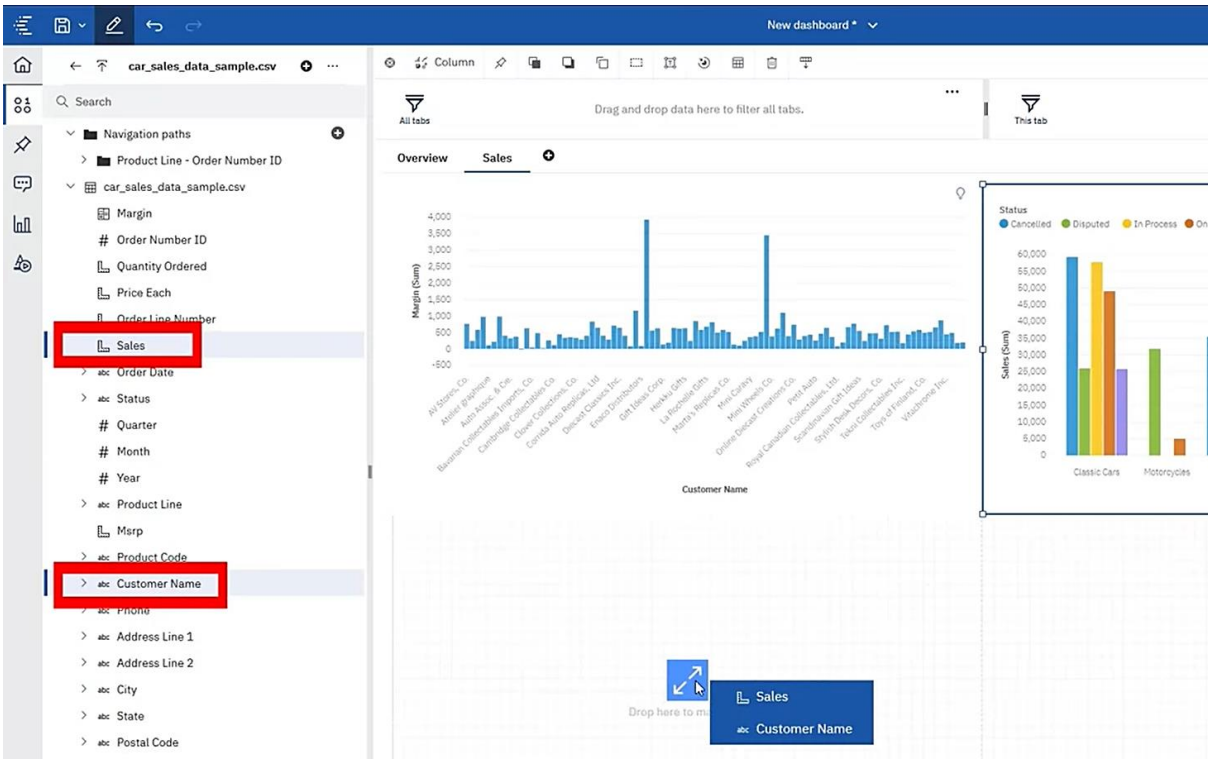

New dashboard *
car_sales_data_sample.csv
Column
Search
Navigation paths
Product Line - Order Number ID
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Overview
Sales
Margin (Sum)
Order Number
Status
In Process
Shipped
Sales (Sum)
Classic Cars
Motorcycles
Ships
Trains
Trucks and Buses
Vintage Cars
Product Line

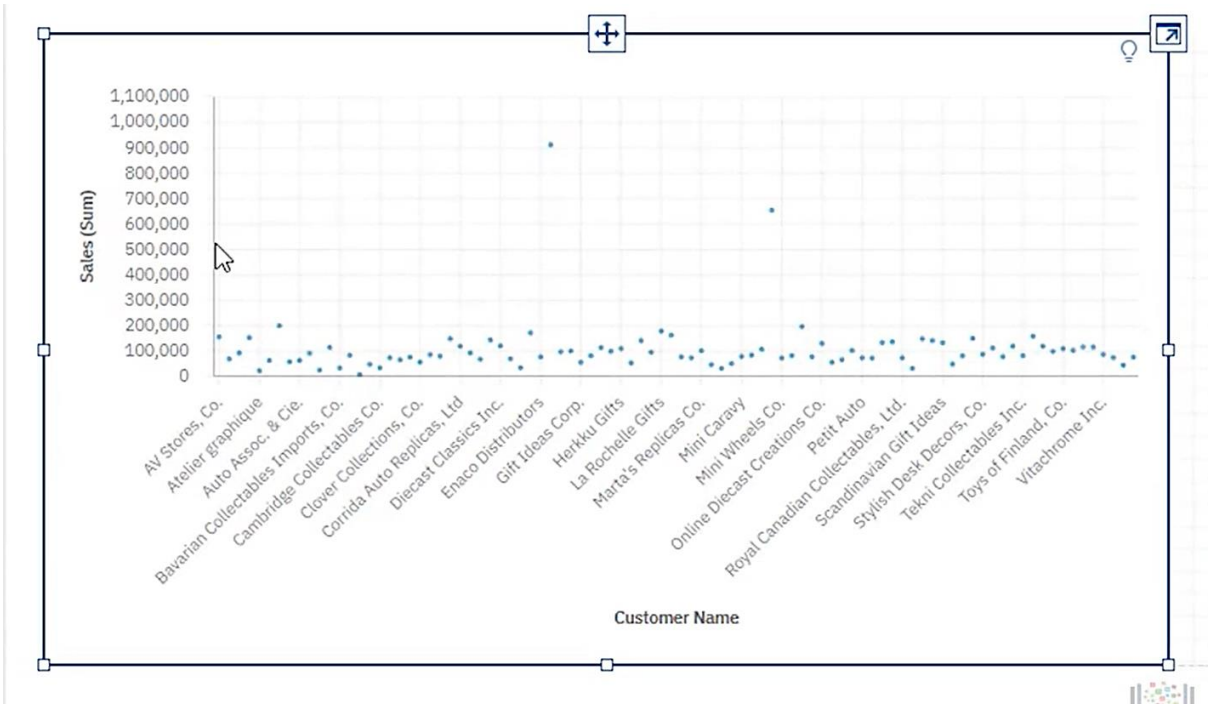

New dashboard *
car_sales_data_sample.csv
Column
Search
Navigation paths
Product Line - Order Number ID
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Overview
Sales
Margin (Sum)
Customer Name
Status
Cancelled
Disputed
In Process
On Hold
Resolved
Shipped
Sales (Sum)
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars
Product Line

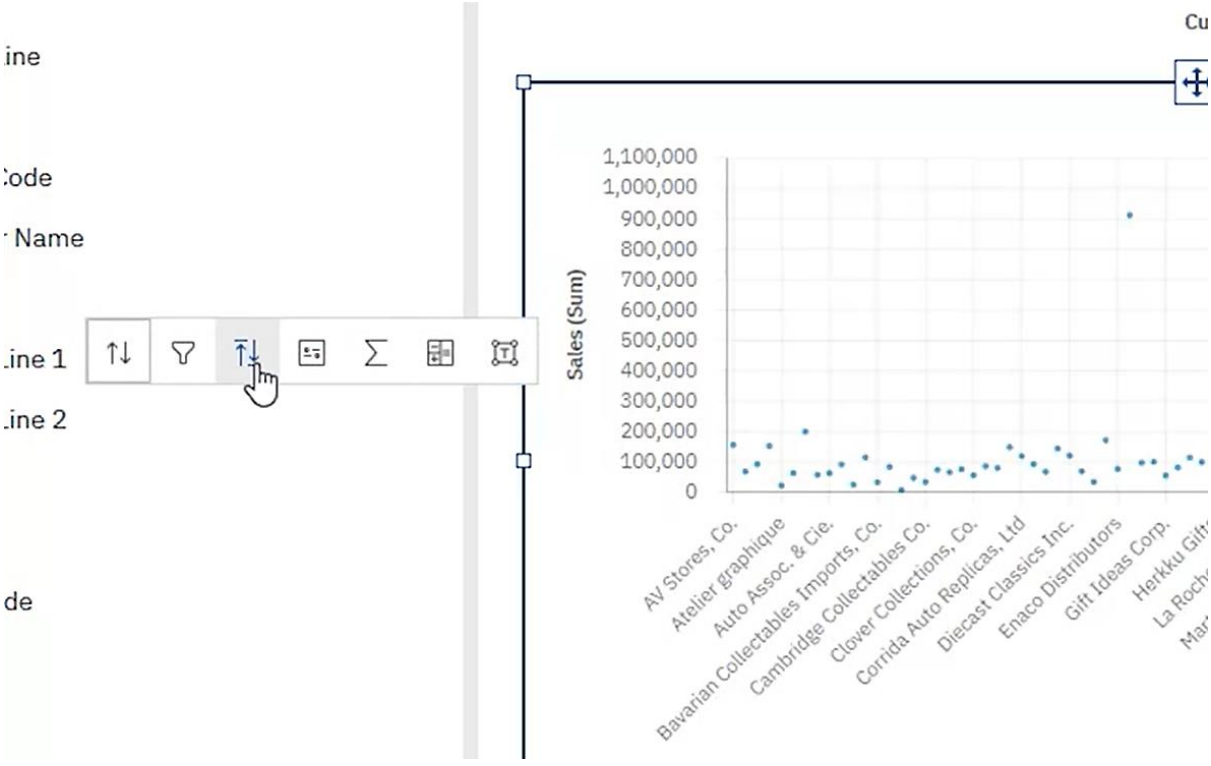

Status
Cancelled
Disputed
Shipped
Shipped
Sales: 9,291,501.08
4,000,000
3,500,000
3,000,000
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Sales (Sum)
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars
Product Line

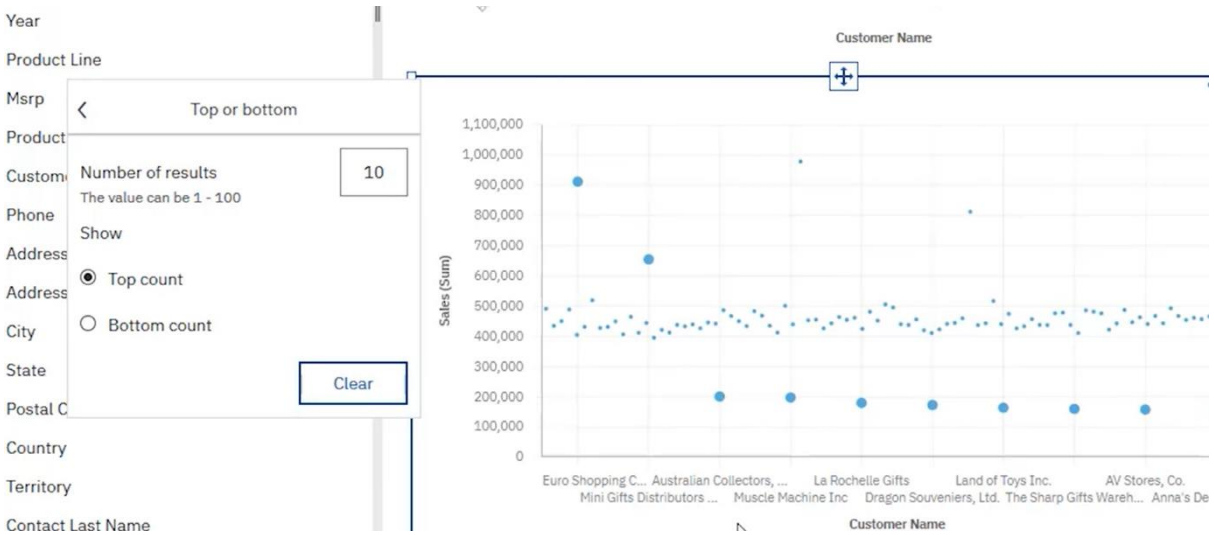

Shipped
Sales: 9,291,501.08
Status
Cancelled
Disputed
Shipped
4,000,000
3,500,000
3,000,000
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Sales (Sum)
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars

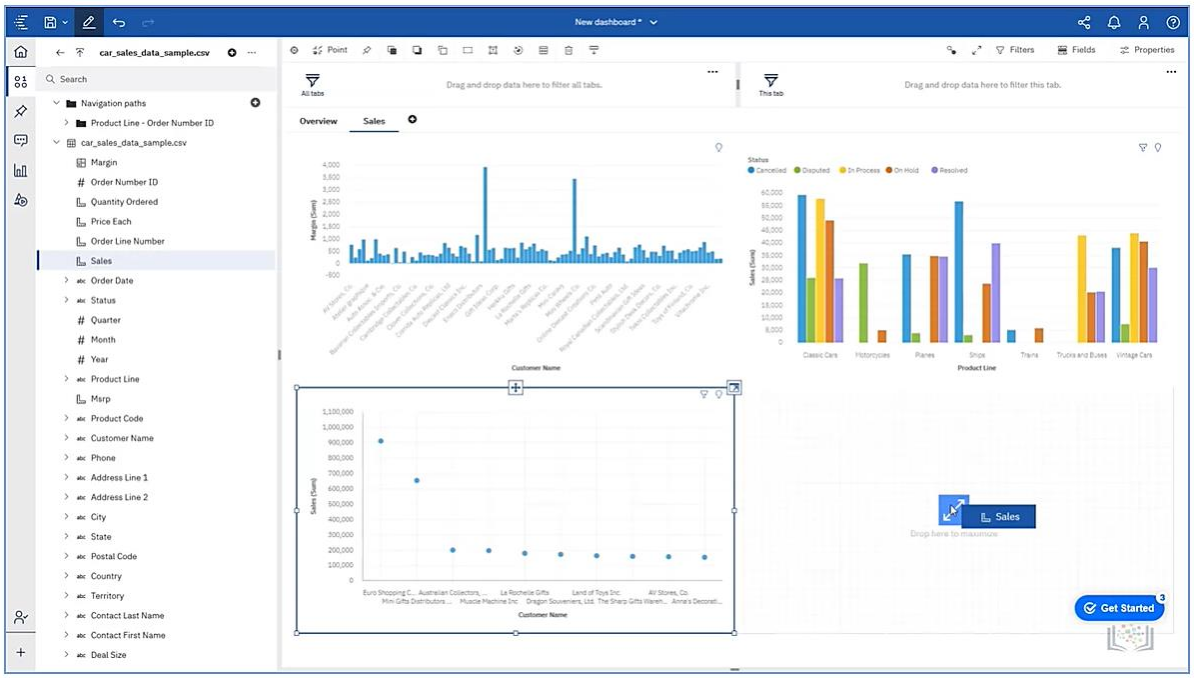

Status
Cancelled
Disputed
In Process
On Hold
Resolved
60,000
55,000
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
Sales (Sum)
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars
Product Line

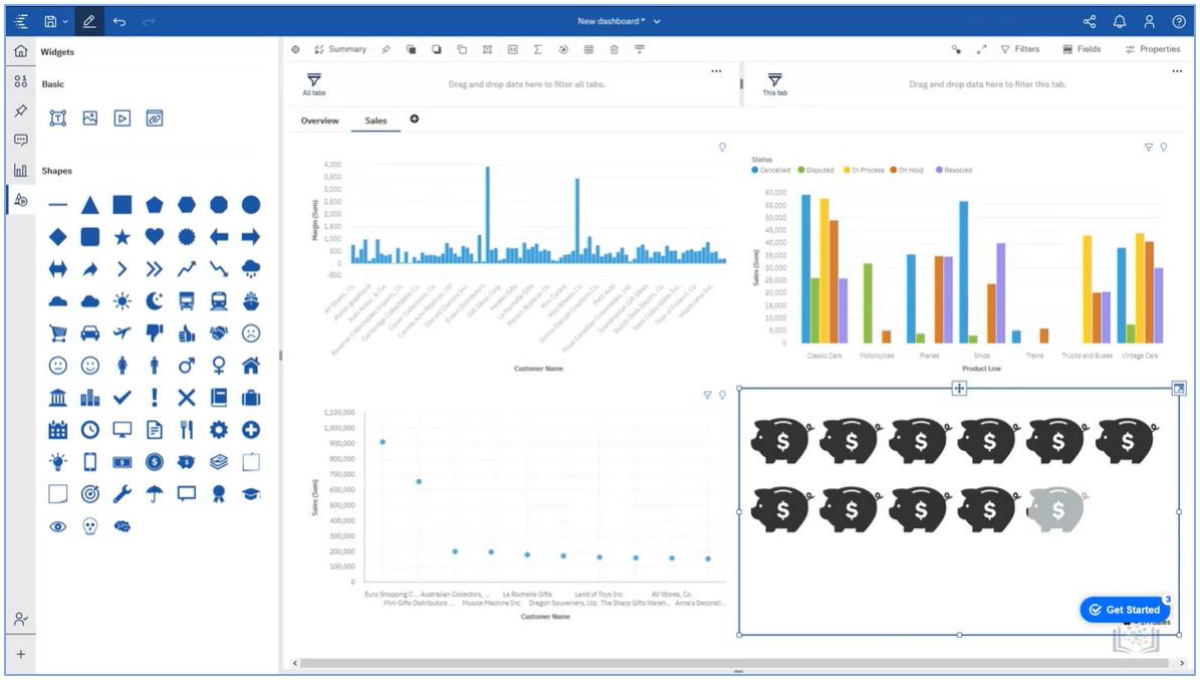

New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
Product Line - Order Number ID
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Column
All tabs
Drag and drop data here to filter all tabs.
This tab
Overview
Sales
Margin (Sum)
Customer Name
Status
Cancelled
Disputed
In Process
Sales (Sum)
Classic Cars
Motorcycles
Sales
Customer Name


1,100,000
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
Sales (Sum)
AV Stores, Co.
Atelier graphique
Auto Assoc. & Cie.
Bavarian Collectables Imports, Co.
Cambridge Collectables Co.
Clover Collections, Co.
Corrida Auto Replicas, Ltd
Diecast Classics Inc.
Enaco Distributors
Gift Ideas Corp.
Herkku Gifts
La Rochelle Gifts
Marta's Replicas Co.
Mini Caravy
Mini Wheels Co.
Online Diecast Creations Co.
Petit Auto
Royal Canadian Collectables, Ltd.
Scandinavian Gift Ideas
Stylish Desk Decors, Co.
Tekni Collectables Inc.
Toys of Finland, Co.
Vitachrome Inc.
Customer Name


ine
Code
Name
Line 1
Line 2
de
Cu
1,100,000
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
Sales (Sum)
AV Stores, Co.
Atelier graphique
Auto Assoc. & Cie.
Bavarian Collectables Imports, Co.
Cambridge Collectables Co.
Clover Collections, Co.
Corrida Auto Replicas, Ltd
Diecast Classics Inc.
Enaco Distributors
Gift Ideas Corp.


Year
Product Line
Msrp
Product
Custom
Phone
Address
Address
City
State
Postal C
Country
Territory
Contact Last Name
Top or bottom
Number of results
10
The value can be 1 - 100
Show
Top count
Bottom count
Clear
Customer Name
Sales (Sum)
1,100,000
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
Euro Shopping C...
Australian Collectors, ...
La Rochelle Gifts
Land of Toys Inc.
AV Stores, Co.
Mini Gifts Distributors ...
Muscle Machine Inc
Dragon Souveniers, Ltd.
The Sharp Gifts Wareh...
Anna's De
Customer Name


New dashboard *
car_sales_data_sample.csv
Search
Navigation paths
Product Line - Order Number ID
car_sales_data_sample.csv
Margin
Order Number ID
Quantity Ordered
Price Each
Order Line Number
Sales
Order Date
Status
Quarter
Month
Year
Product Line
Msrp
Product Code
Customer Name
Phone
Address Line 1
Address Line 2
City
State
Postal Code
Country
Territory
Contact Last Name
Contact First Name
Deal Size
Point
Filters
Fields
Properties
All tabs
Drag and drop data here to filter all tabs.
This tab
Drag and drop data here to filter this tab.
Overview
Sales
Status
Canceled
Disputed
In Process
On Hold
Resolved
Product Line
Classic Cars
Motorcycles
Planes
Ships
Trains
Trucks and Buses
Vintage Cars
Customer Name
Sales
Drop here to maximize
Get Started
3



5] Accessing Your Warehouse with Cognos

Accessing databases – overview –

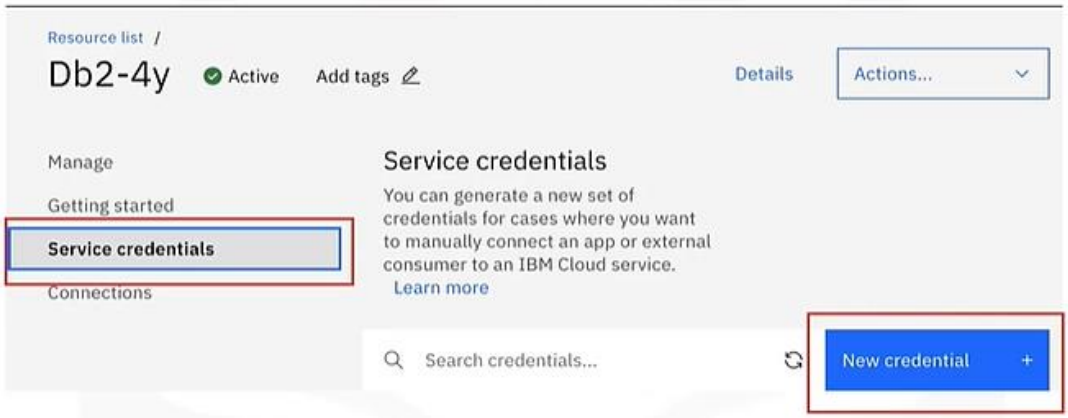
Flexible access to many databases and data warehouse system:

- IBM Db2 Warehouse
- Amazon Redshift
- Oracle
- Microsoft SQL Server
- MongoDB
- MySQL
- PostgreSQL
- Snowflake

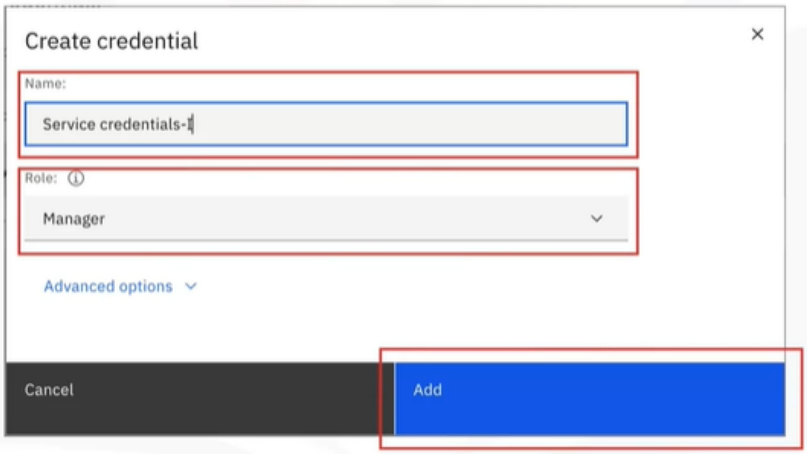
Creating a report or dashboard using a database:

1. Identify the databases, schemas, and tables and obtain connection credentials
2. Create data server connections
3. Create data modules for the selected tables
4. Create a dashboard and add data modules as sources
5. Create visualizations: drag columns from the data module onto the dashboard

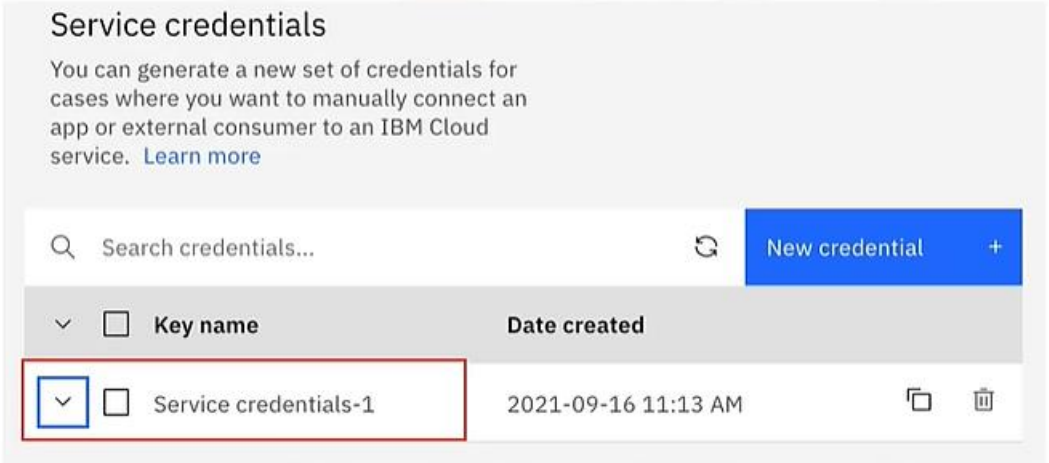
1. Data source: Db2



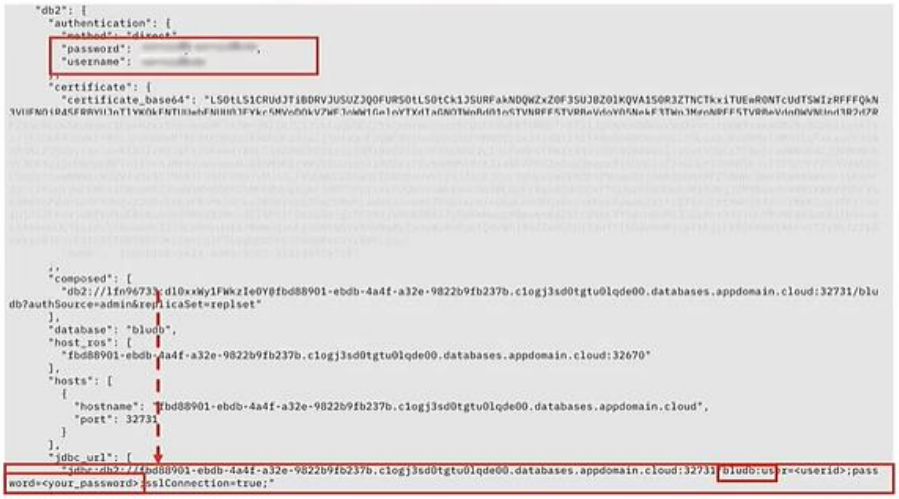
1. Data source: Db2



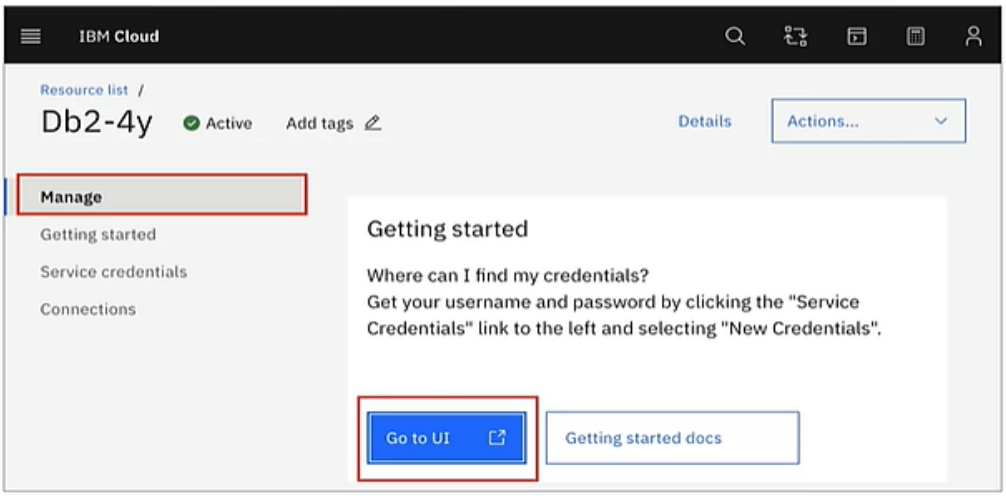
1. Data source: Db2



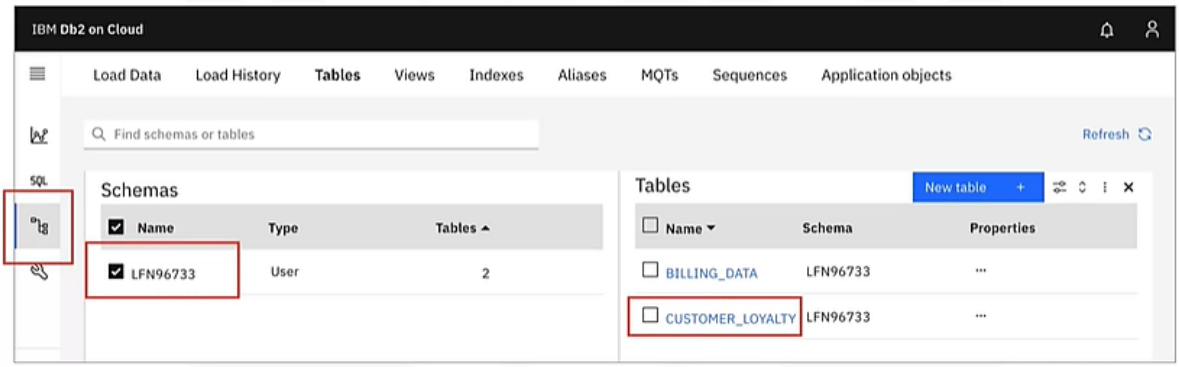
1. Data source: credentials



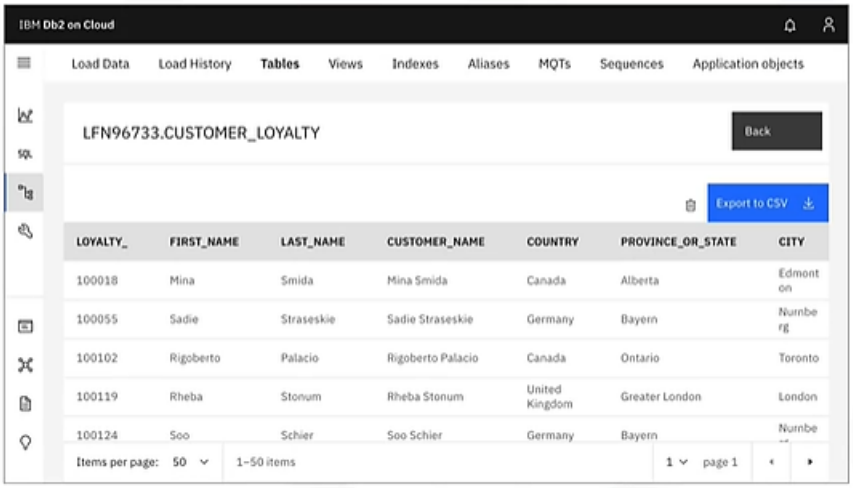
1. Data source: launch Db2 UI



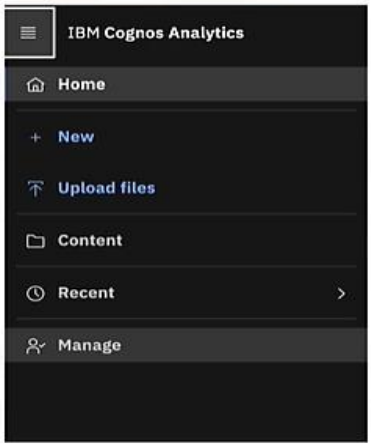
1. Data source: schema and table



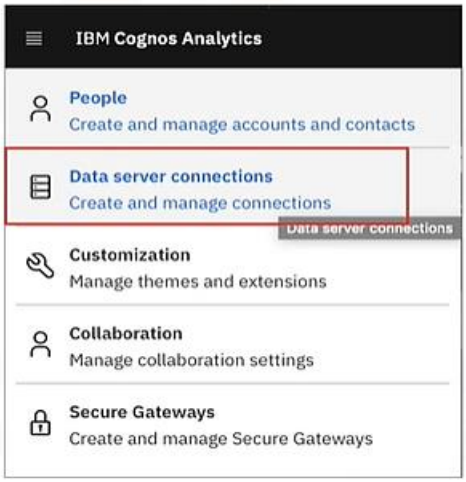
1. Data source: preview table data



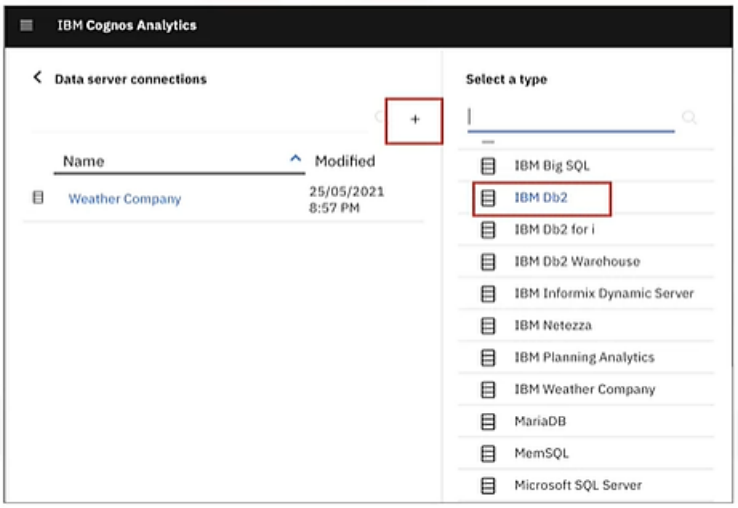
2. Connection: manage



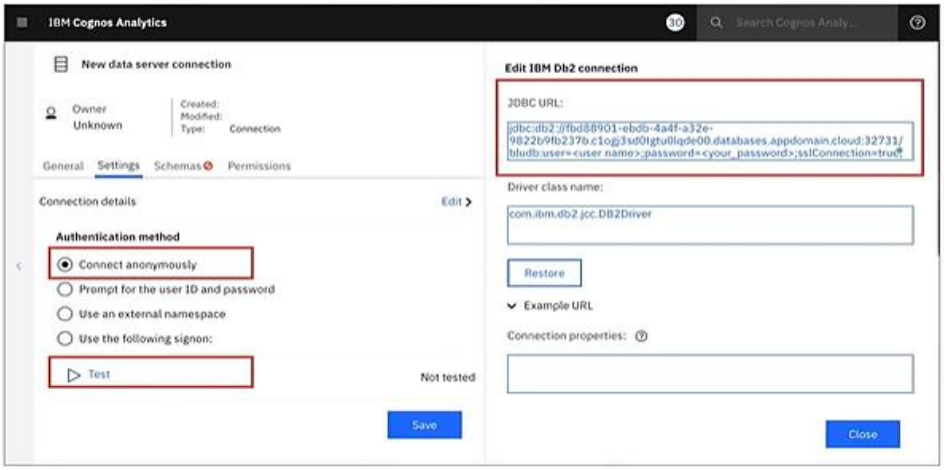
2. Connection: create and manage



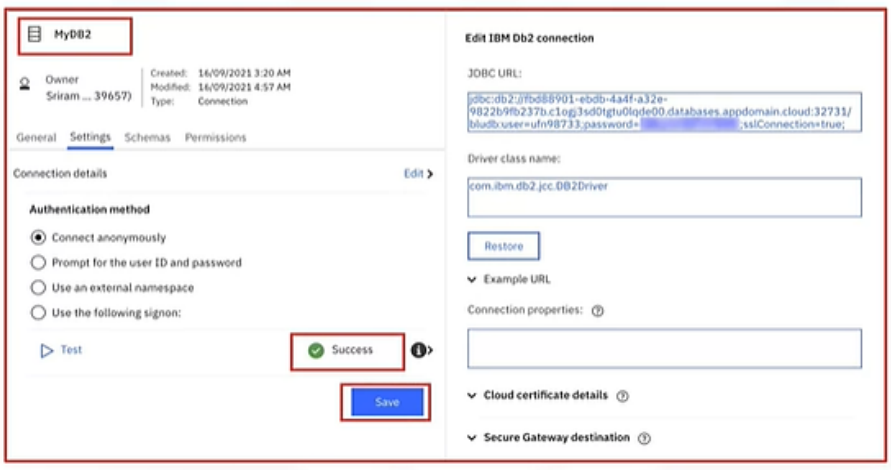
2. Connection: IBM Db2



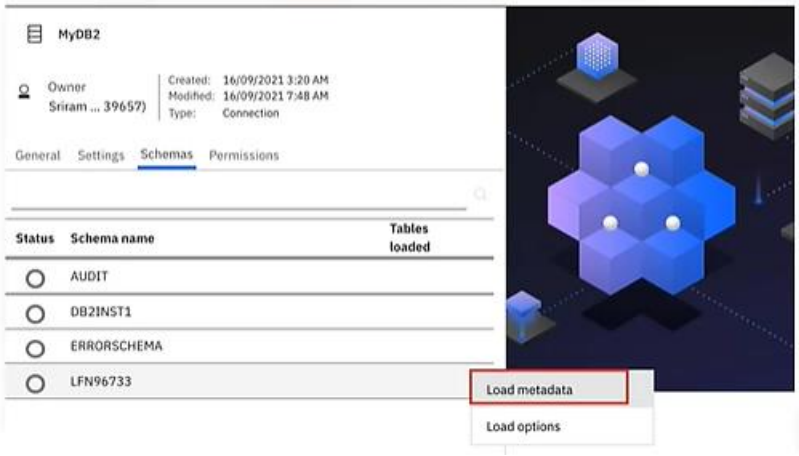
2. Connection: credentials & test



2. Connection: name and save

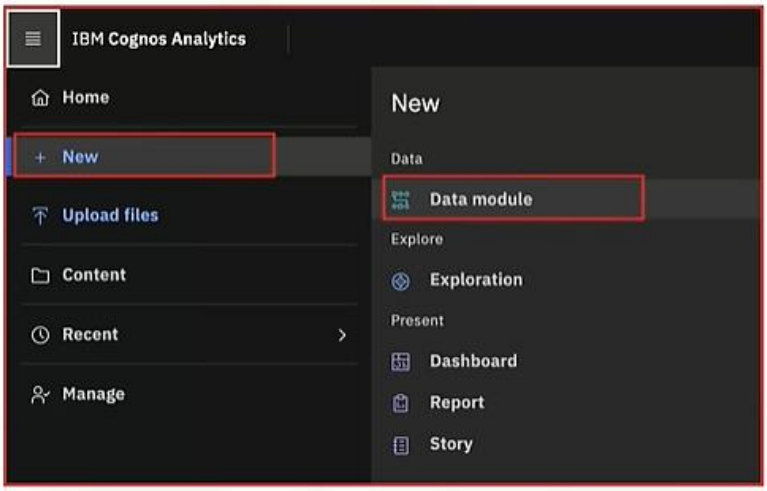


2. Connection: load schema

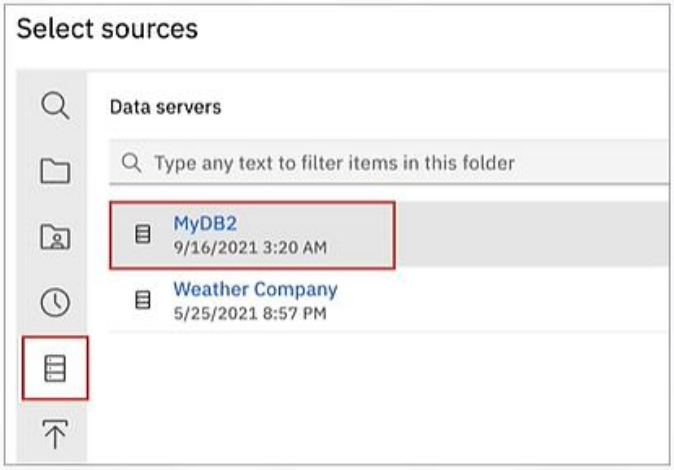


2. Connection: load schema

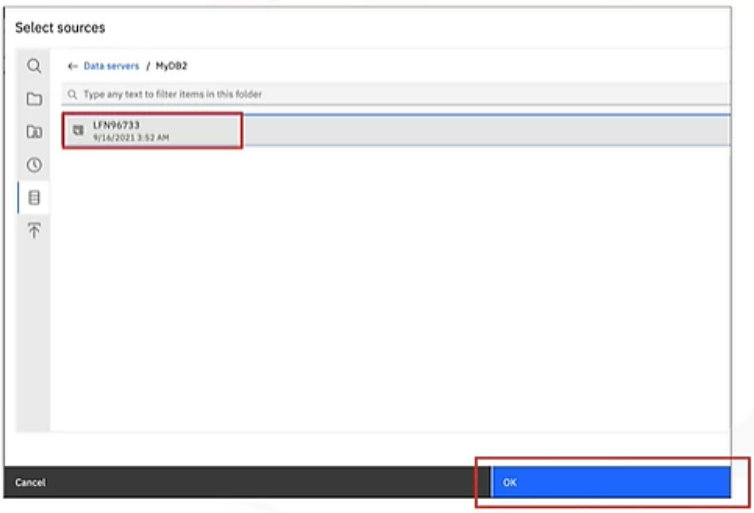
3. Data module: new



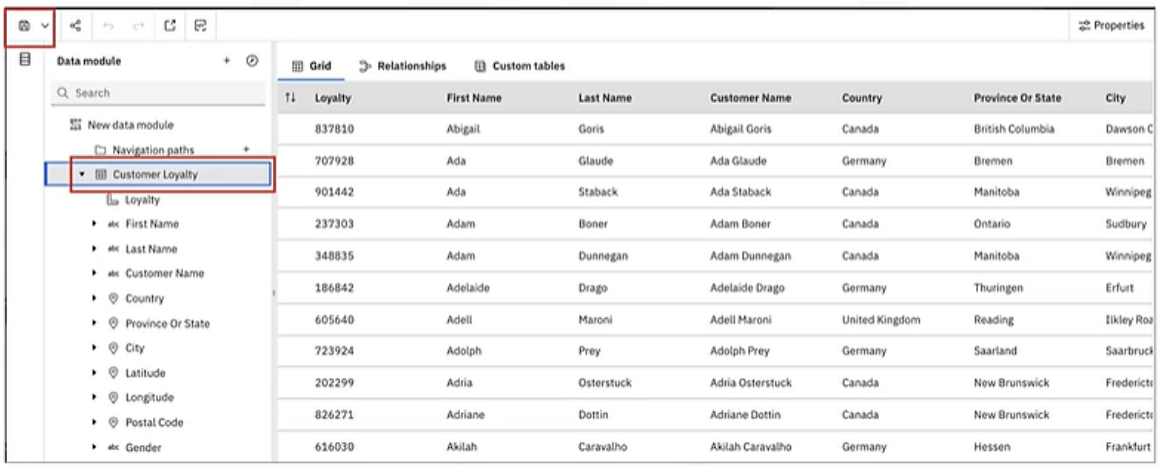
3. Data module: Db2 connection



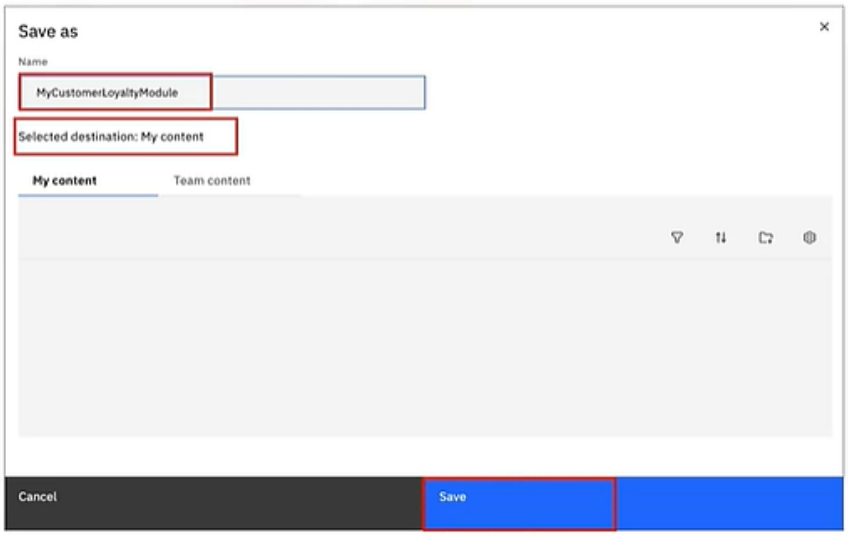
3. Data module: schema



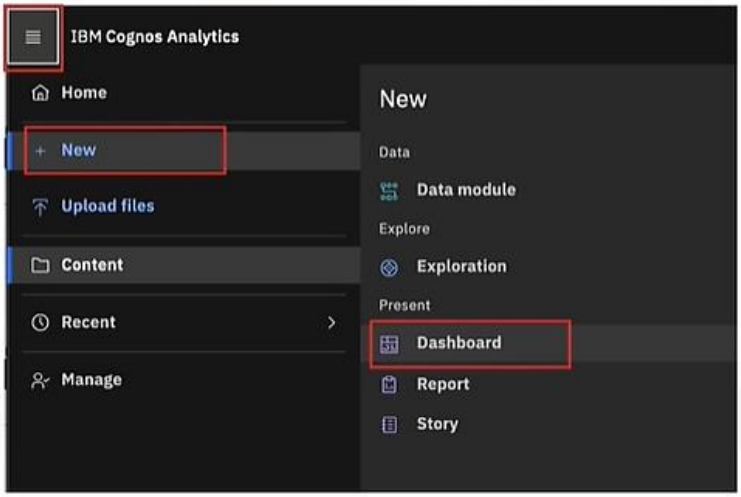
3. Data module: preview table



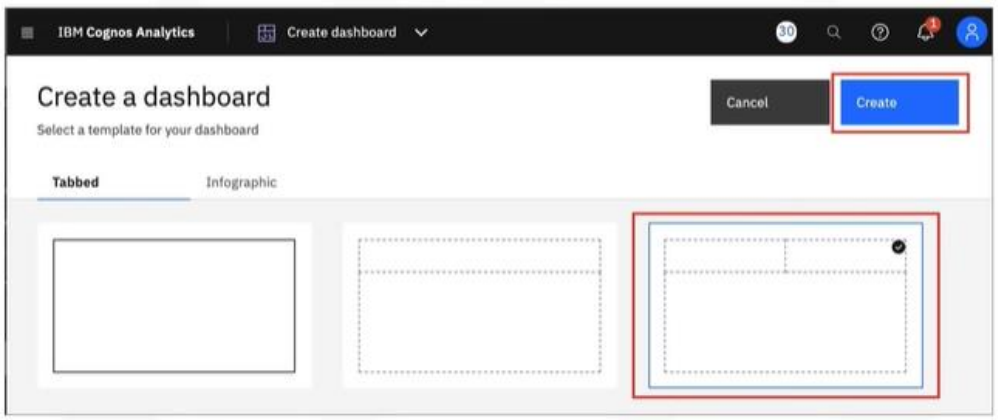
3. Data module: name and save



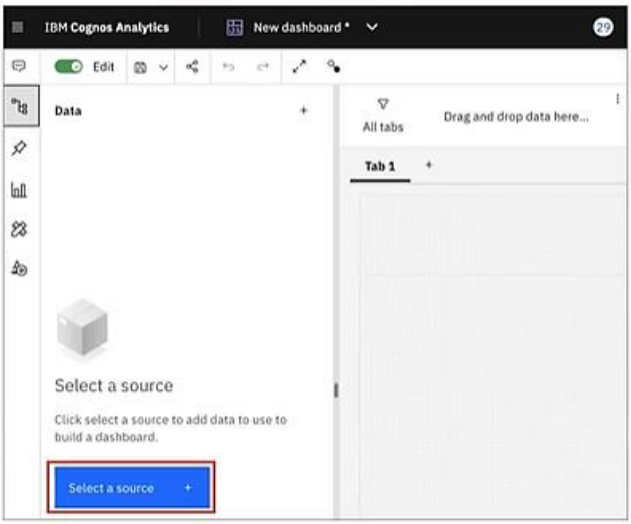
4. Dashboard: new



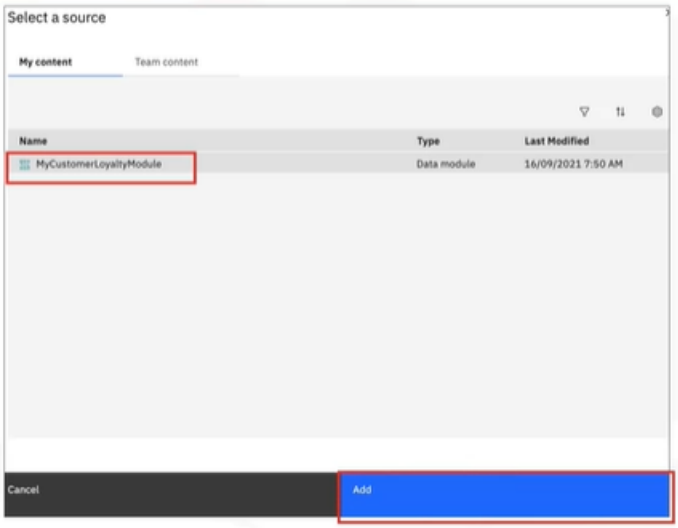
4. Dashboard: template



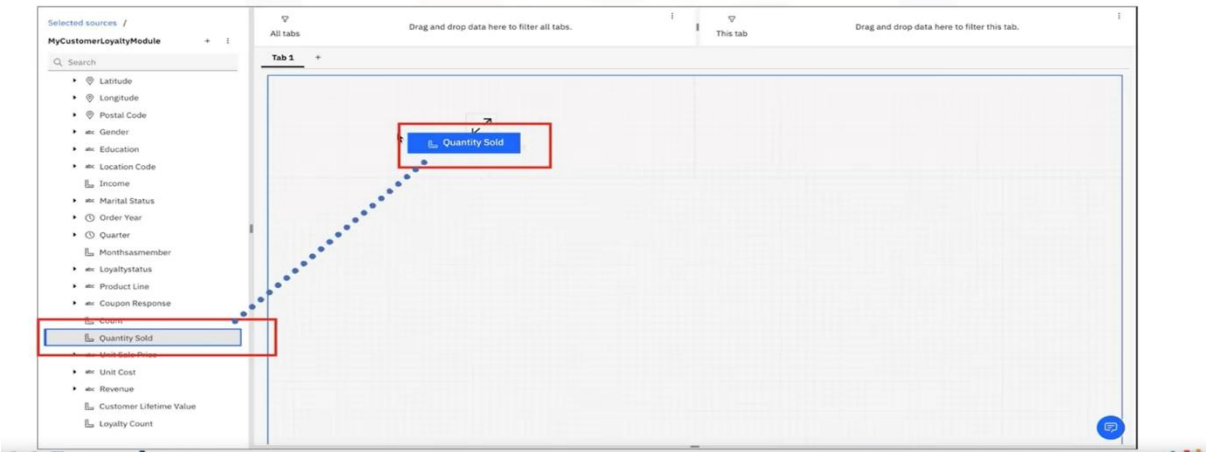
4: Dashboard: add source



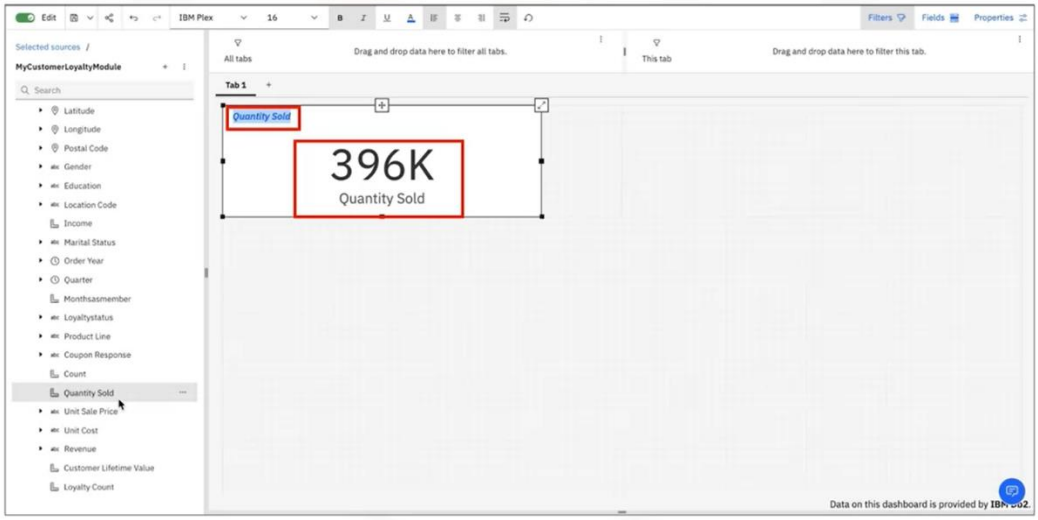
4. Dashboard: select source



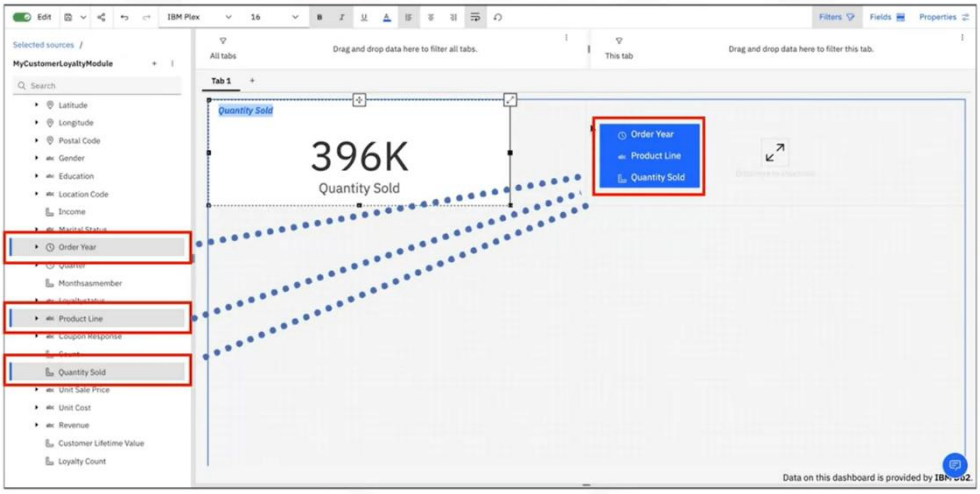
5. Visualizations: add data column



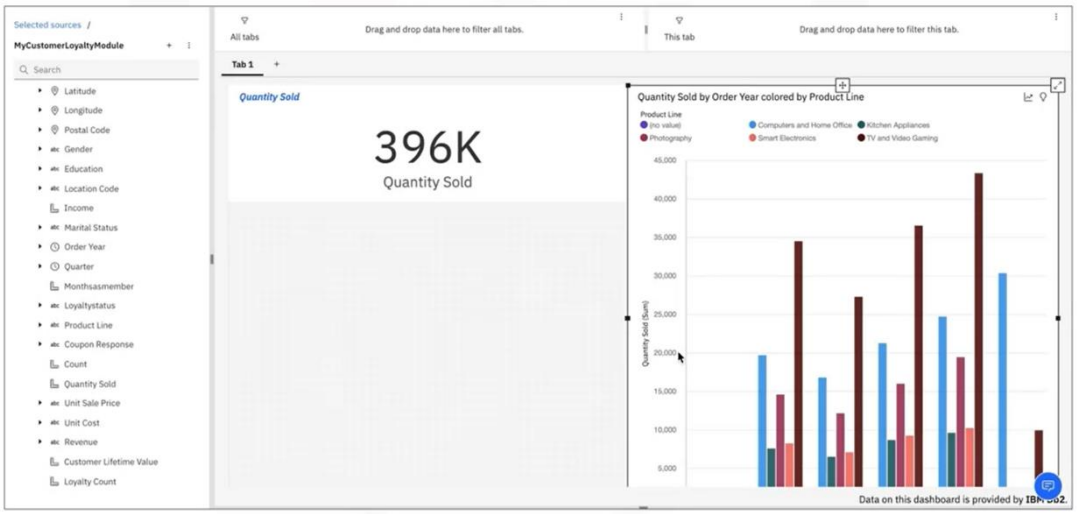
5. Visualizations: format data



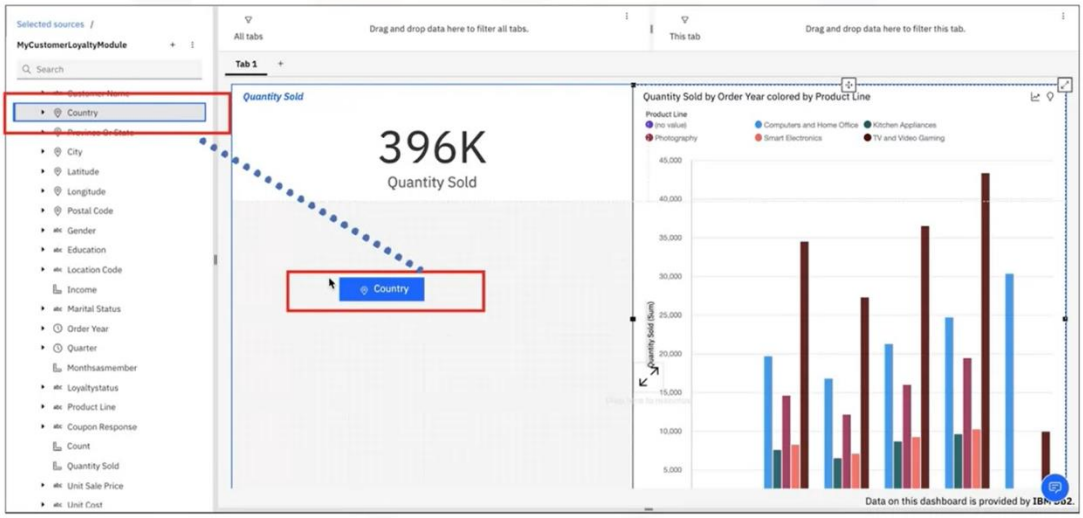
5. Visualizations: multiple columns



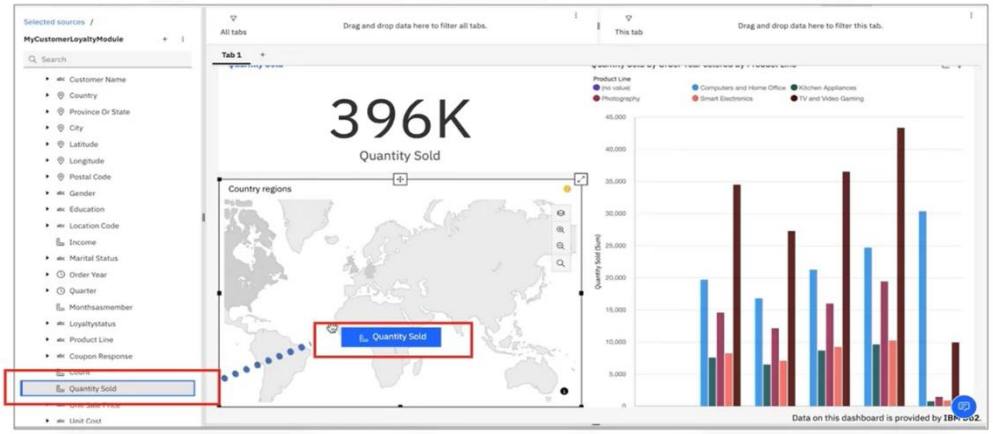
5. Visualizations: bar plot



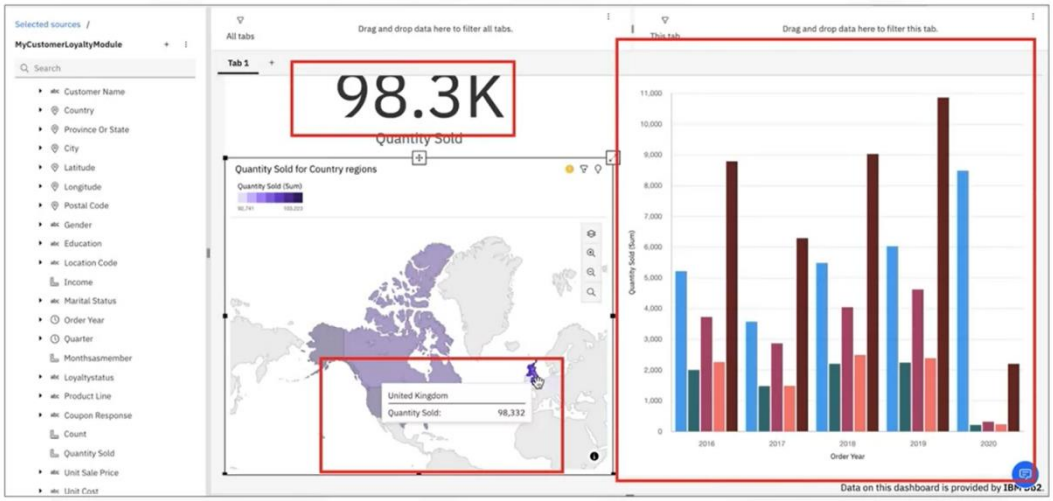
5. Visualizations: country



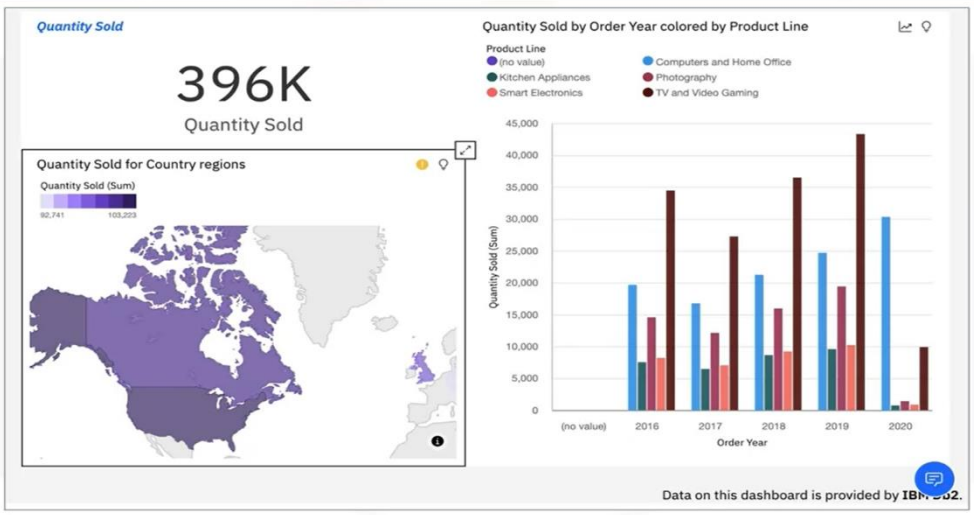
5. Visualizations: quantity sold



5. Visualizations: interactive



5. Visualizations: results



Summary: -

- You can create dashboards and reports in Cognos Analytics using various data sources, including flat files, databases, and data warehouses.
- Accessing a database in Cognos requires a data server connection and a data module.
- A data server connection establishes a connection to the database and a data module is a container, or link, to specific tables and database objects.
- You can add a data module to a dashboard just like any other source and use its fields and columns to create visualizations.
- Cognos Analytics offers a variety of templates and visualization options to create interactive dashboards using data from data warehouses.

Summary & Highlights: -

Analytics is the building of data models to make better decisions, and business intelligence is a technology that enables data preparation, mining, and visualization. You learned that accessing a database in Cognos requires a data server connection and a data module. You also learned how to add a data module to a dashboard and use its fields and columns to create visualizations. And you gained practical hands-on experience with IBM Cognos Analytics to create dashboards and visualizations.