Portfolio Final: Data Warehouse Star Schema / PostgreSQL

Didem B. Aykurt

Colorado State University Global

MIS541; Data Warehousing in Enterprise Environments

Dr.Peter Salemi

May 13, 2023

Table of Contents

Data Warehouse Star Schema, Table Creation, and Data Loading

BUSINESS PROCESS

The business process I will use for the data warehouse is a retail store's inventory management system. The system tracks product information, sales, and customer orders.

The business questions I am looking to answer are:

1. What is the total number of sales for each product?

2. What are the total sales for each product by the customer?

3. What is the most popular product in each store?

4. What is the average price of products sold in each store?

5. What is the total number of orders placed by each customer?

Why use a star scheme: The star schema answers business questions by providing insights into the data from the various dimension tables. For example, to answer the first question, the Product Table, Customer Table, and Store Table can be used to get the total sales for each product. The Sales Table can be used to get the total sales for each product by the customer, and the Store Table can be used to identify the most popular product in each store. Similarly, the Sales Table and Store Table can be used to get the average price of products sold in each store, and the Customer Table and Sales Table can be used to get the total number of orders placed by each customer.

IDENTIFY THE DIMENSIONS

Product Table

-Product ID

-Product Name

-Product Price

-Category ID

-Supplier ID

-Unit Price

-Unit_in_Stock

-Store ID

Customer Table

-Customer ID

-Customer Name

-Customer Address

-City

-Region

-Postal Code

-Country

-Customer Phone

Store Table

-Store ID

-Store Name

-Store Address

-City

-Region

-Postal Code

-Country

-Store Phone

Category Table:

-Category ID

-Category Name

-Description

-Picture

Supplier Table:

-Company Name

-Contact Name

-Contact Address

-City

-Region

-Postal Code

-Country

-Contact Phone

Fact Table:

Sales Table

-Sales ID

-Product ID

-Customer ID

-Store ID

-Sale Price

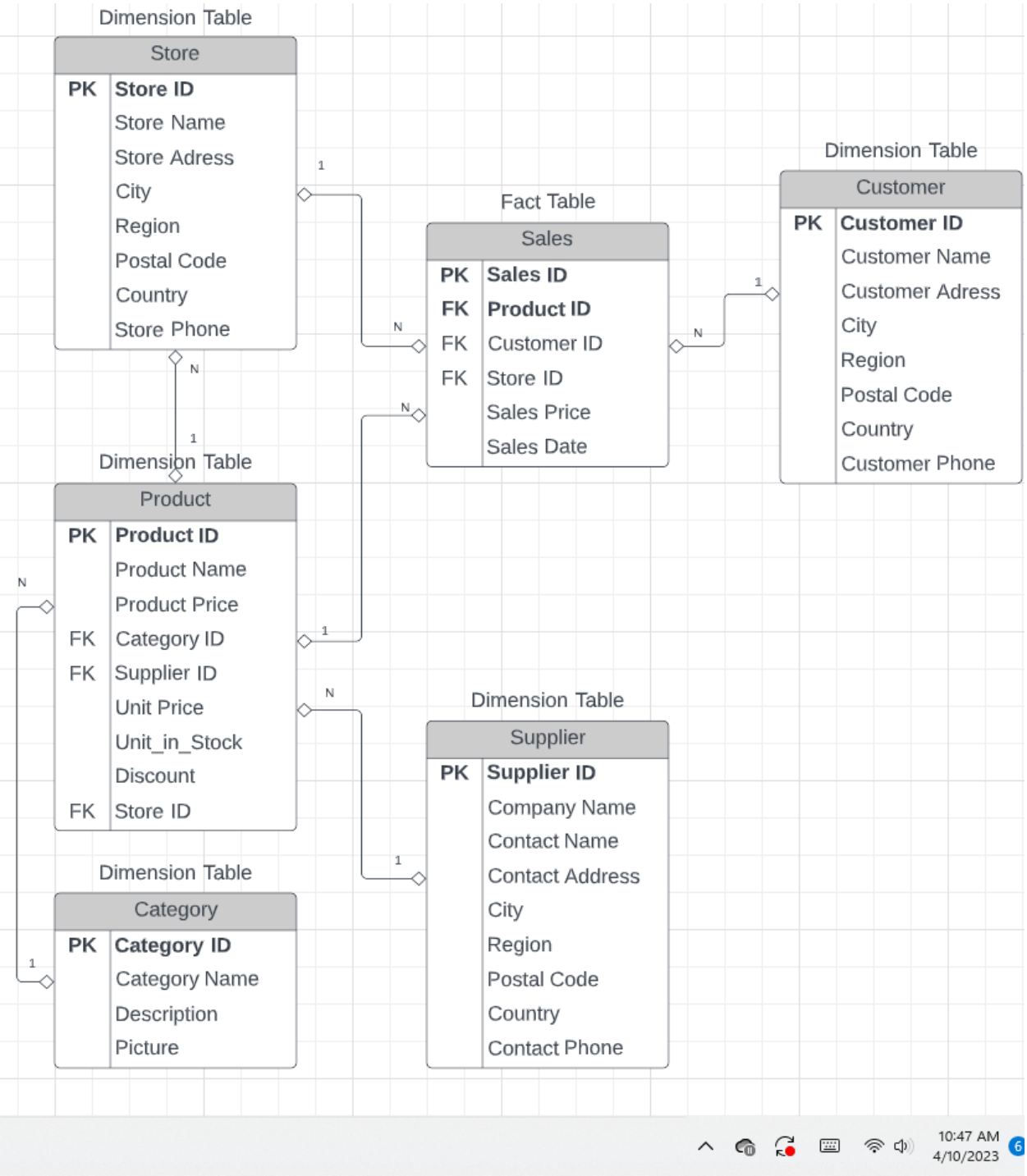
-Sale Date

DECLARE THE GRAIN

The grain of the fact table is the sale. This means that every entry in the fact table represents a single sale. This allows us to easily track the total number of sales for each product and customer and the total sales for each product by the customer.

The star schema is an ideal data warehouse design for this business process and business questions. It simplifies the data by organizing it into dimension tables and fact tables. The dimension tables contain the descriptive attributes of the data, while the fact table contains the numeric data. This makes it easy to query the data and answer business questions. Additionally, the star schema is easy to maintain and update, as data changes must only be made to one table. The diagram of dimensions and fact tables in the dimensional model is shown below. It shows the Product, Customer, Store, Category, Supplier, and Sales Table. The variables for each Table are also shown in the diagram.

Figure 1: Diagram of the dimensions and fact tables in the dimension model.



The **relationships between the fact and dimension tables** are as follows:

- The Category and Supplier Tables are related to the Product Table via the Category ID and Supplier ID

- The Product Table is related to the Sales Table via the Product ID

-The Customer Table is related to the Sales Table via the Customer ID

-The Store Table is related to the Sales Table via the Store ID

The cardinality of the relationships is one-to-many since each product, customer, category, supplier, and store can be associated with multiple sales.

DIMENSION TABLES

1. **Products Dimension Table**: This table will contain information about each product, including product name, description, category, and price.

2. **Customers Dimension Table**: This table will contain information about each customer, such as name, address, and contact information.

3. **Stores Dimension Table**: This table will contain information about each store, such as store name, address, and contact information.

4. **Category Dimension Table:** This table will contain the product's category information, including category name, description, and photo.

5. **Supplier Dimension Table:** This table will contain the product's supplier information, like contact name, address, phone, and company name.

FACT TABLE

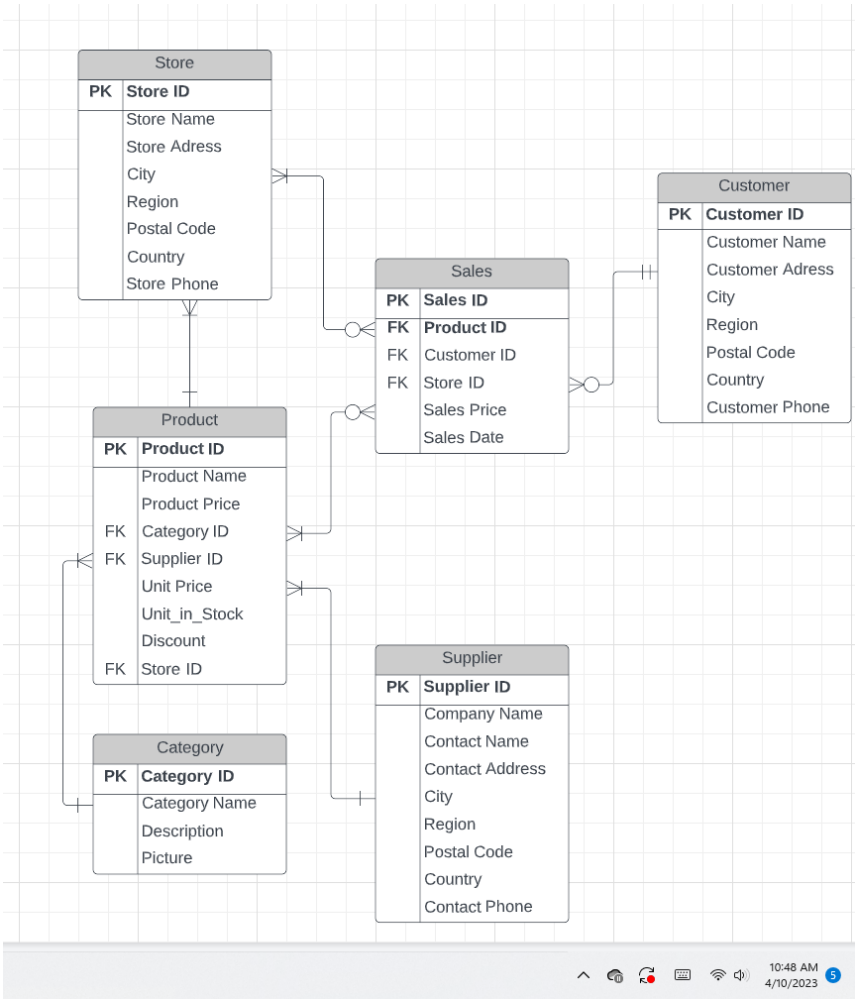
The fact table will contain information about each sale, including the product sold, the customer who purchased it, the store it was sold at, the date and time of the sale, and the total sale price.

GRAIN OF THE FACT TABLE

The fact table's grain is the granularity level at which the data is stored (Kimball & Ross, 2013). For example, if the fact table contains sales data, the grain would be the individual sales. This means that each record in the fact table represents a single sale. The grain of the fact table will be at the individual transaction level. This means that each row in the fact table will contain the details of each sale, including the product sold, the customer who purchased it, and the date and time of the sale. Every entry in the fact table represents a single sale. This allows us to easily track the total number of sales for each product and customer and the total sales for each product by the customer.

The **Entity-Relationship Diagram of the source OLTP database system** is shown below. It shows the relationships between the various tables in the design, including the Product Table, Customer Table, Store Table, Category Table, Supplier Table, and Sales Table.

Figure 2: Entity-Relationship Diagram.



BUSINESS PROCESS

The business process I will use for the data warehouse is a retail store's inventory management system. It can track product information, sales information, and reports on customer orders. No changes were required to the business process, business questions, or fact table grain from Module 3. Refer to the original document (Portfolio Milestone 1) for details.

UPDATED STAR SCHEMA FACT AND DIMENSION TABLES

The screenshot of the fact and dimension tables is shown in Figure 2. It shows the Product, Customer, Store, and Sales Table. The variables for each table are also shown in the diagram. The "Product," "Store," and "Customer" dimension tables are directly related to the fact table "Sales" through their foreign keys. In addition, they provide additional descriptive information about the sales data in the fact table, such as product details, store location, and customer demographics. The "Category" and "Supplier" tables are also dimension tables that are referenced by the "Product" table, but they are not directly related to the fact table "Sales." The following is a screenshot of the fact and dimension tables:

SQL COMMANDS TABLE CREATE

The Kimball method of data warehouse design was used to create a star schema for the retail store's inventory management system (Kimball & Ross, 2013).

The following is a listing of the SQL commands used to create the tables, define the variables, and apply the constraints to the fact and dimension tables:

```
CREATE DATABASE RETAILS;

USE RETAILS;

Drop table  if exists Customer


Drop table  if exists Store
CREATE TABLE Store (

   StoreID INT PRIMARY KEY, -- Primary key for Store Table

   StoreName VARCHAR(50), -- Name of the store

   StoreAddress VARCHAR(50), -- Address of the store
```

```
        City  char(10),     -- City of the Store
        Region varchar(20),  -- Region of the Store
        Postal_Code varchar(20), -- Postal Code of the Store
        Country  CHAR(30),      -- Country of the Store
    StorePhone VARCHAR(20) -- Phone number of the store
);



Drop table if exists Supplier
CREATE TABLE Supplier (
        Supplier_ID int Primary key,  -- Primary key for the Supplier Table
        Comapany_Name Varchar(40),  --Company Name of the Supplier
        Contact_Name  Varchar(40),  -- Contact Name of the Supplier
        Contact_Address Varchar(50), -- Address of the Supplier
        City  char(10),     -- City of the Supplier
        Region varchar(20),  -- Region of the Supplier
        Postal_Code varchar(20), -- Postal Code of the Supplier
        Country  CHAR(30),      -- Country of the Supplier
       Contact_Phone Varchar(15) );--Contact phone of the Supplier


Drop table if exists Category;
Create Table Category ( Category_ID int primary key, -- Primary key for Category Table
```

```
    Category_Name Varchar(30), -- Category Name of the Category
    Description Varchar(50),  --Description for Category
    Picture BYTEA );  -- Picture For the Category


Drop table if exists Product
CREATE TABLE Product (
   ProductID INT PRIMARY KEY, -- Primary key for Product Table
   ProductName VARCHAR(50),   -- Name of the product
   ProductPrice DECIMAL(10,2), -- Price of the product
   Category_ID int,   -- Foreign key for Category Table
   Supplier_ID int,   -- Foreign key for the Supplier table
   Unit_Price  decimal(8,2), -- Unit price of the product
   Unit_in_Stocks int,      --Unit in Stocks of the Product
   Discount Decimal(5,2),   -- Discount on the this Product
   StoreID INT, -- Foreign key for Store Table
   CONSTRAINT fk_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID), -- Constraint to
ensure that the StoreID references the Store Table
   CONSTRAINT fk_Category_ID FOREIGN KEY (Category_ID) REFERENCES
Category(Category_ID), -- Constraint to ensure that the Category_ID references the Category
Table
```

```
    CONSTRAINT fk_Supplier_ID FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID) –
Constraint to ensure that the Supplier_ID references the Supplier Table
);



Drop table  if exists Sales
CREATE TABLE Sales (
    SaleID INT PRIMARY KEY, -- Primary key for Sales Table
    ProductID INT, -- Foreign key for Product Table
    CustomerID INT, -- Foreign key for Customer Table
    StoreID INT, -- Foreign key for Store Table
    SalePrice DECIMAL(10,2), -- Price of the sale
    SaleDate DATE, -- Date of the sale
    CONSTRAINT fk_ProductID FOREIGN KEY (ProductID) REFERENCES Product(ProductID), --
Constraint to ensure that the ProductID references the Product Table
    CONSTRAINT fk_CustomerID FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID), -- Constraint to ensure that the CustomerID references the Customer
Table
    CONSTRAINT fk2_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID) – Constraint to
ensure that the StoreID references the Store Table
);
```

```
Drop table  if exists Sales;


CREATE TABLE Sales (
   SaleID INT PRIMARY KEY, -- Primary key for Sales Table
   ProductID INT, -- Foreign key for Product Table
   CustomerID INT, -- Foreign key for Customer Table
   StoreID INT, -- Foreign key for Store Table
   SalePrice DECIMAL(10,2), -- Price of the sale
   SaleDate DATE, -- Date of the sale
   CONSTRAINT fk_ProductID FOREIGN KEY (ProductID) REFERENCES Product(ProductID), --
Constraint to ensure that the ProductID references the Product Table
   CONSTRAINT fk_CustomerID FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID), -- Constraint to ensure that the CustomerID references the Customer
Table
   CONSTRAINT fk2_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID) – Constraint to
ensure that the StoreID references the Store Table
);
```

Conclusion

The star schema consists of four tables, the Product Table, Customer Table, Store Table, and Sales Table. The variables for each table were defined, and the constraints were applied to the fact and dimension tables. The SQL commands were then used to create the tables, describe

create the variables and their attributes, and apply the constraints. The star schema is now ready to answer business questions and provide insights into the data.

No changes were required to the business process, business questions, or fact table grain from Module 3. Refer to the original document (Portfolio Milestone 2) for details.

SQL command for row count:

SELECT COUNT(*) as total_no_of_row FROM RETAILS.Customer;

This SQL query retrieves the total number of rows in the Sales table from the RETAILS database. The result set will contain a single row with a column named “total_no_of_row” that indicates the count of rows in the Sales table.

Screenshots of the Row count of each table:

Figure 3: Count of Customer Table’s row

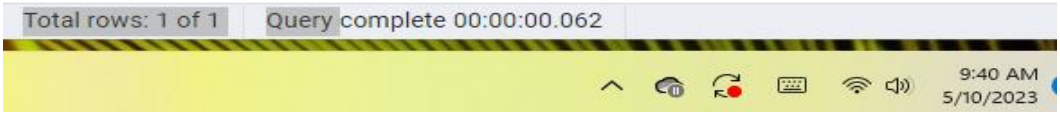


Figure 4: Count of Store Table’s row.

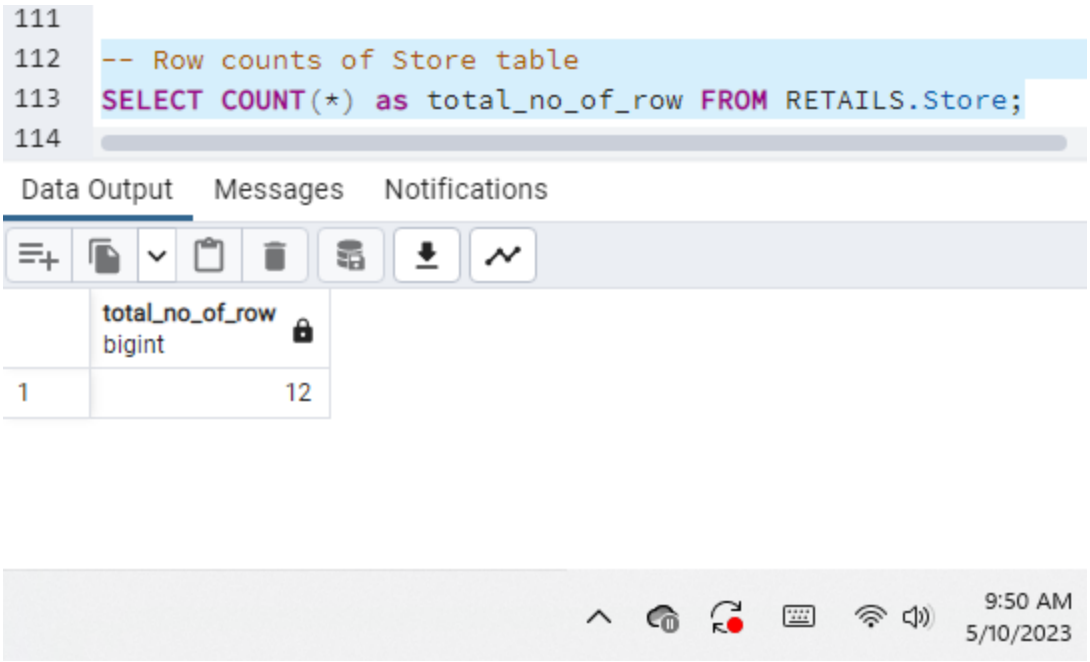


Figure 5: Count of Supplier Table's row.

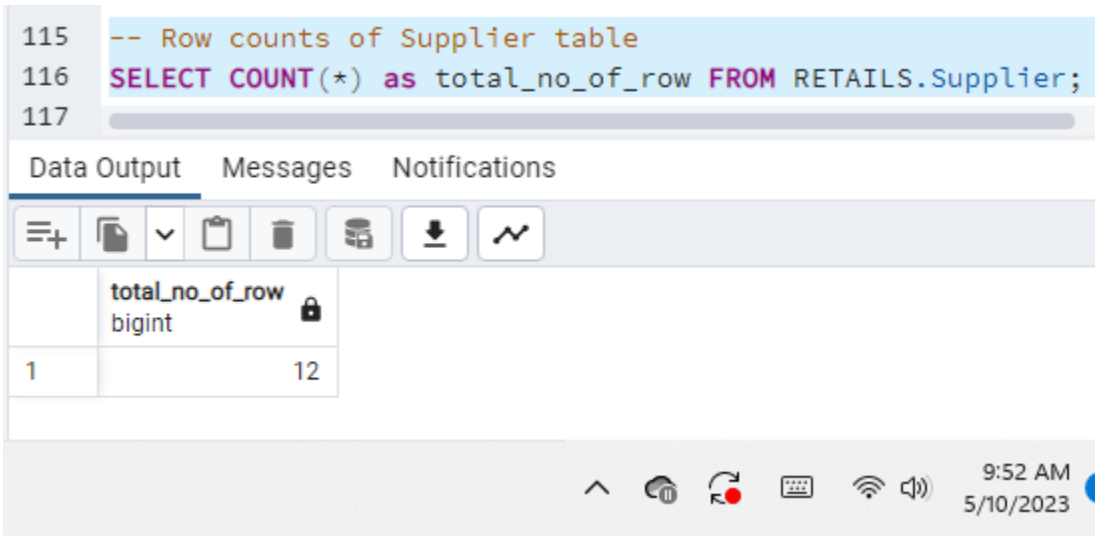


Figure 6: Count of Category Table's row.

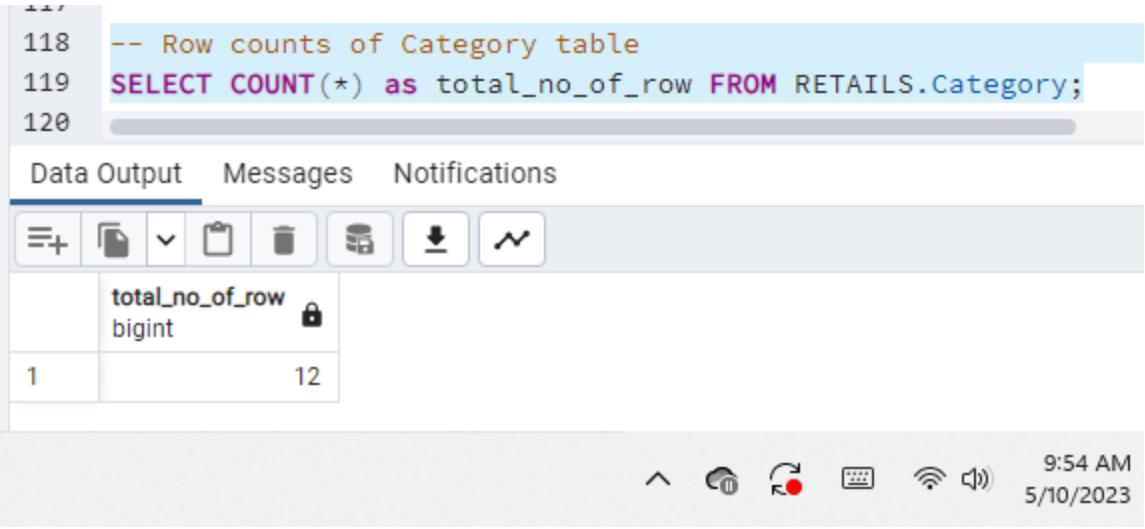


Figure 7: Count of Product Table's row.

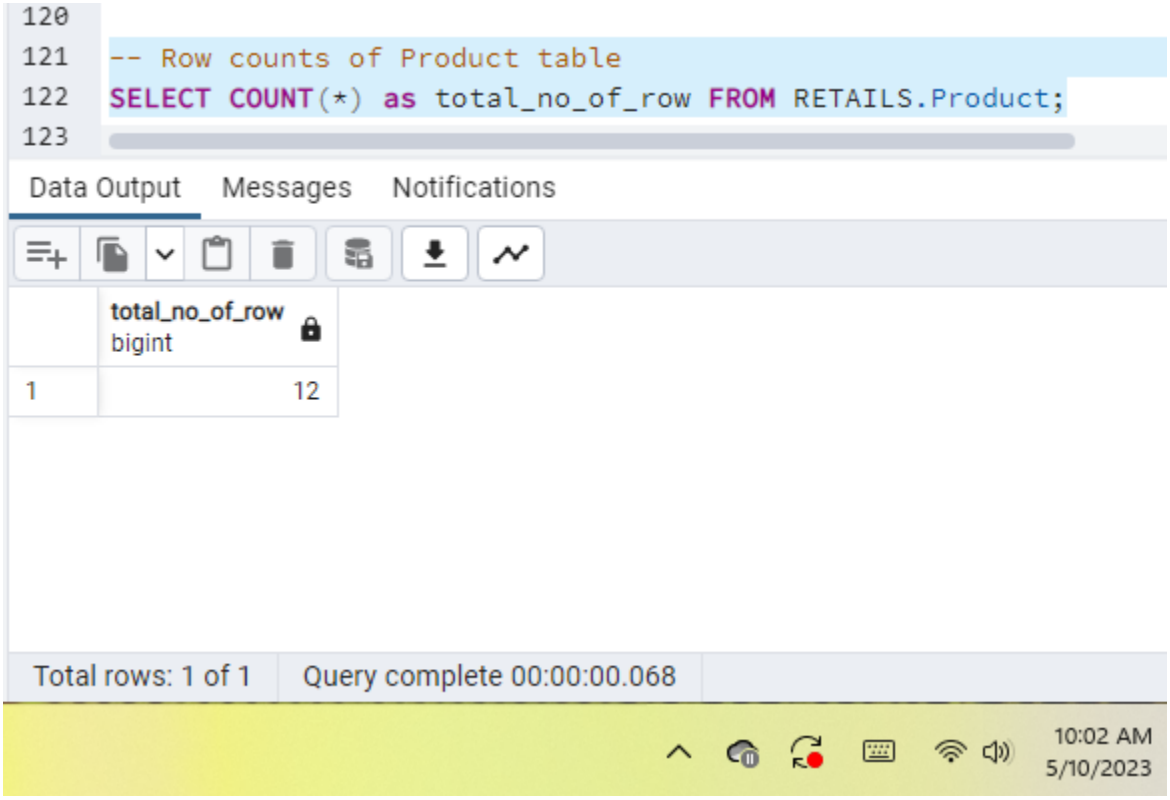
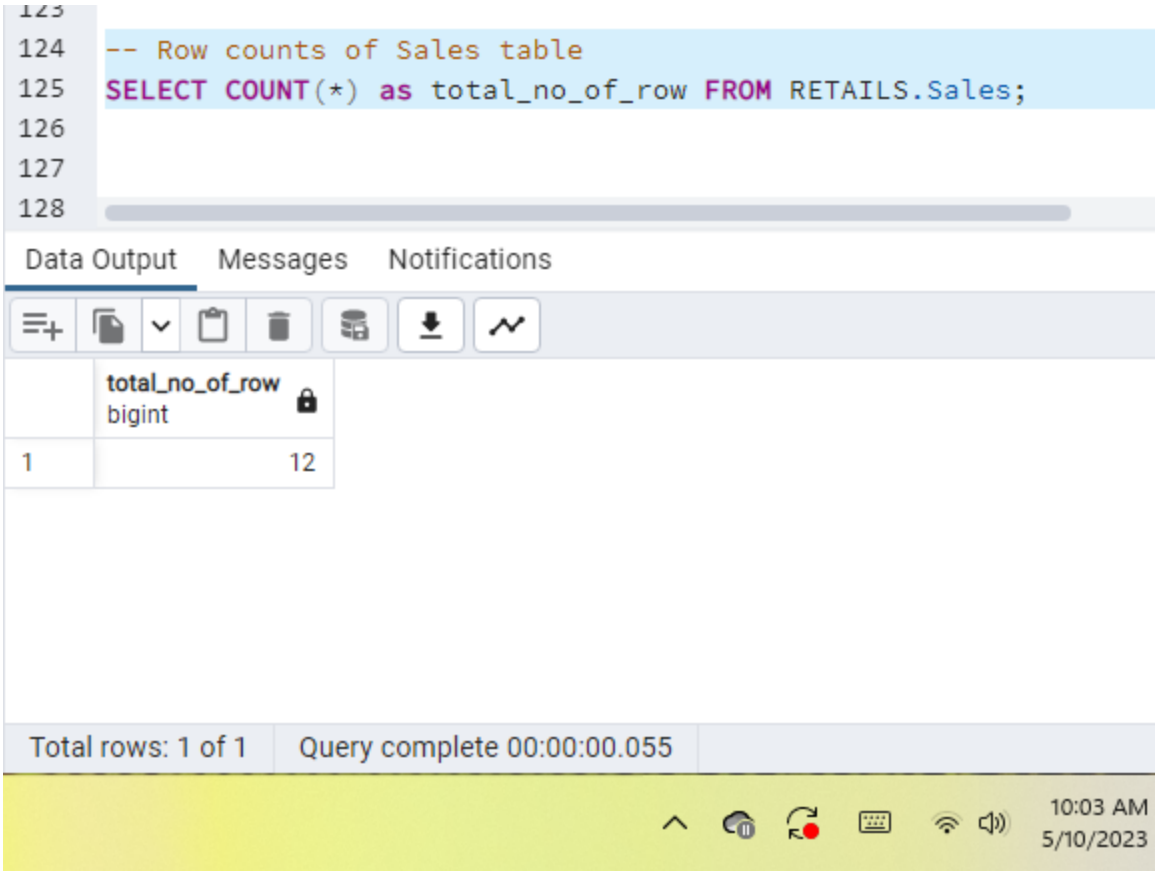


Figure 8: Count of Sales Table's row.



SQL command for first 10 rows:

```
SELECT * FROM RETAILS.Customer FETCH NEXT 10 ONLY;
```

This SQL query retrieves the first ten rows from the Customer table in the RETAILS database. The result set will contain ten rows of data, each representing a customer record in the Customer table. The "FETCH NEXT 10" keyword specifies that only the first ten rows should be returned, and the "*" wildcard character indicates that all columns should be included in the result set.

Screenshots of the first 10 rows of each table:

Figure 9: First ten rows of Customer Table.

```
129
130 -- First ten rows of customer table
131 SELECT * FROM RETAILS.Customer
132 FETCH NEXT 10 ROWS ONLY;
133
```

Data Output | Messages | Notifications

	customerid [PK] integer	customername character varying (50)	customeraddress character varying (50)	city character (10)	region character varying (20)	postal_code character varying (20)	country character (30)	customerphone character varying (20)
1	1	John Doe	123 Main St	New York	NY	10001	[null]	555-1234
2	2	Jane Smith	456 Elm St	LoAngels	CA	90001	[null]	555-5678
3	3	Bob Johnson	789 Oak St	Chicago	IL	60601	[null]	555-9012
4	4	Sarah Lee	321 Pine St	Sanisco	CA	94101	[null]	555-3456
5	5	Mike Chen	654 Maple St	Houston	TX	77001	[null]	555-7890
6	6	Emily Wong	987 Cedar St	Miami	FL	33101	[null]	555-2345
7	7	David Kim	741 Birch St	Atlanta	GA	30301	[null]	555-6789
8	8	Karen Lee	852 Spruce St	Seattle	WA	98101	[null]	555-0123
9	9	Tom Jones	369 Walnut St	Boston	MA	02101	[null]	555-4567
10	10	Mary Brown	258 Ash St	Dallas	TX	75201	[null]	555-8901

Total rows: 10 of 10 | Query complete 00:00:00.052

10:10 AM 5/10/2023

Figure 10: First ten rows of Store Table.

```
134 -- First ten rows of Store table
135 SELECT * FROM RETAILS.Store
136 FETCH NEXT 10 ROWS ONLY;
137
```

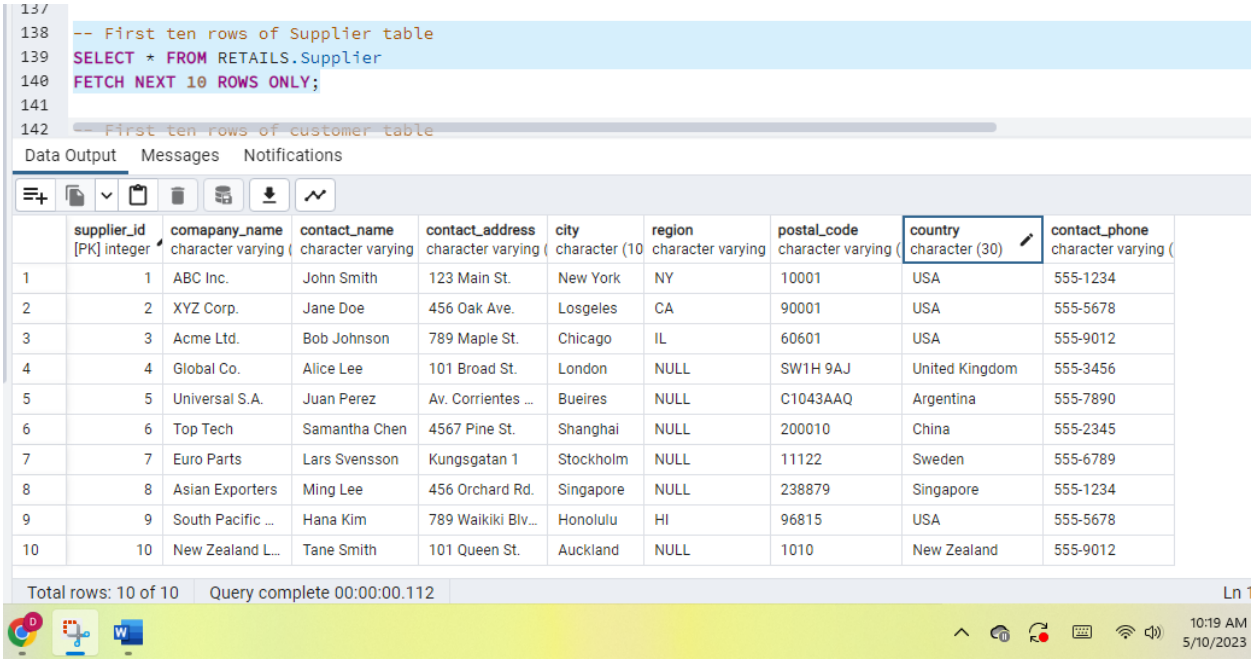
Data Output | Messages | Notifications

	storeid [PK] integer	storename character varying (50)	storeaddress character varying (50)	city character (10)	region character varying (20)	postal_code character varying (20)	country character (30)	storephone character varying
1	1	ABC Supermarket	123 Main St	New York	NY	10001	[null]	123-456-7890
2	2	FreshMart	456 Elm St	Losngeles	CA	90001	[null]	111-222-3333
3	3	GreenGrocer	789 Oak St	Sanncisco	CA	94102	[null]	444-555-6666
4	4	HealthyHarvest	1010 Pine St	Boston	MA	02108	[null]	777-888-9999
5	5	FoodBazaar	1111 Broadway	Chicago	IL	60601	[null]	555-444-3333
6	6	FruitEmporium	2222 Maple St	Miami	FL	33101	[null]	123-456-7890
7	7	FarmersMarket	3333 Vine St	Dallas	TX	75201	[null]	111-222-3333
8	8	NaturalFoods	4444 Chestnut St	Houston	TX	77001	[null]	444-555-6666
9	9	OrganicCorner	5555 Cherry St	Seattle	WA	98101	[null]	777-888-9999
10	10	SuperSaver	6666 Walnut St	Phelphia	PA	19103	[null]	555-444-3333

Total rows: 10 of 10 | Query complete 00:00:00.093 | Ln 136, C

10:17 AM 5/10/2023

Figure 11: First ten rows of Supplier Table.



```
137
138  -- First ten rows of Supplier table
139  SELECT * FROM RETAILS.Supplier
140  FETCH NEXT 10 ROWS ONLY;
141
142  -- First ten rows of customer table
```

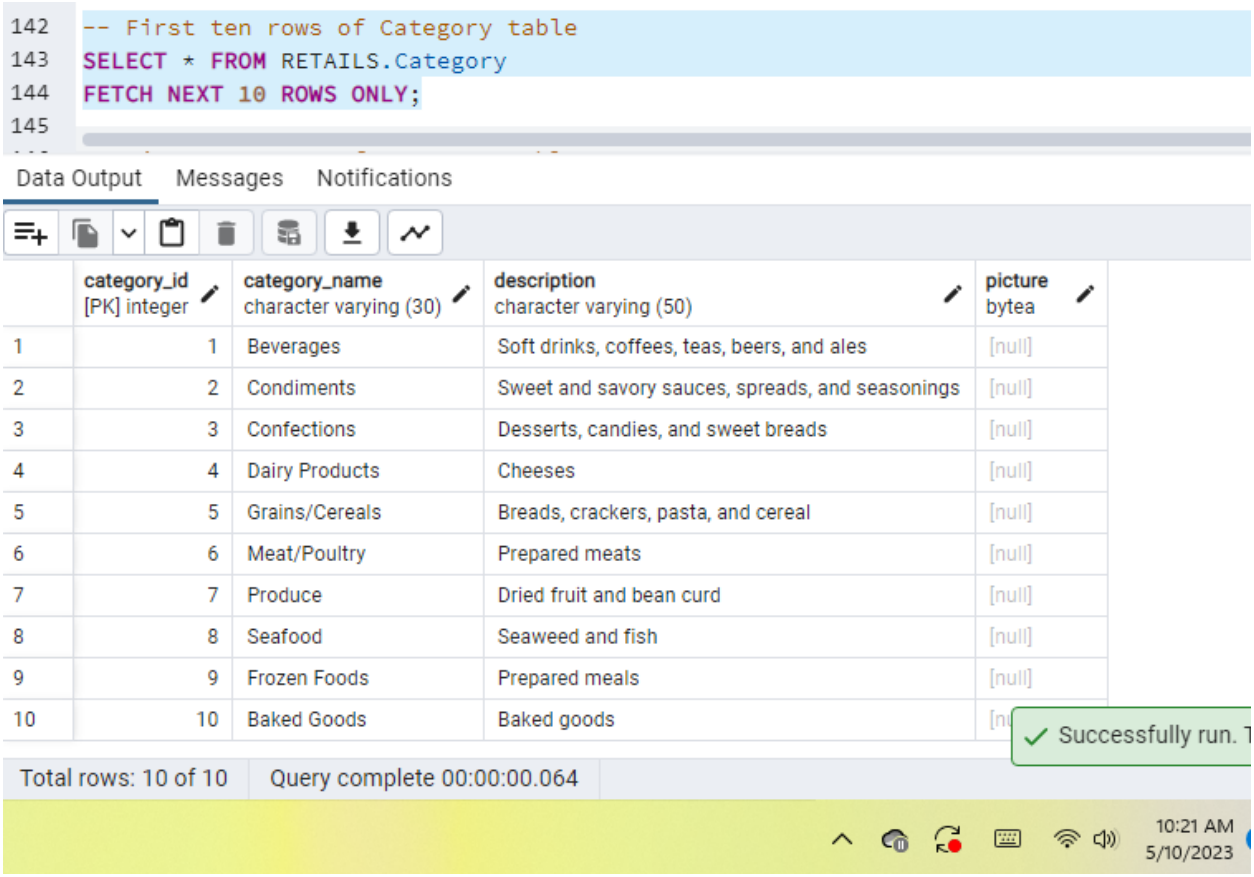
Data Output | Messages | Notifications

	supplier_id [PK] integer	comapany_name character varying (	contact_name character varying	contact_address character varying (	city character (10	region character varying	postal_code character varying (	country character (30)	contact_phone character varying (
1	1	ABC Inc.	John Smith	123 Main St.	New York	NY	10001	USA	555-1234
2	2	XYZ Corp.	Jane Doe	456 Oak Ave.	Losgeles	CA	90001	USA	555-5678
3	3	Acme Ltd.	Bob Johnson	789 Maple St.	Chicago	IL	60601	USA	555-9012
4	4	Global Co.	Alice Lee	101 Broad St.	London	NULL	SW1H 9AJ	United Kingdom	555-3456
5	5	Universal S.A.	Juan Perez	Av. Corrientes ...	Bueires	NULL	C1043AAQ	Argentina	555-7890
6	6	Top Tech	Samantha Chen	4567 Pine St.	Shanghai	NULL	200010	China	555-2345
7	7	Euro Parts	Lars Svensson	Kungsgatan 1	Stockholm	NULL	11122	Sweden	555-6789
8	8	Asian Exporters	Ming Lee	456 Orchard Rd.	Singapore	NULL	238879	Singapore	555-1234
9	9	South Pacific ...	Hana Kim	789 Waikiki Blv...	Honolulu	HI	96815	USA	555-5678
10	10	New Zealand L...	Tane Smith	101 Queen St.	Auckland	NULL	1010	New Zealand	555-9012

Total rows: 10 of 10 | Query complete 00:00:00.112 | Ln 1

10:19 AM 5/10/2023

Figure 12: First ten rows of Category Table.



```
142  -- First ten rows of Category table
143  SELECT * FROM RETAILS.Category
144  FETCH NEXT 10 ROWS ONLY;
145
```

Data Output | Messages | Notifications

	category_id [PK] integer	category_name character varying (30)	description character varying (50)	picture bytea
1	1	Beverages	Soft drinks, coffees, teas, beers, and ales	[null]
2	2	Condiments	Sweet and savory sauces, spreads, and seasonings	[null]
3	3	Confections	Desserts, candies, and sweet breads	[null]
4	4	Dairy Products	Cheeses	[null]
5	5	Grains/Cereals	Breads, crackers, pasta, and cereal	[null]
6	6	Meat/Poultry	Prepared meats	[null]
7	7	Produce	Dried fruit and bean curd	[null]
8	8	Seafood	Seaweed and fish	[null]
9	9	Frozen Foods	Prepared meals	[null]
10	10	Baked Goods	Baked goods	[n

Successfully run. T

Total rows: 10 of 10 | Query complete 00:00:00.064

10:21 AM 5/10/2023

Figure 13: First ten rows of Product Table.

```
146  -- First ten rows of Product table
147  SELECT * FROM RETAILS.Product
148  FETCH NEXT 10 ROWS ONLY;
149
```

Data Output | Messages | Notifications

	productid [PK] integer	productname character varying (50)	productprice numeric (10,2)	category_id integer	supplier_id integer	unit_price numeric (8,2)	unit_in_stocks integer	discount numeric (5,2)	storeid integer
1	1	Product A	25.00	1	1	20.00	50	0.10	[null]
2	2	Product B	20.00	2	2	18.00	25	0.00	[null]
3	3	Product C	35.00	1	1	28.00	100	0.15	[null]
4	4	Product D	50.00	3	3	45.00	75	0.05	[null]
5	5	Product E	10.00	2	2	9.00	10	0.00	[null]
6	6	Product F	45.00	3	3	40.00	50	0.10	[null]
7	7	Product G	15.00	1	1	12.00	150	0.20	[null]
8	8	Product H	30.00	2	2	27.00	30	0.00	[null]
9	9	Product I	40.00	3	3	36.00	40	0.05	[null]
10	10	Product J	12.00	1	1	10.00	80	0.00	[null]

Total rows: 10 of 10 | Query complete 00:00:00.073

10:24 AM 5/10/2023

Figure 13: First ten rows of Sales Table.

```
150  -- First ten rows of Sales table
151  SELECT * FROM RETAILS.Sales
152  FETCH NEXT 10 ROWS ONLY;
```

Data Output | Messages | Notifications

	saleid [PK] integer	productid integer	customerid integer	storeid integer	saleprice numeric (10,2)	saledate date
1	1	1	1	1	25.99	2023-04-01
2	2	2	2	2	19.99	2023-04-02
3	3	3	3	3	35.99	2023-04-03
4	4	4	4	4	28.99	2023-04-04
5	5	5	5	5	15.99	2023-04-05
6	6	6	6	6	42.99	2023-04-06
7	7	7	7	7	31.99	2023-04-07
8	8	8	8	8	49.99	2023-04-08
9	9	9	9	9	22.99	2023-04-09
10	10	10	10	10	18.99	2023-04-10

10:25 AM 5/10/2023

Recommendations/Conclusions

It is essential to clearly understand the business requirements and data sources before building a data warehouse. This ensures that the data warehouse is built to meet the organization's specific needs and is designed to accommodate potential future changes. Several lessons were learned in completing the Portfolio Project: data warehouse architecture that describes an organization's data action of collection and storage frame for sorting, cleaning, and organizing the functional data warehouse to enable actionable data for effective decision-making. The proposed solution includes a star schema of dimension and fact tables, ETL (Extract, Transform, Load) processing from multiple source production systems, data cleansing activities, a loading plan for data refresh, and the two Milestones act table is a table in a star schema of a data warehouse that stores quantitative data for analysis. It comprises a set of foreign keys from the various dimension tables that are joined together. The table also includes a set of measures or values used to analyze the data. The variables in the fact table consist of the foreign keys from the dimension tables and the measure values (Kimball & Ross, 2013). For example, a Sales Fact Table might have the following variables pro Product (foreign key from the Product dimension table); Store (foreign key from the Store dimension table). The attributes in the fact table may include foreign keys, measures, or other information, such as a fact table key, which is a unique identifier for each record. Star schema shows the relationship between a fact table, its dimensions, and the attributes included in each table. Proper planning and documentation are essential. It is important to have a well-defined ETL process that includes data cleaning, transformation, and loading. This process should be thoroughly documented to ensure consistency and accuracy in the data warehouse. Testing and validation of the data warehouse

are crucial. It is important to thoroughly test the data warehouse to ensure the data is accurate, complete, and consistent. This can help identify errors or issues that must be addressed before the data warehouse is deployed.

Advise for Data Warehouse

The organization building a data warehouse system should clearly define the business requirements and data sources before building a data warehouse—star schema to answer business questions by providing insights into the data from the various dimension tables. For example, to answer the first question, the Product Table, Customer Table, and Store Table can be used to get the total sales for each product. At the same time, the fact table will store the quantitative data such as sales, revenue, and other key performance indicators (KPIs). The fact table will be connected to the dimension tables through foreign key relationships, allowing easy navigation and data analysis. All the tables must develop a well-defined ETL process that includes data cleaning, transformation, and loading and thoroughly document it. Perform thorough testing and validation of the data warehouse to ensure the data is accurate, complete, and consistent. Finally, ensure the data warehouse is designed to accommodate potential business requirements and data source changes. I used BI solution step by step, such as exploring the business needs, creating star schema of dimension and fact tables, ETL process, Data cleansing activity, loading plan, and strategy.

References

Kimball, R., & Ross, M. (2013). Chapter 1 Data Warehousing, Business Intelligence, and Dimensional Modeling Primer. In *The Data Warehouse toolkit the Complete Guide to Dimensional Modeling* (3rd ed., pp. 7–16). essay, John Wiley & Sons, Inc.

APPENDIX

- The data warehouse schema diagram provides the star schema of dimension and fact tables by sample data.
- ETL process provides PostgreSQL queries to show sample data's extraction, transformation, and loading activities.
- The data cleansing rules and validation checks ensure data quality and accuracy with data.
- The loading from the OLTP database to the OLAP RETAIL data warehouse draws the data partitioning, indexing, and optimization strategies to improve query performance with data.

The first step is to create OLTP database name DATA.

```
CREATE DATABASE DATA;


Drop table  if exists Customer


Drop table  if exists Store
CREATE TABLE Store (

   StoreID INT PRIMARY KEY, -- Primary key for Store Table

   StoreName VARCHAR(50), -- Name of the store

   StoreAddress VARCHAR(50), -- Address of the store

        City  char(10),     -- City of the Store

        Region varchar(20),  -- Region of the Store

        Postal_Code varchar(20), -- Postal Code of the Store
```

```
        Country  CHAR(30),      -- Country of the Store
   StorePhone VARCHAR(20) -- Phone number of the store
);



Drop table if exists Supplier
CREATE TABLE Supplier (
        Supplier_ID int Primary key,  -- Primary key for the Supplier Table
        Comapany_Name Varchar(40),  --Company Name of the Supplier
        Contact_Name  Varchar(40),  -- Contact Name of the Supplier
        Contact_Address Varchar(50), -- Address of the Supplier
        City  char(10),     -- City of the Supplier
        Region varchar(20),  -- Region of the Supplier
        Postal_Code varchar(20), -- Postal Code of the Supplier
        Country  CHAR(30),      -- Country of the Supplier
       Contact_Phone Varchar(15) );--Contact phone of the Supplier


Drop table if exists Category;
Create Table Category ( Category_ID int primary key, -- Primary key for Category Table
                Category_Name Varchar(30), -- Category Name of the Category
                Description Varchar(50),  --Description for Category
                Picture BYTEA );  -- Picture For the Category
```

```
Drop table if exists Product

CREATE TABLE Product (

   ProductID INT PRIMARY KEY, -- Primary key for Product Table

   ProductName VARCHAR(50),   -- Name of the product

   ProductPrice DECIMAL(10,2), -- Price of the product

   Category_ID int,   -- Foreign key for Category Table

   Supplier_ID int,   -- Foreign key for the Supplier table

   Unit_Price  decimal(8,2), -- Unit price of the product

   Unit_in_Stocks int,      --Unit in Stocks of the Product

   Discount Decimal(5,2),   -- Discount on the this Product

   StoreID INT, -- Foreign key for Store Table

   CONSTRAINT fk_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID), -- Constraint to
ensure that the StoreID references the Store Table

   CONSTRAINT fk_Category_ID FOREIGN KEY (Category_ID) REFERENCES
Category(Category_ID), -- Constraint to ensure that the Category_ID references the Category
Table

   CONSTRAINT fk_Supplier_ID FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID) –
Constraint to ensure that the Supplier_ID references the Supplier Table

);
```

```
Drop table  if exists Sales

CREATE TABLE Sales (

   SaleID INT PRIMARY KEY, -- Primary key for Sales Table

   ProductID INT, -- Foreign key for Product Table

   CustomerID INT, -- Foreign key for Customer Table

   StoreID INT, -- Foreign key for Store Table

   SalePrice DECIMAL(10,2), -- Price of the sale

   SaleDate DATE, -- Date of the sale

   CONSTRAINT fk_ProductID FOREIGN KEY (ProductID) REFERENCES Product(ProductID), --
Constraint to ensure that the ProductID references the Product Table

   CONSTRAINT fk_CustomerID FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID), -- Constraint to ensure that the CustomerID references the Customer
Table

   CONSTRAINT fk2_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID) – Constraint to
ensure that the StoreID references the Store Table

);



Drop table  if exists Sales;


CREATE TABLE Sales (
```

```
    SaleID INT PRIMARY KEY, -- Primary key for Sales Table
    ProductID INT, -- Foreign key for Product Table
    CustomerID INT, -- Foreign key for Customer Table
    StoreID INT, -- Foreign key for Store Table
    SalePrice DECIMAL(10,2), -- Price of the sale
    SaleDate DATE, -- Date of the sale
    CONSTRAINT fk_ProductID FOREIGN KEY (ProductID) REFERENCES Product(ProductID), --
Constraint to ensure that the ProductID references the Product Table
    CONSTRAINT fk_CustomerID FOREIGN KEY (CustomerID) REFERENCES
Customer(CustomerID), -- Constraint to ensure that the CustomerID references the Customer
Table
    CONSTRAINT fk2_StoreID FOREIGN KEY (StoreID) REFERENCES Store(StoreID) – Constraint to
ensure that the StoreID references the Store Table
);

--  data for Customer

    INSERT INTO Customer (CustomerID, CustomerName, CustomerAddress, City, Region,
    Postal_Code, Country, CustomerPhone)
    VALUES
    (1, 'John Doe', '123 Main St', 'New York', 'NY', '10001', 'USA', '555-1234'),
    (2, 'Jane Smith', '456 Elm St', 'LoAngels', 'CA', '90001', 'USA', '555-5678'),
    (3, 'Bob Johnson', '789 Oak St', 'Chicago', 'IL', '60601', 'USA', '555-9012'),
    (4, 'Sarah Lee', '321 Pine St', 'Sanisco', 'CA', '94101', 'USA', '555-3456'),
    (5, 'Mike Chen', '654 Maple St', 'Houston', 'TX', '77001', 'USA', '555-7890'),
    (6, 'Emily Wong', '987 Cedar St', 'Miami', 'FL', '33101', 'USA', '555-2345'),
    (7, 'David Kim', '741 Birch St', 'Atlanta', 'GA', '30301', 'USA', '555-6789'),
    (8, 'Karen Lee', '852 Spruce St', 'Seattle', 'WA', '98101', 'USA', '555-0123'),
    (9, 'Tom Jones', '369 Walnut St', 'Boston', 'MA', '02101', 'USA', '555-4567'),
    (10, 'Mary Brown', '258 Ash St', 'Dallas', 'TX', '75201', 'USA', '555-8901'),
```

```
(11, 'Chris Evans', '147 Pine St', 'Phoenix', 'AZ', '85001', 'USA', '555-2345'),
(12, 'Laura Lee', '753 Oak St', 'Washing', 'DC', '20001', 'USA', '555-6789');

--  data for Store
INSERT INTO Store (StoreID, StoreName, StoreAddress, City, Region, Postal_Code, Country,
StorePhone)
VALUES
(1, 'ABC Supermarket', '123 Main St', 'New York', 'NY', '10001', 'USA', '123-456-7890'),
(2, 'FreshMart', '456 Elm St', 'Losngeles', 'CA', '90001', 'USA', '111-222-3333'),
(3, 'GreenGrocer', '789 Oak St', 'Sanncisco', 'CA', '94102', 'USA', '444-555-6666'),
(4, 'HealthyHarvest', '1010 Pine St', 'Boston', 'MA', '02108', 'USA', '777-888-9999'),
(5, 'FoodBazaar', '1111 Broadway', 'Chicago', 'IL', '60601', 'USA', '555-444-3333'),
(6, 'FruitEmporium', '2222 Maple St', 'Miami', 'FL', '33101', 'USA', '123-456-7890'),
(7, 'FarmersMarket', '3333 Vine St', 'Dallas', 'TX', '75201', 'USA', '111-222-3333'),
(8, 'NaturalFoods', '4444 Chestnut St', 'Houston', 'TX', '77001', 'USA', '444-555-6666'),
(9, 'OrganicCorner', '5555 Cherry St', 'Seattle', 'WA', '98101', 'USA', '777-888-9999'),
(10, 'SuperSaver', '6666 Walnut St', 'Phelphia', 'PA', '19103', 'USA', '555-444-3333'),
(11, 'WholeFoods', '7777 Pine St', 'Austin', 'TX', '78701', 'USA', '123-456-7890'),
(12, 'TraderJoes', '8888 Oak St', 'San Diego', 'CA', '92101', 'USA', '111-222-3333');


--  data for Supplier
INSERT INTO Supplier
(Supplier_ID, Comapany_Name, Contact_Name, Contact_Address, City, Region,
Postal_Code, Country, Contact_Phone)
VALUES
(1, 'ABC Inc.', 'John Smith', '123 Main St.', 'New York', 'NY', '10001', 'USA', '555-1234'),
(2, 'XYZ Corp.', 'Jane Doe', '456 Oak Ave.', 'Losgeles', 'CA', '90001', 'USA', '555-5678'),
(3, 'Acme Ltd.', 'Bob Johnson', '789 Maple St.', 'Chicago', 'IL', '60601', 'USA', '555-9012'),
(4, 'Global Co.', 'Alice Lee', '101 Broad St.', 'London', 'NULL', 'SW1H 9AJ', 'United Kingdom',
'555-3456'),
(5, 'Universal S.A.', 'Juan Perez', 'Av. Corrientes 1234', 'Bueires', 'NULL', 'C1043AAQ',
'Argentina', '555-7890'),
(6, 'Top Tech', 'Samantha Chen', '4567 Pine St.', 'Shanghai', 'NULL', '200010', 'China', '555-
2345'),
(7, 'Euro Parts', 'Lars Svensson', 'Kungsgatan 1', 'Stockholm', 'NULL', '11122', 'Sweden', '555-
6789'),
(8, 'Asian Exporters', 'Ming Lee', '456 Orchard Rd.', 'Singapore', 'NULL', '238879', 'Singapore',
'555-1234'),
(9, 'South Pacific Co.', 'Hana Kim', '789 Waikiki Blvd.', 'Honolulu', 'HI', '96815', 'USA', '555-
5678'),
(10, 'New Zealand Ltd.', 'Tane Smith', '101 Queen St.', 'Auckland', 'NULL', '1010', 'New
Zealand', '555-9012'),
```

```
(11, 'Island Traders', 'Leilani Wong', '456 Waikoloa Rd.', 'KailuKona', 'HI', '96740', 'USA', '555-3456'),
(12, 'Latin Export', 'Jose Martinez', 'Av. de los Incas 123', 'Lima', 'NULL', 'L18', 'Peru', '555-7890');


-- data for category

INSERT INTO Category (Category_ID, Category_Name, Description, Picture)
VALUES
(1, 'Beverages', 'Soft drinks, coffees, teas, beers, and ales', NULL),
(2, 'Condiments', 'Sweet and savory sauces, spreads, and seasonings', NULL),
(3, 'Confections', 'Desserts, candies, and sweet breads', NULL),
(4, 'Dairy Products', 'Cheeses', NULL),
(5, 'Grains/Cereals', 'Breads, crackers, pasta, and cereal', NULL),
(6, 'Meat/Poultry', 'Prepared meats', NULL),
(7, 'Produce', 'Dried fruit and bean curd', NULL),
(8, 'Seafood', 'Seaweed and fish', NULL),
(9, 'Frozen Foods', 'Prepared meals', NULL),
(10, 'Baked Goods', 'Baked goods', NULL),
(11, 'Sweets', 'Candies and chocolates', NULL),
(12, 'Snacks', 'Nuts and chips', NULL);

-- data for Product tables
INSERT INTO Product (ProductID, ProductName, ProductPrice, Category_ID, Supplier_ID,
Unit_Price, Unit_in_Stocks, Discount, StoreID)
VALUES
(1, 'Product A', 25.00, 1, 1, 20.00, 50, 0.10, 1),
(2, 'Product B', 20.00, 2, 2, 18.00, 25, 0.00, 2),
(3, 'Product C', 35.00, 1, 1, 28.00, 100, 0.15, 1),
(4, 'Product D', 50.00, 3, 3, 45.00, 75, 0.05, 3),
(5, 'Product E', 10.00, 2, 2, 9.00, 10, 0.00, 2),
(6, 'Product F', 45.00, 3, 3, 40.00, 50, 0.10, 3),
(7, 'Product G', 15.00, 1, 1, 12.00, 150, 0.20, 1),
(8, 'Product H', 30.00, 2, 2, 27.00, 30, 0.00, 2),
(9, 'Product I', 40.00, 3, 3, 36.00, 40, 0.05, 3),
(10, 'Product J', 12.00, 1, 1, 10.00, 80, 0.00, 1),
(11, 'Product K', 55.00, 2, 2, 50.00, 20, 0.10, 2),
(12, 'Product L', 28.00, 3, 3, 24.00, 60, 0.00, 3);


-- Data for Sales
INSERT INTO Sales (SaleID, ProductID, CustomerID, StoreID, SalePrice, SaleDate)
VALUES (1, 1, 1, 1, 25.99, '2023-04-01'),
```

```
(2, 2, 2, 2, 19.99, '2023-04-02'),
(3, 3, 3, 3, 35.99, '2023-04-03'),
(4, 4, 4, 4, 28.99, '2023-04-04'),
(5, 5, 5, 5, 15.99, '2023-04-05'),
(6, 6, 6, 6, 42.99, '2023-04-06'),
(7, 7, 7, 7, 31.99, '2023-04-07'),
(8, 8, 8, 8, 49.99, '2023-04-08'),
(9, 9, 9, 9, 22.99, '2023-04-09'),
(10, 10, 10, 10, 18.99, '2023-04-10'),
(11, 11, 11, 11, 39.99, '2023-04-11'),
(12, 12, 12, 12, 27.99, '2023-04-12');
```

Data query following populate the table and count and top ten list below.

```
-- Populate the customer table from the source database

INSERT INTO RETAILS.Customer (CustomerID, CustomerName,
CustomerAddress,city,Region,Postal_code, CustomerPhone)
SELECT CustomerID, CustomerName,
CustomerAddress,city,Region,Postal_code, CustomerPhone
FROM data.Customer;

-- Populate the Store table from the source database
INSERT INTO RETAILS.Store(StoreID, StoreName,
StoreAddress,city,Region,Postal_code, StorePhone)
SELECT StoreID, StoreName, StoreAddress,city,Region,Postal_code,
StorePhone
FROM data.Store;

-- Populate the Supplier table from source database
INSERT INTO RETAILS.Supplier (Supplier_ID, Comapany_Name,
Contact_Name, Contact_Address, City, Region, Postal_Code, Country,
Contact_Phone)
select Supplier_ID, Comapany_Name, Contact_Name, Contact_Address,
City, Region, Postal_Code, Country, Contact_Phone
from data.Supplier;

-- Populate the Category table from source database
INSERT INTO RETAILS.Category (Category_ID, Category_Name, Description,
Picture)
select Category_ID, Category_Name, Description, Picture
from data.Category;

-- populate the Product table from source database
```

```
INSERT INTO RETAILS.Product (ProductID, ProductName, ProductPrice,
Category_ID, Supplier_ID, Unit_Price, Unit_in_Stocks, Discount)
SELECT ProductID, ProductName, ProductPrice, Category_ID, Supplier_ID,
Unit_Price, Unit_in_Stocks, Discount
From data.Product;

-- populate the Sales table from source database
INSERT INTO RETAILS.Sales (SaleID, ProductID, CustomerID, StoreID,
SalePrice, SaleDate)
SELECT SaleID, ProductID, CustomerID, StoreID, SalePrice, SaleDate
from data.Sales;


-- Row counts of the customer table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Customer;

-- Row counts of Store table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Store;

-- Row counts of Supplier table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Supplier;

-- Row counts of Category table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Category;

-- Row counts of the Product table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Product;

-- Row counts of the Sales table
SELECT COUNT(*) as total_no_of_row FROM RETAILS.Sales;




-- First ten rows of the customer table
SELECT * FROM RETAILS.Customer
FETCH NEXT 10 ROWS ONLY;

-- First ten rows of Store table
SELECT * FROM RETAILS.Store
FETCH NEXT 10 ROWS ONLY;

-- First ten rows of the Supplier table
SELECT * FROM RETAILS.Supplier
FETCH NEXT 10 ROWS ONLY;
```

```sql
-- First ten rows of Category table
SELECT * FROM RETAILS.Category
FETCH NEXT 10 ROWS ONLY;

-- First ten rows of Product table
SELECT * FROM RETAILS.Product
FETCH NEXT 10 ROWS ONLY;

-- First ten rows of Sales table
SELECT * FROM RETAILS.Sales
FETCH NEXT 10 ROWS ONLY;
```