E-commerce Database Application

Final Project Report

Subject: - ISM6208 Data Warehousing

Dinesh Datta Pendyala (U96239813)
Sri Vaishnav Vedanaparthi (U29421544)
Arun Kumar Pathipati ((U37833602)
Deeksha Katara (U88504979)
Siva Sankari Ravipati (U36650128)

Major in

BUSINESS ANALYTICS AND INFORMATION SYSTEMS

Under the guidance of

Dr. Don Berndt

MUMA COLLEGE OF BUSINESS

UNIVERSITY OF SOUTH FLORIDA

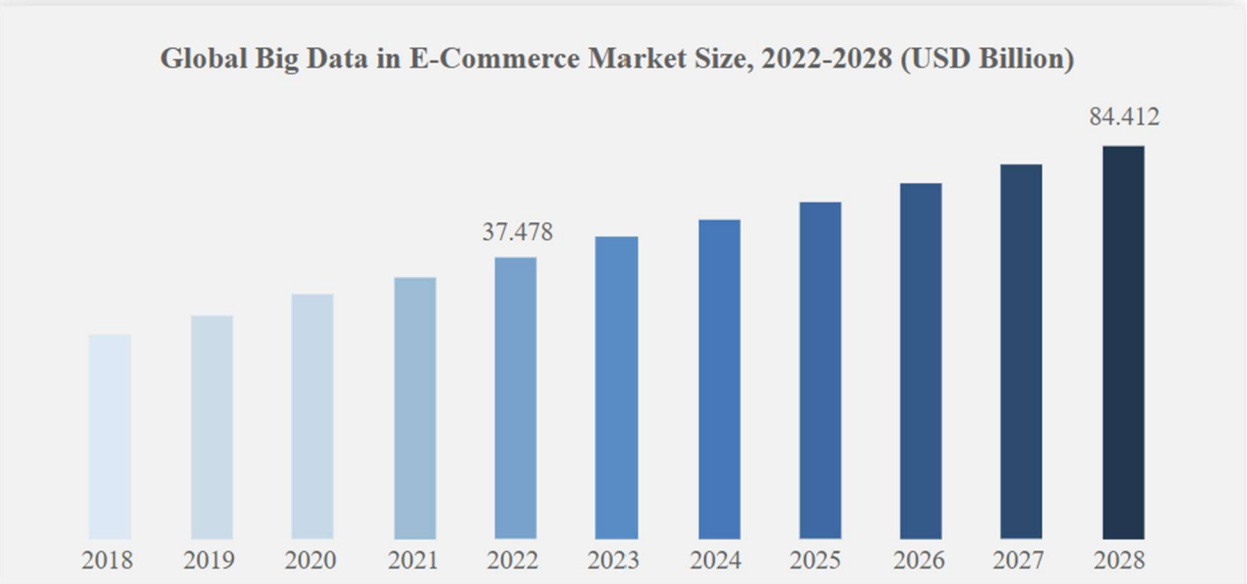
Table of Contents

Page. No

Executive Summary:

The main purpose of this project is to Build and Maintain a Database of an E-commerce company. Nowadays selling a product through a website or Online has increased due to N number of products and varieties. It is important for them to keep track of the Sales and Performance of the best products. So, having a Database to keep track of the data will help in understanding the customer better and offering the services in a better way. Through this project, we will build an Oracle SQL database where it stores information related to customers, payments, products, shipping-related information, etc.

Snapshot 1: Global E-commerce Big Data Market Size



From the above snapshot 1, we can conclude that Global Big Data Market Size is increasing year over year. It is expected to be 84 billion dollars by the year 2028. Storage of sales data is going to play a vital role in E-commerce and understanding customer behavior.

Problem Statement:

Online shopping has been increasing a lot since 2016 onwards due to an increase in connectivity and a feasible supply chain. Many companies are selling their products through stores, websites, and through third parties (Amazon, Walmart). While customers browse the various products, shop, and sell from the website. All the data has to be recorded and stored in the Database. During the initial days, data has been stored in Oracle SQL Server, MYSQL server, or MariaDB. Storing the data has become easy because of Cloud services Like AWS, AZURE, and GCP. Customers' transaction data has been stored in the database and the data can be used for analysis and can serve them in a better way. Building a SQL database could solve the problem of understanding and offering services to the customer.

Literature Review:

In this project, we are making use of various resources to build an E-commerce database. Starting with online resources Kaggle and data have been transformed using options available in Excel. Data has been divided into 8 different tables which follow normalization rules. Below is the list of tools and sources:

1. Kaggle
2. Microsoft Excel
3. Oracle SQL Developer
4. Data Mining
5. Tableau for Visualization

Data Collection and Preparation:

In terms of data, It has been gathered from multiple sources and a few columns are generated manually. Data has been designed according to normalization rules to avoid redundancy. There are 8 tables for the E-commerce database and all data sources are uploaded to the Oracle SQL server. The orders dataset consists of 50k rows and the customer dataset consists of 25k rows of data which has been taken from Kaggle. Below is the sources and references database that has been designed.

Database Design:

Below are the steps involved in defining business processes and dimensional modelling:

- Grain 1: Identify the business process. In this case, we have chosen to build an E-commerce database as a business idea. We will be analyzing the frequency of products and trends in products sold across the regions.
- Grain 2: In this case, we will be analyzing products ordered by the customers.
- Grain 3: Dimensional tables store textual information about the business process. Dimensional tables are PRODUCT_DETAILS, PRODUCTID_DETAILS, CATEGORY_DETAILS, PAYMENT_DETAILS, CUSTOMER_DETAILS, SHIPPER_DETAILS, and SUPPLIER_DETAILS.
- Grain 4: Fact tables are used to store numeric measures of a business process. Numeric tables are ORDER_DETAILS

STAR Schema:

Overall, in this application, we have 8 tables in which 7-dimensional tables, and 1 Fact table which stores the all facts. Below is the STAR Schema design of an E-commerce database:

Fact table:

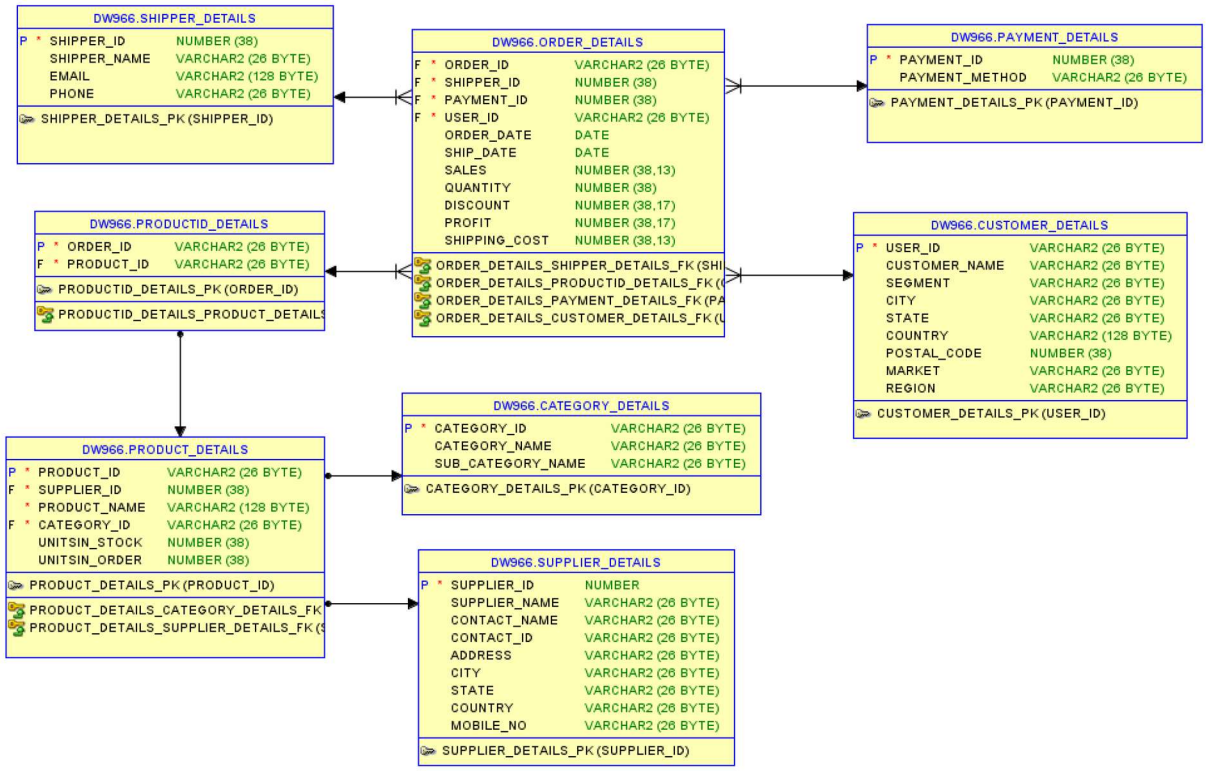
- ORDER_DETAILS

Dimensional tables:

- PRODUCT_DETAILS
- PRODUCTID_DETAILS
- CATEGORY_DETAILS
- PAYMENT_DETAILS
- CUSTOMER_DETAILS

- SHIPPER_DETAILS
- SUPPLIER_DETAILS

Snapshot 2: STAR Schema of E-commerce Database design



From the above snapshot, the Fact table orders data at the center, and dimensional tables are joined to the fact table using primary keys.

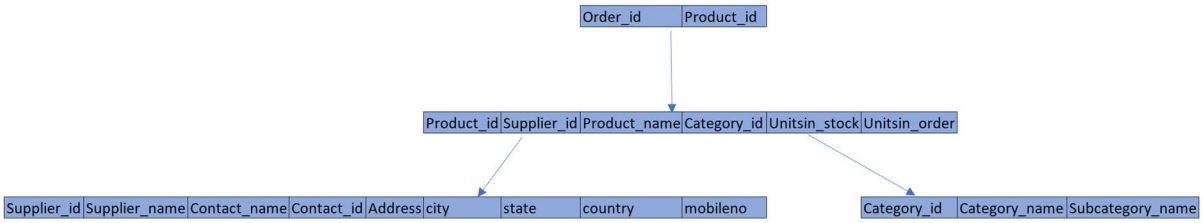
Tables are joined using the below combinations:

1. Orders table (SHIPPER_ID -FK) and Shipper Table (SHIPPER_ID – PK)
2. Orders table (ORDER_ID-FK) and Productid Table (ORDER_ID – PK)
3. Productid table (PRODUCT_ID - FK) and Product details table (PRODUCT_ID – PK)
4. Product details table (SUPPLIER_ID - FK) and Supplier details table (SUPPLIER_ID - PK)
5. Product details table (CATEGORY_ID - FK) and Category table (CATEGORY_ID - PK)
6. Orders table (PAYMENT_ID-FK) and Payment Table (PAYMENT_ID – PK)
7. Orders table (USER_ID-FK) and Users Table (USER_ID – PK)

Normalization:

Normalization is a technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update, and Deletion Anomalies. In this database, In product details table has been divided into three different databases to avoid redundancy. Below are the design details:

Snapshot 3: Database Normalization

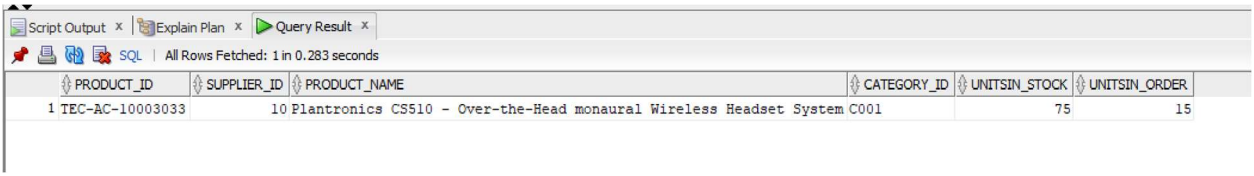


From the above snapshot, category_id consists of various category names and sub-category names which leads to data redundancy due to repetition. Because of the category_id column, category names and sub-category names are stored in another table where redundancy is discarded. Now, data has been normalized.

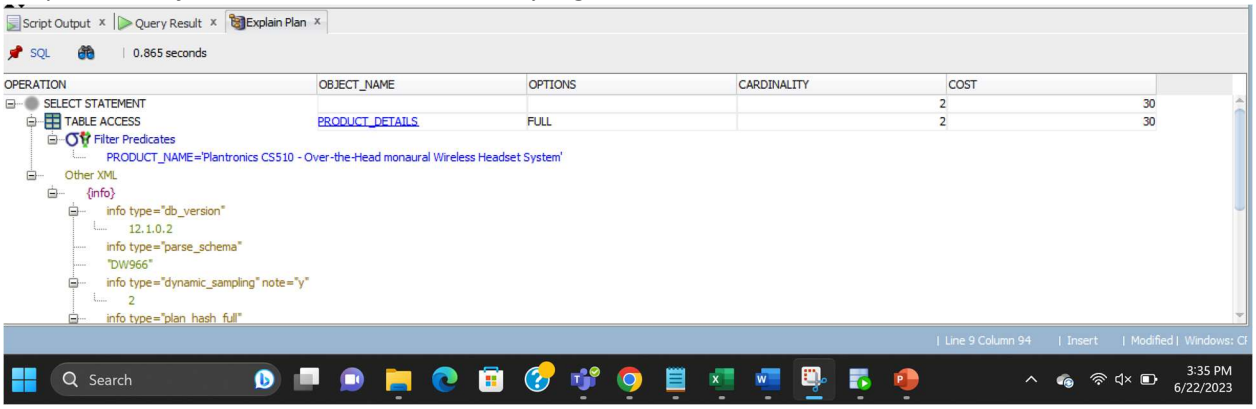
Indexing:

Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Because of indexing performance will be improved and the time to locate the data will be decreased. Below are indexes implemented for better performance.

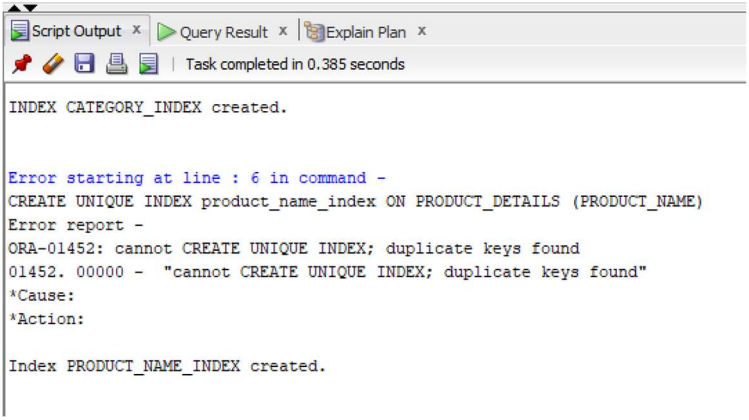
Snapshot 4: Output of query with product_name filter.



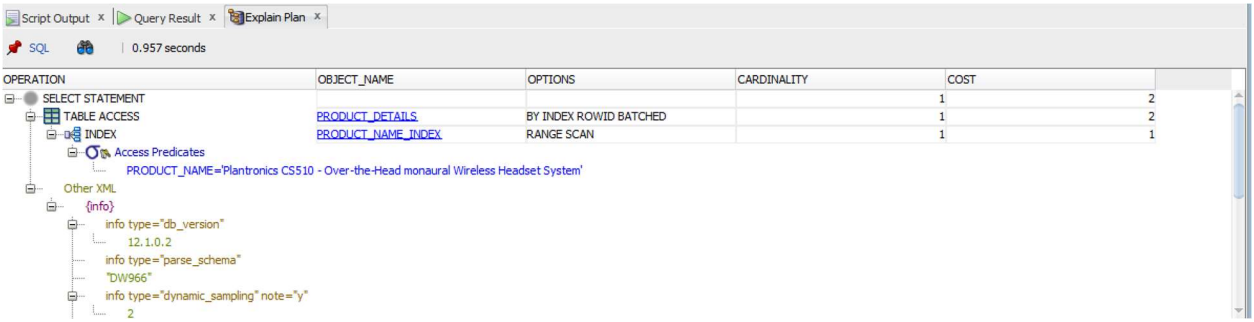
Snapshot 5: Before index creation, cost is very high



Snapshot 6: Category index creation



Snapshot 7: Range scan and improved performance



Initially without indexing, the COST and CARDINALITY of the query are very high but after creating Index on SUBCATEGORY_NAME, cost and cardinality both are reduced. Indexing improves the performance of the query.

Query writing:

1. Write a query to understand the country, city wise orders.

   ```
   SELECT COUNTRY,REGION,STATE,CITY,COUNT(ORDER_ID) as orders
   from customer_details cid
   inner join order_details od
   on cid.user_id=cid.user_id
   group by COUNTRY,REGION,STATE,CITY
   order by orders desc;
   ```

   Output:

	COUNTRY	REGION	STATE	CITY	ORDERS
1	United States	East	New York	New York City	5468
2	United States	West	California	Los Angeles	4771
3	United States	East	Pennsylvania	Philadelphia	3879
4	United States	West	California	San Francisco	3282
5	Dominican Republic	Caribbean	Santo Domingo	Santo Domingo	3024
6	United States	West	Washington	Seattle	2691
7	United States	Central	Texas	Houston	2562
8	Indonesia	Southeast Asia	Jakarta	Jakarta	2369
9	Australia	Oceania	New South Wales	Sydney	2278
10	Philippines	Southeast Asia	National Capital	Manila	2182
11	United States	Central	Illinois	Chicago	2040
12	Nicaragua	Central	Managua	Managua	1908
13	United Kingdom	North	England	London	1740
14	Germany	Central	Berlin	Berlin	1721
15	Thailand	Southeast Asia	Bangkok	Bangkok	1686
16	Honduras	Central	Francisco Morazán	Tegucigalpa	1670
17	Mexico	North	Distrito Federal	Mexico City	1665

2. Write a query to understand payment methods preferred by customers.

```
SELECT DISTINCT PAYMENT_METHOD, COUNT(ORDER_ID) as orders
from PAYMENT_DETAILS PD
INNER JOIN ORDER_DETAILS OD
ON PD.PAYMENT_ID=OD.PAYMENT_ID
GROUP BY PAYMENT_METHOD
ORDER BY orders desc;
```

Output:

	PAYMENT_METHOD	ORDERS
1	Venmo	5257
2	Cryptocurrency	5194
3	Mobile Payment	5161
4	PayPal	5141
5	Credit Card	5110
6	Check	5108
7	Cash	5088
8	Apple Pay	5082
9	Bank Transfer	5078
10	Google Pay	5071

3. Write a query to understand products that are in demand.

```
SELECT PRODUCT_NAME,COUNT(ORDER_ID) as orders
from PRODUCT_DETAILS PD
INNER JOIN PRODUCTID_DETAILS PID
ON PID.PRODUCT_ID=PD.PRODUCT_ID
GROUP BY PRODUCT_NAME
ORDER BY orders desc;
```

Output:

	PRODUCT_NAME	ORDERS
1	Staples	234
2	Cardinal Index Tab, Clear	108
3	Ibico Index Tab, Clear	108
4	Eldon File Cart, Single Width	103
5	Sanford Pencil Sharpener, Water Color	89
6	Rogers File Cart, Single Width	87
7	Stanley Pencil Sharpener, Water Color	82
8	Smead File Cart, Single Width	77
9	Avery Index Tab, Clear	76
10	Tenex File Cart, Single Width	75
11	Acco Index Tab, Clear	75
12	Stockwell Paper Clips, Assorted Sizes	69
13	Acco Binder Covers, Recycled	66
14	Binney & Smith Pencil Sharpener, Water Color	66
15	Sanford Pencil Sharpener, Easy-Erase	65
16	Avery 3-Hole Punch, Recycled	63
17	Avery Binder, Economy	61

4. Write a query to understand shippers who are in demand.

```
SELECT DISTINCT SHIPPER_NAME, COUNT(ORDER_ID) as orders
from shipper_details sid
inner join ORDER_DETAILS OD
ON SID.SHIPPER_ID=OD.SHIPPER_ID
GROUP BY SHIPPER_NAME
ORDER BY ORDERS DESC;
```

Output:

	SHIPPER_NAME	ORDERS
1	Quick Cargo	4800
2	Swift Trans	4755
3	Speedy Ship	4686
4	ABC Shipping	4670
5	XYZ Logistics	4653
6	Fast Express	4631
7	Mega Movers	4627
8	Stellar Shippers	4602
9	Reliable Trans	4582
10	Eagle Freight	4561

Exploratory Data Analysis:

In this phase, we are going to explore more about the data.

The shape of the dataset:

```
In [6]: global_data.shape
Out[6]: (51290, 24)
```

Null values:

```
In [9]: global_data.isna().sum()
Out[9]: Row ID               0
        Order ID             0
        Order Date           0
        Ship Date            0
        Ship Mode            0
        Customer ID          0
        Customer Name        0
        Segment              0
        City                 0
        State                0
        Country              0
        Postal Code      41296
        Market               0
        Region               0
        Product ID           0
        Category             0
        Sub-Category         0
        Product Name         0
        Sales                0
        Quantity             0
        Discount             0
        Profit               0
        Shipping Cost        0
        Order Priority       0
        dtype: int64
```

- There are null values exists in Postal code column and whole column removed as they doesn't add much value to the dataset.

Unique Category values:

```
In [14]: global_data["Category"].unique()
Out[14]: array(['Technology', 'Furniture', 'Office Supplies'], dtype=object)
```

Unique sub-category values:

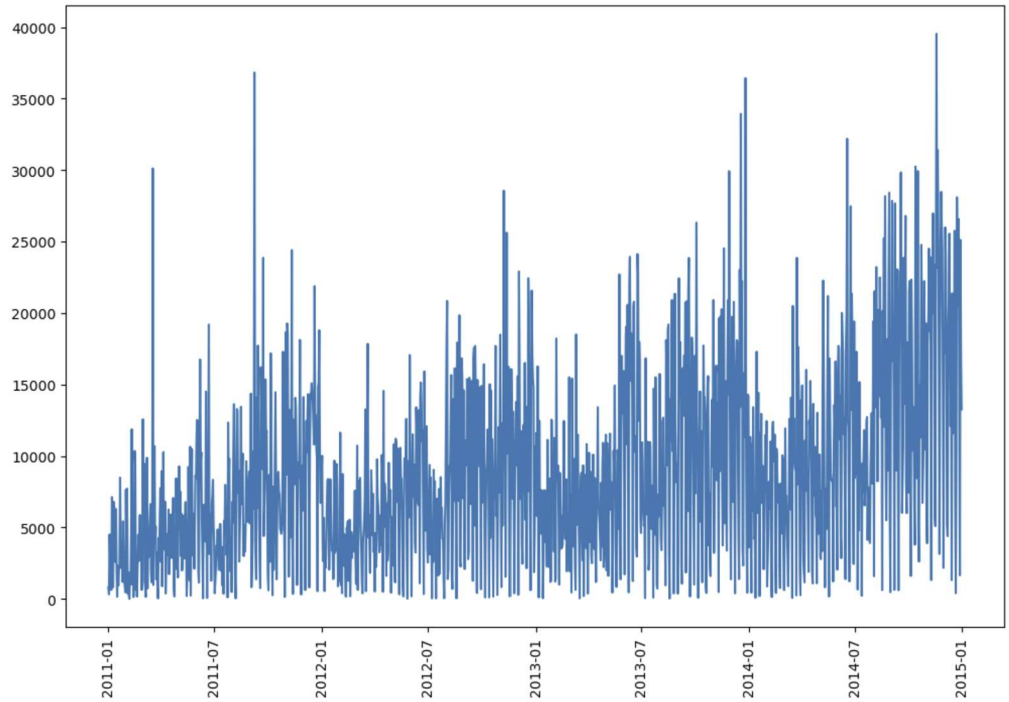
```
In [15]: global_data["Sub-Category"].unique()
Out[15]: array(['Accessories', 'Chairs', 'Phones', 'Copiers', 'Tables', 'Binders',
                'Supplies', 'Appliances', 'Machines', 'Bookcases', 'Storage',
                'Furnishings', 'Art', 'Paper', 'Envelopes', 'Fasteners', 'Labels'],
               dtype=object)
```

Unique product names:

```
In [17]: global_data['Product Name'].nunique()
```

```
Out[17]: 3788
```

Sales trend:



Model Building using LSTM:

Train and Test Data:

```
In [26]: #train and test data
training_size = int(len(df_final)*0.75)
test_size = len(df_final)-training_size

train_data = df_final[0:training_size, :]
test_data = df_final[training_size:len(df_final), :1]
```

```
In [27]: training_size, test_size
```

```
Out[27]: (1072, 358)
```

Model Summary:

```
In [32]: from tensorflow.keras.layers import Dense
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import LSTM
```

```
In [33]: n_steps = 10
         n_features = 1
         #the sequential function processess the data as a stream of sequential integers
         model = Sequential()
         #defining the model
         model.add(LSTM(50, return_sequences=True, input_shape=(n_steps, n_features)))
         model.add(LSTM(50, return_sequences=True))
         model.add(LSTM(50))
         model.add(Dense(1))
         model.compile(optimizer='adam', loss='mse' )
```

```
In [34]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 50)            10400

 lstm_1 (LSTM)               (None, 10, 50)            20200

 lstm_2 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Train and Test RMSE values:

```
In [39]: import math
         from sklearn.metrics import mean_squared_error
         math.sqrt(mean_squared_error(y_train, train_predict))

Out[39]: 7759.1103669438435
```

```
In [40]: math.sqrt(mean_squared_error(y_test, test_predict))

Out[40]: 10128.623132462271
```

From the above values, we can say that Test RMSE values are greater than Train RMSE. Model has overfitting issue. This can be improved by changing input selections.

Reporting, Modelling, and Story Telling:

Analytical SQL Functions usage:

Query 1: Rank the customers based on their purchase frequency

```
with orders as (
SELECT DISTINCT CUSTOMER_NAME,COUNT(ORDER_ID) as orders
from CUSTOMER_DETAILS CD
INNER JOIN ORDER_DETAILS OD
ON CD.USER_ID=OD.USER_ID
GROUP BY CUSTOMER_NAME
ORDER BY orders desc)
```

```
select customer_name, orders,
rank() over (order by orders desc) as orders_rank
from orders
```

Output:

	CUSTOMER_NAME	ORDERS	ORDERS_RANK
1	Anna Andreadi	1185	1
2	Patrick O'Brill	1082	2
3	Shahid Collister	1078	3
4	Harry Marie	1045	4
5	Seth Vernon	1034	5
6	Zuschuss Carroll	1008	6
7	Rick Bensley	989	7
8	Joy Bell-	984	8
9	Fred Hopkins	934	9
10	Michael Moore	882	10
11	Tonja Turnell	878	11
12	Laura Armstrong	872	12
13	Kristen Hastings	872	12
14	Michael Chen	870	14
15	Ross Baird	864	15
16	Laurel Beltran	863	16

Query 2: Rank the countries based on their orders.

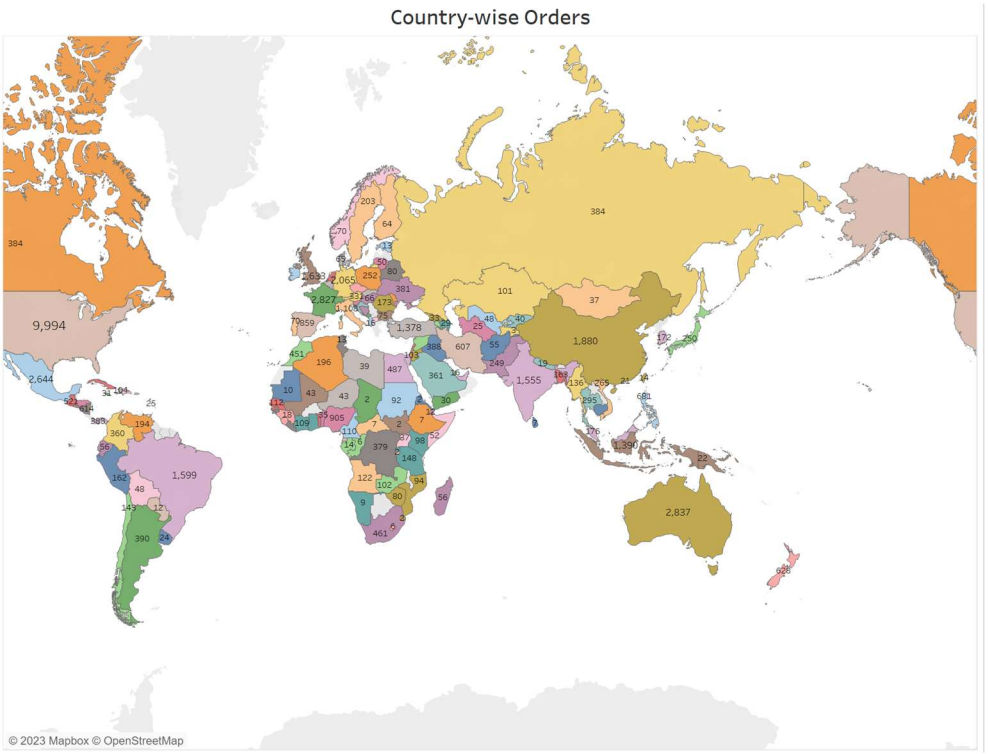
```
with country_data as (
SELECT DISTINCT COUNTRY,REGION,STATE, COUNT(ORDER_ID) as orders
from CUSTOMER_DETAILS CD
JOIN ORDER_DETAILS OD
ON CD.USER_ID=OD.USER_ID
GROUP BY  COUNTRY,REGION,STATE
ORDER BY orders desc)

select COUNTRY,REGION,STATE,orders,
DENSE_RANK() OVER (ORDER BY orders desc) as country_rank
from country_data
```

Output:

	COUNTRY	REGION	STATE	ORDERS	COUNTRY_RANK
1	United States	West	California	13213	1
2	United Kingdom	North	England	11350	2
3	France	Central	Ile-de-France	7167	3
4	United States	East	New York	6863	4
5	United States	Central	Texas	6479	5
6	Australia	Oceania	New South Wales	6358	6
7	Australia	Oceania	Queensland	5634	7
8	Germany	Central	North Rhine-Westphalia	5403	8
9	El Salvador	Central	San Salvador	4016	9
10	United States	East	Pennsylvania	3935	10
11	Australia	Oceania	Victoria	3505	11
12	United States	West	Washington	3238	12
13	United States	Central	Illinois	3158	13
14	Brazil	South	São Paulo	3041	14
15	Dominican Republic	Caribbean	Santo Domingo	3024	15
16	Philippines	Southeast Asia	National Capital	2993	16

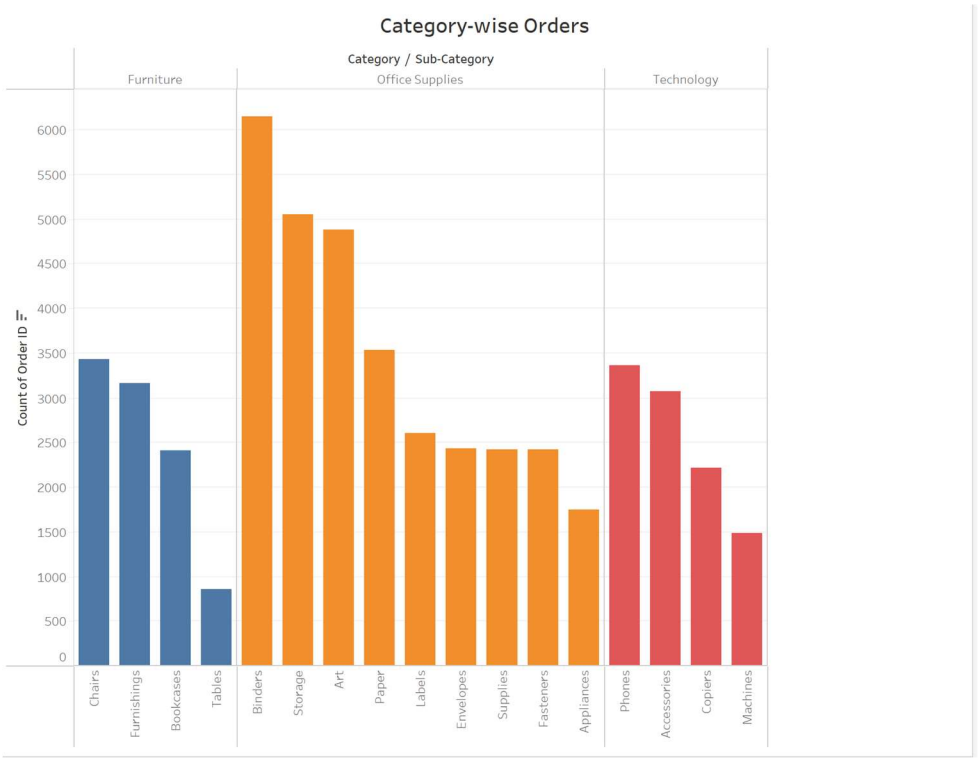
Data Visualization:

1. **Country-wise orders:**

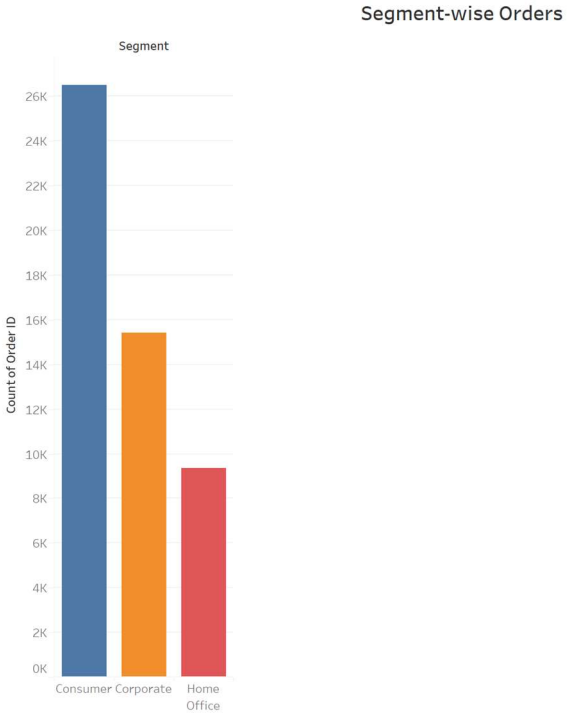


In terms of orders, the United States region has a high number of orders compared to any other region.

2. **Category-wise orders:**

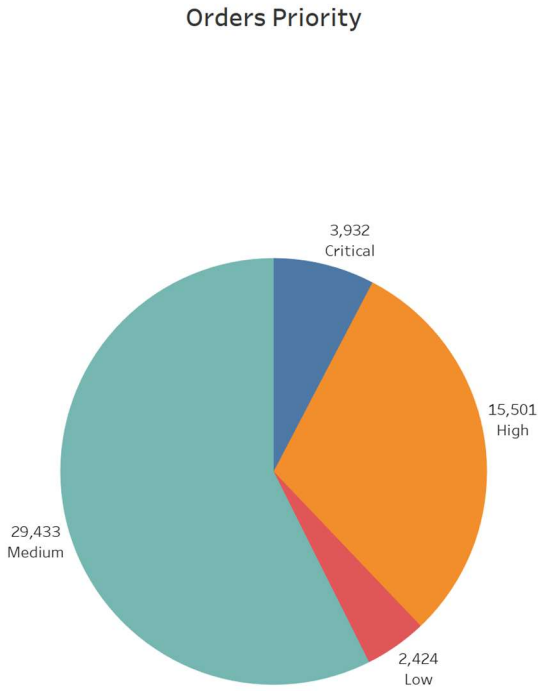


3. Segment-wise orders:

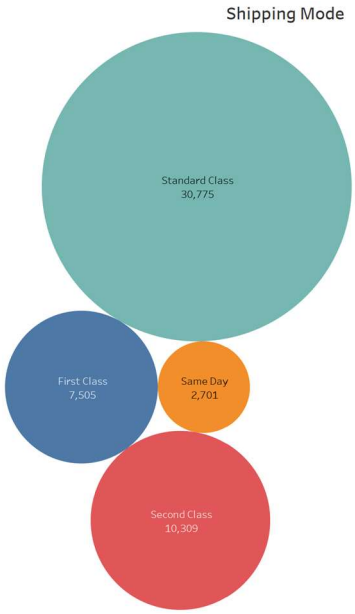


In terms of segment-wise orders, the Consumer segment has the highest number of orders followed by the Corporate and Home office.

4. Orders Priority



5. Shipping Mode:



Majority of customers choose standard class delivery rather than Same day and First Class.

Conclusion:

From the above project, we can conclude that:

1. Database has been built and follows normalization rules without any redundancy.
2. Database helps in analyzing the frequency of the products bought by customers.
3. From the data, we have a diverse range of payment methods and they are used by customers.
4. In terms of orders, the United States region has a high number of orders compared to any other region.
5. In terms of segment-wise orders, the Consumer segment has the highest number of orders followed by the Corporate and Home office.
6. Majority of customers choose standard class delivery rather than Same day and First Class.

References:

Below are the references used for building the E-commerce database:

1. https://www.kaggle.com/code/bharathishalini/notebook043f9714d1/input?select=Global+Superstore.xls
2. https://github.com/abdelatifsd/E-commerce-Database-Project/blob/master/1%20-%20ERD.pdf
3. https://www.princeton.edu/~rcurtis/ultradev/images/storediagram.gif