Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./~~Miss~~ OM ANIRUDHA SHETE of COMPUTER Department, Semester V with Roll No. 2103163 has completed a course of the necessary experiments in the subject Data Warehouse Mining under my supervision in the **Thadomal Shahani Engineering College** Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 23/10/2023

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

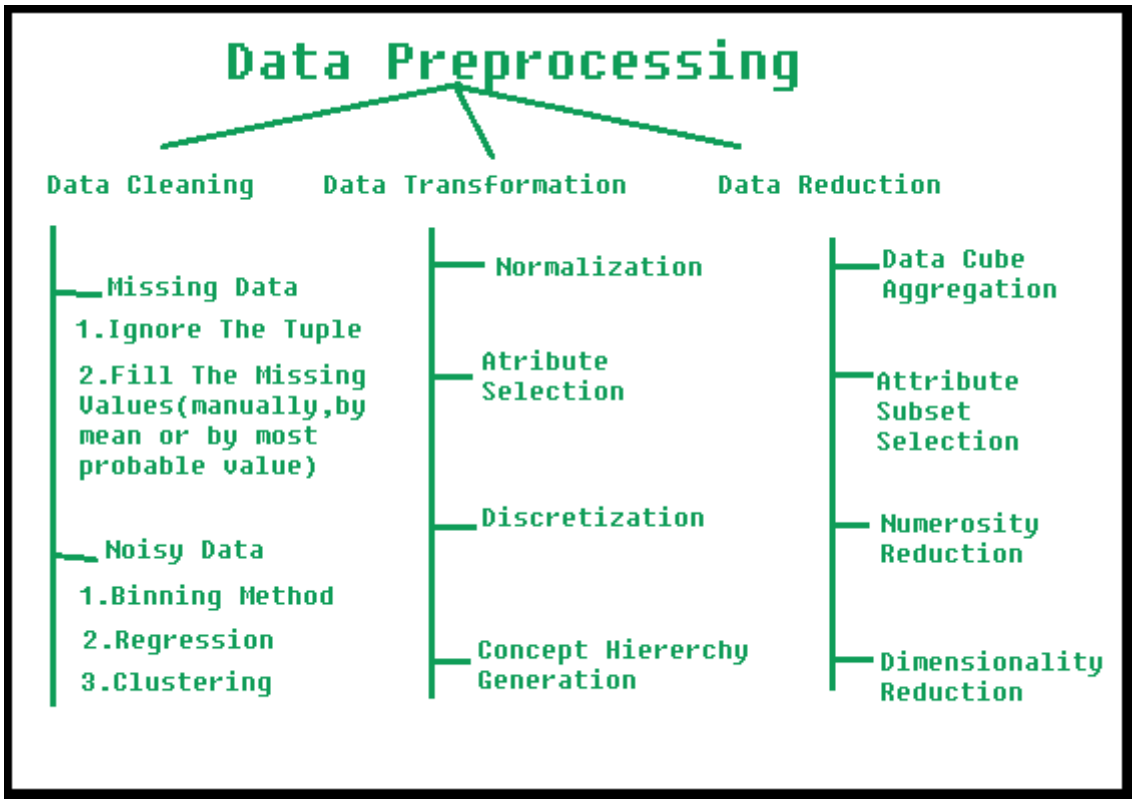
CONTENTS

EXPERIMENT - 01

Aim: Select a dataset and perform exploratory data analysis using Python

Theory:

Data Preprocessing:



Data preprocessing is a crucial step in the data mining process. It involves cleaning, transforming, and preparing raw data to make it suitable for analysis. The steps involved include:

- Data cleaning: This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.

- Data Integration: This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.

- Data Transformation: This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into
  discrete categories.

- Data Reduction: This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as
  feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

Data Transformation:

Data transformation in data mining refers to the process of converting raw data into a format that is suitable for analysis and modelling. The goal of data transformation is to prepare the data for data mining so that it can be used to extract useful insights and knowledge. Data transformation typically involves several steps, including:

- Smoothing: It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It
  helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form. The concept behind data smoothing is that it will be able to identify simple changes to help predict different
  trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn’t see otherwise.

- Aggregation: Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used. Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results. The collection of data is useful for everything from decisions concerning financing or business

strategy of the product, to pricing, operations, and marketing strategies. For example, Sales, data may be
aggregated to compute monthly& annual total amounts.

- Normalization: Data normalization involves converting all data variables into a given range.

- Generalization: It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example, Age initially in Numerical form (22, 25) is converted
into categorical value (young, old). For example, Categorical attributes, such as house addresses, may be generalized to higher-level definitions, such as town or country.

- Attribute Construction: Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

Data Discretization:

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attribute values of continuous data into a finite set of intervals with minimum data loss.

- Histogram analysis: Histogram refers to a plot used to represent the underlying frequency distribution of a continuous data set. Histogram assists the data inspection for data distribution. For example, Outliers, skewness representation, normal
distribution representation, etc.

- Binning: Binning refers to a data smoothing technique that helps to group a huge number of continuous values into smaller values. For data discretization and the development of idea hierarchy, this technique can also be used.

Data Visualization:

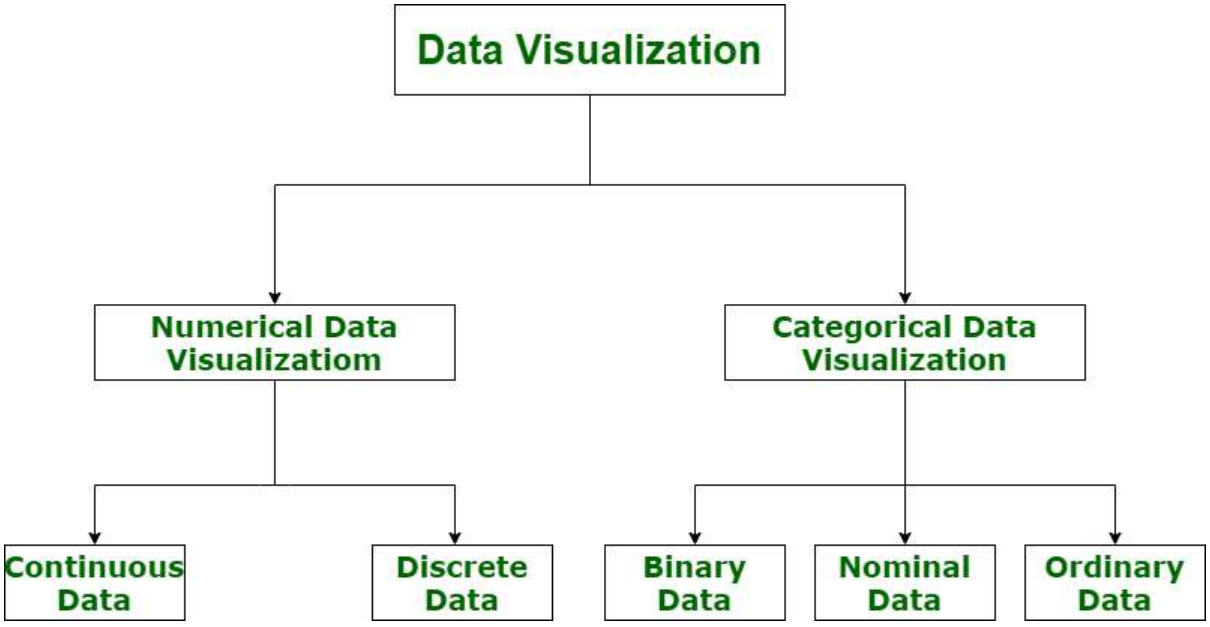
Data visualization is actually a set of data points and information that are represented graphically to make it easy and quick for user to understand. Data visualization is good if it has a clear meaning, purpose, and is very easy to interpret, without requiring context. Tools of data visualization provide an accessible way to see and understand trends, outliers, and patterns in data by using visual effects or elements such as a chart, graphs, and maps.

Categories of Data Visualization :

- Numerical Data : Numerical data is also known as Quantitative data. Numerical data is any data where data generally represents amount such as height, weight, age of a person, etc. Numerical data visualization is easiest way to visualize data. It is generally used for helping others to digest large data sets and raw numbers in a way that makes it easier to interpret into action. Numerical data is categorized into two categories :

1. Continuous Data- It can be narrowed or categorized (Example: Height measurements).
2. Discrete Data- This type of data is not “continuous” (Example: Number of cars or children’s a household has).

The type of visualization techniques that are used to represent numerical data visualization is Charts and Numerical Values. Examples are Pie Charts, Bar Charts, Averages, Scorecards, etc.



- Categorical Data : Categorical data is also known as Qualitative data. Categorical data is any data where data generally represents groups. It simply consists of categorical variables that are used to represent characteristics such as a person’s ranking, a
person’s gender, etc. Categorical data visualization is all about depicting key themes, establishing connections, and lending context. Categorical data is classified into three categories :

1. Binary Data- In this, classification is based on positioning (Example: Agrees or Disagrees).
2. Nominal Data- In this, classification is based on attributes (Example: Male or Female).
3. Ordinal Data- In this, classification is based on ordering of information (Example: Timeline or processes).

The type of visualization techniques that are used to represent categorical data is Graphics, Diagrams, and Flowcharts. Examples are Word clouds, Sentiment Mapping, Venn Diagram, etc.

Code:

```
import csv
import random
from faker import Faker
from datetime import datetime, timedelta

# Set the number of entries for each
dimension num_products = 1000
num_warehouses = 100
num_suppliers = 500
num_contacts = 1000
num_entries =

1000 fake = Faker()

# Function to generate a unique ID
def generate_unique_id(existing_ids):
   new_id = fake.random_int(min=1000,
   max=9999) while new_id in existing_ids:
      new_id = fake.random_int(min=1000,
   max=9999) existing_ids.add(new_id)
   return new_id

# Function to generate random date within a
range def generate_random_date(start_date,
end_date):
   time_delta = end_date - start_date
   random_days = random.randint(0,
   time_delta.days) return start_date +
   timedelta(days=random_days)

# Generate data and write to a single CSV file
with open('inventory_data.csv', mode='w', newline='') as
   file: writer = csv.writer(file)
   writer.writerow(["transaction_id", "product_id", "product_name",
"product_category", "product_brand", "warehouse_id", "warehouse_name",
"warehouse_location", "city_id", "supplier_id", "supplier_name", "contact_id", "phone",
"email", "arrival_date",
```

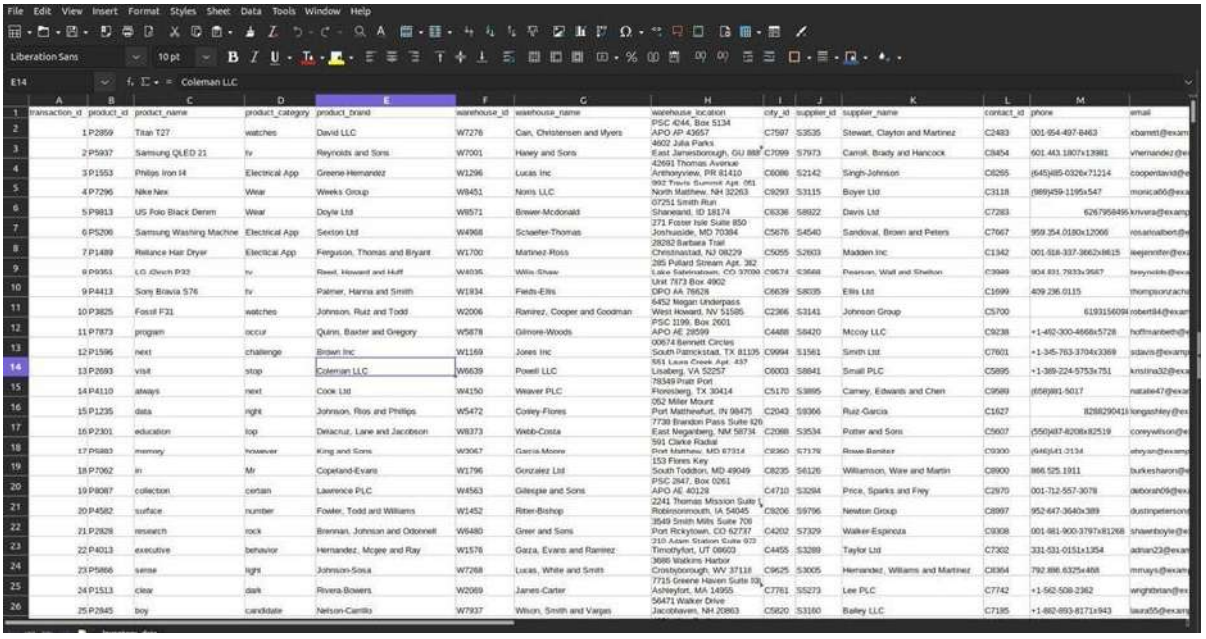
```
"dispatch_date", "number_of_boxes"])

    product_ids = set()
    warehouse_ids =
    set() supplier_ids =
    set()

    contact_ids = set()

    for i in
        range(num_entries):
        transaction_id = i + 1

        # Generate product data
        product_id =
        generate_unique_id(product_ids)
        product_name = fake.word()
        product_category = fake.word()
        product_brand =
        fake.company()

        # Generate warehouse data
        warehouse_id =
        generate_unique_id(warehouse_ids)
        warehouse_name = fake.company()
        warehouse_location = fake.address()
        city_id =
        generate_unique_id(contact_ids)

        # Generate supplier data
        supplier_id =
        generate_unique_id(supplier_ids)
        supplier_name = fake.company()

        # Generate contact data
        contact_id = generate_unique_id(contact_ids)
        phone = fake.phone_number()
        email = fake.email()

        # Generate transaction date data
        arrival_date = generate_random_date(datetime(2020, 1, 1), datetime(2023, 8, 1))
        dispatch_date = generate_random_date(arrival_date, datetime(2023, 8,
```

1)) number_of_boxes = random.randint(1, 100)

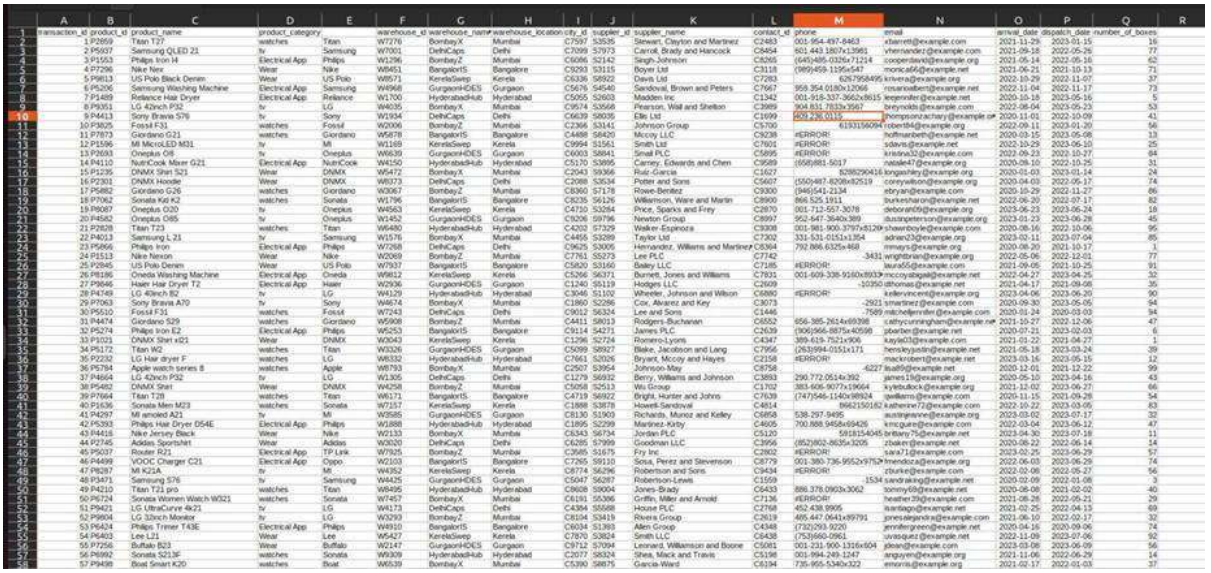
writer.writerow([transaction_id, f"P{product_id}", product_name, product_category, product_brand, f"W{warehouse_id}", warehouse_name, warehouse_location, f"C{city_id}", f"S{supplier_id}", supplier_name, f"C{contact_id}", phone, email, arrival_date.strftime('%Y-%m-%d'), dispatch_date.strftime('%Y-%m-%d'), number_of_boxes])

Output:

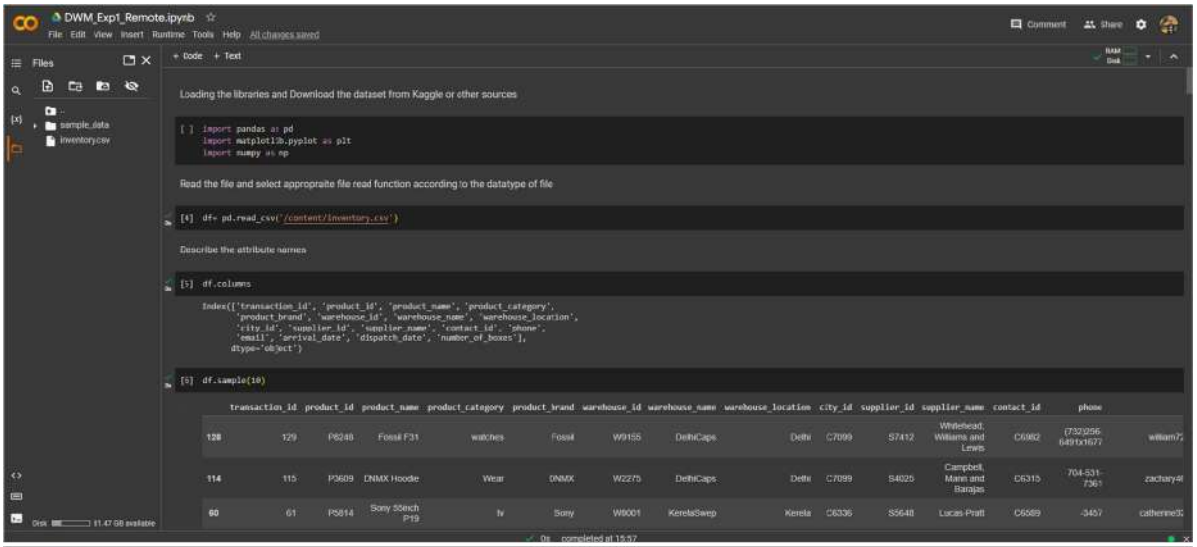
Original Dataset:

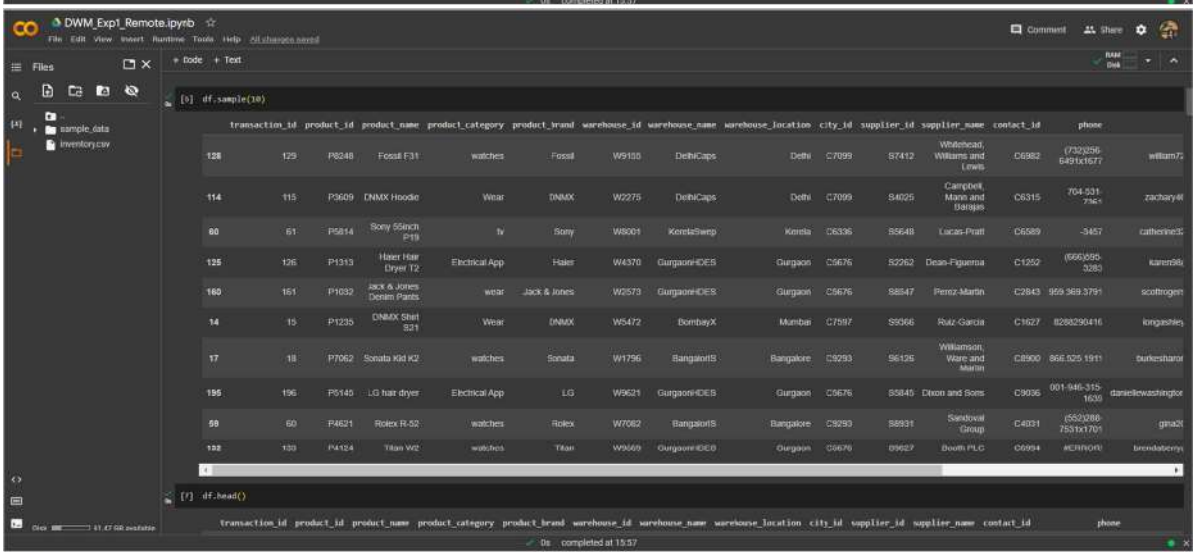


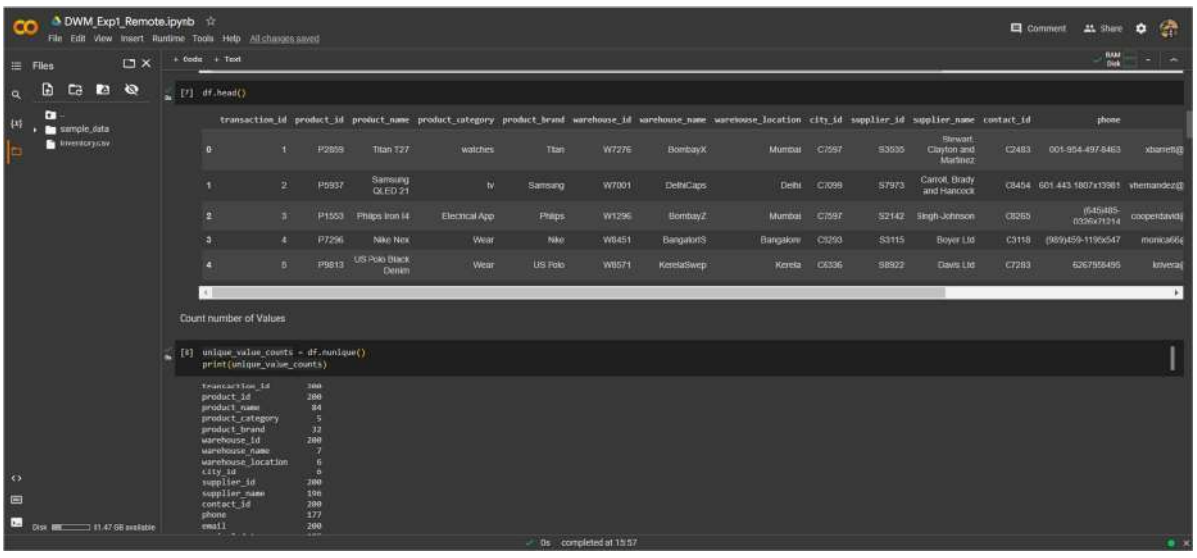
After data preprocessed:



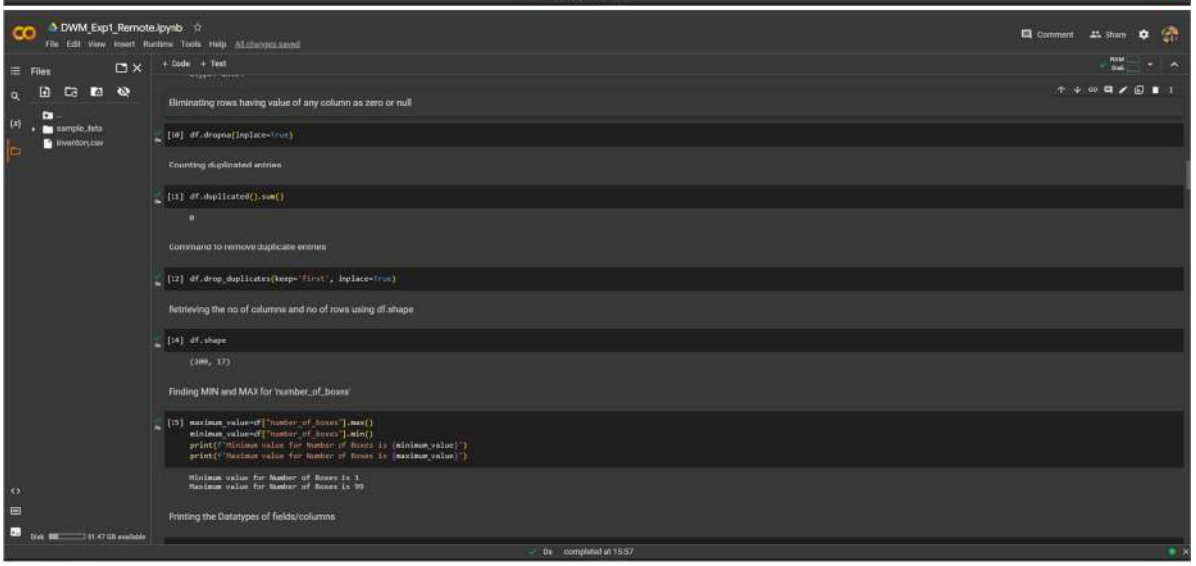
IMPLEMENTATION

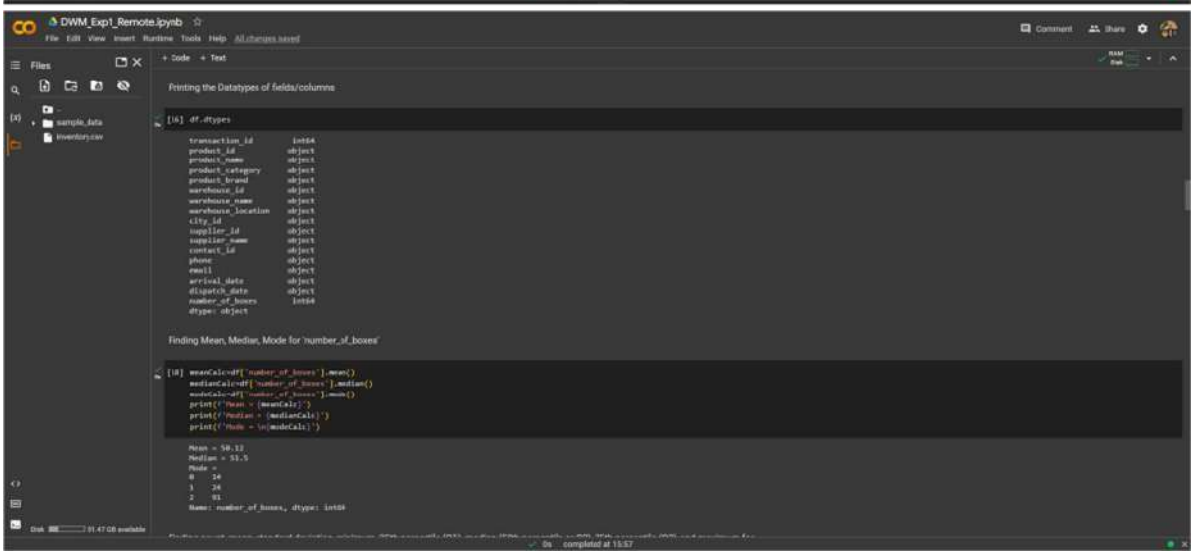


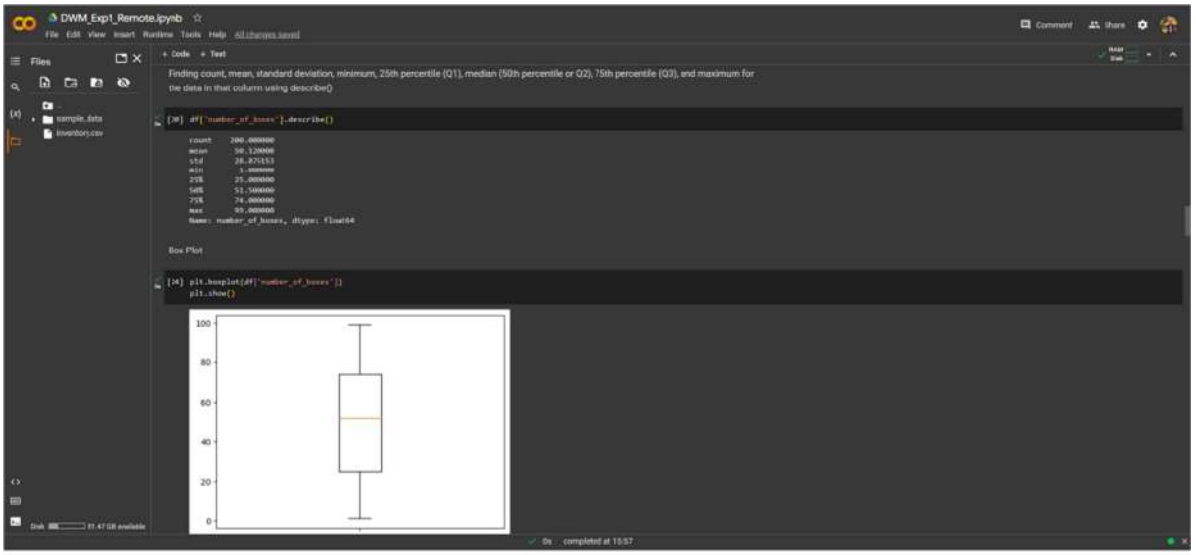


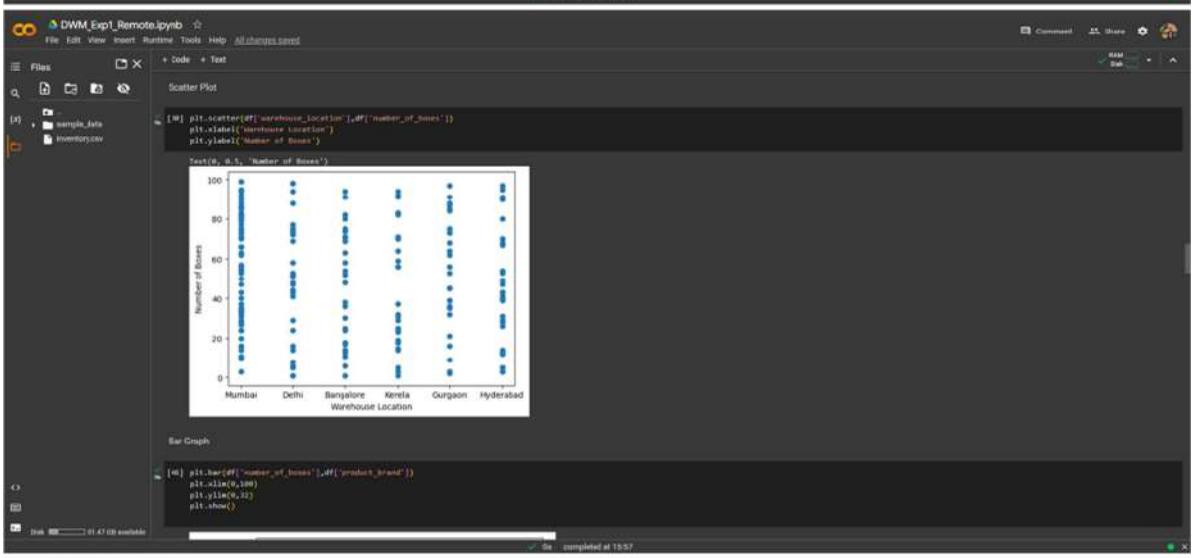


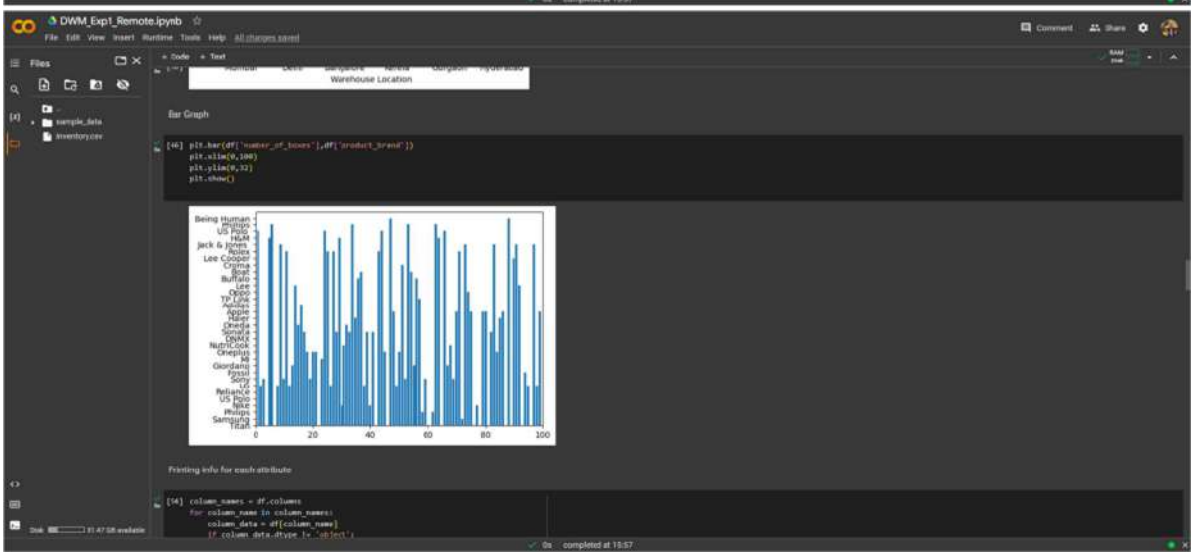

DWM_Exp1_Remote.ipynb
Count number of Values
unique_value_counts = df.nunique()
print(unique_value_counts)
transaction_id 200
product_id 100
product_name 84
product_category 5
product_brand 32
warehouse_id 200
warehouse_name 7
warehouse_location 6
city_id 6
supplier_id 200
supplier_name 196
contact_id 200
phone 177
email 200
arrival_date 185
dispatch_date 169
number_of_boxes 86
dtype: int64
Checking for Null Entries
[9] df.isnull().sum()
transaction_id 0
product_id 0
product_name 0
product_category 0
product_brand 0
warehouse_id 0
warehouse_name 0
warehouse_location 0
city_id 0
supplier_id 0
supplier_name 0
contact_id 0
phone 0
completed at 15:57

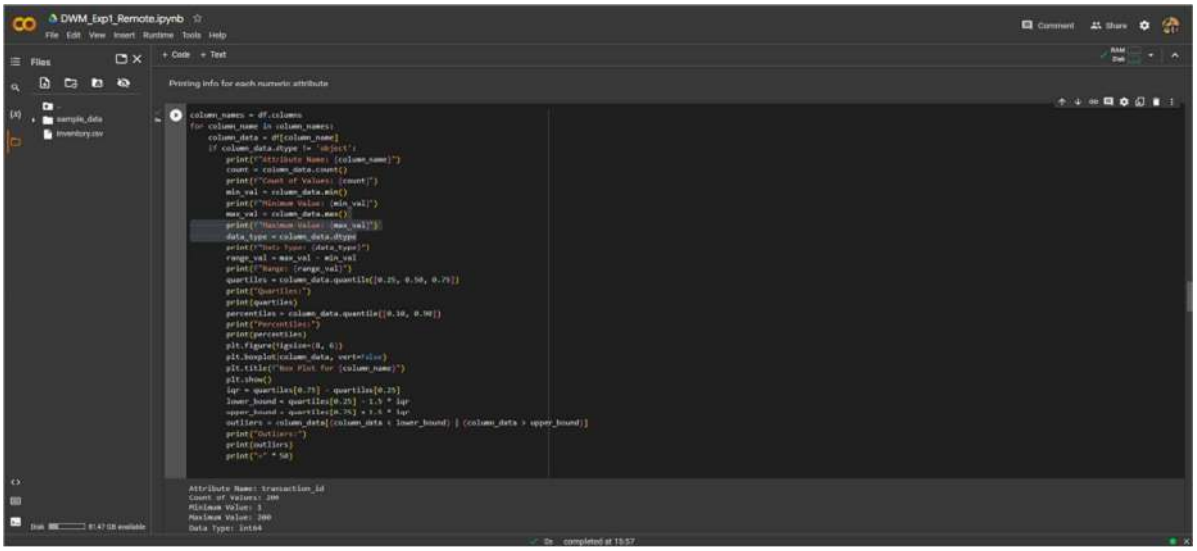

DWM_Exp1_Remote.ipynb
Eliminating rows having value of any column as zero or null
[10] df.dropna(inplace=True)
Counting duplicated entries
[11] df.duplicated().sum()
0
Command to remove duplicate entries
[12] df.drop_duplicates(keep='first', inplace=True)
Retrieving the no of columns and no of rows using df.shape
[14] df.shape
(200, 17)
Finding MIN and MAX for 'number_of_boxes'
[15] maximum_value=df["number_of_boxes"].max()
minimum_value=df["number_of_boxes"].min()
print(f"Minimum value for Number of Boxes is {minimum_value}")
print(f"Maximum value for Number of Boxes is {maximum_value}")
Minimum value for Number of Boxes is 1
Maximum value for Number of Boxes is 99
Printing the Datatypes of fields/columns
completed at 15:57

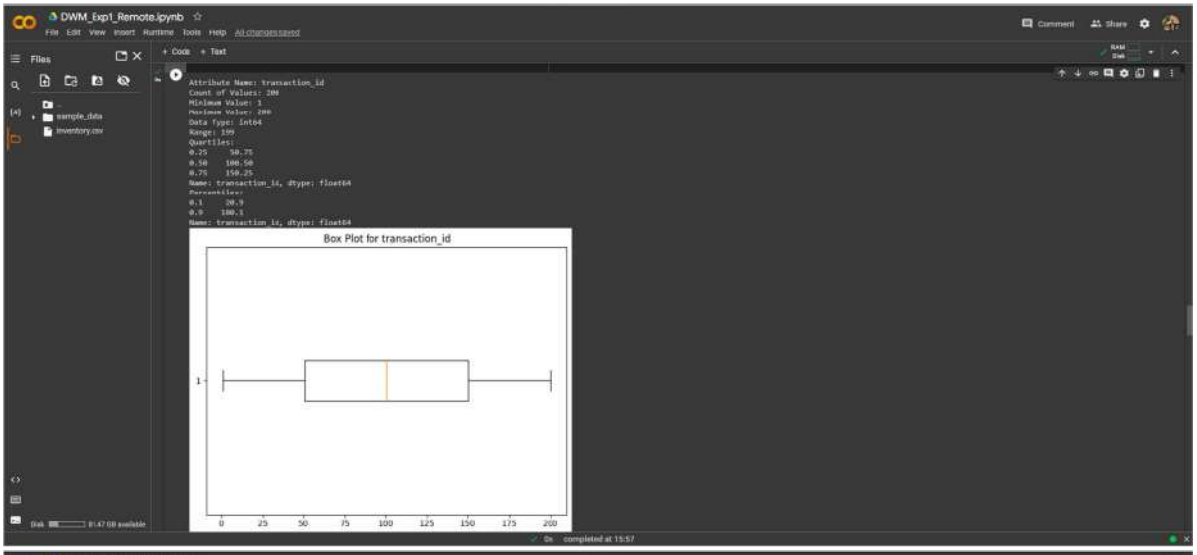

DWM_Exp1_Remote.ipynb
Printing the Datatypes of fields/columns
[16] df.dtypes
transaction_id int64
product_id object
product_name object
product_category object
product_brand object
warehouse_id object
warehouse_name object
warehouse_location object
city_id object
supplier_id object
supplier_name object
contact_id object
phone object
email object
arrival_date object
dispatch_date object
number_of_boxes int64
dtype: object
Finding Mean, Median, Mode for 'number_of_boxes'
[18] meanCalc=df['number_of_boxes'].mean()
medianCalc=df['number_of_boxes'].median()
modeCalc=df['number_of_boxes'].mode()
print(f'Mean = {meanCalc}')
print(f'Median = {medianCalc}')
print(f'Mode = \n{modeCalc}')
Mean = 50.12
Median = 51.5
Mode =
0 14
1 24
2 91
Name: number_of_boxes, dtype: int64
completed at 15:57

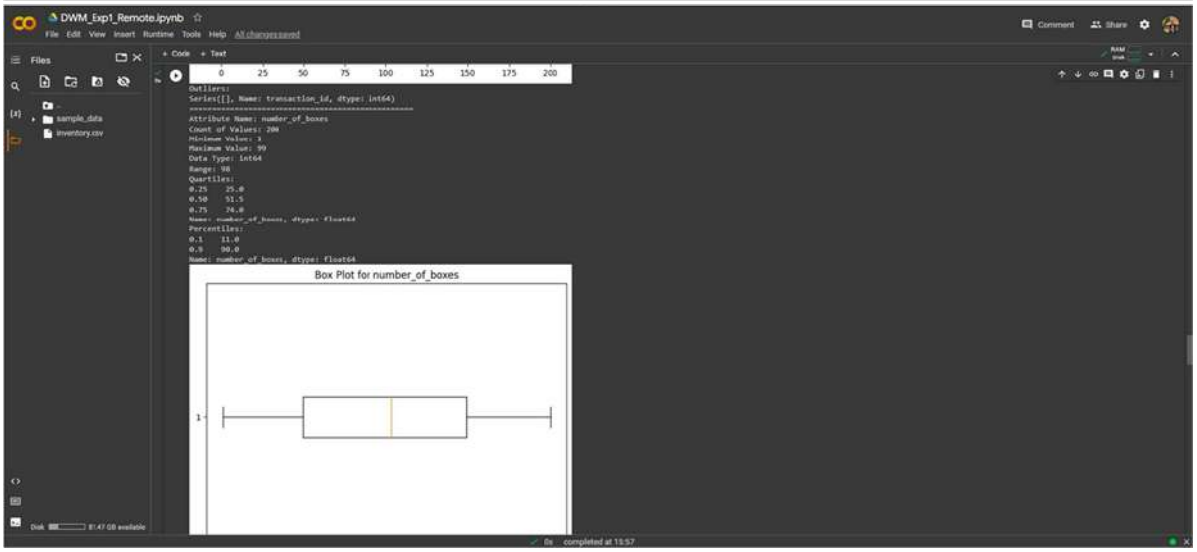

DWM_Exp1_Remote.ipynb
Finding count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile or Q2), 75th percentile (Q3), and maximum for the data in that column using describe()
[30] df['number_of_boxes'].describe()
count 200.000000
mean 50.320000
std 28.875153
min 1.000000
25% 25.000000
50% 51.500000
75% 74.000000
max 99.000000
Name: number_of_boxes, dtype: float64
Box Plot
[34] plt.boxplot(df['number_of_boxes'])
plt.show()

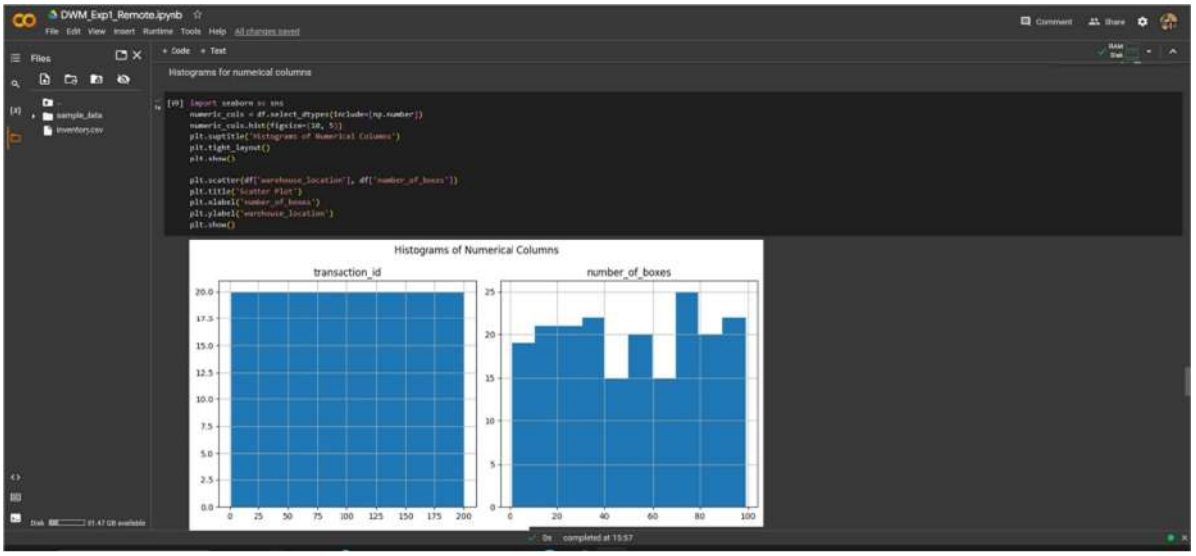

DWM_Exp1_Remote.ipynb
Scatter Plot
[38] plt.scatter(df['warehouse_location'],df['number_of_boxes'])
plt.xlabel('Warehouse Location')
plt.ylabel('Number of Boxes')
Text(0, 0.5, 'Number of Boxes')
Number of Boxes
Mumbai
Delhi
Bangalore
Kerela
Gurgaon
Hyderabad
Warehouse Location
Bar Graph
[46] plt.bar(df['number_of_boxes'],df['product_brand'])
plt.xlim(0,100)
plt.ylim(0,32)
plt.show()

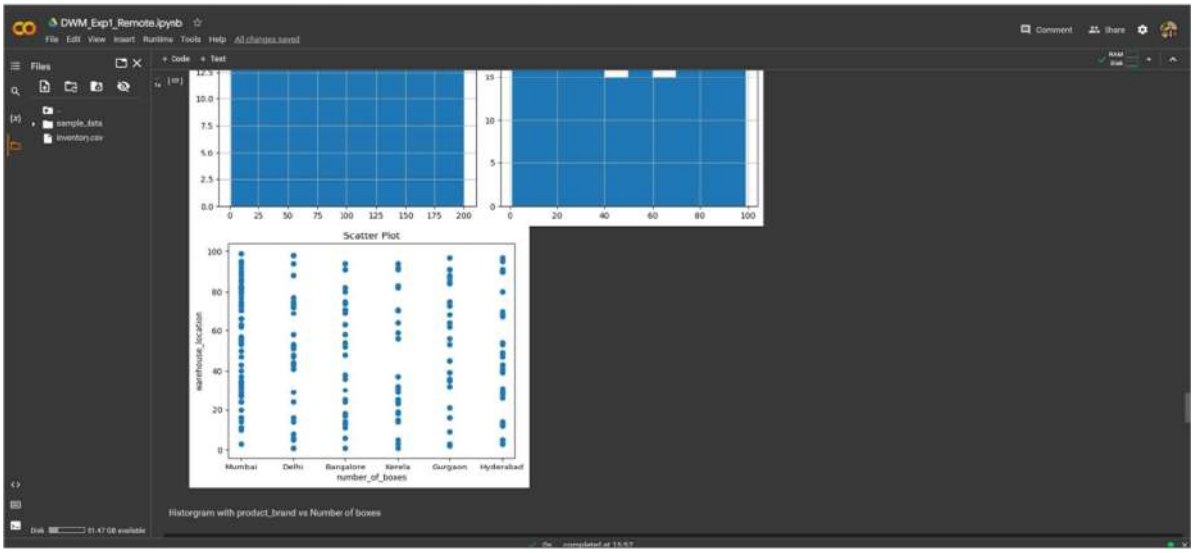

DWM_Exp1_Remote.ipynb
Warehouse Location
Bar Graph
[46] plt.bar(df['number_of_boxes'],df['product_brand'])
plt.xlim(0,100)
plt.ylim(0,32)
plt.show()
Being Human
Philips
US Polo
H&M
Jack & Jones
Rolex
Lee Cooper
Croma
Allen
Buffalo
Lee
Duke
TP Link
Apollo
Apple
Haier
Oneda
Sonata
OMAX
NutriCook
Oneplus
MI
Giordano
Fossil
Sony
LG
Reliance
US Polo
Nike
Philips
Samsung
Titan
0
20
40
60
80
100
Printing info for each attribute
[54] column_names = df.columns
for column_name in column_names:
column_data = df[column_name]
if column_data.dtype == 'object':


DWM_Exp1_Remote.ipynb
Printing info for each numeric attribute
column_names = df.columns
for column_name in column_names:
column_data = df[column_name]
if column_data.dtype != 'object':
print(f"Attribute Name: {column_name}")
count = column_data.count()
print(f"Count of Values: {count}")
min_val = column_data.min()
print(f"Minimum Value: {min_val}")
max_val = column_data.max()
print(f"Maximum Value: {max_val}")
data_type = column_data.dtype
print(f"Data Type: {data_type}")
range_val = max_val - min_val
print(f"Range: {range_val}")
quartiles = column_data.quantile([0.25, 0.50, 0.75])
print("Quartiles:")
print(quartiles)
percentiles = column_data.quantile([0.10, 0.90])
print("Percentiles:")
print(percentiles)
plt.figure(figsize=(8, 6))
plt.boxplot(column_data, vert=False)
plt.title(f"Box Plot for {column_name}")
plt.show()
iqr = quartiles[0.75] - quartiles[0.25]
lower_bound = quartiles[0.25] - 1.5 * iqr
upper_bound = quartiles[0.75] + 1.5 * iqr
outliers = column_data[(column_data < lower_bound) | (column_data > upper_bound)]
print("Outliers:")
print(outliers)
print("-" * 50)
Attribute Name: transaction_id
Count of Values: 200
Minimum Value: 1
Maximum Value: 200
Data Type: int64
completed at 15:57

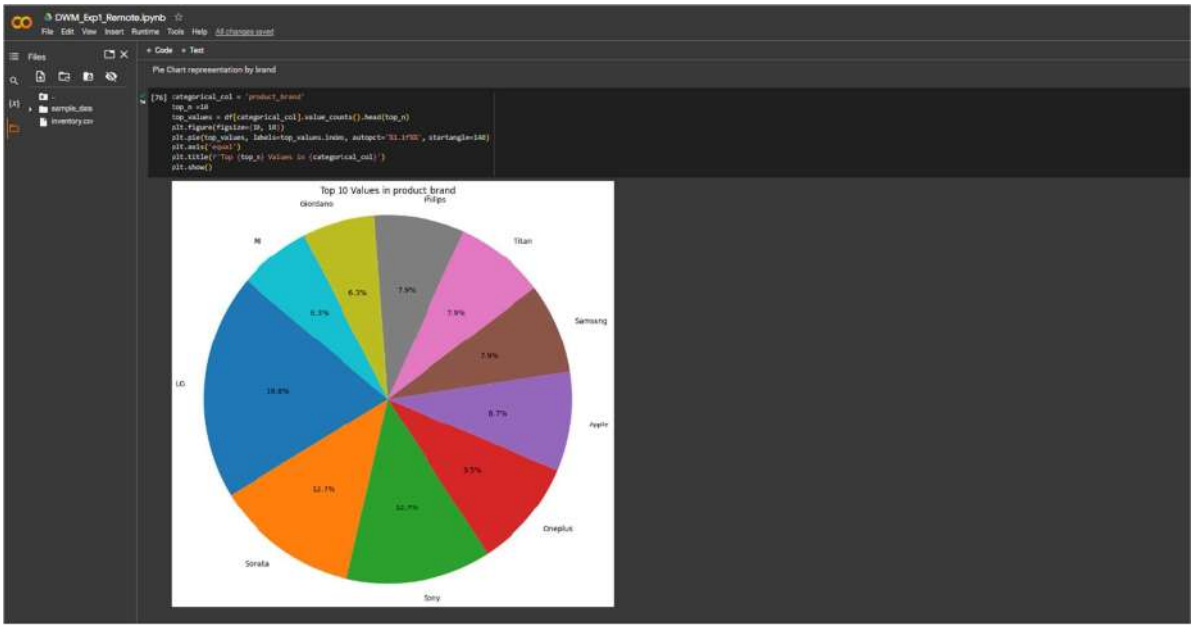

DWM_Exp1_Remote.ipynb
Attribute Name: transaction_id
Count of Values: 200
Minimum Value: 1
Maximum Value: 200
Data Type: int64
Range: 199
Quartiles:
0.25 50.75
0.50 100.50
0.75 150.25
Name: transaction_id, dtype: float64
Percentiles:
0.1 20.9
0.9 180.1
Name: transaction_id, dtype: float64
Box Plot for transaction_id
1
0 25 50 75 100 125 150 175 200
completed at 15:57

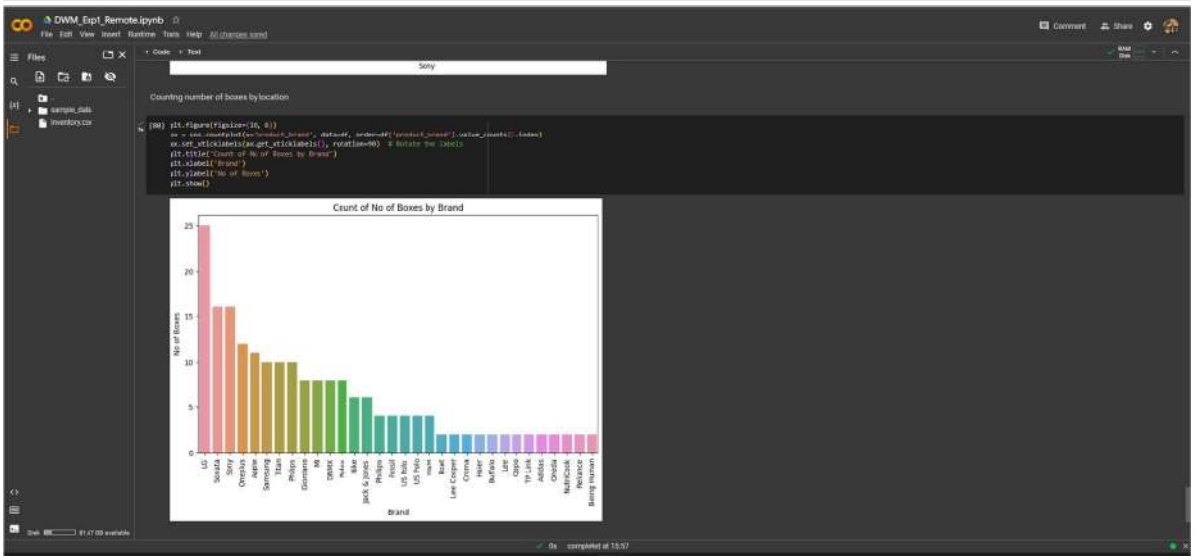

DWM_Exp1_Remote.ipynb
0 25 50 75 100 125 150 175 200
Outliers:
Series([], Name: transaction_id, dtype: int64)
Attribute Name: number_of_boxes
Count of Values: 200
Minimum Value: 1
Maximum Value: 99
Data Type: int64
Range: 98
Quartiles:
0.25 25.0
0.50 51.5
0.75 76.0
Name: number_of_boxes, dtype: float64
Percentiles:
0.1 11.0
0.9 90.0
Name: number_of_boxes, dtype: float64
Box Plot for number_of_boxes
1
completed at 15:57


DWM_Exp1_Remote.ipynb
Histograms for numerical columns
Histograms of Numerical Columns
transaction_id
number_of_boxes


DWM_Exp1_Remote.ipynb
Scatter Plot
warehouse_location
number_of_boxes
Mumbai
Delhi
Bangalore
Kerela
Gurgaon
Hyderabad
Histogram with product_brand vs Number of boxes


DWM_Exp1_Remote.ipynb
Histogram with product_brand vs Number of boxes
Histogram of Mean Prices by Location
Frequency
Mean Price
Pie Chart representation by brand


DWM_Exp1_Remote.ipynb
Pie Chart repreeentation by brand
Top 10 Values in product brand
Giordano
Philips
Titan
MI
Samsung
LG
Apple
Oneplus
Sonata
Sony


DWM_Exp1_Remote.ipynb
Sony
Counting number of boxes by location
Count of No of Boxes by Brand
No of Boxes
Brand

EXPERIMENT - 02

Aim: One case study on building Data warehouse/Data Mart.

Theory:

Difference between the two terms – DBMS & Data Warehouse

A Database Management System (DBMS) stores data in the form of tables, uses ER model and the goal is ACID properties. For example, a DBMS of college has tables for students, faculty, etc.

A Data Warehouse is separate from DBMS, it stores a huge amount of data, which is typically collected from multiple heterogeneous sources like files, DBMS, etc. The goal is to produce statistical results that may help in decision makings. For example, a college might want to see quick different results, like how the placement of CS students has improved over the last 10 years, in terms of salaries, counts, etc.

Benefits of Data Warehouse

1. Better business analytics: Data warehouse plays an important role in every business to store and analysis of all the past data and records of the company. which can further increase the understanding or analysis of data to the company.

2. Faster Queries: Data warehouse is designed to handle large queries that's why it runs queries faster than the database.

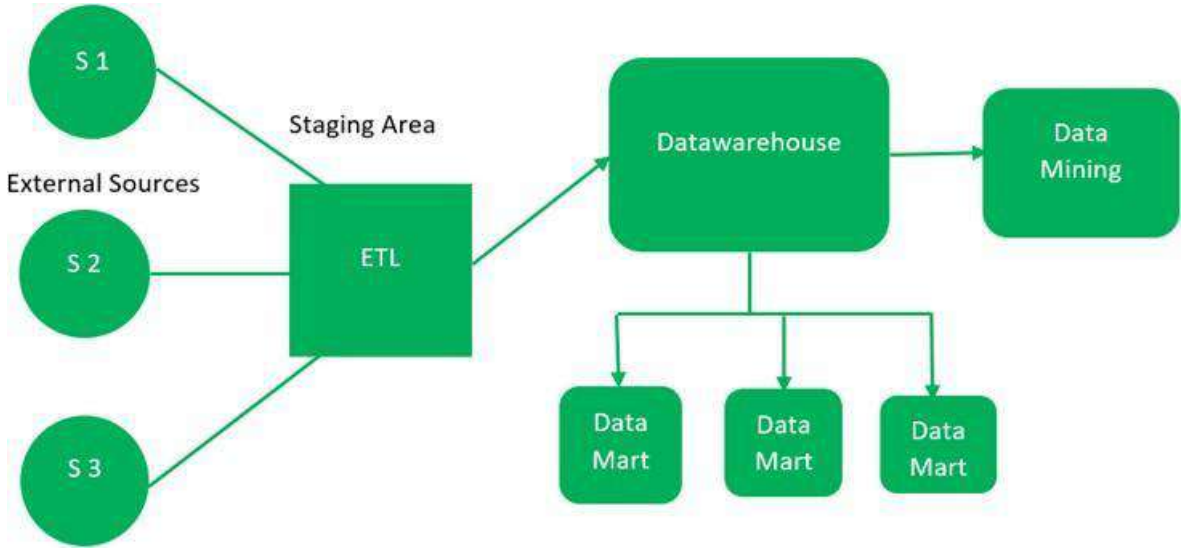
3. Improved data Quality: In the data warehouse the data you gathered from different sources is being stored and analyzed it does not interfere with or add data by itself so your quality of data is maintained and if you get any issue regarding data quality then the data warehouse team will solve this.

4. Historical Insight: The warehouse stores all your historical data which contains details about the business so that one can analyze it at any time and extract insights from it

Construction of a Data Warehouse

There are 2 approaches for constructing data-warehouse: Top-down approach and Bottom-up approach are explained as below.

A. Top-Down Approach:



The essential components are discussed below:

1. External Sources –

External source is a source from where data is collected irrespective of the type of data. Data can be structured, semi structured and unstructured as well.

2. Stage Area –

Since the data, extracted from the external sources does not follow a particular format, so there is a need to validate this data to load into data warehouse. For this purpose, it is recommended to use ETL tool.

3. E(Extracted): Data is extracted from External data source.

4. T(Transform): Data is transformed into the standard format.

5. L(Load): Data is loaded into data warehouse after transforming it into the standard format.

6. Data-warehouse –

After cleansing of data, it is stored in the data warehouse as central repository. It actually stores the meta data and the actual data gets stored in the data marts. Note that data warehouse stores the data in its purest form in this top-down approach.

7. Data Marts –

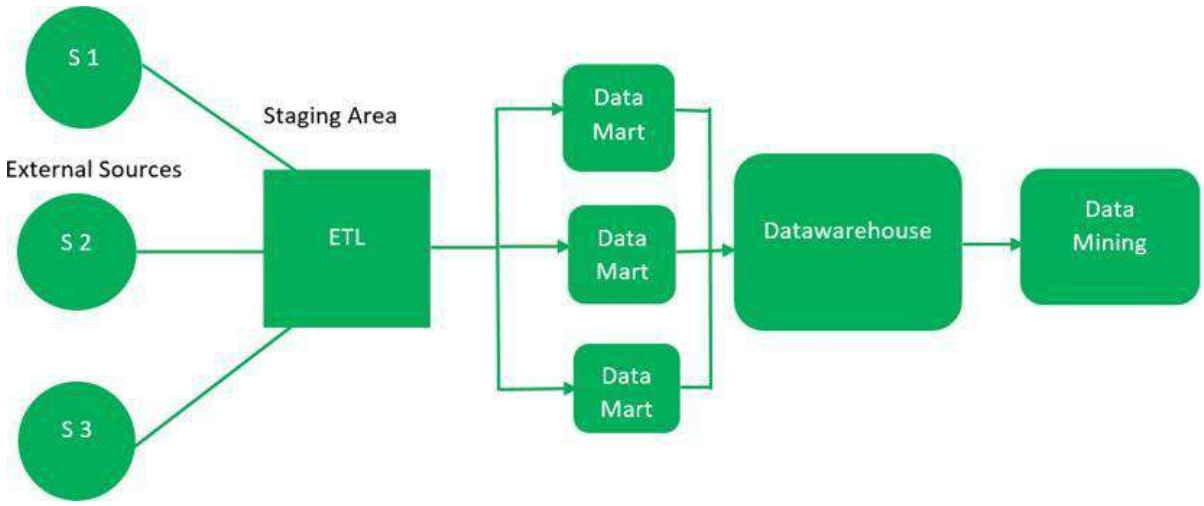
Data mart is also a part of storage component. It stores the information of a particular function of an organisation which is handled by single authority. There can be as many number of data marts in an organisation depending upon the functions. We can also say that data mart contains subset of the data stored in data warehouse.

8. Data Mining –

The practice of analysing the big data present in data warehouse is data mining. It is used to find the hidden patterns that are present in the database or in data warehouse with the help of algorithm of data mining.

This approach is defined by Inmon as – data warehouse as a central repository for the complete organisation and data marts are created from it after the complete data warehouse has been created.

B. Bottom-Up Approach:



1. First, the data is extracted from external sources (same as happens in top-down approach).

2. Then, the data go through the staging area (as explained above) and loaded into data marts instead of data warehouse. The data marts are created first and provide reporting capability. It addresses a single business area.

3. These data marts are then integrated into data warehouse.

This approach is given by Kinball as – data marts are created first and provides a thin view for analyses and data warehouse is created after complete data marts have been created.

Fact Table

- In a data warehouse, a fact table is a table that stores the measurements, metrics, or facts related to a business operation.
- It is located at the centre of a star or snowflake schema and is surrounded by dimension tables.
- When multiple fact tables are used, they can be organized using a "fact constellation schema."
- A fact table has two types of columns: those that contain the facts and those that serve as foreign keys linking to dimension tables.
- The primary key of a fact table is often a composite key made up of all of the foreign keys in the table.
- Fact tables can hold various types of measurements, such as additive, non-additive, and partly additive measures, and store important information in the data warehouse.
- They are useful for evaluating dimensional attributes because they provide additive values that can act as independent variables.

Dimension Table

- Dimension tables contain descriptions of the objects in a fact table and provide information about dimensions such as values, characteristics, and keys.
- These tables are usually small, with a number of rows ranging from a few hundred to a few thousand.
- The term "dimension table" refers to a set of data related to any quantifiable event and is the foundation for dimensional modelling.

- Dimension tables have a column that serves as a primary key, allowing each dimension row or record to be uniquely identified. This key is used to link the dimension table to the fact tables. A surrogate key, which is a system-generated key, is often used to uniquely identify the rows in the dimension table.

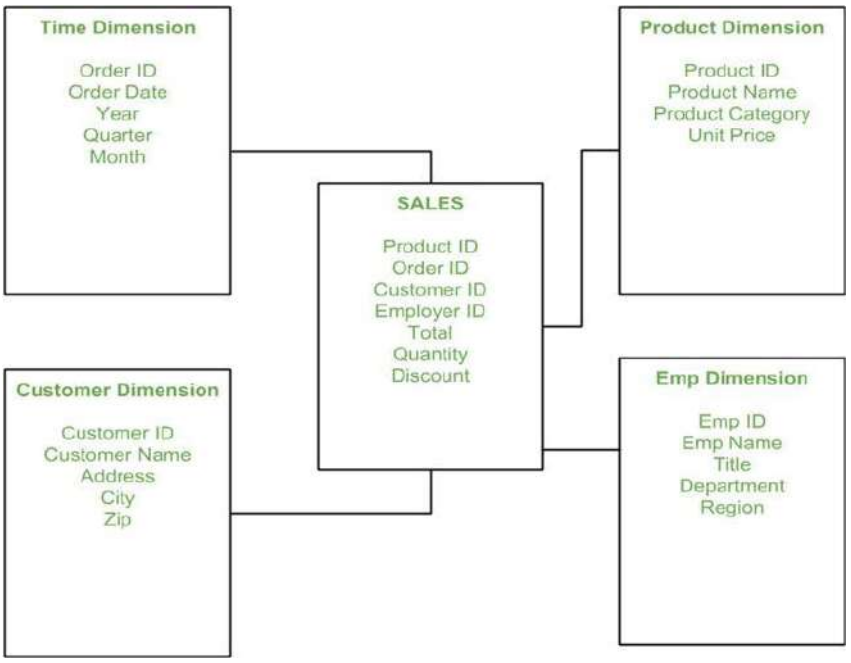
Star Schema

A star schema is a type of data modelling technique used in data warehousing to represent data in a structured and intuitive way. In a star schema, data is organized into a central fact table that contains the measures of interest, surrounded by dimension tables that describe the attributes of the measures.

The fact table in a star schema contains the measures or metrics that are of interest to the user or organization. For example, in a sales data warehouse, the fact table might contain sales revenue, units sold, and profit margins. Each record in the fact table represents a specific event or transaction, such as a sale or order.

The dimension tables in a star schema contain the descriptive attributes of the measures in the fact table. These attributes are used to slice and dice the data in the fact table, allowing users to analyse the data from different perspectives. For example, in a sales data warehouse, the dimension tables might include product, customer, time, and location.

In a star schema, each dimension table is joined to the fact table through a foreign key relationship. This allows users to query the data in the fact table using attributes from the dimension tables.



Advantages of Star Schema

1. Simpler Queries – Join logic of star schema is quite cinch in comparison to other join logic which are needed to fetch data from a transactional schema that is highly normalized.

2. Simplified Business Reporting Logic – In comparison to a transactional schema that is highly normalized, the star schema makes simpler common business reporting logic, such as of reporting and period-over-period.

3. Feeding Cubes – Star schema is widely used by all OLAP systems to design OLAP cubes efficiently. In fact, major OLAP systems deliver a ROLAP mode of operation which can use a star schema as a source without designing a cube structure.

Disadvantages of Star Schema

1. Data integrity is not enforced well since in a highly de-normalized schema state.

2. Not flexible in terms if analytical needs as a normalized data model.

3. Star schemas don't reinforce many-to-many relationships within business entities – at least not frequently.

Snowflake Schema

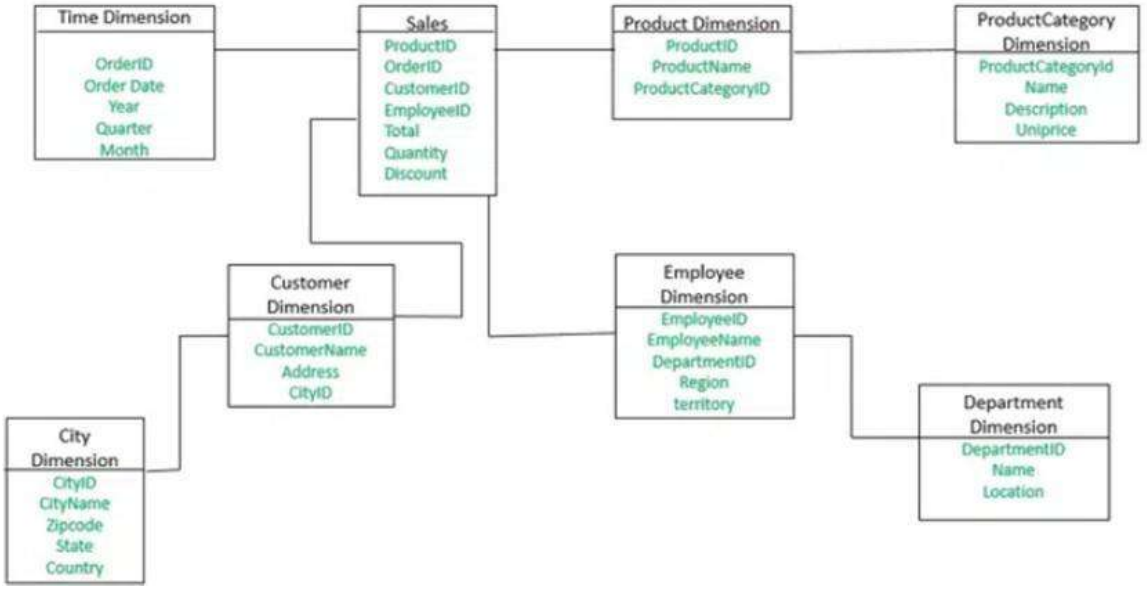
The snowflake schema is a variant of the star schema. Here, the centralized fact table is connected to multiple dimensions. In the snowflake schema, dimensions are present in a normalized form in multiple related tables. The snowflake structure materialized when the dimensions of a star schema are detailed and highly structured, having several levels of relationship, and the child tables have multiple parent tables. The snowflake effect affects only the dimension tables and does not affect the fact tables.

In a snowflake schema, the dimension tables are normalized into multiple related tables, creating a hierarchical or "snowflake" structure.

In a snowflake schema, the fact table is still located at the centre of the schema, surrounded by the dimension tables. However, each dimension table is further broken down into multiple related tables, creating a hierarchical structure that resembles a snowflake.

The Employee dimension table now contains the attributes: EmployeeID, EmployeeName, DepartmentID, Region, and Territory. The DepartmentID attribute links with the Employee table with the Department dimension table. The Department dimension is used to provide detail about each department, such as the Name and Location of the department.

The Customer dimension table now contains the attributes: CustomerID, CustomerName, Address, and CityID. The CityID attributes link the Customer dimension table with the City dimension table. The City dimension table has details about each city such as city name, Zipcode, State, and Country.



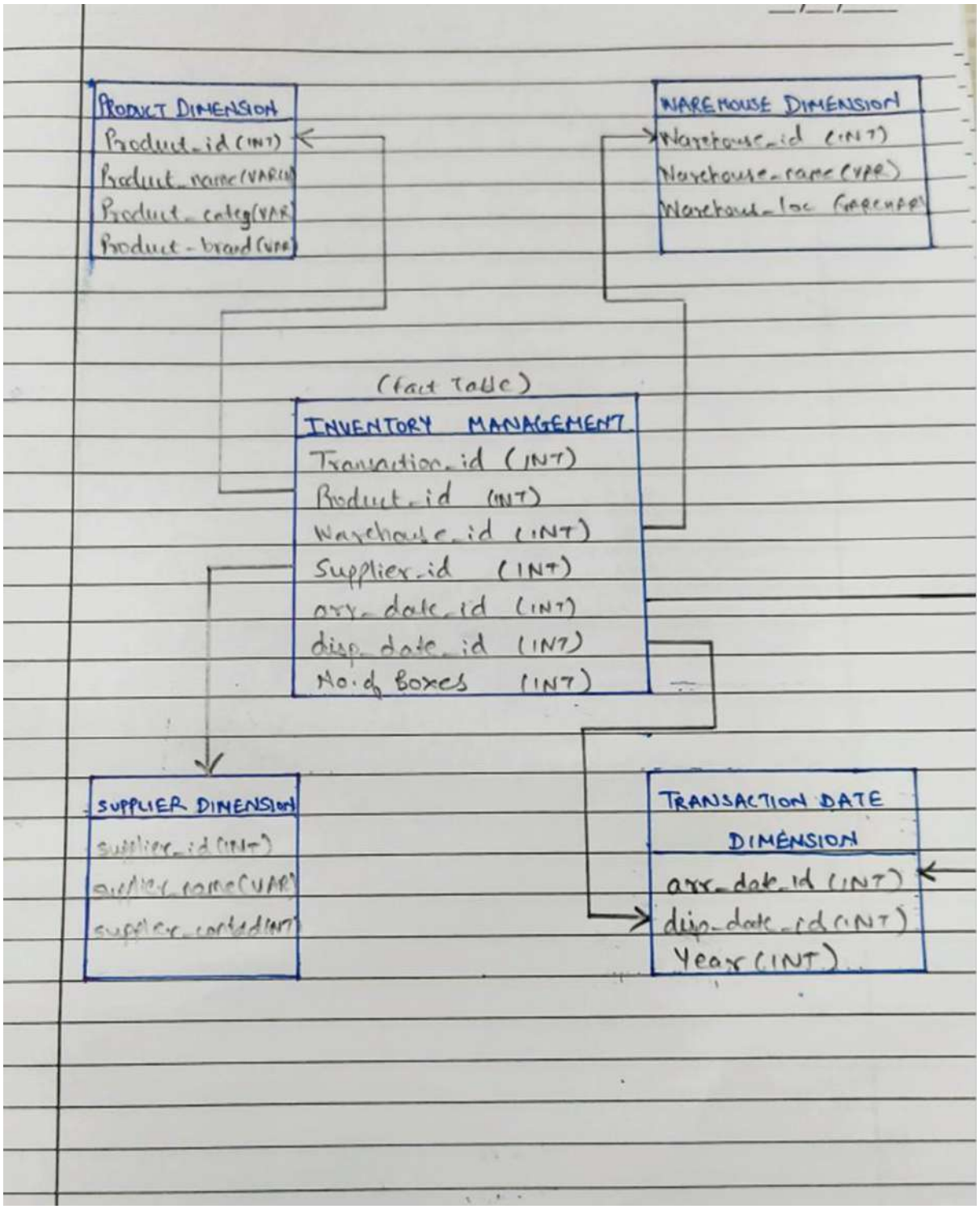
Advantages of Snowflake Schema

1. It provides structured data which reduces the problem of data integrity.
2. It uses small disk space because data are highly structured.

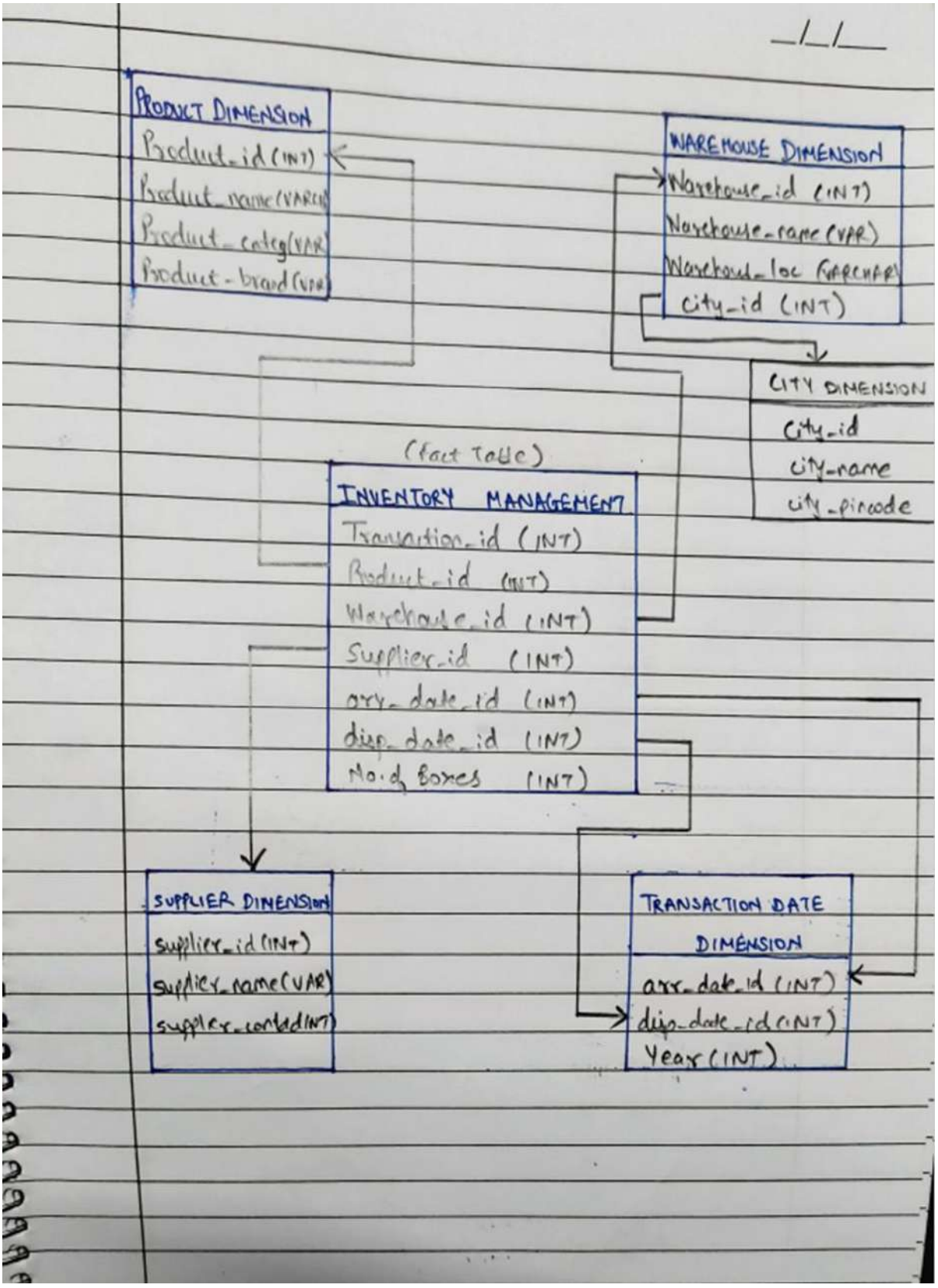
Disadvantages of Snowflake Schema

1. Snowflaking reduces space consumed by dimension tables but compared with the entire data warehouse the saving is usually insignificant.
2. Avoid snowflaking or normalization of a dimension table, unless required and appropriate.
3. Do not snowflake hierarchies of dimension table into separate tables. Hierarchies should belong to the dimension table only and should never be snowflakes.
4. Multiple hierarchies that can belong to the same dimension have been designed at the lowest possible detail.

Star Schema


PRODUCT DIMENSION
Product_id (INT)
Product_name (VARCHAR)
Product_categ (VARCHAR)
Product_brand (VARCHAR)
WAREHOUSE DIMENSION
Warehouse_id (INT)
Warehouse_name (VARCHAR)
Warehouse_loc (VARCHAR)
(Fact Table)
INVENTORY MANAGEMENT
Transaction_id (INT)
Product_id (INT)
Warehouse_id (INT)
Supplier_id (INT)
arr_date_id (INT)
disp_date_id (INT)
No. of Boxes (INT)
SUPPLIER DIMENSION
supplier_id (INT)
supplier_name (VARCHAR)
supplier_contact (INT)
TRANSACTION DATE DIMENSION
arr_date_id (INT)
disp_date_id (INT)
Year (INT)

Snowflake Schema



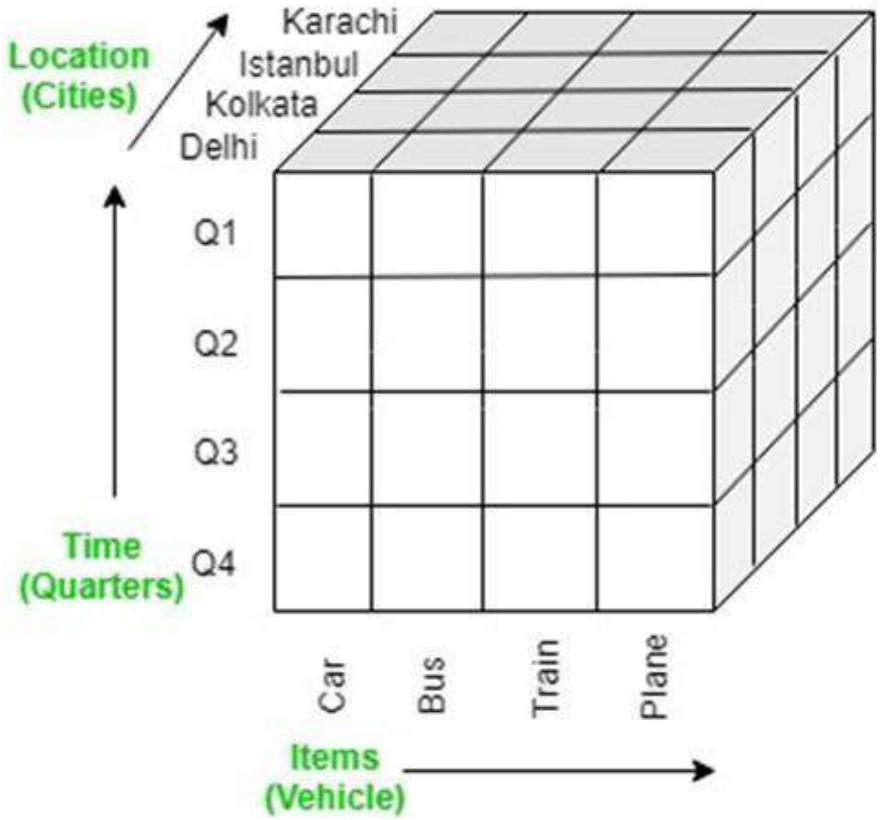
EXPERIMENT - 03

Aim: Implementation of all dimension tables and fact tables and OLAP (in Oracle)

Theory:

In the multidimensional model, the records are organized into various dimensions, and each dimension includes multiple levels of abstraction described by concept hierarchies. This organization support users with the flexibility to view data from various perspectives. Several OLAP data cube operations exist to demonstrate these different views, allowing interactive queries and a search of the record at hand. Hence, OLAP supports a user-friendly environment for interactive data analysis.

Consider the OLAP operations which are to be performed on multidimensional data. The figure shows data cubes for sales of a shop. The cube contains the dimensions, location, time and item, where the location is aggregated about city values, time is aggregated concerning quarters, and an item is aggregated concerning item types.



OLAP operations:

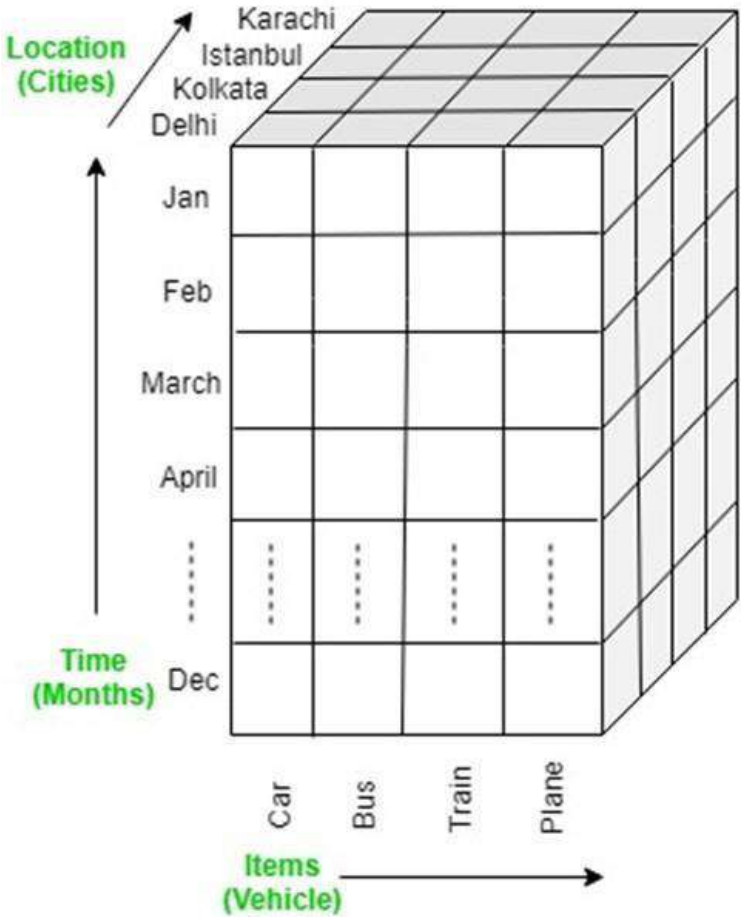
Five basic analytical operations can be performed on an OLAP cube:

1. Drill down:

   In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:

   - Moving down in the concept hierarchy
   - Adding a new dimension

In the cube given in the overview section, the drill-down operation is performed by moving down in the concept hierarchy of the Time dimension (Quarter -> Month).

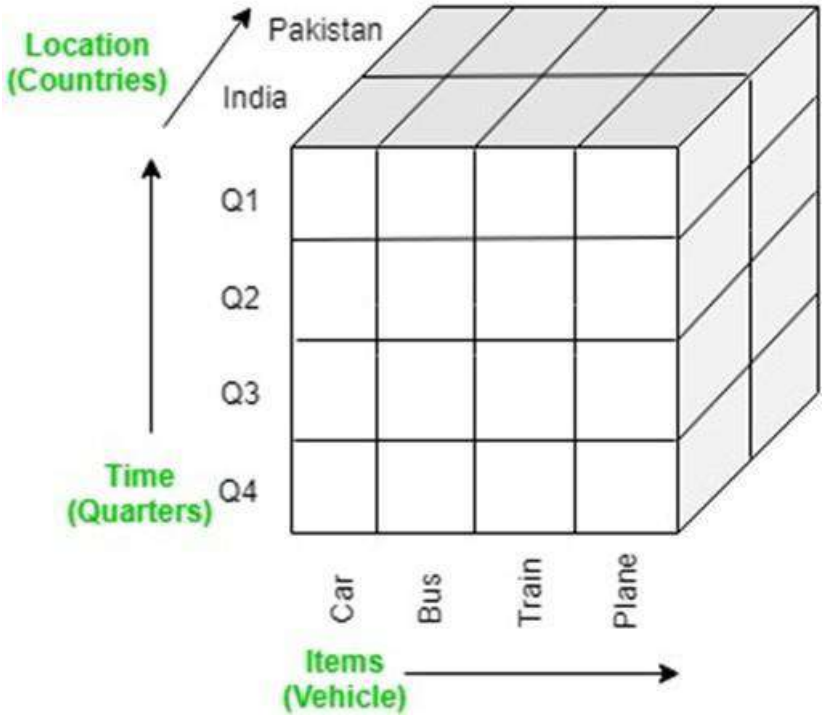


2. Roll up:

   It is just the opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:

   - Climbing up in the concept hierarchy
   - Reducing the dimensions

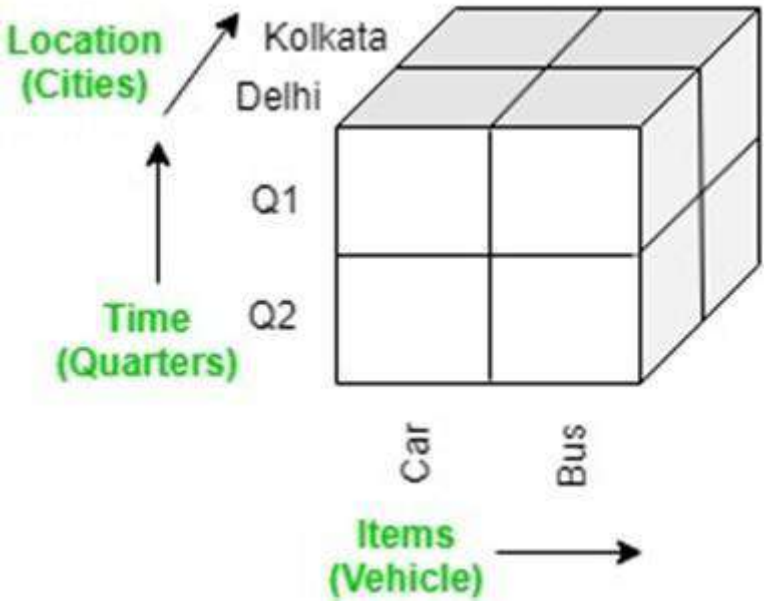
In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of Location dimension (City -> Country).



3. Dice:

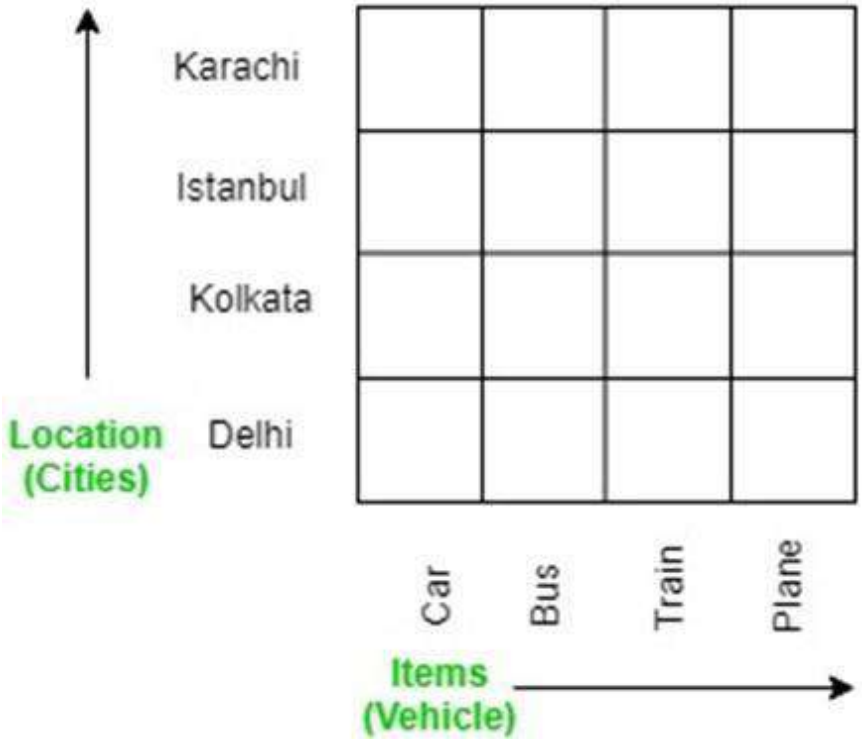
It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting the following dimensions with criteria:

- Location = “Delhi” or “Kolkata”
- Time = “Q1” or “Q2”
- Item = “Car” or “Bus”



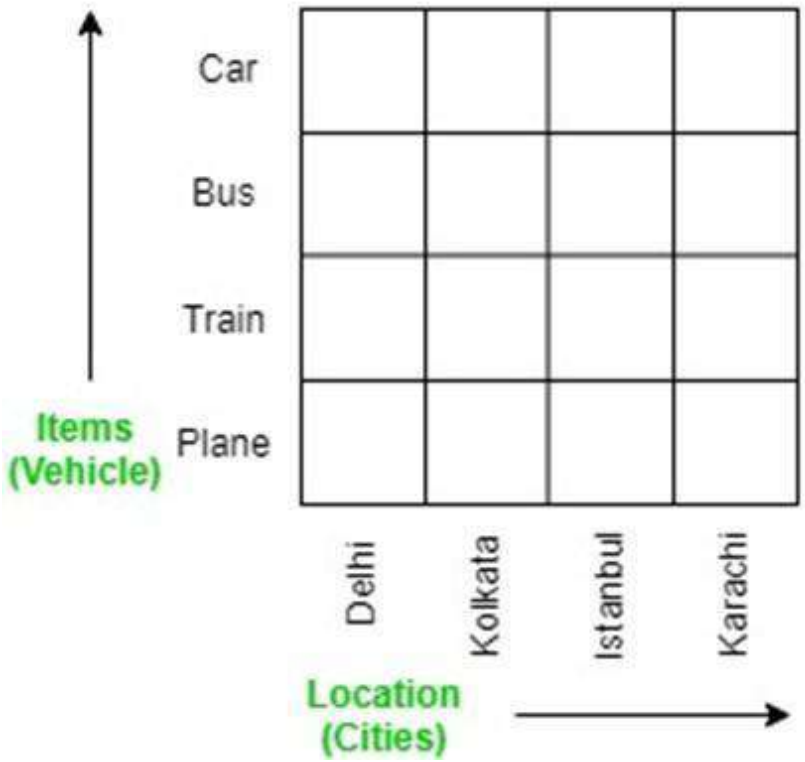
4. Slice:

It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = "Q1".



5. Pivot:

It is also known as rotation operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing the pivot operation gives a new view of it.



Code:

Creating Fact Table and Dimension tables.

DIMENSIONS TABLE – CITY DIMENSIONS

```
CREATE TABLE city_dim (
   city_id INT PRIMARY KEY,
   city_name VARCHAR(255),
   c_state VARCHAR(255),
   pincode VARCHAR(10)
);
```

DIMENSIONS TABLE – CONTACT DIMENSIONS

```
CREATE TABLE contact_dim (
   contact_id VARCHAR(255) PRIMARY
   KEY, phone VARCHAR(15),
   email VARCHAR(255)
);
```

DIMENSIONS TABLE – SUPPLIER DIMENSIONS

```
CREATE TABLE supplier_dim (
   supplier_id VARCHAR(255) PRIMARY
   KEY, supplier_name VARCHAR(255),
   contact_id INT
);
```

DIMENSIONS TABLE – PRODUCT DIMENSIONS

```
CREATE TABLE product_dim (
   product_id VARCHAR(255) PRIMARY
   KEY, product_name VARCHAR(255),
   product_category VARCHAR(255),
   product_brand VARCHAR(255)
);
```

DIMENSIONS TABLE – TRANSACTION_DATE DIMENSIONS

```
CREATE TABLE transaction_date_dim
   ( arrival_date DATE,
   dispatch_date DATE,
   duration_in_warehouse
   INT
);
```

DIMENSIONS TABLE – WAREHOUSE DIMENSIONS

```
CREATE TABLE warehouse_dim (
   warehouse_id VARCHAR(255) PRIMARY KEY,
   warehouse_name VARCHAR(255),
   warehouse_location VARCHAR(255),
   city_id INT
);
```

FACT TABLE - INVENTORY

```
CREATE TABLE inventory_management_fact
   ( transaction_id INT PRIMARY KEY,
   product_id VARCHAR(255),
   warehouse_id VARCHAR(255),
```

```
    supplier_id
    VARCHAR(255),
    arrival_date_id DATE,
    dispatch_date_id DATE,
    number_of_boxes INT,
    FOREIGN KEY (product_id) REFERENCES product_dim(product_id),
    FOREIGN KEY (warehouse_id) REFERENCES
    warehouse_dim(warehouse_id), FOREIGN KEY (supplier_id) REFERENCES
    supplier_dim(supplier_id)
);
```

Inserting Values in the above tables:

city_dimension:

```
INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (7597, 'Mumbai', 'Maharashtra',
'400001');
INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (7099, 'Bangalore', 'Karnataka', '560001');
INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (9293, 'Pune', 'Maharashtra', '411001');
INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (6336, 'Hyderabad', 'Telangana',
'500001');
INSERT INTO city_dim (city_id, city_name, c_state,
```

```
pincode) VALUES (5676, 'Gurgaon', 'Haryana', '122001');

INSERT INTO city_dim (city_id, city_name, c_state, pincode)
VALUES (5055, 'Delhi', 'Delhi', '110001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (7298, 'Kota', 'Rajasthan', '324001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (6244, 'Jaipur', 'Rajasthan', '302001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (4098, 'Chennai', 'Tamil Nadu', '600001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (7544, 'Kolkata', 'West Bengal', '700001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (6003, 'Ahmedabad', 'Gujarat', '380001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (8081, 'Lucknow', 'Uttar Pradesh',

'226001'); INSERT INTO city_dim (city_id, city_name,

c_state, pincode) VALUES (5210, 'Chandigarh',

'Chandigarh', '160001'); INSERT INTO city_dim (city_id,

city_name, c_state, pincode) VALUES (7142, 'Bhopal',

'Madhya Pradesh', '462001'); INSERT INTO city_dim

(city_id, city_name, c_state, pincode) VALUES (6001,

'Patna', 'Bihar', '800001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (9012, 'Guwahati', 'Assam', '781001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (7659, 'Trivandrum', 'Kerala', '695001');
```

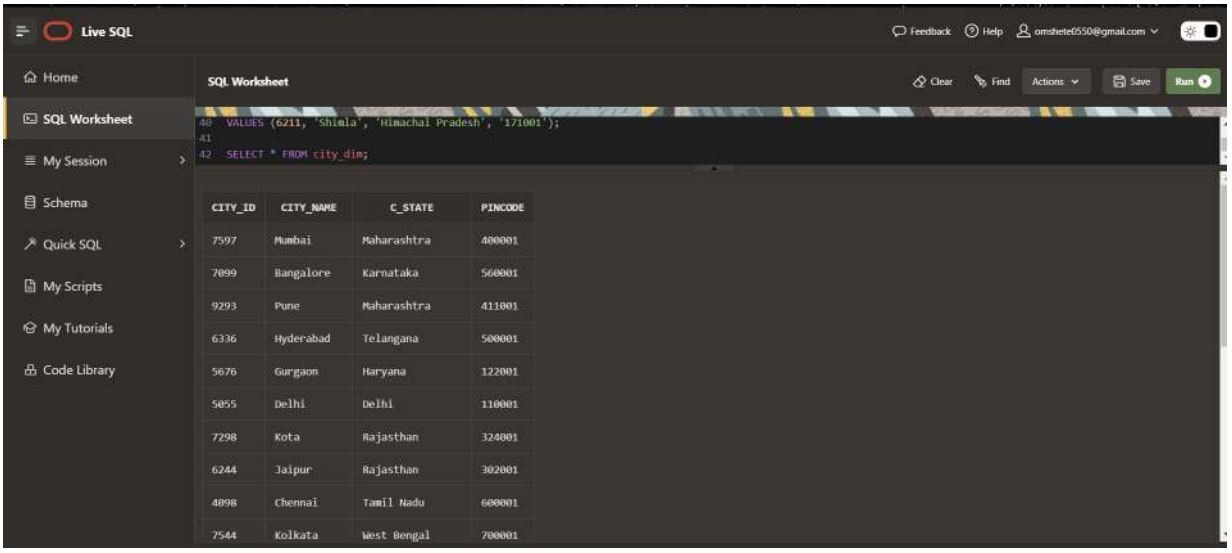
INSERT INTO city_dim (city_id, city_name, c_state, pincode)
VALUES (7245, 'Bhubaneswar', 'Odisha', '751001');

INSERT INTO city_dim (city_id, city_name, c_state,

pincode) VALUES (5222, 'Dehradun', 'Uttarakhand',

'248001'); INSERT INTO city_dim (city_id, city_name,

c_state, pincode) VALUES (6211, 'Shimla', 'Himachal



Pradesh', '171001');

contact_dim:

INSERT INTO contact_dim (contact_id, phone, email)

VALUES ('C2483', '9988776655',

'xbarrett@example.com'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C8454', '9876543210',

'vhernandez@example.com'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C8265', '8899776644',

'cooperdavid@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C3118', '7766554433',

'monica66@example.net'); INSERT INTO contact_dim

(contact_id, phone, email) VALUES ('C7283', '9870123456',

'krivera@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C7667', '9876012345',

'rosarioalbert@example.net'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C1342', '8765432109', 'leejennifer@example.net');

INSERT INTO contact_dim (contact_id, phone, email)

VALUES ('C3989', '8877665544',

'breynolds@example.com'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C1699', '9988001122',

'thompsonzachary@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C5700', '9012345678',

'robert84@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C9238', '9878998899',

'hoffmanbeth@example.net'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C7601', '9988778899', 'sdavis@example.net');

INSERT INTO contact_dim (contact_id, phone, email)

VALUES ('C5895', '9123456789', 'kristina32@example.net');

INSERT INTO contact_dim (contact_id, phone, email)

VALUES ('C9589', '8899889988', 'natalie47@example.org');

INSERT INTO contact_dim (contact_id, phone, email)

VALUES ('C1627', '9900990099',

'longashley@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C5607', '8765432101',

'coreywilson@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C9300', '8901234567',

'ebryan@example.com'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C8900', '9876543211',

'burkesharon@example.net'); INSERT INTO contact_dim

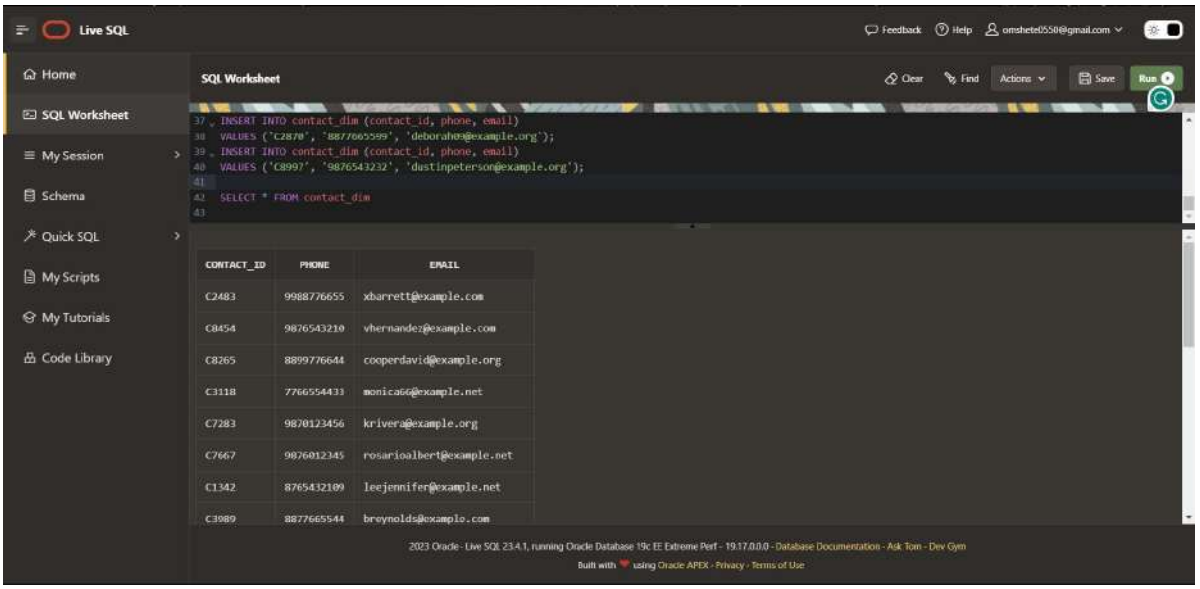
(contact_id, phone, email)

VALUES ('C2870', '8877665599',

'deborah09@example.org'); INSERT INTO contact_dim

(contact_id, phone, email)

VALUES ('C8997', '9876543232', 'dustinpeterson@example.org');



supplier_dimension:

```
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S3535', 'Stewart, Clayton and Martinez',
'C2483');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S7973', 'Carroll, Brady and Hancock', 'C8454');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S2142', 'Singh-Johnson', 'C8265');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S3115', 'Boyer Ltd', 'C3118');
INSERT INTO supplier_dim (supplier_id, supplier_name, contact_id)
VALUES ('S8922', 'Davis Ltd', 'C7283');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S4540', 'Sandoval, Brown and Peters',
'C7667');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S2603', 'Madden Inc', 'C1342');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S3568', 'Pearson, Wall and Shelton', 'C3989');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S8035', 'Ellis Ltd', 'C1699');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S3141', 'Johnson Group', 'C5700');
INSERT INTO supplier_dim (supplier_id, supplier_name,
contact_id) VALUES ('S8420', 'Mccoy LLC', 'C9238');
INSERT INTO supplier_dim (supplier_id, supplier_name,
```

contact_id) VALUES ('S1561', 'Smith Ltd', 'C7601');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S8841', 'Small PLC', 'C5895');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S3895', 'Carney, Edwards and Chen', 'C9589');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S9366', 'Ruiz-Garcia', 'C1627');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S3534', 'Potter and Sons', 'C5607');

INSERT INTO supplier_dim (supplier_id, supplier_name, contact_id)
VALUES ('S7178', 'Rowe-Benitez', 'C9300');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S6126', 'Williamson, Ware and Martin',

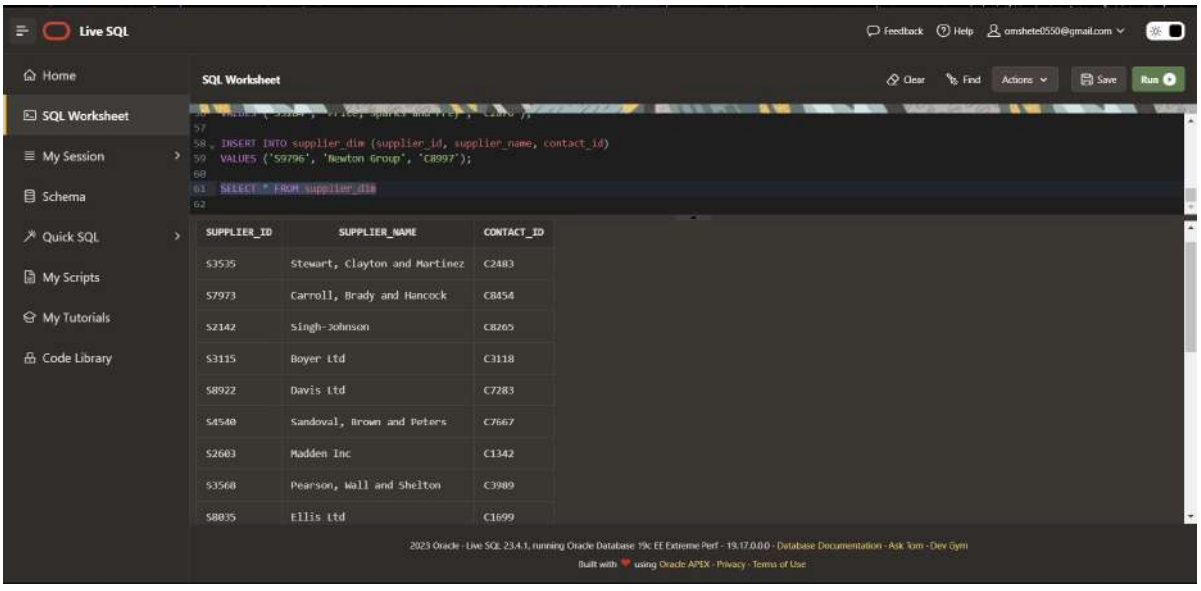
'C8900');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S3284', 'Price, Sparks and Frey', 'C2870');

INSERT INTO supplier_dim (supplier_id, supplier_name,

contact_id) VALUES ('S9796', 'Newton Group', 'C8997');



product_dimension:

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P2859', 'Titan T27', 'watches', 'Titan');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P5937', 'Samsung QLED 21', 'tv', 'Samsung');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P1553', 'Philips Iron I4', 'Electrical App', 'Philips');

INSERT INTO product_dim (product_id, product_name, product_category, product_brand) VALUES ('P7296', 'Nike Nex', 'Wear', 'Nike');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P9813', 'US Polo Black Denim', 'Wear', 'US Polo');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P5206', 'Samsung Washing Machine', 'Electrical App',

'Samsung');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P1489', 'Reliance Hair Dryer', 'Electrical App', 'Reliance');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P9351', 'LG 42inch P32', 'tv', 'LG');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P4413', 'Sony Bravia S76', 'tv', 'Sony');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P3825', 'Fossil F31', 'watches', 'Fossil');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P7873', 'Giordano G21', 'watches', 'Giordano');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P1596', 'MI MicroLED M31', 'tv', 'MI');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P2693', 'Oneplus O8', 'tv', 'Oneplus');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P4110', 'NutriCook Mixer G21', 'Electrical App', 'NutriCook');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P1235', 'DNMX Shirt S21', 'Wear', 'DNMX');

INSERT INTO product_dim (product_id, product_name, product_category, product_brand) VALUES ('P2301', 'DNMX Hoodie', 'Wear', 'DNMX');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P5882', 'Giordano G26', 'watches', 'Giordano');

INSERT INTO product_dim (product_id, product_name, product_category,

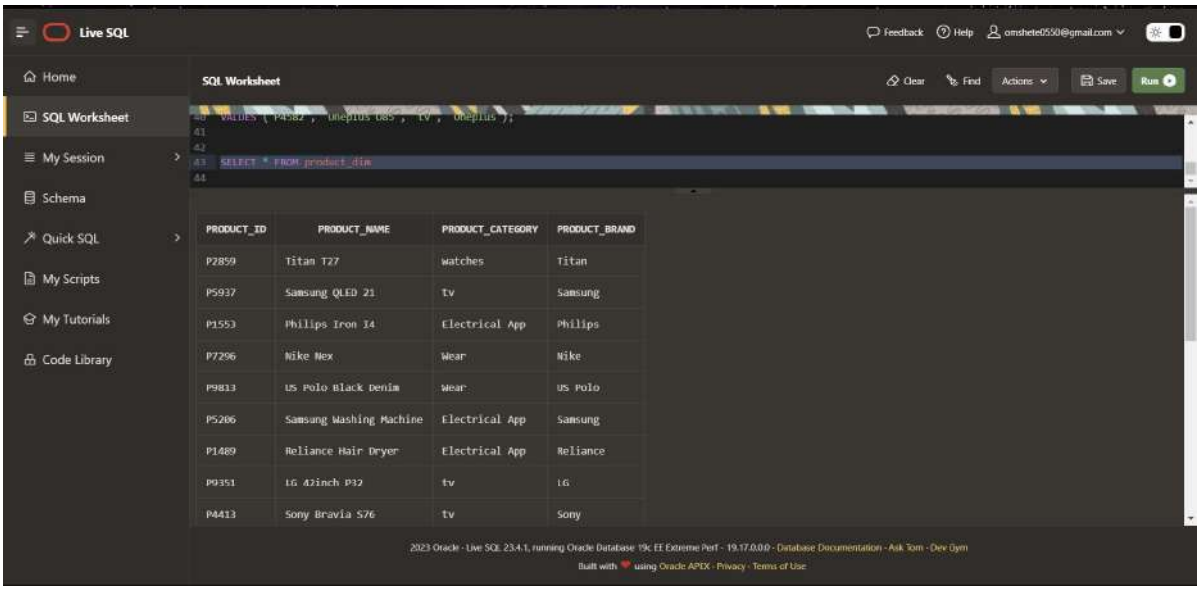
product_brand) VALUES ('P7062', 'Sonata Kid K2', 'watches', 'Sonata');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P8087', 'Oneplus O20', 'tv', 'Oneplus');

INSERT INTO product_dim (product_id, product_name, product_category,

product_brand) VALUES ('P4582', 'Oneplus O85', 'tv', 'Oneplus');

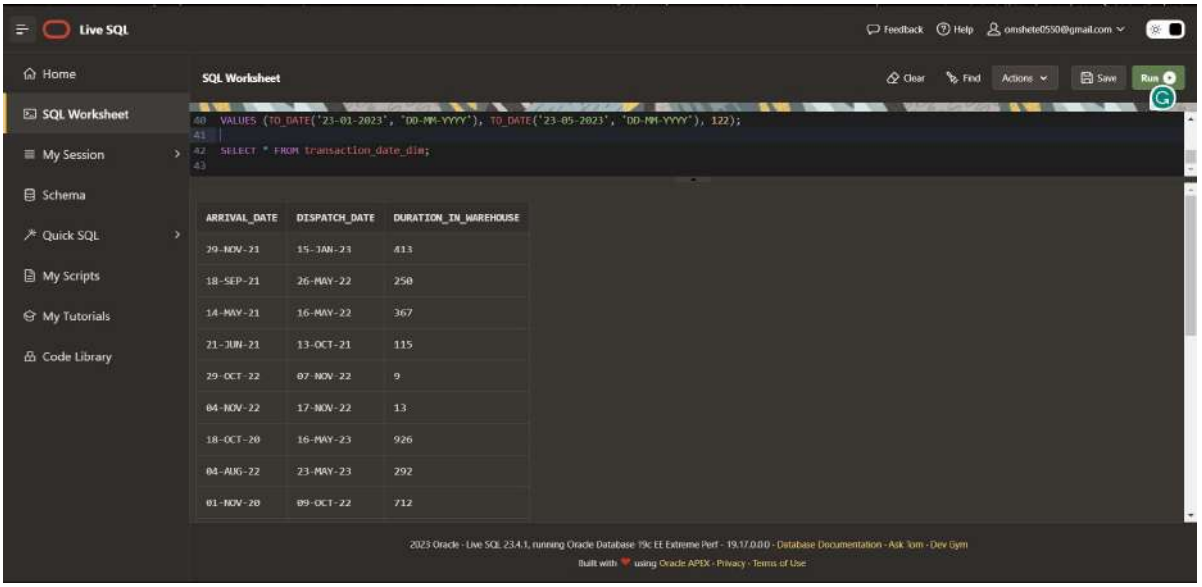


transaction_date_dimension:

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('29-11-2021', 'DD-MM-YYYY'), TO_DATE('15-01-2023',
'DD-MM-YYYY'), 413);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('18-09-2021', 'DD-MM-YYYY'), TO_DATE('26-05-2022',
'DD-MM-YYYY'), 250);


INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('14-05-2021', 'DD-MM-YYYY'), TO_DATE('16-05-2022',
'DD-MM-YYYY'), 367);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('21-06-2021', 'DD-MM-YYYY'), TO_DATE('13-10-2021',
'DD-MM-YYYY'), 115);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('29-10-2022', 'DD-MM-YYYY'), TO_DATE('07-11-2022', 'DD-MM-YYYY'),

9);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('04-11-2022', 'DD-MM-YYYY'), TO_DATE('17-11-2022',
'DD-MM-YYYY'), 13);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('18-10-2020', 'DD-MM-YYYY'), TO_DATE('16-05-2023',
'DD-MM-YYYY'), 926);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('04-08-2022', 'DD-MM-YYYY'), TO_DATE('23-05-2023',
'DD-MM-YYYY'), 292);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('01-11-2020', 'DD-MM-YYYY'), TO_DATE('09-10-2022',
'DD-MM-YYYY'), 712);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
```

```
VALUES (TO_DATE('11-09-2022', 'DD-MM-YYYY'), TO_DATE('20-01-2023',
'DD-MM-YYYY'), 131);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('15-03-2020', 'DD-MM-YYYY'), TO_DATE('08-05-2023',
'DD-MM-YYYY'), 1158);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('29-10-2022', 'DD-MM-YYYY'), TO_DATE('10-06-2023',
'DD-MM-YYYY'), 225);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('23-09-2022', 'DD-MM-YYYY'), TO_DATE('27-10-2022',
'DD-MM-YYYY'), 34);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('10-09-2020', 'DD-MM-YYYY'), TO_DATE('25-10-2022',
'DD-MM-YYYY'), 775);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('03-01-2020', 'DD-MM-YYYY'), TO_DATE('14-01-2023',
'DD-MM-YYYY'), 1093);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('03-04-2020', 'DD-MM-YYYY'), TO_DATE('17-05-2022',
'DD-MM-YYYY'), 765);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('29-10-2020', 'DD-MM-YYYY'), TO_DATE('27-11-2022',
'DD-MM-YYYY'), 753);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('20-06-2022', 'DD-MM-YYYY'), TO_DATE('24-06-2023',
'DD-MM-YYYY'), 369);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('23-06-2023', 'DD-MM-YYYY'), TO_DATE('28-06-2023', 'DD-MM-YYYY'),

5);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
```

VALUES (TO_DATE('23-01-2023', 'DD-MM-YYYY'), TO_DATE('23-05-2023', 'DD-MM-YYYY'), 122);



warehouse_dimension:

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W7276', 'BombayX', 'Mumbai', 7597);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W7001', 'DelhiCaps', 'Delhi', 7099);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W1296', 'BombayZ', 'Mumbai', 7597);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W8451', 'BangalorIS', 'Bangalore', 9293);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W8571', 'KerelaSwep', 'Kerela', 6336);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

```
VALUES ('W4968', 'GurgaonHDES', 'Gurgaon', 5676);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W1700', 'HyderabadHub', 'Hyderabad', 5055);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W4035', 'BombayX', 'Mumbai', 7597);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W1934', 'DelhiCaps', 'Delhi', 7099);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W2006', 'BombayZ', 'Mumbai', 7597);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W5878', 'BangalorIS', 'Bangalore', 9293);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W1169', 'KerelaSwep', 'Kerela', 6336);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W6639', 'GurgaonHDES', 'Gurgaon', 5676);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W4150', 'HyderabadHub', 'Hyderabad', 5055);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name,
warehouse_location, city_id)

VALUES ('W5472', 'BombayX', 'Mumbai', 7597);
```

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W8373', 'DelhiCaps', 'Delhi', 7099);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W3067', 'BombayZ', 'Mumbai', 7597);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

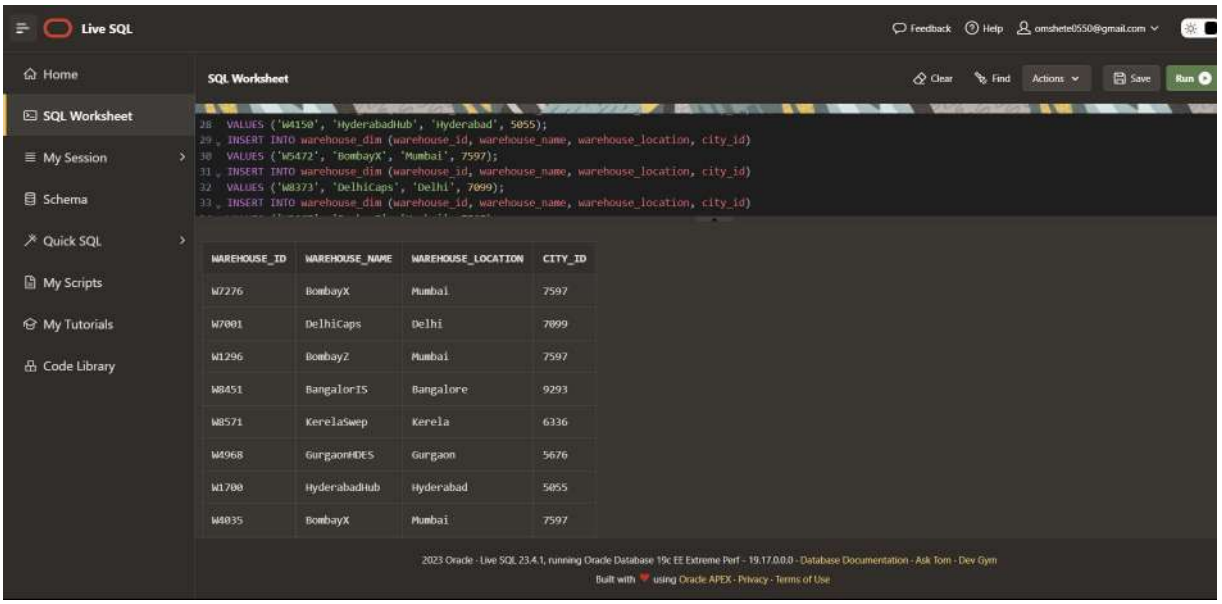
VALUES ('W1796', 'BangalorIS', 'Bangalore', 9293);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W4563', 'KerelaSwep', 'Kerela', 6336);

INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

VALUES ('W1452', 'GurgaonHDES', 'Gurgaon', 5676);

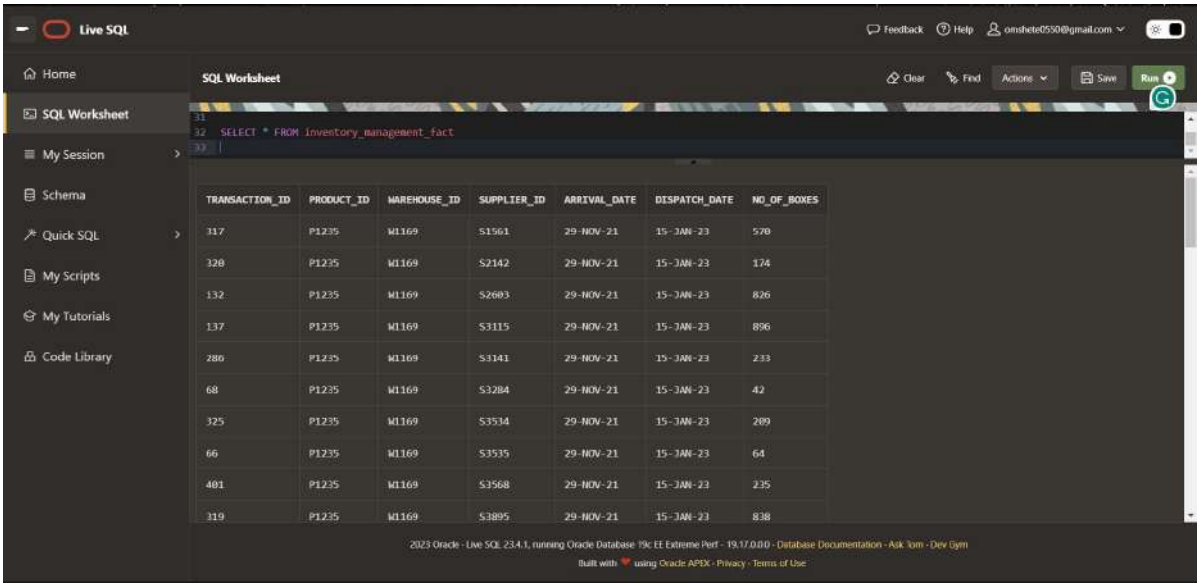


inventory_fact_table:

```
CREATE TABLE inventory_management_fact
AS SELECT
   1 AS
   transaction_id,
   p.product_id,
   w.warehouse_id,
   s.supplier_id,
   ad.arrival_date,
   ad.dispatch_date,
   0 AS no_of_boxes
FROM
   product_dim p
CROSS JOIN
   warehouse_dim
w CROSS JOIN
   supplier_dim s
CROSS JOIN
   transaction_date_dim ad;


Inserting random values for transaction_id and
no_of_boxes BEGIN
   FOR rec IN (SELECT rowid, product_id FROM inventory_management_fact) LOOP
      UPDATE inventory_management_fact
      SET
         transaction_id = ROUND(DBMS_RANDOM.VALUE(50, 500)),
```

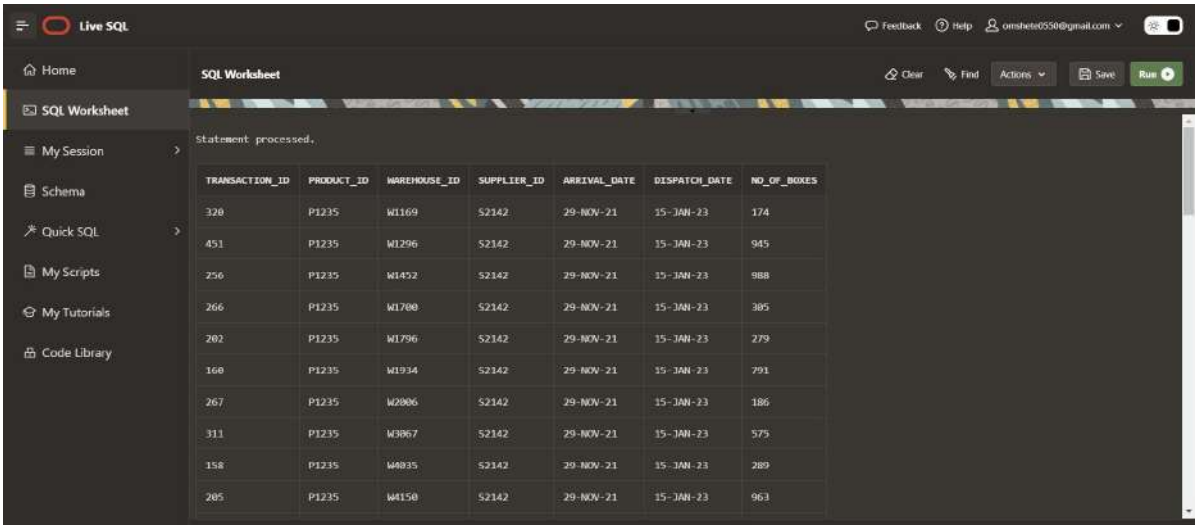
```
        no_of_boxes = ROUND(DBMS_RANDOM.VALUE(1, 1000))
    WHERE rowid =
  rec.rowid; END LOOP;
  COMMIT
; END;
/
```

SELECT * FROM inventory_management_fact



1. **Slice:**

```
CREATE MATERIALIZED VIEW mv_slice1
AS SELECT *
FROM
inventory_management_fact
WHERE SUPPLIER_ID = 'S2142';
SELECT * FROM mv_slice1;
```



2. Dice

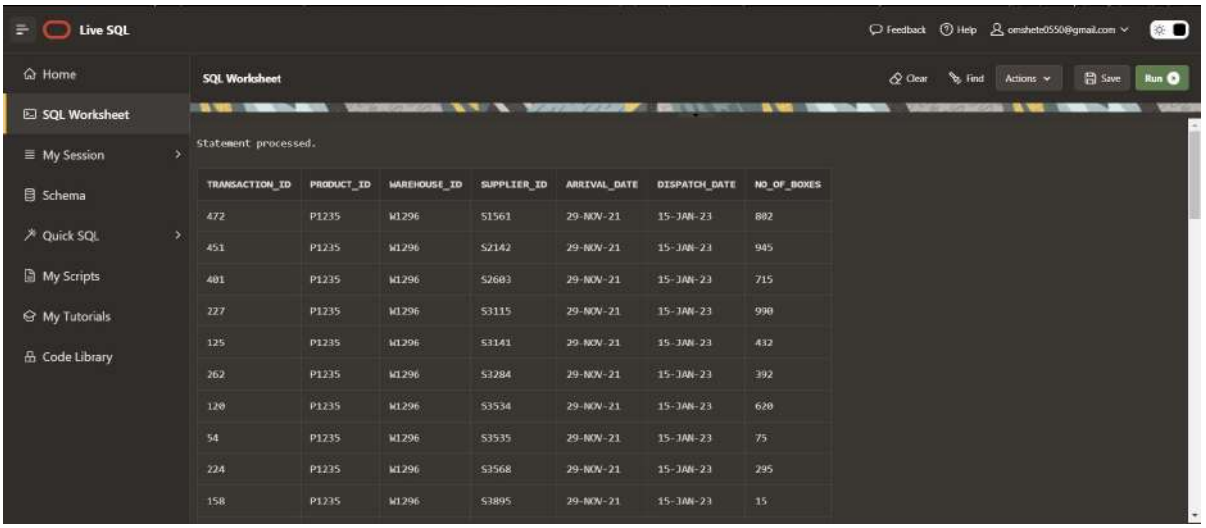
CREATE MATERIALIZED VIEW mv_dice

AS SELECT *

FROM inventory_management_fact

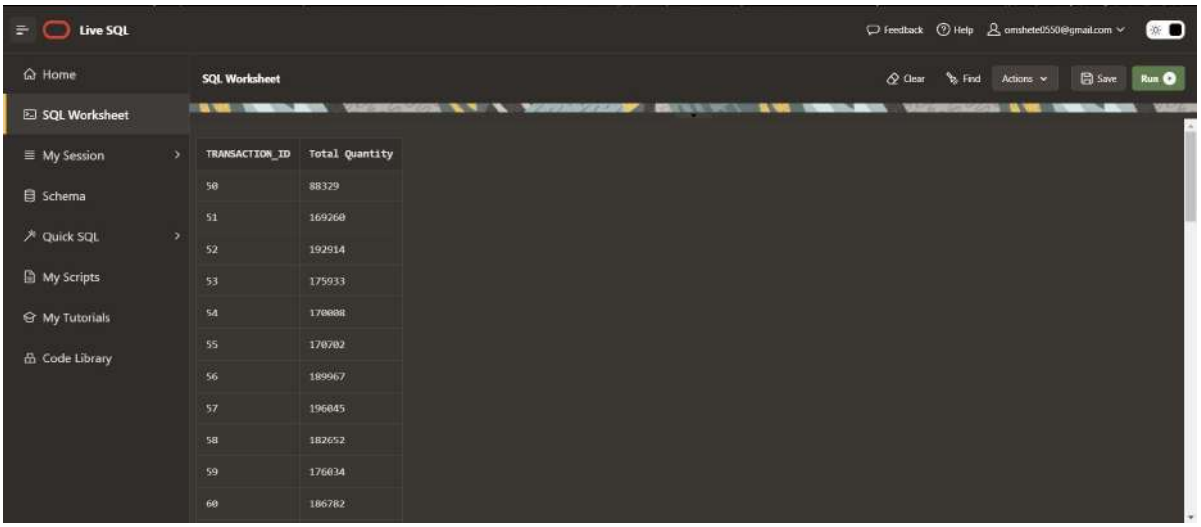
WHERE WAREHOUSE_ID = 'W1296' AND PRODUCT_ID = 'P1235';

SELECT * FROM mv_dice;



3. Rollup

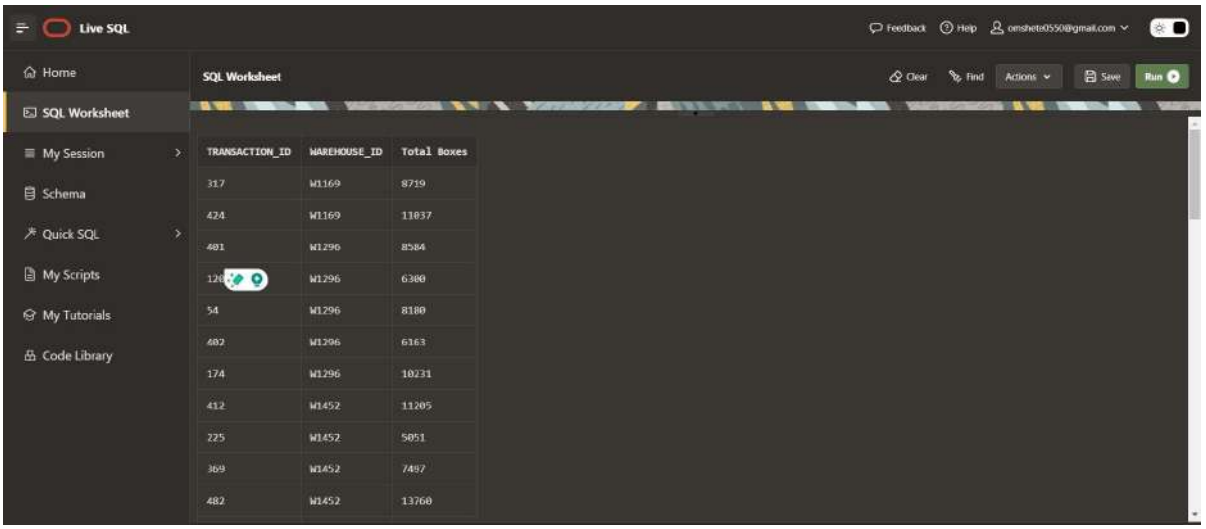
```
CREATE OR REPLACE VIEW vw_rollup
AS SELECT
  CASE
    WHEN GROUPING(TRANSACTION_ID) = 317 THEN 'Grand
    Total' ELSE TO_CHAR(TRANSACTION_ID)
  END AS TRANSACTION_ID,
  SUM(NO_OF_BOXES) AS "Total
Quantity" FROM
inventory_management_fact GROUP BY
ROLLUP(TRANSACTION_ID);

CREATE MATERIALIZED VIEW mv_rollup_1 AS
SELECT * FROM vw_rollup;
```



4. Drilldown

```
CREATE OR REPLACE VIEW vw_drilldown
AS SELECT
```

```
    TRANSACTION_ID,
    WAREHOUSE_ID,
    SUM(no_of_boxes) AS "Total
Boxes" FROM
inventory_management_fact
GROUP BY TRANSACTION_ID, WAREHOUSE_ID;


CREATE MATERIALIZED VIEW mv_drilldown
AS SELECT * FROM vw_drilldown;
```



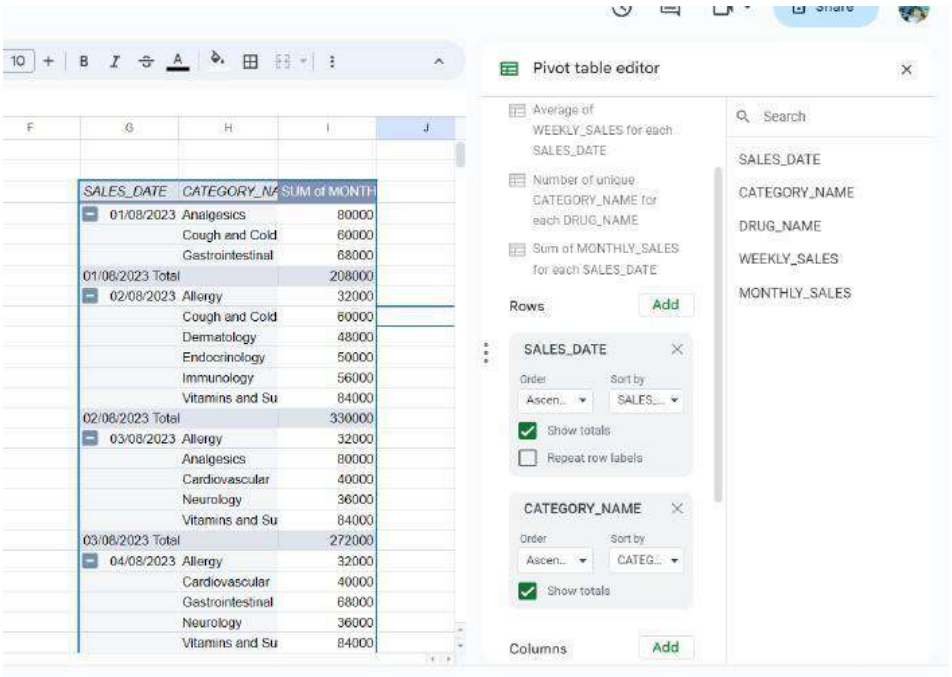
EXPERIMENT - 04

Implementation

: Pivot Table:

SUM of WEE	DRUG_NAM									
CATEGORY_NA	AllerClear 10	CardioGuard 50	Dermacare Oint	DiabetiCare 500	ImmunoGuard 7	MediHealth 100	NeuroCalm XR	Respigen Cough	StomachEase Pl	VitaBoost Multiv	Grand Total
Allergy	64000										64000
Analgesics						120000					120000
Cardiovascular		40000									40000
Cough and Cold								90000			90000
Dermatology			48000								48000
Endocrinology				37500							37500
Gastrointestinal									102000		102000
Immunology					70000						70000
Neurology							36000				36000
Vitamins and Supplements										84000	84000
Grand Total	**64000**	**40000**	**48000**	**37500**	**70000**	**120000**	**36000**	**90000**	**102000**	**84000**	**691500**

Slice:



Dice:

SUM of MON	CATEGORY_						
SALES_DATE	Allergy	Analgesics	Cardiovascular	Cough and Cold	Dermatology	Endocrinology	G
01/08/2023		80000		60000			
02/08/2023	32000			60000	48000	50000	
03/08/2023	32000	80000	40000				

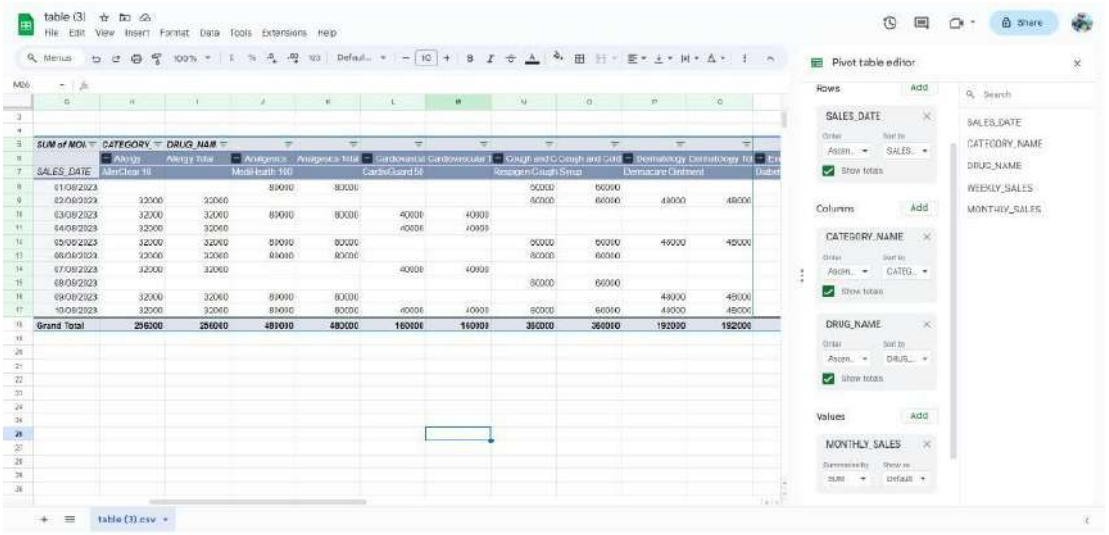
	Endocrinology	Gastrointestinal	Immunology	Neurology	Vitamins and Su	Grand Total
		68000				208000
00	50000		56000		84000	330000
				36000	84000	272000

Rollup:

SUM of MONTH	CATEGORY_NAME						
SALES_DATE	Allergy	Analgesics	Cardiovascular	Cough and Cold	Dermatology	Endocrinology	G
01/08/2023		80000		60000			
02/08/2023	32000			60000	48000	50000	
03/08/2023	32000	80000	40000				
04/08/2023	32000		40000				
05/08/2023	32000	80000		60000	48000		
06/08/2023	32000	80000		60000		50000	
07/08/2023	32000		40000			50000	
08/08/2023				60000			
09/08/2023	32000	80000			48000		
10/08/2023	32000	80000	40000	60000	48000		
Grand Total	**256000**	**480000**	**160000**	**360000**	**192000**	**150000**	

	Endocrinology	Gastrointestinal	Immunology	Neurology	Vitamins and Su	Grand Total
		68000				208000
00	50000		56000		84000	330000
				36000	84000	272000
		68000		36000	84000	260000
00		68000	56000			344000
	50000	68000	56000			346000
	50000	68000	56000		84000	330000
		68000				128000
00			56000	36000		252000
00				36000		296000
00	**150000**	**408000**	**280000**	**144000**	**336000**	**2766000**

Drilldown:



	R	S	T	U	V	W	X	Y	Z	AA	AB
6	Endocrinolog	Endocrinology To	Gastrointestir	Gastrointestinal	Immunology	Immunology Tot	Neurology	Neurology Total	Vitamins and	Vitamins and Su	Grand Total
7	DiabetiCare 500		StomachEase Plus		ImmunoGuard 75		NeuroCalm XR		VitaBoost Multivitamin		
8			68000	68000							208000
9	50000	50000			56000	56000			84000	84000	330000
10							36000	36000	84000	84000	272000
11			68000	68000			36000	36000	84000	84000	260000
12			68000	68000	56000	56000					244000
13	50000	50000	68000	68000	56000	56000					348000
14	50000	50000	68000	68000	56000	56000			84000	84000	330000
15			68000	68000							128000
16					56000	56000	36000	36000			252000
17							36000	36000			296000
18	150000	150000	408000	408000	280000	280000	144000	144000	336000	336000	2766000

EXPERIMENT - 05

Aim: Implementation of Bayesian algorithm

Theory:

Data Mining Bayesian Classifiers

The connection between the attribute set and the class variable is non-deterministic in numerous applications. In other words, we can say the class label of a test record can't be assumed with certainty even though its attribute set is the same as some of the training examples. These circumstances may emerge due to the noisy data or the presence of certain confusing factors that influence classification, but it is not included in the analysis. For example, consider the task of predicting the occurrence of whether an individual is at risk for liver illness based on the individual's eating habits and working efficiency. Although most people who eat healthy and exercise consistently have less probability of occurrence of liver disease, they may still do so due to other factors. For example, due to the consumption of high-calorie street foods and alcohol abuse. Determining whether an individual's eating routine is healthy or the workout efficiency is sufficient is also subject to analysis, which may introduce vulnerabilities to the learning issue.

Bayesian classification uses Bayes theorem to predict the occurrence of any event. Bayesian classifiers are the statistical classifiers with the Bayesian probability understandings. The theory expresses how a level of belief, is expressed as a probability.

Bayes theorem came into existence after Thomas Bayes, who first utilized conditional probability to provide an algorithm that uses evidence to calculate limits on an unknown parameter.

Bayes's theorem is expressed mathematically by the following equation that is given

below. For proposition X and evidence Y,

- P(X), the prior, is the primary degree of belief in X
- P(X/Y), the posterior is the degree of belief having accounted for Y.
- The quotient $\frac{P(Y/X)}{P(Y)}$ represents the supports Y provides for

X. Bayes theorem can be derived from the conditional probability:

$$P(X/Y) = \frac{P(X \cap Y)}{P(Y)}, \text{ if } P(Y) \neq 0$$

$$P(Y/X) = \frac{P(Y \cap X)}{P(X)}, \text{ if } P(X) \neq 0$$

Where P (X∩Y) is the **joint probability** of both X and Y being true, because

$$P(Y \cap X) = P(X \cap Y)$$

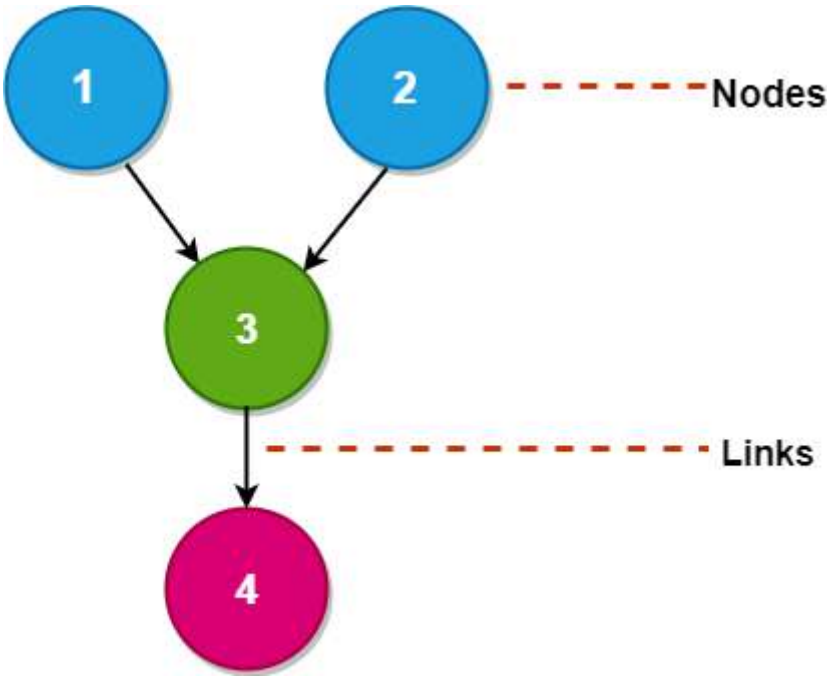
$$\text{or, } P(X \cap Y) = P(X/Y)P(Y) = P(Y/X)P(X)$$

$$\text{or, } P(X/Y) = \frac{P(Y/X)P(X)}{P(Y)}, \text{ if } P(Y) \neq 0$$

Bayesian network:

A Bayesian Network falls under the classification of Probabilistic Graphical Modelling (PGM) procedure that is utilized to compute uncertainties by utilizing the probability concept. Generally known as **Belief Networks, Bayesian Networks** are used to show uncertainties using **Directed Acyclic Graphs** (DAG)

A **Directed Acyclic Graph** is used to show a Bayesian Network, and like some other statistical graph, a DAG consists of a set of nodes and links, where the links signify the connection between the nodes.



The nodes here represent random variables, and the edges define the relationship between these variables.

A DAG models the uncertainty of an event taking place based on the Conditional Probability Distribution (CDP) of each random variable. A **Conditional Probability Table** (CPT) is used to represent the CPD of each variable in a network.

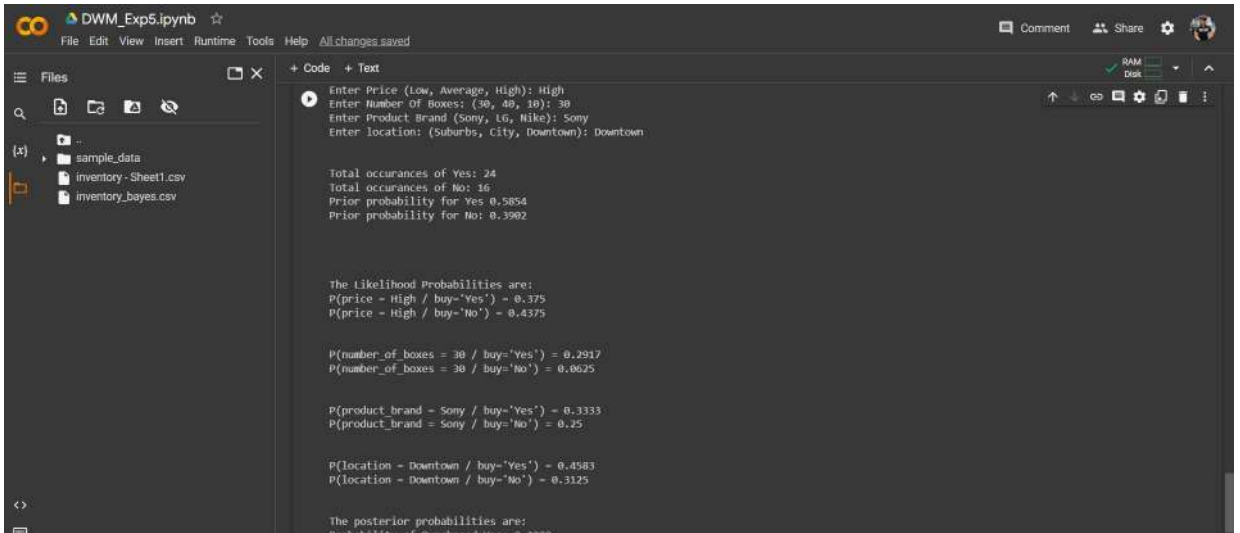
CODE:

```
import pandas as
pd import math
data =


pd.read_csv("/content/inventory_bayes.csv") price


= input("Enter Price (Low, Average, High): ")


number_of_boxes = int(input("Enter Number Of Boxes: (30, 40, 10): "))
product_brand = input("Enter Product Brand (Sony, LG, Nike):
") location = input("Enter location: (Suburbs, City, Downtown):
")

print("\n")
#prior probabilities
total_instances =
len(data)
package_left_yes = len(data[data['package_left'] ==
'Yes']) package_left_no = len(data[data['package_left'] ==
'No'])

p_yes = round(package_left_yes/total_instances,
4) p_no = round(package_left_no/total_instances,
4)

print(f"Total occurances of Yes:
{package_left_yes}") print(f"Total occurances of No:
{package_left_no}")

print(f"Prior probability for Yes
{p_yes}") print(f"Prior probability for
No: {p_no}")

#Likelihood Probabilities
```

```
def calculate_likelihood(attribute, value,
 package_left): subset = data[data['package_left'] ==
 package_left] count = len(subset[subset[attribute] ==
 value])


 total = len(subset)

 return count /

 total

print('\n')
print('\n')
print("The Likelihood Probabilities are: ")

#
p_price_given_yes = round(calculate_likelihood('price', price, 'Yes'),
4) p_price_given_no = round(calculate_likelihood('price', price, 'No'),
4)

print(f"P(price = {price} / buy='Yes') =
{p_price_given_yes}") print(f"P(price = {price} / buy='No') =
{p_price_given_no}") print('\n')


#
p_number_of_boxes_given_yes = round(calculate_likelihood('number_of_boxes',
number_of_boxes, 'Yes'), 4)
p_number_of_boxes_given_no = round(calculate_likelihood('number_of_boxes',
number_of_boxes, 'No'), 4)

print(f"P(number_of_boxes = {number_of_boxes} / buy='Yes') =
{p_number_of_boxes_given_yes}")
print(f"P(number_of_boxes = {number_of_boxes} / buy='No')
=
{p_number_of_boxes_given_no}"
) print('\n')

#
p_product_brand_given_yes = round(calculate_likelihood('product_brand',
product_brand, 'Yes'), 4)
```

```
p_product_brand_given_no = round(calculate_likelihood('product_brand',
product_brand, 'No'), 4)

print(f"P(product_brand = {product_brand} / buy='Yes') =
{p_product_brand_given_yes}") print(f"P(product_brand = {product_brand} / buy='No') =
{p_product_brand_given_no}") print('\n')

#
p_location_given_yes = round(calculate_likelihood('location', location, 'Yes'), 4)
p_location_given_no = round(calculate_likelihood('location', location, 'No'), 4)

print(f"P(location = {location} / buy='Yes') =
{p_location_given_yes}") print(f"P(location = {location} / buy='No') =
{p_location_given_no}") print('\n')

# Posterior probabilities
print("The posterior probabilities are: ")

p_yes_given_x = round(p_yes * p_price_given_yes * p_number_of_boxes_given_yes
* p_product_brand_given_yes * p_location_given_yes,4)
p_no_given_x = round(p_no * p_price_given_no * p_number_of_boxes_given_yes
* p_product_brand_given_no * p_location_given_no,4)

if p_yes_given_x > p_no_given_x:
  prediction =
'Yes' else:
  prediction = 'No'

print(f'Probability of Purchased=Yes:
{p_yes_given_x}') print(f'Probability of Purchased=No:
{p_no_given_x:}') print(f'Prediction: {prediction}')
```

Output:



EXPERIMENT - 06

Aim: Perform data Pre-processing task and Demonstrate performing Classification, Clustering, and Association algorithms on data sets using a data mining tool (WEKA/R tool)

Theory:

Classification:

The term "classification" is usually used when there are exactly two target classes called binary classification. When more than two classes may be predicted, specifically in pattern recognition problems, this is often referred to as multinomial classification. However, multinomial classification is also used for categorical response data, where one wants to predict which category amongst several categories has the instances with the highest probability.

Classification is one of the most important tasks in data mining. It refers to a process of assigning pre-defined class labels to instances based on their attributes. There is a similarity between classification and clustering, it looks similar, but it is different. The major difference between classification and clustering is that classification includes the levelling of items according to their membership in pre-defined groups. Let's understand this concept with the help of an example; suppose you are using a self-organizing map neural network algorithm for image recognition where there are 10 different kinds of objects. If you label each image with one of these 10 classes, the classification task is solved.

Clustering:

Clustering refers to a technique of grouping objects so that objects with the same functionalities come together and objects with different functionalities go apart. In other words, we can say that clustering is a process of portioning a data set into a set of meaningful subclasses, known as clusters. Clustering is the same as classification in which data is grouped. However, unlike classification, the groups are not previously defined.
Instead, the grouping is achieved by determining similarities between data according to characteristics found in the real data. The groups are called Clusters.

Association Rules:

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database.

Association rule learning is one of the very important concepts of machine learning, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailers to discover the associations between items. We can understand it by taking the example of a supermarket, as in a supermarket, all products that are purchased together are put together.

Association rule learning can be divided into three types of algorithms:

1. **Apriori**
2. **Eclat**
3. **F-P Growth Algorithm**

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp(X)} = \frac{Freq(X)}{T}$$

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

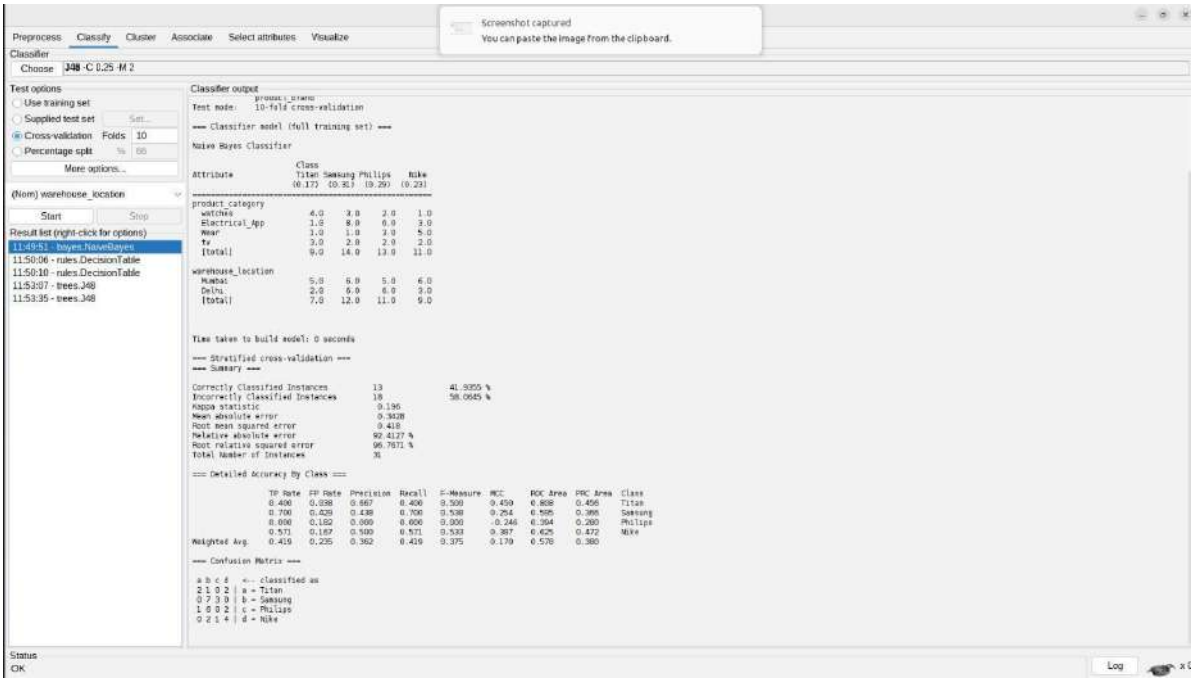
It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

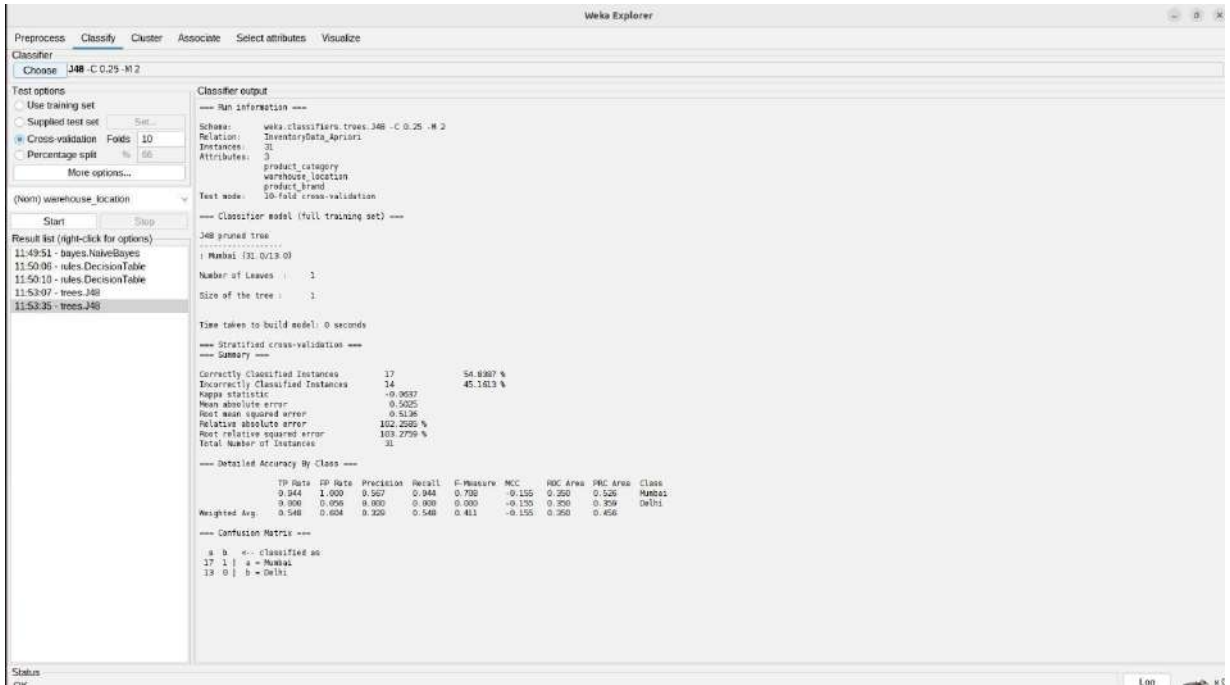
- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Output:


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Clusterer
Choose HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set Set...
Percentage split % 66
Classes to clusters evaluation
(Nom) product_brand
Store clusters for visualization
Ignore attributes
Start Stop
Result list (right-click for options)
11:50:24 - SimpleKMeans
11:50:28 - HierarchicalClusterer
Clusterer output
=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation: InventoryData_Apriori
Instances: 31
Attributes: 3
product_category
warehouse_location
product_brand
Test mode: evaluate on training data
=== Clustering model (full training set) ===
kMeans
======
Number of iterations: 3
Within cluster sum of squared errors: 42.0
Initial starting points (random):
Cluster 0: Wear,Mumbai,Nike
Cluster 1: watches,Mumbai,Titan
Missing values globally replaced with mean/mode
Final cluster centroids:
Cluster#
Attribute Full Data 0 1
(31.0) (25.0) (6.0)
product_category Electrical_App Electrical_App watches
warehouse_location Mumbai Mumbai Mumbai
product_brand Samsung Samsung Titan
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 25 (81%)
1 6 (19%)
Status
OK
Log


Screenshot captured
You can paste the image from the clipboard.
Preprocess Classify Cluster Associate Select attributes Visualize
Clusterer
Choose HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Cluster mode
Use training set
Supplied test set Set...
Percentage split % 66
Classes to clusters evaluation
(Nom) product_brand
Store clusters for visualization
Ignore attributes
Start Stop
Result list (right-click for options)
11:50:24 - SimpleKMeans
11:50:28 - HierarchicalClusterer
Clusterer output
=== Run information ===
Scheme: weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation: InventoryData_Apriori
Instances: 31
Attributes: 3
product_category
warehouse_location
product_brand
Test mode: evaluate on training data
=== Clustering model (full training set) ===
Cluster 0
((0.0:0,0.0:0):1,(0.0:0,0.0:0):1)
Cluster 1
Time taken to build model (full training data) : 0 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 4 (13%)
1 27 (87%)
Status
OK
Log


Screenshot captured
You can paste the image from the clipboard.
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) warehouse_location
Start Stop
Result list (right-click for options)
11:49:51 - bayes.NaiveBayes
11:50:06 - rules.DecisionTable
11:50:10 - rules.DecisionTable
11:53:07 - trees.J48
11:53:35 - trees.J48
Classifier output
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
Naive Bayes Classifier
Class
Attribute Titan Samsung Philips Nike
(0.17) (0.31) (0.29) (0.23)
product_category
watches 6.0 3.0 2.0 1.0
Electrical_App 1.0 8.0 0.0 3.0
Wear 1.0 1.0 3.0 5.0
Tv 3.0 2.0 2.0 2.0
[total] 9.0 14.0 13.0 11.0
warehouse_location
Mumbai 5.0 6.0 5.0 6.0
Delhi 2.0 6.0 6.0 3.0
[total] 7.0 12.0 11.0 9.0
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 13 41.9355 %
Incorrectly Classified Instances 18 58.0645 %
Kappa statistic 0.1196
Mean absolute error 0.3828
Root mean squared error 0.418
Relative absolute error 92.4127 %
Root relative squared error 96.7671 %
Total Number of Instances 31
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.400 0.038 0.667 0.400 0.500 0.459 0.808 0.456 Titan
0.700 0.429 0.438 0.700 0.538 0.254 0.585 0.395 Samsung
0.000 0.182 0.000 0.000 0.000 -0.246 0.394 0.280 Philips
0.571 0.167 0.500 0.571 0.533 0.387 0.625 0.472 Nike
Weighted Avg. 0.419 0.205 0.362 0.419 0.375 0.179 0.578 0.380
=== Confusion Matrix ===
a b c d <-- classified as
2 1 0 2 | a = Titan
0 7 3 0 | b = Samsung
1 0 0 2 | c = Philips
0 2 1 4 | d = Nike
Status
OK
Log


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) warehouse_location
Start Stop
Result list (right-click for options)
11:49:51 - bayes.NaiveBayes
11:50:06 - rules.DecisionTable
11:50:10 - rules.DecisionTable
11:53:07 - trees.J48
11:53:35 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: InventoryData_Apriori
Instances: 31
Attributes: 3
product_category
warehouse_location
product_brand
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
: Mumbai (31.0/13.0)
Number of Leaves : 1
Size of the tree : 1
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 17 54.8387 %
Incorrectly Classified Instances 14 45.1613 %
Kappa statistic -0.0697
Mean absolute error 0.5025
Root mean squared error 0.5136
Relative absolute error 102.2585 %
Root relative squared error 103.2759 %
Total Number of Instances 31
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.944 1.000 0.567 0.944 0.708 0.155 0.350 0.526 Mumbai
0.000 0.056 0.000 0.000 0.000 -0.155 0.350 0.359 Delhi
Weighted Avg. 0.548 0.604 0.320 0.548 0.411 -0.155 0.350 0.456
=== Confusion Matrix ===
a b <-- classified as
17 1 | a = Mumbai
13 0 | b = Delhi
Status
OK
Log


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
FPGrowth -P 2 -I -1 -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1
Start
Stop
Result list (right-clic...
11:48:58 - Apriori
11:54:19 - Apriori
Associator output
=== Run information ===
Scheme: weka.associations.Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Relation: InventoryData_Apriori
Instances: 31
Attributes: 3
product_category
warehouse_location
product_brand
=== Associator model (full training set) ===
Apriori
=======
Minimum support: 0.1 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18
Generated sets of large itemsets:
Size of set of large itemsets L(1): 10
Size of set of large itemsets L(2): 15
Size of set of large itemsets L(3): 3
Best rules found:
1. warehouse_location=Mumbai product_brand=Gaasung 5 ==> product_category=Electrical_App 5 <conf:(1)> lift:(2.21) lev:(0.09) [2] conv:(2.74)
2. product_category=Wear product_brand=Nike 4 ==> warehouse_location=Mumbai 4 <conf:(1)> lift:(1.72) lev:(0.05) [1] conv:(1.68)
Status
OK
Log

EXPERIMENT - 07

Aim: Implementation of Clustering algorithm (K-means/Kmedoids)

Theory:

K means Clustering

K-Means Clustering is an Unsupervised Machine Learning algorithm, which groups the unlabeled dataset into different clusters.

Unsupervised Machine Learning is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine's job in this case is to organize unsorted data according to parallels, patterns, and variations.

The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm; an unsupervised learning algorithm. 'K' in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

(It will help if you think of items as points in an n-dimensional space). The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the Euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The "points" mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive

method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature x, the items have values in [0,3], we will initialize the means with values for x at [0,3]).

The above algorithm in pseudocode is as follows:

Initialize k means with random values

--> For a given number of iterations:

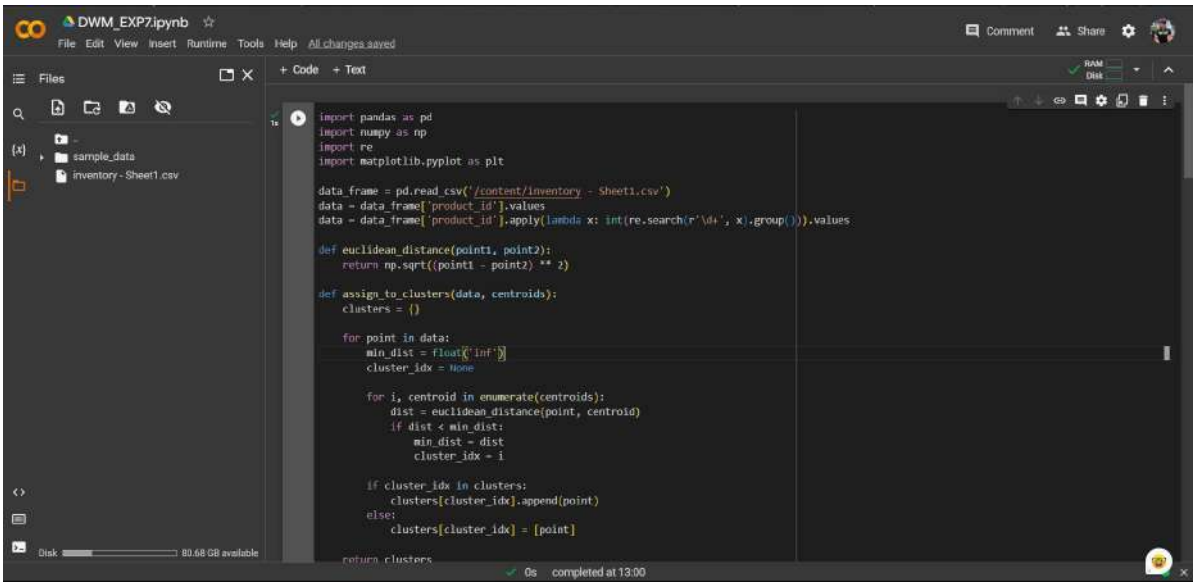
--> Iterate through items:

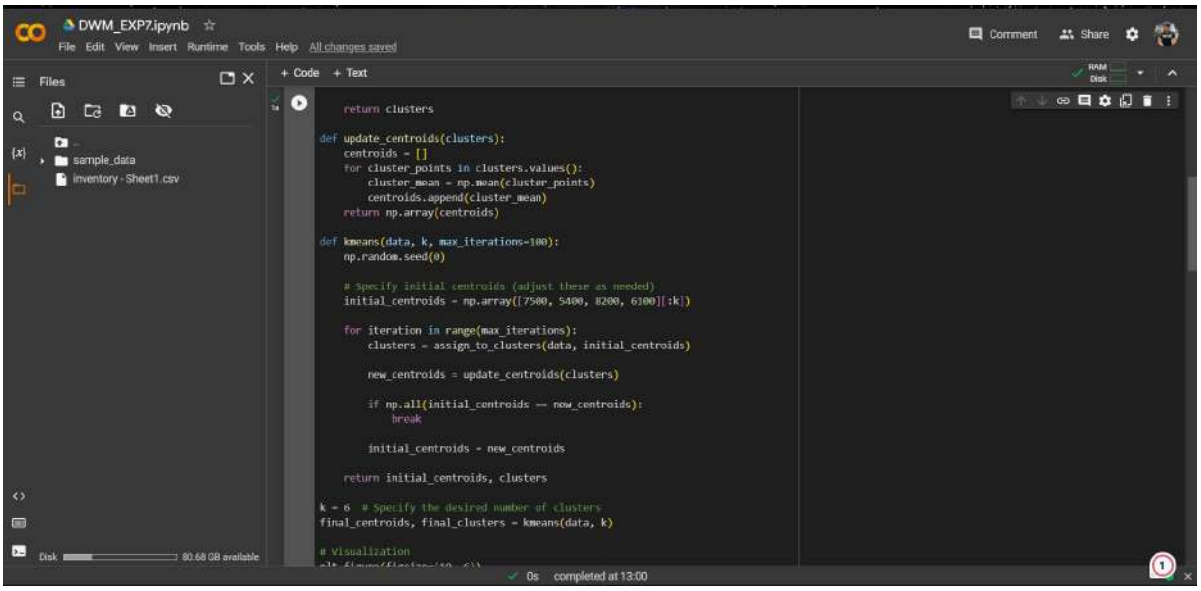
--> Find the mean closest to the item by calculating
the Euclidean distance of the item with each of the means

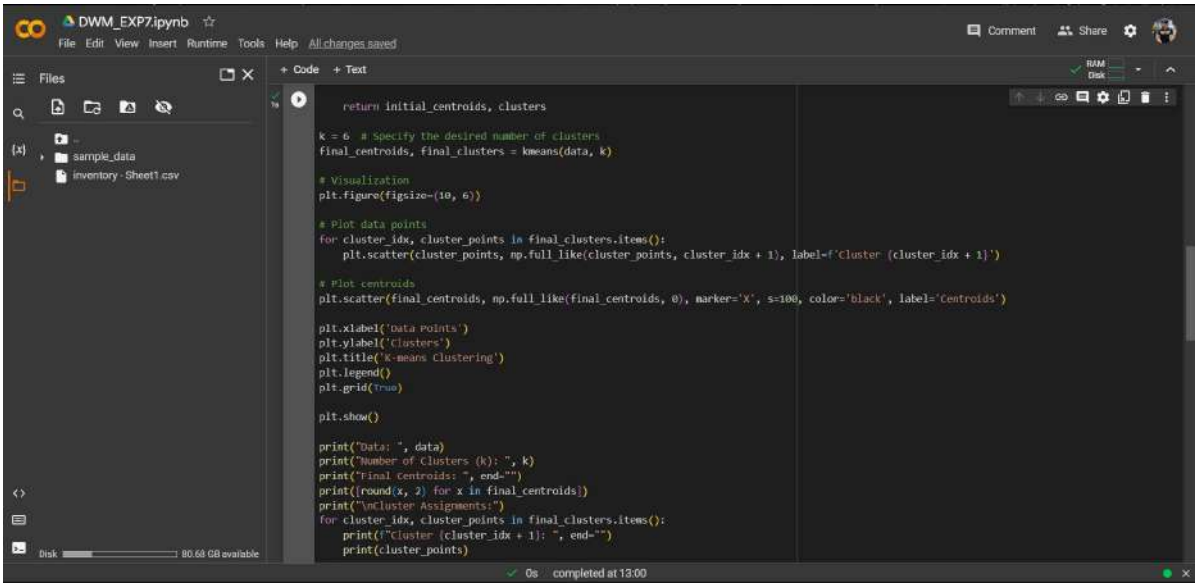
--> Assign item to mean

--> Update mean by shifting it to the average of the items in that cluster

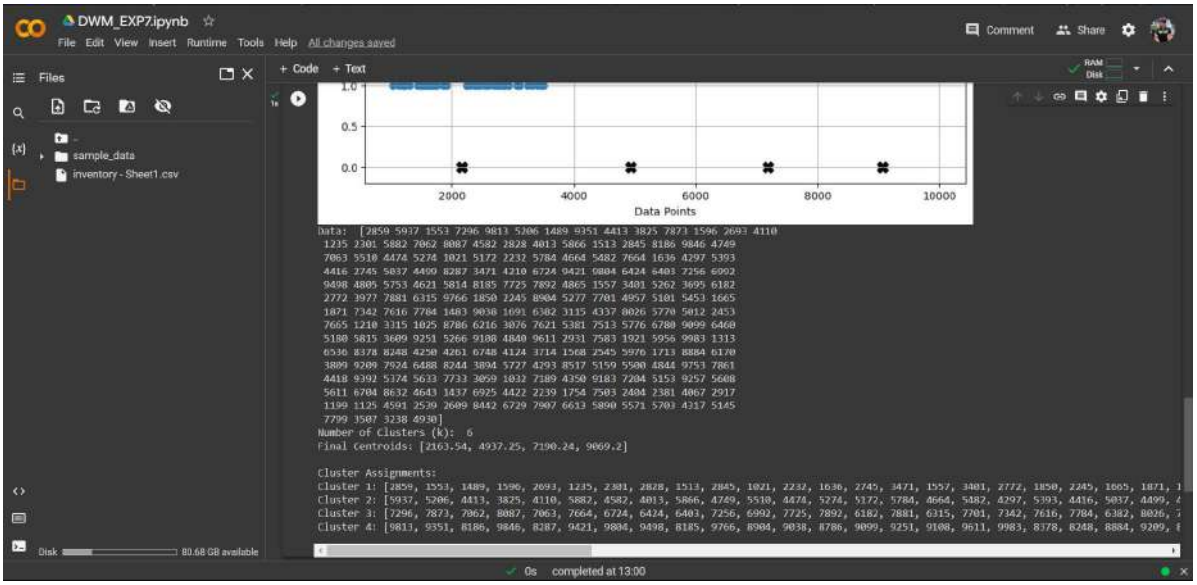
Output:


DWM_EXP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
sample_data
inventory - Sheet1.csv
Disk
80.68 GB available
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
data_frame = pd.read_csv('/content/inventory - Sheet1.csv')
data = data_frame['product_id'].values
data = data_frame['product_id'].apply(lambda x: int(re.search(r'\d+', x).group())).values
def euclidean_distance(point1, point2):
return np.sqrt((point1 - point2) ** 2)
def assign_to_clusters(data, centroids):
clusters = {}
for point in data:
min_dist = float('inf')
cluster_idx = None
for i, centroid in enumerate(centroids):
dist = euclidean_distance(point, centroid)
if dist < min_dist:
min_dist = dist
cluster_idx = i
if cluster_idx in clusters:
clusters[cluster_idx].append(point)
else:
clusters[cluster_idx] = [point]
return clusters
0s completed at 13:00


DWM_EXP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
sample_data
inventory - Sheet1.csv
Disk
80.68 GB available
return clusters
def update_centroids(clusters):
centroids = []
for cluster_points in clusters.values():
cluster_mean = np.mean(cluster_points)
centroids.append(cluster_mean)
return np.array(centroids)
def kmeans(data, k, max_iterations=100):
np.random.seed(0)
# Specify initial centroids (adjust these as needed)
initial_centroids = np.array([7500, 5400, 8200, 6100][:k])
for iteration in range(max_iterations):
clusters = assign_to_clusters(data, initial_centroids)
new_centroids = update_centroids(clusters)
if np.all(initial_centroids == new_centroids):
break
initial_centroids = new_centroids
return initial_centroids, clusters
k = 6 # Specify the desired number of clusters
final_centroids, final_clusters = kmeans(data, k)
Visualization
0s completed at 13:00


DWM_EXP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
sample_data
inventory - Sheet1.csv
return initial_centroids, clusters
k = 6 # Specify the desired number of clusters
final_centroids, final_clusters = kmeans(data, k)
# Visualization
plt.figure(figsize=(10, 6))
# Plot data points
for cluster_idx, cluster_points in final_clusters.items():
plt.scatter(cluster_points, np.full_like(cluster_points, cluster_idx + 1), label=f'Cluster {cluster_idx + 1}')
# Plot centroids
plt.scatter(final_centroids, np.full_like(final_centroids, 0), marker='X', s=100, color='black', label='Centroids')
plt.xlabel('Data Points')
plt.ylabel('Clusters')
plt.title('K-means Clustering')
plt.legend()
plt.grid(True)
plt.show()
print("Data: ", data)
print("Number of Clusters (k): ", k)
print("Final Centroids: ", end="")
print([round(x, 2) for x in final_centroids])
print("\nCluster Assignments:")
for cluster_idx, cluster_points in final_clusters.items():
print(f"Cluster {cluster_idx + 1}: ", end="")
print(cluster_points)
Disk 80.68 GB available
0s completed at 13:00


DWM_EXP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
sample_data
inventory - Sheet1.csv
K-means Clustering
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Centroids
Clusters
Data Points
Data: [2859 5917 1553 7296 9813 5206 1489 9351 4413 3825 7873 1596 2693 4110
1235 2301 5882 7062 8087 4582 2828 4013 5866 1513 2845 8186 9846 4769
7063 5510 4474 5274 1021 5172 2232 5784 4664 5482 7664 1636 4297 5393
Disk 80.68 GB available
0s completed at 13:00


DWM_EXP7.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
sample_data
inventory - Sheet1.csv
1.0
0.5
0.0
2000
4000
6000
8000
10000
Data Points
Data: [2859 5937 1553 7296 9813 5206 1489 9351 4413 3825 7873 1596 2693 4110
1235 2301 5882 7062 8087 4582 2828 4013 5866 1513 2845 8186 9846 4749
7063 5510 4474 5274 1021 5172 2232 5784 4664 5482 7664 1636 4297 5393
4416 2745 5037 4499 8287 3471 4210 6724 9421 9804 6424 6403 7256 6992
9498 4805 5753 4621 5814 8185 7725 7892 4865 1557 3401 5262 3695 6182
2772 3977 7881 6315 9766 1850 2245 8904 5277 7701 4957 5101 5453 1665
1071 7342 7616 7784 1483 9038 1691 6382 3115 4337 8026 5770 5012 2453
7665 1210 3315 1025 8786 6216 3076 7621 5381 7513 5776 6780 9099 6460
5180 5815 3609 9251 5266 9108 4840 9611 2931 7583 1921 5956 9983 1313
6530 8378 8248 4250 4261 6748 4124 3714 1568 2545 5976 1713 8884 6170
3809 9209 7924 6488 8244 3894 5727 4293 8517 5159 5500 4844 9751 7861
4418 9392 5374 5633 7733 3059 1032 7189 4350 9183 7204 5153 9257 5608
5611 6704 8632 4643 1437 6925 4422 2239 1754 7503 2404 2381 4067 2917
1199 1125 4593 2539 3609 8442 6729 7907 6613 5890 5571 5703 4317 5145
7799 3507 3238 4930]
Number of Clusters (k): 6
Final Centroids: [2163.54, 4937.25, 7190.24, 9069.2]
Cluster Assignments:
Cluster 1: [2859, 1553, 1489, 1596, 2693, 1235, 2301, 2828, 1513, 2845, 1021, 2232, 1636, 2745, 1471, 1557, 1401, 2772, 1850, 2245, 1665, 1871,
Cluster 2: [5937, 5206, 4413, 3825, 4110, 5882, 4582, 4013, 5866, 4749, 5510, 4474, 5274, 5172, 5784, 4664, 5482, 4297, 5393, 4416, 5037, 4499,
Cluster 3: [7296, 7873, 7062, 8087, 7063, 7664, 6724, 6424, 6403, 7256, 6992, 7725, 7892, 6182, 7881, 6315, 7701, 7342, 7616, 7784, 6382, 8026,
Cluster 4: [9813, 9351, 8186, 9846, 8287, 9421, 9804, 9498, 8185, 9766, 8904, 9038, 8786, 9099, 9251, 9108, 9611, 9983, 8378, 8248, 8884, 9209,
Disk 80.68 GB available
0s completed at 13:00

EXPERIMENT - 08

Aim: Implementation of any one Hierarchical Clustering method

Theory:

A Hierarchical clustering method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

Identify the 2 clusters which can be closest together, and
Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.
In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called a Dendrogram (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view).

Hierarchical clustering is a method of cluster analysis in data mining that creates a hierarchical representation of the clusters in a dataset. The method starts by treating each data point as a separate cluster and then iteratively combines the closest clusters until a stopping criterion is reached. The result of hierarchical clustering is a tree-like structure, called a dendrogram, which illustrates the hierarchical relationships among the clusters.

Hierarchical clustering has a number of advantages over other clustering methods, including:

1. The ability to handle non-convex clusters and clusters of different sizes and densities.
2. The ability to handle missing data and noisy data.
3. The ability to reveal the hierarchical structure of the data, can be useful for understanding the relationships among the clusters. However, it also has some drawbacks, such as:
4. The a need for a criterion to stop the clustering process and determine the final number of clusters.
5. The computational cost and memory requirements of the method can be high, especially for large datasets.
6. The results can be sensitive to the initial conditions, linkage criterion, and distance metric used.

In summary, Hierarchical clustering is a method of data mining that groups similar data points into clusters by creating a hierarchical structure of the clusters.

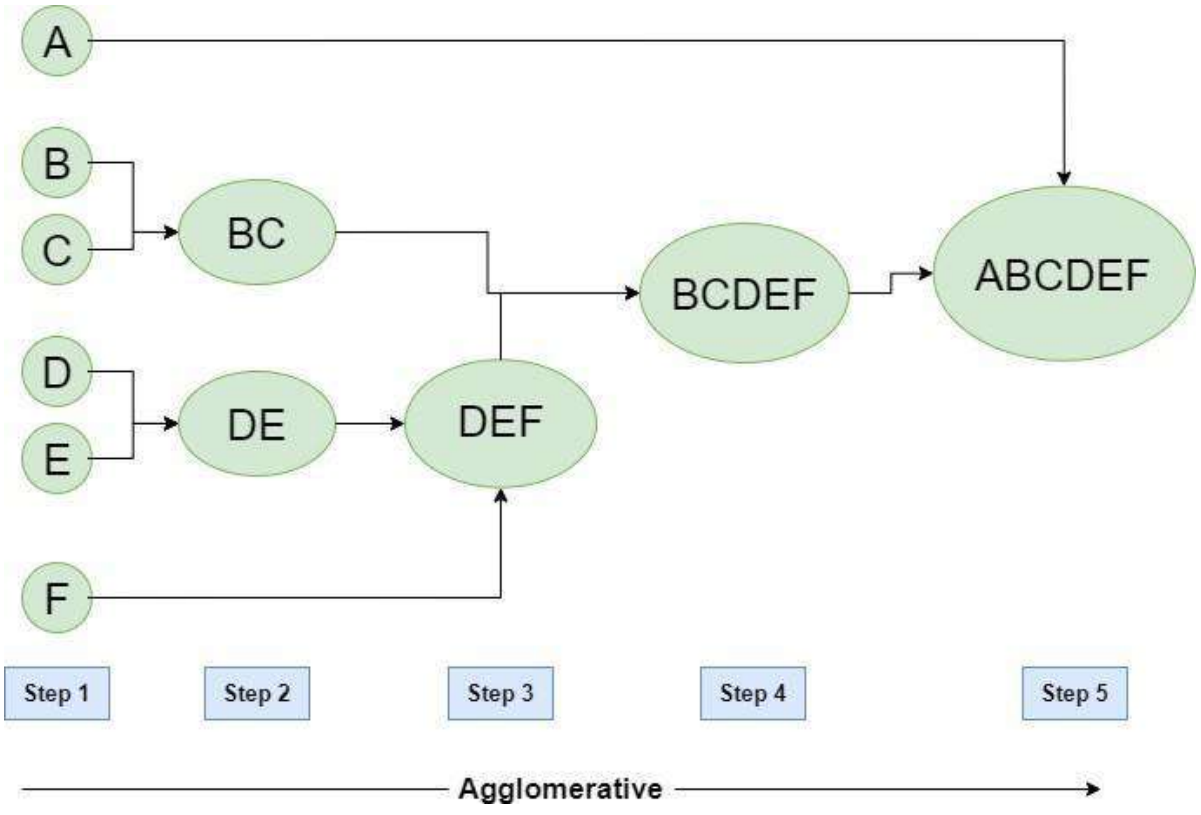
7. This method can handle different types of data and reveal the relationships among the clusters. However, it can have high computational cost and results can be sensitive to some conditions.

Agglomerative: Initially consider every data point as an individual Cluster and at every step, merge the nearest pairs of the cluster. (It is a bottom-up method). At first, every dataset is considered an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed.

The algorithm for Agglomerative Hierarchical Clustering is:

- Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
- Consider every data point as an individual cluster
- Merge the clusters which are highly similar or close to each other.
- Recalculate the proximity matrix for each cluster
- Repeat Steps 3 and 4 until only a single cluster remains.

Let's say we have six data points A, B, C, D, E, and F.



Step 1: Consider each alphabet as a single cluster and calculate the distance of one cluster from all the other clusters.

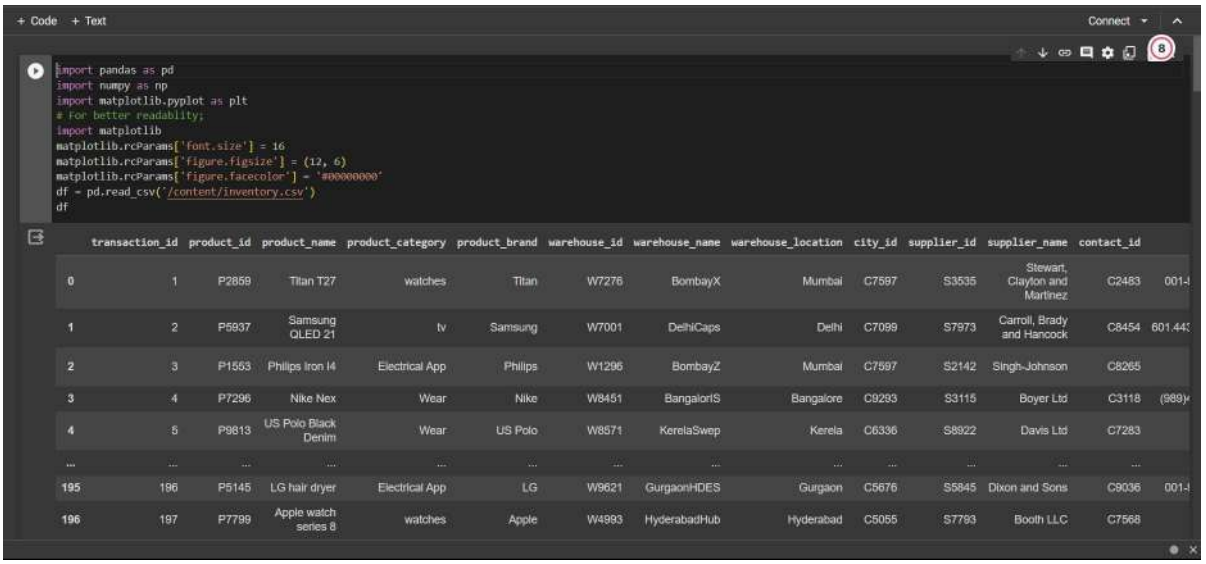
Step 2: In the second step comparable clusters are merged together to form a single cluster. Let's say cluster (B) and cluster (C) are very similar to each other therefore we merge them in the second step similarly to cluster (D) and (E) and at last, we get the clusters [(A), (BC), (DE), (F)]

Step 3: We recalculate the proximity according to the algorithm and merge the two nearest clusters([(DE), (F)]) together to form new clusters as [(A), (BC), (DEF)]

Step 4: Repeating the same process; The clusters DEF and BC are comparable and merged together to form a new cluster. We're now left with clusters [(A), (BCDEF)].

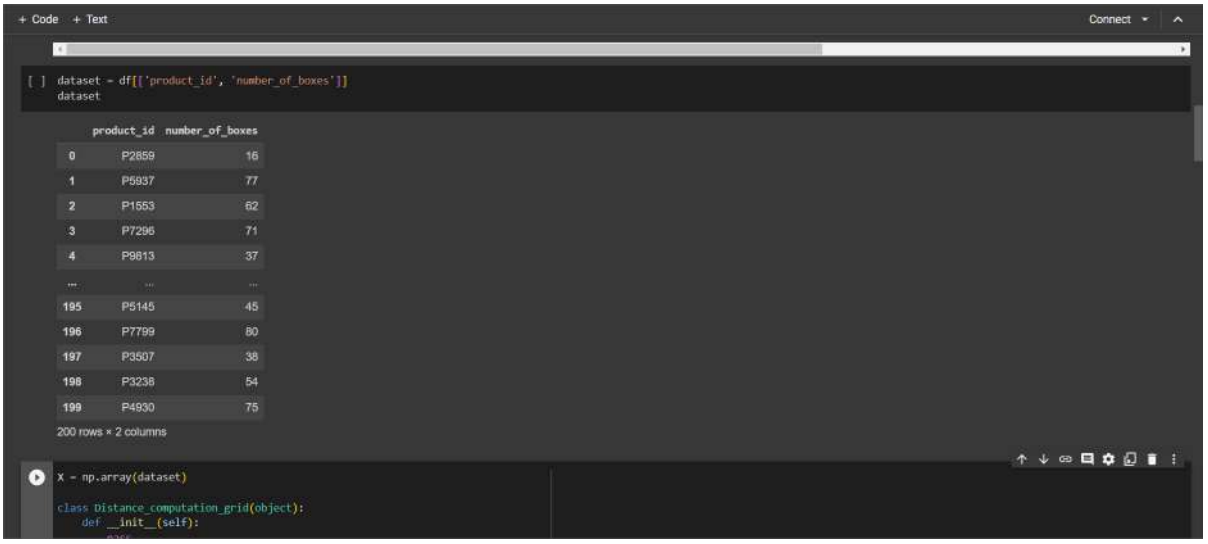
Step 5: At last the two remaining clusters are merged together to form a single cluster [(ABCDEF)].

Output:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# For better readability;
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (12, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
df = pd.read_csv('/content/inventory.csv')
df
```

	transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	
0	1	P2859	Titan T27	watches	Titan	W7276	BombayX	Mumbai	C7597	S3535	Stewart, Clayton and Martinez	C2483	001-(
1	2	P5937	Samsung QLED 21	tv	Samsung	W7001	DelhiCaps	Delhi	C7099	S7973	Carroll, Brady and Hancock	C8454	601.44(
2	3	P1553	Philips Iron I4	Electrical App	Philips	W1296	BombayZ	Mumbai	C7597	S2142	Singh-Johnson	C8265	
3	4	P7296	Nike Nex	Wear	Nike	W8451	BangaloriS	Bangalore	C9293	S3115	Boyer Ltd	C3118	(989)(
4	5	P9813	US Polo Black Denim	Wear	US Polo	W8571	KerelaSwep	Kerela	C6336	S8922	Davis Ltd	C7283	
...	...	...	...	...	...	...	...	...	...	...	...	...	
195	196	P5145	LG hair dryer	Electrical App	LG	W9621	GurgaonHDES	Gurgaon	C5676	S5845	Dixon and Sons	C9036	001-(
196	197	P7799	Apple watch series 8	watches	Apple	W4993	HyderabadHub	Hyderabad	C5055	S7793	Booth LLC	C7568	

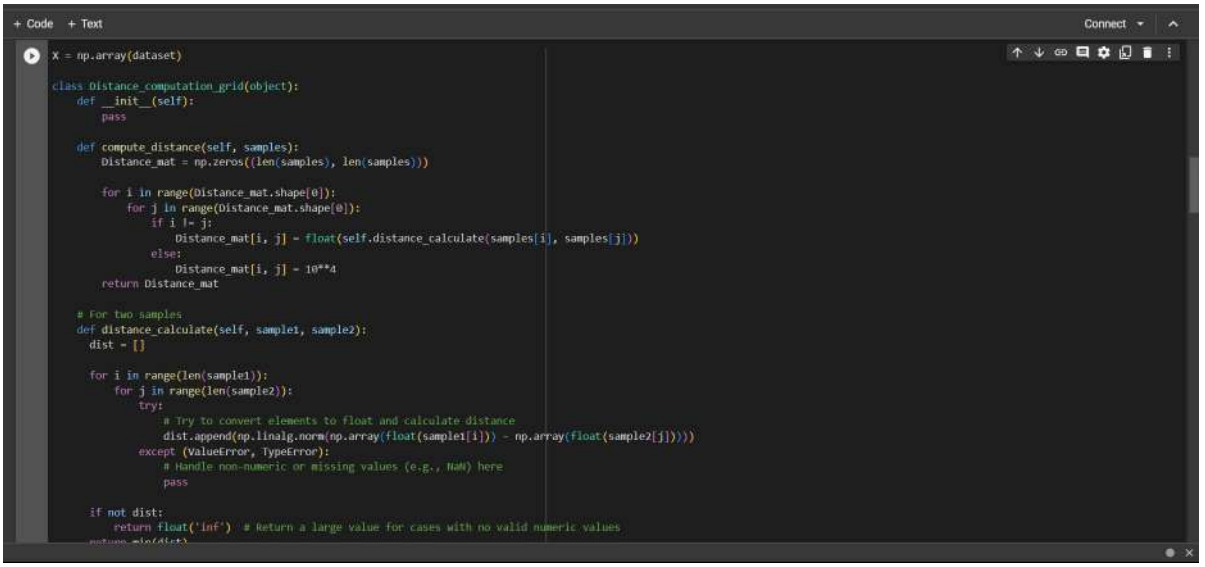


```
dataset = df[['product_id', 'number_of_boxes']]
dataset
```

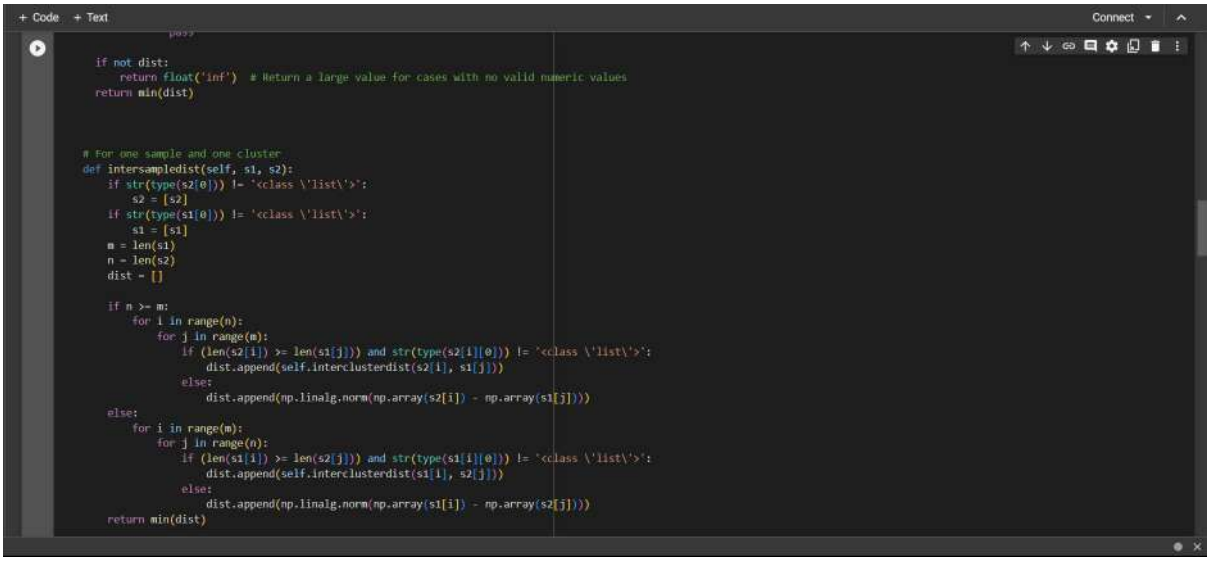
	product_id	number_of_boxes
0	P2859	16
1	P5937	77
2	P1553	62
3	P7296	71
4	P9813	37
...	...	...
195	P5145	45
196	P7799	80
197	P3507	38
198	P3238	54
199	P4930	75

200 rows × 2 columns

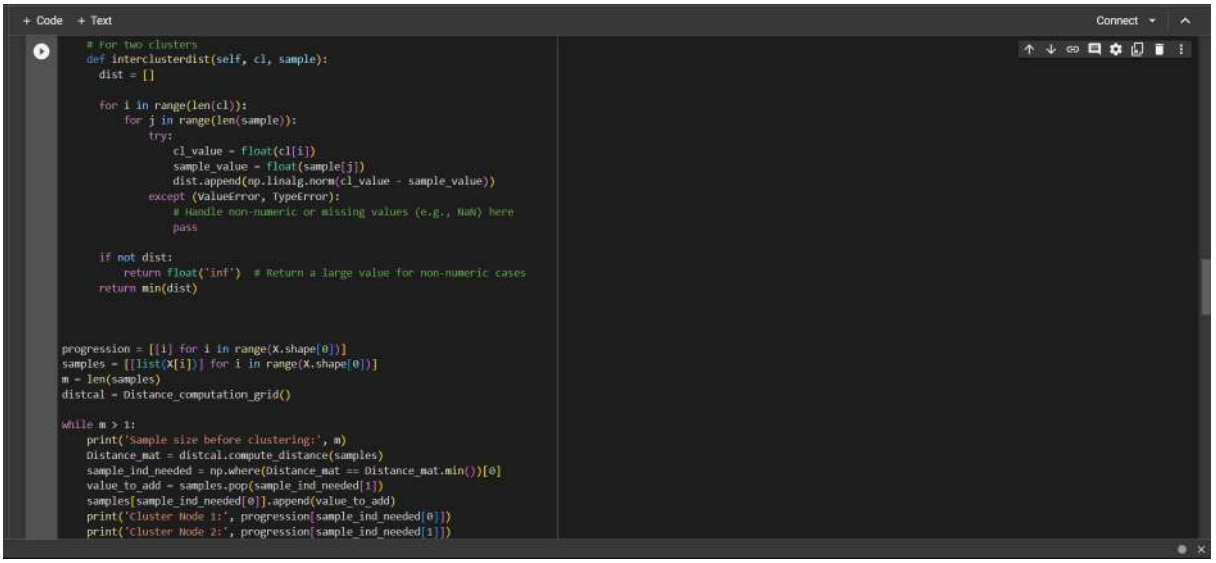
```
X = np.array(dataset)

class Distance_computation_grid(object):
    def __init__(self):
```



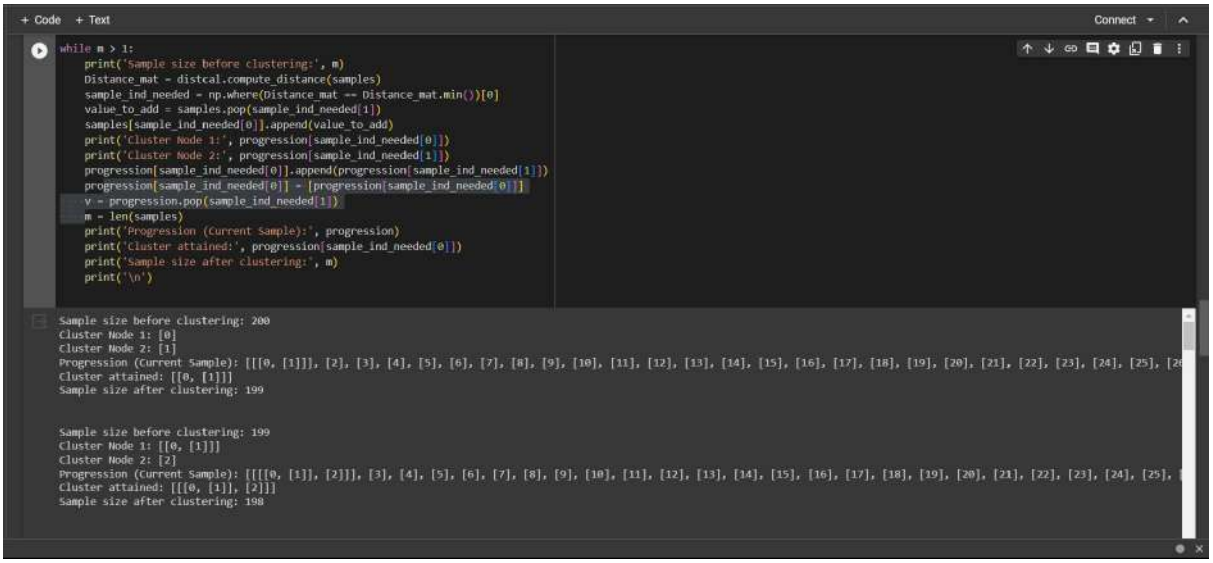
```
X = np.array(dataset)

class Distance_computation_grid(object):
    def __init__(self):
        pass

    def compute_distance(self, samples):
        Distance_mat = np.zeros((len(samples), len(samples)))

        for i in range(Distance_mat.shape[0]):
            for j in range(Distance_mat.shape[0]):
                if i != j:
                    Distance_mat[i, j] = float(self.distance_calculate(samples[i], samples[j]))
                else:
                    Distance_mat[i, j] = 10**4
        return Distance_mat

    # For two samples
    def distance_calculate(self, sample1, sample2):
      dist = []

      for i in range(len(sample1)):
          for j in range(len(sample2)):
              try:
                  # Try to convert elements to float and calculate distance
                  dist.append(np.linalg.norm(np.array(float(sample1[i])) - np.array(float(sample2[j]))))
              except (ValueError, TypeError):
                  # Handle non-numeric or missing values (e.g., NaN) here
                  pass

      if not dist:
          return float('inf')  # Return a large value for cases with no valid numeric values
      return min(dist)
```



```
    if not dist:
        return float('inf')  # Return a large value for cases with no valid numeric values
    return min(dist)



# For one sample and one cluster
def intersampledist(self, s1, s2):
    if str(type(s2[0])) != "<class \'list\'>":
        s2 = [s2]
    if str(type(s1[0])) != "<class \'list\'>":
        s1 = [s1]
    m = len(s1)
    n = len(s2)
    dist = []

    if n >= m:
        for i in range(n):
            for j in range(m):
                if (len(s2[i]) >= len(s1[j])) and str(type(s2[i][0])) != "<class \'list\'>":
                    dist.append(self.interclusterdist(s2[i], s1[j]))
                else:
                    dist.append(np.linalg.norm(np.array(s2[i]) - np.array(s1[j])))
    else:
        for i in range(m):
            for j in range(n):
                if (len(s1[i]) >= len(s2[j])) and str(type(s1[i][0])) != "<class \'list\'>":
                    dist.append(self.interclusterdist(s1[i], s2[j]))
                else:
                    dist.append(np.linalg.norm(np.array(s1[i]) - np.array(s2[j])))
    return min(dist)
```

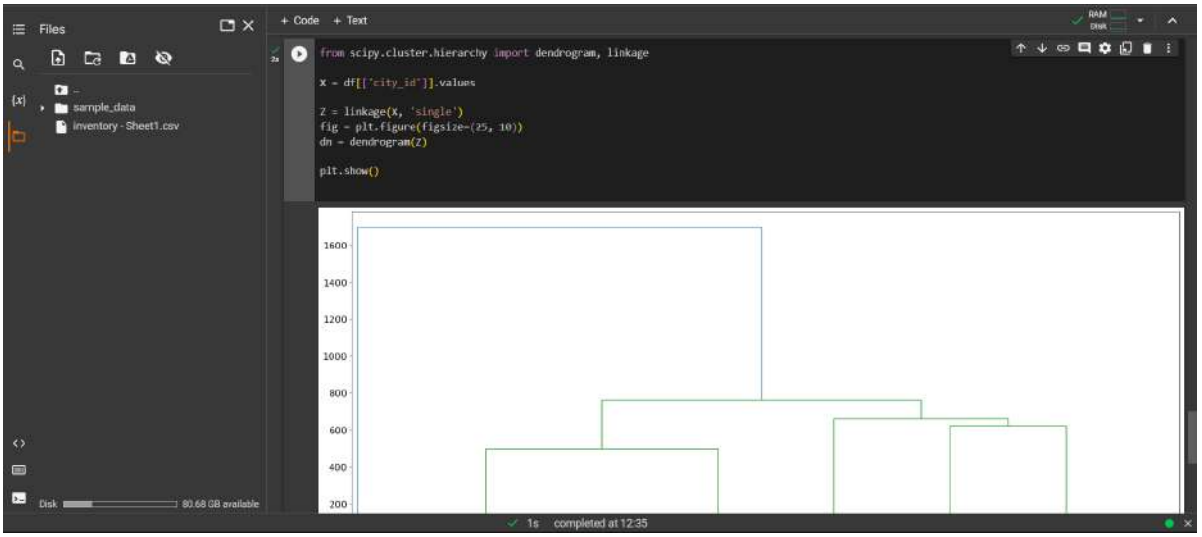


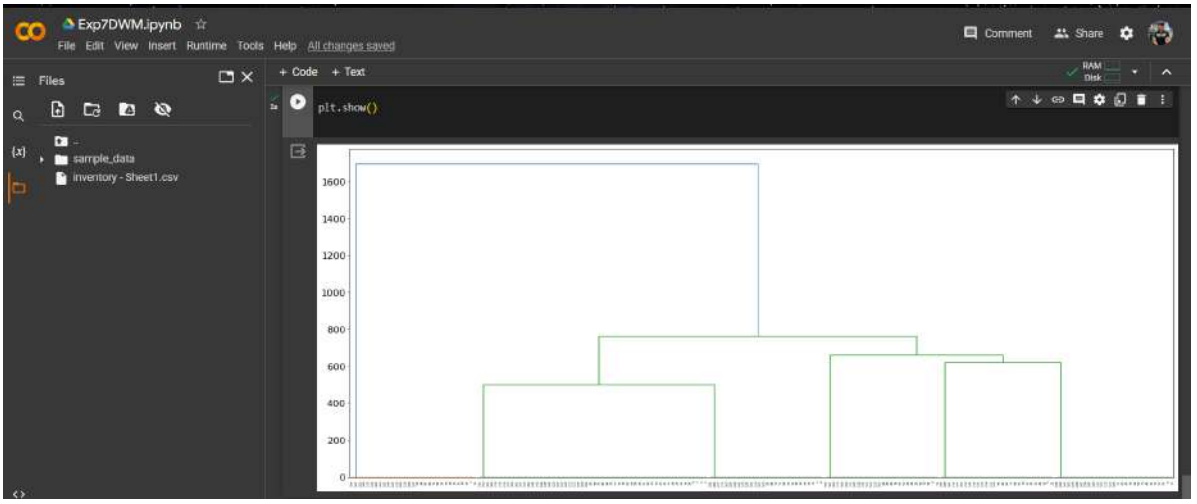
```
    # For two clusters
    def interclusterdist(self, cl, sample):
      dist = []

      for i in range(len(cl)):
          for j in range(len(sample)):
              try:
                  cl_value = float(cl[i])
                  sample_value = float(sample[j])
                  dist.append(np.linalg.norm(cl_value - sample_value))
              except (ValueError, TypeError):
                  # Handle non-numeric or missing values (e.g., NaN) here
                  pass

      if not dist:
          return float('inf')  # Return a large value for non-numeric cases
      return min(dist)



progression = [[i] for i in range(X.shape[0])]
samples = [[list(X[i])] for i in range(X.shape[0])]
m = len(samples)
distcal = Distance_computation_grid()

while m > 1:
    print('Sample size before clustering:', m)
    Distance_mat = distcal.compute_distance(samples)
    sample_ind_needed = np.where(Distance_mat == Distance_mat.min())[0]
    value_to_add = samples.pop(sample_ind_needed[1])
    samples[sample_ind_needed[0]].append(value_to_add)
    print('Cluster Node 1:', progression[sample_ind_needed[0]])
    print('Cluster Node 2:', progression[sample_ind_needed[1]])
```



```
while m > 1:
    print('Sample size before clustering:', m)
    Distance_mat = distcal.compute_distance(samples)
    sample_ind_needed = np.where(Distance_mat == Distance_mat.min())[0]
    value_to_add = samples.pop(sample_ind_needed[1])
    samples[sample_ind_needed[0]].append(value_to_add)
    print('Cluster Node 1:', progression[sample_ind_needed[0]])
    print('Cluster Node 2:', progression[sample_ind_needed[1]])
    progression[sample_ind_needed[0]].append(progression[sample_ind_needed[1]])
    progression[sample_ind_needed[0]] = [progression[sample_ind_needed[0]]]
    v = progression.pop(sample_ind_needed[1])
    m = len(samples)
    print('Progression (Current Sample):', progression)
    print('Cluster attained:', progression[sample_ind_needed[0]])
    print('Sample size after clustering:', m)
    print('\n')
```

```
Sample size before clustering: 200
Cluster Node 1: [0]
Cluster Node 2: [1]
Progression (Current Sample): [[[0, [1]]], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26
Cluster attained: [[0, [1]]]
Sample size after clustering: 199


Sample size before clustering: 199
Cluster Node 1: [[0, [1]]]
Cluster Node 2: [2]
Progression (Current Sample): [[[[0, [1]], [2]]], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25],
Cluster attained: [[[0, [1]], [2]]]
Sample size after clustering: 198
```


Files
sample_data
inventory - Sheet1.csv
Disk 80.68 GB available
+ Code + Text
from scipy.cluster.hierarchy import dendrogram, linkage
X = df[['city_id']].values
Z = linkage(X, 'single')
fig = plt.figure(figsize=(25, 10))
dn = dendrogram(Z)
plt.show()
1600
1400
1200
1000
800
600
400
200
1s completed at 12:35


Exp7DWM.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
inventory - Sheet1.csv
+ Code + Text
plt.show()
1600
1400
1200
1000
800
600
400
200
0

EXPERIMENT - 09

Aim: Implementation of Association Rule Mining algorithm (Apriori)

Theory:

Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.

The primary objective of the apriori algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions. Let's understand the apriori algorithm with the help of an example; suppose you go to Big Bazar and buy different products. It helps the customers buy their products with ease and increases the sales performance of the Big Bazar. In this tutorial, we will discuss the apriori algorithm with examples.

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of the Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating to biscuits) / (Total transactions)

= 400/4000 = 10 percent.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get confidence.

Hence,

Confidence = (Transactions relating to both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 per cent of customers who bought biscuits bought chocolates also.

Lift

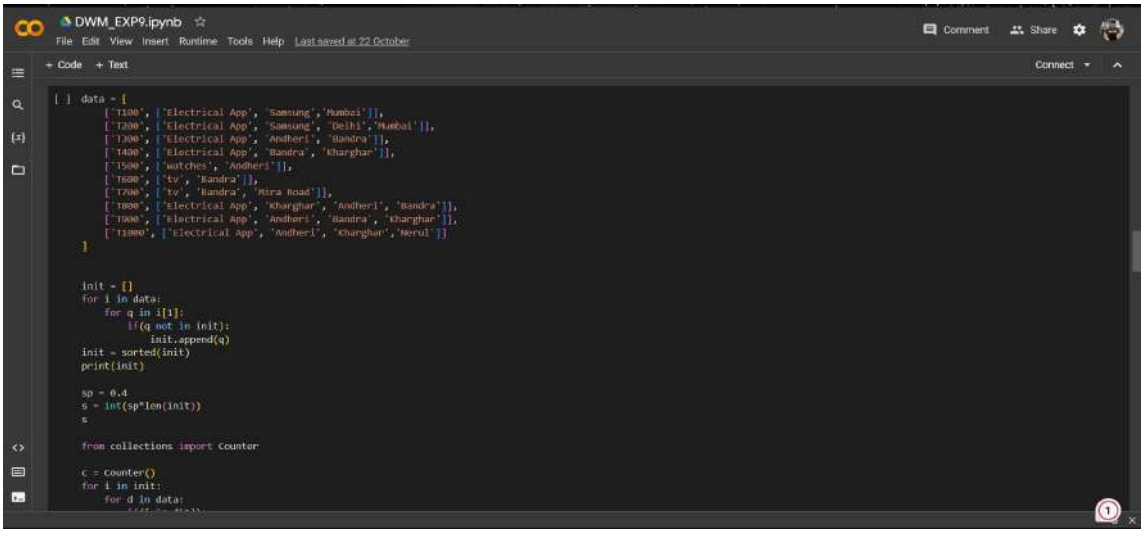
Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

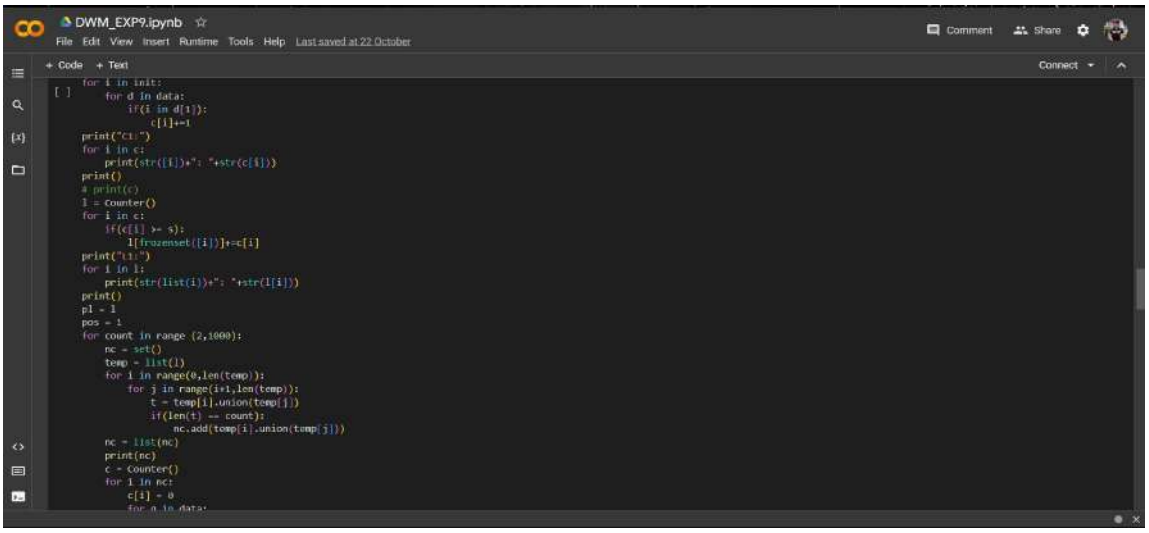
Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

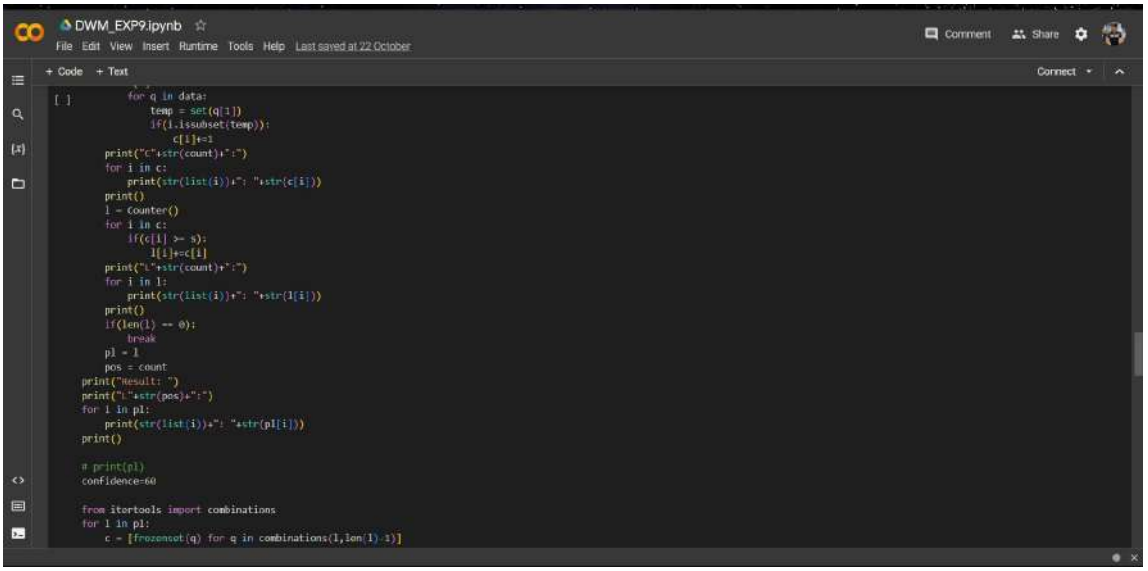
= 50/10 = 5

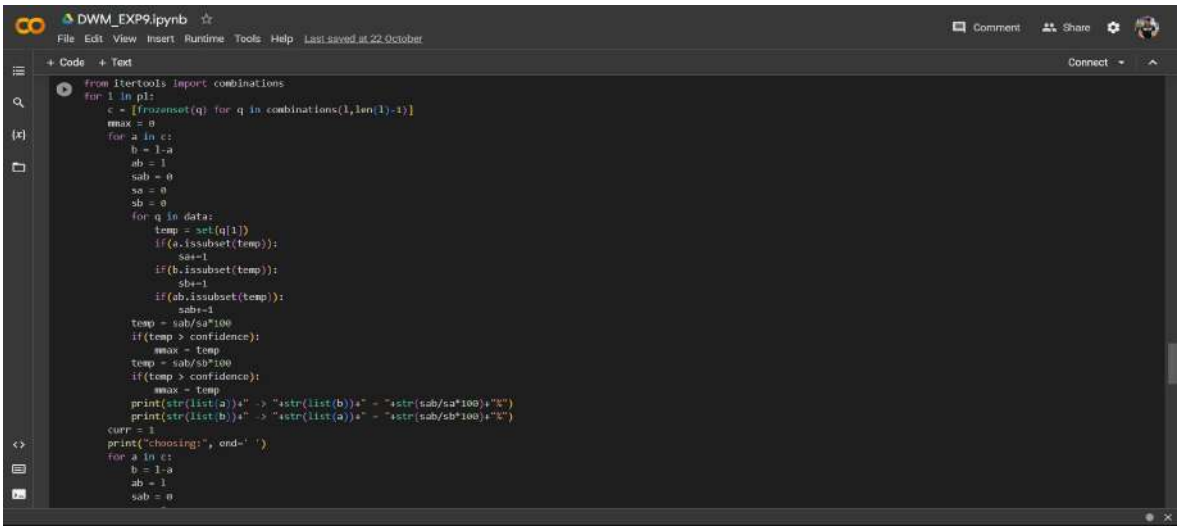
It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both items together. The larger the value, the better the combination.

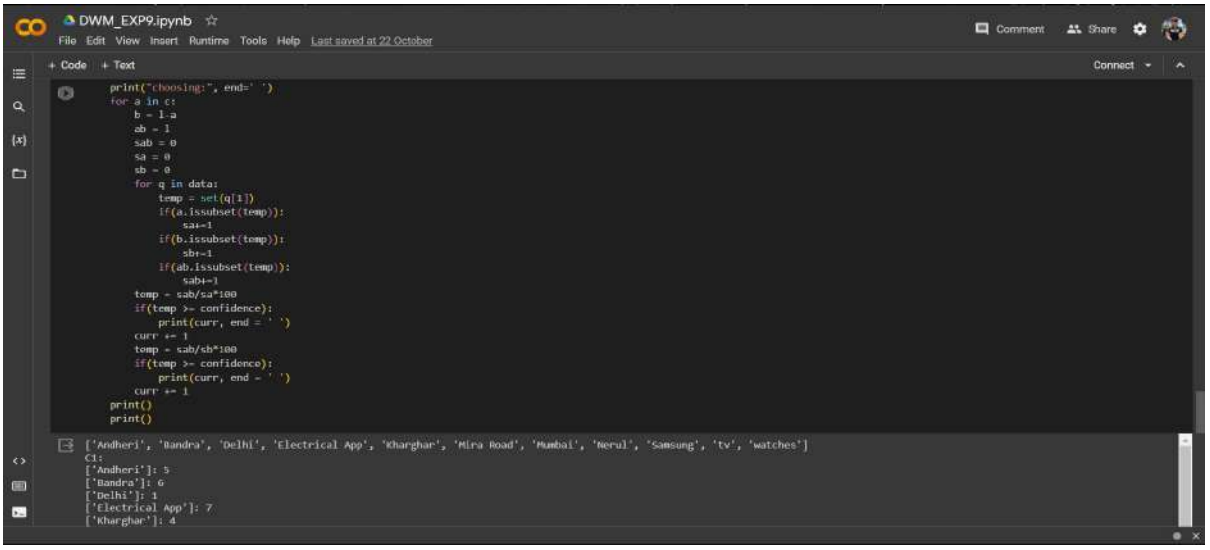
Output:

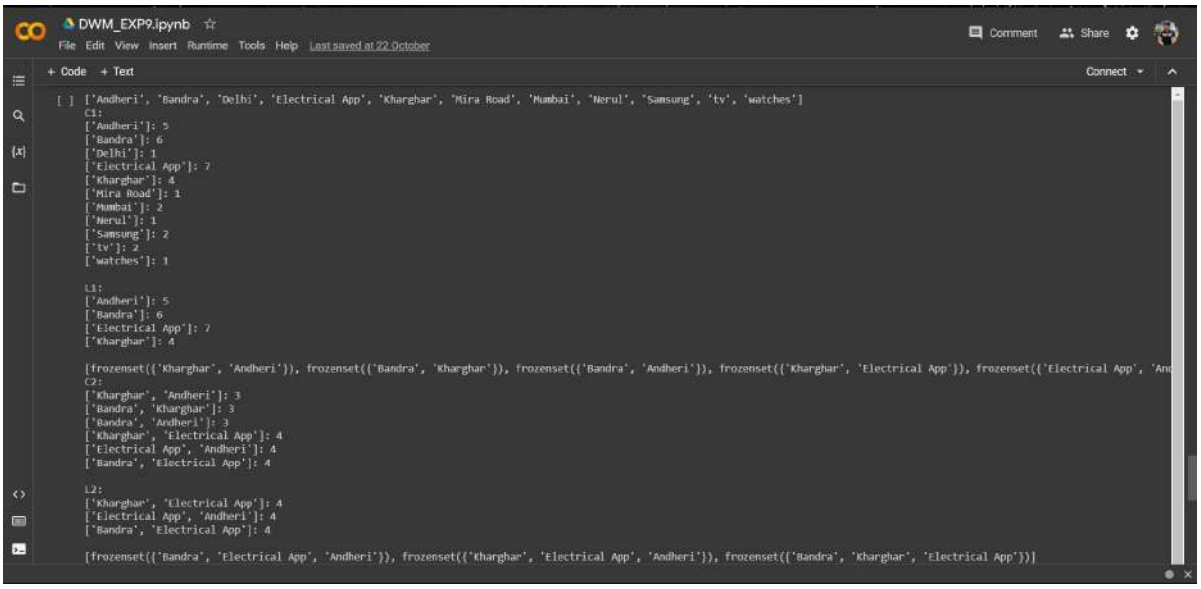


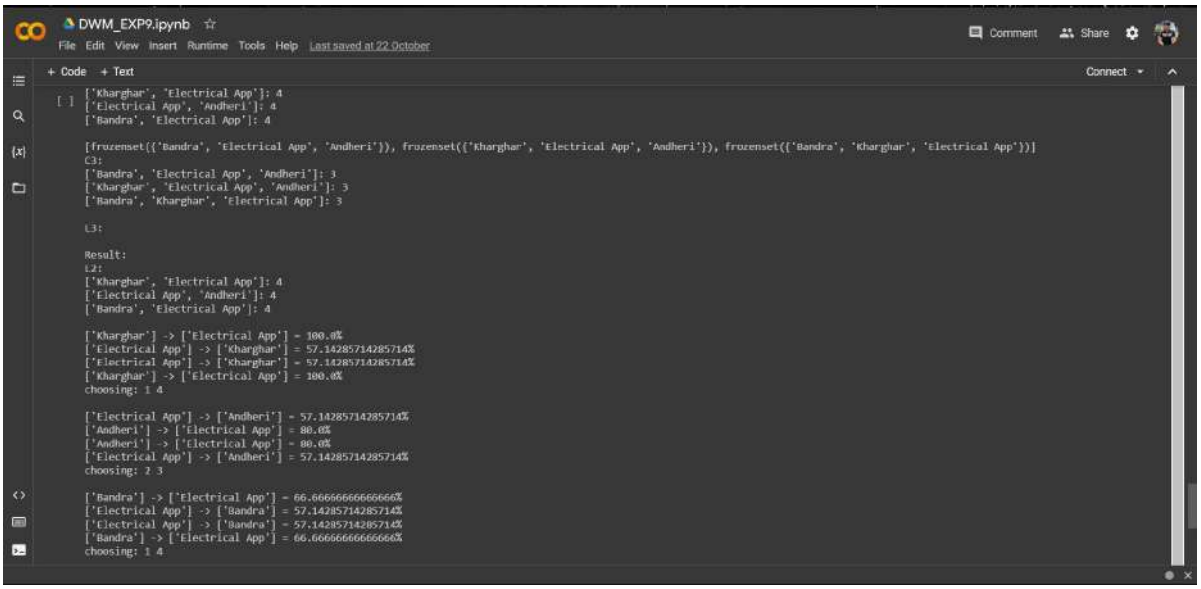



DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
Comment
Share
Connect
+ Code + Text
[ ]
for q in data:
temp = set(q[1])
if(l.issubset(temp)):
c[i]+=1
print("C"+str(count)+":")
for i in c:
print(str(list(i))+": "+str(c[i]))
print()
l = Counter()
for i in c:
if(c[i] >= s):
l[i]+=c[i]
print("L"+str(count)+":")
for i in l:
print(str(list(i))+": "+str(l[i]))
print()
if(len(l) == 0):
break
pl = l
pos = count
print("Result: ")
print("L"+str(pos)+":")
for i in pl:
print(str(list(i))+": "+str(pl[i]))
print()
# print(pl)
confidence=60
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]


DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
Comment
Share
Connect
+ Code + Text
from itertools import combinations
for l in pl:
c = [frozenset(q) for q in combinations(l,len(l)-1)]
mmax = 0
for a in c:
b = l-a
ab = l
sab = 0
sa = 0
sb = 0
for q in data:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp > confidence):
mmax = temp
temp = sab/sb*100
if(temp > confidence):
mmax = temp
print(str(list(a))+" -> "+str(list(b))+" = "+str(sab/sa*100)+"%")
print(str(list(b))+" -> "+str(list(a))+" = "+str(sab/sb*100)+"%")
curr = 1
print("choosing:", end=' ')
for a in c:
b = l-a
ab = l
sab = 0


DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
Comment Share
+ Code + Text
Connect
print("choosing:", end=' ')
for a in c:
b = l-a
ab = 1
sab = 0
sa = 0
sb = 0
for q in data:
temp = set(q[1])
if(a.issubset(temp)):
sa+=1
if(b.issubset(temp)):
sb+=1
if(ab.issubset(temp)):
sab+=1
temp = sab/sa*100
if(temp >= confidence):
print(curr, end = ' ')
curr += 1
temp = sab/sb*100
if(temp >= confidence):
print(curr, end = ' ')
curr += 1
print()
print()
['Andheri', 'Bandra', 'Delhi', 'Electrical App', 'Kharghar', 'Mira Road', 'Mumbai', 'Nerul', 'Samsung', 'tv', 'watches']
C1:
['Andheri']: 5
['Bandra']: 6
['Delhi']: 1
['Electrical App']: 7
['Kharghar']: 4


DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
Comment Share
+ Code + Text
Connect
['Andheri', 'Bandra', 'Delhi', 'Electrical App', 'Kharghar', 'Mira Road', 'Mumbai', 'Nerul', 'Samsung', 'tv', 'watches']
C1:
['Andheri']: 5
['Bandra']: 6
['Delhi']: 1
['Electrical App']: 7
['Kharghar']: 4
['Mira Road']: 1
['Mumbai']: 2
['Nerul']: 1
['Samsung']: 2
['tv']: 2
['watches']: 1
L1:
['Andheri']: 5
['Bandra']: 6
['Electrical App']: 7
['Kharghar']: 4
[frozenset({'Kharghar', 'Andheri'}), frozenset({'Bandra', 'Kharghar'}), frozenset({'Bandra', 'Andheri'}), frozenset({'Kharghar', 'Electrical App'}), frozenset({'Electrical App', 'An
C2:
['Kharghar', 'Andheri']: 3
['Bandra', 'Kharghar']: 3
['Bandra', 'Andheri']: 3
['Kharghar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4
L2:
['Kharghar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4
[frozenset({'Bandra', 'Electrical App', 'Andheri'}), frozenset({'Kharghar', 'Electrical App', 'Andheri'}), frozenset({'Bandra', 'Kharghar', 'Electrical App'})]


DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
Comment Share
+ Code + Text
Connect
['Kharghar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4
[frozenset({'Bandra', 'Electrical App', 'Andheri'}), frozenset({'Kharghar', 'Electrical App', 'Andheri'}), frozenset({'Bandra', 'Kharghar', 'Electrical App'})]
C3:
['Bandra', 'Electrical App', 'Andheri']: 3
['Kharghar', 'Electrical App', 'Andheri']: 3
['Bandra', 'Kharghar', 'Electrical App']: 3
L3:
Result:
L2:
['Kharghar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4
['Kharghar'] -> ['Electrical App'] = 100.0%
['Electrical App'] -> ['Kharghar'] = 57.142857142857142%
['Electrical App'] -> ['Kharghar'] = 57.142857142857142%
['Kharghar'] -> ['Electrical App'] = 100.0%
choosing: 1 4
['Electrical App'] -> ['Andheri'] = 57.142857142857142%
['Andheri'] -> ['Electrical App'] = 80.0%
['Andheri'] -> ['Electrical App'] = 80.0%
['Electrical App'] -> ['Andheri'] = 57.142857142857142%
choosing: 2 3
['Bandra'] -> ['Electrical App'] = 66.66666666666666%
['Electrical App'] -> ['Bandra'] = 57.142857142857142%
['Electrical App'] -> ['Bandra'] = 57.142857142857142%
['Bandra'] -> ['Electrical App'] = 66.66666666666666%
choosing: 1 4

EXPERIMENT - 10

Aim: Implementation of Page rank/HITS algorithm

Theory:

PageRank and HITS (Hyperlink-Induced Topic Search) are two important algorithms used in web search and information retrieval to rank web pages. They both aim to determine the importance or relevance of web pages based on their connections and links to other pages. Here's an overview of both algorithms:

1. PageRank Algorithm:

PageRank is an algorithm developed by Larry Page and Sergey Brin, the co-founders of Google. It is used to rank web pages based on the importance of their incoming links. The basic idea behind PageRank is that important pages are more likely to be linked to by other pages.

- PageRank treats the web as a graph, where web pages are nodes, and hyperlinks between them are edges.
- Each page is assigned an initial PageRank value. Typically, all pages are given equal initial PageRank values.
- PageRank is calculated iteratively. At each iteration, each page distributes a fraction of its PageRank to the pages it links to.
- The PageRank of a page is determined by the PageRank of the pages that link to it. A link from a page with high PageRank is more valuable.
- Pages that have more incoming links and are linked from important pages have higher PageRank values.
- The PageRank of a page converges to a stable value after many iterations, and this value represents the page's importance.

Mathematically, the PageRank of a page 'P' is calculated as follows:

PR(P) = (1 - d) + d * (PR(Q1)/L(Q1) + PR(Q2)/L(Q2) + ... +

PR(Qn)/L(Qn))

Where:
- PR(P) is the PageRank of page P.
- d is a damping factor (usually set to 0.85) to account for the probability that a user might randomly jump to any page.
- PR(Q1), PR(Q2), ..., PR(Qn) are the PageRank values of pages that link to page P.

- L(Q1), L(Q2), ..., L(Qn) are the number of outbound links on pages Q1, Q2, ..., Qn.

2. HITS (Hyperlink-Induced Topic Search) Algorithm:

HITS, also known as "Hubs and Authorities," is an algorithm that assesses web page authority and hub status. It was developed by Jon Kleinberg.

- In the HITS algorithm, each web page is assigned two scores: a hub score and an authority score.
- Hub pages are those that link to many authority pages, while authority pages are those linked to by many hub pages.
- The HITS algorithm iteratively updates the hub and authority scores until they converge to stable values.
- The final hub and authority scores provide a way to rank pages based on their role as hubs and authorities in the web graph.

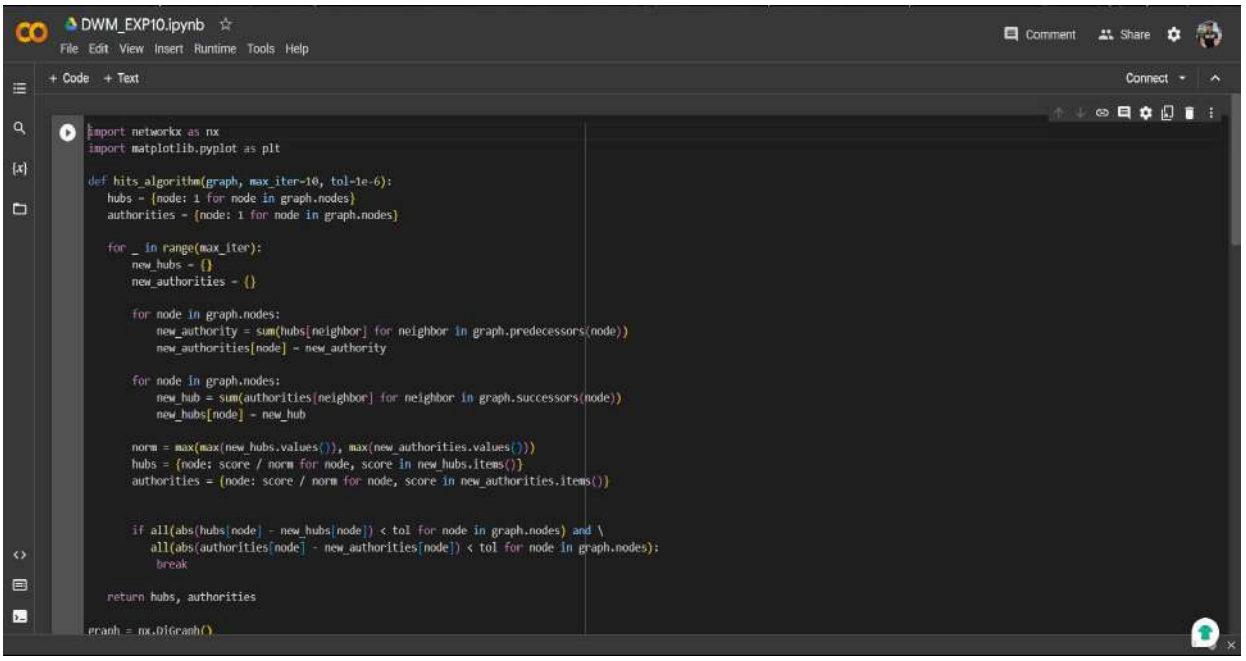
Mathematically, the HITS algorithm works as follows:

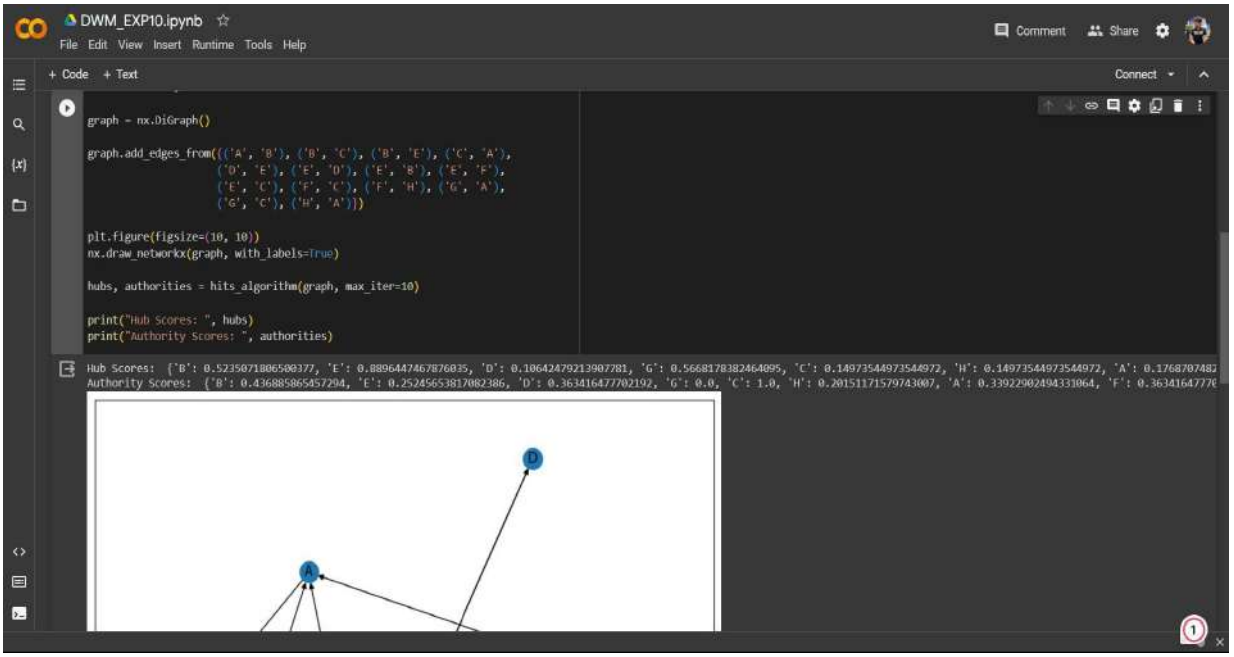
1. Initialize all hub and authority scores to 1.
2. Iteratively update the hub and authority scores based on the links between pages.
3. Normalize the hub and authority scores to ensure they sum to 1.

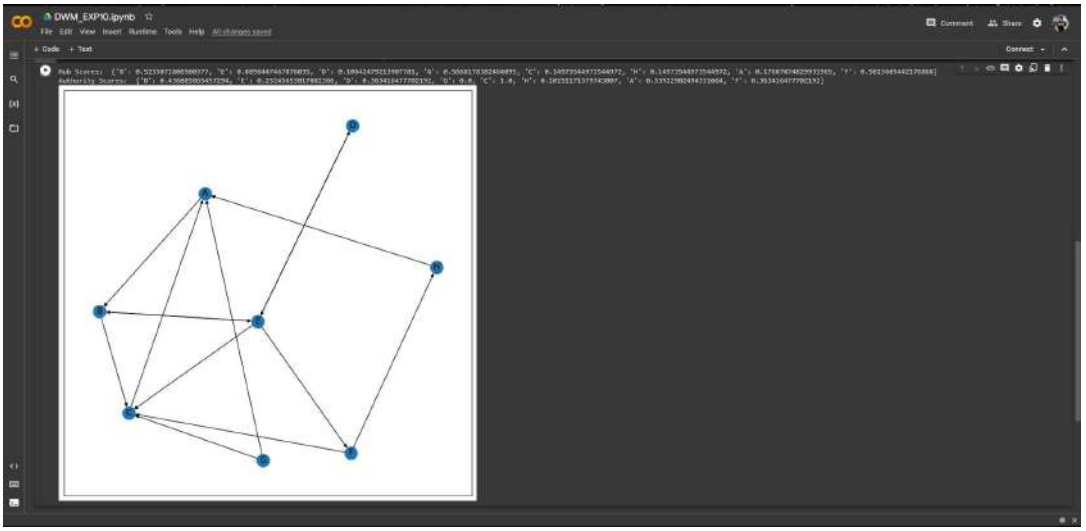
HITS algorithm can be used to find not only the most authoritative pages but also to find good hubs that link to these authoritative pages.

Both PageRank and HITS are foundational algorithms in web search and information retrieval. PageRank is used by Google in its search engine, and HITS provides valuable insights into the link structure of the web. These algorithms have evolved and are part of a broader set of techniques used to rank web pages and discover relevant content on the internet.

Output:


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
import networkx as nx
import matplotlib.pyplot as plt
def hits_algorithm(graph, max_iter=10, tol=1e-6):
hubs = {node: 1 for node in graph.nodes}
authorities = {node: 1 for node in graph.nodes}
for _ in range(max_iter):
new_hubs = {}
new_authorities = {}
for node in graph.nodes:
new_authority = sum(hubs[neighbor] for neighbor in graph.predecessors(node))
new_authorities[node] = new_authority
for node in graph.nodes:
new_hub = sum(authorities[neighbor] for neighbor in graph.successors(node))
new_hubs[node] = new_hub
norm = max(max(new_hubs.values()), max(new_authorities.values()))
hubs = {node: score / norm for node, score in new_hubs.items()}
authorities = {node: score / norm for node, score in new_authorities.items()}
if all(abs(hubs[node] - new_hubs[node]) < tol for node in graph.nodes) and \
all(abs(authorities[node] - new_authorities[node]) < tol for node in graph.nodes):
break
return hubs, authorities
graph = nx.DiGraph()


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
graph = nx.DiGraph()
graph.add_edges_from([('A', 'B'), ('B', 'C'), ('B', 'E'), ('C', 'A'),
('D', 'E'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
('E', 'C'), ('F', 'C'), ('F', 'H'), ('G', 'A'),
('G', 'C'), ('H', 'A')])
plt.figure(figsize=(10, 10))
nx.draw_networkx(graph, with_labels=True)
hubs, authorities = hits_algorithm(graph, max_iter=10)
print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
Hub Scores: {'B': 0.523507180650377, 'E': 0.889644746787603S, 'D': 0.10642479213907781, 'G': 0.5668178382464095, 'C': 0.14973544973544972, 'H': 0.14973544973544972, 'A': 0.176870748
Authority Scores: {'B': 0.436885865457294, 'E': 0.25245653817082386, 'D': 0.363416477702192, 'G': 0.0, 'C': 1.0, 'H': 0.20151171579743007, 'A': 0.3392290249433106, 'F': 0.36341647776
A
D


DWM_EXP10.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Comment
Share
Connect

THADOMAL SHAHANI ENGINEERING COLLEGE
TSEC
Om Shele
TE C32
2103163

Assignment 1

What is Metadata? Explain data warehouse with Example.

- Data warehouse metadata is a piece of information stored in one or more special-purpose metadata repository that include:

1) Information about data warehouse, their structure and their location.
2) Information about refreshment of warehouse cleanup
3) Information regarding characteristics of components.

- Types of Metadata :

1) Operational Metadata

⇒

- In an enterprise, data for the data warehouse comes from various operational systems.
- Different source systems contain different data structures having different field lengths and data types.
- So the information of operational data source is given by operational metadata.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2) Extraction & Transformation Metadata

⇒

- This contains the information about the extraction of data from heterogenous source system.
- It also contains information about data transformation in data staging area.

3) End-user Metadata

⇒

- This particular category provide the end user the flexibility of looking for information in their own way.

- For e.g.:,

1) A library catlog may be considered metadata. The directory metadata consists of several components representing specific attribute of resource and each item can have one or more values. These components could be the name of author, name of publishment document, the publication, date, etc.

2) The table of content and index in a book may be as treated metadata for the book.

23/12/23

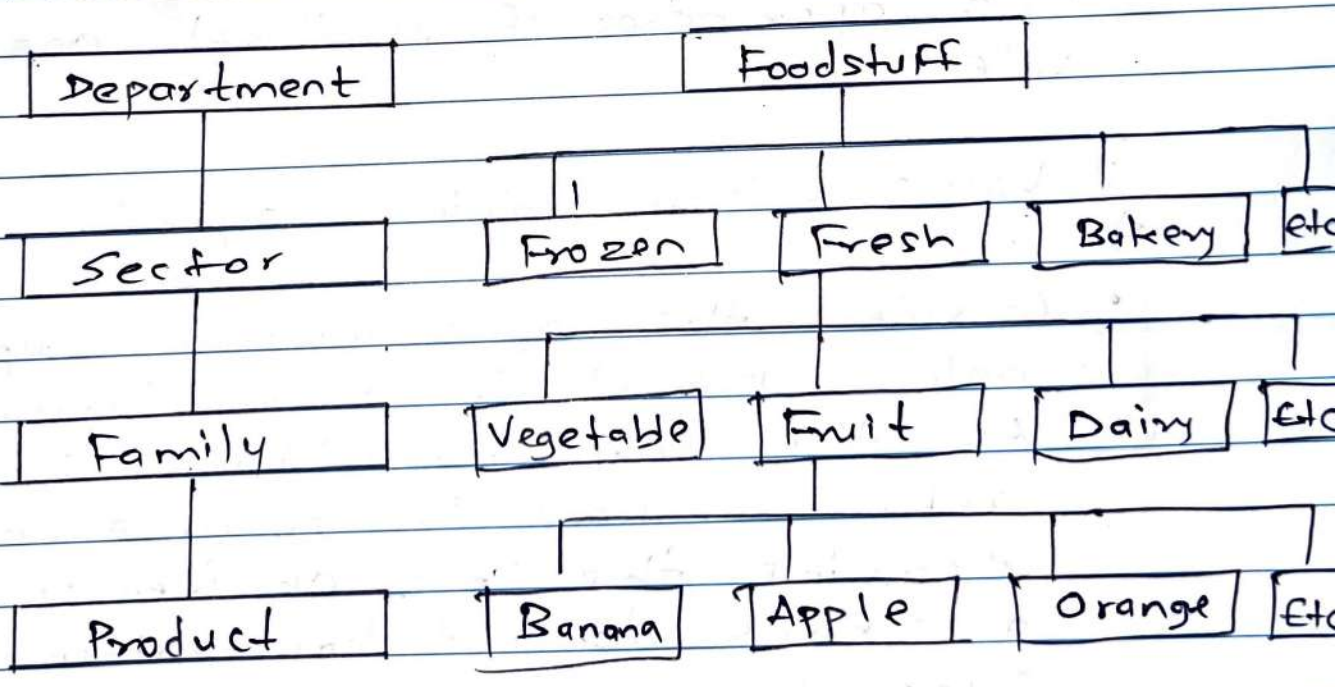
Assignment 2

Q.1 Write short note on

a) Multilevel Association Rules and Multidimensional association rules

⇒

- Items are always in the form of hierarchy.
- Items which are at leaf nodes are having lower support.
- An item can be either generalized or specialized as per the described hierarchy of that item and its level can be powerfully preset in transactions.
- Rules which combines associations with hierarchy of concepts are called Multilevel Association Rules.



(Hierarchy of Concept)

- Support and confidence of multilevel association rules :

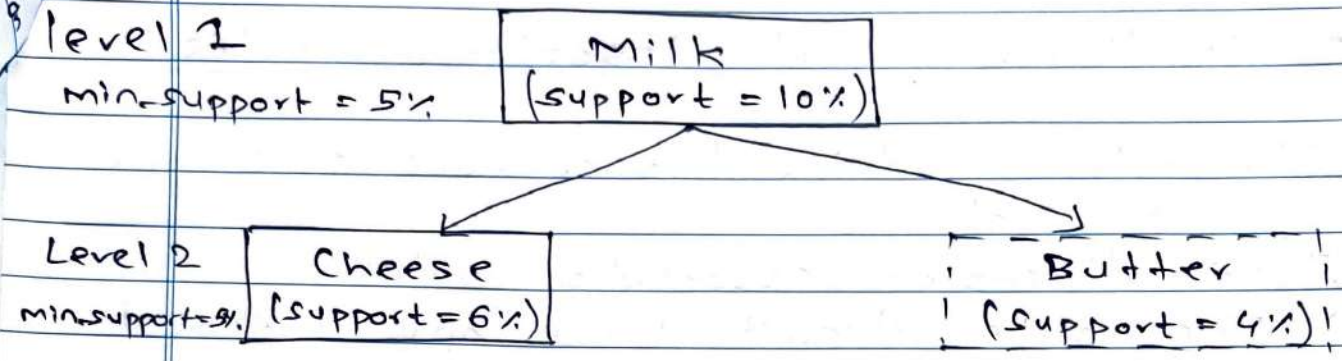
- The support and confidence of an item is affected due to its generalized or specialized value of attributes.
- The support of generalized item is more than the support of specialized item.
- Similarly the support of rules increases from specialized to generalized itemsets.
- If the support is below the threshold value then that rules become invalid.
- Confidence is not affected for general or specialized.

- Two approaches of multilevel association rules :

1) Using uniform minimum support for all levels

⇒

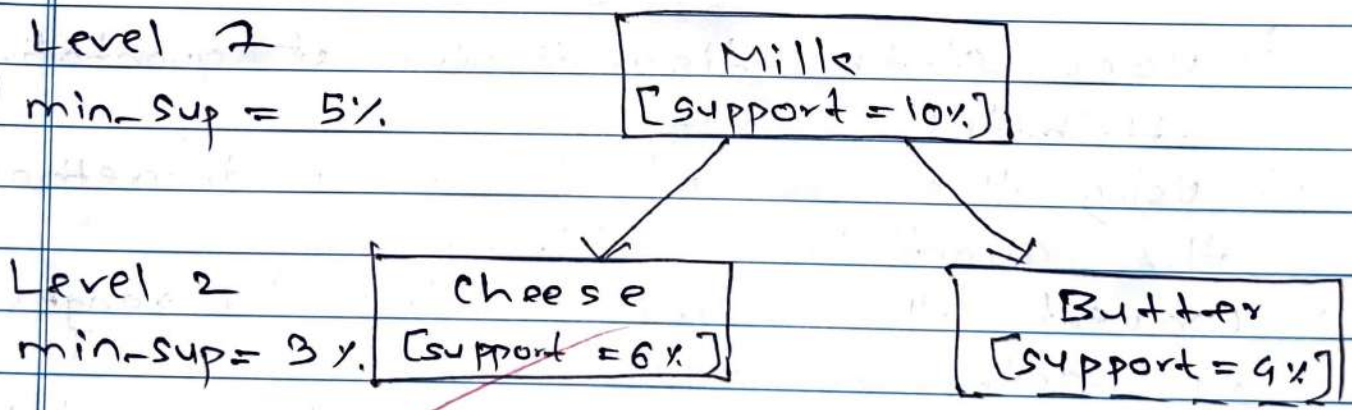
- Consider the same minimum support is below for all levels of hierarchy.
- As the only one minimum support is set, so there is no necessary to examine the items of itemset whose ancestors do not have minimum support.



2) Using reduced minimum support at lowest level

⇒

- Consider separate minimum support at each level of hierarchy.
- As the every level is having its own minimum support, the support at lowest level reduces.



• Multidimensional Association Rules :

⇒

- The rules contains two or more dimensions or predicates.
- Inter-dimension association rules
  ⇒ The rule doesn't have any repeated predicate.

- Hybrid-dimension association rules
  ⇒ The rule have many occurences of same predicate, i.e. buys.

gender (X, "Male") ∧ buys (X, "TV")
⇒ buys (X, "DVD").

- Techniques of Mining MD Associations:

1) Using static discretization of quantative attributes
   ⇒ Using the concept hierarchy, discretizes the quantitative attributes.
- Convert the numeric values by ranges or categorical values.
- Mining is always more faster on data cube.
- Predicates sets are determined by the cells of n-dimensional cuboid.

2) Quantitative association rules:

⇒

- Numeric attributes are dynamically discretized such that the confidence of the rules mined is maximized.
- In quantitative association rule, the left hand side of rule contains 2-D quantitative attributes and right hand side of rules contains one categorical attribute.

$A_{quantitative1} \wedge A_{quantitative2} \Rightarrow$ Categorical

3) Distance-based association rules

⇒

- This is a dynamic discretization process that considers the distance between data points.
- This mining process has only two steps:
  - Perform clustering to find the interval of attributes involved.
  - Obtain association rules by searching for groups of clusters that occur together.

b) Web Usage Mining

⇒

- Web usage mining is the type of web mining which predicts about which pages are likely to be visited in near future based on the active user's behavior.
- Such pages can be pre-fetched to reduce access times.
- The usage data records the user's behaviour when the user browsers or makes transactions on the web site in order to better understand and server the needs of users or Web-based applications.
- Automatic discovery of patterns from one or more web servers.
- Organization often generate and collect large volumes of data can help these organizations to determine:
  - The value of particular customers.
  - Cross marketing strategies across products.
  - The effectiveness of promotional campaigns, etc.
- The first web analysis tools simply provide mechanisms to report user activity as recorded in the servers.
- Using such tools, it was possible to determine such information as:

- The number of accesses to the server.
- The times or time intervals of visits.
- The domain names and the URLs of users of the Web Server.

- A very little or no analysis is provided by these tools of the data relationships among the accessed files and the directories within web spaces.

3. - The tools for discovery and analysis of patterns may be classified into following two categories:
  - Pattern Discovery Tools
  - Pattern Analysis Tools.
- Web server, web proxies and client applications can quite easily capture web usage data.
- Web server log: It is a file that is created by the server to record all the activities it performs.
- For example, when a user enters URL into the browsers address bar a requests by clicking on a link.

23/10/25