Experiment - 1

Aim: To understand the data through Exploratory data analysis

- Data cleaning-Missing Values,remove outliers
- Data transformation - Min-max normalisation,Z-score normalisation,Decimal normalisation
- Data Discretization- Binning
- Data analysis and Visualization

Steps:

1) Load the libraries Download the data set from kaggle/ other sources
2) Read the file –select appropriate file read function according to data type of file 3) Describe the attributes name, count no of values, and find min, max, data type, range, quartile, percentile, box plot and outliers.
4) Perform cleaning,transformation , discretization and analysis
4) Give visualisation of statistical description of data – in form of histogram, scatter plot, pie chart,Give correlation matrix

Theory:

1. Data Preprocessing:

Data preprocessing is a crucial phase in the realm of data warehouse management (DWM), where raw data is transformed and refined into a structured format suitable for analysis and reporting. Effective data preprocessing sets the foundation for accurate insights and informed decision-making. In this context, here are eight essential points to consider when undertaking data preprocessing within a data warehouse management framework.

Points for Data Preprocessing in DWM:

a.Data Collection and Integration:

Data preprocessing begins with collecting data from various sources, including databases, APIs, and external files. Integrating this diverse data ensures a comprehensive view and minimizes data silos.

b.Data Cleaning:

Cleaning involves identifying and handling inconsistencies, missing values, and errors in the data. Imputing missing values and rectifying anomalies ensures that downstream analyses are not compromised by flawed data.

c.Data Transformation

Data often requires transformation to be usable. This could involve normalisation (scaling data to a standard range), encoding categorical variables, or aggregating data to a suitable granularity for analysis.

d.Data Deduplication:

Duplicate records can distort analysis results. Implementing deduplication techniques ensures that the same data point is not counted multiple times, improving accuracy.

e.Data Reduction:

In cases of large datasets, data reduction techniques like dimensionality reduction (PCA, t-SNE) can help retain essential information while reducing computational complexity.

f.Outlier Detection and Handling:

Outliers can skew analysis and modeling results. Identifying and dealing with outliers through techniques like statistical tests or clustering can lead to more reliable insights.

g.Data Formatting and Validation:

Ensuring data consistency and adherence to defined formats is crucial. Validation checks are applied to confirm that the data meets the required standards before it's loaded into the data warehouse.

2.Data Transformation:

Data transformation is a critical process in data preprocessing that involves converting raw data into a more suitable format for analysis, reporting, and modeling. It enhances the usability of the data by standardizing, scaling, and reorganizing it. Data transformation enables better insights and more accurate decision-making by preparing the data to be compatible with various algorithms and analytical techniques.

Five Points on Data Transformation:

a.Normalisation:

Normalisation scales numerical data to a common range, usually between 0 and 1. This eliminates the impact of differing scales on algorithms and ensures that each feature contributes equally to analysis.

b.Encoding Categorical Variables:

Categorical variables, such as gender or product category, need to be encoded into numerical values for analysis. Techniques like one-hot encoding or label encoding are used to represent categorical data in a format that algorithms can process.

c.Aggregation:

Aggregating data involves summarizing information by grouping it based on certain attributes. For instance, sales data might be aggregated by month to analyze monthly trends. Aggregation simplifies complex datasets and makes them more manageable.

d.Feature Creation:

Feature creation involves generating new attributes from existing data. These new features can capture patterns that are not immediately evident. For example, creating a "customer loyalty score" based on purchase frequency and total spending can provide valuable insights.

e.Binning and Discretization:

Binning involves dividing continuous data into discrete intervals or bins. This can simplify complex datasets and reveal trends that might not be apparent in the raw data. Binning can be particularly useful when dealing with numerical data that has a wide range.

3.Data Discretization:

Data discretization is a data transformation technique that involves converting continuous data into a discrete format by grouping values into intervals or bins. This process simplifies complex data, reduces noise, and can make it easier to uncover patterns and trends that might not be immediately apparent in the raw data.

Points on Data Discretization:

a.Benefits of Discretization

Discretization can simplify analysis and modeling processes by converting continuous data into categories. This can make data more manageable, especially when dealing with large datasets. Additionally, it can reduce the impact of outliers and small fluctuations in the data.

b.Methods of Discretization:

There are various methods for discretizing data, including equal width binning, equal frequency binning, and clustering-based binning. Equal width binning divides the data range into intervals of equal width, while equal frequency binning ensures each interval contains approximately the same number of data points. Clustering-based binning groups data based on similarity measured by clustering algorithms.

c.Impact on Analysis and Interpretation:

Discretization can affect the results of data analysis and modelling. While it simplifies data, it may also lead to loss of information due to grouping similar values together. The choice of binning method and the number of bins should be guided by the underlying characteristics of the data and the objectives of the analysis.

4.Data Visualization:

Data visualisation is the practice of representing data in graphical or visual formats to aid understanding, analysis, and communication of insights. By translating raw data into intuitive visual representations, data visualisation enhances the ability to identify trends, patterns, and relationships within the data, making complex information more accessible and actionable.

Points on Data Visualization:

a.Enhanced Understanding:

Data visualisation transforms abstract data into visual cues, enabling viewers to quickly grasp information and identify trends that might not be apparent in raw data. It provides a clear overview of data distributions, correlations, and anomalies, making it easier to derive insights.

b.Effective Communication:

Visualisations are powerful tools for communicating findings to both technical and non-technical audiences. Complex data can be presented in a digestible manner, allowing stakeholders to make informed decisions based on the visual representation of information.

c.Visualisation Types:

There are various types of visualisations, each suited for different data characteristics and objectives. Common types include bar charts, line graphs, scatter plots, histograms, and heatmaps. Choosing the right visualisation type depends on the data's nature and the specific insights you want to convey.

Implementation:

```
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('/content/swiggy.csv')

data.head()


df.dropna(inplace=True)

data.fillna(1000000000, inplace=True)

data.drop_duplicates(inplace=True)
```

Output:

```
ID,Area,City,Restaurant,Price,Avg ratings,Total ratings,Food type,Address,Delivery time
211,Koramangala,Bangalore,Tandoor Hut,300.0,4.4,100,"Biryani Chinese North Indian South Indian",5Th Block,59
221,Koramangala,Bangalore,Tunday Kababi,300.0,4.1,100,"Mughlai Lucknowi",5Th Block,56
246,Jogupalya,Bangalore,Kim Lee,650.0,4.4,100,Chinese,Double Road,50
248,Indiranagar,Bangalore,New Punjabi Hotel,250.0,3.9,500,"North Indian Punjabi Tandoor Chinese",80 Feet Road,57
249,Indiranagar,Bangalore,Nh8,350.0,4.0,50,"Rajasthani Gujarati North Indian Snacks Desserts Beverages Thalis Chaat",80 Feet Road,63
254,Indiranagar,Bangalore,Treat,800.0,4.5,100,"Mughlai North Indian",100 Feet Road,56
258,Indiranagar,Bangalore,Chinita Real Mexican Food,1000.0,4.5,500,"Mexican Beverages Salads",Double Road,53
263,Koramangala,Bangalore,Cupcake Noggins - Cakespastries And Desserts,150.0,4.3,100,"Desserts British Bakery Pizzas Snacks",4Th Block
267,Domlur,Bangalore,Tea Brew,350.0,4.1,100,"American Italian Beverages Continental Chinese Pastas Pizzas Fast Food",Double Road,57
308,Koramangala,Bangalore,Bangaliana,300.0,4.0,500,Bengali,7Th Block,57
435,Koramangala,Bangalore,Bamey'S Restro Cafe,400.0,4.4,100,"Chinese Thai Nepalese",5Th Block,56
446,Cooke Town,Bangalore,Altaf'S Chillies Restaurants,250.0,4.3,500,"North Indian Chinese",Cooke Town,55
453,Pulikeshi Nagar,Bangalore,Chichabas Taj,532.0,4.2,100,"Mughlai North Indian Biryani",Mm Road,54
464,Sivanchetti Gardens,Bangalore,Urban Solace,500.0,2.9,80,"American Continental",Ulsoor Lake,48
504,Kodihalli,Bangalore,Kaati Zone Rolls & Wraps,150.0,4.3,100,"Fast Food Beverages Bengali Biryani Indian North Indian Snacks Street
604,Koramangala,Bangalore,Aalishan Restaurant & Caterer,290.0,4.3,100,"Mughlai North Indian",7Th Block,52
608,Domlur,Bangalore,Kerala Pavilion,300.0,4.1,100,"Kerala Seafood",Domlur Layout,55
621,Indiranagar,Bangalore,Kitchen Of Joy,250.0,4.4,100,"Bengali Fast Food Chinese",Domlur,55
1050,Jayanagar,Bangalore,Nagarjuna Chimney,800.0,4.2,1000,"Seafood Andhra Biryani South Indian Combo Desserts Beverages",3Rd Block,48
1299,Film Nagar,Hyderabad,So. The Sky Kitchen,800.0,2.9,80,"Mughlai North Indian Chinese Thai Asian",Near Apollo Health City Gate,90
1304,Indiranagar,Bangalore,Esplanade,1200.0,4.3,100,"Mughlai Bengali",I Indiranagar,54
```

```python
numerical_columns = ['Price','Total ratings','Delivery time']

def min_max_normalization(column):
  return (column-column.min())/(column.max()-column.min())

data_min_max = data.copy()
data_min_max[numerical_columns] = data_min_max[numerical_columns].apply(min_max_normalization)

print(data_min_max[numerical_columns])

def z_score_normalization(column):
  return (column-column.mean())/column.std()

data_z_score = data.copy()
data_z_score[numerical_columns] = data_z_score[numerical_columns].apply(z_score_normalization)

print(data_z_score[numerical_columns])

def decimal_normalization(column):
  max_value = column.max()
  num_decimals = len(str(int(max_value)))
  divisor = 10 ** num_decimals
  return column / divisor

data_decimal = data.copy()
data_decimal[numerical_columns] = data_decimal[numerical_columns].apply(decimal_normalization)
print(data_decimal[numerical_columns])

data_min_max.to_csv('normalized_data_min_max.csv', index=False)

data_z_score.to_csv('normalized_data_z_score.csv', index=False)
data_decimal.to_csv('normalized_data_decimal.csv', index=False)
```

Output:

Min-Max

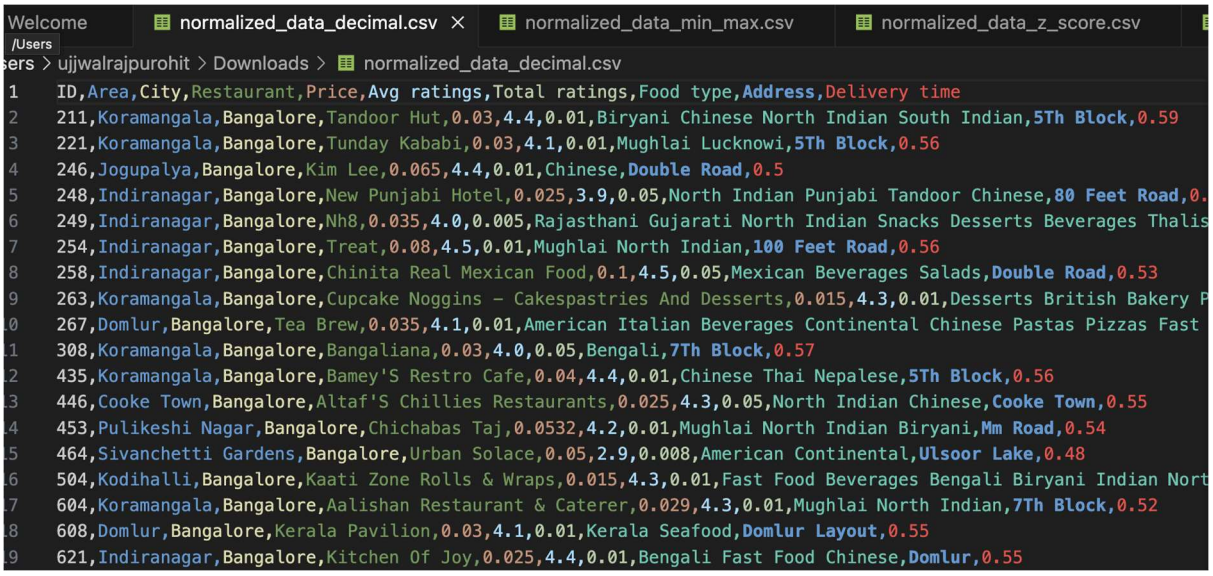
lcome | normalized_data_decimal.csv | normalized_data_min_max.csv × | normalized_data_z_score.csv

> ujjwalrajpurohit > Downloads > normalized_data_min_max.csv

```
ID,Area,City,Restaurant,Price,Avg ratings,Total ratings,Food type,Address,Delivery time
211,Koramangala,Bangalore,Tandoor Hut,0.14285714285714285,4.4,0.01010101010101010102,Biryani Chinese North Indian
221,Koramangala,Bangalore,Tunday Kababi,0.14285714285714285,4.1,0.01010101010101010102,Mughlai Lucknowi,5Th Block
246,Jogupalya,Bangalore,Kim Lee,0.47619047619047616,4.4,0.01010101010101010102,Chinese,Double Road,0.3333333333
248,Indiranagar,Bangalore,New Punjabi Hotel,0.09523809523809523,3.9,0.09090909090909091,North Indian Punjabi Ta
249,Indiranagar,Bangalore,Nh8,0.19047619047619047,4.0,0.0,Rajasthani Gujarati North Indian Snacks Desserts Beve
254,Indiranagar,Bangalore,Treat,0.6190476190476191,4.5,0.01010101010101010102,Mughlai North Indian,100 Feet Road
258,Indiranagar,Bangalore,Chinita Real Mexican Food,0.8095238095238095,4.5,0.09090909090909091,Mexican Beverage
263,Koramangala,Bangalore,Cupcake Noggins - Cakespastries And Desserts,0.0,4.3,0.01010101010101010102,Desserts Br
267,Domlur,Bangalore,Tea Brew,0.19047619047619047,4.1,0.01010101010101010102,American Italian Beverages Continent
308,Koramangala,Bangalore,Bangaliana,0.14285714285714285,4.0,0.09090909090909091,Bengali,7Th Block,0.45
435,Koramangala,Bangalore,Bamey'S Restro Cafe,0.23809523809523808,4.4,0.01010101010101010102,Chinese Thai Nepales
446,Cooke Town,Bangalore,Altaf'S Chillies Restaurants,0.09523809523809523,4.3,0.09090909090909091,North Indian
453,Pulikeshi Nagar,Bangalore,Chichabas Taj,0.3638095238095238,4.2,0.01010101010101010102,Mughlai North Indian Bi
464,Sivanchetti Gardens,Bangalore,Urban Solace,0.3333333333333333,2.9,0.006060606060606061,American Continenta
504,Kodihalli,Bangalore,Kaati Zone Rolls & Wraps,0.0,4.3,0.01010101010101010102,Fast Food Beverages Bengali Birya
604,Koramangala,Bangalore,Aalishan Restaurant & Caterer,0.13333333333333333,4.3,0.01010101010101010102,Mughlai No
608,Domlur,Bangalore,Kerala Pavilion,0.14285714285714285,4.1,0.01010101010101010102,Kerala Seafood,Domlur Layout
621,Indiranagar,Bangalore,Kitchen Of Joy,0.09523809523809523,4.4,0.01010101010101010102,Bengali Fast Food Chinese
1050,Jayanagar,Bangalore,Nagarjuna Chimney,0.6190476190476191,4.2,0.1919191919191919,Seafood Andhra Biryani Sou
```

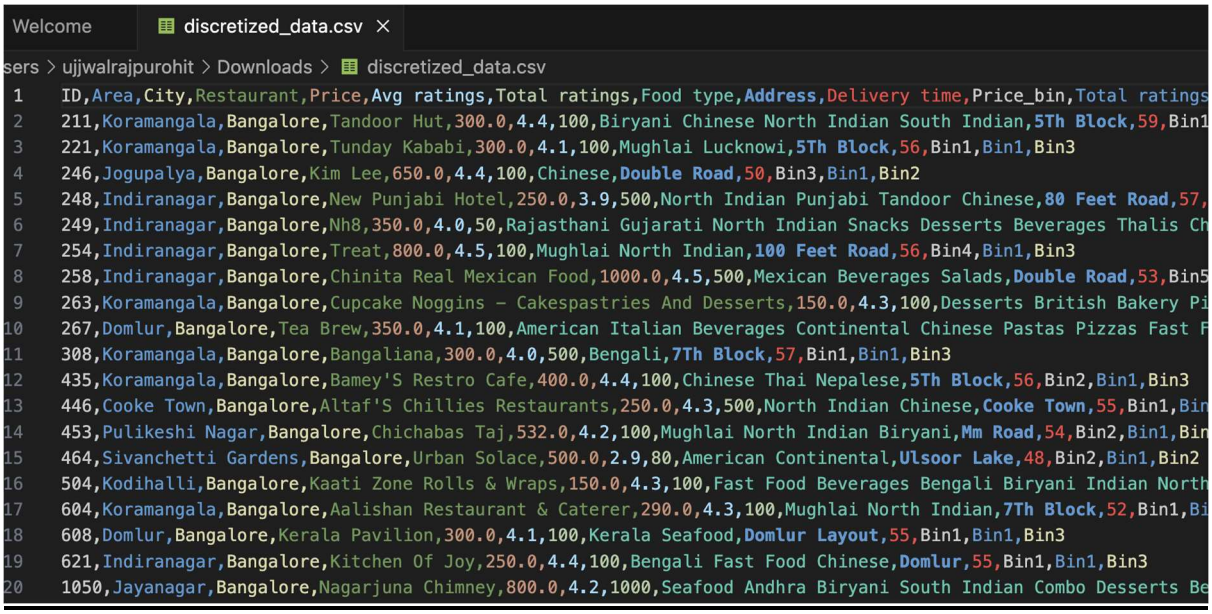
Z-Score

Decimal



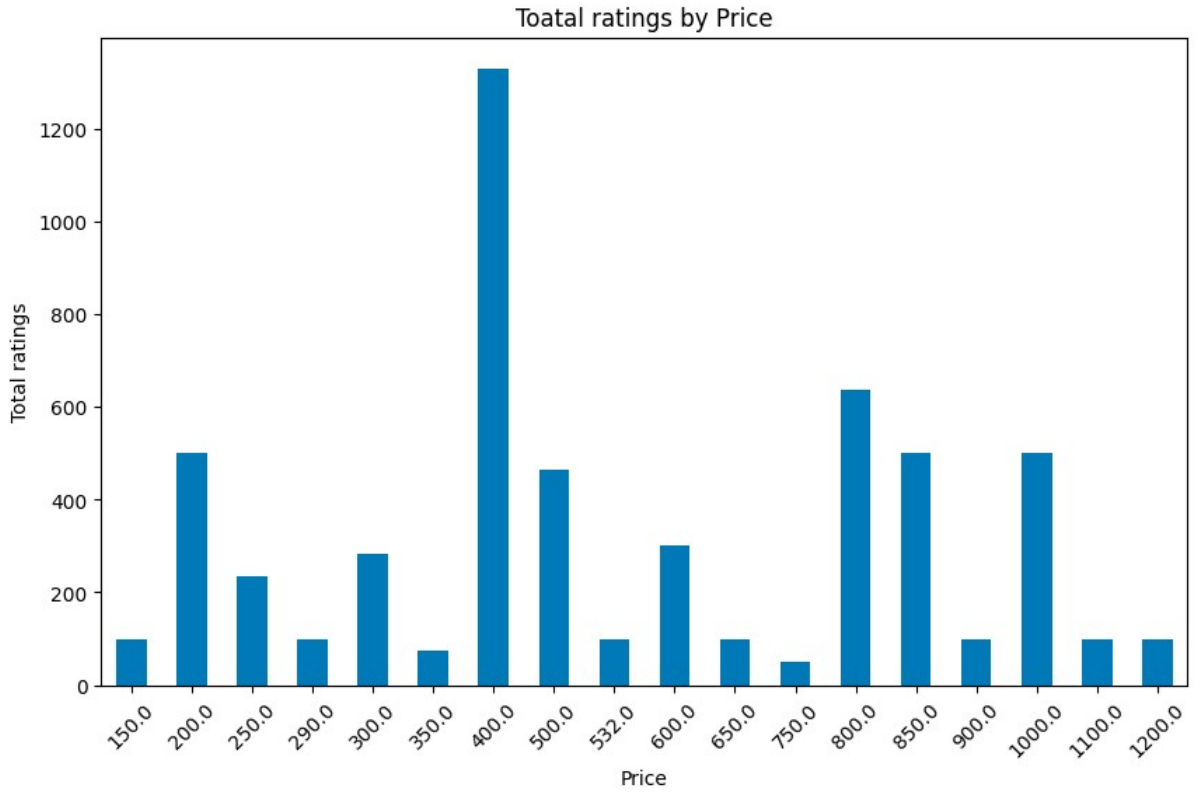
```
num_bins = 5

columns_to_discretize = ['Price','Total ratings','Delivery time']

def equal_width_binning(column, num_bins):
  bin_width = (column.max() - column.min()) / num_bins
  bins = [column.min() + i * bin_width for i in range(num_bins +1)]
  labels = [f'Bin{i+1}' for i in range(num_bins)]
  return pd.cut(column,bins=bins,labels=labels,include_lowest=True)

for column in columns_to_discretize:
  data[f'{column}_bin'] = equal_width_binning(data[column],num_bins)
  print(data[f'{column}_bin'])

data.to_csv('discretized_data.csv', index=False)
```

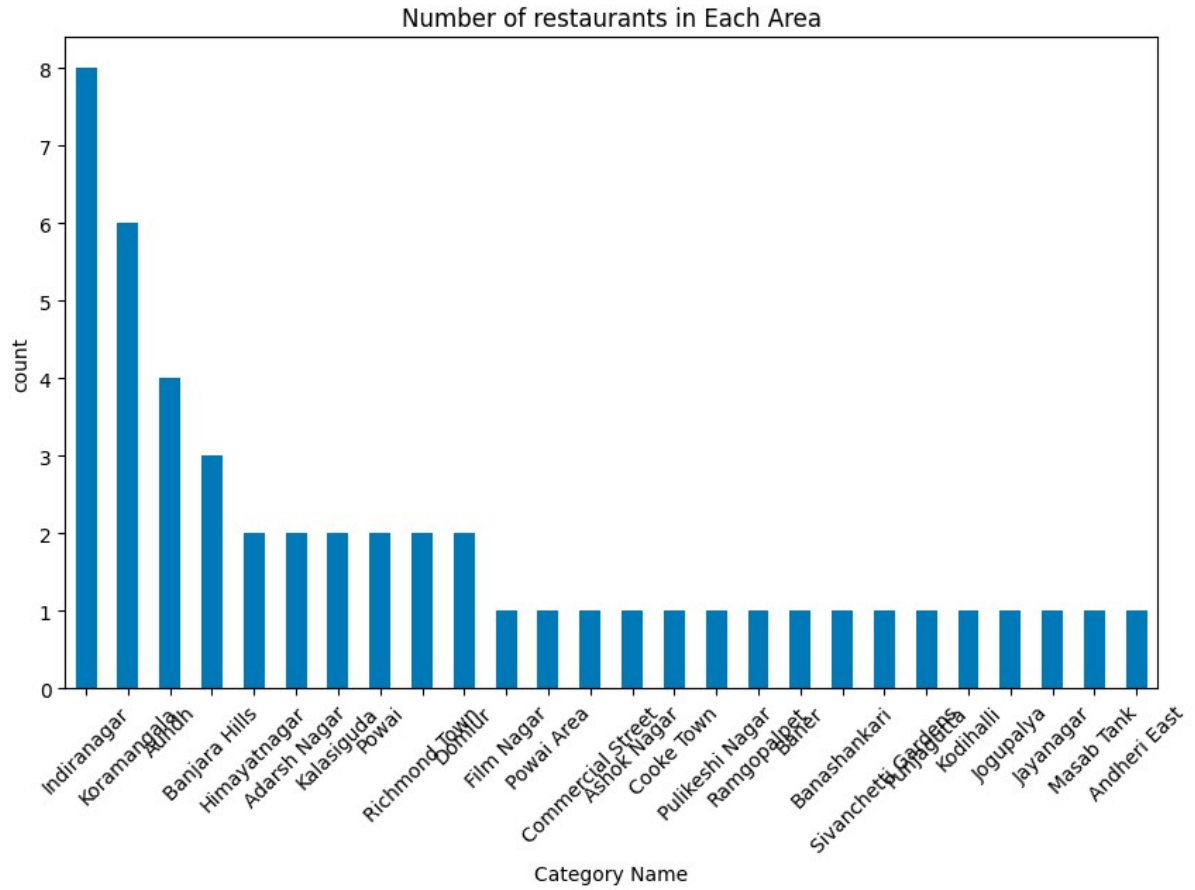
Output:

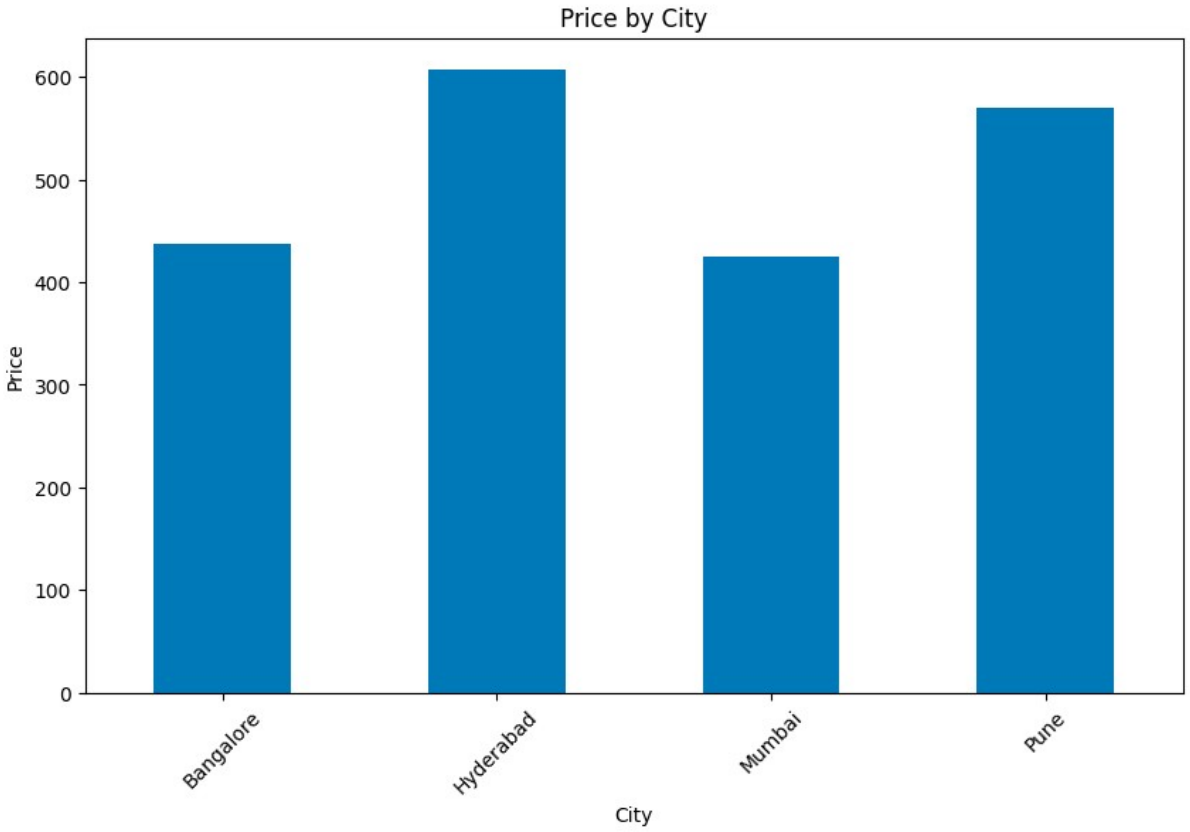


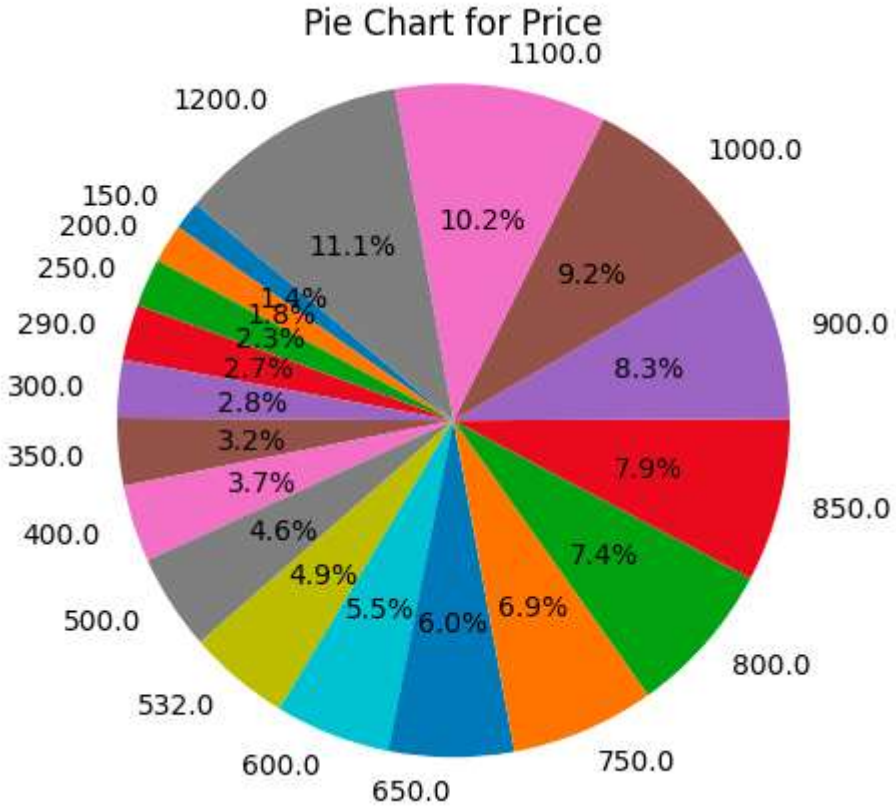
```
dcr = pd.read_csv('discretized_data.csv')

avg_monthly_sales = dcr.groupby('Price')['Total ratings'].mean()
area_counts = dcr['Area'].value_counts()

avg_drug_price_by_category = dcr.groupby('City')['Price'].mean()


plt.figure(figsize=(10,6))
avg_monthly_sales.plot(kind='bar')
plt.title('Toatal ratings by Price')
plt.xlabel('Price')
plt.ylabel('Total ratings')
plt.xticks(rotation=45)
plt.show()


plt.figure(figsize=(10,6))
area_counts.plot(kind='bar')
plt.title('Number of restaurants in Each Area')
plt.xlabel('Category Name')
plt.ylabel('count')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10,6))
avg_drug_price_by_category.plot(kind='bar')
plt.title('Price by City')
plt.xlabel('City')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.show()
```

Output:


Toatal ratings by Price
Total ratings
Price
0
200
400
600
800
1000
1200
150.0
200.0
250.0
290.0
300.0
350.0
400.0
500.0
532.0
600.0
650.0
750.0
800.0
850.0
900.0
1000.0
1100.0
1200.0


Number of restaurants in Each Area
count
Category Name
0
1
2
3
4
5
6
7
8
Indiranagar
Koramangala
Banjara Hills
Himayatnagar
Adarsh Nagar
Kalasiguda
Powai
Richmond Town
Film Nagar
Powai Area
Commercial Street
Ashok Nagar
Cooke Town
Pulikeshi Nagar
Banashankari
Kodihalli
Jogupalya
Jayanagar
Masab Tank
Andheri East


Price by City
Price
City
0
100
200
300
400
500
600
Bangalore
Hyderabad
Mumbai
Pune


Pie Chart for Price
1100.0
1200.0
1000.0
150.0
200.0
250.0
290.0
300.0
350.0
400.0
500.0
532.0
600.0
650.0
750.0
800.0
850.0
900.0
10.2%
11.1%
9.2%
8.3%
7.9%
7.4%
6.9%
6.0%
5.5%
4.9%
4.6%
3.7%
3.2%
2.8%
2.7%
2.3%
1.8%
1.4%

Experiment: 2

Aim : Write Detailed Problem statement and design dimensional modeling (creation of star and snowflake schema) .

Theory:

In this case study project, the objective is to design and implement an efficient data warehousing system for a fictional restaurant chain. The primary aim is to establish a robust data architecture capable of seamlessly managing various aspects of the restaurant's operations, encompassing restaurant-specific details, location information, and temporal data.

Problem Statement : Restaurant system

Description :

The Dimension consists of the following attributes:

Restaurant Table Attributes:

- RESTAURENT_ID - Unique identifier for each restaurant.
- TYPE - The type of restaurant.
- NAME - Name of restaurant.
- RATE - rating of restaurant.
- VOTES - number of votes

LOCATION Table Attributes:

- LOCATION_ID - Unique identifier for each location.
- ADDRESS - Address of restaurant.
- AREA -Area of restaurant.
- CITY - City of restaurant.
- PINCODE - pincode of location

TIME Table Attributes:

- TIME_ID - Unique timestamp identifier for each time.
- YEAR - The year component of the timestamp.
- MONTH - The month of the year.
- WEEK - The week of the year.

DAY - The day of the month.

FACT_TABLE_BANK_SYSTEM Table Attributes:

RESTAURANT_ID - Foreign key referencing the RESTAURANT table.
LOCATION_ID - Foreign key referencing the LOCATION table.
TIME_ID - Foreign key referencing the TIME table.
AVG_COST - Average cost of food.
QUANTITY - Quantity of food.

Star schema :

A **star schema** is a type of data modeling 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 analyze 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. For example, a user might want to see sales revenue by product category, or by region and time period.

The star schema is a popular data modeling technique in data warehousing because it is easy to understand and query. The simple structure of the star schema allows for fast query response times and efficient use of database resources. Additionally, the star schema can be easily extended by adding new dimension tables or measures to the fact table, making it a scalable and flexible solution for data warehousing.

Star schema is the fundamental schema among the data mart schema and it is simplest. This schema is widely used to develop or build a data warehouse and dimensional data marts. It includes one or more fact tables indexing any number of dimensional tables. The star schema is a necessary cause of the snowflake schema. It is also efficient for handling basic queries.

It is said to be a star as its physical model resembles the star shape having a fact table at its center and the dimension tables at its peripheral representing the star's points. Below is an example to demonstrate the Star Schema:

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.

Features:

Central fact table: The star schema revolves around a central fact table that contains the numerical data being analyzed. This table contains foreign keys to link to dimension tables.

Dimension tables: Dimension tables are tables that contain descriptive attributes about the data being analyzed. These attributes provide context to the numerical data in the fact table. Each dimension table is linked to the fact table through a foreign key.

Denormalized structure: A star schema is denormalized, which means that redundancy is allowed in the schema design to improve query performance. This is because it is easier and faster to join a small number of tables than a large number of tables.

Simple queries: Star schema is designed to make queries simple and fast. Queries can be written in a straightforward manner by joining the fact table with the appropriate dimension tables.

Aggregated data: The numerical data in the fact table is usually aggregated at different levels of granularity, such as daily, weekly, or monthly. This allows for analysis at different levels of detail.

Fast performance: Star schema is designed for fast query performance. This is because the schema is denormalized and data is pre-aggregated, making queries faster and more efficient.

Easy to understand: The star schema is easy to understand and interpret, even for non-technical users. This is because the schema is designed to provide context to the numerical data through the use of dimension tables.

Snowflake schema :

The snowflake schema is a variant of the start 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.

A snowflake schema is a type of data modeling technique used in data warehousing to represent data in a structured way that is optimized for querying large amounts of data efficiently. 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 center 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.

For Example, in a sales data warehouse, the product dimension table might be normalized into multiple related tables, such as product category, product subcategory, and product details. Each of these tables would be related to the product dimension table through a foreign key relationship.

Characteristics of Snowflake Schema

- The snowflake schema uses small disk space.
- It is easy to implement the dimension that is added to the schema.
- There are multiple tables, so performance is reduced.
- The dimension table consists of two or more sets of attributes that define information at different grains.
- The sets of attributes of the same dimension table are populated by different source systems.

Features of the Snowflake Schema

- Normalization: The snowflake schema is a normalized design, which means that data is organized into multiple related tables. This reduces data redundancy and improves data consistency.
- Hierarchical Structure: The snowflake schema has a hierarchical structure that is organized around a central fact table. The fact table contains the measures or metrics of interest, and the dimension tables contain the attributes that provide context to the measures.
- Multiple Levels: The snowflake schema can have multiple levels of dimension tables, each related to the central fact table. This allows for more granular analysis of data and enables users to drill down into specific subsets of data.

- Joins: The snowflake schema typically requires more complex SQL queries that involve multiple table joins. This can impact performance, especially when dealing with large data sets.
- Scalability: The snowflake schema is scalable and can handle large volumes of data. However, the complexity of the schema can make it difficult to manage and maintain.

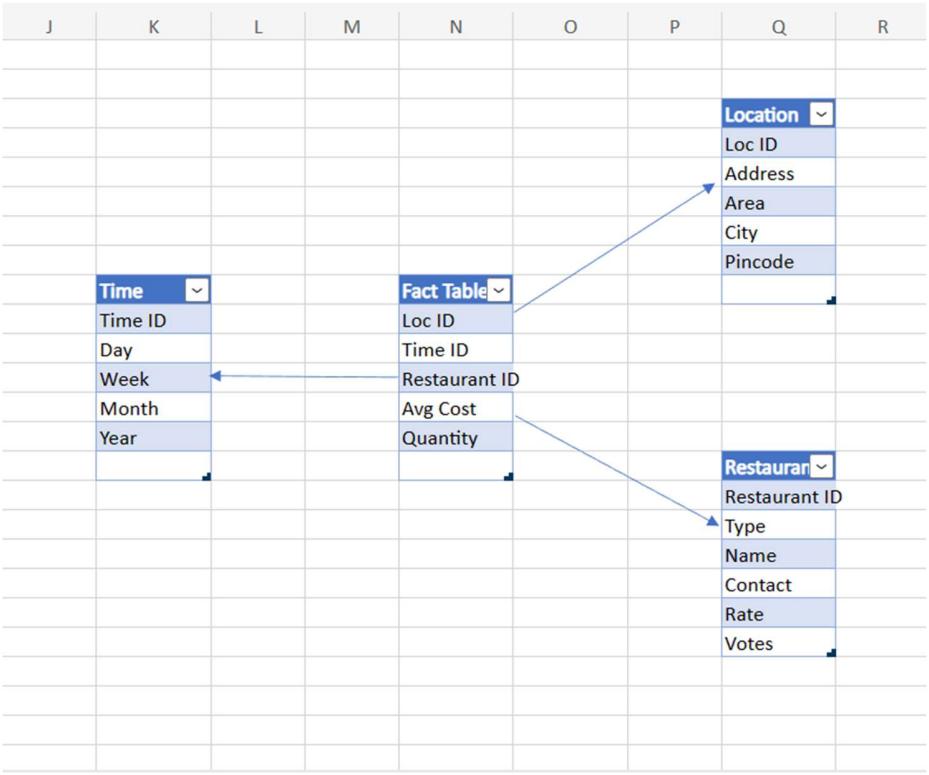
Advantages of Snowflake Schema

- It provides structured data which reduces the problem of data integrity.
- It uses small disk space because data is highly structured.

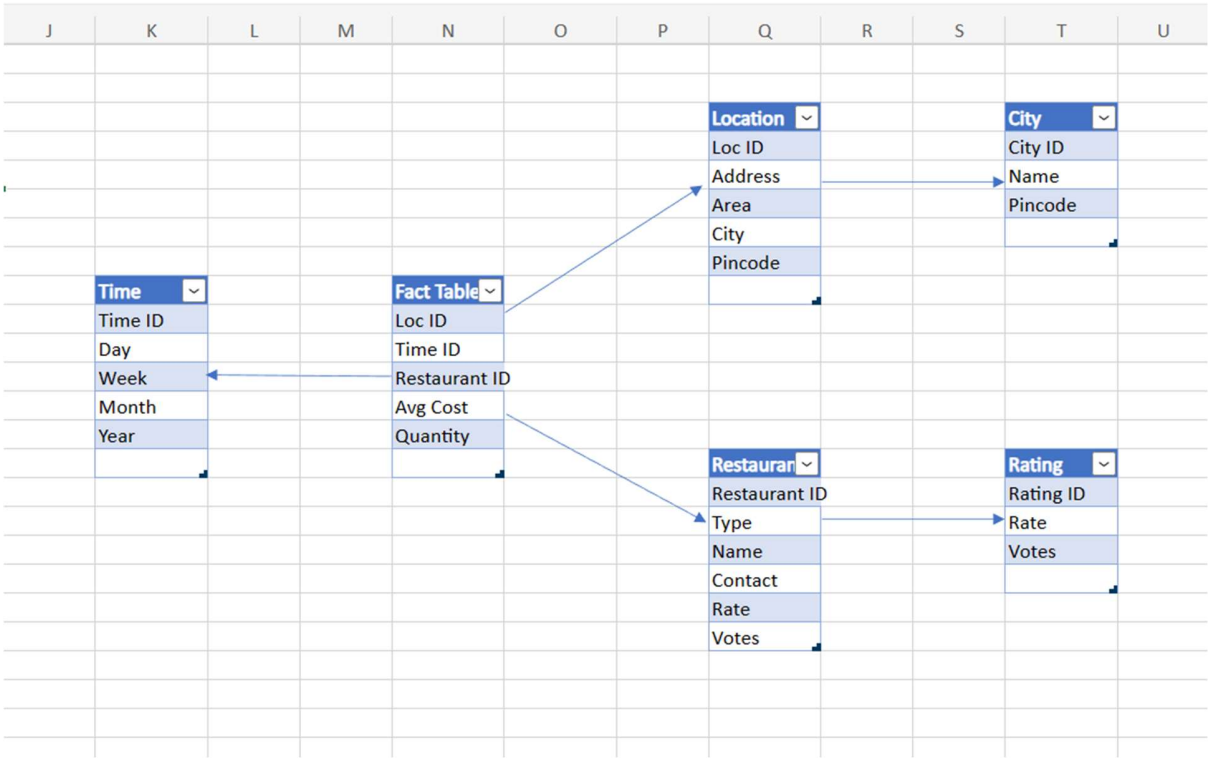
Disadvantages of Snowflake Schema

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

STAR SCHEMA:



SNOWFLAKE SCHEMA:



Experiment 3

Aim: Implementation of Dimension Table and Fact Tables on the case study

Theory:

In data warehousing and data mining, dimension tables and fact tables play essential roles in structuring and organizing data for efficient analysis and reporting. These tables are key components of a star schema or snowflake schema, which are common data modeling techniques used in data warehousing.

Dimension Table:

A dimension table is used to store descriptive attributes or characteristics that provide context to the data in a fact table. These attributes are often used as criteria for slicing and dicing the data during analysis. Dimension tables contain hierarchies of attributes, such as time, geography, products, customers, or any other relevant information. For example, a time dimension table might include attributes like date, month, quarter, and year. Dimension tables are relatively small compared to fact tables and help in categorizing and organizing the data. They serve as the primary reference point for querying and reporting in a data warehouse.

Fact Table:

A fact table, on the other hand, stores the quantitative measures or metrics, often referred to as facts, that are the focus of analysis. These facts are typically numeric values like sales revenue, quantity sold, profit, or any other performance indicators. The fact table also contains foreign keys that link to dimension tables, providing a way to relate facts to descriptive attributes. Fact tables are generally much larger than dimension tables and contain the essential data for analytical operations. By establishing relationships with dimension tables, they enable users to aggregate, filter, and drill down into data to gain insights, make decisions, and discover patterns.

In summary, dimension tables and fact tables are foundational elements in data warehousing and mining. Dimension tables provide the necessary context and descriptive attributes, while fact tables store the quantitative data, forming a robust foundation for conducting complex analytical tasks, including data aggregation, drill-down analysis, and reporting in a data warehouse environment. These two types of tables, when combined in a well-designed schema, facilitate efficient data retrieval and analysis, enabling organizations to extract valuable insights from their data.

Code:

1. **Restaurant Dimension Table:**

```
create table Restaurent(
Restaurent_id INT PRIMARY KEY,
Type VARCHAR2(20),
Name VARCHAR2(20) NOT NULL,
Contact INT NOT NULL,
Rate FLOAT,
Votes INT
);
DESC RESTAURENT;

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(1,'Casual Dining','Jalsa',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(2,'Casual Dining','Spice Elephant
',08041714161
,4.1,787);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(3,'Cafe  Casual Dining
','San Churro Cafe
',9663487993
,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(4,'Quick Bites
','Addhuri Udupi Bhojana
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(5,'Casual Dining
','Grand Village
',08042297555,4.1,775);


INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(6,'Casual Dining
','Timepass Dinner
```

```
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(7,'Casual Dining','Rosewood Hotel',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(8,'Casual Dining Cafe
','Onesta
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(9,'Cafe
','Penthouse Cafe
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(10,'Cafe
','Smacznego
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(11,'Cafe
','Cafe Down The Alley',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(12,'Cafe
','Cafe Shuffle
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(13,'Cafe
','The Coffee Shack
',08042297555,4.1,775);

INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
VALUES(14,'Cafe
','Caf-Eleven
',08042297555,4.1,775);
);
INSERT INTO Restaurent(Restaurent_id,Type,Name,Contact,Rate,Votes)
```

VALUES(15,'Cafe Casual Dining
','San Churro Cafe
',08042297555,4.1,775);
);

RESTAURENT_ID	TYPE	NAME	CONTACT	RATE	VOTES
1	Casual Dining	Jalsa	8042297555	4.1	775
3	Cafe Casual Dining	San Churro Cafe	9663487993	4.1	775
13	Cafe	The Coffee Shack	8042297555	4.1	775
8	Casual Dining Cafe	Onesta	8042297555	4.1	775
10	Cafe	Smacznego	8042297555	4.1	775
4	Quick Bites	Addhuri Udupi	8042297555	4.1	775
2	Casual Dining	Spice Elephant	8041714161	4.1	787
5	Casual Dining	Grand Village	8042297555	4.1	775
6	Casual Dining	Timepass Dinner	8042297555	4.1	775
7	Casual Dining	Rosewood Hotel	8042297555	4.1	775
14	Cafe	Caf-Eleven	8042297555	4.1	775
15	Cafe Casual Dining	San Churro Cafe	8042297555	4.1	775
11	Cafe	Cafe Down The Alley	8042297555	4.1	775
9	Cafe	Penthouse Cafe	8042297555	4.1	775

12	Cafe	Cafe Shuffle	8042297555	4.1	775

2.Time Dimension:

```
CREATE TABLE time (
time_id int PRIMARY KEY,
month varchar2(10),
year varchar2(4),
week int,
tdate DATE NOT NULL
);
desc time;

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (1,'January','2020',2,TO_DATE('2020-01-15', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (2, 'March', '2021', 3, TO_DATE('2021-03-18', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (3, 'April', '2021', 4, TO_DATE('2021-04-22', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (4, 'May', '2021', 1, TO_DATE('2021-05-08', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
VALUES (5, 'June', '2021', 2, TO_DATE('2021-06-15', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
VALUES (6, 'July', '2022', 3, TO_DATE('2022-07-20', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (7, 'August', '2022', 4, TO_DATE('2022-08-25', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
```

```
VALUES (8, 'September', '2022', 1, TO_DATE('2022-09-05', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (9, 'October', '2022', 2, TO_DATE('2022-10-10', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
VALUES (10, 'November', '2023', 3, TO_DATE('2023-11-18', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
VALUES (11, 'December', '2023', 4, TO_DATE('2023-12-23', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (12, 'January', '2023', 1, TO_DATE('2023-01-07', 'YYYY-MM-DD'));


INSERT INTO time (time_id, month, year, week, tdate)
VALUES (13, 'February', '2023', 2, TO_DATE('2023-02-15', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (14, 'March', '2023', 3, TO_DATE('2023-03-20', 'YYYY-MM-DD'));

INSERT INTO time (time_id, month, year, week, tdate)
VALUES (15, 'April', '2023', 4, TO_DATE('2023-04-25', 'YYYY-MM-DD'));
```

TIME_ID	MONTH	YEAR	WEEK	TDATE
1	January	2020	2	15-JAN-20
2	March	2021	3	18-MAR-21
3	April	2021	4	22-APR-21
4	May	2021	1	08-MAY-21

5	June	2021	2	15-JUN-21
6	July	2022	3	20-JUL-22
7	August	2022	4	25-AUG-22
8	September	2022	1	05-SEP-22
9	October	2022	2	10-OCT-22
10	November	2023	3	18-NOV-23
11	December	2023	4	23-DEC-23
12	January	2023	1	07-JAN-23
13	February	2023	2	15-FEB-23
14	March	2023	3	20-MAR-23
15	April	2023	4	25-APR-23

3.Location Table:

```
CREATE TABLE location (
    loc_id INT PRIMARY KEY,
    address VARCHAR2(255),
    area VARCHAR2(50),
    city VARCHAR2(50),
    pincode VARCHAR2(10)
);
desc location

-- Statement 8
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('1', '456 Oak Lane, Apt 10A', 'Oakville', 'Greensboro', '12345');
```

```
-- Statement 9
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('2', '789 Elm Street', 'Elmsville', 'Oaktown', '67890');

-- Statement 10
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('3', '1010 Willow Way', 'Willowville', 'Willowtown', '54321');

-- Statement 11
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('4', '222 Pine Avenue', 'Pinehurst', 'Pinewood', '98765');

-- Statement 12
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('5', '333 Birch Road', 'Birchwood', 'Birchtown', '87654');

-- Statement 13
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('6', '444 Maple Street, Apt 5B', 'Mapleville', 'Maplewood', '23456');

-- Statement 14
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('7', '555 Cedar Lane', 'Cedarwood', 'Cedartown', '34567');

-- Statement 15
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('8', '666 Oak Avenue, Apt 8D', 'Oakdale', 'Oakland', '45678');

-- Statement 1
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('9', '1234 Main Street, Suite 101', 'Downtown', 'Metropolis', '12345');

-- Statement 2
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('10', '5678 Elm Avenue', 'Suburbia', 'Townsville', '67890');

-- Statement 3
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('11', '9876 Oak Street', 'Woodland', 'Greenville', '54321');
```

-- Statement 4
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('12', '555 Maple Lane, Unit 7B', 'Maplewood', 'Arlington', '98765');

-- Statement 5
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('13', '789 Pine Road', 'Pineville', 'Evergreen', '87654');

-- Statement 6
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('14', '222 Birch Street, Apt 3C', 'Birchville', 'Springfield', '23456');

-- Statement 7
INSERT INTO location (loc_id, address, area, city, pincode)
VALUES ('15', '333 Cedar Avenue', 'Cedarville', 'Rivertown', '34567');\

LOC_ID	ADDRESS	AREA	CITY	PINCODE
1	456 Oak Lane, Apt 10A	Oakville	Greensboro	12345
2	789 Elm Street	Elmsville	Oaktown	67890
3	1010 Willow Way	Willowville	Willowtown	54321
4	222 Pine Avenue	Pinehurst	Pinewood	98765
5	333 Birch Road	Birchwood	Birchtown	87654
6	444 Maple Street, Apt 5B	Mapleville	Maplewood	23456
7	555 Cedar Lane	Cedarwood	Cedartown	34567
8	666 Oak Avenue, Apt 8D	Oakdale	Oakland	45678
9	1234 Main Street, Suite 101	Downtown	Metropolis	12345

10	5678 Elm Avenue	Suburbia	Townsville	67890
11	9876 Oak Street	Woodland	Greenville	54321
12	555 Maple Lane, Unit 7B	Maplewood	Arlington	98765
13	789 Pine Road	Pineville	Evergreen	87654
14	222 Birch Street, Apt 3C	Birchville	Springfield	23456
15	333 Cedar Avenue	Cedarville	Rivertown	34567

4.Fact table

```
CREATE TABLE fact_table AS
SELECT
   location.loc_id,
   Restaurent.Restaurent_id,
   time.time_id,
   0.0 AS avg_cost, -- Example initial value for average cost
   0 AS quantity    -- Example initial value for quantity
FROM
   location
CROSS JOIN
   Restaurent
CROSS JOIN
   time;
```

Inserting random values for avg_cost and quantity

```
BEGIN
   FOR rec IN (SELECT rowid, loc_id FROM fact_table) LOOP
      UPDATE fact_table
      SET
         avg_cost = ROUND(DBMS_RANDOM.VALUE(50, 500), 2),
         quantity = ROUND(DBMS_RANDOM.VALUE(1, 100), 0)
      WHERE rowid = rec.rowid;
   END LOOP;
   COMMIT; -- Commit the changes
END;
/
```

SELECT * FROM fact_table where time_id<6

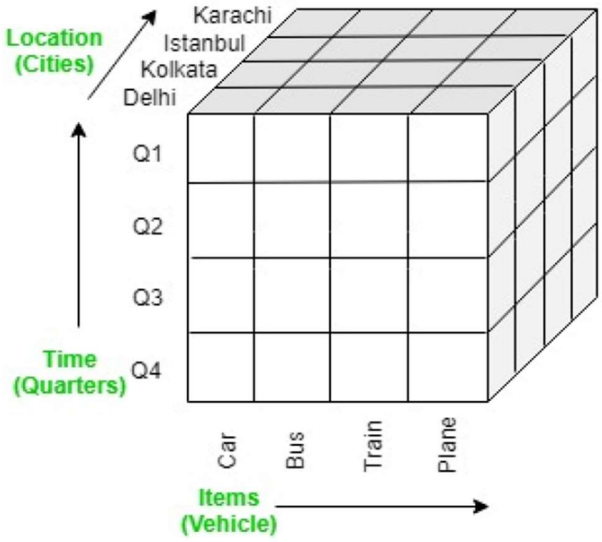
LOC_ID	RESTAURENT_ID	TIME_ID	AVG_COST	QUANTITY
1	1	1	470.85	62
1	1	2	128.85	8
1	1	3	424.06	23
1	1	4	131.22	6
1	1	5	92.66	57
1	2	1	157.66	58
1	2	2	432.53	50
1	2	3	81.23	95
1	2	4	146.83	47
1	2	5	482.44	9
1	3	1	407.79	43
1	3	2	331.27	95
1	3	3	75.82	83
1	3	4	77.98	34
1	3	5	226.38	76
1	4	1	186.52	2

1	4	2	449.21	59
1	4	3	62.34	30
1	4	4	181.14	22

Experiment 4

Aim: Implementation of OLAP Operations on the selected case study.

Theory:
OLAP stands for Online Analytical Processing Server. It is a software technology that allows users to analyse information from multiple database systems at the same time. It is based on multidimensional data model and allows the user to query on multi-dimensional data (eg. Delhi -> 2018 -> Sales data). OLAP databases are divided into one or more cubes and these cubes are known as Hyper-cubes.



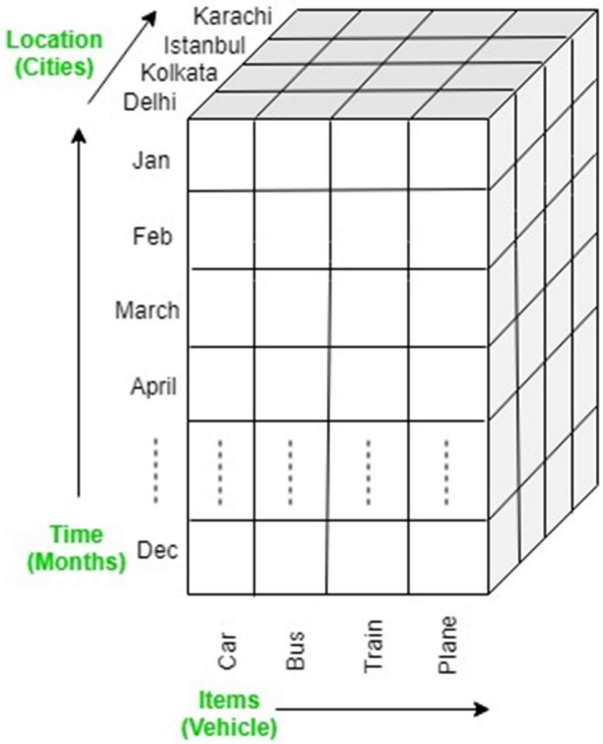
A.

OLAP operations:

There are five basic analytical operations that 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
2. In the cube given in overview section, the drill down operation is

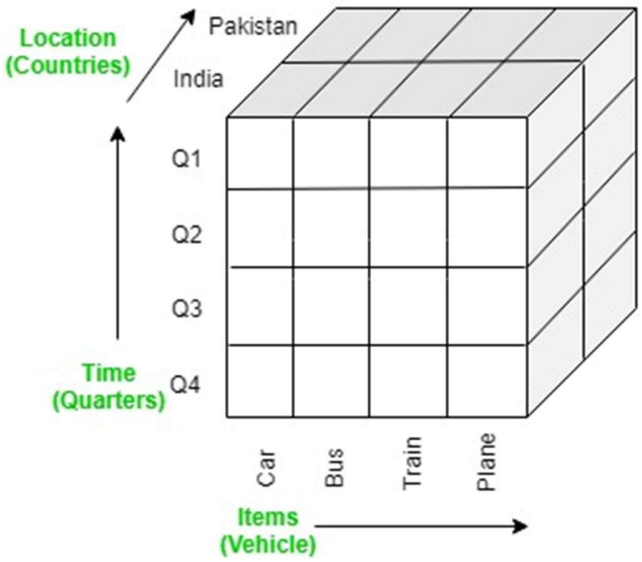
performed by moving down in the concept hierarchy of Time dimension (Quarter -> Month



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

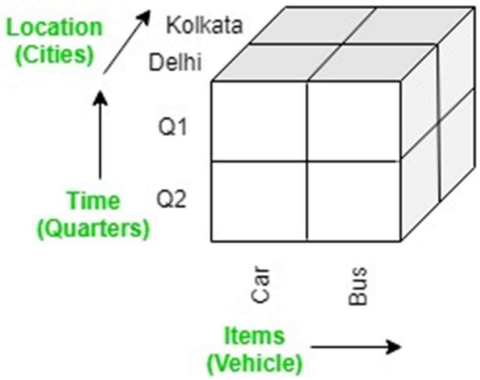
- Climbing up in the concept hierarchy
- Reducing the dimensions

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).

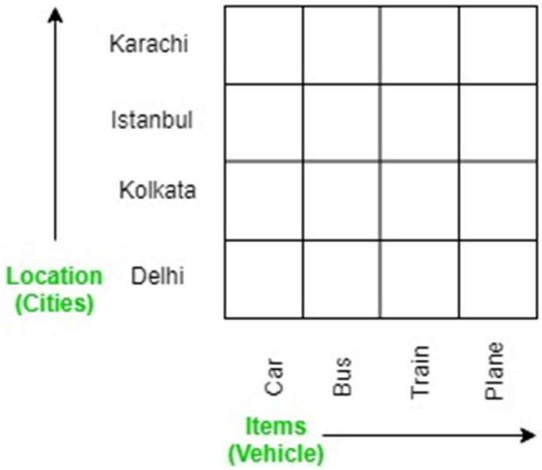


Dice: It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected
by selecting following dimensions with criteria:

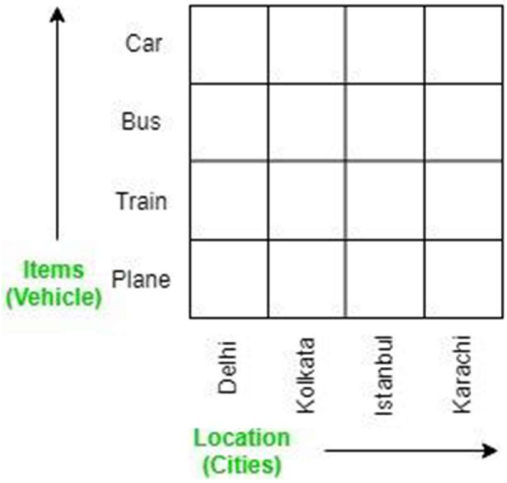
- Location = “Delhi” or “Kolkata”
- Time = “Q1” or “Q2”
- Item = “Car” or “Bus”



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”.



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 pivot operation gives a new view of it.



1. Slice

```
CREATE MATERIALIZED VIEW mv_slice AS
SELECT *
FROM fact_table
WHERE loc_id = 1;
SELECT * FROM mv_slice;
```

LOC_ID	RESTAURENT_ID	TIME_ID	AVG_COST	QUANTITY
1	1	1	470.85	62
1	1	2	128.85	8
1	1	3	424.06	23
1	1	4	131.22	6
1	1	5	92.66	57
1	1	6	392.12	12
1	1	7	101.51	93
1	1	8	410.23	11
1	1	9	374.18	58
1	1	10	164.91	6
1	1	11	413.74	70
1	1	12	77.19	59
1	1	13	481.46	26
1	1	14	446.42	21

1	1	15	283.85	58
1	2	1	157.66	58
1	2	2	432.53	50
1	2	3	81.23	95
1	2	4	146.83	47
1	2	5	482.44	9
1	2	6	156.99	47
1	2	7	119.28	92
1	2	8	185.83	68
1	2	9	86.64	74
1	2	10	213.7	35
1	2	11	494.59	64
1	2	12	100.18	34
1	2	13	213.41	90
1	2	14	441.98	7
1	2	15	169.39	36
1	3	1	407.79	43
1	3	2	331.27	95
1	3	3	75.82	83

1	3	4	77.98	34
1	3	5	226.38	76

2.Dice

```
CREATE MATERIALIZED VIEW mv_dice AS
SELECT *
FROM fact_table
WHERE loc_id = 1 AND Restaurent_id = 2;
SELECT * FROM mv_dice;
```

LOC_ID	RESTAURENT_ID	TIME_ID	AVG_COST	QUANTITY
1	2	1	157.66	58
1	2	2	432.53	50
1	2	3	81.23	95
1	2	4	146.83	47
1	2	5	482.44	9
1	2	6	156.99	47
1	2	7	119.28	92
1	2	8	185.83	68
1	2	9	86.64	74
1	2	10	213.7	35
1	2	11	494.59	64

1	2	12	100.18	34
1	2	13	213.41	90
1	2	14	441.98	7
1	2	15	169.39	36

3.Pivot

```
CREATE MATERIALIZED VIEW mv_pivot AS
SELECT
   loc_id,
   MAX(CASE WHEN measure = 'Avg Cost' THEN value END) AS "Avg Cost",
   MAX(CASE WHEN measure = 'Quantity' THEN value END) AS "Quantity"
FROM (
   SELECT loc_id, measure, value
   FROM fact_table
   UNPIVOT INCLUDE NULLS (
      value FOR measure IN (avg_cost AS 'Avg Cost', quantity AS 'Quantity')
   )
)
GROUP BY loc_id;
SELECT * FROM mv_pivot;
```

LOC_ID	Avg Cost	Quantity
6	494.54	99
14	499.2	99
1	499.87	100
7	494.17	100
15	497.13	100
2	499.45	100

8	498.89	100
11	498.95	100
12	499.44	100
4	498.95	99
5	497.96	99
10	499.46	100
3	499.09	100
9	499.77	99
13	497.07	99

4.Rollup

```
CREATE OR REPLACE VIEW vw_rollup AS
SELECT
   CASE
      WHEN GROUPING(loc_id) = 1 THEN 'Grand Total'
      ELSE TO_CHAR(loc_id)
   END AS loc_id,
   SUM(avg_cost) AS "Total Avg Cost",
   SUM(quantity) AS "Total Quantity"
FROM fact_table
GROUP BY ROLLUP(loc_id);

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

LOC_ID	Total Avg Cost	Total Quantity

1	64048.47	11131
2	61644.08	11397
3	65245.29	11732
4	62866.2	11210
5	61041.43	10819
6	59757.59	12001
7	63568.43	11134
8	60805.87	11388
9	62592.83	11138
10	63589.64	11853
11	60455.84	11370
12	63942.84	12002
13	61972.84	11612
14	63829.14	11384
15	61534.53	11855
Grand Total	936895.02	172026

5.Drilldown

CREATE OR REPLACE VIEW vw_drilldown AS

```
SELECT
   loc_id,
   Restaurent_id,
   SUM(avg_cost) AS "Total Avg Cost",
   SUM(quantity) AS "Total Quantity"
FROM fact_table
GROUP BY loc_id, Restaurent_id;

-- Create a materialized view based on the view
CREATE MATERIALIZED VIEW mv_drilldown AS
SELECT * FROM vw_drilldown;
```

LOC_ID	RESTAURENT_ID	Total Avg Cost	Total Quantity
1	7	4288.23	769
1	14	3438.9	946
2	10	4180.82	705
2	14	4594.69	631
3	1	4559.47	749
4	5	4539.03	590
4	8	3884	700
4	14	4236.51	639
5	14	4541.36	817
6	3	4528.04	892
6	5	4171.63	932
6	7	3679.14	803

6	15	3481.54	678
7	9	4380.58	666
7	10	4543.15	809
7	12	3959.81	839
8	10	4344.21	872
9	11	3892.59	879
10	4	4290.26	829
10	9	3807.35	699
10	10	4233.76	820
10	12	4344.4	885

EXPERIMENT: 5

AIM: Implementation of Bayesian Algorithm

THEORY:

Bayesian algorithms are a class of statistical algorithms used in data analysis, machine learning, and data mining. They are based on the principles of Bayesian probability theory, which is a probabilistic approach for reasoning under uncertainty.

Bayesian Algorithm:

The Bayesian algorithm uses Bayes' theorem to update the probability for a hypothesis as new evidence or data becomes available. In the context of data mining or machine learning, Bayes' Theorem Formula:

P(A|B) = (P(B|A) * P(A)) / P(B)

Example: For the email classification example:

P(S): The prior probability of an email being spam.

P(¬S): The prior probability of an email not being spam.

P(E|S): The likelihood of observing the evidence (word occurrences) given that the email is spam.

P(E|¬S): The likelihood of observing the evidence given that the email is not spam.

Then, you can use Bayes' theorem as follows:

P(S|E) = (P(E|S) * P(S)) / P(E) P(¬S|E) = (P(E|¬S) * P(¬S)) / P(E)

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
# Load the data from the CSV file
data = pd.read_csv('/content/zomato (1).csv')
# Drop unnecessary columns
data.drop(['Unnamed: 0', 'Unnamed: 0.1'], axis=1, inplace=True)
# Encode categorical features using LabelEncoder
le = LabelEncoder()
data['restaurant type'] = le.fit_transform(data['restaurant type'])
data['online_order'] = le.fit_transform(data['online_order'])
data['table booking'] = le.fit_transform(data['table booking'])
# Handle empty strings or NaN values in 'avg cost (two people)'
data['avg cost (two people)'] = pd.to_numeric(data['avg cost (two
people)'], errors='coerce')
# Remove rows with NaN values in 'avg cost (two people)'
data.dropna(subset=['avg cost (two people)'], inplace=True)
# Convert 'avg cost (two people)' to categorical levels
data['avg cost level'] = pd.cut(data['avg cost (two people)'],
bins=[0, 300, 500, float('inf')],
labels=['low', 'medium', 'high'])
# Calculate probabilities for each rating level
def calculate_probabilities(data):
total_entries = len(data)
rating_probabilities = defaultdict(int)
for entry in data:
rating_probabilities[entry['RateLevel']] += 1
for rating_level in rating_probabilities:
rating_probabilities[rating_level] /= total_entries
```

```
return rating_probabilities
def calculate_conditional_probabilities(data, feature, feature_value,
rating_level):
# Initialize counts to 0
count_feature_and_rating = 0
count_rating = 0
try:
for entry in data:
if entry['RateLevel'] == rating_level:
# Count entries with the specified rating level
count_rating += 1
if entry[feature] == feature_value:
# Count entries where both feature and rating match
count_feature_and_rating += 1
if count_rating == 0:
# Handle cases where there are no entries with the
specified rating level
return 0.0
else:
# Calculate the conditional probability
return count_feature_and_rating / count_rating
except KeyError:
# Handle KeyError by returning 0.0 (feature doesn't exist in
the dataset)
return 0.0
# Naive Bayes prediction
def predict_rating(user_input, data, rating_probabilities):
predictions = {}
for rating_level in rating_probabilities:
probability = rating_probabilities[rating_level]
for feature, feature_value in user_input.items():
```

```
probability *= calculate_conditional_probabilities(data,
feature, feature_value, rating_level)
predictions[rating_level] = probability
predicted_rating = max(predictions, key=predictions.get)
return predicted_rating
if __name__ == "__main__":
file_path = '/content/zomato (1).csv' # Replace with your CSV file
path
data = load_data(file_path)
rating_probabilities = calculate_probabilities(data)
user_input = {
'restaurant type': input("Enter restaurant type: "),
'avg cost level': input("Enter avg cost level
(high/medium/low): "),
'table booking': input("Table booking (Yes/No): "),
'online order': input("Online order (Yes/No): ")
}
predicted_rating = predict_rating(user_input, data,
rating_probabilities)
print("Predicted Rating Level:", predicted_rating)
```

OUTPUT:

```
Enter restaurant type: Cafe
Enter avg cost level (high/medium/low): high
Table booking (Yes/No): Yes
Online order (Yes/No): No
Predicted Rating Level: Medium
```

Experiment No. 6

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

Theory:

Performing data pre-processing, classification, clustering, and association tasks using data mining tools like WEKA or R involves a series of steps. Let's delve into each of these aspects in detail.

Data Pre-processing:

Data pre-processing is a crucial step in data mining, as it involves transforming raw data into a format suitable for analysis. It includes data cleaning, integration, transformation, and reduction.

1. Data Cleaning: Handling missing values, smoothing noisy data, and resolving inconsistencies in the data.
2. Data Integration: Combining data from multiple sources into a coherent data store.
3. Data Transformation: Normalizing or standardizing data, and transforming it into appropriate forms for analysis.
4. Data Reduction: Reducing the volume but producing the same or similar analytical results. This can be achieved through techniques like feature selection or extraction.

Classification:

Classification is a predictive modeling technique that assigns a class label to a given input data item. It involves training a model on a labeled dataset, then using that model to predict the class label of new or unseen data.

Common classification algorithms include:
- Decision Trees
- Naive Bayes
- Support Vector Machines
- Random Forests
- k-Nearest Neighbors

Clustering:

Clustering is an unsupervised learning technique that involves grouping similar data points into clusters. It's used to discover inherent patterns or groupings within data.

Popular clustering algorithms include:
- K-Means
- Hierarchical Clustering
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- Mean-Shift Clustering

Association:

Association rule mining is a technique to uncover how items are associated with each other within a transactional database. It helps in discovering interesting relationships between variables in large datasets.

Famous association rule algorithms are:
- Apriori algorithm
- FP-growth (Frequent Pattern Growth) algorithm

Implementation in WEKA or R:

Let's briefly outline how these tasks can be performed using the WEKA tool:

1. Data Pre-processing: WEKA provides various filters for data pre-processing tasks such as the "Filter" panel, which allows you to apply filters for attribute selection, transformation, and more.
2. Classification: In WEKA, you can access the "Classify" panel to select and apply various classification algorithms. You can load your data, select an appropriate classifier, and evaluate its performance using cross-validation or other methods.
3. Clustering: You can perform clustering in WEKA by using the "Cluster" panel, which provides options to choose algorithms and settings for clustering analysis. You can visualize and evaluate clusters using the tools available.
4. Association: WEKA supports association rule mining. The "Associate" panel can be used to apply association algorithms to your dataset and extract interesting associations between items.

In R, you can use various packages such as 'caret' for classification, 'cluster' for clustering, and 'arules' for association rule mining.

Remember, thorough understanding of the underlying theory behind these techniques is crucial for effective and accurate application in any data mining tool.

ARFF (Attribute-Relation File Format) is a file format that is used to store data in WEKA. It is a simple text file format that is used to describe datasets. ARFF files have two distinct sections. The first section is the Header section, which is used to

provide information about the dataset such as the name of the relation, a list of the attributes (variables), and their types. The second section is the Data section, where the actual data instances are stored.

To apply classification, association, and clustering algorithms in WEKA, follow these steps:

Classification in WEKA:

Open WEKA.
Load your dataset in ARFF format.
Go to the "Explorer" tab.
Select the "Classify" tab.
Choose a classifier (e.g., J48, Naive Bayes) from the dropdown menu.
Click on the "Start" button to run the classification process.
Evaluate the results using various metrics.

Association in WEKA:

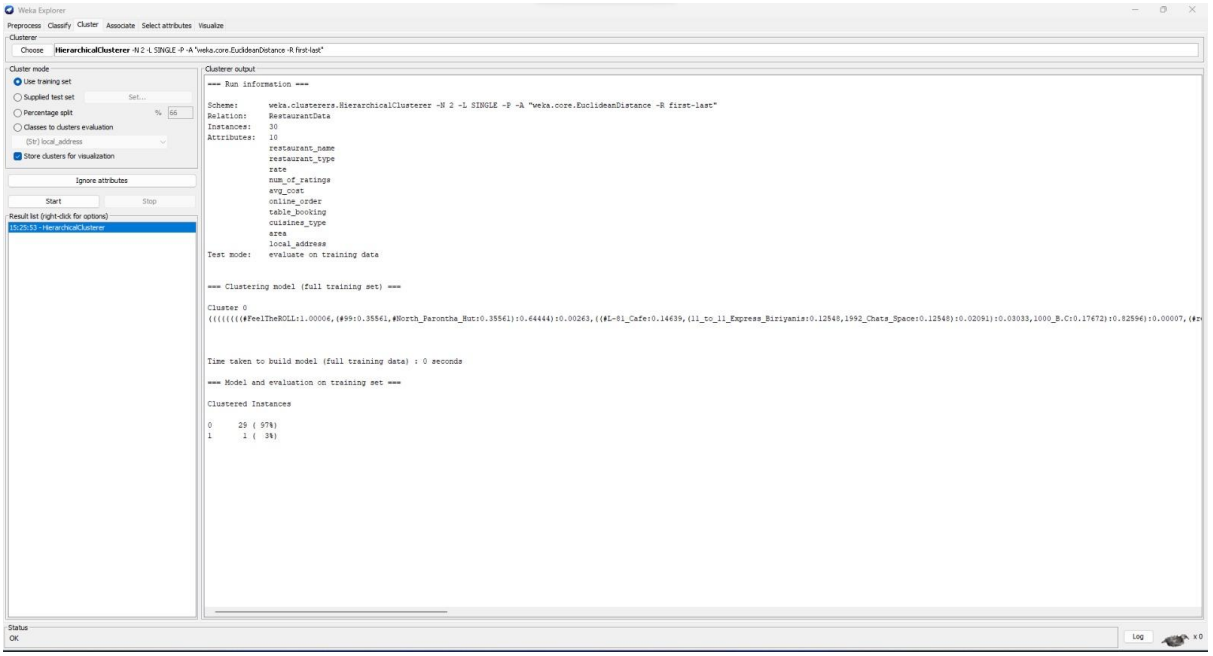
Load your dataset in ARFF format.
Open WEKA and go to the "Explorer" tab.
Select the "Associate" tab.
Choose an association rule algorithm (e.g., Apriori).
Set the parameters for the algorithm.
Click on the "Start" button to run the association analysis.
Examine the generated association rules.

Clustering in WEKA:

Load your dataset in ARFF format.
Open WEKA and go to the "Explorer" tab.
Select the "Cluster" tab.
Choose a clustering algorithm (e.g., k-Means, hierarchical clustering).
Configure the parameters for the chosen algorithm.
Click on the "Start" button to run the clustering process.
Visualize and interpret the clustering results.
Remember that before applying these algorithms, it is important to preprocess the data, which involves handling missing values, normalizing or standardizing data, and dealing with outliers, among other tasks.

Code:

Hierarchical Clustering:



Data:
@relation RestaurantData

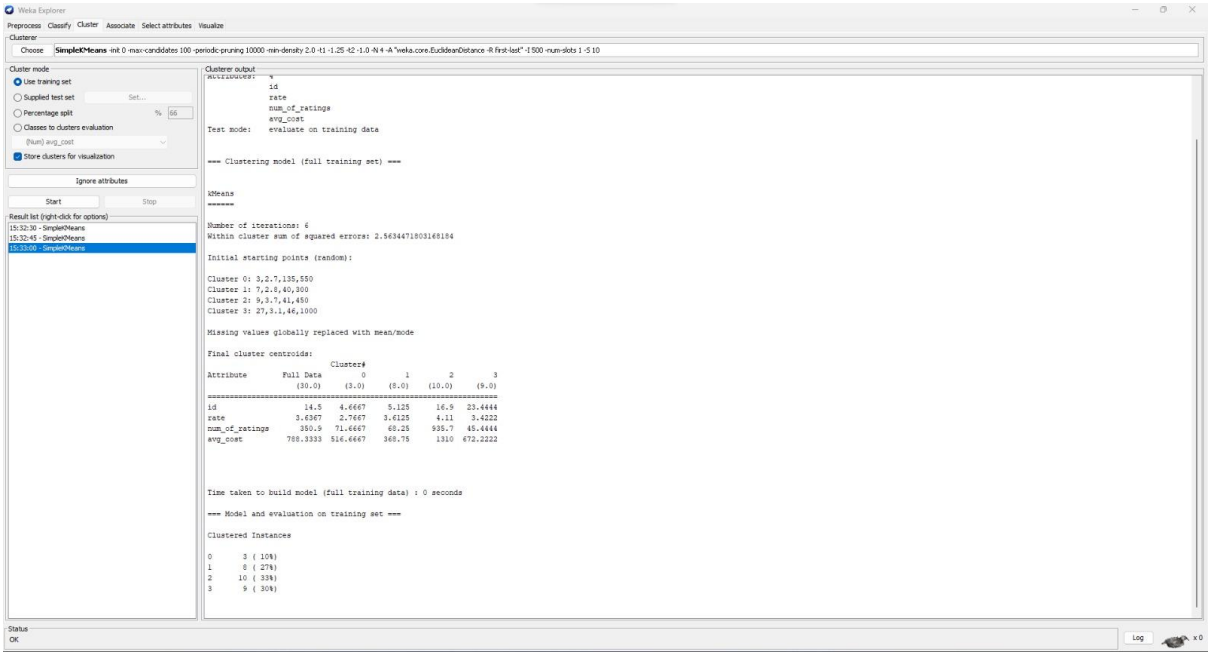
@attribute restaurant_name string
@attribute restaurant_type {Quick_Bites, Cafe, Casual_Dining, Bar, Fine_Dining, Pub, Delivery, Takeaway}
@attribute rate real
@attribute num_of_ratings integer
@attribute avg_cost real
@attribute online_order {Yes, No}
@attribute table_booking {Yes, No}
@attribute cuisines_type string
@attribute area string
@attribute local_address string

@data
#FeelTheROLL, Quick_Bites, 3.4, 7, 200, No, No, Fast_Food, Bellandur, Bellandur
#L-81_Cafe, Quick_Bites, 3.9, 48, 400, Yes, No, Fast_Food, Byresandra_Tavarekere_Madiwala, HSR
#refuel, Cafe, 3.7, 37, 400, Yes, No, Cafe_Beverages, Bannerghatta_Road, Bannerghatta_Road
#Biryani_Central, Casual_Dining, 2.7, 135, 550, Yes, No, Biryani_Mughlai_Chinese, Marathahalli, Marathahalli
#The_Bbq, Casual_Dining, 2.8, 40, 700, Yes, No, BBQ_Continental_North_Indian_Chinese_Beverages, Bellandur, Bellandur
#99, Takeaway, 3.4, 37, 200, No, No, Mughlai_Biryani_Chinese_North_Indian, Whitefield, Whitefield

#Italy, Casual_Dining, 4.1, 305, 700, Yes, No, Italian, Banashankari, Kumaraswamy_Layout
#North_Parontha_Hut, Takeaway, 2.8, 40, 300, No, No, North_Indian, Indiranagar, Old_Airport_Road
1000_B.C, Quick_Bites, 3.2, 49, 300, Yes, No, Arabian_Sandwich_Rolls_Burger, Byresandra_Tavarekere_Madiwala, Koramangala_5th_Block
100°C, Casual_Dining, 3.7, 41, 450, No, No, Biryani_North_Indian, Byresandra_Tavarekere_Madiwala, BTM
11_to_11_Express_Biriyanis, Quick_Bites, 3.5, 22, 300, Yes, No, Biryani_Kebab, Electronic_City, Electronic_City
1131_Bar_Kitchen, Bar, 4.4, 2861, 1500, No, Yes, Continental_Asian_Italian_North_Indian, Old_Airport_Road, Indiranagar
12th_Main_Grand_Mercure, Fine_Dining, 4.1, 353, 2000, No, Yes, European_Asian, Byresandra_Tavarekere_Madiwala, Koramangala_3rd_Block
1441_Pizzeria, Casual_Dining, 4.1, 119, 800, Yes, No, Pizza_Salad, Basavanagudi, JP_Nagar
1522_The_Pub, Pub, 4.2, 1731, 1400, Yes, No, Chinese_Continental_North_Indian, Malleshwaram, Malleshwaram
154_Breakfast_Club, Cafe, 4, 1509, 900, Yes, Yes, Cafe_Continental, Byresandra_Tavarekere_Madiwala, Koramangala_3rd_Block
1722_Urban_Bistro, Casual_Dining, 4.1, 218, 600, Yes, Yes, North_Indian_Chinese_Fast_Food_Biryani, Byresandra_Tavarekere_Madiwala, Koramangala_5th_Block
18+_Ice_Cafe, Cafe, 3.5, 24, 550, No, No, Cafe_Fast_Food_Chinese_Beverages, Kalyan_Nagar, Kammanahalli
1947, Casual_Dining, 3.9, 620, 1000, No, Yes, North_Indian_Chinese_Continental, Rajajinagar, Rajajinagar
1980s_Games_Cafe, Cafe, 3.4, 58, 400, Yes, No, Cafe_Fast_Food, Banashankari, Basavanagudi
1992_Chats_Space, Quick_Bites, 3.7, 33, 200, Yes, No, Street_Food, Malleshwaram, Rajajinagar
1Q1, Casual_Dining, 4.3, 595, 2500, No, Yes, Asian_Japanese_Thai_Malaysian_Vietnamese_Korean_Chinese, Brigade_Road, Infantry_Road
1TO3_Kitchen, Delivery, 3.1, 11, 800, No, No, North_Indian_Chinese_Continental_Italian_South_Indian, Marathahalli, Marathahalli
2_Statez, Casual_Dining, 3.7, 209, 600, Yes, No, Kerala_Seafood_South_Indian_North_Indian_Chinese, Kalyan_Nagar, Kalyan_Nagar
20_Char_Sterlings_MAC_Hotel, Casual_Dining, 4, 419, 1400, Yes, Yes, Chinese_Continental_North_Indian_South_Indian_Arabian, Indiranagar, Old_Airport_Road
20_Feet_High, Casual_Dining, 4, 932, 1000, Yes, Yes, American_Continental_Steak_Salad, Brigade_Road, Church_Street

21_C_La_Marvella_Sarovar_Premiere_Hotel, Casual_Dining, 3.6, 15, 1000, No, No, Continental, Basavanagudi, Jayanagar
24_Carats_The_Capitol, Casual_Dining, 3.1, 46, 1000, No, No, Asian_Continental, Brigade_Road, Cunningham_Road
24_Hours_Cake_Delivery, Delivery, 3.3, 4, 700, No, No, Bakery, Indiranagar, Old_Airport_Road
24_Hours_Coffee_Drop_La_Classic, Cafe, 3.4, 9, 800, No, No, Cafe, Electronic_City, Electronic_City

K-means Clustering:



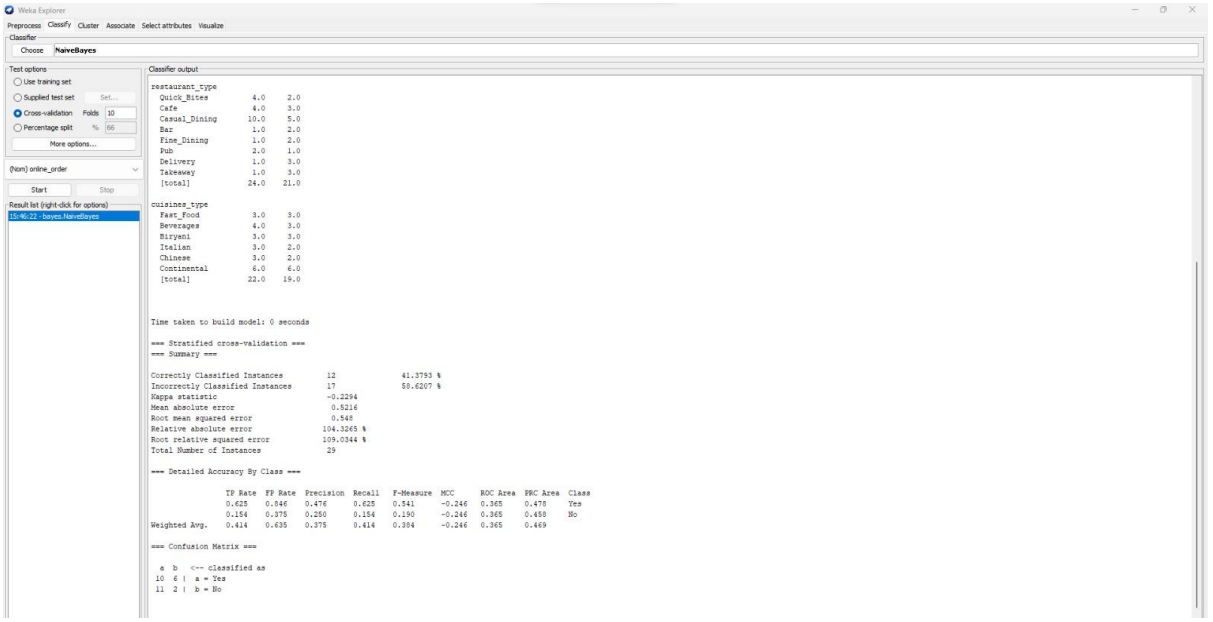
Data:
@relation RestaurantData

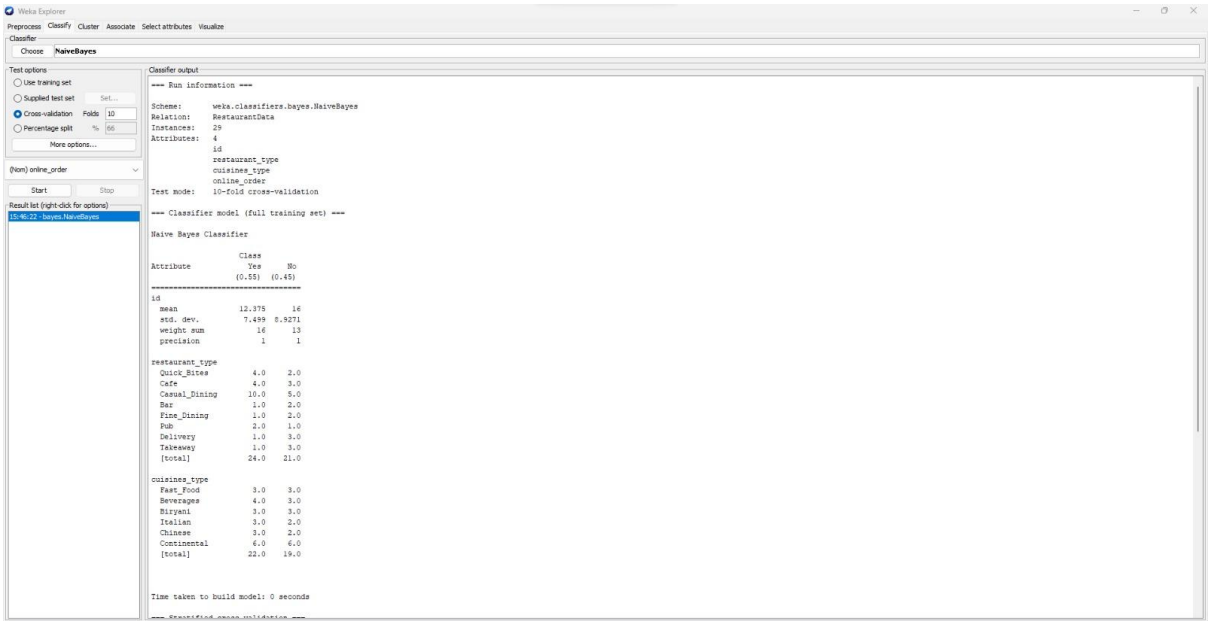
@attribute id integer
@attribute rate real
@attribute num_of_ratings integer
@attribute avg_cost real

@data
0, 3.4, 7, 200
1, 3.9, 48, 400
2, 3.7, 37, 400
3, 2.7, 135, 550
4, 2.8, 40, 700
5, 3.4, 37, 200
6, 4.1, 305, 700

7, 2.8, 40, 300
8, 3.2, 49, 300
9, 3.7, 41, 450
10, 3.5, 22, 300
11, 4.4, 2861, 1500
12, 4.1, 353, 2000
13, 4.1, 119, 800
14, 4.2, 1731, 1400
15, 4, 1509, 900
16, 4.1, 218, 600
17, 3.5, 24, 550
18, 3.9, 620, 1000
19, 3.4, 58, 400
20, 3.7, 33, 200
21, 4.3, 595, 2500
22, 3.1, 11, 800
23, 3.7, 209, 600
24, 4, 419, 1400
25, 4, 932, 1000
26, 3.6, 15, 1000
27, 3.1, 46, 1000
28, 3.3, 4, 700
29, 3.4, 9, 800

Naive Bayes:





Data:

@relation RestaurantData

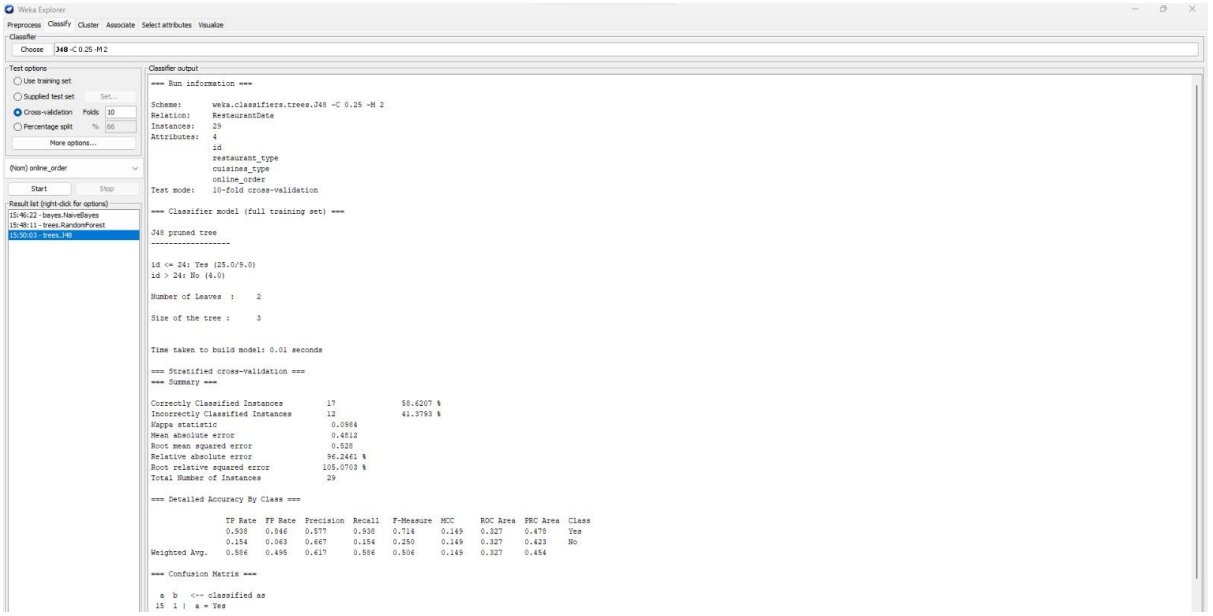
@attribute id integer
@attribute restaurant_type {Quick_Bites, Cafe, Casual_Dining, Bar, Fine_Dining, Pub, Delivery, Takeaway}
@attribute cuisines_type {Fast_Food, Beverages, Biryani, Italian, Chinese, Continental}

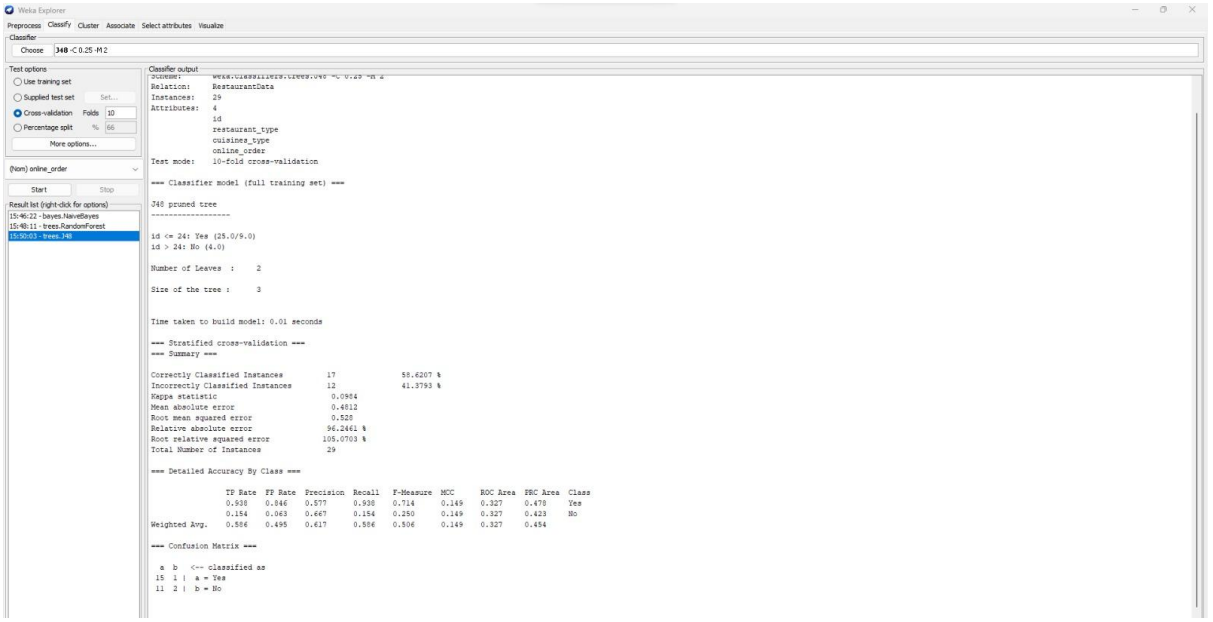
@attribute online_order {Yes, No}

@data

0, Quick_Bites, Fast_Food, No
1, Quick_Bites, Fast_Food, Yes
2, Cafe, Beverages, Yes
3, Casual_Dining, Biryani, Yes
4, Casual_Dining, Continental, Yes
5, Takeaway, Biryani, No
6, Casual_Dining, Italian, Yes
7, Takeaway, Continental, No
8, Quick_Bites, Fast_Food, Yes
9, Casual_Dining, Biryani, No
10, Quick_Bites, Biryani, Yes
11, Bar, Continental, No
12, Fine_Dining, Continental, No
13, Casual_Dining, Italian, Yes
14, Pub, Chinese, Yes
15, Cafe, Beverages, Yes
16, Casual_Dining, Continental, Yes
17, Cafe, Beverages, No
18, Casual_Dining, Continental, Yes
19, Cafe, Beverages, Yes
20, Casual_Dining, Chinese, No
21, Delivery, Italian, No
22, Casual_Dining, Chinese, Yes
23, Casual_Dining, Continental, Yes
24, Casual_Dining, Continental, Yes
25, Casual_Dining, Continental, No
26, Casual_Dining, Continental, No
27, Delivery, Fast_Food, No
28, Cafe, Beverages, No

ID3(IJ48):


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) online_order
Start Stop
Result list (right-click for options)
15:46:22 - bayes.NaiveBayes
15:48:11 - trees.RandomForest
15:50:03 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: RestaurantData
Instances: 29
Attributes: 4
id
restaurant_type
cuisines_type
online_order
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
id <= 24: Yes (25.0/9.0)
id > 24: No (4.0)
Number of Leaves : 2
Size of the tree : 3
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 17 58.6207 %
Incorrectly Classified Instances 12 41.3793 %
Kappa statistic 0.0984
Mean absolute error 0.4812
Root mean squared error 0.528
Relative absolute error 96.2461 %
Root relative squared error 105.0703 %
Total Number of Instances 29
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.938 0.846 0.577 0.938 0.714 0.149 0.327 0.478 Yes
0.154 0.063 0.667 0.154 0.250 0.149 0.327 0.423 No
Weighted Avg. 0.586 0.495 0.617 0.586 0.506 0.149 0.327 0.454
=== Confusion Matrix ===
a b <-- classified as
15 1 | a = Yes


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) online_order
Start Stop
Result list (right-click for options)
15:46:22 - bayes.NaiveBayes
15:48:11 - trees.RandomForest
15:50:03 - trees.J48
Classifier output
Relation: RestaurantData
Instances: 29
Attributes: 4
id
restaurant_type
cuisines_type
online_order
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
id <= 24: Yes (25.0/9.0)
id > 24: No (4.0)
Number of Leaves : 2
Size of the tree : 3
Time taken to build model: 0.01 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 17 58.6207 %
Incorrectly Classified Instances 12 41.3793 %
Kappa statistic 0.0984
Mean absolute error 0.4812
Root mean squared error 0.528
Relative absolute error 96.2461 %
Root relative squared error 105.0703 %
Total Number of Instances 29
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.938 0.846 0.577 0.938 0.714 0.149 0.327 0.478 Yes
0.154 0.063 0.667 0.154 0.250 0.149 0.327 0.423 No
Weighted Avg. 0.586 0.495 0.617 0.586 0.506 0.149 0.327 0.454
=== Confusion Matrix ===
a b <-- classified as
15 1 | a = Yes
11 2 | b = No

Apriore:

Data:
@relation RestaurantData_Apriori

@attribute cuisine_type {Fast_Food, Italian, Chinese, Continental}
@attribute online_order {Yes, No}
@attribute restaurant_type {Quick_Bites, Cafe, Casual_Dining, Bar, Fine_Dining, Pub, Delivery, Takeaway}

@data
"Fast_Food", "No", "Quick_Bites"
"Fast_Food", "Yes", "Quick_Bites"
"Italian", "Yes", "Cafe"
"Fast_Food", "Yes", "Casual_Dining"
"Chinese", "Yes", "Casual_Dining"
"Fast_Food", "No", "Takeaway"
"Italian", "Yes", "Casual_Dining"
"Continental", "No", "Takeaway"
"Italian", "Yes", "Quick_Bites"
"Fast_Food", "No", "Casual_Dining"
"Fast_Food", "Yes", "Quick_Bites"
"Continental", "No", "Fine_Dining"
"Continental", "No", "Fine_Dining"
"Italian", "Yes", "Cafe"
"Chinese", "Yes", "Cafe"
"Continental", "Yes", "Pub"
"Fast_Food", "Yes", "Quick_Bites"

"Fast_Food", "No", "Casual_Dining"
"Italian", "No", "Delivery"
"Fast_Food", "Yes", "Delivery"
"Continental", "No", "Delivery"
"Continental", "No", "Delivery"
"Chinese", "Yes", "Delivery"
"Fast_Food", "No", "Takeaway"
"Italian", "Yes", "Quick_Bites"
"Continental", "Yes", "Takeaway"
"Fast_Food", "Yes", "Cafe"
"Continental", "No", "Cafe"
"Continental", "No", "Casual_Dining"
"Continental", "No", "Casual_Dining"

EXPERIMENT – 7

AIM: Implementation of Clustering Algorithm (K-Means Clustering and K-Means Methods)

THEORY:

K-means is a popular clustering algorithm in machine learning and data analysis. It is used to group similar data points or objects together into clusters based on their features or attributes. The primary goal of K-means clustering is to partition a dataset into K distinct, non-overlapping clusters, with each data point belonging to the cluster with the nearest mean (centroid).

K-means algorithm

k : number of clusters

n : sample feature vectors x1, x2, ..., xn

mi : the mean of the vectors in cluster i

Assume k < n.

Make initial guesses for the means m1, m2, ..., mk .

Until there are no changes in any mean.

Use the estimated means to classify the samples into clusters. o for i = 1 to k

Replace mi with the mean of all of the samples for cluster i

o end_for

o end_until

Following three steps are repeated until convergence :

Iterate till no object moves to a different group :

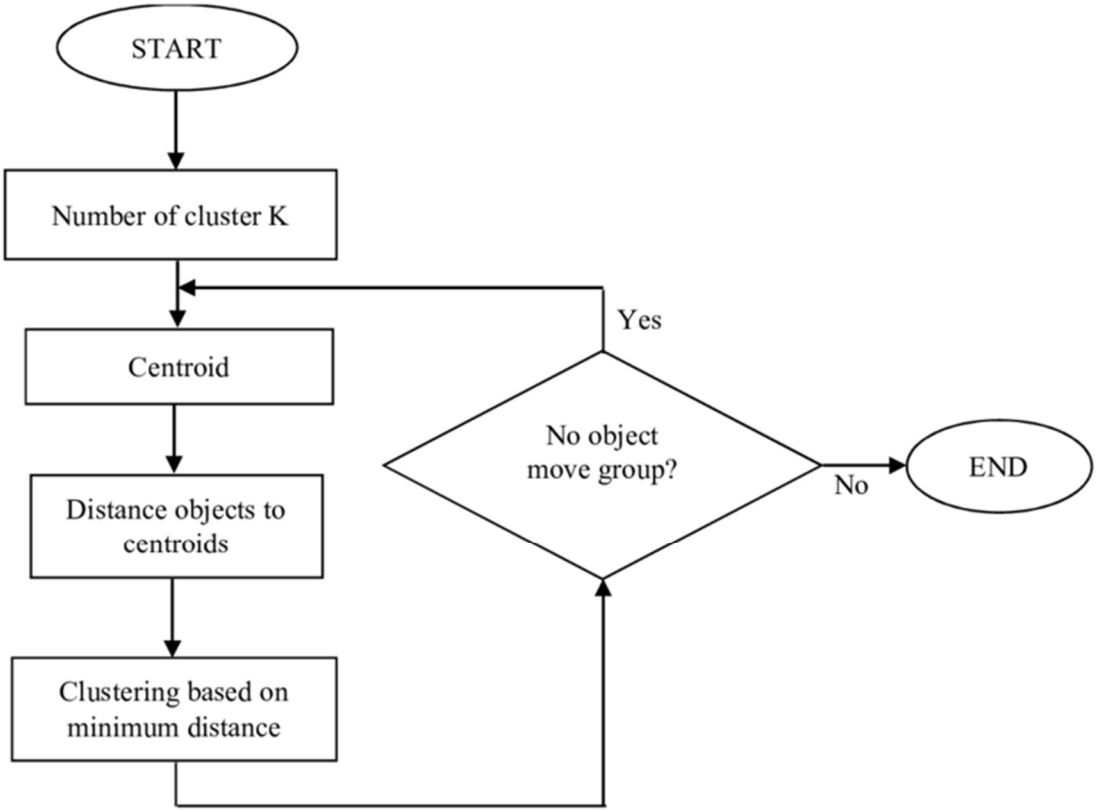
1. Find the centroid coordinate.

2. Find the distance of each object to the centroids.

3. Based on minimum distance, group the objects.

K-means is an iterative algorithm that aims to minimize the within-cluster variance, which measures the distance between data points within the same cluster. It is a relatively simple and efficient algorithm but has some limitations, such as its sensitivity to the initial placement of centroids and its assumption that clusters have similar sizes and shapes (spherical).

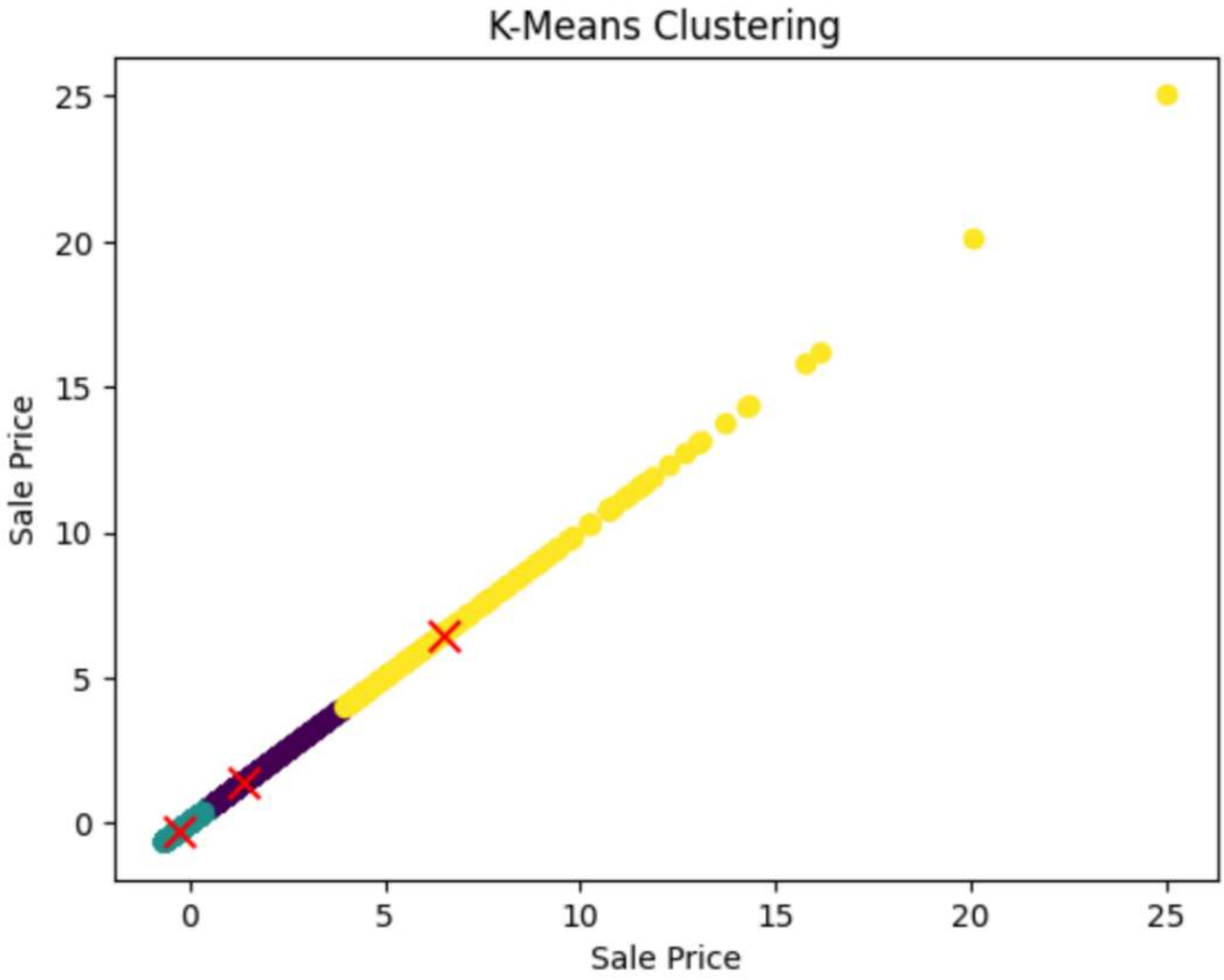
K-means clustering is widely used in various fields, including image segmentation, customer segmentation, anomaly detection, and more, for discovering patterns and groupings within data.




```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random

def most_common(lst):
    """
    Return the most frequently occurring element in a list.
    """
    return max(set(lst), key=lst.count)

def euclidean(point, data):
    """
    Return Euclidean distances between a point and a dataset.
    """
    return np.sqrt(np.sum((point - data) ** 2, axis=1))

class KMeans:
    def __init__(self, n_clusters=8, max_iter=300):
        self.n_clusters = n_clusters
        self.max_iter = max_iter

    def fit(self, X_train):
```

```
        # Initialize centroids using random data points
        self.centroids = random.sample(list(X_train), self.n_clusters)

        iteration = 0
        prev_centroids = None
        while iteration < self.max_iter:
            # Assign data points to the nearest centroid
            labels = [np.argmin([np.linalg.norm(x - c) for c in self.centroids]) for x
in X_train]

            # Update centroids as the mean of assigned data points
            new_centroids = [np.mean(X_train[np.array(labels) == i], axis=0) for i in
range(self.n_clusters)]

            # Check for convergence
            if prev_centroids is not None and all((prev == new).all() for prev, new in
zip(prev_centroids, new_centroids)):
                break

            self.centroids = new_centroids
            prev_centroids = new_centroids
            iteration += 1

    def evaluate(self, X):
        centroids = []
        centroid_idxs = []
        for x in X:
            dists = euclidean(x, self.centroids)
            centroid_idx = np.argmin(dists)
            centroids.append(self.centroids[centroid_idx])
            centroid_idxs.append(centroid_idx)
        return centroids, centroid_idxs

# Read the dataset
data = pd.read_csv('BigBasket Products.csv')

# Select the columns you want to use for clustering
selected_data = data[['sale_price']]
```

```
# Convert the selected data to a NumPy array
X = selected_data.values

# Standardize the data for the 'sale_price' column
means = np.mean(X, axis=0)
stds = np.std(X, axis=0)
X = (X - means) / stds

# Determine the number of clusters (you can adjust this)
n_clusters = 3  # Adjust the number of clusters as needed

# Maximum number of iterations
max_iterations = 100  # Adjust the maximum number of iterations as needed

# Initialize KMeans with the desired number of clusters and max iterations
kmeans = KMeans(n_clusters=n_clusters, max_iter=max_iterations)

# Fit the data
kmeans.fit(X)

# Get the cluster labels from your KMeans object
labels = kmeans.evaluate(X)[1]

# Add cluster labels to the original dataset
data['cluster'] = labels

# View results
plt.scatter(X[:, 0], X[:, 0], c=labels, cmap='viridis')
plt.scatter([centroid[0] for centroid in kmeans.centroids], [centroid[0] for centroid
in kmeans.centroids], c='red', marker='x', s=100)
plt.title("K-Means Clustering")
plt.xlabel('Sale Price')
plt.ylabel('Sale Price')
plt.show()
```


K-Means Clustering
Sale Price
Sale Price
0
5
10
15
20
25

Experiment 8 (Hierarchical Clustering)

Aim : Implementation of any one Hierarchical Clustering method.

Theory :

Clustering is the method of dividing objects into sets that are similar, and dissimilar to the objects belonging to another set. There are two different types of clustering, each divisible into two subsets

1. Hierarchical clustering
- Agglomerative
- Divisive
- Partial clustering
- K-means
- Fuzzy c-means

Every kind of clustering has its own purpose and numerous use cases.
Customer Segmentation

In customer segmentation, clustering can help answer the questions:

- What people belong together?
- How do we group them together?

Social Network Analysis

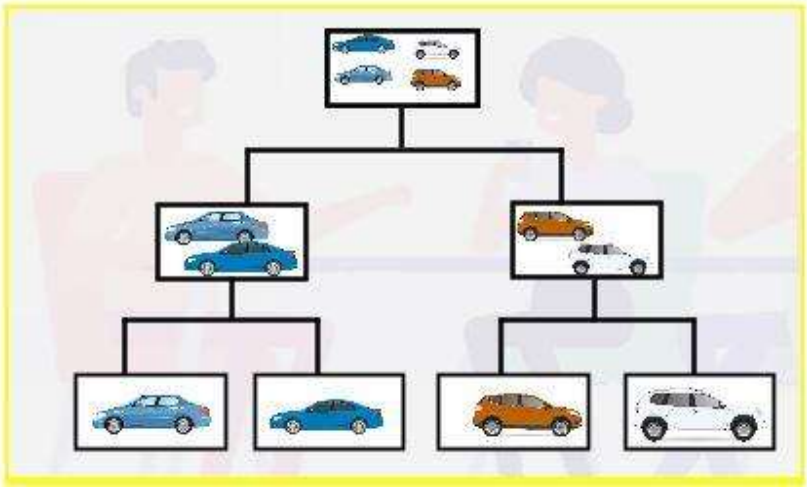
User personas are a good use of clustering for social networking analysis. We can look for similarities between people and group them accordingly.

City Planning
Clustering is popular in the realm of city planning. Planners need to check that an industrial zone isn't near a residential area, or that a commercial zone somehow wound up in the middle of an industrial zone.

An Example of Hierarchical Clustering

Hierarchical clustering is separating data into groups based on some measure of similarity, finding a way to measure how they’re alike and different, and further narrowing down the data.

Let's consider that we have a set of cars and we want to group similar ones together. Look at the image shown below:



For starters, we have four cars that we can put into two clusters of car types: sedan and SUV. Next, we'll bunch the sedans and the SUVs together. For the last step, we can group everything into one cluster and finish when we’re left with only one cluster.

Types of Hierarchical Clustering

Hierarchical clustering is divided into:

- Agglomerative
- Divisive

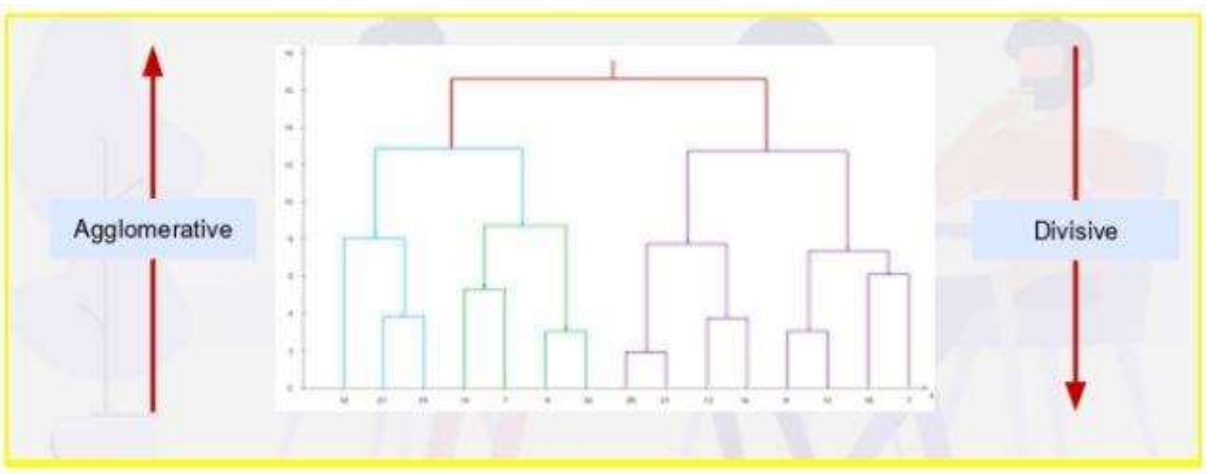
Divisive Clustering

Divisive clustering is known as the top-down approach. We take a large cluster and start dividing it into two, three, four, or more clusters.

Agglomerative Clustering

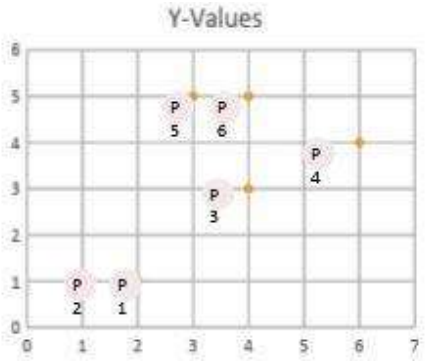
Agglomerative clustering is known as a bottom-up approach. Consider it as bringing things together.

Both of these approaches are as shown below:



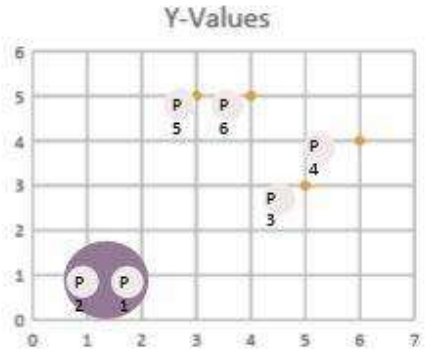
How Does Hierarchical Clustering Work?

Let's consider that we have a few points on a 2D plane with x-y coordinates.

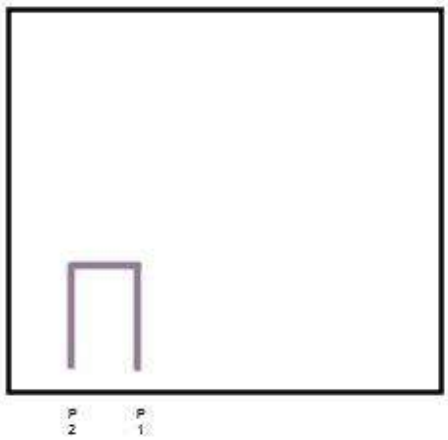


Here, each data point is a cluster of its own. We want to determine a way to compute the distance between each of these points. For this, we try to find the shortest distance between any two data points to form a cluster.

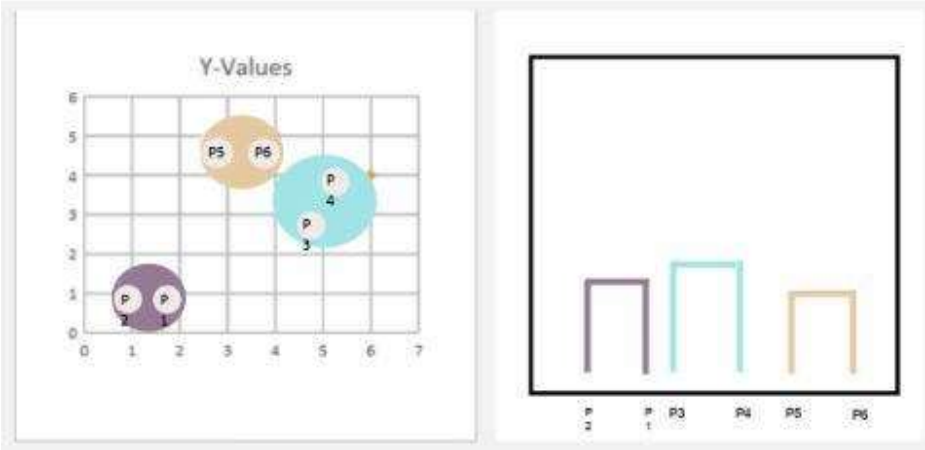
Once we find those with the least distance between them, we start grouping them together and forming clusters of multiple points.



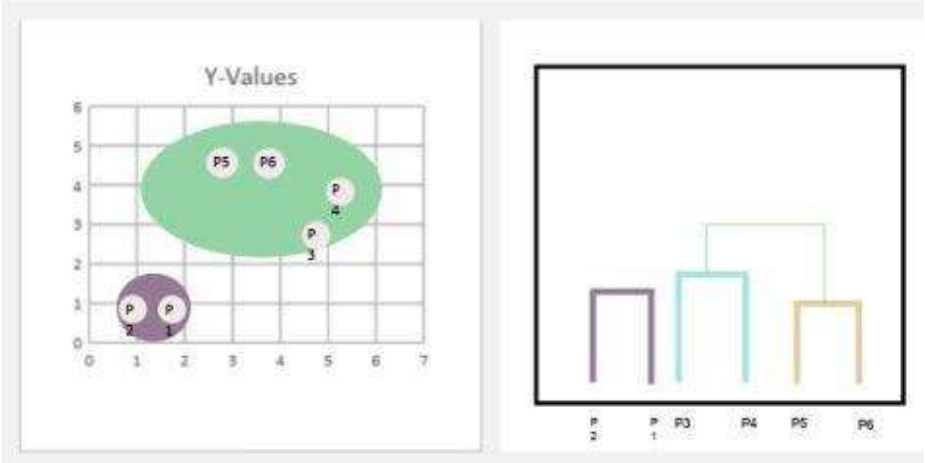
This is represented in a tree-like structure called a dendrogram.



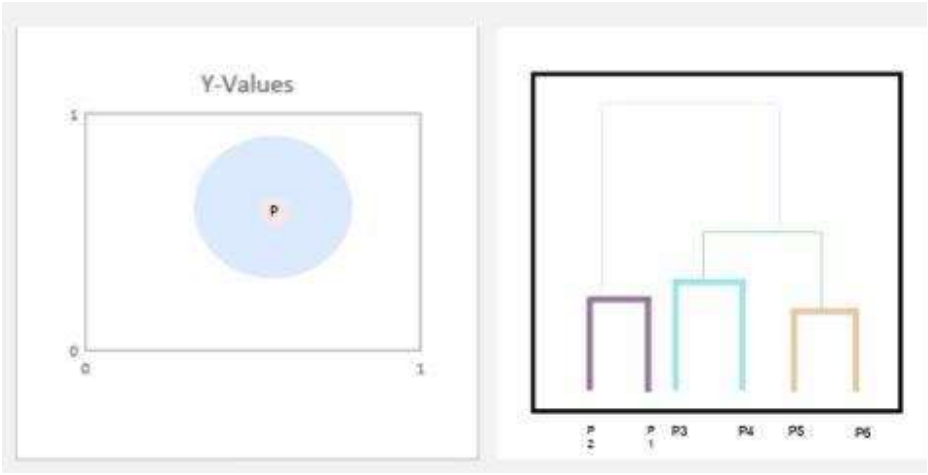
As a result, we have three groups: P1-P2, P3-P4, and P5-P6. Similarly, we have three dendrograms, as shown below:



In the next step, we bring two groups together. Now the two groups P3-P4 and P5-P6 are all under one dendrogram because they're closer together than the P1-P2 group. This is as shown below:



We finish when we’re left with one cluster and finally bring everything together.



You can see how the cluster on the right went to the top with the gray hierarchical box connecting them.

What is the Distance Measure?

Distance measure determines the similarity between two elements and it influences the shape of the clusters.

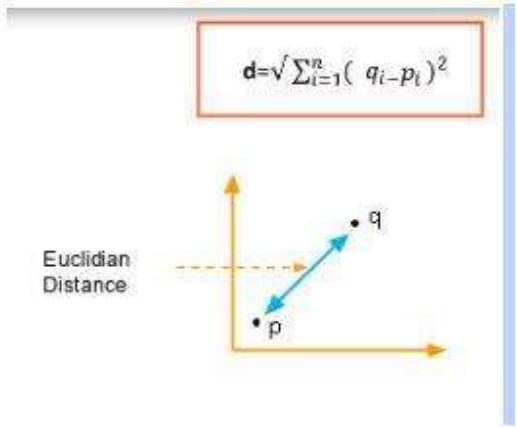
Some of the ways we can calculate distance measures include:

- Euclidean distance measure
- Squared Euclidean distance measure
- Manhattan distance measure
- Cosine distance measure

Euclidean Distance Measure

The most common method to calculate distance measures is to determine the distance between the two points. Let's say we have a point P and point Q: the Euclidean distance is the direct straight-line distance between the two points.

The formula for distance between two points is shown below:



As this is the sum of more than two dimensions, we calculate the distance between each of the different dimensions squared and then take the square root of that to get the actual distance between them.

Squared Euclidean Distance Measurement

This is identical to the Euclidean measurement method, except we don't take the square root at the end. The formula is shown below:

$$d=\sum_{i=1}^{n}(q_i-p_i)^2$$

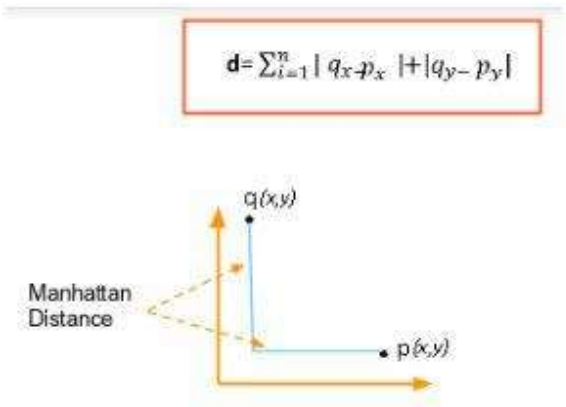
Depending on whether the points are farther apart or closer together, then the difference in distances can be computed faster by using squared Euclidean distance measurement.

While this method gives us the exact distance, it won't make a difference when calculating which is smaller and which is larger. Removing the square root can make the computation faster.

Manhattan Distance Measurement

This method is a simple sum of horizontal and vertical components or the distance between two points measured along axes at right angles.

The formula is shown below:



This method is different because you're not looking at the direct line, and in certain cases, the individual distances measured will give you a better result.

Most of the time, you'll go with the Euclidean squared method because it's faster. But when using the Manhattan distance, you measure either the X difference or the Y difference and take the absolute value of it.

Code :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (12, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
df = pd.read_csv('/content/swiggy (2).csv').head(15)
df
```

	ID	Area	City	Restaurant	Price	Avg ratings	Total ratings	Food type	Address	Delivery time
0	211	Koramangala	Bangalore	Tandoor Hut	300.0	4.4	100	Biryani,Chinese,North Indian,South Indian	5Th Block	59
1	221	Koramangala	Bangalore	Tunday Kababi	300.0	4.1	100	Mughlai,Lucknowi	5Th Block	56
2	246	Jogupalya	Bangalore	Kim Lee	650.0	4.4	100	Chinese	Double Road	50
3	248	Indiranagar	Bangalore	New Punjabi Hotel	250.0	3.9	500	North Indian,Punjabi,Tandoor,Chinese	80 Feet Road	57
4	249	Indiranagar	Bangalore	Nh8	350.0	4.0	50	Rajasthani,Gujarati,North Indian,Snacks,Desser...	80 Feet Road	63
5	254	Indiranagar	Bangalore	Treat	800.0	4.5	100	Mughlai,North Indian	100 Feet Road	56
6	258	Indiranagar	Bangalore	Chinita Real Mexican Food	100	4.5	500	Mexican,Beverages,Salads	Double Road	53
7	263	Koramangala	Bangalore	Cupcake Noggins - Cakespastries And Desserts	150.0	4.3	100	Desserts,British,Bakery,Pizzas,Snacks	4Th Block	57
8	267	Domlur	Bangalore	Tea Brew	350.0	4.1	100	American,Italian,Beverages,Continental,Chinese...	Double Road	57

```
dataset = df[['Price', 'Avg ratings']]
dataset
```

	Price	Avg ratings
0	300.0	4.4
1	300.0	4.1
2	650.0	4.4
3	250.0	3.9
4	350.0	4.0
5	800.0	4.5
6	1000.0	4.5
7	150.0	4.3
8	350.0	4.1
9	300.0	4.0
10	400.0	4.4
11	250.0	4.3
12	532.0	4.2
13	500.0	2.9
14	150.0	4.3

```
X = np.array(dataset)
X
```

```
array([[ 300. ,    4.4],
       [ 300. ,    4.1],
       [ 650. ,    4.4],
       [ 250. ,    3.9],
       [ 350. ,    4. ],
       [ 800. ,    4.5],
       [1000. ,    4.5],
       [ 150. ,    4.3],
       [ 350. ,    4.1],
       [ 300. ,    4. ],
       [ 400. ,    4.4],
       [ 250. ,    4.3],
       [ 532. ,    4.2],
       [ 500. ,    2.9],
       [ 150. ,    4.3]])
```

```
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
def distance_calculate(self,sample1,sample2):
dist = []
for i in range(len(sample1)):
for j in range(len(sample2)):
try:
dist.append(np.linalg.norm(np.array(sample1[i])-np.array(sample2[j])))
except:
dist.append(self.intersampledist(sample1[i],sample2[j]))
return min(dist)
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)

def interclusterdist(self,cl,sample):
if sample[0]!='<class \'list\'>':
sample = [sample]
dist = []
for i in range(len(cl)):
for j in range(len(sample)):
dist.append(np.linalg.norm(np.array(cl[i])-np.array(sample[j])))
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]])
progression[sample_ind_needed[0]].append(progression[sample_ind_needed[1]])
progression[sample_ind_needed[0]] = [progression[sample_ind_needed[0]]]
```

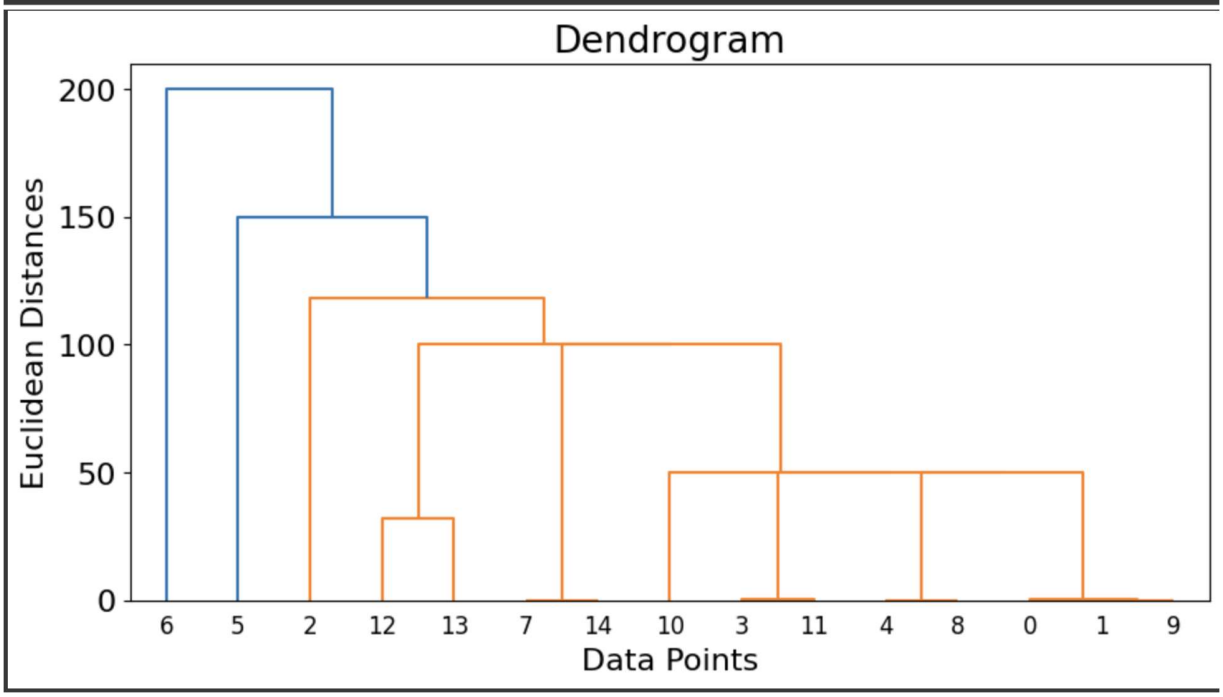
```
v = progression.pop(sample_ind_needed[1])
m = len(samples)
print('Progression(Current Sample) :-',progression)
print('Cluster attained :-',progression[sample_ind_needed[0]])
print('Sample size after clustering :-',m)
print('\n')
# Calculate the linkage matrix for hierarchical clustering
linkage_matrix = linkage(X, method='single')

# Plot the dendrogram
plt.figure(figsize=(10, 5))
dendrogram(linkage_matrix)
plt.title('Dendrogram')
plt.xlabel('Data Points')
plt.ylabel('Euclidean Distances')
plt.show()
```

```
Sample size before clustering :-  15
Cluster Node 1 :- [7]
Cluster Node 2 :- [14]
Progression(Current Sample) :- [[0], [1], [2], [3], [4], [5], [6], [[7, [14]]], [8], [9], [10], [11], [12], [13]]
Cluster attained :- [[7, [14]]]
Sample size after clustering :- 14


Sample size before clustering :-  14
Cluster Node 1 :- [1]
Cluster Node 2 :- [4]
Progression(Current Sample) :- [[0], [[1, [4]]], [2], [3], [5], [6], [[7, [14]]], [8], [9], [10], [11], [12], [13]]
Cluster attained :- [[1, [4]]]
Sample size after clustering :- 13


Sample size before clustering :-  13
Cluster Node 1 :- [[1, [4]]]
Cluster Node 2 :- [[1, [4]]]
Progression(Current Sample) :- [[0], [2], [3], [5], [6], [[7, [14]]], [8], [9], [10], [11], [12], [13]]
Cluster attained :- [2]
Sample size after clustering :- 12


Sample size before clustering :-  12
Cluster Node 1 :- [2]
Cluster Node 2 :- [8]
Progression(Current Sample) :- [[0], [[2, [8]]], [3], [5], [6], [[7, [14]]], [9], [10], [11], [12], [13]]
Cluster attained :- [[2, [8]]]
Sample size after clustering :- 11
```

```
Sample size before clustering :-  11
Cluster Node 1 :- [3]
Cluster Node 2 :- [11]
Progression(Current Sample) :- [[0], [[2, [8]]], [[3, [11]]], [5], [6], [[7, [14]]], [9], [10], [12], [13]]
Cluster attained :- [[3, [11]]]
Sample size after clustering :- 10


Sample size before clustering :-  10
Cluster Node 1 :- [0]
Cluster Node 2 :- [9]
Progression(Current Sample) :- [[[0, [9]]], [[2, [8]]], [[3, [11]]], [5], [6], [[7, [14]]], [10], [12], [13]]
Cluster attained :- [[0, [9]]]
Sample size after clustering :- 9


Sample size before clustering :-  9
Cluster Node 1 :- [12]
Cluster Node 2 :- [13]
Progression(Current Sample) :- [[[0, [9]]], [[2, [8]]], [[3, [11]]], [5], [6], [[7, [14]]], [10], [[12, [13]]]]
Cluster attained :- [[12, [13]]]
Sample size after clustering :- 8


Sample size before clustering :-  8
Cluster Node 1 :- [[0, [9]]]
Cluster Node 2 :- [[2, [8]]]
Progression(Current Sample) :- [[[[0, [9]], [[2, [8]]]]], [[3, [11]]], [5], [6], [[7, [14]]], [10], [[12, [13]]]]
Cluster attained :- [[[0, [9]], [[2, [8]]]]]
Sample size after clustering :- 7
```

```
Sample size before clustering :-  7
Cluster Node 1 :- [[[0, [9]], [[2, [8]]]]]
Cluster Node 2 :- [[3, [11]]]
Progression(Current Sample) :- [[[[[0, [9]], [[2, [8]]]], [[3, [11]]]]], [5], [6], [[7, [14]]], [10], [[12, [13]]]]
Cluster attained :- [[[[0, [9]], [[2, [8]]]], [[3, [11]]]]]
Sample size after clustering :- 6


Sample size before clustering :-  6
Cluster Node 1 :- [[[[0, [9]], [[2, [8]]]], [[3, [11]]]]]
Cluster Node 2 :- [10]
Progression(Current Sample) :- [[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]]], [5], [6], [[7, [14]]], [[12, [13]]]]
Cluster attained :- [[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]]]
Sample size after clustering :- 5


Sample size before clustering :-  5
Cluster Node 1 :- [[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]]]
Cluster Node 2 :- [[7, [14]]]
Progression(Current Sample) :- [[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]]], [5], [6], [[12, [13]]]]
Cluster attained :- [[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]]]
Sample size after clustering :- 4


Sample size before clustering :-  4
Cluster Node 1 :- [[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]]]
Cluster Node 2 :- [[12, [13]]]
Progression(Current Sample) :- [[[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]]], [5], [6]]
Cluster attained :- [[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]]]
Sample size after clustering :- 3
```

```
Sample size before clustering :-  3
Cluster Node 1 :- [5]
Cluster Node 2 :- [6]
Progression(Current Sample) :- [[[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]]], [[5, [6]]]]
Cluster attained :- [[5, [6]]]
Sample size after clustering :- 2


Sample size before clustering :-  2
Cluster Node 1 :- [[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]]]
Cluster Node 2 :- [[5, [6]]]
Progression(Current Sample) :- [[[[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]], [[5, [6]]]]]
Cluster attained :- [[[[[[[[0, [9]], [[2, [8]]]], [[3, [11]]]], [10]], [[7, [14]]]], [[12, [13]]]], [[5, [6]]]]]
Sample size after clustering :- 1
```



Experiment - 9

Aim : Implementation of Association rule mining algorithm (Apriori algorithm)

Theory:

The frequent pattern mining algorithm is one of the most important techniques of data mining to discover relationships between different items in a dataset. These relationships are represented in the form of association rules. It helps to find the irregularities in data.

FPM has many applications in the field of data analysis, software bugs, cross-marketing, sale campaign analysis, market basket analysis, etc.

Frequent itemsets discovered through Apriori have many applications in data mining tasks. Tasks such as finding interesting patterns in the database, finding sequence and Mining of association rules are the most important of them.

Association rules apply to supermarket transaction data, that is, to examine the customer behavior in terms of the purchased products. Association rules describe how often the items are purchased together.

Association Rules

Association Rule Mining is defined as:

"Let I= { …} be a set of 'n' binary attributes called items. Let D= { ….} be a set of transactions called a database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of form X->Y where X, Y? I and X?Y=?. The set of items X and Y are called antecedent and consequent of the rule respectively."

Learning of Association rules is used to find relationships between attributes in large databases. An association rule, A=> B, will be of the form" for a set of transactions, some value of itemset A determines the values of itemset B under the condition in which minimum support and confidence are met".

Support and Confidence can be represented by the following example:

Bread=> butter [support=2%, confidence-60%]

The above statement is an example of an association rule. This means that there is a 2% transaction that bought bread and butter together and there are 60% of customers who bought bread as well as butter.

Support and Confidence for Itemset A and B are represented by formulas:

$$\text{Support (A)} = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

$$\text{Confidence (A}\longrightarrow\text{B)} = \frac{\text{Support(AUB)}}{\text{Support(A)}}$$

Association rule mining consists of 2 steps:

1. Find all the frequent itemsets.
2. Generate association rules from the above frequent itemsets.

Why Frequent Itemset Mining?

Frequent itemset or pattern mining is broadly used because of its wide applications in mining association rules, correlations and graph patterns constraint that is based on frequent patterns, sequential patterns, and many other data mining tasks.

Apriori Algorithm – Frequent Pattern Algorithms

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps “join” and “prune” to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

Apriori says:

The probability that item I is not frequent is if:

- P(I) < minimum support threshold, then I is not frequent.
- P (I+A) < minimum support threshold, then I+A is not frequent, where A also belongs to the itemset.
- If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Anti Monotone property.

The steps followed in the Apriori Algorithm of data mining are:

1. **Join Step**: This step generates (K+1) itemset from K-itemsets by joining each item with itself.
2. **Prune Step**: This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

Steps In Apriori

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.

#1) In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate. The algorithm will count the occurrences of each item.

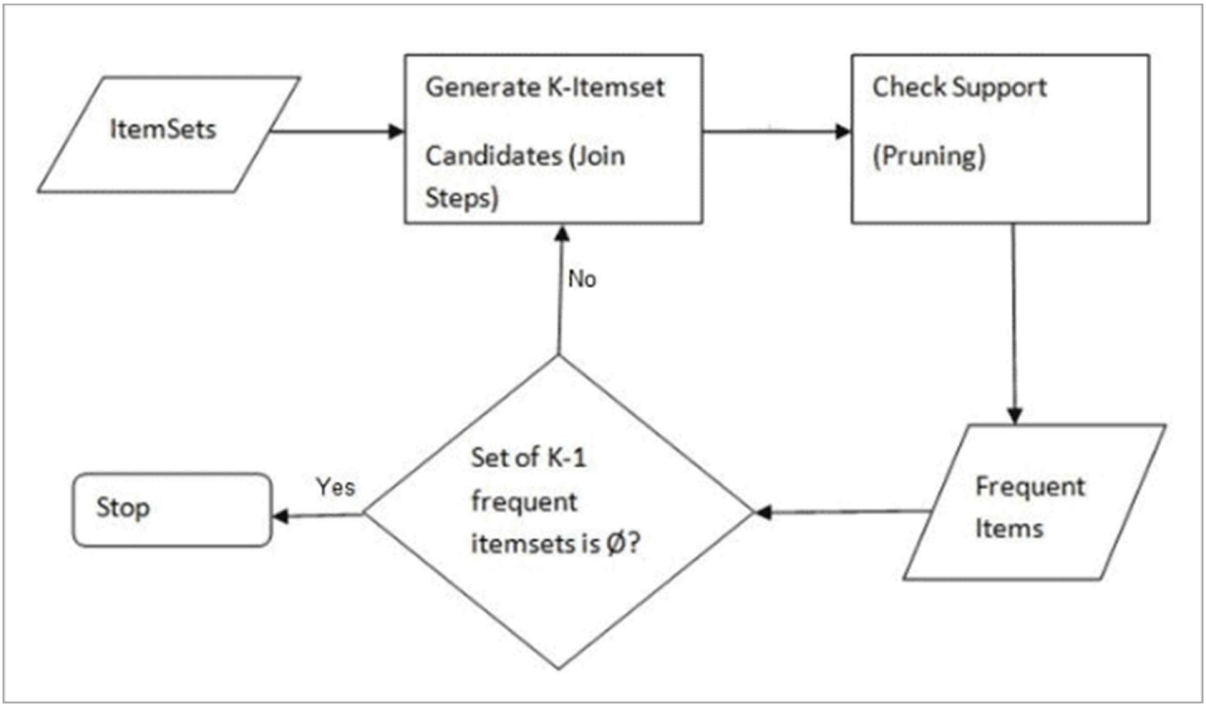
#2) Let there be some minimum support, min_sup (eg 2). The set of 1 – itemsets whose occurrence is satisfying the min sup are determined. Only those candidates which count more than or equal to min_sup, are taken ahead for the next iteration and the others are pruned.

#3) Next, 2-itemset frequent items with min_sup are discovered. For this in the join step, the 2-itemset is generated by forming a group of 2 by combining items with itself.

#4) The 2-itemset candidates are pruned using min-sup threshold value. Now the table will have 2 –itemsets with min-sup only.

#5) The next iteration will form 3 –itemsets using join and prune step. This iteration will follow anti monotone property where the subsets of 3-itemsets, that is the 2 –itemset subsets of each group fall in min_sup. If all 2-itemset subsets are frequent then the superset will be frequent otherwise it is pruned.

#6) Next step will follow making 4-itemset by joining 3-itemset with itself and pruning if its subset does not meet the min_sup criteria. The algorithm is stopped when the most frequent itemset is achieved.



Advantages

1. Easy to understand algorithm
2. Join and Prune steps are easy to implement on large itemsets in large databases

Disadvantages

1. It requires high computation if the itemsets are very large and the minimum support is kept very low.
2. The entire database needs to be scanned.

Methods To Improve Apriori Efficiency

Many methods are available for improving the efficiency of the algorithm.

1. **Hash-Based Technique:** This method uses a hash-based structure called a hash table for generating the k-itemsets and its corresponding count. It uses a hash function for generating the table.
2. **Transaction Reduction:** This method reduces the number of transactions scanning in iterations. The transactions which do not contain frequent items are marked or removed.
3. **Partitioning:** This method requires only two database scans to mine the frequent itemsets. It says that for any itemset to be potentially frequent in the database, it should be frequent in at least one of the partitions of the database.
4. **Sampling:** This method picks a random sample S from Database D and then searches for a frequent itemset in S. It may be possible to lose a global frequent itemset. This can be reduced by lowering the min_sup.
5. **Dynamic Itemset Counting:** This technique can add new candidate itemsets at any marked start point of the database during the scanning of the database.

Applications Of Apriori Algorithm

Some fields where Apriori is used:

1. **In Education Field:** Extracting association rules in data mining of admitted students through characteristics and specialties.
2. **In the Medical field:** For example Analysis of the patient's database.
3. **In Forestry:** Analysis of probability and intensity of forest fire with the forest fire data.
4. Apriori is used by many companies like Amazon in the **Recommender System** and by Google for the auto-complete feature.

Code :

```
import pandas as pd
import itertools

# Load your CSV data into a DataFrame
df = pd.read_csv("/content/swiggy (2).csv")

# Create a list of transactions
transactions = []

# Iterate through the rows in the DataFrame
for index, row in df.iterrows():
transaction = [str(row[i]) for i in range(10)] # Assuming you want to
consider the first 10 columns
transactions.append(transaction)

# Apriori algorithm parameters
min_support = 0.0045 # Adjust the minimum support as needed
min_confidence = 0.2 # Adjust the minimum confidence as needed

# Function to generate candidate itemsets
def generate_candidates(Lk, k):
candidates = []
for i in range(len(Lk)):
for j in range(i + 1, len(Lk)):
itemset1, itemset2 = list(Lk[i]), list(Lk[j])
itemset1.sort()
itemset2.sort()
if itemset1[:k - 2] == itemset2[:k - 2]:
candidates.append(set(itemset1).union(itemset2))
return candidates

# Function to prune itemsets that don't meet support
def prune_itemsets(Ck, transactions, min_support):
counts = {}
for transaction in transactions:
for itemset in Ck:
if itemset.issubset(transaction):
if itemset in counts:
counts[itemset] += 1
```

```
else:
counts[itemset] = 1
frequent_itemsets = [itemset for itemset in Ck if
counts.get(frozenset(itemset), 0) / len(transactions) >= min_support]
return frequent_itemsets

# Function to generate association rules
def generate_rules(frequent_itemsets, min_confidence):
rules = []
for itemset in frequent_itemsets:
if len(itemset) > 1:
itemset = list(itemset)
for i in range(len(itemset)):
antecedent = frozenset(itemset[:i])
consequent = frozenset(itemset[i:])
confidence = counts[frozenset(itemset)] / counts[antecedent]
if confidence >= min_confidence:
rules.append((antecedent, consequent, confidence))
return rules

# Apriori algorithm
def apriori(transactions, min_support, min_confidence):
global counts
Ck = [set(item) for item in set(itertools.chain(*transactions))]
Ck = prune_itemsets(Ck, transactions, min_support)
Lk = Ck
k = 2
while Lk:
print(f"Frequent itemsets of length {k - 1}: {Lk}")
Ck = generate_candidates(Lk, k)
Ck = prune_itemsets(Ck, transactions, min_support)
Lk = Ck
k += 1
counts = {}
for transaction in transactions:
for itemset in Lk:
if itemset.issubset(transaction):
if itemset in counts:
counts[itemset] += 1
else:
```

```
counts[itemset] = 1
rules = generate_rules(Lk, min_confidence)
for rule in rules:
antecedent, consequent, confidence = rule
print(f"Rule: {list(rule.items)} =>
{list(rule.ordered_statistics[0].items_base)}")
print(f"Support: {rule.support}")
print(f"Confidence: {rule.ordered_statistics[0].confidence}")
print("Lift: "+ str(item [2][0][3]))
print("-------")


apriori(transactions, min_support, min_confidence)
```

OUTPUT:-

```
==========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
```

```
==========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
==========
```

```
Rule: ['4.4', 'Double Road', '50', 'Bangalore', '650.0', '100'] => ['4.4']
Support: 0.1111111111111111
Confidence: 0.5
--------
Rule: ['4.4', '50', 'Bangalore', 'Jogupalya', '650.0', '100'] => ['4.4']
Support: 0.1111111111111111
Confidence: 0.5
--------
Rule: ['4.4', '50', 'Kim Lee', 'Bangalore', '650.0', '100'] => ['4.4']
Support: 0.1111111111111111
Confidence: 0.5
--------
```

```
Rule: ['Cupcake Noggins - Cakespastries And Desserts', '57', 'Koramangala', 'Bangalore', '4Th Block', '100'] => ['4Th Block']
Support: 0.1111111111111111
Confidence: 1.0
--------
Rule: ['57', 'Koramangala', 'Bangalore', 'Desserts,British,Bakery,Pizzas,Snacks', '4Th Block', '100'] => ['4Th Block']
Support: 0.1111111111111111
Confidence: 1.0
--------
```

```
Rule: ['4.5', '800.0', '100 Feet Road', 'Indiranagar', '56', '100', '254'] => ['100 Feet Road']
Support: 0.1111111111111111
Confidence: 1.0
--------
Rule: ['4.5', '800.0', 'Mughlai,North Indian', '100 Feet Road', '56', '100', '254'] => ['100 Feet Road']
Support: 0.1111111111111111
Confidence: 1.0
--------
Rule: ['Treat', '4.5', '800.0', '100 Feet Road', '56', '100', '254'] => ['100 Feet Road']
Support: 0.1111111111111111
Confidence: 1.0
--------
```

```
=========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
```

```
=========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> Rajasthani,Gujarati,North Indian,Snacks,Desserts,Beverages,Thalis,Chaat
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
Rule: 350.0 -> 80 Feet Road
Support: 0.2
Confidence: 1.0
Lift: 5.0
=========
```

SUBJECT : Data-Warehouse And Mining

AIM : Implementation of Page rank/HITS algorithm

THEORY :

Hyperlink-Induced Topic Search (HITS) Algorithm

HITS is an algorithm used in link analysis to discover and rank webpages relevant to a search. It utilizes the concept that a quality website should both link to other relevant sites and be linked by other important sites.

Key Concepts:

1. Authority A node is considered high-quality if many high-quality nodes link to it.

2. Hub: A node is considered high-quality if it links to many high-quality nodes.

Algorithm Steps:

- Initialize the hub and authority of each node with a value of 1.
- In each iteration, update the hub and authority of every node in the graph.
- The new authority is the sum of the hub of its parents.
- The new hub is the sum of the authority of its children.
- Normalize the new authority and hub.

How HITS Works:

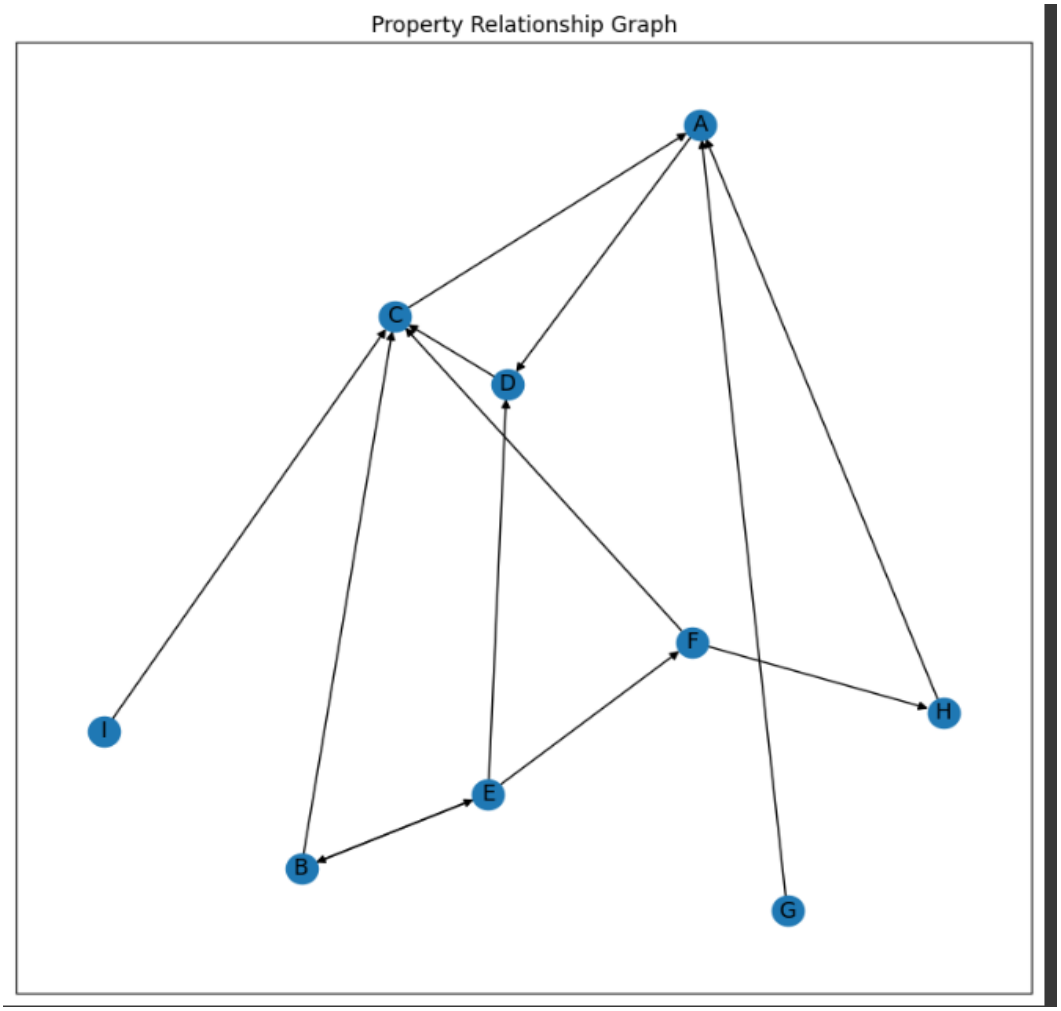
1. Retrieve the search query data.

2. Computation is performed considering the search results, excluding other websites.
3. Define authoritative and hub values.
4. Initiate an iterative process involving authority and hub updates.
5. Authoritative websites are those with valuable content.
6. HITS is executed at query time and considers the keyword or content people are searching for.
7. HITS computes two scores per document (authority and hub).
8. It is processed on a small subset of documents, not all documents like PageRank.
9. HITS is not commonly used by search engines.

This structured format provides a concise summary of the HITS algorithm, its key concepts, and how it differs from PageRank.

Code and output:

```
import networkx as nx
import matplotlib.pyplot as plt

# Create a directed graph
graph = nx.DiGraph()

# Define relationships (customize this based on your dataset)
edges = [
    ('A', 'D'), ('B', 'C'), ('B', 'E'), ('C', 'A'),
    ('D', 'C'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
    ('F', 'C'), ('F', 'H'), ('G', 'A'),
    ('I', 'C'), ('H', 'A')
]

graph.add_edges_from(edges)

# Visualize the graph (optional)
plt.figure(figsize=(10, 10))
pos = nx.spring_layout(graph, seed=42)
nx.draw_networkx(graph, pos, with_labels=True)
plt.title("Property Relationship Graph")
plt.show()

# Calculate HITS scores
hubs, authorities = nx.hits(graph, max_iter=10, normalized=True)

# Print HITS scores
for node in graph.nodes:
    print(f"Node {node}:")
    print(f"Hub Score: {hubs[node]}")
    print(f"Authority Score: {authorities[node]}\n")
```




```
Node A:
Hub Score: -2.0458796793463883e-19
Authority Score: 0.0

Node D:
Hub Score: 0.21922359359558488
Authority Score: -5.976351314820015e-19

Node B:
Hub Score: 0.28077640640441526
Authority Score: -1.5299459365939238e-16

Node C:
Hub Score: 0.0
Authority Score: 0.6403882032022077

Node E:
Hub Score: -1.311408874461035e-16
Authority Score: 0.17980589839889644

Node F:
Hub Score: 0.2807764064044152
Authority Score: -2.294918904890886e-16

Node H:
Hub Score: 0.0
Authority Score: 0.17980589839889624

Node G:
Hub Score: 0.0
Authority Score: 1.5299459365939238e-16

Node I:
Hub Score: 0.21922359359558488
Authority Score: -1.147459452445443e-16
```

s

Parth D
32
C12

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Assignment no: 1

1. What is metadata? Explain data warehouse metadata with Example.

⇒ Data warehouse metadata plays a crucial role in managing and understanding the data stored within a warehouse. Metadata are pieces of information stored in one or more special purpose metadata repositories. It refers to the data about the data, providing information about the structure, content and context of data within the data warehouse.
Metadata encompasses various types of information about data, including data definitions, data image/lineage, data transformation, data sources and more. Its primary purpose is to help users and data professionals understand, manage and utilize data effectively.
Metadata in repositories can include:-

a. Information on the contents of data warehouse, their locations and their structures.
b. Information on the processes that take place in the data warehouse back-stage, concerning the refreshment of the warehouse with clean, up-to-date, semantically and structurally reconciled data.
c. Information on the implicit semantics of the data, along with any other kind of data that aids the end user exploit the information about the warehouse.
d. Information on the infrastructure and physical characteristics of data components and the sources of the data warehouse.
e. Information including security, authentication and usage statistics that aids the administrator, tunes the operation of warehouse.

Types of Metadata:

a. Technical metadata:
Describes the technical aspects of data, including data types, field lengths, data source locations and storage formats.
It helps in data integration and ETL (Extraction, Transformation and Loading) processes.

b. Buisness Metadata:
Provides buisness context to data, such as data definitions, buisness rules, and the relationship between different data elements.
It aids buisness users in making informed decisions based on the data.

c. Operational Metadata:
Focusses on tracking the operational aspects of data, including the loading times, usage statistics and access patterns.
Helps in monitoring data warehouse performance and identifying bottlenecks.

d. Data-Lineage metadata:
Shows the history of data, including its source, transformations, and usage over time.
Useful for auditing, data quality assurance and compliance purposes.

Benefits of Data warehouse metadata:

a. Data understanding:

Metadata provides a clear understanding of the data, reducing confusion and misinterpretations.

b. Data Quality :
It helps in data quality assessment by highlighting issues like missing or inconsistent data.

c. Data Governance and Compliance:
Metadata assists in enforcing data governance policies and ensures compliance with regulations.

d. Efficient data retrieval :
Users can locate and retrieve data more efficiently with metadata, reducing querying time.

e. Data Lineage :
It supports tracking the origin of data and analyzing the impact of changes.

f. Data Integration :
Metadata facilitates the integration of data from various sources by providing information on data structures and transformation.

Disadvantages :

a. Managing metadata can be complex and time-consuming, especially in large data warehouses.

b. Ensuring metadata accuracy and consistency is a challenge.

Let us explore metadata with an example of a Library management system:

To efficiently manage the book and help library users find the right books, one would use metadata. Here's how meta data can be applied in this scenario:

1. Title (Descriptive metadata): Describes title of book.
2. Author (Descriptive metadata): Provides info about author.
3. Publication Date (Descriptive): Indicates date of publication.
4. Genre (Descriptive): Categorize into a literary genre
5. ISBN (Technical metadata): Unique id for books
6. Location (Technical): Specifies the physical location in library
7. Availability (Operational): Indicates the availability of book.
8. Borrower history (Operational): History of book borrowers
9. Related books (Relational): Sequels or books by same author.
10. Catalog Date (Operational): Records the date of book addition.
11. Usage Statistics (Operational): Includes data on how frequently the book has been checked out.

Meta Data	Values
Title	To kill a Mockingbird
Author	Harper Lee
Publication Date	1960
Genre	Fiction
ISBN	0-06-112008-1
Location	Aisle 2, Shelf 3
Availability	Available
Borrower history	John Smith (15/9/23)
Related Books	Go set a watchman
Cataloging date	10/10/23
Usage	Checked out 25 times

Parth D
32
C12

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

ASSIGNMENT NO: 2

a. Multilevel Association rules and Multidimensional association rules.

⇒ Multilevel association rules:
Association rule created from mining information at different degrees of reflection are called various level or staggered association rules.
These can be mined effectively utilizing idea progressions under a help certainty system.
Rules at a high level may add to good judgement while rules at low level may not be valuable consistently.
Utilizing uniform least help for all levels:
i) At the point when a uniform least help edge is utilized the pursuit is rearranged.
ii) The technique is likewise straightforward in that clients are needed to indicate just a single least help edge.
iii) A similar least help edge is utilized when mining at each degree of deliberation.

Needs of multilevel rule:
- Sometimes at low data level, data does not show any significant pattern but there is useful inteformation hiding behind it.
- The aim is to find the hidden information in or between levels of abstraction.

Approaches used in multilevel association rule mining:
i) Uniform support: At the point when a uniform least help edge is used the search methodology is simplified. The technique is likewise basic in that clients are needed to

determine just a single least help threshold. In advancement technique can be adopted, based on the information that a progenitor is a superset of its descendants

ii) Reduce support: It has other types like level by level independence level cross seperating by single things, level cross seperating by k-itemset, level by level independence and more. That use reduced minimum support at lower levels

iii) Grou Based Support: The group wise threshold value for support and confidence is input by user or expert. The group is selected based on a product item set because often expert has insight as to which groups are more important.

Multidimensional association rule:

In this type, the quantity can be absolute or quantitive.

- Quantitive characteristics are numeric and consolidates order.
- Numeric traits should be descrihized.
- Multidimensional affiliation that comprises of more than one measurement.

Example: buys (x, "IBM laptop computer") buys (x, "HP inkjet printer")

Approaches in mining multi-dimensional affiliation rules:-

1) Using static discretization of quantative qualities.

- Discretization is static and happens preceeding mining.
- Descretized ascribes are treated as unmitigated.
- Use apriori calculation.

2) Using powerful discretization of quantative traits:

- Known as mining Quantitative Association Rule.
- Numeric properties are progressively discretized.

3) Guide for Tuples:
- Perform bunching to discover the time period included.
- Get affiliation rules via looking for gatherings/gatherings of groups that happen together.

b) Web usage Mining:
Web usage mining is used to derive useful data information, knowledge from the weblog data and helps in identifying the users access designs for web pages.
In mining, the management of web resources, the individual is thinking about the data or resources of visitors of a website that are composed as web server logs while the content and mechanism of the set of web pages follow the intentions of the author of the pages, the single requests shows how the users view these pages. Web usage and mining can disclose relationships that were not suggested by the designer of the webpage.
In mining, the management of web resources, the individual. A web server generally registers a weblog entry or weblog entry for each access of a web page, it contains the URL requested, the IP address from which the request introduced and a timestamp.
For web based e-commerce servers, a large number of web access log data are being collected they are famous websites can register weblog records in order of thousands of megabytes each day. Weblog databases supports rich data about web dynamics. Therefore it is essential to produce sophisticated weblog mining approaches.

In developing methods for web usage mining, it can consider the following. First, although it is encouraging and simulating to conciere the several applications of weblog

file analysis. It is essential to understand that the success of such applications based on how much true and reliable knowledge can be found from the large raw log records.

Second, with the available URL time, IP address and webpage content data, a multidimensional view can be built on the weblog be implemented to discover the top N users, top N accessed web servers etc.

Third data mining can be implemented on weblog records to discover association patterns, sequential patterns and trends of web accessing.

Advantages:

1. User Behaviour Insight: web usage mining helps understand how users interact with websites, allowing buisnesses to tailor their content according to preferences.

2. Personalisation: Enables personalized reccomendations and content suggestions based on user history.

3. Fraud Detection: Used for detecting frauds and identifying unsual patters, such as suspicious login attempts.

Disadvantages:

1. Privacy concerns: Collecting and analyzing user data can raise privacy concerns.
2. Data Quality: The quality of data can vary, leading to inaccuracies in analysis and insights.