

Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

© CERTIFICATE ©

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


Teacher In-Charge

Head of the Department

Date 23/10/2023

Principal

CONTENTS

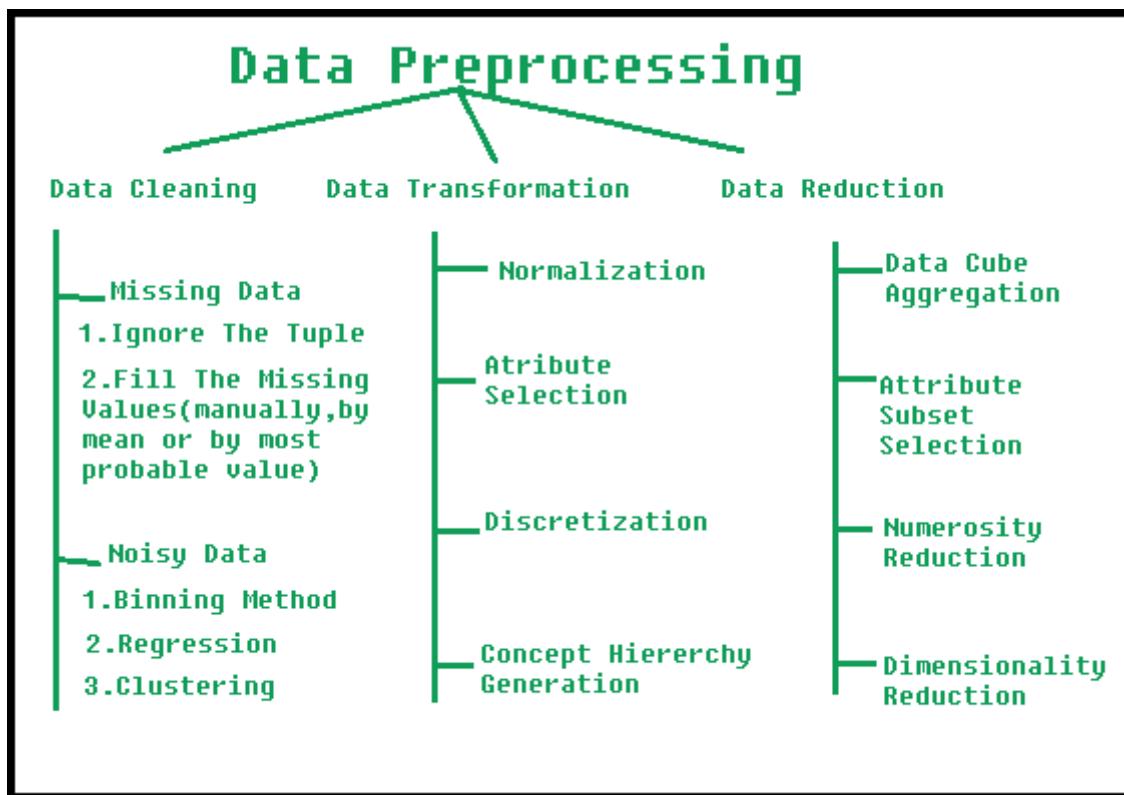
SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1)	Select a dataset and perform explanatory data analysis using python.	1-6	21/8/23	
2)	One case study on building data warehouse / data marts	7-16	31/6/23	
3)	Implementation of all dimension table and fact table based on experiment 2 case study	17-43	10/8/23	
4)	Implementation of OLAP operational Slice, dice, rollup drilldown, and pivot.	44-46	24/8/23	
5)	Implementation of Bayesian theorem	47-51	31/8/23	
6)	Perform data preprocessing and perform classification, clustering and association rules	52-57	5/9/23	(1)
7)	Implementation of clustering algorithm	58-62	14/9/23	
8)	Implementation of any one hierarchical clustering	63-68	21/9/23	
9)	Implementation of association rule mining algorithm	69-72	19/10/23	
10)	Implementation of Page Rank/Hits Algorithm.	73-76	19/10/23	
11)	Assignment 1	77-78	21/10/23	
12)	Assignment 2	79-85	21/10/23	

EXPERIMENT - 01

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

Theory:

Data Preprocessing:



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

- Data cleaning: This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.
- Data Integration: This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.

- Data Transformation: This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.
- Data Reduction: This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

Data Transformation:

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

- Smoothing: It is a process that is used to remove noise from the dataset using some algorithms. It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form. The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn't see otherwise.
- Aggregation: Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used. Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results. The collection of data is useful for everything from decisions concerning financing or business

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

- Normalization: Data normalization involves converting all data variables into a given range.
- Generalization: It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example, Age initially in Numerical form (22, 25) is converted into categorical value (young, old). For example, Categorical attributes, such as house addresses, may be generalized to higher-level definitions, such as town or country.
- Attribute Construction: Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

Data Discretization:

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

- Histogram analysis: Histogram refers to a plot used to represent the underlying frequency distribution of a continuous data set. Histogram assists the data inspection for data distribution. For example, Outliers, skewness representation, normal distribution representation, etc.
- Binning: Binning refers to a data smoothing technique that helps to group a huge number of continuous values into smaller values. For data discretization and the development of idea hierarchy, this technique can also be used.

Data Visualization:

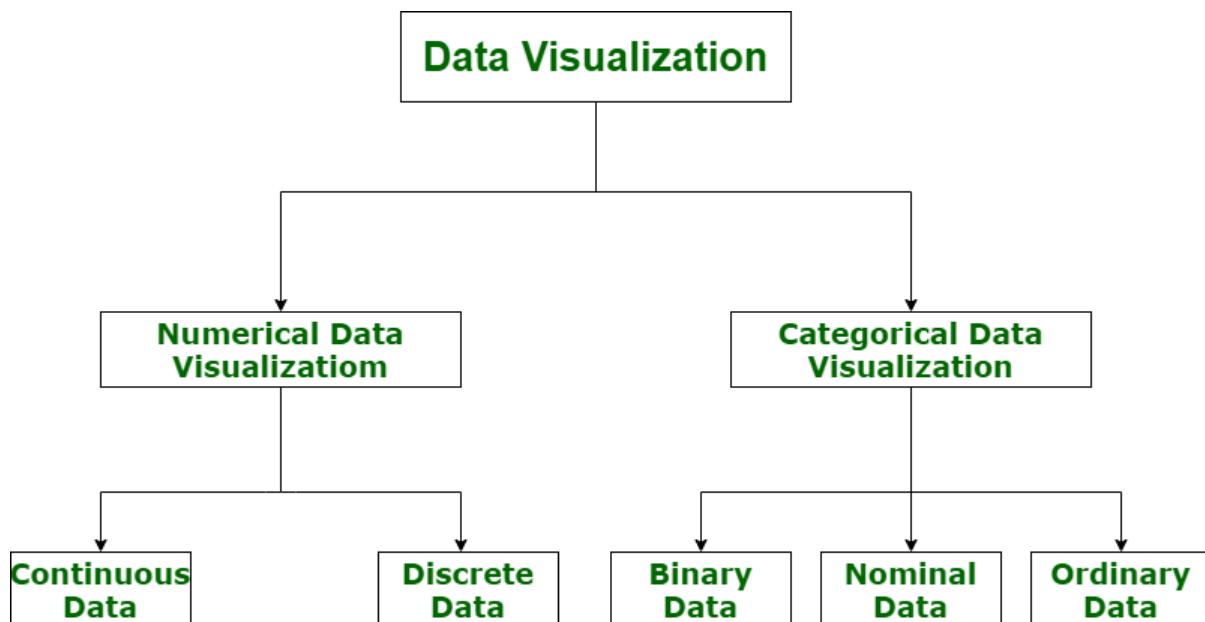
Data visualization is actually a set of data points and information that are represented graphically to make it easy and quick for user to understand. Data visualization is good if it has a clear meaning, purpose, and is very easy to interpret, without requiring context. Tools of data visualization provide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

Categories of Data Visualization :

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

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

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



- Categorical Data : Categorical data is also known as Qualitative data. Categorical data is any data where data generally represents groups. It simply consists of categorical variables that are used to represent characteristics such as a person's ranking, a person's gender, etc. Categorical data visualization is all about depicting key themes, establishing connections, and lending context. Categorical data is classified into three categories :
1. Binary Data- In this, classification is based on positioning (Example: Agrees or Disagrees).
 2. Nominal Data- In this, classification is based on attributes (Example: Male or Female).
 3. Ordinal Data- In this, classification is based on ordering of information (Example: Timeline or processes).

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

Code:

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

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

1000 fake = Faker()

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

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

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

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

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

contact_ids = set()

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

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

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

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

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

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

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

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

Output:

Original Dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	
	transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	email
1	P2859	Titan T27	watches	David LLC	W7276	Cain, Christensen and Myers	PSIC 4244, Box 5134	C7597	53535	Stewart, Clayton and Martinez	C4833	001-954-497-8463	xbarrett@example.com	
2	P5937	Samsung QLED 21	tv	Reynolds and Sons	W7001	Haney and Sons	APO AP 43657	C7597	53535	Stewart, Clayton and Martinez	C4834	001-443-1807x1981	vherandez@e...	
3	P1553	Philips Iron 14	Electrical App	Greene-Hernandez	W7296	Lucas	4602 Julia Parks	Bethel, Pennsylvania, GU 8889	C7099	57973	Carmel, Brady and Hancock	C4835	(645)495-0326x7124	cospentav@e...
4	P7296	Nike New	Wear	Weeks Group	W8451	Norris LLC	42691 Thomas Avenue	Anthonyview, PR 81410	C6096	52142	Singh-Johnson	C4836	(989)459-1195x547	monica66@e...
5	P9813	US Polo Black Denim	Wear	Doyal Ltd	W8571	Brewer-McDonald	902-200-1100, Apt. 051	North Manchester, NH 03256	C8293	53115	Boyer Ltd	C3118	6267958495	kmcrae@e...
6	P5206	Samsung Washing Machine	Electrical App	Sexton Ltd	W8468	Schaefer-Thomas	271 Foster Street, Suite 850	Jessiewood, MD 20884	C5676	54540	Sandoval, Brown and Peters	C7667	999-354-0180x1206	rosanabeben@e...
7	P1489	Reliance Hair Dryer	Electrical App	Ferguson, Thomas and Bryant	W7100	Martinez-Robis	28282 Barbera Trail	Chesapeake, VA 23322	C5055	52603	Madden Inc	C1342	001-918-33662x9815	leegjenifer@e...
8	P8935	LG Airon P32	tv	Reed, Howard and Huff	W8405	Willis-Shaw	295 Pollard Stream Apt. 437	Lake Sabineview, CO 37099	C5974	53568	Pearson, Wall and Shelton	C3989	904-831-7832x5667	breydens@e...
9	P9413	Sony Bravia 57"	tv	Palmer, Hanna and Smith	W1934	Fields-Ellis	Unit 787A Box 4902	DRG, PO Box 2929	C6639	58035	Ellis Ltd	C1699	409-236-0115	thompsonzach...
10	P3825	Fossil F31	watches	Johnson, Todd and Todd	W2006	Ramirez, Cooper and Goodman	6482 Megan Underpass	West Houston, TX 77058	C2366	53141	Johnson Group	C5700	619315694	robertt@e...
11	P7873	program	occur	Quinn, Baxter and Gregory	W5878	Gilmores	271 Foster Street, Suite 2001	Easton, Maryland, MD 20801	C4489	58420	Mccoy LLC	C9238	+1-492-300-4668x5728	hoffmannbernd@e...
12	P1596	next	challenge	Brown Inc	W1169	Jones Inc	00674 Burnett Circles	South Portland, ME 04105	C5163	51563	Smith Ltd	C7601	+1-345-763-370xx3369	sdavis@e...
13	P2893	visit	stop	Coleman LLC	W6639	Powell LLC	551 Main Creek Apt. 437	Ulsterberg, VA 52257	C0009	58841	Small PLC	C5895	+1-389-224-5753x751	krinna32@e...
14	P4110	always	next	Cook Ltd	W4150	Weaver PLC	78349 Prat Port	Port Mattheus, ID 84042	C5170	53565	Carney, Edwards and Chen	C5689	089881-9017	natalie47@e...
15	P1235	data	right	Johnson, Ross and Phillips	W5472	Conley-Flores	002 Miller Mount	Port Mattheut, ID 84042	C5118	59366	Rutz-Garcia	C1527	8288290416	longasthey@e...
16	P2301	education	imp	Detraczuk, Lane and Jacobson	W8373	Webb-Costa	901 Clarke Radios	Port Mattheut, ID 84042	C5634	53534	Potter and Sons	C5607	550487-8208x82519	coreywilson@e...
17	P5882	memory	however	King and Sons	W3067	Garcia-Moore	995 354 0180x1206	Port Mattheut, ID 84042	C3860	57978	Rowe-Benitez	C9300	(948)541-2134	objyan@e...
18	P1762	in	Mr	Copeland-Evans	W1796	Gonzalez Ltd	South Toddton, MD 49094	C4720	53294	Williams, Ware and Martin	C8900	866-525-1911	bukesharon@e...	
19	P8087	collector	certain	Lawrence PLC	W1563	Gillespie and Sons	2241 The Mission Suite 1	Robbinsmouth, IA 50405	C5142	59796	Newton Group	C8997	952-647-3640x389	dakota05@e...
20	P4582	surface	number	Fowler, Todd and Williams	W1452	Ritter-Bishop	3849 Smith Mills, Suite 709	Port Mattheut, ID 84042	C4202	57329	Walker-Espinoza	C9308	001-981-307-371x81268	shawboyle@e...
21	P2928	research	rock	Brennan, Johnson and O'Donnell	W6440	Greer and Sons	210 Adam Station Suite 972	Timothyton, WV 06003	C4485	53289	Taylor Ltd	C7302	331-531-0151x1354	adrian23@e...
22	P2403	executive	behavior	Hernandez, McGee and Ray	W1576	Garcia-Evans and Ramirez	3rd Watch	Christopher, WV 27118	C6781	53005	Hernandez, Williams and Martinez	C8344	792-886-6325x458	mmyays@e...
23	P5866	sense	light	Johnson-Sosa	W7268	Lucas, White and Smith	7715 Greene Haven Suite 09	Anderson, Indiana, IN 46955	C7742	55273	Lee PLC	C7742	+1-562-508-2382	wrightbrun@e...
24	P1513	clear	dark	Rivers-Bowers	W2069	James-Carter	56471 Waller Drive	Jacobchew, NH 03863	C5620	53100	Baley LLC	C7185	+1-882-893-817x1943	laure55@e...
25	P2845	boy	candidate	Nelson-Cantillo	W7937	Wilson, Smith and Vergas	79349	Port Mattheut, ID 84042	C5621	53100	Baley LLC	C7185	+1-882-893-817x1943	laure55@e...

After data preprocessed:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	transaction_id	product_id	product_name	product_category	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	email	arrival_date	dispatch_date	data_number_of_boxes		
1	P2859	Titan T27	watches	Team	W7276	Bombay-X	Mumbai	C7597	53535	Stewart, Clayton and Martinez	C4833	001-954-497-8463	xbarrett@example.com	2021-11-29	2023-01-15	36		
2	P5937	Samsung QLED 21	tv	Sensing	W7001	DenhCaps	Delhi	C7099	57973	Brady and Hancock	C4834	001-443-1807x1981	vherandez@e...	2021-09-18	2022-05-26	77		
3	P1553	Philips Iron 14	Electrical App	Greene-Hernandez	W7296	Bangalore	C2923	53115	Boyer Ltd	C5118	(989)459-1156x547	monica66@e...	2021-06-21	2023-10-13	71			
4	P7296	Nike New	Wear	Weeks Group	W8451	KeralaS	Kerala	C6336	59922	Devils	C3860	999-354-0180x1206	robertt@e...	2022-10-29	2023-11-07	37		
5	P9813	US Polo Black Denim	Wear	Doyal Ltd	W8571	KereSeems	Kerala	C6336	59922	Devils	C5620	999-354-0180x1206	robertt@e...	2022-10-29	2023-11-17	73		
6	P5206	Samsung Washing Machine	Electrical App	Sexton Ltd	W8468	GurugramIDES	Gurugram	C5055	52603	Madden Inc	C1342	001-918-33662x9815	leegjenifer@e...	2022-10-18	2023-05-16	5		
7	P1489	Reliance Hair Dryer	Electrical App	Reliance	W7100	HyderabadHub	Hyderabad	C5055	52603	Madden Inc	C7740	5331-531-0151x1354	lucaskaren@e...	2022-08-04	2023-05-23	53		
8	P8935	LG 42inch P32	tv	Reed, Howard and Huff	W8405	Willis-Shaw	Hyderabad	C5055	52603	Madden Inc	C5667	5998-8208x82519	lucaskaren@e...	2022-08-10	2023-01-29	41		
9	P3825	Fossil F31	watches	Fossil	W2006	Bombay-Z	Mumbai	C2366	53141	Johnson Group	C5700	61315150049	robertt@e...	2022-09-11	2023-01-30	56		
10	P1410	Sony Bravia 57"	tv	Johnson, Todd and Todd	W2006	Bombay-Z	Mumbai	C2366	53141	Johnson Group	C4489	001-981-307-371x81268	shawnboyle@e...	2022-09-06	2023-05-08	13		
11	P7873	Gardien G21	watch	Gardien	W6639	GurugramIDES	Gurugram	C6003	56841	Small PLC	C5667	912-337-3662x6155	lucaskaren@e...	2022-09-11	2023-05-23	25		
12	P2403	DMX Hoodie	Wear	DNMX	W8773	DehraCaps	Dehra	C2088	53034	Potter and Sons	C5667	0502-487-8208x82519	coreywilson@e...	2020-04-03	2023-06-17	74		
13	P1235	data	right	Johnson, Ross and Phillips	W5472	Conley-Flores	Bengaluru	C2088	53034	Potter and Sons	C5667	0461-541-2134	coreywilson@e...	2020-10-29	2023-11-27	86		
14	P1762	in	Mr	Detraczuk, Lane and Jacobson	W1796	Gonzalez Ltd	Bengaluru	C4485	53289	Rowe-Benitez	C2870	999-354-0180x1206	lucaskaren@e...	2022-03-26	2023-06-24	18		
15	P5882	memory	however	King and Sons	W3067	Garcia-Moore	Bengaluru	C4485	53289	Rowe-Benitez	C2870	999-354-0180x1206	lucaskaren@e...	2023-01-23	2023-06-28	45		
16	P1596	next	stop	Coleman LLC	W6639	GurugramIDES	Gurugram	C3860	57978	Rowe-Benitez	C2870	999-354-0180x1206	lucaskaren@e...	2023-02-11	2023-07-04	85		
17	P2893	visit	stop	Coleman LLC	W7937	Willis-Shaw	Gurgaon	C5055	52603	Rowe-Benitez	C7742	3431-531-0151x1354	lucaskaren@e...	2022-08-20	2023-10-17	1		
18	P1513	program	occur	Quinn, Baxter and Gregory	W1169	Wills	Gurgaon	C5055	52603	Rowe-Benitez	C7742	3431-531-0151x1354	lucaskaren@e...	2022-08-20	2023-10-23	23		
19	P2403	education	imp	Detraczuk, Lane and Jacobson	W6639	GurugramIDES	Gurgaon	C5055	52603	Rowe-Benitez	C7742	3431-531-0151x1354	lucaskaren@e...	2022-09-05	2023-10-25	91		
20	P1410	memory	however	King and Sons	W1452	Ritter-Bishop	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2020-09-10	2023-05-26	32		
21	P7232	surface	number	Fowler, Todd and Williams	W1419	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2020-04-06	2023-06-17	74		
22	P2928	research	rock	Brennan, Johnson and O'Donnell	W1419	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2020-10-29	2023-11-27	86		
23	P1235	data	right	Johnson, Ross and Phillips	W6440	Conley-Flores	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-03-26	2023-06-24	18		
24	P1762	in	Mr	Detraczuk, Lane and Jacobson	W1796	Gonzalez Ltd	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2023-01-23	2023-06-28	45		
25	P5882	memory	however	King and Sons	W3067	Garcia-Moore	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-08-20	2023-10-17	1		
26	P1596	next	stop	Coleman LLC	W6639	GurugramIDES	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-09-05	2023-10-25	91		
27	P2893	visit	stop	Coleman LLC	W7937	Willis-Shaw	Gurgaon	C5055	52603	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-04-06	2023-06-17	74		
28	P1410	program	occur	Quinn, Baxter and Gregory	W1575	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-08-20	2023-10-23	23		
29	P7232	education	imp	Detraczuk, Lane and Jacobson	W1575	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-09-05	2023-10-25	91		
30	P1427	memory	however	King and Sons	W1575	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-08-20	2023-10-17	74		
31	P2403	surface	number	Fowler, Todd and Williams	W1575	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-09-06	2023-10-26	91		
32	P1474	research	rock	Brennan, Johnson and O'Donnell	W1575	Wills	Gurgaon	C5170	53565	Rutherford, Jones and Williams	C7831	001-608-336-9100x3629	mccoyle@e...	2022-08-20	2023-10-17	74		
33	P27																	

IMPLEMENTATION

DWM_Expt1_Remote.ipynb

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Loading the libraries and Download the dataset from Kaggle or other sources
[ ] import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

Read the file and select appropriate file read function according to the datatype of file
[4] df= pd.read_csv("/content/inventory.csv")

Describe the attribute names
[5] df.columns
Index(['transaction_id', 'product_id', 'product_name', 'product_category',
       'product_brand', 'warehouse_id', 'warehouse_name', 'warehouse_location',
       'city_id', 'supplier_id', 'supplier_name', 'contact_id', 'phone',
       'email', 'arrival_date', 'dispatch_date', 'number_of_boxes'],
      dtype='object')

[6] df.sample(10)

```

transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	
128	129	P8248	Fossil F31	watches	Fossil	W9155	DelhiCaps	Delhi	C7099	S7412	Whitehead, Williams and Lewis	(732)256-6491x1677	william72
114	115	P3609	DNMX Hoodie	Wear	DNMX	W2275	DelhiCaps	Delhi	C7099	S4025	Campbell, Mann and Barajas	(704)331-7361	zachary4l
60	61	P5814	Sony 55inch P19	tv	Sony	W8001	KeralaSweP	Kerala	C6336	S5648	Lucas-Pratt	(650)589-3457	catherine3G

0s completed at 15:57

DWM_Expt1_Remote.ipynb

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[6] df.sample(10)

```

transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	
128	129	P8248	Fossil F31	watches	Fossil	W9155	DelhiCaps	Delhi	C7099	S7412	Whitehead, Williams and Lewis	(732)256-6491x1677	william72
114	115	P3609	DNMX Hoodie	Wear	DNMX	W2275	DelhiCaps	Delhi	C7099	S4025	Campbell, Mann and Barajas	(704)331-7361	zachary4l
60	61	P5814	Sony 55inch P19	tv	Sony	W8001	KeralaSweP	Kerala	C6336	S5648	Lucas-Pratt	(650)589-3457	catherine3G
125	126	P1313	Hairer Hair Dryer T2	Electrical App	Haire	W4370	GurgaonHDES	Gurgaon	C5676	S2162	Dean-Figueroa	(660)98-3283	karen99k
160	161	P1032	Jack & Jones Denim Pants	wear	Jack & Jones	W2573	GurgaonHDES	Gurgaon	C5676	S8547	Perez-Martin	(281)369-3791	scottrogers
14	15	P1235	DNMX Shift S21	Wear	DNMX	W5472	BombayX	Mumbai	C7597	S9366	Raul-Garcia	(223)288-90416	longashley
17	18	P7052	Sonata Khd K2	watches	Sonata	W1795	BangaloreS	Bangalore	C5293	S6126	Wilkinson, Ware and Martin	(866)525.1911	barkesharor
195	196	P5145	LG hair dryer	Electrical App	LG	W9621	GurgaonHDES	Gurgaon	C5676	S3845	Dixon and Sons	(001)946-315-1639	daniellewashington
59	60	P4621	Rolex R-52	watches	Rolex	W7082	BangaloreS	Bangalore	C5293	S8931	Sandow Group	(522)286-7531x1701	gina01
132	133	P4124	Titan W2	watches	Titan	W9569	GurgaonHDES	Gurgaon	C5676	S9627	Booth PLC	(001)946-315-1639	brendaberry

[7] df.head()

transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	
0	1	P2859	Titan T27	watches	Titan	W7276	BombayX	Mumbai	C7597	S3535	Stewart, Clayton and Martinez	(248)341-9463	xbarrett@
1	2	P5937	Samsung QLED 21	tv	Samsung	W7001	DelhiCaps	Delhi	C7099	S7973	Carroll, Brady and Hancock	(845)485-0326x71214	vhemandez@
2	3	P1553	Philips Iron 14	Electrical App	Philips	W1296	BombayZ	Mumbai	C7597	S2142	Singh-Johnson	(650)589-1956x547	monica66j
3	4	P7296	Nike Nax	Wear	Nike	W8451	BangaloreS	Bangalore	C5293	S3115	Boyer Ltd	(989)559-1956x547	monica66j
4	5	P9813	US Polo Black Denim	Wear	US Polo	W8571	KeralaSweP	Kerala	C6336	S8922	Davis Ltd	(626)7956495	krajeff

0s completed at 15:57

DWM_Expt1_Remote.ipynb

```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[7] df.head()

```

transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone	
0	1	P2859	Titan T27	watches	Titan	W7276	BombayX	Mumbai	C7597	S3535	Stewart, Clayton and Martinez	(248)341-9463	xbarrett@
1	2	P5937	Samsung QLED 21	tv	Samsung	W7001	DelhiCaps	Delhi	C7099	S7973	Carroll, Brady and Hancock	(845)485-0326x71214	vhemandez@
2	3	P1553	Philips Iron 14	Electrical App	Philips	W1296	BombayZ	Mumbai	C7597	S2142	Singh-Johnson	(650)589-1956x547	monica66j
3	4	P7296	Nike Nax	Wear	Nike	W8451	BangaloreS	Bangalore	C5293	S3115	Boyer Ltd	(989)559-1956x547	monica66j
4	5	P9813	US Polo Black Denim	Wear	US Polo	W8571	KeralaSweP	Kerala	C6336	S8922	Davis Ltd	(626)7956495	krajeff

Count number of Values

```

[8] unique_value_counts = df.unique()
print(unique_value_counts)

```

	transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id	phone
transaction_id	200	200	84	5	32	200	7	6	14	5	10	177	200
product_id	200	200	84	5	32	200	7	6	14	5	10	177	200
product_name	200	200	84	5	32	200	7	6	14	5	10	177	200
product_category	200	200	84	5	32	200	7	6	14	5	10	177	200
product_brand	200	200	84	5	32	200	7	6	14	5	10	177	200
warehouse_id	200	200	84	5	32	200	7	6	14	5	10	177	200
warehouse_name	200	200	84	5	32	200	7	6	14	5	10	177	200
warehouse_location	200	200	84	5	32	200	7	6	14	5	10	177	200
city_id	200	200	84	5	32	200	7	6	14	5	10	177	200
supplier_id	200	200	84	5	32	200	7	6	14	5	10	177	200
supplier_name	200	200	84	5	32	200	7	6	14	5	10	177	200
contact_id	200	200	84	5	32	200	7	6	14	5	10	177	200
phone	200	200	84	5	32	200	7	6	14	5	10	177	200
email	200	200	84	5	32	200	7	6	14	5	10	177	200

0s completed at 15:57

C32_Om_2103163_DWM

The image displays three vertically stacked screenshots of a Jupyter Notebook interface, titled "DWM_Exp1_Remote.ipynb". Each screenshot shows a code editor with several cells of Python code and their corresponding outputs.

Screenshot 1:

```

Count number of Values
[1] unique_value_counts = df.unique()
print(unique_value_counts)

transaction_id    200
product_id       200
product_name      44
product_category     5
product_brand      32
warehouse_id      200
warehouse_name      7
warehouse_location    6
city_id          200
supplier_id      200
supplier_name     196
contact_id        200
phone           277
email            200
arrival_date     185
dispatch_date     169
number_of_boxes     99
dtype: int64

Checking for Null Entries
[2] df.isnull().sum()

transaction_id    0
product_id       0
product_name      0
product_category    0
product_brand      0
warehouse_id      0
warehouse_name      0
warehouse_location    0
city_id          0
supplier_id      0
supplier_name     0
contact_id        0
phone            0
...
```

Screenshot 2:

```

Eliminating rows having value of any column as zero or null
[3] df.dropna(inplace=True)

Counting duplicated entries
[4] df.duplicated().sum()

0

Command to remove duplicate entries
[5] df.drop_duplicates(keep='first', inplace=True)

Retrieving the no of columns and no of rows using df.shape
[6] df.shape

(200, 17)

Finding MIN and MAX for 'number_of_boxes'
[7] maximum_value=df['number_of_boxes'].max()
minimum_value=df['number_of_boxes'].min()
print("Minimum value for Number of Boxes is ",minimum_value)
print("Maximum value for Number of Boxes is ",maximum_value)

Minimum value for Number of Boxes is 1
Maximum value for Number of Boxes is 99

Printing the Datatypes of fields/columns
[8] df.dtypes

```

Screenshot 3:

```

Printing the Datatypes of fields/columns
[9] df.dtypes

transaction_id    int64
product_id       object
product_name      object
product_category   object
product_brand      object
warehouse_id      object
warehouse_name      object
warehouse_location    object
city_id          object
supplier_id      object
supplier_name     object
contact_id        object
phone            object
email             object
arrival_date     object
dispatch_date     object
number_of_boxes     int64
dtype: object

Finding Mean, Median, Mode for 'number_of_boxes'
[10] meanCalc=df['number_of_boxes'].mean()
medianCalc=df['number_of_boxes'].median()
modeCalc=df['number_of_boxes'].mode()
print("Mean = ", meanCalc)
print("Median = ", medianCalc)
print("Mode = ", modeCalc)
print("Name: number_of_boxes, dtype: int64")

Mean = 50.12
Median = 51.5
Mode =
0  54
1  50
2  91
Name: number_of_boxes, dtype: int64

```

C32_Om_2103163_DWM

DWM_Expt1_Remote.ipynb

```
[26] df['number_of_boxes'].describe()
   count    200.000000
   mean     59.129600
   std      28.875153
   min      10.000000
   25%     25.000000
   50%     51.500000
   75%     75.000000
   max     99.000000
Name: number_of_boxes, dtype: float64
```

Box Plot

```
[28] plt.boxplot(df['number_of_boxes'])
plt.show()
```

```
[30] plt.scatter(df['warehouse_location'],df['number_of_boxes'])
plt.xlabel('Warehouse Location')
plt.ylabel('Number of Boxes')
```

```
[46] plt.bar(df['number_of_boxes'],df['product_brand'])
plt.xlabel(0,100)
plt.ylabel(0,100)
plt.show()
```

Bar Graph

```
[46] plt.bar(df['number_of_boxes'],df['product_brand'])
plt.xlabel(0,100)
plt.ylabel(0,100)
plt.show()
```

```
[54] column_names = df.columns
for column_name in column_names:
    column_data = df[column_name]
    if column_data.dtype != <class 'object'>:
```

DWM_Exp1_Remote.ipynb

```

File Edit View Insert Runtime Tools Help
+ Code + Text
Files
sample_data
inventory.csv
Print info for each numeric attribute

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

```

Attribute Name: transaction_id
Count of Values: 200
Minimum Value: 1
Maximum Value: 200
Data Type: int64
Range: 199
Quartiles:
0.25 50.75
0.50 100.50
0.75 150.25
Name: transaction_id, dtype: float64
Percentiles:
0.1 20.9
0.5 100.0
Name: transaction_id, dtype: float64

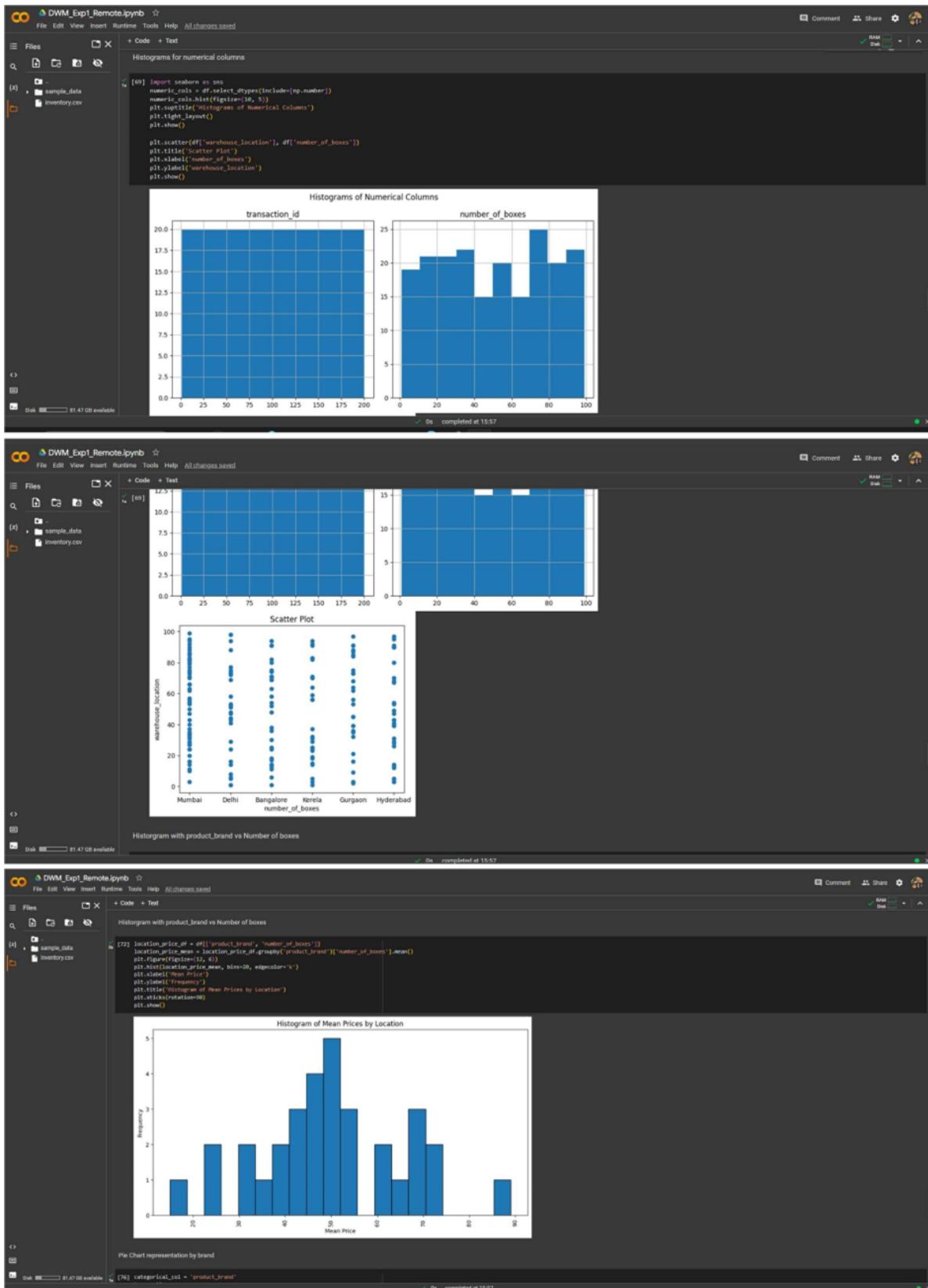
Box Plot for transaction_id

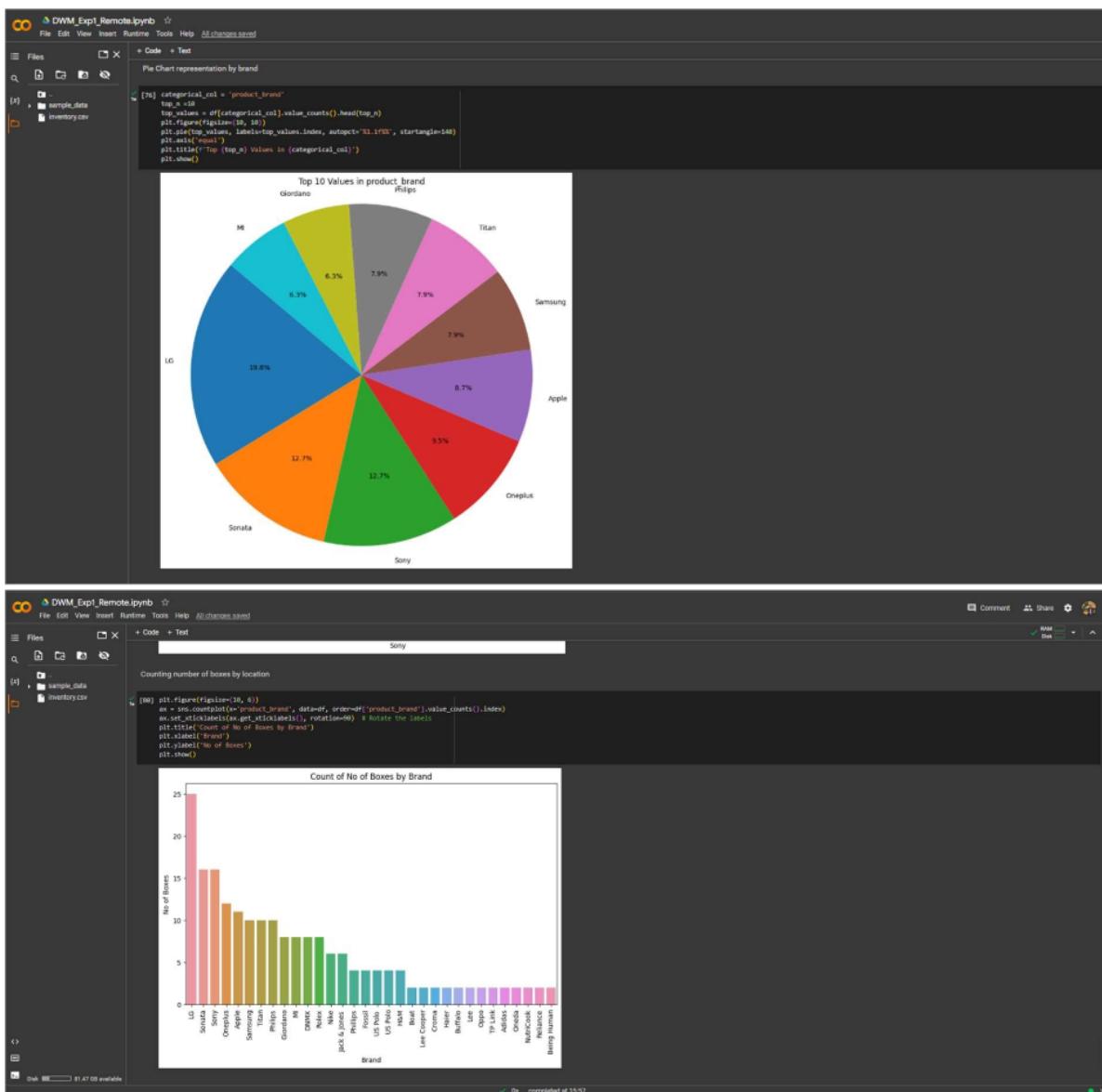
Outliers: Series[], Name: transaction_id, dtype: int64

Attribute Name: number_of_boxes
Count of Values: 200
Minimum Value: 1
Maximum Value: 99
Data Type: int64
Range: 98
Quartiles:
0.25 25.0
0.50 53.5
0.75 74.0
Name: number_of_boxes, dtype: float64
Percentiles:
0.1 11.0
0.9 90.0
Name: number_of_boxes, dtype: float64

Box Plot for number_of_boxes

C32_Om_2103163_DWM





EXPERIMENT - 02

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

Theory:

Difference between the two terms – DBMS & Data Warehouse

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

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

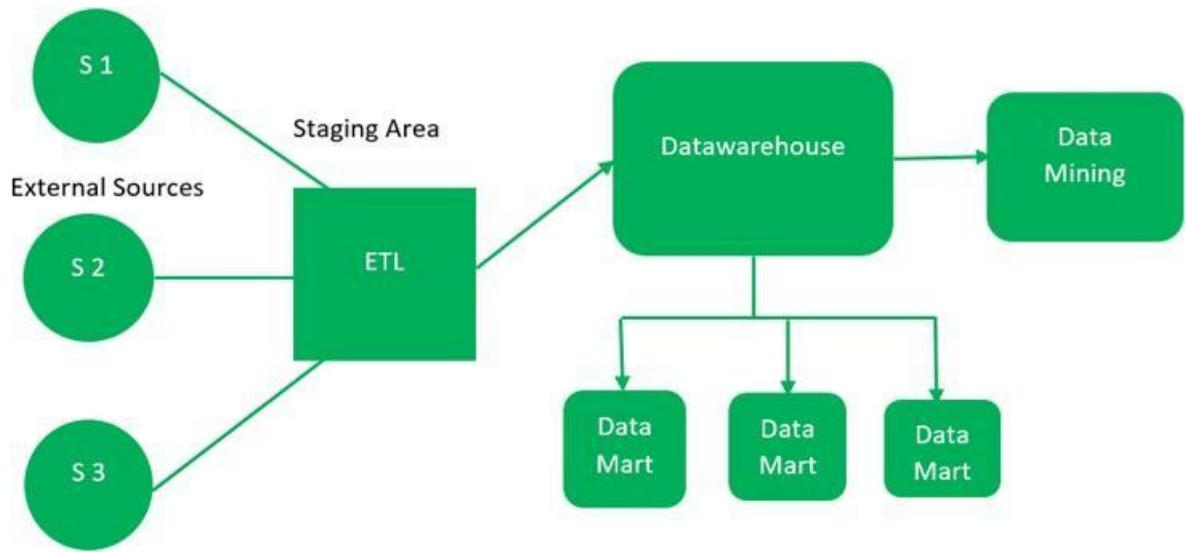
Benefits of Data Warehouse

1. Better business analytics: Data warehouse plays an important role in every business to store and analysis of all the past data and records of the company. which can further increase the understanding or analysis of data to the company.
2. Faster Queries: Data warehouse is designed to handle large queries that's why it runs queries faster than the database.
3. Improved data Quality: In the data warehouse the data you gathered from different sources is being stored and analyzed it does not interfere with or add data by itself so your quality of data is maintained and if you get any issue regarding data quality then the data warehouse team will solve this.
4. Historical Insight: The warehouse stores all your historical data which contains details about the business so that one can analyze it at any time and extract insights from it

Construction of a Data Warehouse

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

A. Top-Down Approach:



The essential components are discussed below:

1. External Sources –

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

2. Stage Area –

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

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

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

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

6. Data-warehouse –

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

7. Data Marts –

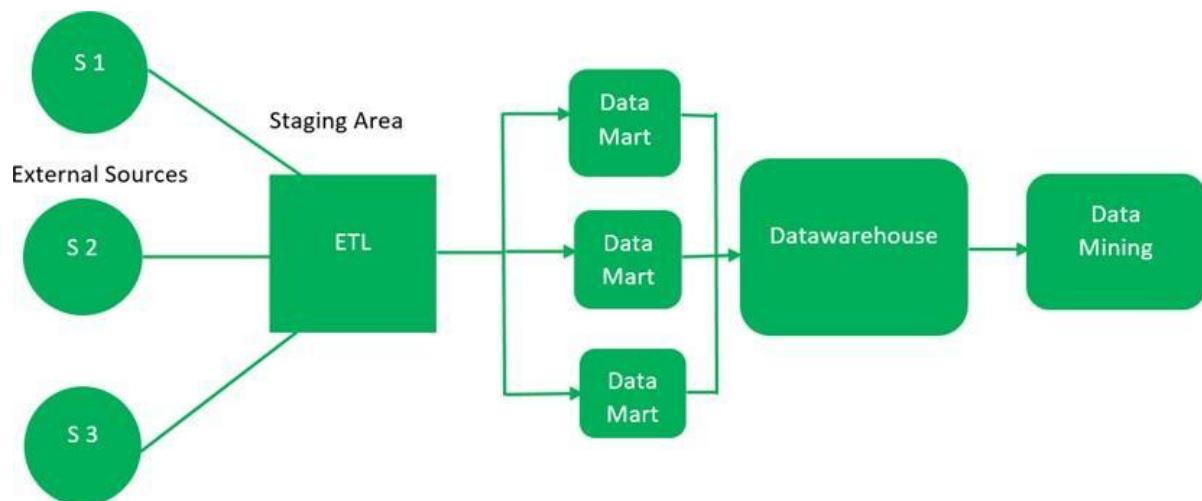
Data mart is also a part of storage component. It stores the information of a particular function of an organisation which is handled by single authority. There can be as many number of data marts in an organisation depending upon the functions. We can also say that data mart contains subset of the data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

8. Data Mining –

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

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

B. Bottom-Up Approach:



1. First, the data is extracted from external sources (same as happens in top-down approach).
2. Then, the data go through the staging area (as explained above) and loaded into data marts instead of data warehouse. The data marts are created first and provide reporting capability. It addresses a single business area.
3. These data marts are then integrated into data warehouse.

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

Fact Table

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

Dimension Table

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

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

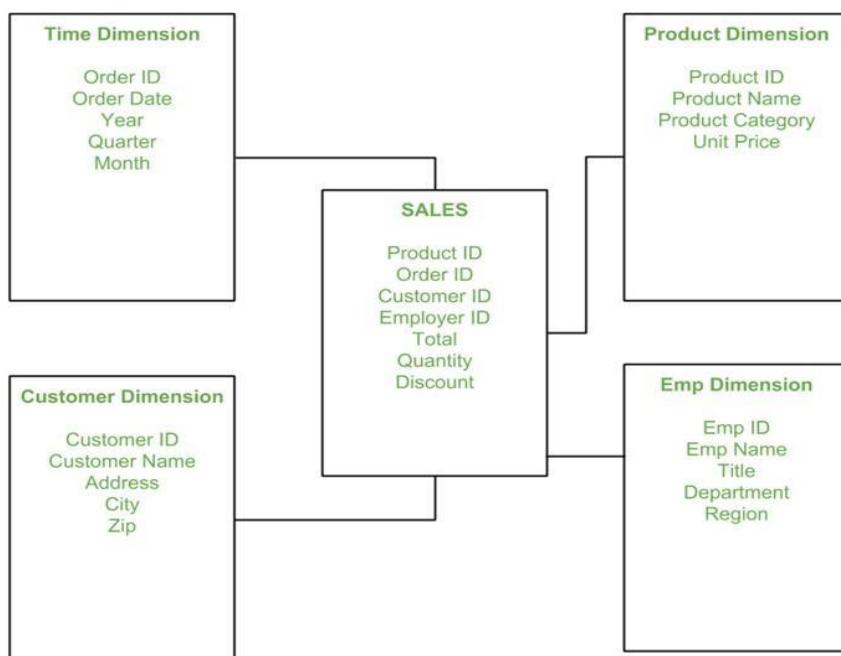
Star Schema

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

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

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

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



Advantages of Star Schema

1. Simpler Queries – Join logic of star schema is quite cinch in comparison to other join logic which are needed to fetch data from a transactional schema that is highly normalized.
2. Simplified Business Reporting Logic – In comparison to a transactional schema that is highly normalized, the star schema makes simpler common business reporting logic, such as of reporting and period-over-period.
3. Feeding Cubes – Star schema is widely used by all OLAP systems to design OLAP cubes efficiently. In fact, major OLAP systems deliver a ROLAP mode of operation which can use a star schema as a source without designing a cube structure.

Disadvantages of Star Schema

1. Data integrity is not enforced well since in a highly de-normalized schema state.
2. Not flexible in terms of analytical needs as a normalized data model.
3. Star schemas don't reinforce many-to-many relationships within business entities – at least not frequently.

Snowflake Schema

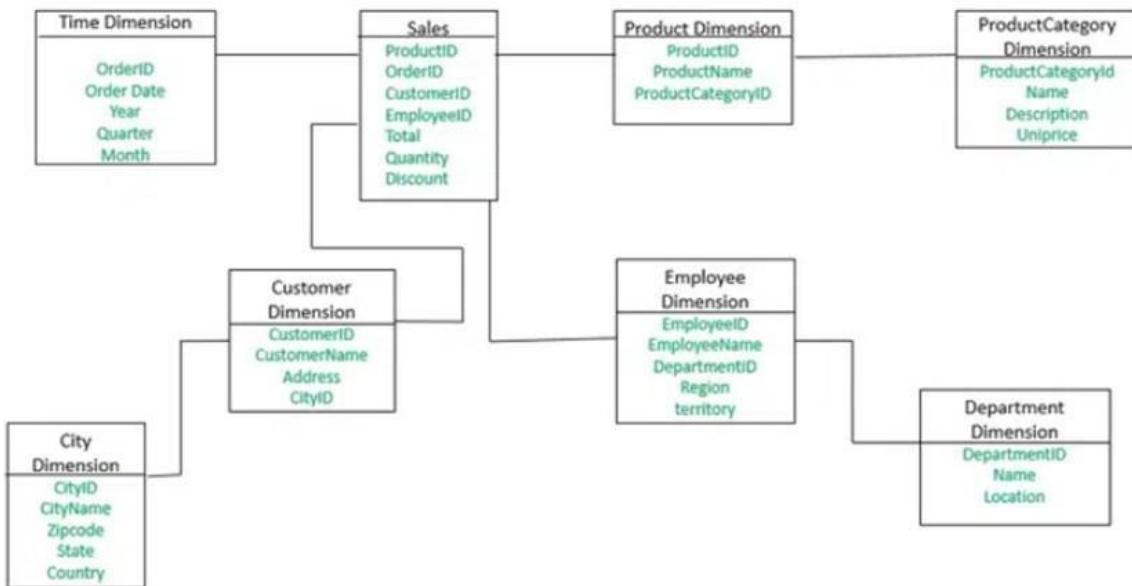
The snowflake schema is a variant of the star schema. Here, the centralized fact table is connected to multiple dimensions. In the snowflake schema, dimensions are present in a normalized form in multiple related tables. The snowflake structure materializes when the dimensions of a star schema are detailed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

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

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

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

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



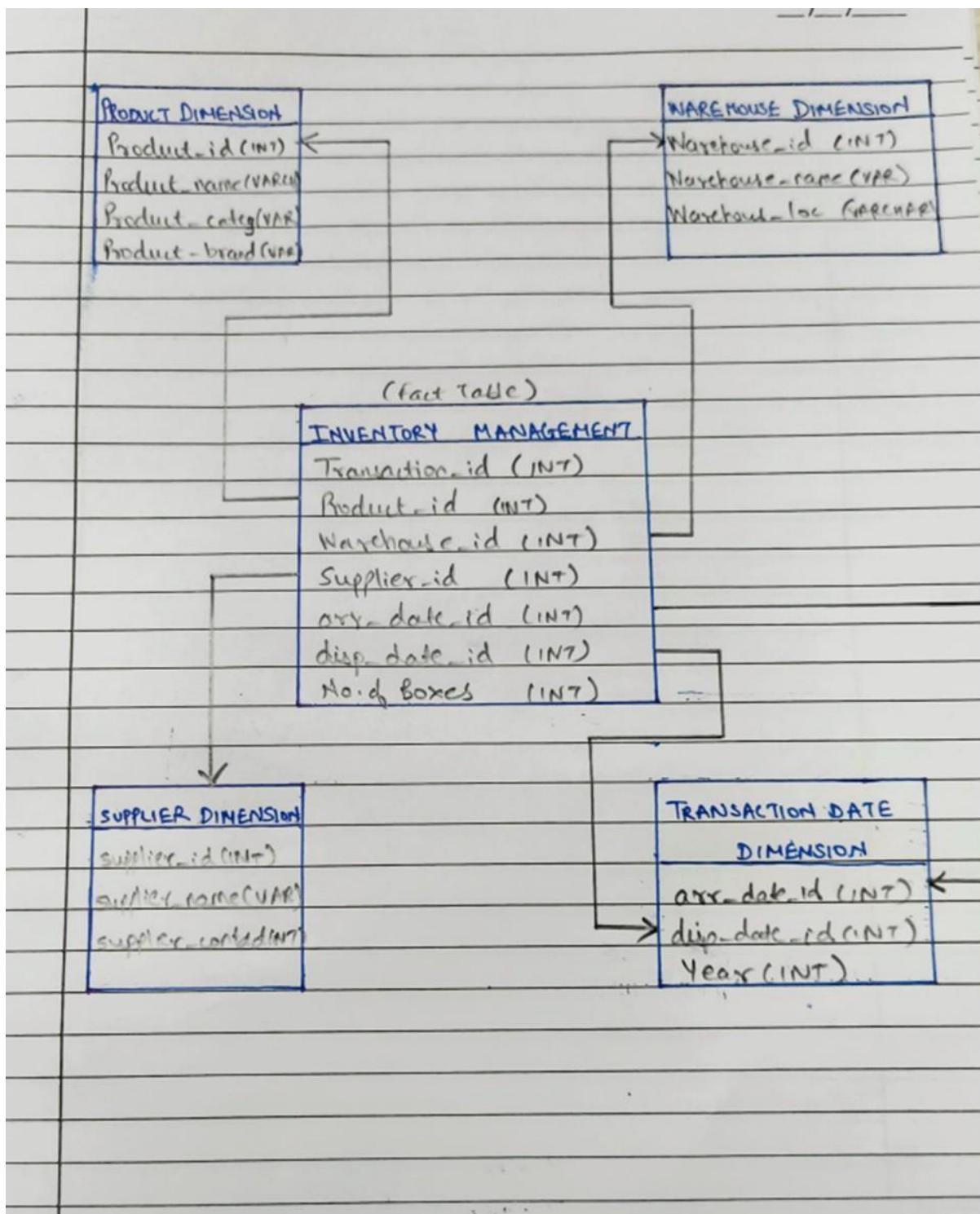
Advantages of Snowflake Schema

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

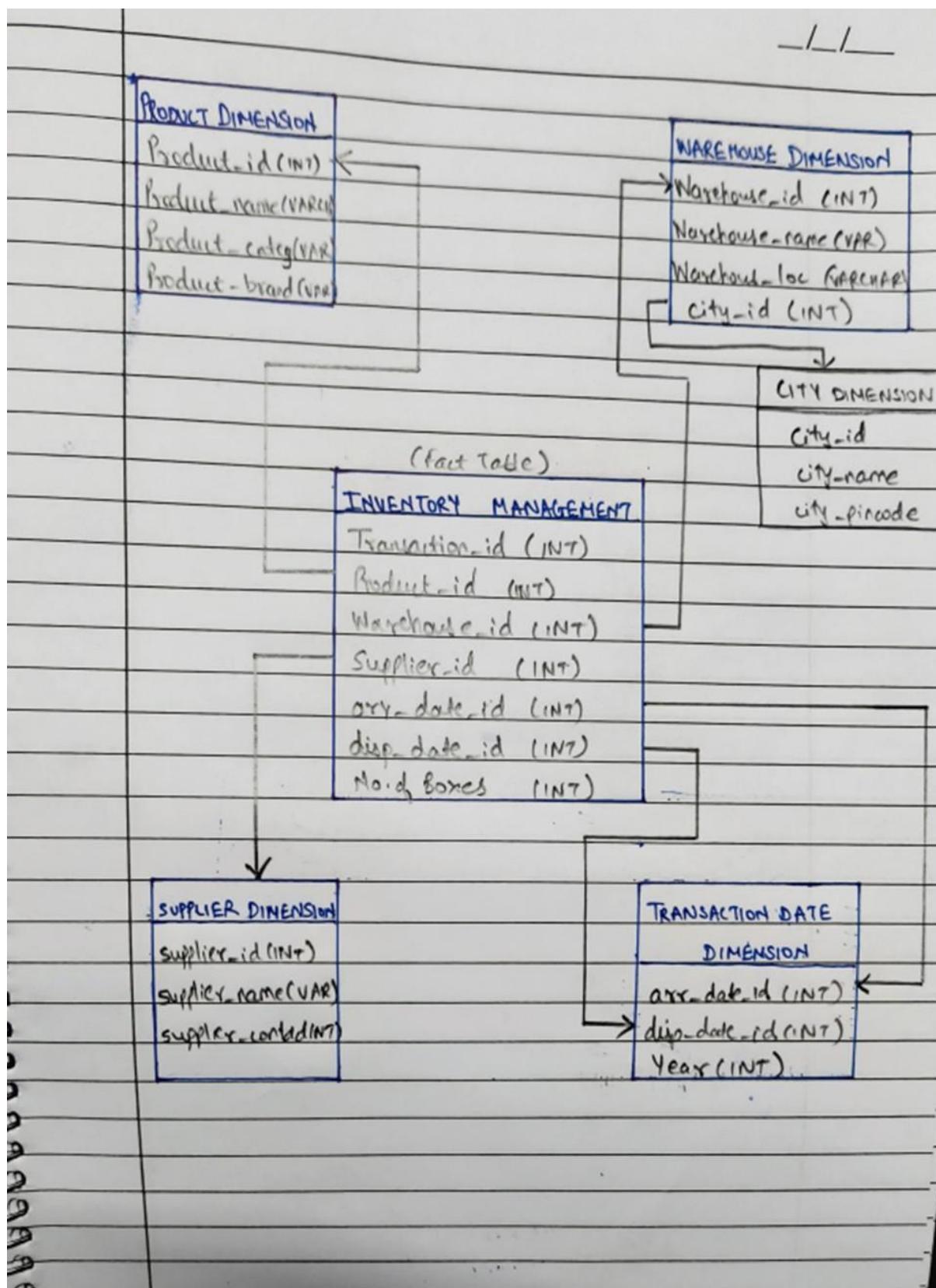
Disadvantages of Snowflake Schema

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

Star Schema



Snowflake Schema



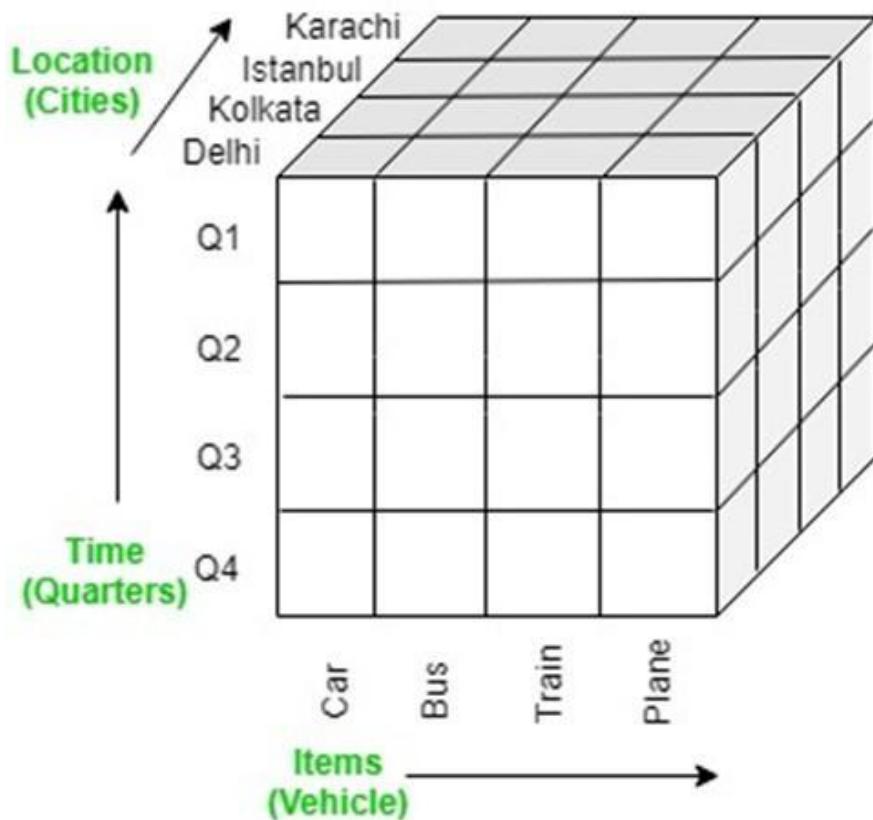
EXPERIMENT - 03

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

Theory:

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

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



OLAP operations:

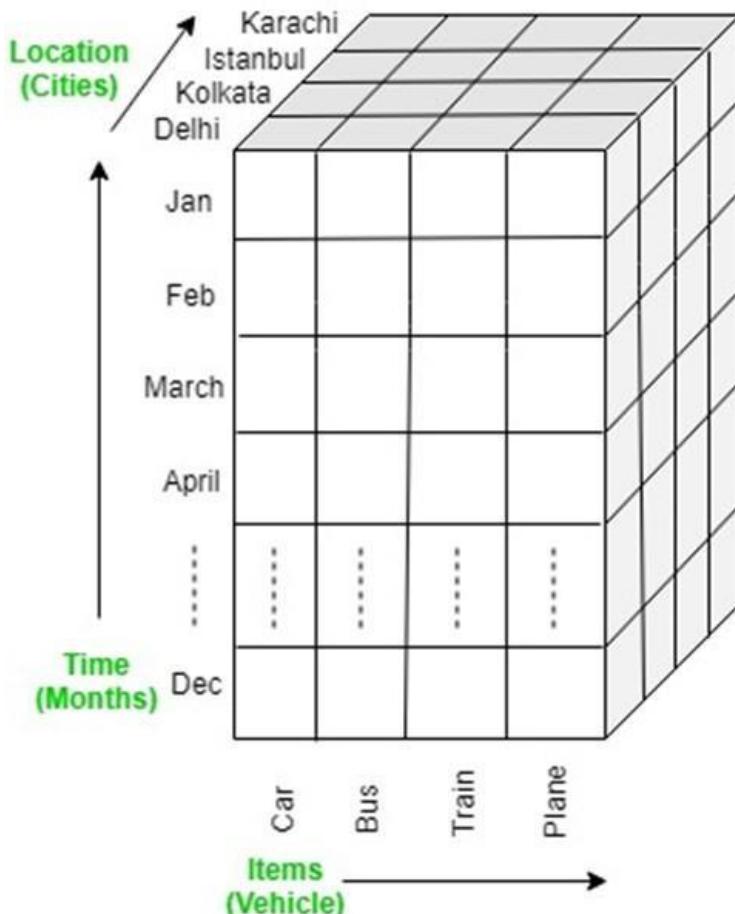
Five basic analytical operations can be performed on an OLAP cube:

1. Drill down:

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

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

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

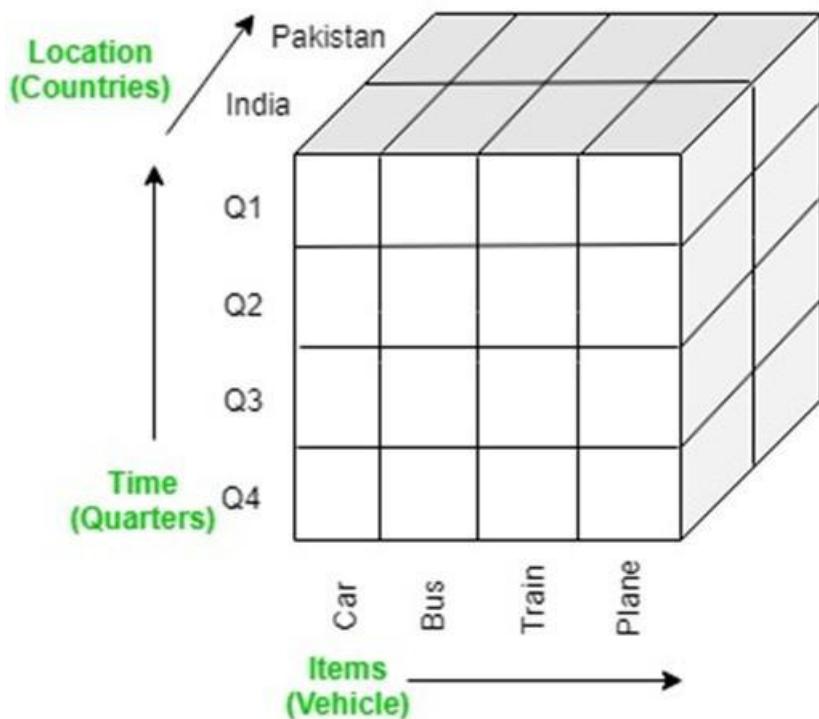


2. Roll up:

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

- Climbing up in the concept hierarchy
- Reducing the dimensions

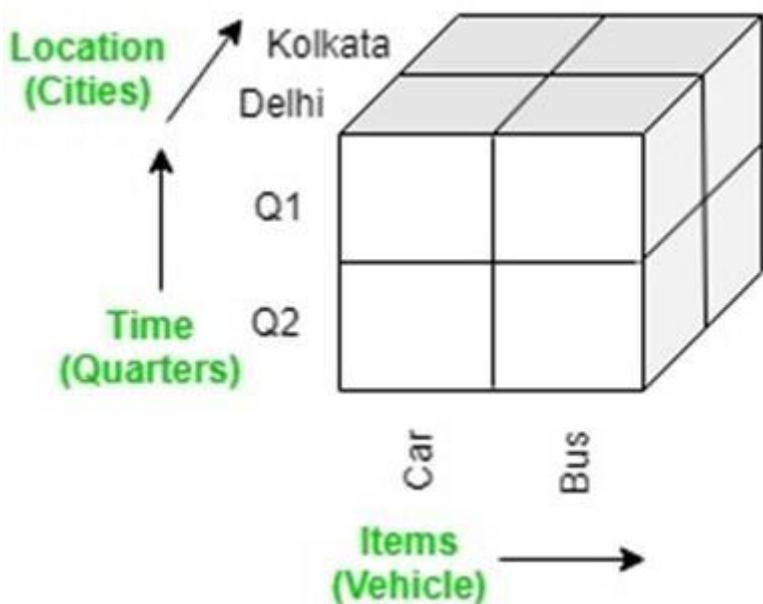
In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of Location dimension (City -> Country).



3. Dice:

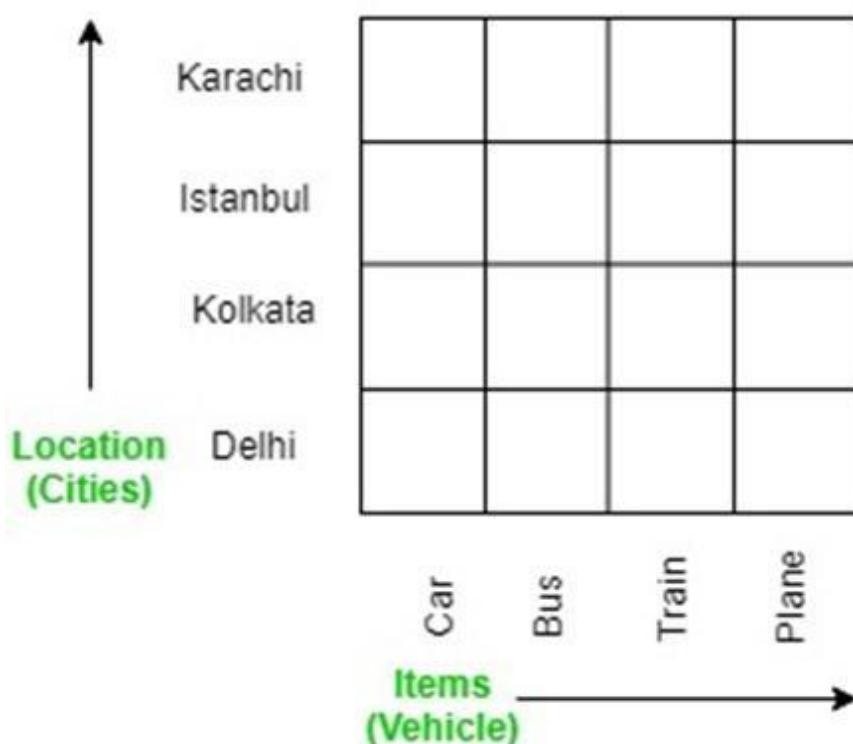
It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting the following dimensions with criteria:

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



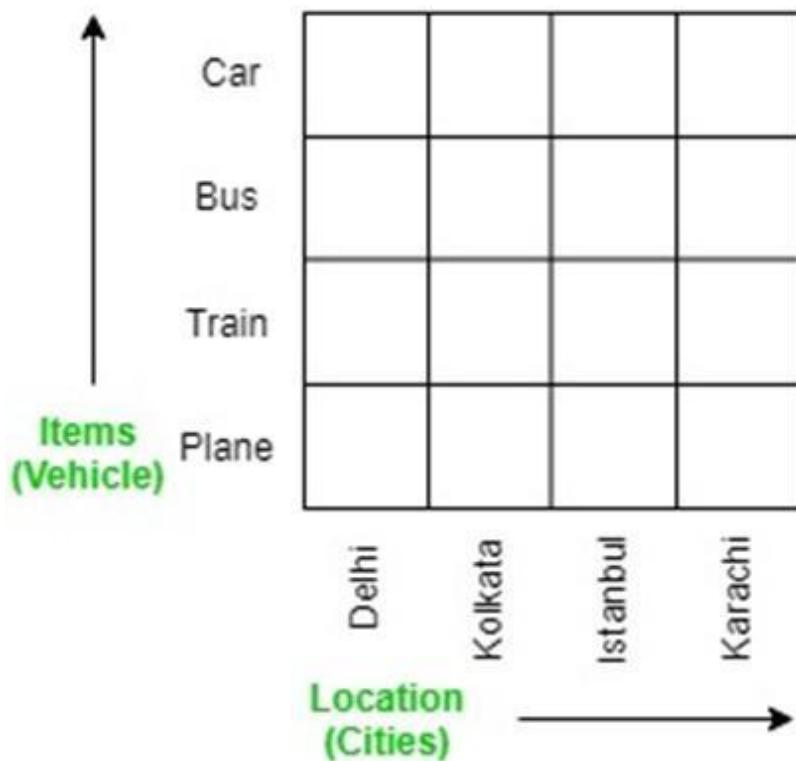
4. Slice:

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



5. Pivot:

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



Code:

Creating Fact Table and Dimension tables.

DIMENSIONS TABLE – CITY DIMENSIONS

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

DIMENSIONS TABLE – CONTACT DIMENSIONS

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

DIMENSIONS TABLE – SUPPLIER DIMENSIONS

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

DIMENSIONS TABLE – PRODUCT DIMENSIONS

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

DIMENSIONS TABLE – TRANSACTION_DATE DIMENSIONS

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

DIMENSIONS TABLE – WAREHOUSE DIMENSIONS

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

FACT TABLE - INVENTORY

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

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

Inserting Values in the above tables:

city_dimension:

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

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

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

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (7298, 'Kota', 'Rajasthan', '324001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (6244, 'Jaipur', 'Rajasthan', '302001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (4098, 'Chennai', 'Tamil Nadu', '600001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (7544, 'Kolkata', 'West Bengal', '700001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (6003, 'Ahmedabad', 'Gujarat', '380001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (8081, 'Lucknow', 'Uttar Pradesh',
'226001'); INSERT INTO city_dim (city_id, city_name,
c_state, pincode) VALUES (5210, 'Chandigarh',
'Chandigarh', '160001'); INSERT INTO city_dim (city_id,
city_name, c_state, pincode) VALUES (7142, 'Bhopal',
'Madhya Pradesh', '462001'); INSERT INTO city_dim
(city_id, city_name, c_state, pincode) VALUES (6001,
'Patna', 'Bihar', '800001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (9012, 'Guwahati', 'Assam', '781001');

INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (7659, 'Trivandrum', 'Kerala', '695001');
```

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

```
INSERT INTO city_dim (city_id, city_name, c_state,
pincode) VALUES (5222, 'Dehradun', 'Uttarakhand',
'248001'); INSERT INTO city_dim (city_id, city_name,
c_state, pincode) VALUES (6211, 'Shimla', 'Himachal
```

The screenshot shows a SQL worksheet interface with the following details:

- Toolbar:** Feedback, Help, User (omshete0550@gmail.com), Actions, Save, Run.
- Left Sidebar:** Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, Code Library.
- SQL Worksheet Area:**

```

40 VALUES (6211, 'Shimla', 'Himachal Pradesh', '171001');
41
42 SELECT * FROM city_dim;

```
- Result Grid:**

CITY_ID	CITY_NAME	C_STATE	PINCODE
7597	Mumbai	Maharashtra	400001
7899	Bangalore	Karnataka	560001
9293	Pune	Maharashtra	411001
6336	Hyderabad	Telangana	500001
5676	Gurgaon	Haryana	122801
5055	Delhi	Delhi	110001
7298	Kota	Rajasthan	324001
6244	Jaipur	Rajasthan	302001
4098	Chennai	Tamil Nadu	600001
7544	Kolkata	West Bengal	700001

Pradesh', '171001');

contact_dim:

```
INSERT INTO contact_dim (contact_id, phone, email)
VALUES ('C2483', '9988776655',
'xbarrett@example.com'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C8454', '9876543210',
'vernandez@example.com'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C8265', '8899776644',
'cooperdavid@example.org'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C3118', '7766554433',
```

```
'monica66@example.net'); INSERT INTO contact_dim  
(contact_id, phone, email) VALUES ('C7283', '9870123456',  
'krivera@example.org'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C7667', '9876012345',  
'rosarioalbert@example.net'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C1342', '8765432109', 'leejennifer@example.net');  
INSERT INTO contact_dim (contact_id, phone, email)  
VALUES ('C3989', '8877665544',  
'breynolds@example.com'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C1699', '9988001122',  
'thompsonzachary@example.org'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C5700', '9012345678',  
'robert84@example.org'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C9238', '9878998899',  
'hoffmanbeth@example.net'); INSERT INTO contact_dim  
(contact_id, phone, email)  
VALUES ('C7601', '9988778899', 'sdavis@example.net');  
INSERT INTO contact_dim (contact_id, phone, email)  
VALUES ('C5895', '9123456789', 'kristina32@example.net');  
INSERT INTO contact_dim (contact_id, phone, email)  
VALUES ('C9589', '8899889988', 'natalie47@example.org');
```

```

INSERT INTO contact_dim (contact_id, phone, email)
VALUES ('C1627', '9900990099',
'longashley@example.org'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C5607', '8765432101',
'coreywilson@example.org'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C9300', '8901234567',
'ebryan@example.com'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C8900', '9876543211',
'burkesharon@example.net'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C2870', '8877665599',
'deborah09@example.org'); INSERT INTO contact_dim
(contact_id, phone, email)
VALUES ('C8997', '9876543232', 'dustinpeterson@example.org');

```

The screenshot shows the Oracle Live SQL interface. On the left is a sidebar with navigation links: Home, SQL Worksheet (which is selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area is titled "SQL Worksheet". It contains the following SQL code:

```

37 v INSERT INTO contact_dim (contact_id, phone, email)
38 VALUES ('C2870', '8877665599', 'deborah09@example.org');
39 v INSERT INTO contact_dim (contact_id, phone, email)
40 VALUES ('C8997', '9876543232', 'dustinpeterson@example.org');
41
42 SELECT * FROM contact_dim
43

```

Below the code, a table displays the data from the contact_dim table:

CONTACT_ID	PHONE	EMAIL
C2483	9988776655	xbarrett@example.com
C8454	9876543210	vhernandez@example.com
C8265	8899776644	cooperdavid@example.org
C3118	7766554433	monica66@example.net
C7283	9870123456	krivera@example.org
C7667	9876012345	rosarioalbert@example.net
C1342	8765432109	leejennifer@example.net
C3989	8877665544	breynolds@example.com

At the bottom of the interface, there is a footer with the text: "2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym" and "Built with ❤ using Oracle APEX · Privacy · Terms of Use".

supplier_dimension:

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S3535', 'Stewart, Clayton and Martinez',  
'C2483');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S7973', 'Carroll, Brady and Hancock', 'C8454');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S2142', 'Singh-Johnson', 'C8265');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S3115', 'Boyer Ltd', 'C3118');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name, contact_id)  
VALUES ('S8922', 'Davis Ltd', 'C7283');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S4540', 'Sandoval, Brown and Peters',  
'C7667');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S2603', 'Madden Inc', 'C1342');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S3568', 'Pearson, Wall and Shelton', 'C3989');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S8035', 'Ellis Ltd', 'C1699');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S3141', 'Johnson Group', 'C5700');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,  
contact_id) VALUES ('S8420', 'Mccoy LLC', 'C9238');
```

```
INSERT INTO supplier_dim (supplier_id, supplier_name,
```

```

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

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S8841', 'Small PLC', 'C5895');

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S3895', 'Carney, Edwards and Chen', 'C9589');

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S9366', 'Ruiz-Garcia', 'C1627');

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S3534', 'Potter and Sons', 'C5607');

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

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S6126', 'Williamson, Ware and Martin',
                           'C8900');

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S3284', 'Price, Sparks and Frey', 'C2870');

INSERT INTO supplier_dim (supplier_id, supplier_name,
                           contact_id) VALUES ('S9796', 'Newton Group', 'C8997');

```

The screenshot shows the Oracle Live SQL interface. The SQL Worksheet pane contains the following code:

```

56 VALUES ('S3535', 'Stewart, Clayton and Martinez', 'C2483');
57
58 INSERT INTO supplier_dim (supplier_id, supplier_name, contact_id)
59 VALUES ('S9796', 'Newton Group', 'C8997');
60
61 SELECT * FROM supplier_dim
62

```

The results pane displays the data from the supplier_dim table:

SUPPLIER_ID	SUPPLIER_NAME	CONTACT_ID
S3535	Stewart, Clayton and Martinez	C2483
S7973	Carroll, Brady and Hancock	C8454
S2142	Singh-Johnson	C8265
S3115	Boyer Ltd	C3118
S8922	Davis Ltd	C7283
S4540	Sandoval, Brown and Peters	C7667
S2603	Madden Inc	C1342
S3568	Pearson, Wall and Shelton	C3989
S8035	Ellis Ltd	C1699

At the bottom of the interface, there is a footer with the text: "2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym".

product_dimension:

```
INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P2859', 'Titan T27', 'watches', 'Titan');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P5937', 'Samsung QLED 21', 'tv', 'Samsung');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P1553', 'Philips Iron I4', 'Electrical App', 'Philips');

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

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P9813', 'US Polo Black Denim', 'Wear', 'US Polo');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P5206', 'Samsung Washing Machine', 'Electrical App',
'Samsung');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P1489', 'Reliance Hair Dryer', 'Electrical App', 'Reliance');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P9351', 'LG 42inch P32', 'tv', 'LG');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P4413', 'Sony Bravia S76', 'tv', 'Sony');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P3825', 'Fossil F31', 'watches', 'Fossil');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P7873', 'Giordano G21', 'watches', 'Giordano');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P1596', 'MI MicroLED M31', 'tv', 'MI');
```

```

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P2693', 'Oneplus O8', 'tv', 'Oneplus');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P4110', 'NutriCook Mixer G21', 'Electrical App', 'NutriCook');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P1235', 'DNMX Shirt S21', 'Wear', 'DNMX');

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

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P5882', 'Giordano G26', 'watches', 'Giordano');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P7062', 'Sonata Kid K2', 'watches', 'Sonata');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P8087', 'Oneplus O20', 'tv', 'Oneplus');

INSERT INTO product_dim (product_id, product_name, product_category,
product_brand) VALUES ('P4582', 'Oneplus O85', 'tv', 'Oneplus');

```

The screenshot shows the Oracle Live SQL interface. On the left, there's a sidebar with various navigation options like Home, SQL Worksheet, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area is titled "SQL Worksheet". In the worksheet, there is a code editor with the following content:

```

40   VALUES ( P4582 , 'Oneplus O85' , 'tv' , 'Oneplus' );
41
42
43 SELECT * FROM product_dim
44

```

Below the code editor, a table is displayed with the following data:

PRODUCT_ID	PRODUCT_NAME	PRODUCT_CATEGORY	PRODUCT_BRAND
P2859	Titan T27	watches	Titan
P5937	Samsung QLED 21	tv	Samsung
P1553	Philips Iron 14	Electrical App	Philips
P7296	Nike Nex	Wear	Nike
P9813	US Polo Black Denim	Wear	US Polo
P5206	Samsung Washing Machine	Electrical App	Samsung
P1489	Reliance Hair Dryer	Electrical App	Reliance
P9351	LG 42inch P32	tv	LG
P4413	Sony Bravia S76	tv	Sony

At the bottom of the interface, there is some footer text: "2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤️ using Oracle APEX - Privacy - Terms of Use".

transaction_date_dimension:

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('29-11-2021', 'DD-MM-YYYY'), TO_DATE('15-01-2023',
'DD-MM-YYYY'), 413);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('18-09-2021', 'DD-MM-YYYY'), TO_DATE('26-05-2022',
'DD-MM-YYYY'), 250);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('14-05-2021', 'DD-MM-YYYY'), TO_DATE('16-05-2022',
'DD-MM-YYYY'), 367);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('21-06-2021', 'DD-MM-YYYY'), TO_DATE('13-10-2021',
'DD-MM-YYYY'), 115);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('29-10-2022', 'DD-MM-YYYY'), TO_DATE('07-11-2022', 'DD-MM-YYYY'),
9);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('04-11-2022', 'DD-MM-YYYY'), TO_DATE('17-11-2022',
'DD-MM-YYYY'), 13);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('18-10-2020', 'DD-MM-YYYY'), TO_DATE('16-05-2023',
'DD-MM-YYYY'), 926);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('04-08-2022', 'DD-MM-YYYY'), TO_DATE('23-05-2023',
'DD-MM-YYYY'), 292);
```

```
INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('01-11-2020', 'DD-MM-YYYY'), TO_DATE('09-10-2022',
'DD-MM-YYYY'), 712);
```

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

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

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

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

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

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

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

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('10-09-2020', 'DD-MM-YYYY'), TO_DATE('25-10-2022',
'DD-MM-YYYY'), 775);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('03-01-2020', 'DD-MM-YYYY'), TO_DATE('14-01-2023',
'DD-MM-YYYY'), 1093);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

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

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)
VALUES (TO_DATE('29-10-2020', 'DD-MM-YYYY'), TO_DATE('27-11-2022',
'DD-MM-YYYY'), 753);

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

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

INSERT INTO transaction_date_dim (arrival_date, dispatch_date, duration_in_warehouse)

VALUES (TO_DATE('23-06-2023', 'DD-MM-YYYY'), TO_DATE('28-06-2023', 'DD-MM-YYYY'),
5);

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

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

The screenshot shows the Oracle Live SQL interface. On the left is a sidebar with options like Home, SQL Worksheet (which is selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area has tabs for SQL Worksheet and Results. In the SQL Worksheet tab, there are two rows of code:

```
40 VALUES (TO_DATE('23-01-2023', 'DD-MM-YYYY'), TO_DATE('23-05-2023',
41 'DD-MM-YYYY'), 122);
42 SELECT * FROM transaction_date_dim;
```

Below the code is a results grid titled "transaction_date_dim" with columns ARRIVAL_DATE, DISPATCH_DATE, and DURATION_IN_WAREHOUSE. The data is as follows:

ARRIVAL_DATE	DISPATCH_DATE	DURATION_IN_WAREHOUSE
29-NOV-21	15-JAN-23	413
18-SEP-21	26-MAY-22	250
14-MAY-21	16-MAY-22	367
21-JUN-21	13-OCT-21	115
29-OCT-22	07-NOV-22	9
04-NOV-22	17-NOV-22	13
18-OCT-20	16-MAY-23	926
04-AUG-22	23-MAY-23	292
01-NOV-20	09-OCT-22	712

At the bottom of the interface, it says "2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym" and "Built with ❤ using Oracle APEX - Privacy - Terms of Use".

warehouse_dimension:

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

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

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

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

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

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

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

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

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

```
VALUES ('W8571', 'KerelaSweP', 'Kerela', 6336);
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
VALUES ('W4563', 'KerelaSweP', 'Kerela', 6336);
```

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

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

The screenshot shows the Oracle Live SQL interface. On the left, there's a sidebar with navigation links like Home, SQL Worksheet, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area is titled "SQL Worksheet". It contains the following SQL code:

```

28 VALUES ('W4150', 'HyderabadHub', 'Hyderabad', 5055);
29 v INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)
30 VALUES ('W5472', 'BombayX', 'Mumbai', 7597);
31 v INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)
32 VALUES ('W8373', 'DelhiCaps', 'Delhi', 7099);
33 v INSERT INTO warehouse_dim (warehouse_id, warehouse_name, warehouse_location, city_id)

```

Below the code, a table displays the data inserted into the warehouse_dim table:

WAREHOUSE_ID	WAREHOUSE_NAME	WAREHOUSE_LOCATION	CITY_ID
W7276	BombayX	Mumbai	7597
W7001	DelhiCaps	Delhi	7099
W1296	BombayZ	Mumbai	7597
W8451	BangalorIS	Bangalore	9293
W8571	KerelaSweP	Kerela	6336
W4968	GurgaonHDES	Gurgaon	5676
W1700	HyderabadHub	Hyderabad	5055
W4035	BombayX	Mumbai	7597

At the bottom of the interface, there is a footer with the text: "2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation · Ask Tom · Dev Gym" and "Built with ❤️ using Oracle APEX · Privacy · Terms of Use".

inventory_fact_table:

CREATE TABLE inventory_management_fact

AS SELECT

1 AS

transaction_id,

p.product_id,

w.warehouse_id,

s.supplier_id,

ad.arrival_date,

ad.dispatch_date,

0 AS no_of_boxes

FROM

product_dim p

CROSS JOIN

warehouse_dim

w CROSS JOIN

supplier_dim s

CROSS JOIN

transaction_date_dim ad;

Inserting random values for transaction_id and

no_of_boxes BEGIN

FOR rec IN (SELECT rowid, product_id FROM inventory_management_fact) LOOP

UPDATE inventory_management_fact

SET

transaction_id = ROUND(DBMS_RANDOM.VALUE(50, 500)),

```

no_of_boxes = ROUND(DBMS_RANDOM.VALUE(1, 1000))

WHERE rowid =

rec.rowid; END LOOP;

COMMIT

; END;

/

```

SELECT * FROM inventory_management_fact

The screenshot shows the Oracle Live SQL interface. On the left, there's a sidebar with 'Home', 'SQL Worksheet' (selected), 'My Session', 'Schema', 'Quick SQL', 'My Scripts', 'My Tutorials', and 'Code Library'. The main area is titled 'SQL Worksheet' and contains the following code:

```

31
32 SELECT * FROM inventory_management_fact
33

```

Below the code is a table with the following data:

	TRANSACTION_ID	PRODUCT_ID	WAREHOUSE_ID	SUPPLIER_ID	ARRIVAL_DATE	DISPATCH_DATE	NO_OF_BOXES
317	P1235	W1169	S1561	29-NOV-21	15-JAN-23	570	
320	P1235	W1169	S2142	29-NOV-21	15-JAN-23	174	
132	P1235	W1169	S2603	29-NOV-21	15-JAN-23	826	
137	P1235	W1169	S3115	29-NOV-21	15-JAN-23	896	
286	P1235	W1169	S3141	29-NOV-21	15-JAN-23	233	
68	P1235	W1169	S3284	29-NOV-21	15-JAN-23	42	
325	P1235	W1169	S3534	29-NOV-21	15-JAN-23	289	
66	P1235	W1169	S3535	29-NOV-21	15-JAN-23	64	
401	P1235	W1169	S3568	29-NOV-21	15-JAN-23	235	
319	P1235	W1169	S3895	29-NOV-21	15-JAN-23	838	

At the bottom of the interface, it says: 2023 Oracle - Live SQL 23.4.1, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 - Database Documentation - Ask Tom - Dev Gym
Built with ❤ using Oracle APEX - Privacy - Terms of Use

1. Slice:

CREATE MATERIALIZED VIEW mv_slice1

AS SELECT *

FROM

inventory_management_fact

WHERE SUPPLIER_ID = 'S2142';

SELECT * FROM mv_slice1;

The screenshot shows a SQL worksheet interface with a sidebar containing navigation links like Home, SQL Worksheet, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area displays a table titled 'SQL Worksheet' with the following data:

TRANSACTION_ID	PRODUCT_ID	WAREHOUSE_ID	SUPPLIER_ID	ARRIVAL_DATE	DISPATCH_DATE	NO_OF_BOXES
320	P1235	W1169	S2142	29-NOV-21	15-JAN-23	174
451	P1235	W1296	S2142	29-NOV-21	15-JAN-23	945
256	P1235	W1452	S2142	29-NOV-21	15-JAN-23	988
266	P1235	W1700	S2142	29-NOV-21	15-JAN-23	305
282	P1235	W1796	S2142	29-NOV-21	15-JAN-23	279
168	P1235	W1934	S2142	29-NOV-21	15-JAN-23	791
267	P1235	W2006	S2142	29-NOV-21	15-JAN-23	186
311	P1235	W3067	S2142	29-NOV-21	15-JAN-23	575
158	P1235	W4035	S2142	29-NOV-21	15-JAN-23	289
205	P1235	W4150	S2142	29-NOV-21	15-JAN-23	963

2. Dice

CREATE MATERIALIZED VIEW mv_dice

AS SELECT *

FROM inventory_management_fact

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

SELECT * FROM mv_dice;

The screenshot shows a SQL worksheet interface with a sidebar containing navigation links like Home, SQL Worksheet, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area displays a table titled 'SQL Worksheet' with the following data:

TRANSACTION_ID	PRODUCT_ID	WAREHOUSE_ID	SUPPLIER_ID	ARRIVAL_DATE	DISPATCH_DATE	NO_OF_BOXES
472	P1235	W1296	S1561	29-NOV-21	15-JAN-23	802
451	P1235	W1296	S2142	29-NOV-21	15-JAN-23	945
401	P1235	W1296	S2683	29-NOV-21	15-JAN-23	715
227	P1235	W1296	S3115	29-NOV-21	15-JAN-23	990
125	P1235	W1296	S3141	29-NOV-21	15-JAN-23	432
262	P1235	W1296	S3284	29-NOV-21	15-JAN-23	392
120	P1235	W1296	S3534	29-NOV-21	15-JAN-23	620
54	P1235	W1296	S3535	29-NOV-21	15-JAN-23	75
224	P1235	W1296	S3568	29-NOV-21	15-JAN-23	295
158	P1235	W1296	S3895	29-NOV-21	15-JAN-23	15

3. Rollup

```

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

```

```

CREATE MATERIALIZED VIEW mv_rollup_1 AS
SELECT * FROM vw_rollup;

```

TRANSACTION_ID	Total Quantity
50	88329
51	169260
52	192914
53	175933
54	170008
55	170702
56	189967
57	196045
58	182652
59	176034
60	186782

4. Drilldown

```

CREATE OR REPLACE VIEW vw_drilldown
AS SELECT

```

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

```
CREATE MATERIALIZED VIEW mv_drilldown
```

```
AS SELECT * FROM vw_drilldown;
```

The screenshot shows a SQL worksheet interface with a sidebar on the left containing navigation links like Home, SQL Worksheet, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area displays a table titled 'SQL Worksheet' with three columns: TRANSACTION_ID, WAREHOUSE_ID, and Total Boxes. The data in the table is as follows:

TRANSACTION_ID	WAREHOUSE_ID	Total Boxes
317	W1169	8719
424	W1169	11037
401	W1296	8584
126	W1296	6300
54	W1296	8180
402	W1296	6163
174	W1296	10231
412	W1452	11205
225	W1452	5051
369	W1452	7497
482	W1452	13760

EXPERIMENT - 04

Implementation

: Pivot Table:

SUM of WEEKLY_SALES	DRUG_NAME	AllerClear 10	CardioGuard 50	Dermacare Q1ntr	DiabetCare 500	ImmunoGuard 7	MediHealth 100	NeuroCalm XR	RespiGen Cough	StomachEase PI	VitaBoost Multi	Grand Total
CATEGORY_NAME												
Allergy		64000										64000
Anaesthetics												120000
Cardiovascular		40000										40000
Cough and Cold												90000
Dermatology			48000									48000
Endocrinology				37500								37500
Gastrointestinal												102000
Immunology					70000							70000
Neurology							36000					36000
Vitamins and Supplements												84000
Grand Total		64000	40000	48000	37500	70000	120000	36000	90000	102000	84000	691500

Slice:

The screenshot shows a Microsoft Power BI Pivot Table Editor window. The main area displays a pivot table with columns for Sales Date, Category Name, and Sum of Month. The data is grouped by Sales Date (01/08/2023, 02/08/2023, 03/08/2023, 04/08/2023) and Category Name (Allergy, Analgesics, Cough and Cold, Dermatology, Endocrinology, Gastrointestinal, Immunology, Vitamins and Su). The total sales for each date range are also shown. The right side of the window contains the 'Pivot table editor' interface with sections for Rows, Columns, and Values, along with search and filter options.

SALES_DATE	CATEGORY_NAME	SUM of MONTH
01/08/2023	Allergy	32000
01/08/2023	Analgesics	80000
01/08/2023	Cough and Cold	60000
01/08/2023	Gastrointestinal	68000
01/08/2023 Total		208000
02/08/2023	Allergy	32000
02/08/2023	Analgesics	80000
02/08/2023	Cough and Cold	60000
02/08/2023	Dermatology	48000
02/08/2023	Endocrinology	50000
02/08/2023	Immunology	56000
02/08/2023	Vitamins and Su	84000
02/08/2023 Total		330000
03/08/2023	Allergy	32000
03/08/2023	Analgesics	80000
03/08/2023	Cardiovascular	40000
03/08/2023	Neurology	36000
03/08/2023	Vitamins and Su	84000
03/08/2023 Total		272000
04/08/2023	Allergy	32000
04/08/2023	Cardiovascular	40000
04/08/2023	Gastrointestinal	68000
04/08/2023	Neurology	36000
04/08/2023	Vitamins and Su	84000
04/08/2023 Total		240000

Dice:

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

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

Rollup:

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

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

Drilldown:

Screenshot of a Microsoft Excel PivotTable editor showing a sales analysis. The PivotTable displays weekly sales data from January 1, 2023, to January 10, 2023, categorized by Sales Date, Category Name, and Drug Name.

Pivot Table Editor Fields:

- Rows:** SALES_DATE (Order: Ascen., Sort by: SALES, Show totals checked)
- Columns:** CATEGORY_NAME (Order: Ascen., Sort by: CATEG., Show totals checked)
- Values:** MONTHLY_SALES (Summarize by: SUM, Show as: Default)

Data Summary:

SALES_DATE	CATEGORY_NAME	MONTHLY_SALES
01/01/2023	Allergy	80000
02/01/2023	MediClear 10	32000
03/01/2023	MediHealth 100	32000
04/01/2023	CardioGuard 50	40000
05/01/2023	Resigen Cough Syrup	60000
06/01/2023	Dermacare Ointment	60000
07/01/2023	AlleviCare 10	40000
08/01/2023	MediGuard 50	40000
09/01/2023	NeuroCalm XR	60000
10/01/2023	VitaBoost Multivitamin	60000
Grand Total		288000

	R	S	T	U	V	W	X	Y	Z	AA	AB
3											
4											
5											
6	Endocrinology	Endocrinology	T	Gastrointestinal	Gastrointestinal	Immunology	Immunology	Tox	Neurology	Neurology	Total
7	DiabetCare 500			StomachEase Plus		ImmunoGuard 75			NeuroCalm XR		Vitamins and Su
8				68000		68000					Grand Total
9	50000	50000				56000	56000			84000	84000
10								36000	36000	84000	84000
11			68000	68000			36000	36000	84000	84000	272000
12			68000	68000	56000	56000					344000
13	50000	50000	68000	68000	56000	56000					346000
14	50000	50000	68000	68000	56000	56000			84000	84000	330000
15			68000	68000							128000
16					56000	56000	36000	36000			252000
17							36000	36000			296000
18	150000	150000	408000	408000	280000	280000	144000	144000	336000	336000	2766000
19											
20											

EXPERIMENT - 05

Aim: Implementation of Bayesian algorithm

Theory:

Data Mining Bayesian Classifiers

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

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

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

Bayes's theorem is expressed mathematically by the following equation that is given below. For proposition X and evidence Y,

- $P(X)$, the prior, is the primary degree of belief in X
- $P(X/Y)$, the posterior is the degree of belief having accounted for Y.

- The quotient $\frac{P(Y/X)}{P(Y)}$ represents the supports Y provides for

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

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

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

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

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

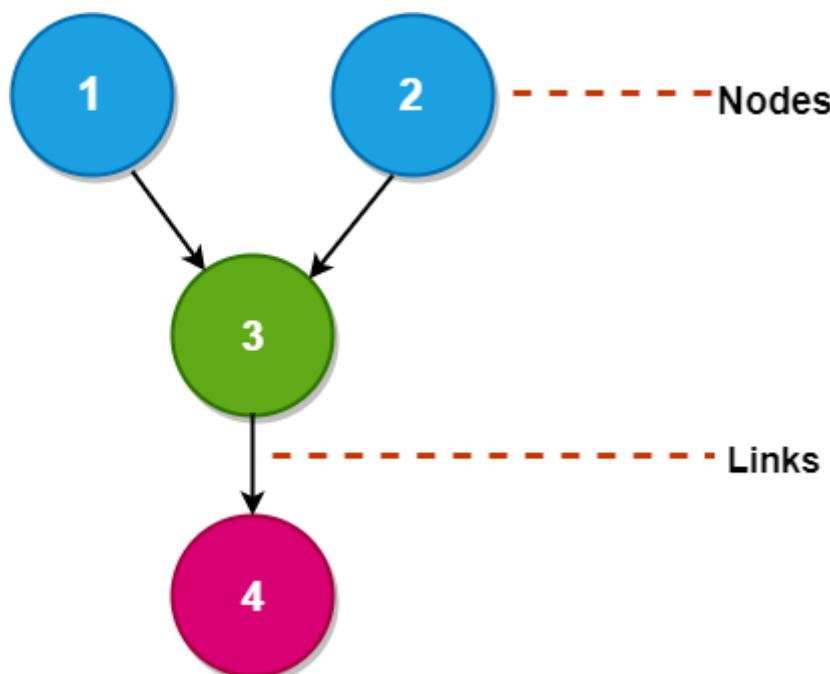
$$\text{or, } P(X \cap Y) = P(X/Y)P(Y) = P(Y/X)P(X)$$

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

Bayesian network:

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

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



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

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

CODE:

```
import pandas as
pd import math
data =


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

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

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

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

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

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

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

#Likelihood Probabilities
```

```

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

total = len(subset)

return count /
total

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

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

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

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

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

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

```

```

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

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

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

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

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

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

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

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

```

Output:

The screenshot shows a Jupyter Notebook interface with the following details:

- File:** DWM_Exp5.ipynb
- Cells:** One cell is visible, showing the execution of the provided Python code.
- Output:**
 - Input: Enter Price (Low, Average, High): High
Enter Number Of Boxes: (30, 40, 10): 30
Enter Product Brand (Sony, LG, Nike): Sony
Enter location: (Suburbs, City, Downtown): Downtown
 - Intermediate Results:
Total occurrences of Yes: 24
Total occurrences of No: 16
Prior probability for Yes 0.5854
Prior probability for No: 0.3902
 - Posterior Probabilities:
The Likelihood Probabilities are:
 $P(\text{price} = \text{High} / \text{buy} = \text{'Yes'}) = 0.375$
 $P(\text{price} = \text{High} / \text{buy} = \text{'No'}) = 0.4375$
 - $P(\text{number_of_boxes} = 30 / \text{buy} = \text{'Yes'}) = 0.2917$
 $P(\text{number_of_boxes} = 30 / \text{buy} = \text{'No'}) = 0.0625$
 - $P(\text{product_brand} = \text{Sony} / \text{buy} = \text{'Yes'}) = 0.3333$
 $P(\text{product_brand} = \text{Sony} / \text{buy} = \text{'No'}) = 0.25$
 - $P(\text{location} = \text{Downtown} / \text{buy} = \text{'Yes'}) = 0.4583$
 $P(\text{location} = \text{Downtown} / \text{buy} = \text{'No'}) = 0.3125$
 - Final Output:
The posterior probabilities are:
Probability of Purchased Yes: 0.6250

EXPERIMENT - 06

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

Theory:

Classification:

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

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

Clustering:

Clustering refers to a technique of grouping objects so that objects with the same functionalities come together and objects with different functionalities go apart. In other words, we can say that clustering is a process of portioning a data set into a set of meaningful subclasses, known as clusters. Clustering is the same as classification in which data is grouped. However, unlike classification, the groups are not previously defined.

Instead, the grouping is achieved by determining similarities between data according to characteristics found in the real data. The groups are called Clusters.

Association Rules:

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

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



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

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

Support

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

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

Confidence

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

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

Lift

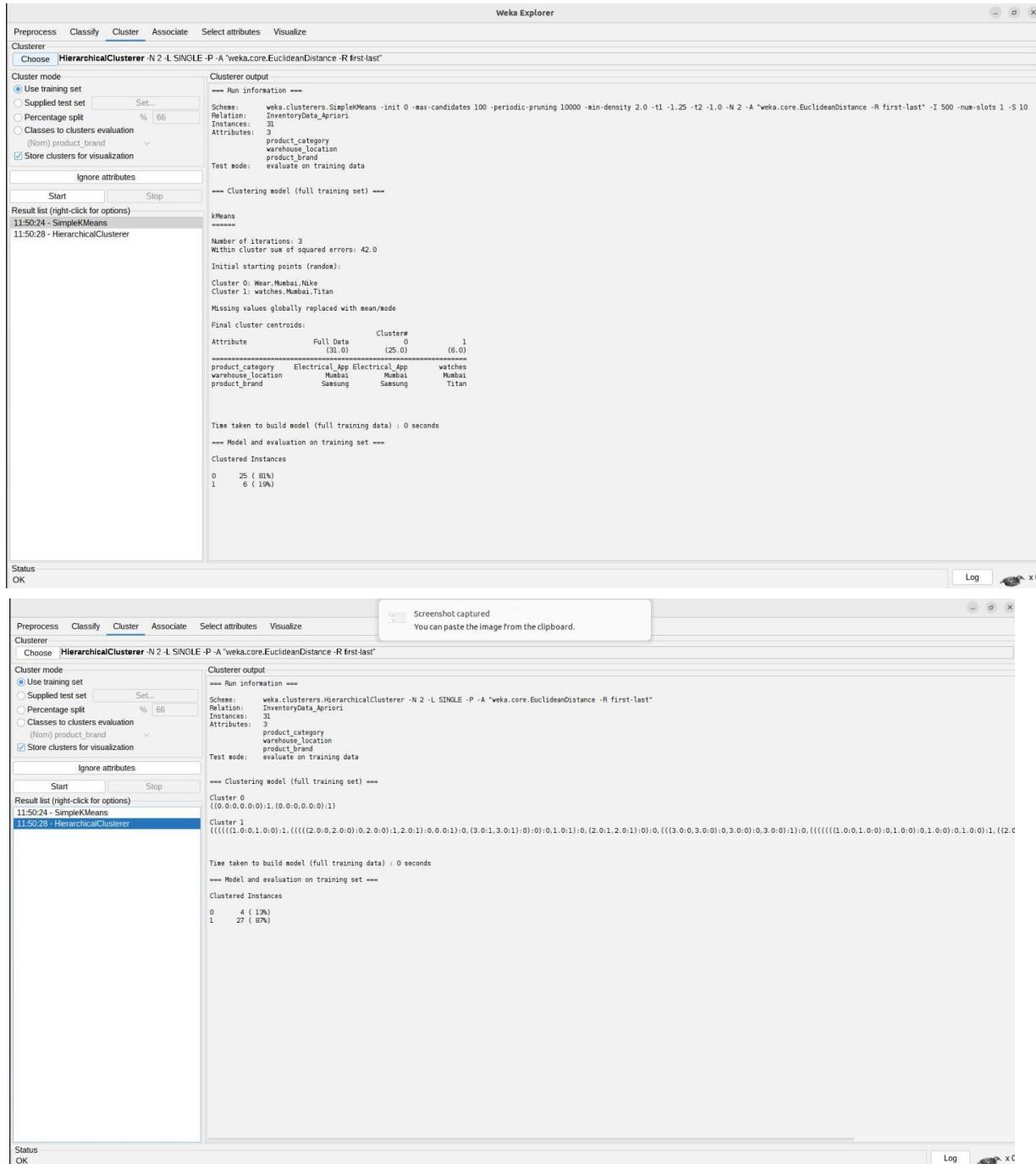
It is the strength of any rule, which can be defined as below formula:

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

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

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

Output:



Screenshot captured
You can paste the image from the clipboard.

Classifier output

```

Test mode: 10-fold cross-validation
--- Classifier model (full training set) ---
Naive Bayes Classifier
  Attribute Class
    product_brand
      Titan  Samsung Philips  Nike
        (0.17) (0.31) (0.29) (0.23)

  project_category
    watches   4.0   3.0   2.0   1.0
    Electrical_App  1.0   8.0   6.0   3.0
    Wear     1.0   1.0   3.0   5.0
    TV      3.0   2.0   2.0   1.0
    [total]  9.0  14.0  13.0  11.0

  warehouse_location
    Mumbai   5.0   6.0   5.0   6.0
    Delhi    2.0   6.0   6.0   3.0
    [total]  7.0  12.0  11.0   9.0

Time taken to build model: 0 seconds
--- Stratified cross-validation ---
--- Summary ---
Correctly Classified Instances      13          41.935 %
Incorrectly Classified Instances   18          58.0645 %
Kappa statistic                   0.196
Mean absolute error               0.3428
Root mean square error            0.418
Relative absolute error           92.127 %
Root relative squared error       96.7671 %
Total Number of Instances         31

--- Detailed Accuracy By Class ---
           TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
  0.400  0.038  0.667  0.400  0.500  0.450  0.808  0.456  Titan
  0.700  0.429  0.458  0.700  0.538  0.725  0.595  0.366  Samsung
  0.800  0.167  0.800  0.800  0.800  0.745  0.800  0.800  Philips
  0.571  0.167  0.500  0.571  0.533  0.387  0.625  0.472  Nike

Weighted Avg.  0.419  0.295  0.362  0.419  0.375  0.170  0.578  0.380

--- Confusion Matrix ---
  a | b -> classified as
  2 | 1 | 0 | 2 | a = Titan
  0 | 7 | 3 | 0 | b = Samsung
  1 | 6 | 0 | 2 | c = Philips
  0 | 2 | 1 | 4 | d = Nike

```

Weka Explorer

```

Test options
  ○ Use training set
  ○ Supplied test set [ Set... ]
  ● Cross-validation Folds 10
  ○ Percentage split % 66
  More options...

(Nom) warehouse_location
  Start | Stop

Result list (right-click for options)
11:49:51 - bayes.NaiveBayes
11:50:06 - rules.DecisionTable
11:50:10 - rules.DecisionTable
11:53:07 - trees.J48
11:53:35 - trees.J48

Status OK

```



EXPERIMENT - 07

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

Theory:

K means Clustering

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

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

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

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

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

The algorithm works as follows:

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

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

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

The above algorithm in pseudocode is as follows:

Initialize k means with random values

--> For a given number of iterations:

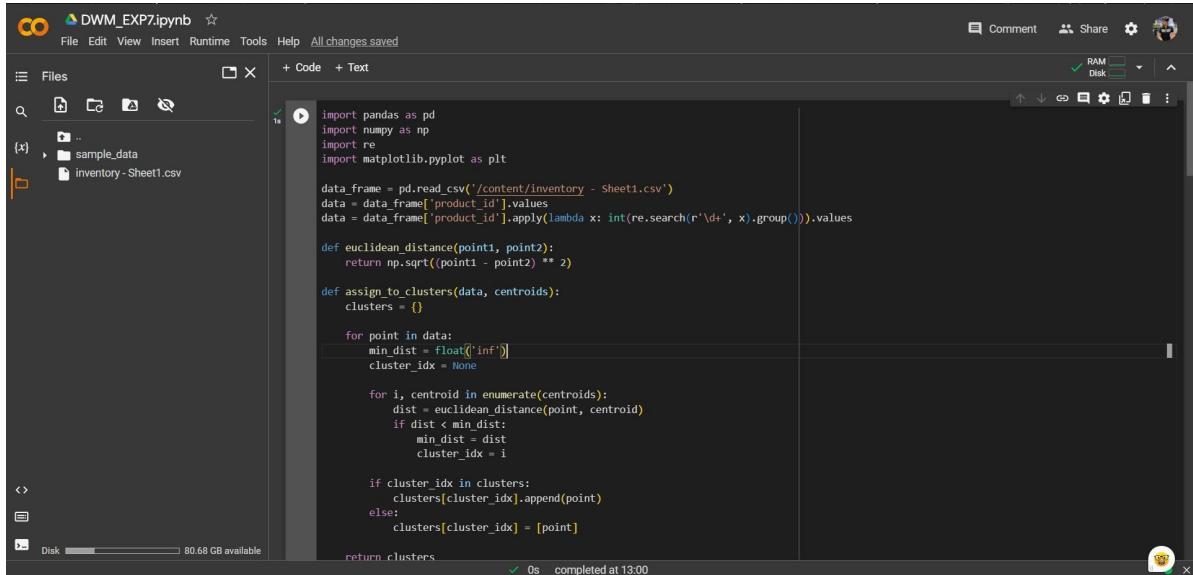
--> Iterate through items:

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

--> Assign item to mean

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

Output:



```

File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
1s
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt

data_frame = pd.read_csv('/content/inventory - Sheet1.csv')
data = data_frame['product_id'].values
data = data_frame['product_id'].apply(lambda x: int(re.search(r'\d+', x).group(0))).values

def euclidean_distance(point1, point2):
    return np.sqrt((point1 - point2) ** 2)

def assign_to_clusters(data, centroids):
    clusters = {}

    for point in data:
        min_dist = float('inf')
        cluster_idx = None

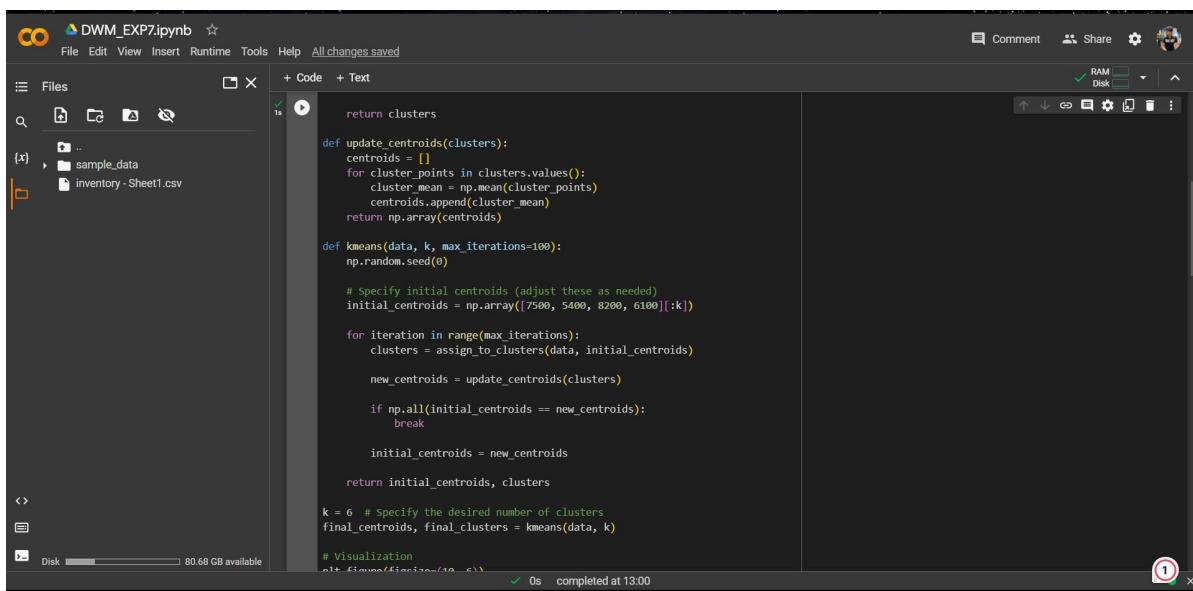
        for i, centroid in enumerate(centroids):
            dist = euclidean_distance(point, centroid)
            if dist < min_dist:
                min_dist = dist
                cluster_idx = i

        if cluster_idx in clusters:
            clusters[cluster_idx].append(point)
        else:
            clusters[cluster_idx] = [point]

    return clusters

```

Disk 80.68 GB available 0s completed at 13:00



```

return clusters

def update_centroids(clusters):
    centroids = []
    for cluster_points in clusters.values():
        cluster_mean = np.mean(cluster_points)
        centroids.append(cluster_mean)
    return np.array(centroids)

def kmeans(data, k, max_iterations=100):
    np.random.seed(0)

    # Specify initial centroids (adjust these as needed)
    initial_centroids = np.array([7500, 5400, 8200, 6100])[:k]

    for iteration in range(max_iterations):
        clusters = assign_to_clusters(data, initial_centroids)

        new_centroids = update_centroids(clusters)

        if np.all(initial_centroids == new_centroids):
            break

        initial_centroids = new_centroids

    return initial_centroids, clusters

k = 6 # Specify the desired number of clusters
final_centroids, final_clusters = kmeans(data, k)

# Visualization
plt.figure(figsize=(10, 6))

```

Disk 80.68 GB available 0s completed at 13:00

C32_Om_2103163_DWM

DWM_EXP7.ipynb

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
16
def kmeans_clustering(data):
    # specify the desired number of clusters
    k = 6
    final_centroids, final_clusters = kmeans(data, k)

    # visualization
    plt.figure(figsize=(10, 6))

    # Plot data points
    for cluster_idx, cluster_points in final_clusters.items():
        plt.scatter(cluster_points, np.full_like(cluster_points, cluster_idx + 1), label=f'Cluster {cluster_idx + 1}')

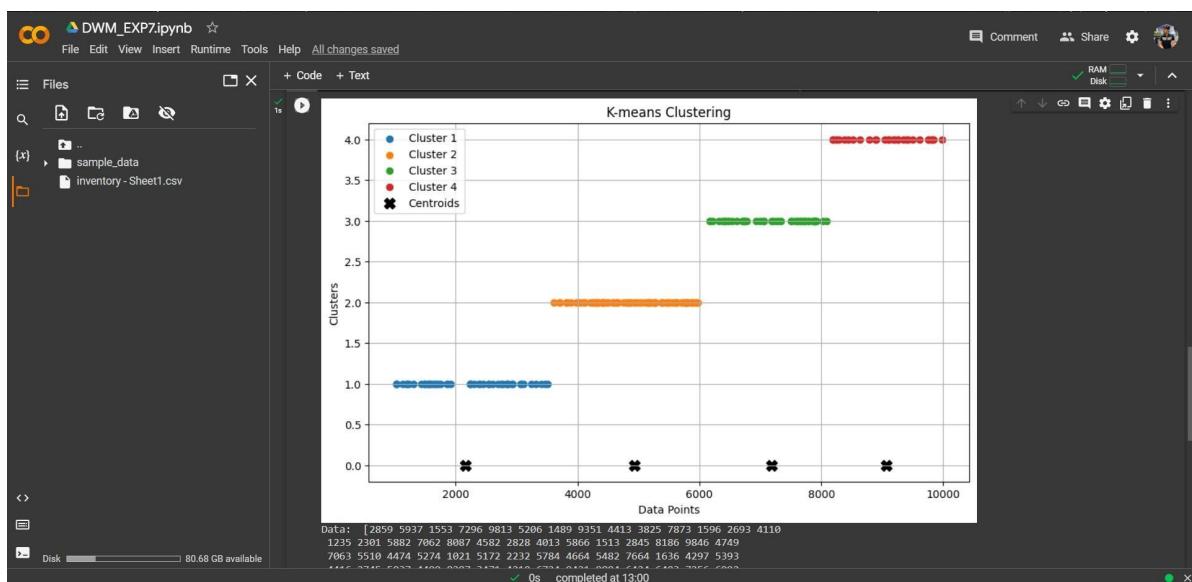
    # Plot centroids
    plt.scatter(final_centroids, np.full_like(final_centroids, 0), marker='x', s=100, color='black', label='Centroids')

    plt.xlabel('Data Points')
    plt.ylabel('Clusters')
    plt.title('K-means Clustering')
    plt.legend()
    plt.grid(True)

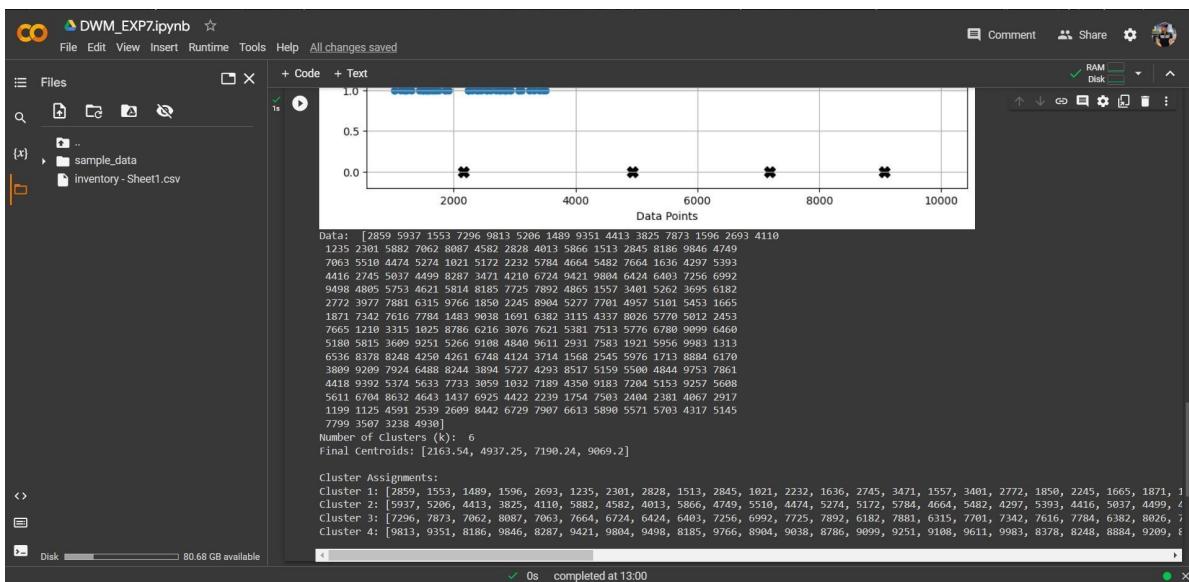
    plt.show()

print("Data: ", data)
print("Number of Clusters (k): ", k)
print("Final Centroids: ", end="")
print([round(x, 2) for x in final_centroids])
print("\nCluster Assignments:")
for cluster_idx, cluster_points in final_clusters.items():
    print(f"Cluster {cluster_idx + 1}: ", end="")
    print(cluster_points)
```

Disk 80.68 GB available 0s completed at 13:00



C32_Om_2103163_DWM



EXPERIMENT - 08

Aim: Implementation of any one Hierarchical Clustering method

Theory:

A Hierarchical clustering method works via grouping data into a tree of clusters.

Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

Identify the 2 clusters which can be closest together, and

Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called a Dendrogram (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view).

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

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

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

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

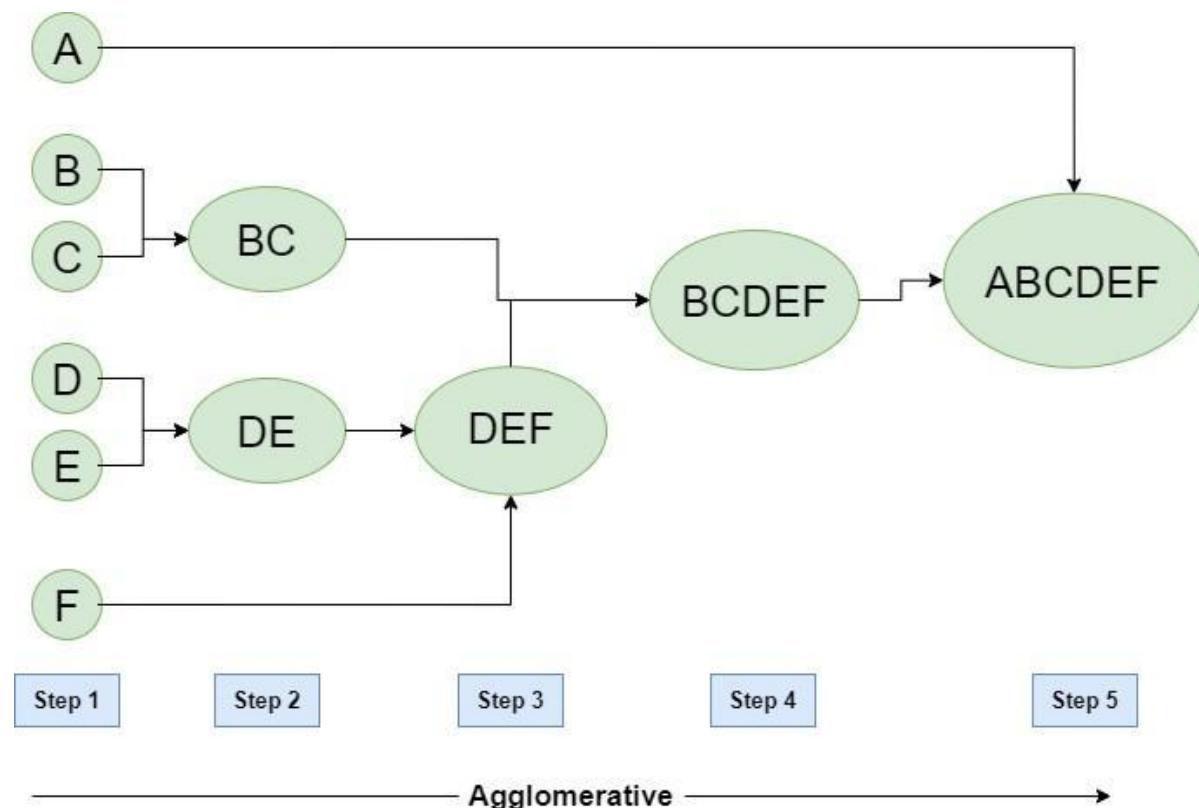
7. This method can handle different types of data and reveal the relationships among the clusters. However, it can have high computational cost and results can be sensitive to some conditions.

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

The algorithm for Agglomerative Hierarchical Clustering is:

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

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



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

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

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

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

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

Output:

+ Code + Text Connect

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

	transaction_id	product_id	product_name	product_category	product_brand	warehouse_id	warehouse_name	warehouse_location	city_id	supplier_id	supplier_name	contact_id
0	1	P2859	Titan T27	watches	Titan	W7276	BombayX	Mumbai	C7597	S3535	Stewart, Clayton and Martinez	C2483 001-1
1	2	P5937	Samsung QLED 21	tv	Samsung	W7001	DelhiCaps	Delhi	C7099	S7973	Carroll, Brady and Hancock	C8454 601.44
2	3	P1553	Philips Iron 14	Electrical App	Philips	W1296	BombayZ	Mumbai	C7597	S2142	Singh-Johnson	C8265
3	4	P7296	Nike Nex	Wear	Nike	W8451	BangaloreS	Bangalore	C9293	S3115	Boyer Ltd	C3118 (989)k
4	5	P8813	US Polo Black Denim	Wear	US Polo	W8571	KerelaSwept	Kerala	C6336	S8922	Davis Ltd	C7283
...
195	196	P5145	LG hair dryer	Electrical App	LG	W9621	GurgaonHDES	Gurgaon	C5676	S5845	Dixon and Sons	C9036 001-4
196	197	P7799	Apple watch series 8	watches	Apple	W4993	HyderabadHub	Hyderabad	C5055	S7793	Booth LLC	C7568

The screenshot shows a Jupyter Notebook interface with the following code:

```
+ Code + Text  
Connect ▾  
  
▶ X = np.array(dataset)  
  
class Distance_computation_grid(object):  
    def __init__(self):  
        pass  
  
    def compute_distance(self, samples):  
        Distance_mat = np.zeros((len(samples), len(samples)))  
  
        for i in range(Distance_mat.shape[0]):  
            for j in range(Distance_mat.shape[0]):  
                if i != j:  
                    Distance_mat[i, j] = float(self.distance_calculate(samples[i], samples[j]))  
                else:  
                    Distance_mat[i, j] = 10**4  
        return Distance_mat  
  
    # For two samples  
    def distance_calculate(self, sample1, sample2):  
        dist = []  
  
        for i in range(len(sample1)):  
            for j in range(len(sample2)):  
                try:  
                    # Try to convert elements to float and calculate distance  
                    dist.append(np.linalg.norm(np.array(float(sample1[i])) - np.array(float(sample2[j]))))  
                except (ValueError, TypeError):  
                    # Handle non-numeric or missing values (e.g., NaN) here  
                    pass  
  
        if not dist:  
            return float('inf') # Return a large value for cases with no valid numeric values  
        else:  
            mindist =
```

The image shows three vertically stacked code cells from a Jupyter Notebook interface. Each cell contains Python code related to clustering operations.

Cell 1:

```

+ Code + Text
    pass

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

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

    if n >= m:
        for i in range(m):
            for j in range(n):
                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)

```

Cell 2:

```

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

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

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

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

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

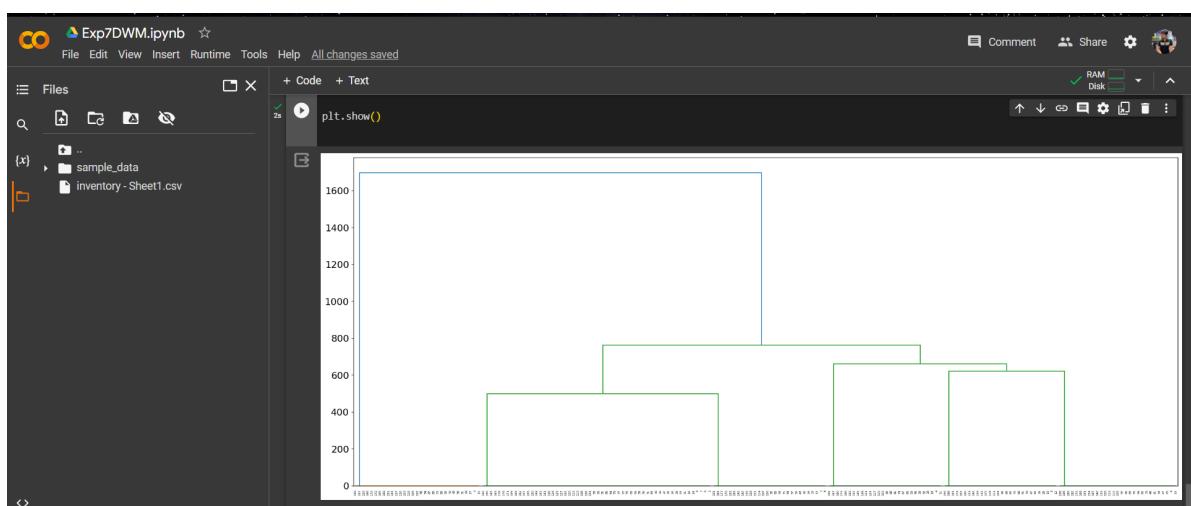
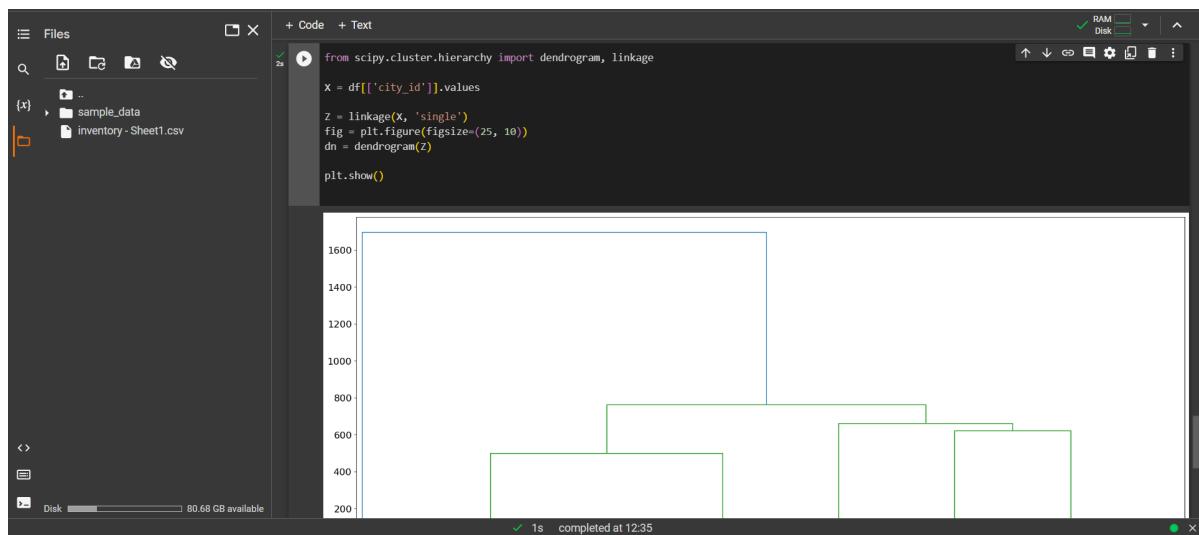
```

Cell 3:

```

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

Sample size before clustering: 200
Cluster Node 1: [0]
Cluster Node 2: [1]
Progression (Current Sample): [[[0, [1]], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], [146], [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167], [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178], [179], [180], [181], [182], [183], [184], [185], [186], [187], [188], [189], [190], [191], [192], [193], [194], [195], [196], [197], [198], [199], [200], [201], [202], [203], [204], [205], [206], [207], [208], [209], [210], [211], [212], [213], [214], [215], [216], [217], [218], [219], [220], [221], [222], [223], [224], [225], [226], [227], [228], [229], [230], [231], [232], [233], [234], [235], [236], [237], [238], [239], [240], [241], [242], [243], [244], [245], [246], [247], [248], [249], [250], [251], [252], [253], [254], [255], [256], [257], [258], [259], [260], [261], [262], [263], [264], [265], [266], [267], [268], [269], [270], [271], [272], [273], [274], [275], [276], [277], [278], [279], [280], [281], [282], [283], [284], [285], [286], [287], [288], [289], [290], [291], [292], [293], [294], [295], [296], [297], [298], [299], [300], [301], [302], [303], [304], [305], [306], [307], [308], [309], [310], [311], [312], [313], [314], [315], [316], [317], [318], [319], [320], [321], [322], [323], [324], [325], [326], [327], [328], [329], [330], [331], [332], [333], [334], [335], [336], [337], [338], [339], [340], [341], [342], [343], [344], [345], [346], [347], [348], [349], [350], [351], [352], [353], [354], [355], [356], [357], [358], [359], [360], [361], [362], [363], [364], [365], [366], [367], [368], [369], [370], [371], [372], [373], [374], [375], [376], [377], [378], [379], [380], [381], [382], [383], [384], [385], [386], [387], [388], [389], [390], [391], [392], [393], [394], [395], [396], [397], [398], [399], [400], [401], [402], [403], [404], [405], [406], [407], [408], [409], [410], [411], [412], [413], [414], [415], [416], [417], [418], [419], [420], [421], [422], [423], [424], [425], [426], [427], [428], [429], [430], [431], [432], [433], [434], [435], [436], [437], [438], [439], [440], [441], [442], [443], [444], [445], [446], [447], [448], [449], [450], [451], [452], [453], [454], [455], [456], [457], [458], [459], [460], [461], [462], [463], [464], [465], [466], [467], [468], [469], [470], [471], [472], [473], [474], [475], [476], [477], [478], [479], [480], [481], [482], [483], [484], [485], [486], [487], [488], [489], [490], [491], [492], [493], [494], [495], [496], [497], [498], [499], [500], [501], [502], [503], [504], [505], [506], [507], [508], [509], [510], [511], [512], [513], [514], [515], [516], [517], [518], [519], [520], [521], [522], [523], [524], [525], [526], [527], [528], [529], [530], [531], [532], [533], [534], [535], [536], [537], [538], [539], [540], [541], [542], [543], [544], [545], [546], [547], [548], [549], [550], [551], [552], [553], [554], [555], [556], [557], [558], [559], [550], [551], [552], [553], [554], [555], [556], [557], [558], [559], [560], [561], [562], [563], [564], [565], [566], [567], [568], [569], [570], [571], [572], [573], [574], [575], [576], [577], [578], [579], [580], [581], [582], [583], [584], [585], [586], [587], [588], [589], [581], [582], [583], [584], [585], [586], [587], [588], [589], [590], [591], [592], [593], [594], [595], [596], [597], [598], [599], [591], [592], [593], [594], [595], [596], [597], [598], [599], [600], [601], [602], [603], [604], [605], [606], [607], [608], [609], [610], [611], [612], [613], [614], [615], [616], [617], [618], [619], [611], [612], [613], [614], [615], [616], [617], [618], [619], [620], [621], [622], [623], [624], [625], [626], [627], [628], [629], [621], [622], [623], [624], [625], [626], [627], [628], [629], [630], [631], [632], [633], [634], [635], [636], [637], [638], [639], [631], [632], [633], [634], [635], [636], [637], [638], [639], [640], [641], [642], [643], [644], [645], [646], [647], [648], [649], [641], [642], [643], [644], [645], [646], [647], [648], [649], [650], [651], [652], [653], [654], [655], [656], [657], [658], [659], [660], [661], [662], [663], [664], [665], [666], [667], [668], [669], [661], [662], [663], [664], [665], [666], [667], [668], [669], [670], [671], [672], [673], [674], [675], [676], [677], [678], [679], [671], [672], [673], [674], [675], [676], [677], [678], [679], [680], [681], [682], [683], [684], [685], [686], [687], [688], [689], [681], [682], [683], [684], [685], [686], [687], [688], [689], [690], [691], [692], [693], [694], [695], [696], [697], [698], [699], [691], [692], [693], [694], [695], [696], [697], [698], [699], [700], [701], [702], [703], [704], [705], [706], [707], [708], [709], [710], [711], [712], [713], [714], [715], [716], [717], [718], [719], [711], [712], [713], [714], [715], [716], [717], [718], [719], [720], [721], [722], [723], [724], [725], [726], [727], [728], [729], [721], [722], [723], [724], [725], [726], [727], [728], [729], [730], [731], [732], [733], [734], [735], [736], [737], [738], [739], [731], [732], [733], [734], [735], [736], [737], [738], [739], [740], [741], [742], [743], [744], [745], [746], [747], [748], [749], [741], [742], [743], [744], [745], [746], [747], [748], [749], [750], [751], [752], [753], [754], [755], [756], [757], [758], [759], [760], [761], [762], [763], [764], [765], [766], [767], [768], [769], [761], [762], [763], [764], [765], [766], [767], [768], [769], [770], [771], [772], [773], [774], [775], [776], [777], [778], [779], [771], [772], [773], [774], [775], [776], [777], [778], [779], [780], [781], [782], [783], [784], [785], [786], [787], [788], [789], [781], [782], [783], [784], [785], [786], [787], [788], [789], [790], [791], [792], [793], [794], [795], [796], [797], [798], [799], [791], [792], [793], [794], [795], [796], [797], [798], [799], [800], [801], [802], [803], [804], [805], [806], [807], [808], [809], [810], [811], [812], [813], [814], [815], [816], [817], [818], [819], [811], [812], [813], [814], [815], [816], [817], [818], [819], [820], [821], [822], [823], [824], [825], [826], [827], [828], [829], [821], [822], [823], [824], [825], [826], [827], [828], [829], [830], [831], [832], [833], [834], [835], [836], [837], [838], [839], [831], [832], [833], [834], [835], [836], [837], [838], [839], [840], [841], [842], [843], [844], [845], [846], [847], [848], [849], [841], [842], [843], [844], [845], [846], [847], [848], [849], [850], [851], [852], [853], [854], [855], [856], [857], [858], [859], [860], [861], [862], [863], [864], [865], [866], [867], [868], [869], [861], [862], [863], [864], [865], [866], [867], [868], [869], [870], [871], [872], [873], [874], [875], [876], [877], [878], [879], [871], [872], [873], [874], [875], [876], [877], [878], [879], [880], [881], [882], [883], [884], [885], [886], [887], [888], [889], [881], [882], [883], [884], [885], [886], [887], [888], [889], [890], [891], [892], [893], [894], [895], [896], [897], [898], [899], [891], [892], [893], [894], [895], [896], [897], [898], [899], [900], [901], [902], [903], [904], [905], [906], [907], [908], [909], [910], [911], [912], [913], [914], [915], [916], [917], [918], [919], [911], [912], [913], [914], [915], [916], [917], [918], [919], [920], [921], [922], [923], [924], [925], [926], [927], [928], [929], [921], [922], [923], [924], [925], [926], [927], [928], [929], [930], [931], [932], [933], [934], [935], [936], [937], [938], [939], [931], [932], [933], [934], [935], [936], [937], [938], [939], [940], [941], [942], [943], [944], [945], [946], [947], [948], [949], [941], [942], [943], [944], [945], [946], [947], [948], [949], [950], [951], [952], [953], [954], [955], [956], [957], [958], [959], [960], [961], [962], [963], [964], [965], [966], [967], [968], [969], [961], [962], [963], [964], [965], [966], [967], [968], [969], [970], [971], [972], [973], [974], [975], [976], [977], [978], [979], [971], [972], [973], [974], [975], [976], [977], [978], [979], [980], [981], [982], [983], [984], [985], [986], [987], [988], [989], [981], [982], [983], [984], [985], [986], [987], [988], [989], [990], [991], [992], [993], [994], [995], [996], [997], [998], [999], [991], [992], [993], [994], [995], [996], [997], [998], [999], [1000], [1001], [1002], [1003], [1004], [1005], [1006], [1007], [1008], [1009], [1010], [1011], [1012], [1013], [1014], [1015], [1016], [1017], [1018], [1019], [1011], [1012], [1013], [1014], [1015], [1016], [1017], [1018], [1019], [1020], [1021], [1022], [1023], [1024], [1025], [1026], [1027], [1028], [1029], [1021], [1022], [1023], [1024], [1025], [1026], [1027], [1028], [1029], [1030], [1031], [1032], [1033], [1034], [1035], [1036], [1037], [1038], [1039], [1031], [1032], [1033], [1034], [1035], [1036], [1037], [1038], [1039], [1040], [1041], [1042], [1043], [1044], [1045], [1046], [1047], [1048], [1049], [1041], [1042], [1043], [1044], [1045], [1046], [1047], [1048], [1049], [1050], [1051], [1052], [1053], [1054], [1055], [1056], [1057], [1058], [1059], [1060], [1061], [1062], [1063], [1064], [1065], [1066], [1067], [1068], [1069], [1061], [1062], [1063], [1064], [1065], [1066], [1067], [1068], [1069], [1070], [1071], [1072], [1073], [1074], [1075], [1076], [1077], [1078], [1079], [1071], [1072], [1073], [1074], [1075], [1076], [1077], [1078], [1079], [1080], [1081], [1082], [1083], [1084], [1085], [1086], [1087], [1088], [1089], [1081], [1082], [1083], [1084], [1085], [1086], [1087], [1088], [1089], [1090], [1091], [1092], [1093], [1094], [1095], [1096], [1097], [1098], [1099], [1091], [1092], [1093], [1094], [1095], [1096], [1097], [1098], [1099], [1100], [1101], [1102], [1103], [1104], [1105], [1106], [1107], [1108], [1109], [1110], [1111], [1112], [1113], [1114], [1115], [1116], [1117], [1118], [1119], [1111], [1112], [1113], [1114], [1115], [1116], [1117], [1118], [1119], [1120], [1121], [1122], [1123], [1124], [1125], [1126], [1127], [1128], [1129], [1121], [1122], [1123], [1124], [1125], [1126], [1127], [1128], [1129], [1130], [1131], [1132], [1133], [1134], [1135], [1136], [1137], [1138], [1139], [1131], [1132], [1133], [1134], [1135], [1136], [1137], [1138], [1139], [1140], [1141], [1142], [1143], [1144], [1145], [1146], [1147], [1148], [1149], [1141], [1142], [1143], [1144], [1145], [1146], [1147], [1148], [1149], [1150], [1151], [1152], [1153], [1154], [1155], [1156], [1157], [1158], [1159], [1151], [1152], [1153], [1154], [1155], [1156], [1157], [1158], [1159], [1160], [1161], [1162], [1163], [1164], [1165], [1166], [1167], [1168], [1169], [1161], [1162], [1163], [1164], [1165], [1166], [1167], [1168], [1169], [1170], [1171], [1172], [1173], [1174], [1175], [1176], [1177], [1178], [1179], [1171], [1172], [1173], [1174], [1175], [1176], [1177], [1178], [1179], [1180], [1181], [1182], [1183], [1184], [1185], [1186], [1187], [1188], [1189], [1181], [1182], [1183], [1184], [1185], [1186], [1187], [1188], [1189], [1190], [1191], [1192], [1193], [1194], [1195], [1196], [1197], [1198], [1199], [1191], [1192], [1193], [1194], [1195], [1196], [1197], [1198], [1199], [1200], [1201], [1202], [1203], [1204], [1205], [1206], [1207], [1208], [1209], [1210], [1211], [1212], [1213], [1214], [1215], [1216], [1217], [1218], [1219], [1211], [1212], [1213], [1214], [1215], [1216], [1217], [1218], [1219], [1220], [1221], [1222], [1223], [1224], [1225], [1226], [1227], [1228], [1229], [1221], [1222], [1223], [1224], [1225], [1226], [1227], [1228], [1229], [1230], [1231], [1232], [1233], [1234], [1235], [1236], [1237], [1238], [1239], [1231], [1232], [1233], [1234], [1235], [1236], [1237], [1238], [1239], [1240], [1241], [1242], [1243], [1244], [1245], [1246], [1247], [1248], [1249], [1241], [1242], [1243], [1244], [1245], [1246], [1247], [1248], [1249], [1250], [1251], [1252], [1253], [1254], [1255], [1256], [1257], [1258], [1259], [1251], [1252], [1253], [1254], [1255], [1256], [1257], [1258], [1259], [1260], [1261], [1262], [1263], [1264], [1265], [1266], [1267], [1268], [1269], [1261], [1262], [1263], [1264], [1265], [1266], [1267], [1268], [1269], [1270], [1271], [1272], [1273], [1274], [1275], [1276], [1277], [1278], [1279], [1271], [1272], [1273], [1274], [1275], [1276], [1277], [1278], [1279], [1280], [1281], [1282], [1283], [1284], [1285], [1286], [1287], [1288], [1289], [1281], [1282], [1283], [1284], [1285], [1286], [1287], [1288], [1289], [1290], [1291], [1292], [1293], [1294], [1295], [1296], [1297], [1298], [1299], [1291], [1292], [1293], [1294], [1295], [1296], [1297], [1298], [1299], [1300], [1301], [1302], [1303], [1304], [1305], [1306], [1307], [1308], [1309], [1310], [1311], [1312], [1313], [1314], [1315], [1316], [1317], [1318], [1319], [1311], [1312], [1313], [1314], [1315], [1316], [1317], [1318], [1319], [1320], [1321], [1322], [1323], [1324], [1325], [1326], [1327], [1328], [1329], [1321], [1322], [1323], [1324], [1325], [1326], [1327], [1328], [1329], [1330], [1331], [1332], [1333], [1334], [1335], [1336], [1337], [1338], [1339], [1331], [1332], [1333], [1334], [1335], [1336], [1337], [1338], [1339], [1
```



EXPERIMENT - 09

Aim: Implementation of Association Rule Mining algorithm (Apriori)

Theory:

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

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

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

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

Components of the Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Support

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

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

$$= 400/4000 = 10 \text{ percent.}$$

Confidence

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

Hence,

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

$$= 200/400$$

$$= 50 \text{ percent.}$$

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

Lift

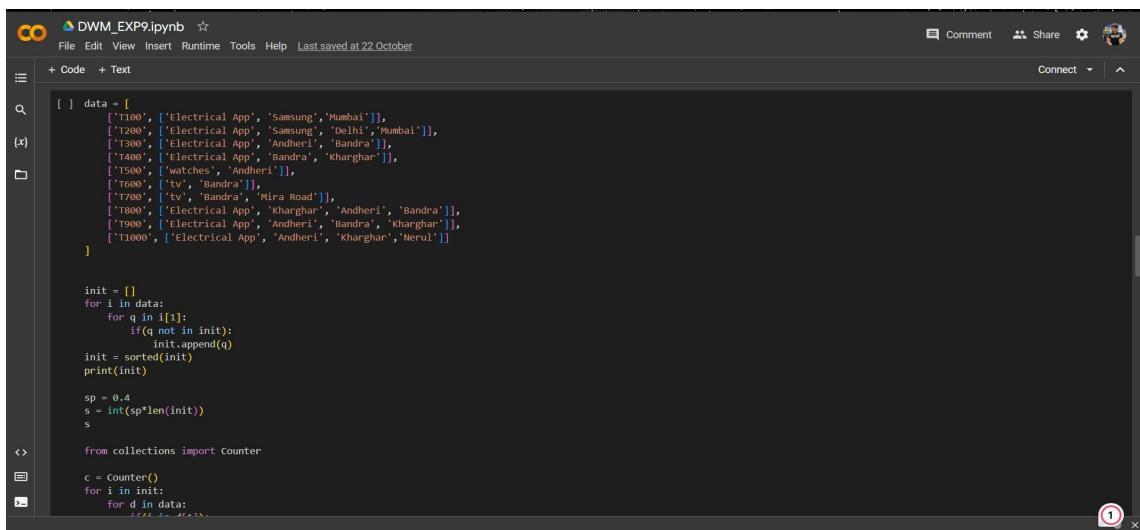
Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

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

$$= 50/10 = 5$$

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

Output:



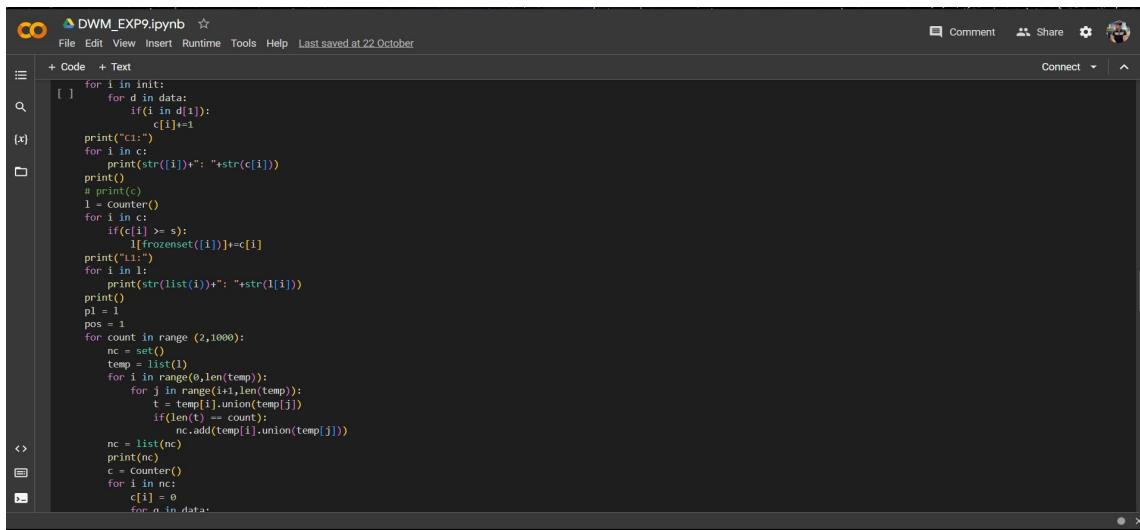
```
[ ] data = [
    ['T100', ['Electrical App', 'Samsung', 'Mumbai']],
    ['T200', ['Electrical App', 'Samsung', 'Delhi', 'Mumbai']],
    ['T300', ['Electrical App', 'Andheri', 'Bandra']],
    ['T400', ['Electrical App', 'Bandra', 'Kharhgar']],
    ['T500', ['watches', 'Andheri']],
    ['T600', ['tv', 'Bandra']],
    ['T700', ['tv', 'Bandra', 'Mira Road']],
    ['T800', ['Electrical App', 'Kharhgar', 'Andheri', 'Bandra']],
    ['T900', ['Electrical App', 'Andheri', 'Bandra', 'Kharhgar']],
    ['T1000', ['Electrical App', 'Andheri', 'Kharhgar', 'Nerul']]
]

init = []
for i in data:
    for q in i[1]:
        if(q not in init):
            init.append(q)
init = sorted(init)
print(init)

sp = 0.4
s = int(sp*len(init))
s

<>
from collections import Counter

c = Counter()
for i in init:
    for d in data:
        for n in d[1]:
```



```
[ ] for i in init:
    for d in data:
        if(i in d[1]):
            c[i]+=1
(x) print("c1:")
for i in c:
    print(str([i])+" : "+str(c[i]))
print()
a.print(c)
l = Counter()
for i in c:
    if(c[i] >= s):
        l[frozenset([i])]+=c[i]
print("l1:")
for i in l:
    print(str(list(i))+" : "+str(l[i]))
print()
pl = 1
pos = 1
for count in range (2,1000):
    nc = set()
    temp = list(l)
    for i in range(0,len(temp)):
        for j in range(i+1,len(temp)):
            t = temp[i].union(temp[j])
            if(len(t) == count):
                nc.add(temp[i].union(temp[j]))
    nc = list(nc)
    print(nc)
    c = Counter()
    for i in nc:
        c[i] = 0
        for n in data:
```

```

File Edit View Insert Runtime Tools Help Last saved at 22 October
+ Code + Text
[ ] for q in data:
    temp = set(q[1])
    if(1.issubset(temp)):
        c[1]+=1
print("C"+str(count)+"::")
for i in c:
    print(str(list(i))+":: "+str(c[i]))
print()
l = Counter()
for i in c:
    if(c[i] >= s):
        l[i]+<c[i]
print("L"+str(count)+"::")
for i in l:
    print(str(list(i))+":: "+str(l[i]))
print()
if(len(l) == 0):
    break
pl = 1
pos = count
print("Result:")
print("P"+str(pos)+"::")
for i in pl:
    print(str(list(i))+":: "+str(pl[i]))
print()
# print(pl)
confidence=60

from itertools import combinations
for l in pl:
    c = [frozenset(q) for q in combinations(l,len(l)-1)]

```

```

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

```

```

DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
+ Code + Text
[ ] print("choosing:", end=' ')
for a in c:
    b = 1-a
    ab = 1
    sa = 0
    sb = 0
    for q in data:
        temp = set(q[1])
        if(a.issubset(temp)):
            sa+=1
        if(b.issubset(temp)):
            sb+=1
        if(ab.issubset(temp)):
            sab+=1
    temp = sab/sa*100
    if(temp >= confidence):
        print(curr, end = ' ')
    curr += 1
    temp = sab/sb*100
    if(temp >= confidence):
        print(curr, end = ' ')
    curr += 1
print()
print()

C1: ['Andheri': 5, ['Bandra']: 6, ['Delhi']: 1, ['Electrical App']: 7, ['Kharhgar']: 4

```

```

DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
+ Code + Text
[ ] [ ] ['Andheri', 'Bandra', 'Delhi', 'Electrical App', 'Kharhgar', 'Mira Road', 'Mumbai', 'Nerul', 'Samsung', 'tv', 'watches']
C1: ['Andheri']: 5
['Bandra']: 6
['Delhi']: 1
['Electrical App']: 7
['Kharhgar']: 4
['Mira Road']: 1
['Mumbai']: 2
['Nerul']: 1
['Samsung']: 2
['tv']: 2
['watches']: 1

L1: ['Andheri']: 5
['Bandra']: 6
['Electrical App']: 7
['Kharhgar']: 4

[frozenset({'Kharhgar', 'Andheri'}), frozenset({'Bandra', 'Kharhgar'}), frozenset({'Bandra', 'Andheri'}), frozenset({'Kharhgar', 'Electrical App'}), frozenset({'Electrical App', 'Andheri'})]
C2: ['Kharhgar', 'Andheri']: 3
['Bandra', 'Kharhgar']: 3
['Bandra', 'Andheri']: 3
['Kharhgar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4

L2: ['Kharhgar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4

[frozenset({'Bandra', 'Electrical App', 'Andheri'}), frozenset({'Kharhgar', 'Electrical App', 'Andheri'}), frozenset({'Bandra', 'Kharhgar', 'Electrical App'})]

```

```

DWM_EXP9.ipynb
File Edit View Insert Runtime Tools Help Last saved at 22 October
+ Code + Text
[ ] [ ] ['Kharhgar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4

[[frozenset({'Bandra', 'Electrical App', 'Andheri'}), frozenset({'Kharhgar', 'Electrical App', 'Andheri'}), frozenset({'Bandra', 'Kharhgar', 'Electrical App'})]]
C3: ['Bandra', 'Electrical App', 'Andheri']: 3
['Kharhgar', 'Electrical App', 'Andheri']: 3
['Bandra', 'Kharhgar', 'Electrical App']: 3

L3:
Result:
L2:
['Kharhgar', 'Electrical App']: 4
['Electrical App', 'Andheri']: 4
['Bandra', 'Electrical App']: 4

['Kharhgar'] -> ['Electrical App'] = 100.0%
['Electrical App'] -> ['Kharhgar'] = 57.14285714285714%
['Electrical App'] -> ['Kharhgar'] = 57.14285714285714%
['Kharhgar'] -> ['Electrical App'] = 100.0%
choosing: 1 4

['Electrical App'] -> ['Andheri'] = 57.14285714285714%
['Andheri'] -> ['Electrical App'] = 80.0%
['Andheri'] -> ['Electrical App'] = 80.0%
['Electrical App'] -> ['Andheri'] = 57.14285714285714%
choosing: 2 3

['Bandra'] -> ['Electrical App'] = 66.66666666666666%
['Electrical App'] -> ['Bandra'] = 57.14285714285714%
['Electrical App'] -> ['Bandra'] = 57.14285714285714%
['Bandra'] -> ['Electrical App'] = 66.66666666666666%
choosing: 1 4

```

EXPERIMENT - 10

Aim: Implementation of Page rank/HITS algorithm

Theory:

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

1. PageRank Algorithm:

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

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

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

$$\begin{aligned} PR(P) = & (1 - d) + d * (PR(Q1)/L(Q1) + PR(Q2)/L(Q2) + \dots + \\ & PR(Qn)/L(Qn)) \end{aligned}$$

Where:

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

- $L(Q_1), L(Q_2), \dots, L(Q_n)$ are the number of outbound links on pages Q_1, Q_2, \dots, Q_n .

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

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

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

Mathematically, the HITS algorithm works as follows:

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

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

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

Output:

DWM_EXP10.ipynb

```

import networkx as nx
import matplotlib.pyplot as plt

def hits_algorithm(graph, max_iter=10, tol=1e-6):
    hubs = {node: 1 for node in graph.nodes}
    authorities = {node: 1 for node in graph.nodes}

    for _ in range(max_iter):
        new_hubs = {}
        new_authorities = {}

        for node in graph.nodes:
            new_authority = sum(hubs[neighbor] for neighbor in graph.predecessors(node))
            new_authorities[node] = new_authority

        for node in graph.nodes:
            new_hub = sum(authorities[neighbor] for neighbor in graph.successors(node))
            new_hubs[node] = new_hub

        norm = max(max(new_hubs.values()), max(new_authorities.values()))
        hubs = {node: score / norm for node, score in new_hubs.items()}
        authorities = {node: score / norm for node, score in new_authorities.items()}

        if all(abs(hubs[node] - new_hubs[node]) < tol for node in graph.nodes) and \
           all(abs(authorities[node] - new_authorities[node]) < tol for node in graph.nodes):
            break

    return hubs, authorities

graph = nx.DiGraph()

```

DWM_EXP10.ipynb

```

graph = nx.DiGraph()

graph.add_edges_from([('A', 'B'), ('B', 'C'), ('B', 'E'), ('C', 'E'), ('C', 'A'),
                     ('D', 'E'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
                     ('F', 'C'), ('F', 'H'), ('G', 'A'),
                     ('G', 'C'), ('H', 'A')])

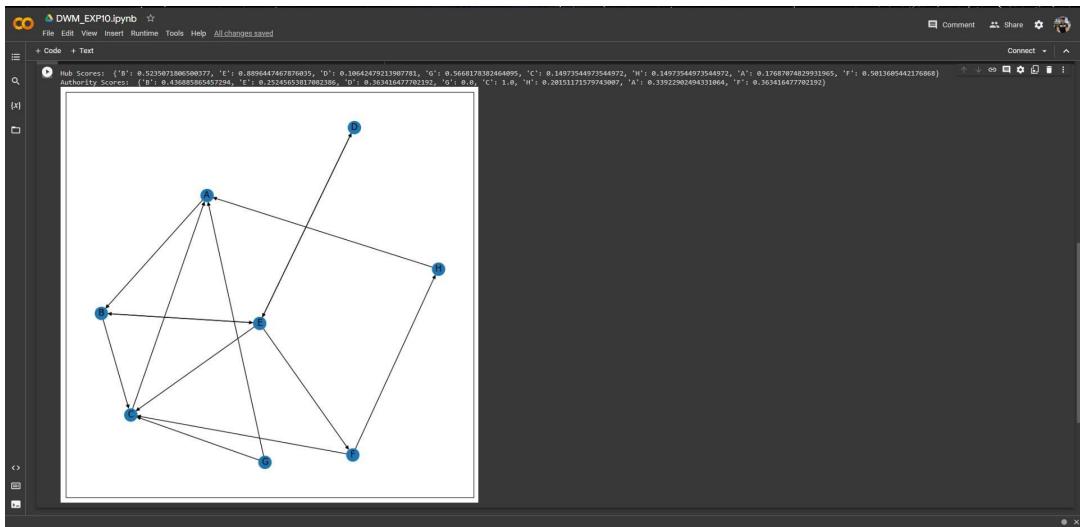
plt.figure(figsize=(10, 10))
nx.draw_networkx(graph, with_labels=True)

hubs, authorities = hits_algorithm(graph, max_iter=10)

print("Hub Scores: ", hubs)
print("Authority scores: ", authorities)

```

Hub Scores: {'B': 0.5235071806500377, 'E': 0.8896447467876035, 'D': 0.10642479213907781, 'G': 0.5668178382464095, 'C': 0.14973544973544972, 'H': 0.14973544973544972, 'A': 0.1768707482, 'F': 0.3634164776}
Authority Scores: {'B': 0.436885865457294, 'E': 0.2524565381782386, 'D': 0.36341647762192, 'G': 0.0, 'C': 1.0, 'H': 0.20151171579743007, 'A': 0.33922902494331064, 'F': 0.3634164776}



Assignment 1

What is Metadata? Explain data warehouse with Example.

- Data warehouse metadata is a piece of information stored in one or more special-purpose metadata repository that include:
 - 1) Information about data warehouse, their structure and their location.
 - 2) Information about refreshment of warehouse cleanup.
 - 3) Information regarding characteristics of components.
- Types of Metadata :
 - 1) Operational Metadata
 -
 - In an enterprise, data for the data warehouse comes from various operational systems.
 - Different source systems contain different data structures having different field lengths and data types.
 - So the information of operational data source is given by operational metadata.

2) Extraction & Transformation Metadata

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

3) End-user Metadata

- ⇒ This particular category provide the end user the flexibility of looking for information in their own way.
- For e.g.:

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

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

~~23/10/2020~~

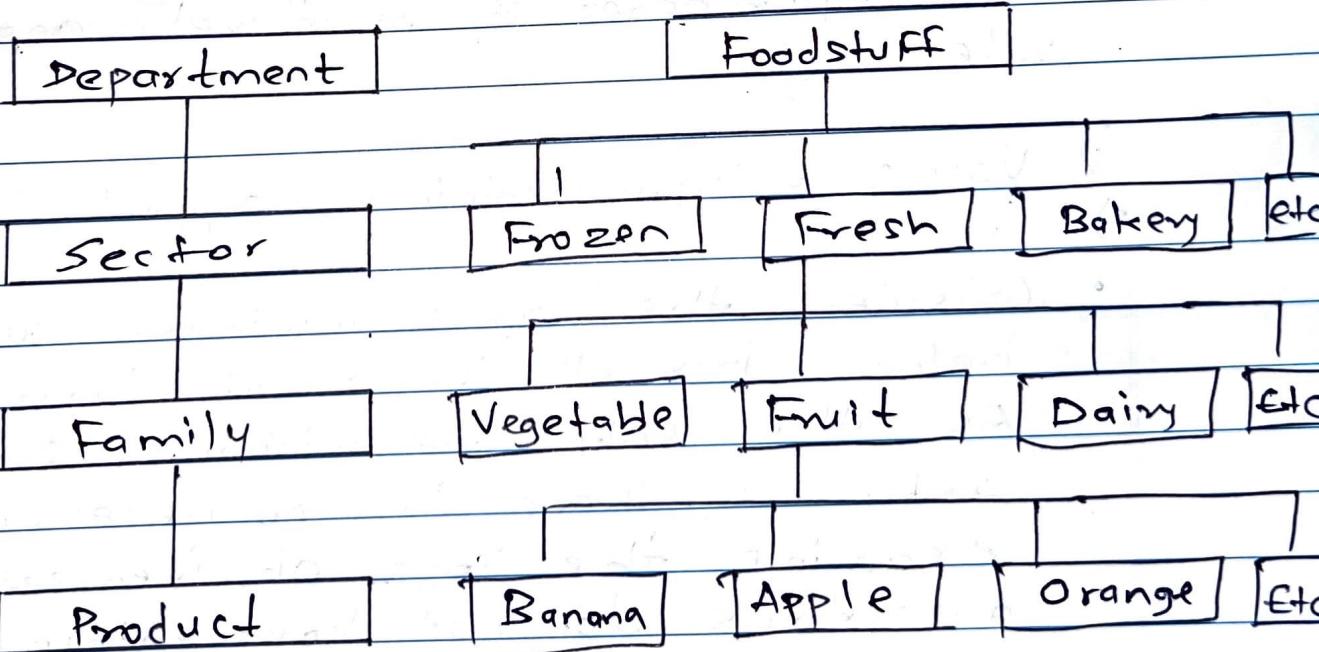
Assignment 2

Q.1 Write short note on

a) Multilevel Association Rules and Multi-dimensional association rules

→

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



(Hierarchy of Concept)

- Support and confidence of multilevel association rules:

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

- Two approaches of multilevel association rules:

1) Using uniform minimum support for all levels

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

level 1

min-support = 5%

Milk
(support = 10%)

Level 2

Cheese

min-support = 5%

(support = 6%)

Butter

(support = 4%)

- 2) Using reduced minimum support at lowest level



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

Level 2

min-sup = 5%

Milk
[support = 10%]

Level 2

min-sup = 3%

Cheese

(support = 6%)

Butter

(support = 9%)

- Multidimensional Association Rules:

- The rules contains two or more dimensions
 - predicates.
- Inter-dimension association rules
 - ⇒ The rule doesn't have any repeated predicate.
- Hybrid-dimension association rules
 - ⇒ The rule have many occurrences of same predicate , i.e. buys.

gender (x, "Male") \wedge buys (x, "TV")
⇒ buys (x, "DVD").

- Techniques of Mining MD Associations:

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

2) Quantitative association rules:



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

A quantitative 1 A quantitative 2 ⇒ Categorical

3) Distance-based association rules



- This is a dynamic discretization process that considers the distance between data points.
- This mining process has only two steps:

- Perform clustering to find the interval of attributes involved.
- Obtain association rules by searching for groups of clusters that occur together.

b) Web Usage Mining

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

- The number of accesses to the server.
 - The times or time intervals of visits.
 - The domain names and the URLs of users of the Web Server.
- A very little or no analysis is provided by these tools of the data relationships among the accessed files and the directories within web spaces.
- Q. - The tools for discovery and analysis of patterns may be classified into following two categories :
- Pattern Discovery Tools
 - Pattern Analysis Tools.
- Web server, web proxies and client applications can quite easily capture web usage data.
- Web server log : It is a file that is created by the server to record all the activities it performs.
- For example, when a user enters URL into the browser's address bar or requests by clicking on a link.

Q1
13/10/29