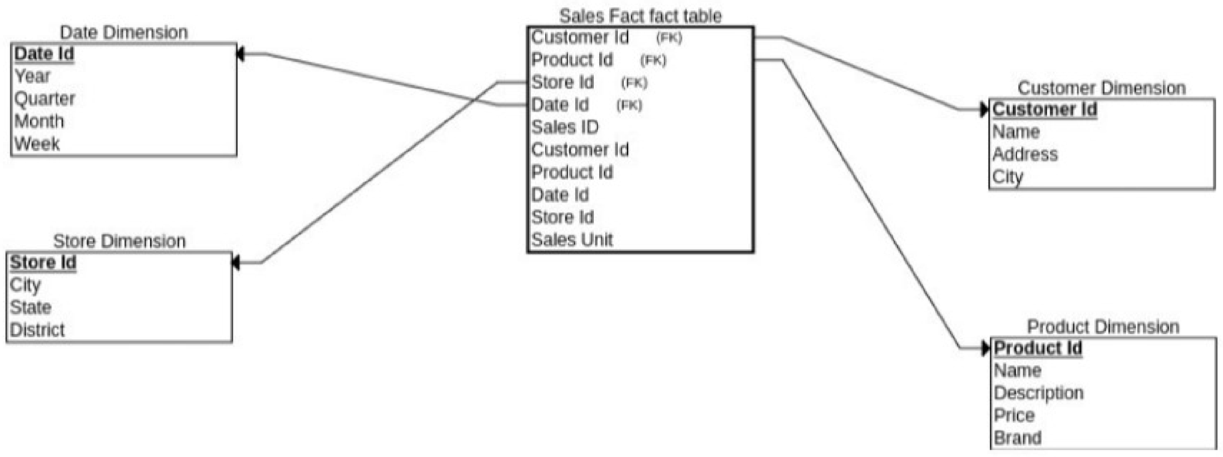
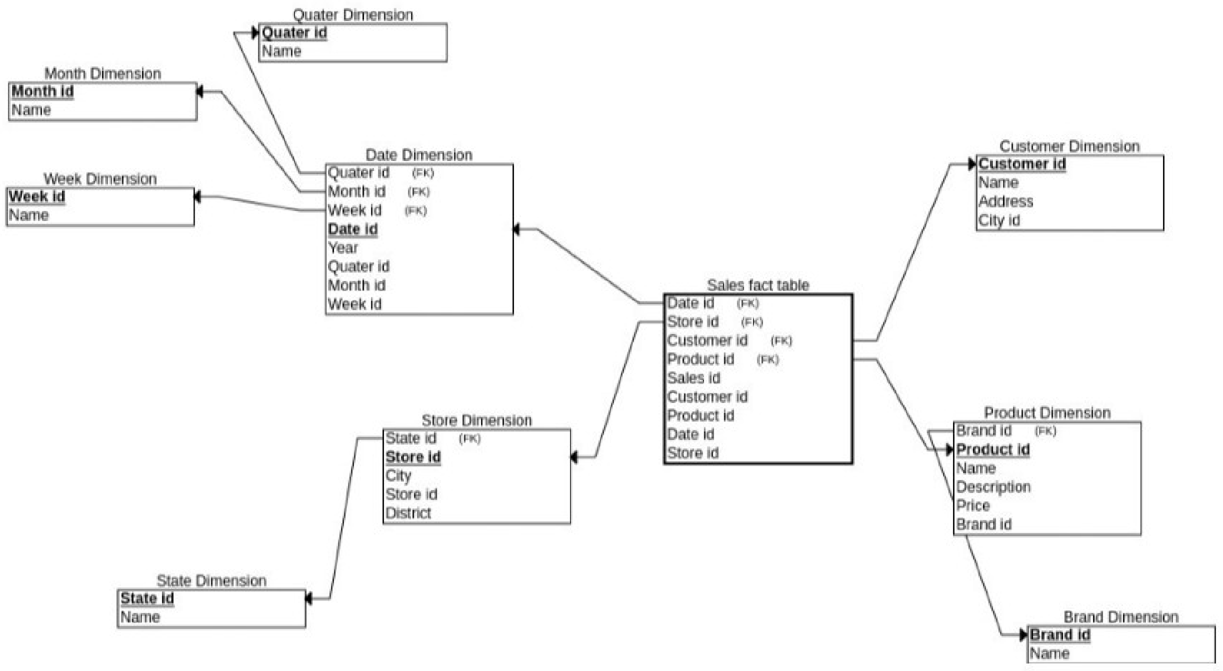
EXP 01: One case study on building Data warehouse/Data Mart
Write Detailed Problem statement and design dimensional modelling (creation of star and snowflake schema)





EXP 02: Implementation of all dimension table and fact table based on experiment 1 case study
i) Identifying the source tables and populating sample data
ii) Implementation of dimensional data model i.e. Star schema, Snowflake schema and Fact Constellation schema.
Perform following for the above warehouse:
1.Maximum/minimum sale in first quarter 2. Maximum/minimum sale of item “skirts” throughout the year 3. Maximum/minimum sale of item throughout the year 4. Maximum/minimum sale during the second & third quarter w.r.t. location and item 5. List out the items in increasing order w.r.t. sales 6. List out the suppliers who supply maximum number of “jeans” during year 7. Find out the

customer who purchase maximum number of items and also find out all the details of customer along with region.

Create the Database:

```
CREATE DATABASE clothing_store;

USE clothing_store;
```

Create Tables:

```
CREATE TABLE items (

    item_id INT AUTO_INCREMENT PRIMARY KEY,

    item_name VARCHAR(100),

    price DECIMAL(10, 2)

);



CREATE TABLE suppliers (

    supplier_id INT AUTO_INCREMENT PRIMARY KEY,

    supplier_name VARCHAR(100)

);



CREATE TABLE sales (

    sale_id INT AUTO_INCREMENT PRIMARY KEY,

    item_id INT,

    supplier_id INT,

    quantity INT,

    sale_date DATE,

    location VARCHAR(100),

    FOREIGN KEY (item_id) REFERENCES items(item_id),

    FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)

);



CREATE TABLE customers (

    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_name VARCHAR(100),

    region VARCHAR(100)

);


CREATE TABLE purchases (

    purchase_id INT AUTO_INCREMENT PRIMARY KEY,

    customer_id INT,

    sale_id INT,

    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),

    FOREIGN KEY (sale_id) REFERENCES sales(sale_id)

);
```

Insert Sample Data

```
-- Insert items

INSERT INTO items (item_name, price) VALUES ('skirts', 25.00), ('jeans', 30.00),
('shirts', 20.00);


-- Insert suppliers

INSERT INTO suppliers (supplier_name) VALUES ('Supplier A'), ('Supplier B');


-- Insert sales

INSERT INTO sales (item_id, supplier_id, quantity, sale_date, location) VALUES

(1, 1, 10, '2024-01-15', 'New York'),

(1, 1, 5, '2024-02-10', 'Los Angeles'),

(2, 1, 20, '2024-03-20', 'Chicago'),

(2, 2, 15, '2024-04-18', 'New York'),

(3, 1, 30, '2024-05-25', 'Los Angeles'),

(1, 2, 25, '2024-07-15', 'Chicago'),

(2, 1, 10, '2024-08-12', 'New York');
```

```
-- Insert customers

INSERT INTO customers (customer_name, region) VALUES ('Alice', 'East'), ('Bob', 'West');



-- Insert purchases

INSERT INTO purchases (customer_id, sale_id) VALUES

(1, 1), (1, 2), (2, 3);
```

1. **Maximum/Minimum Sale in the First Quarter**:

```
SELECT MAX(quantity) AS max_sale, MIN(quantity) AS min_sale

FROM sales

WHERE sale_date BETWEEN '2024-01-01' AND '2024-03-31';
```

2. **Maximum/Minimum Sale of Item "skirts" Throughout the Year**:

```
SELECT MAX(quantity) AS max_sale, MIN(quantity) AS min_sale

FROM sales

WHERE item_id = (SELECT item_id FROM items WHERE item_name = 'skirts');
```

3. **Maximum/Minimum Sale of Item Throughout the Year**:

```
SELECT item_id, MAX(quantity) AS max_sale, MIN(quantity) AS min_sale

FROM sales

GROUP BY item_id;
```

4. **Maximum/Minimum Sale During the Second & Third Quarter w.r.t. Location and Item**:

```
SELECT location, item_id, MAX(quantity) AS max_sale, MIN(quantity) AS min_sale

FROM sales

WHERE sale_date BETWEEN '2024-04-01' AND '2024-09-30'

GROUP BY location, item_id;
```

5. **List Out the Items in Increasing Order w.r.t. Sales**:

```
SELECT item_id, SUM(quantity) AS total_sales

FROM sales

GROUP BY item_id

ORDER BY total_sales ASC;
```

6. **List Out the Suppliers Who Supply Maximum Number of "jeans" During the Year**:

```
SELECT supplier_id, SUM(quantity) AS total_jeans

FROM sales

WHERE item_id = (SELECT item_id FROM items WHERE item_name = 'jeans')

GROUP BY supplier_id

ORDER BY total_jeans DESC

LIMIT 1;
```

7. **Find Out the Customer Who Purchases Maximum Number of Items**:

```
SELECT c.customer_id, c.customer_name, c.region, COUNT(p.purchase_id) AS
total_items

FROM customers c

JOIN purchases p ON c.customer_id = p.customer_id

GROUP BY c.customer_id

ORDER BY total_items DESC

LIMIT 1;
```

EXP 03: Wwrite SQL DML queries to demonstrate

following operations

SLICE

DICE

ROLL UP

DRILL DOWN

1. SLICING - **Slicing** refers to selecting a single dimension from a multi-dimensional data set.

```
SELECT s.sale_id, s.quantity, s.sale_date, s.location

FROM sales s

JOIN items i ON s.item_id = i.item_id

WHERE i.item_name = 'jeans';
```

2. DICING - **Dicing** refers to selecting two or more dimensions from a data set.

```
SELECT s.sale_id, s.quantity, s.sale_date

FROM sales s

JOIN items i ON s.item_id = i.item_id

WHERE i.item_name IN ('jeans', 'skirts') AND s.location = 'New York';
```

3. ROLLING UP - **Rolling Up** refers to aggregating data along a dimension, reducing the detail level.

```
SELECT i.item_name, SUM(s.quantity) AS total_sales

FROM sales s

JOIN items i ON s.item_id = i.item_id

GROUP BY i.item_name;
```

4. DRILLING DOWN - **Drilling Down** refers to breaking down data into more detailed levels.

```
SELECT MONTH(s.sale_date) AS sale_month, SUM(s.quantity) AS total_sales

FROM sales s

JOIN items i ON s.item_id = i.item_id

WHERE i.item_name = 'jeans' AND YEAR(s.sale_date) = 2024

GROUP BY MONTH(s.sale_date);
```

EXP 04: Installation and study of weka tool

```
sudo apt-get install weka

weka
```

1. save the file in .csv format

2. go to weka and then click on open file

3. after clicking on open file select the file in .csv format

4. then you will be able to see the output

EXP 05: