Assignment:- 05
Operators

Instructions:

Please share your answers filled in line in the Word document. Submit code separately wherever applicable.

Please ensure you update all the details:

Name: ULLI VENKATA SAI KUMAR Batch ID: 04072024HYD10AM

Topic: Introduction to Database

1. Create a Supermart_DB with the tables created from the datasets shared (Customer.csv, Sales.csv and Product.csv files)

    a. Create a new database in your database management system, and name it Supermart_DB.
    ```
    CREATE DATABASE SUPERMART_DB;
    USE SUPERMART_DB;
    ```
    b. Create a new table called "customers" in the Supermart_DB database
    ```
    CREATE TABLE CUSTOMERS (
        CUSTOMER_ID VARCHAR(100) PRIMARY KEY,
        CUSTOMER_NAME VARCHAR(25),
        segment VARCHAR(30),
        Age INT,
        Country VARCHAR(30),
        region VARCHAR(30),
        city VARCHAR(30),
        state VARCHAR(30),
        postal_code VARCHAR(30)
    );
    ```
    c. Load the data from the Customer.csv file into the customers table
    d. Create a new table called "products" in the Supermart_DB database
    ```
    CREATE TABLE PRODUCTS (
        PRODUCT_ID VARCHAR(30) PRIMARY KEY,
        Category VARCHAR(20),
        Sub_category VARCHAR(30),
        product_name VARCHAR(400)
    );
    ```
    e. Load the data from the Product.csv file into the products table

    f. Create a new table called "sales" in the Supermart_DB database
    ```
    CREATE TABLE SALES (
        order_line INT,
        order_id VARCHAR(20),
        order_date VARCHAR(30),
        ship_date VARCHAR(30),
        ship_mode VARCHAR(40),
    ```

Assignment:- 05
Operators

```
    customer_id VARCHAR(30),
    product_id VARCHAR(30),
    sales INT,
    quantity INT,
    discount INT,
    profit INT,
    PRIMARY KEY (order_id, order_line),
  FOREIGN KEY (customer_id) REFERENCES PRODUCTS(PRODUCT_ID)
    FOREIGN KEY (product_id) REFERENCES PRODUCTS(PRODUCT_ID)
);
```

g. Load the data from the Sales.csv file into the sales table

SELECTION OPERATORS:- (FILTERING):- in, like, between

Note: use products, customers and sales table

1. Define the relationship between the tables using constraints/keys.

   Step-by-Step Relationship Definition

   Define Primary Keys (PK) for each table.

   In CUSTOMERS, the primary key is likely CUSTOMER_ID.

   In PRODUCTS, the primary key is PRODUCT_ID.

   In SALES, there's no clear single field that acts as a unique identifier for each row. Therefore, a composite primary key of order_id and order_line might be appropriate.

   Define Foreign Keys (FK) to establish relationships between the tables.

   The CUSTOMER_ID in SALES should reference the CUSTOMER_ID in CUSTOMERS.

   The PRODUCT_ID in SALES should reference the PRODUCT_ID in PRODUCTS.

   Explanation of the Relationships

   CUSTOMERS → SALES:

   The SALES table has a foreign key CUSTOMER_ID referencing the CUSTOMER_ID in the CUSTOMERS table. This creates a one-to-many relationship: one customer can have many sales.

   PRODUCTS → SALES:

   The SALES table also has a foreign key PRODUCT_ID referencing the PRODUCT_ID in the PRODUCTS table. This creates another one-to-many relationship: one product can be associated with many sales.

   Primary Keys:

   CUSTOMERS: The primary key is CUSTOMER_ID, which ensures each customer is unique.

Assignment:- 05
Operators

PRODUCTS: The primary key is PRODUCT_ID, ensuring each product is unique.

SALES: The composite primary key order_id + order_line ensures that each combination of order and line is unique within the SALES table.

Constraints Overview:

Primary Key on CUSTOMER_ID and PRODUCT_ID ensures the uniqueness of each customer and product.

Foreign Key constraints in the SALES table ensure that customer_id and product_id values in SALES must exist in the CUSTOMERS and PRODUCTS tables respectively. This ensures referential integrity across the tables.

2. In the database Supermart _DB, find the following:

   a. Get the list of all the cities where the region is north or east without any duplicates using the IN statement.

      - SELECT DISTINCT city FROM CUSTOMERS WHERE region IN ('North', 'East');

   b. Get the list of all orders where the ‘sales’ value is between 100 and 500 using the BETWEEN operator.

      - SELECT * FROM SALES WHERE sales BETWEEN 100 AND 500;

   c. Get the list of customers whose last name contains only 4 characters using LIKE.

      - SELECT * FROM CUSTOMERS WHERE CUSTOMER_NAME LIKE '____';

SELECTION OPERATORS:- ordering

1. Retrieve all orders where the ‘discount’ value is greater than zero ordered in descending order basis ‘discount’ value

   - SELECT * FROM SALES WHERE discount > 0 ORDER BY discount DESC;

2. Limit the number of results in the above query to the top 10.

   - SELECT * FROM SALES WHERE discount > 0 ORDER BY discount DESC LIMIT 10;

Aggregate operators:-

1. Find the sum of all ‘sales’ values.

   - SELECT SUM(sales) AS total_sales FROM SALES;

2. Find count of the number of customers in the north region with ages between 20 and 30

   - SELECT COUNT(*) AS customer_count FROM CUSTOMERS WHERE region = 'North' AND Age BETWEEN 20 AND 30;

Assignment:- 05
Operators

3. Find the average age of east region customers
    - SELECT AVG(Age) AS average_age FROM CUSTOMERS WHERE region = 'East';
4. Find the minimum and maximum aged customers from Philadelphia
    - SELECT MIN(Age) AS min_age, MAX(Age) AS max_age FROM CUSTOMERS WHERE city = 'Philadelphia';

GROUP BY OPERATORS:-

1. Create a display with the information below for each product ID.
    a. Total sales (in $) order by this column in descending
    b. Total sales quantity
    c. The number of orders
    d. Max Sales value
    e. Min Sales value
    f. Average sales value

```
SELECT
    product_id,
    SUM(sales) AS total_sales,
    SUM(quantity) AS total_quantity,
    COUNT(order_id) AS num_orders,
    MAX(sales) AS max_sales,
    MIN(sales) AS min_sales,
    AVG(sales) AS avg_sales
FROM SALES
GROUP BY product_id
ORDER BY total_sales DESC;
```

2. Get the list of product ID's where the quantity of product sold is greater than 10
    - SELECT product_id FROM SALES GROUP BY product_id HAVING SUM(quantity) > 10;