**Assignment:- 05**

**SELECTION COMMANDS:- (FILTERING):- in, like, between**

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

Sol: To create the Supermart\_DB with the tables from the datasets provided (Customer.csv, Sales.csv and Product.csv files), we can follow these steps:

1. Create a new database in your database management system, and name it Supermart\_DB.

Create database Supermart\_DB;

1. Create a new table called "customers" in the Supermart\_DB database

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

customer\_name CHAR(50),

customer\_email CHAR(50),

customer\_address CHAR(100),

customer\_phone CHAR(20)

);

1. Load the data from the Customer.csv file into the customers table

LOAD DATA INFILE '/path/to/Customer.csv'

INTO TABLE customers

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

1. Create a new table called "products" in the Supermart\_DB database

CREATE TABLE products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(50),

product\_category VARCHAR(50),

product\_price DECIMAL(10,2)

);

1. Load the data from the Product.csv file into the products table

LOAD DATA INFILE '/path/to/Product.csv'

INTO TABLE products

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

1. Create a new table called "sales" in the Supermart\_DB database

CREATE TABLE sales (

sale\_id INT PRIMARY KEY,

customer\_id INT,

product\_id INT,

sale\_date DATE,

sale\_quantity INT,

sale\_total DECIMAL(10,2),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

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

LOAD DATA INFILE '/path/to/Sales.csv'

INTO TABLE sales

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

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

In the Supermart\_DB, we have three tables: customers, products, and sales. To define the relationship between these tables using constraints/keys, we can use.

1. The customers table has a primary key column called **customer\_id**. We can define a foreign key constraint on the sales table to link the customer\_id column to the customer\_id column in the customers table. This will ensure that a sale can only be associated with a valid customer.

ALTER TABLE sales

ADD CONSTRAINT fk\_sales\_customer\_id

FOREIGN KEY (customer\_id)

REFERENCES customers(customer\_id);

1. The products table has a primary key column called **product\_id**. We can define a foreign key constraint on the sales table to link the product\_id column to the product\_id column in the products table. This will ensure that a sale can only be associated with a valid product.

ALTER TABLE sales

ADD CONSTRAINT fk\_sales\_product\_id

FOREIGN KEY (product\_id)

REFERENCES products(product\_id);

1. **In the database Supermart\_DB, find the following:**
2. Get the list of all the cities where the region is north or east without any duplicates using IN statement.

SELECT DISTINCT city

FROM customers

WHERE region IN ('North', 'East');

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

SELECT \*

FROM orders

WHERE sales BETWEEN 100 AND 500;

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

SELECT \*

FROM customers

WHERE last\_name LIKE '\_\_\_\_';

**SELECTION COMMANDS: - ordering**

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

SELECT \*

FROM orders

WHERE discount > 0

ORDER BY discount DESC;

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

SELECT \*

FROM orders

WHERE discount > 0

ORDER BY discount DESC

LIMIT 10;

**Aggregate commands: -**

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

SELECT SUM(sales) AS total\_sales

FROM orders;

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

SELECT COUNT(\*) AS num\_customers

FROM customers

WHERE region = 'North' AND age BETWEEN 20 AND 30;

1. Find the average age of east region customers

SELECT AVG(age) AS avg\_age

FROM customers

WHERE region = 'East';

1. 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 COMMANDS: -**

1. Make a dashboard showing the following figures for each product ID
2. Total sales (in $) order by this column in descending
3. Total sales quantity
4. The number of orders
5. Max Sales value
6. Min Sales value
7. Average sales value

SELECT product\_id,

SUM(sales) AS total\_sales,

SUM(quantity) AS total\_quantity,

COUNT(DISTINCT order\_id) AS num\_orders,

MAX(sales) AS max\_sales,

MIN(sales) AS min\_sales,

AVG(sales) AS avg\_sales

FROM orders

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 orders

GROUP BY product\_id

HAVING SUM(quantity) > 10;