1. **SQL 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: NISHANTH V Batch ID: CDA15052023**

**Topic: SQL Operators**

**Assignments: -**

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

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

In a relational database, relationships between tables are established using constraints and keys. These constraints ensure data integrity and enforce rules regarding how tables are related to each other. Here are some common types of constraints/keys used to define relationships:

1. Primary Key (PK): A primary key is a unique identifier for each record in a table. It ensures that each row in the table is uniquely identifiable. A primary key constraint is applied to one or more columns in a table, and it enforces uniqueness and non-nullability of those columns. A primary key is typically used to establish a one-to-many relationship with another table.

2. Foreign Key (FK): A foreign key is a column or set of columns in one table that refers to the primary key of another table. It establishes a relationship between two tables, known as a foreign key relationship. The foreign key constraint ensures referential integrity, meaning that the values in the foreign key column(s) must match an existing value in the primary key column of the referenced table.

3. Unique Constraint: A unique constraint ensures that the values in one or more columns of a table are unique, but unlike a primary key, it allows null values. It can be used to establish a one-to-one relationship between tables, where each record in one table is associated with at most one record in another table.

4. Check Constraint: A check constraint defines a condition that must be satisfied for the values in a column. It allows you to enforce domain-specific rules on the data. While check constraints are not directly used to establish relationships between tables, they can be used to maintain data integrity within a table and indirectly affect relationships.

These constraints and keys work together to define and maintain relationships between tables in a relational database, ensuring data consistency and referential integrity. By enforcing these relationships, you can perform operations such as joins to retrieve related data across multiple tables.

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 the IN statement.

Select distinct city

From supermart

Where region in (‘norht’, ‘east’);

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

Select \*

From supermart

Where sales between 100 and 200;

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

Select last\_name

From supermart

Where last\_name like ‘\_\_\_\_’;

**SELECTION Operators: - ordering**

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

SELECT \* FROM supermart

Where discount > 0

Order by discount desc;

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

SELECT \* FROM supermart

Where discount > 0

Order by discount desc

Limit 10;

**Aggregate commands: -**

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

Select sum(sales)

From table;

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

Select count(\*)

From customers

Where region = ‘north region’ and age between 20 and 30;

1. Find the average age of East region customers.

Select avg(age)

From customers

Where region = ‘east’;

1. Find the minimum and maximum aged customers from Philadelphia.

Select region, min(age) as min\_age, max(age) as max\_age

From customers

Where state = ‘philadelphia’;

**GROUP BY Operators: -**

1. Create a display with the information below 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 max(sales) as max\_sales,

Min(sales) as min\_sales,

Avg(sales) as avg\_sales,

Count(\*) as number\_of\_orders,

Sum(sales) as total\_quantity\_sales

From supermart

Group by product\_id

1. Get the list of product ID’s where the quantity of product sold is greater than 10.

Select product\_id,

From supermart,

Group by product\_id,

Having count(\*) >10;