Part 1

A)

Between:

The SQL comparison operator "BETWEEN" is used to ascertain if a value is inside a certain range. You may use a database query to filter data according to a range of values for a certain attribute. For the operator to function properly, it is crucial to first supply the lower range value.

For example:

If we want to find a price of a certain product from a PRODUCTS table where the PRICE of the product we want to find should be between 250 to 300; then the query for that will be like following:

SELECT \*

FROM PRODUCTS

WHERE PRICE BETWEEN 250.00 AND 300.00;

In the above scenario:

The table "PRODUCTS" is where you may find details about every product in your business.

The property called "PRICE" is used to store each product's pricing.

The goods whose prices fall within the designated range of $250.00 to $300.00 are filtered using the "BETWEEN" operator.

A list of all the goods in your database that fit the pricing requirements will be the output of this query. This may be helpful for many things, such creating reports, revising prices, or looking at the stock of goods that fall within a certain price range.

b)

Cross Join:

In relational databases, a cross join, also called a cartesian product, is a kind of join operation that joins all the rows from one table with all the rows from another, creating a new table in which every row from the first table is matched with every row from the second. It creates a result set whose size is equal to the multiplication of the number of rows in each of the two tables involved, thereby acting as the product of the two tables.

Example:

SELECT Students.Student\_Name, Courses.Course\_Name

FROM Students

CROSS JOIN Courses;

In the above scenario:

There are two database tables with information about students and courses, they are called "Students" with “Student\_Name” column and "Courses" with “Course\_Name” column, containing 3 records in each table respectively, in this example. The SQL query creates a comprehensive list of all potential student-course pairings by utilizing a Cross Join to join every student from the "Students" dataset with every course from the "Courses" table. As a consequence, a new table is created in which the names of all the students enrolled in the courses are linked. The output will give nine rows overall.

c)

EXISTS:

"EXISTS" is a SQL operator used in computer database systems that determines whether a subquery returns any results. The "EXISTS" condition is met if the subquery yields one or more rows, in which case the main query is executed. On the other hand, the execution of the main query is ignored if the subquery returns no results, indicating that the condition is false. When paired with correlated subqueries, "EXISTS" may assess correlations between the subquery and the main query. It's a useful tool that's frequently used in database queries to filter data according to particular criteria.

Example:

SELECT EmployeeName

FROM Employees

WHERE EXISTS (SELECT \* FROM TrainingHistory WHERE TrainingHistory.EmployeeID = Employees.EmployeeID);

The "EXISTS" operator in the above SQL query is used to find out which workers have finished their training. The subquery verifies whether the "EmployeeID" values in the primary "Employees" table and the "TrainingHistory" table match. The "EXISTS" condition is satisfied and the employee's name is obtained from the result set if at least one match is discovered. Using keywords like "SELECT," "WHERE," and "EXISTS" to filter data based on related records in a linked subquery, this query effectively finds workers with training records.

D)

IS NULL:

The comparison operator "IS NULL" in SQL is used to ascertain whether a field or attribute in a database table is null, or more specifically, if it has a value. It is important to distinguish between a null value, which denotes the lack of any data or value, and a specific value, such as 0 or an empty string. When "IS NULL" is used in a query, it evaluates to false if an attribute has a value and to true otherwise.

Example:

SELECT Order\_ID, Order\_Date, Customer\_ID

FROM Orders

WHERE Order\_Status IS NULL;

In order to filter and obtain entries in this SQL query when the "Order\_Status" property is null—a hint that the order is pending and has not yet been assigned a status—the expression "IS NULL" is used. This is a useful use of "IS NULL" that helps companies effectively manage their order fulfillment process by identifying orders that require attention or further processing.

e)

Recursive Query:

A kind of database query called a recursive query joins a table to itself. When working with hierarchical or self-referential data structures, including network graphs, family trees, or organizational charts, where data items have associations with other pieces inside the same dataset, this is often utilized. Recursive queries are helpful because they allow you to traverse relationships inside a table to retrieve data from these kinds of structures.

Example:

A recursive query iterates through a table containing the following columns: Dept\_Name (department name), Parent\_Dept\_ID (parent department ID), and Dept\_ID (department ID) in the department hierarchy of a university. Starting with the upper level "University," all departments are recursively extracted and categorized according to levels by the query. The output that is produced shows the university's departments' hierarchical structure.

Part 2:

1. F
2. T
3. T
4. F
5. T

Part 3:

1. 3
2. A. select from tenant orderby rent
3. C. select side date due +30 as due date from service
4. 2
5. b. **Rows match the join condition and rows in table1 with unmatched id on table2.**

Part 4: