50 SQL Interview Q/A

1. What is SQL?

Answer: SQL stands for Structured Query Language, and it is a domain-specific language used for managing and manipulating relational databases.

2. What is a database?

Answer: A database is a structured collection of data that is organized and stored for efficient retrieval and manipulation.

3. Explain the difference between SQL and MySQL.

Answer: SQL is a language used to manage and manipulate relational databases, while MySQL is an open-source relational database management system (RDBMS) that uses SQL as its query language.

4. What is a primary key?

Answer: A primary key is a unique identifier for a record in a database table. It ensures that each record can be uniquely identified and helps maintain data integrity.

5. What is a foreign key?

Answer: A foreign key is a field in a database table that is used to establish a link between the data in two tables. It creates a relationship between the tables by referencing the primary key of another table.

6. Explain the types of SQL commands.

Answer: SQL commands can be broadly categorized into Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL), and Data Control Language (DCL).

7. What is the difference between CHAR and VARCHAR data types?

Answer: CHAR is a fixed-length character data type, while VARCHAR is a variable-length character data type. VARCHAR is more flexible as it only stores the characters entered, whereas CHAR pads the remaining spaces with blanks.

8. What is normalization?

Answer: Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller tables and defining relationships between them.

9. Explain the difference between INNER JOIN and LEFT JOIN.

Answer: INNER JOIN returns only the rows where there is a match in both tables, while LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

10. What is the purpose of the GROUP BY clause?

Answer: The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows, like finding the total or average for each group.

11. Explain the difference between UNION and UNION ALL.

Answer: UNION combines the result sets of two queries, eliminating duplicate rows, while UNION ALL combines the result sets including duplicates.

12. What is an index, and why is it used?

Answer: An index is a data structure that improves the speed of data retrieval operations on a database table. It is used to quickly locate and access the rows in a table based on the values in one or more columns.

13. What is a stored procedure?

Answer: A stored procedure is a precompiled collection of one or more SQL statements that can be executed as a single unit. It is stored in the database and can be called by name.

14. Explain the ACID properties of a transaction.

Answer: ACID stands for Atomicity, Consistency, Isolation, and Durability. It is a set of properties that guarantee that database transactions are processed reliably.

15. What is the purpose of the HAVING clause?

Answer: The HAVING clause is used in combination with the GROUP BY clause to filter the results of a group based on a specified condition.

16. What is a view in SQL?

Answer: A view is a virtual table derived from one or more tables. It does not store the data itself but provides a way to represent the result of a stored query.

17. Explain the difference between a clustered and a non-clustered index.

Answer: A clustered index determines the physical order of data in a table, whereas a non-clustered index does not affect the physical order and creates a separate structure for faster data retrieval.

18. What is the purpose of the SQL ORDER BY clause?

Answer: The ORDER BY clause is used to sort the result set of a query in ascending or descending order based on one or more columns.

19. What is a trigger?

Answer: A trigger is a set of instructions that are automatically executed (or "triggered") in response to certain events on a particular table or view in a database.

20. Explain the difference between a candidate key, primary key, and a super key.

Answer: A super key is any set of columns that uniquely identifies a row, a candidate key is a minimal super key, and the primary key is the chosen candidate key for a table.

21. What is the purpose of the SQL LIKE statement?

Answer: The LIKE statement is used in a WHERE clause to search for a specified pattern in a column.

22. What is the difference between a view and a table?

Answer: A table is a physical storage structure that stores data, while a view is a virtual table derived from one or more tables, presenting a way to represent the result of a stored query.

23. What is a self-join?

Answer: A self-join is a regular join, but the table is joined with itself. It is often used when a table has a foreign key that references its own primary key.

24. Explain the purpose of the SQL IN operator.

Answer: The IN operator is used in a WHERE clause to filter the result set based on a specified list of values.

25. What is a subquery?

Answer: A subquery is a query nested inside another query. It can be used to retrieve data that will be used in the main query as a condition to further restrict the data to be retrieved.

26. What is the purpose of the SQL GROUP BY and COUNT() functions?

Answer: The GROUP BY clause is used to group rows based on specified columns, and COUNT() is an aggregate function that counts the number of rows in each group.

27. Explain the difference between a DELETE and TRUNCATE statement.

Answer: DELETE is used to remove rows from a table based on a condition, while TRUNCATE removes all rows from a table and is faster but cannot be rolled back.

28. What is a cross join?

Answer: A cross join, or Cartesian join, returns the Cartesian product of the sets of rows from the joined tables. It results in every combination of rows from both tables.

29. What is the purpose of the SQL BETWEEN operator?

Answer: The BETWEEN operator is used in a WHERE clause to filter the result set based on a range of values.

30. Explain the purpose of the SQL UPDATE statement.

Answer: The UPDATE statement is used to modify the existing records in a table. It allows you to update specific columns with new values.

31. What is a composite key?

Answer: A composite key is a combination of two or more columns in a table that uniquely identifies a record. Each column in the composite key contributes to the uniqueness of the key.

32. What is the difference between a view and a materialized view?

Answer: A view is a virtual table derived from one or more tables, while a materialized view is a physical copy of the result set of a query stored for faster data retrieval.

33. Explain the purpose of the SQL MAX() function.

Answer: The MAX() function is an aggregate function that returns the maximum value in a set of values.

34. What is the purpose of the SQL DISTINCT keyword?

Answer: The DISTINCT keyword is used in a SELECT statement to eliminate duplicate records from the result set.

35. Explain the purpose of the SQL JOIN statement.

Answer: The JOIN statement is used to combine rows from two or more tables based on a related column between them.

36. What is a cursor in SQL?

Answer: A cursor is a database object used to traverse the result set of a query. It allows you to process each row individually.

37. Explain the difference between a left outer join and a right outer join.

Answer: A left outer join returns all rows from the left table and the matched rows from the right table, while a right outer join returns all rows from the right table and the matched rows from the left table.

38. What is the purpose of the SQL AVG() function?

Answer: The AVG() function is an aggregate function that returns the average value of a numeric column.

39. Explain the purpose of the SQL CASE statement.

Answer: The CASE statement is used to perform conditional logic in SQL queries, similar to the "if-else" statements in programming languages.

40. What is the difference between a database and a schema?

Answer: A database is a collection of tables, while a schema is a collection of database objects, including tables, views, and stored procedures.

41. What is the purpose of the SQL COUNT() function?

Answer: The COUNT() function is an aggregate function that returns the number of rows in a result set or the number of non-null values in a column.

42. Explain the difference between the SQL WHERE and HAVING clauses.

Answer: The WHERE clause is used to filter rows before grouping in a query, while the HAVING clause is used to filter groups after they have been formed.

43. What is the purpose of the SQL ROLLBACK statement?

Answer: The ROLLBACK statement is used to undo transactions that have not been saved to the database. It is typically used in error-handling scenarios.

44. Explain the difference between UNION and JOIN.

Answer: UNION combines the result sets of two queries, while JOIN combines rows from two or more tables based on a related column.

45. What is the purpose of the SQL TRIGGER statement?

Answer: The TRIGGER statement is used to specify a set of actions that are automatically performed when a certain event occurs in a particular table or view.

46. Explain the purpose of the SQL SUM() function.

Answer: The SUM() function is an aggregate function that returns the sum of all values in a numeric column.

47. What is a natural join?

Answer: A natural join is a type of join that combines tables based on columns with the same name and automatically eliminates one of the duplicate columns.

48. What is the purpose of the SQL NULL value?

Answer: The NULL value in a database represents the absence of a value in a column. It is not the same as an empty string or zero.

49. Explain the difference between a unique key and a primary key.

Answer: A unique key ensures that all values in a column are unique, while a primary key is a unique key that also serves as the main identifier for a record in a table.

50. What is the purpose of the SQL INSERT statement?

Answer: The INSERT statement is used to insert new records into a table. It allows you to specify the values for each column or use a subquery to select the values from another table.

1. What is Database?

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

2. What is DBMS?

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

3. What is RDBMS? How is it different from DBMS?

RDBMS stands for Relational Database Management System. The key difference here, compared to DBMS, is that RDBMS stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2, and Amazon Redshift are based on RDBMS.

4. What is SQL?

SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

5. What is the difference between SQL and MySQL?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

6. What are Tables and Fields?

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

7. What are Constraints in SQL?

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

- **NOT NULL** - Restricts NULL value from being inserted into a column.
- **CHECK** - Verifies that all values in a field satisfy a condition.
- **DEFAULT** - Automatically assigns a default value if no value has been specified for the field.
- **UNIQUE** - Ensures unique values to be inserted into the field.
- **INDEX** - Indexes a field providing faster retrieval of records.
- **PRIMARY KEY** - Uniquely identifies each record in a table.
- **FOREIGN KEY** - Ensures referential integrity for a record in another table.

8. What is a Primary Key?

The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.
A table in SQL is strictly restricted to have one and only one primary key, which is comprised of single or multiple fields (columns).

```
CREATE TABLE Students (   /* Create table with a single field as primary key
*/
   ID INT NOT NULL
   Name VARCHAR(255)
   PRIMARY KEY (ID)
);

CREATE TABLE Students (   /* Create table with multiple fields as primary key
*/
   ID INT NOT NULL
   LastName VARCHAR(255)
   FirstName VARCHAR(255) NOT NULL,
   CONSTRAINT PK_Student
   PRIMARY KEY (ID, FirstName)
);

ALTER TABLE Students   /* Set a column as primary key */
ADD PRIMARY KEY (ID);
ALTER TABLE Students   /* Set multiple columns as primary key */
ADD CONSTRAINT PK_Student   /*Naming a Primary Key*/
PRIMARY KEY (ID, FirstName);
```

9. What is a UNIQUE constraint?

A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

```
CREATE TABLE Students (   /* Create table with a single field as unique */
   ID INT NOT NULL UNIQUE
   Name VARCHAR(255)
);

CREATE TABLE Students (   /* Create table with multiple fields as unique */
   ID INT NOT NULL
   LastName VARCHAR(255)
   FirstName VARCHAR(255) NOT NULL
   CONSTRAINT PK_Student
   UNIQUE (ID, FirstName)
);

ALTER TABLE Students   /* Set a column as unique */
ADD UNIQUE (ID);
ALTER TABLE Students   /* Set multiple columns as unique */
ADD CONSTRAINT PK_Student   /* Naming a unique constraint */
UNIQUE (ID, FirstName);
```

10. What is a Foreign Key?

A FOREIGN KEY comprises of single or collection of fields in a table that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.
The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

```
CREATE TABLE Students (   /* Create table with foreign key - Way 1 */
```

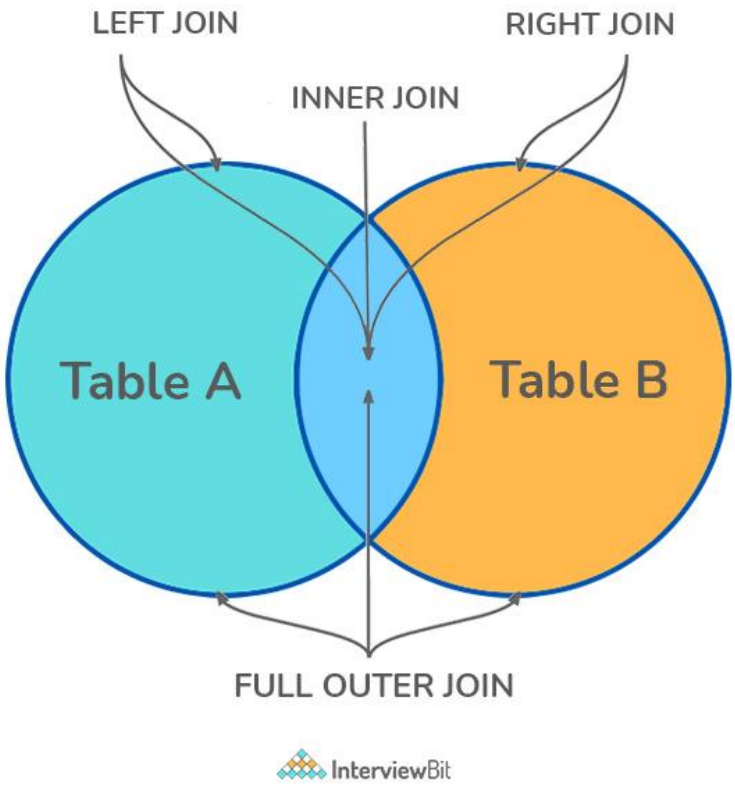
```
   ID INT NOT NULL
   Name VARCHAR(255)
   LibraryID INT
   PRIMARY KEY (ID)
   FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

CREATE TABLE Students (   /* Create table with foreign key - Way 2 */
   ID INT NOT NULL PRIMARY KEY
   Name VARCHAR(255)
   LibraryID INT FOREIGN KEY (Library_ID) REFERENCES Library(LibraryID)
);

ALTER TABLE Students   /* Add a new foreign key */
ADD FOREIGN KEY (LibraryID)
REFERENCES Library (LibraryID);
```

11. What is a Join? List its different types.

The SQL Join clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.



There are four different types of JOINs in SQL:

- **(INNER) JOIN:** Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

```
SELECT *
```

```
FROM Table_A
JOIN Table_B;
SELECT *
FROM Table_A
INNER JOIN Table_B;
```

- **LEFT (OUTER) JOIN:** Retrieves all the records/rows from the left and the matched records/rows from the right table.

```
SELECT *
FROM Table_A A
LEFT JOIN Table_B B
ON A.col = B.col;
```

- **RIGHT (OUTER) JOIN:** Retrieves all the records/rows from the right and the matched records/rows from the left table.

```
SELECT *
FROM Table_A A
RIGHT JOIN Table_B B
ON A.col = B.col;
```

- **FULL (OUTER) JOIN:** Retrieves all the records where there is a match in either the left or right table.

```
SELECT *
FROM Table_A A
FULL JOIN Table_B B
ON A.col = B.col;
```

12. What is a Self-Join?

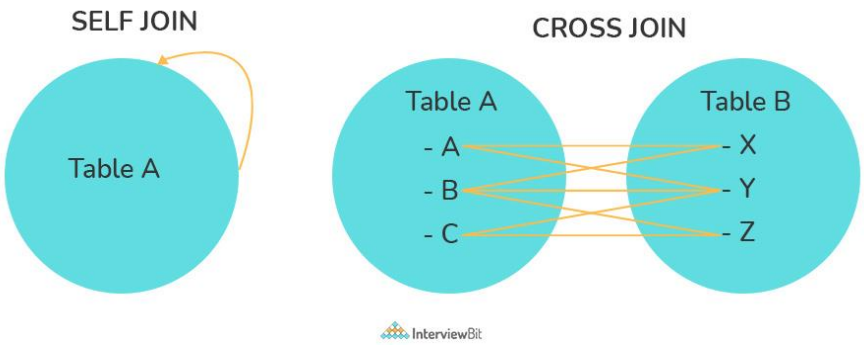
A **self JOIN** is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS "Emp_ID",A.emp_name AS "Employee",
B.emp_id AS "Sup_ID",B.emp_name AS "Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

13. What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of the number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject
FROM students AS stu
CROSS JOIN subjects AS sub;
```



14. What is an Index? Explain its different types.

A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.

```
CREATE INDEX index_name   /* Create Index */
ON table_name (column_1, column_2);
DROP INDEX index_name;    /* Drop Index */
```

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

  Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

  ```
  CREATE UNIQUE INDEX myIndex
  ON students (enroll_no);
  ```

  Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non-unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

  Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

  The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

  Clustering indexes can improve the performance of most query operations because they provide a linear-access path to data stored in the database.

15. What is the difference between Clustered and Non-clustered index?

As explained above, the differences can be broken down into three small factors -

- Clustered index modifies the way records are stored in a database based on the indexed column. A non-clustered index creates a separate entity within the table which references the original table.

- Clustered index is used for easy and speedy retrieval of data from the database, whereas, fetching records from the non-clustered index is relatively slower.
- In SQL, a table can have a single clustered index whereas it can have multiple non-clustered indexes.

16. What is Data Integrity?

Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle and is a critical aspect of the design, implementation, and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

17. What is a Query?

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

```
SELECT fname, lname    /* select query */
FROM myDb.students
WHERE student_id = 1;
UPDATE myDB.students    /* action query */
SET fname = 'Captain', lname = 'America'
WHERE student_id = 1;
```

18. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a **nested query** or **inner query**. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
 SELECT roll_no
 FROM myDb.students
 WHERE subject = 'Maths');
```

There are two types of subquery - **Correlated** and **Non-Correlated**.

- A **correlated** subquery cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
- A **non-correlated** subquery can be considered as an independent query and the output of the subquery is substituted in the main query.

19. What is the SELECT statement?

SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

```
SELECT * FROM myDB.students;
```

20. What are some common clauses used with SELECT query in SQL?

Some common SQL clauses used in conjuction with a SELECT query are as follows:

- **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions.
- **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC**).

```
SELECT *
FROM myDB.students
WHERE graduation_year = 2019
ORDER BY studentID DESC;
```

- **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
- **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
SELECT COUNT(studentId), country
FROM myDB.students
WHERE country != "INDIA"
GROUP BY country
HAVING COUNT(studentID) > 5;
```

21. What are UNION, MINUS and INTERSECT commands?

The **UNION** operator combines and returns the result-set retrieved by two or more SELECT statements.
The **MINUS** operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.
The **INTERSECT** clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

- Each SELECT statement within the clause must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement should necessarily have the same order

```
SELECT name FROM Students   /* Fetch the union of queries */
UNION
SELECT name FROM Contacts;
SELECT name FROM Students   /* Fetch the union of queries with duplicates*/
UNION ALL
SELECT name FROM Contacts;
SELECT name FROM Students   /* Fetch names from students */
MINUS     /* that aren't present in contacts */
SELECT name FROM Contacts;
SELECT name FROM Students   /* Fetch names from students */
INTERSECT    /* that are present in contacts as well */
SELECT name FROM Contacts;
```

22. What is Cursor? How to use a Cursor?

A database cursor is a control structure that allows for the traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition, and deletion of database records. They can be viewed as a pointer to one row in a set of rows.

Working with SQL Cursor:

1. **DECLARE** a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.

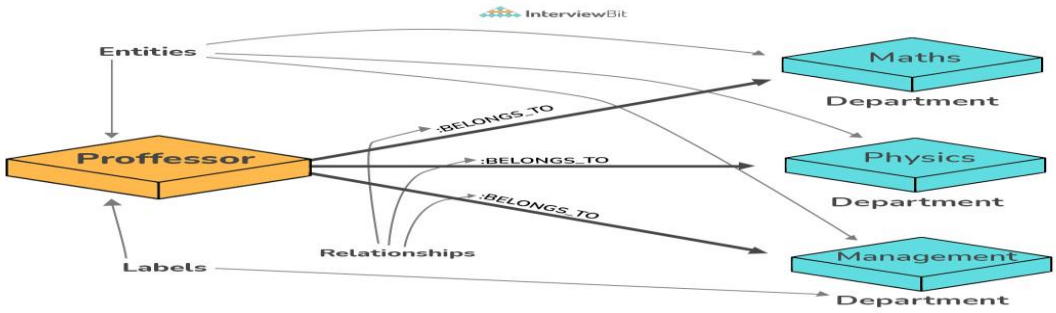
2. Open cursor to initialize the result set. The **OPEN** statement must be called before fetching rows from the result set.
3. **FETCH** statement to retrieve and move to the next row in the result set.
4. Call the **CLOSE** statement to deactivate the cursor.
5. Finally use the **DEALLOCATE** statement to delete the cursor definition and release the associated resources.

```
DECLARE @name VARCHAR(50)   /* Declare All Required Variables */
DECLARE db_cursor CURSOR FOR   /* Declare Cursor Name*/
SELECT name
FROM myDB.students
WHERE parent_name IN ('Sara', 'Ansh')
OPEN db_cursor   /* Open cursor and Fetch data into @name */
FETCH next
FROM db_cursor
INTO @name
CLOSE db_cursor   /* Close the cursor and deallocate the resources */
DEALLOCATE db_cursor
```

23. What are Entities and Relationships?

Entity: An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.



24. List the different types of relationships in SQL.

- **One-to-One** - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- **One-to-Many & Many-to-One** - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- **Many-to-Many** - This is used in cases when multiple instances on both sides are needed for defining a relationship.
- **Self-Referencing Relationships** - This is used when a table needs to define a relationship with itself.

25. What is an Alias in SQL?

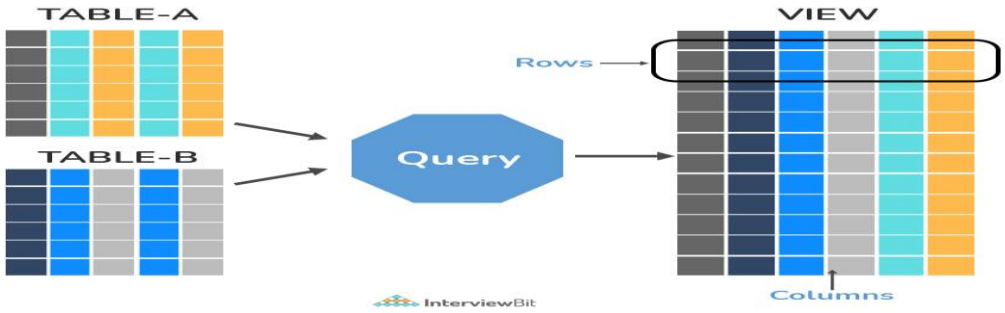
An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

An alias is represented explicitly by the AS keyword but in some cases, the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee"  /* Alias using AS keyword */
B.emp_name AS "Supervisor"
FROM employee A, employee B   /* Alias without AS keyword */
WHERE A.emp_sup = B.emp_id;
```

26. What is a View?

A view in SQL is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.



27. What is Normalization?

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

28. What is Denormalization?

Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

29. What are the various forms of Normalization?

Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

- **First Normal Form:**
  A relation is in first normal form if every attribute in that relation is a **single-valued attribute**. If a relation contains a composite or multi-valued attribute, it violates the first normal form. Let's consider the following **students** table. Each student in the table, has a name, his/her address, and the books they issued from the public library -

Students Table

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (Dan Brown)	Mr.

Student	Address	Books Issued	Salutation
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form -

Students Table (1st Normal Form)

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

- **Second Normal Form:**

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has **no partial dependency**, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples -

Example 1 - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new ***Primary Key*** attribute to identify the individual records in the Students table. The ***Foreign Key*** constraint will be set on the other table to ensure referential integrity.

Students Table (2nd Normal Form)

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.
4	Ansh	Windsor Street 777	Mr.

Books Table (2nd Normal Form)

Student_ID	Book Issued

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Example 2 - Consider the following dependencies in relation to R(W,X,Y,Z)

```
WX -> Y     [W and X together determine Y]
XY -> Z     [X and Y together determine Z]
```

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn't determine any non-prime attribute in the relation.

- **Third Normal Form**

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is **no transitive dependency** between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

Example 1 - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single candidate key Student_ID (primary key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new ***Foreign Key*** constraint to identify the salutations for individual records in the Students table. The ***Primary Key*** constraint for the same will be set on the Salutations table to identify each record uniquely.

Students Table (3rd Normal Form)

Student_ID	Student	Address	Salutation_ID
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

Books Table (3rd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)

Student_ID	Book Issued
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Salutations Table (3rd Normal Form)

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

Example 2 - Consider the following dependencies in relation to R(P,Q,R,S,T)

```
P -> QR     [P together determine C]
RS -> T     [B and C together determine D]
Q -> S
T -> P
```

For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

- **Boyce-Codd Normal Form**

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand-Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form X –> Y, such that X is always a super key. For example - In the above example, Student_ID serves as the sole unique identifier for the Students Table and Salutation_ID for the Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student_ID.

30. What are the TRUNCATE, DELETE and DROP statements?

DELETE statement is used to delete rows from a table.

```
DELETE FROM Candidates
WHERE CandidateId > 1000;
```

TRUNCATE command is used to delete all the rows from the table and free the space containing the table.

```
TRUNCATE TABLE Candidates;
```

DROP command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database.

```
DROP TABLE Candidates;
```

31. What is the difference between DROP and TRUNCATE statements?

If a table is dropped, all things associated with the tables are dropped as well. This includes - the relationships defined on the table with other tables, the integrity checks and constraints, access privileges and other grants that the table has. To create and use the table again in its original form, all these relations, checks, constraints, privileges and relationships need to be redefined. However, if a table is truncated, none of the above problems exist and the table retains its original structure.

32. What is the difference between DELETE and TRUNCATE statements?

The **TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.
The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.

33. What are Aggregate and Scalar functions?

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

- **AVG()** - Calculates the mean of a collection of values.
- **COUNT()** - Counts the total number of records in a specific table or view.
- **MIN()** - Calculates the minimum of a collection of values.
- **MAX()** - Calculates the maximum of a collection of values.
- **SUM()** - Calculates the sum of a collection of values.
- **FIRST()** - Fetches the first element in a collection of values.
- **LAST()** - Fetches the last element in a collection of values.

Note: All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

- **LEN()** - Calculates the total length of the given field (column).
- **UCASE()** - Converts a collection of string values to uppercase characters.
- **LCASE()** - Converts a collection of string values to lowercase characters.
- **MID()** - Extracts substrings from a collection of string values in a table.
- **CONCAT()** - Concatenates two or more strings.
- **RAND()** - Generates a random collection of numbers of a given length.
- **ROUND()** - Calculates the round-off integer value for a numeric field (or decimal point values).
- **NOW()** - Returns the current date & time.
- **FORMAT()** - Sets the format to display a collection of values.

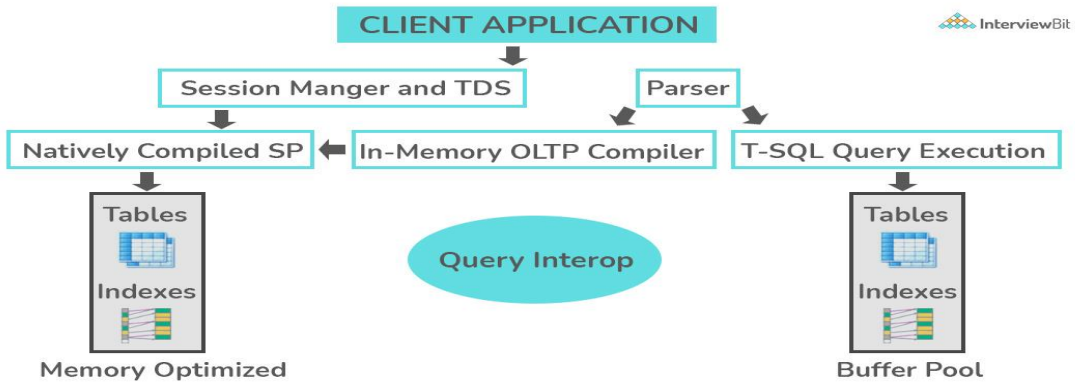
34. What is User-defined function? What are its various types?

The user-defined functions in SQL are like functions in any other programming language that accept parameters, perform complex calculations, and return a value. They are written to use the logic repetitively whenever required. There are two types of SQL user-defined functions:

- Scalar Function: As explained earlier, user-defined scalar functions return a single scalar value.
- Table-Valued Functions: User-defined table-valued functions return a table as output.
  - **Inline:** returns a table data type based on a single SELECT statement.
  - **Multi-statement:** returns a tabular result-set but, unlike inline, multiple SELECT statements can be used inside the function body.

35. What is OLTP?

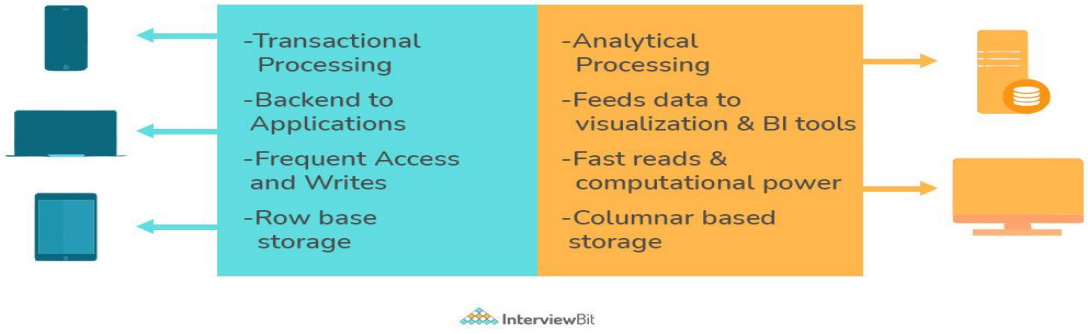
OLTP stands for Online Transaction Processing, is a class of software applications capable of supporting transaction-oriented programs. An essential attribute of an OLTP system is its ability to maintain concurrency. To avoid single points of failure, OLTP systems are often decentralized. These systems are usually designed for a large number of users who conduct short transactions. Database queries are usually simple, require sub-second response times, and return relatively few records. Here is an insight into the working of an OLTP system *[Note - The figure is not important for interviews] -*



36. What are the differences between OLTP and OLAP?

OLTP stands for **Online Transaction Processing**, is a class of software applications capable of supporting transaction-oriented programs. An important attribute of an OLTP system is its ability to maintain concurrency. OLTP systems often follow a decentralized architecture to avoid single points of failure. These systems are generally designed for a large audience of end-users who conduct short transactions. Queries involved in such databases are generally simple, need fast response times, and return relatively few records. A number of transactions per second acts as an effective measure for such systems.

OLAP stands for **Online Analytical Processing**, a class of software programs that are characterized by the relatively low frequency of online transactions. Queries are often too complex and involve a bunch of aggregations. For OLAP systems, the effectiveness measure relies highly on response time. Such systems are widely used for data mining or maintaining aggregated, historical data, usually in multi-dimensional schemas.



37. What is Collation? What are the different types of Collation Sensitivity?

Collation refers to a set of rules that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case sensitivity, accent marks, kana character types, and character width. Below are the different types of collation sensitivity:

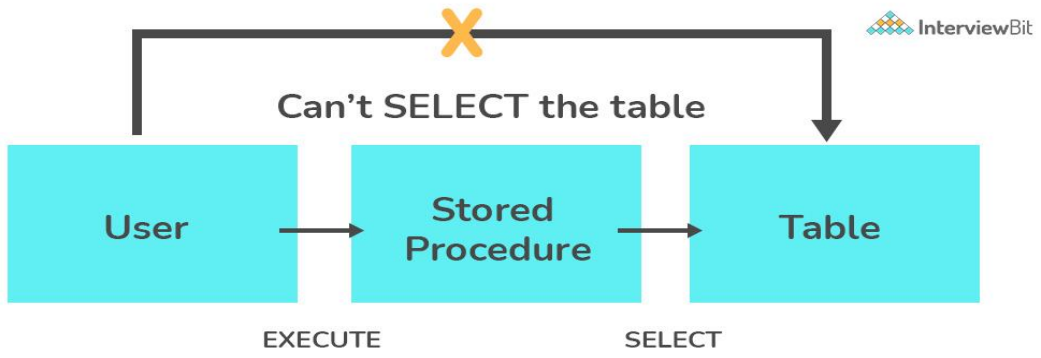
- **Case sensitivity: A** and **a** are treated differently.

- **Accent sensitivity: a** and **á** are treated differently.
- **Kana sensitivity:** Japanese kana characters Hiragana and Katakana are treated differently.
- **Width sensitivity:** Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

38. What is a Stored Procedure?

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

```
DELIMITER $$
CREATE PROCEDURE FetchAllStudents()
BEGIN
SELECT *  FROM myDB.students;
END $$
DELIMITER ;
```



39. What is a Recursive Stored Procedure?

A stored procedure that calls itself until a boundary condition is reached, is called a recursive stored procedure. This recursive function helps the programmers to deploy the same set of code several times as and when required. Some SQL programming languages limit the recursion depth to prevent an infinite loop of procedure calls from causing a stack overflow, which slows down the system and may lead to system crashes.

```
DELIMITER $$     /* Set a new delimiter => $$ */
CREATE PROCEDURE calctotal( /* Create the procedure */
   IN number INT,   /* Set Input and Ouput variables */
   OUT total INT
) BEGIN
DECLARE score INT DEFAULT NULL;   /* Set the default value => "score" */
SELECT awards FROM achievements   /* Update "score" via SELECT query */
WHERE id = number INTO score;
IF score IS NULL THEN SET total = 0;   /* Termination condition */
ELSE
CALL calctotal(number+1);   /* Recursive call */
SET total = total + score;   /* Action after recursion */
END IF;
END $$     /* End of procedure */
DELIMITER ;     /* Reset the delimiter */
```

40. How to create empty tables with the same structure as another table?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```
SELECT * INTO Students_copy
FROM Students WHERE 1 = 2;
```

41. What is Pattern Matching in SQL?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

- **Using the % wildcard to perform a simple search**

  The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *
FROM students
WHERE first_name LIKE 'K%'
```

- **Omitting the patterns using the NOT keyword**

  Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *
FROM students
WHERE first_name NOT LIKE 'K%'
```

- **Matching a pattern anywhere using the % wildcard twice**

  Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *
FROM students
WHERE first_name LIKE '%Q%'
```

- **Using the _ wildcard to match pattern at a specific position**

  The _ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```
SELECT *
FROM students
WHERE first_name LIKE '__K%'
```

- **Matching patterns for a specific length**

The _ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```
SELECT *   /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '___%'

SELECT *   /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'
```

BOSSCODER ACADEMY

MOST ASKED SQL INTERVIEW QUESTIONS

at MAANG Companies

Disclaimer

Everyone learns uniquely.

Master SQL concepts by through these important interview questions.

Take the help of this doc, and crack your SQL questions like a pro.

Medium-Level SQL Questions

Q1.

Explain the difference between INNER JOIN and LEFT JOIN.

Ans.

An **‘INNER JOIN’** returns only the rows where there is a match in both tables. A **‘LEFT JOIN’** returns all rows from the left table and matched rows from the right table, and **‘NULL’** for non-matching rows in the right table.

Q2.

What is the purpose of the GROUP BY clause?

Ans.

The GROUP BY clause groups rows that have the same values in specified columns into summary rows, like "find the number of customers in each country."

Q3.

How can you delete duplicate rows in a SQL table?

Ans.

```sql
DELETE FROM your_table
WHERE id NOT IN (
  SELECT MIN(id)
  FROM your_table
  GROUP BY column_with_duplicates
);
```

Q4.

What are subqueries and how are they used?

Ans.

Subqueries are nested queries used within a main query to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Q5.

Explain the concept of a ‘VIEW’ in SQL.

Ans.

A ‘VIEW’ is a virtual table based on the result set of an SQL query. It can simplify complex queries, improve security by restricting data access, and present a consistent, logical view of the data.

Q6.

What is a ‘UNION’ operator?

Ans.

The ‘UNION’ operator is used to combine the result sets of two or more ‘SELECT’ statements. Each ‘SELECT’ statement must have the same number of columns in the result sets with similar data types.

Q7.

How do you use the ‘CASE’ statement in SQL?

Ans.

```sql
SELECT name,
  CASE
    WHEN score >= 90 THEN 'A'
    WHEN score >= 80 THEN 'B'
    ELSE 'C'
  END as grade
FROM students;
```

Q8.

What is an ‘INDEX’ and why is it important?

Ans.

An index is a database object that improves the speed of data retrieval operations on a table at the cost of additional writes and storage space.

Q9.

Describe the use of ‘TRIGGERS’ in SQL.

Ans.

Triggers are database objects that automatically execute a specified SQL procedure when a certain event occurs on a table or view, such as an ‘INSERT’, ‘UPDATE’, or ‘DELETE’.

Q10.

Explain the concept of ‘NORMALIZATION’.

Ans.

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.

Advanced SQL Questions

Q11.

What is a ‘CTE’ (Common Table Expression)?

Ans.

```sql
WITH Sales_CTE (SalesPerson, TotalSales) AS (
  SELECT SalesPerson, SUM(SalesAmount)
  FROM Sales
  GROUP BY SalesPerson
)
SELECT * FROM Sales_CTE;
```

Q12.

Explain the differences between ‘DELETE’, ‘TRUNCATE’, and ‘DROP’ commands.

Ans.

‘DELETE’: Removes rows one at a time and logs individual row deletions. It can include a WHERE clause.

‘TRUNCATE’: Removes all rows from a table without logging individual row deletions. Cannot include a ‘WHERE’ clause.

‘DROP’: Removes a table from the database entirely, including its structure.

Q13.

How do you optimize a slow-running query?

Ans.

Techniques include indexing, query rewriting, reducing the number of joins, avoiding SELECT *, and analyzing the execution plan.

Q14.

What are ‘window functions’ in SQL?

Ans.

```sql
SELECT name, department,
       salary,
       RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank
FROM employees;
```

Q15.

Discuss the concept of ‘ACID’ properties in transactions.

Ans.

Atomicity: Ensures that all operations within a transaction are completed; if not, the transaction is aborted.

Consistency: Ensures that a transaction brings the database from one valid state to another.

Isolation: Ensures that transactions are securely and independently processed at the same time without interference.

Durability: Ensures that once a transaction is committed, it will remain so, even in the event of a power loss, crash, or error.

Q16.

How do you handle transactions in SQL?

Ans.

```sql
BEGIN TRANSACTION;
-- SQL statements
COMMIT;
```

Q17.

What is the difference between ‘HAVING’ and ‘WHERE’ clauses?

Ans.

‘WHERE’ is used to filter records before any groupings are made, while ‘HAVING’ is used to filter values after grouping.

Q18.

Explain the concept of ‘sharding’ in databases.

Ans.

Sharding is a type of database partitioning that splits large databases into smaller, more manageable pieces, called shards, which can be spread across multiple servers.

Q19.

What are ‘stored procedures’ and how are they used?

Ans.

Stored procedures are precompiled collections of SQL statements stored under a name and processed as a unit. They can accept parameters and are used to perform tasks like data validation or access control.

Q20.

Explain the concept of ‘Data Warehousing’.

Ans.

Data warehousing involves the collection, storage, and management of large volumes of data from multiple sources to provide meaningful business insights. It typically involves ETL processes and uses schemas like star or snowflake schema.

Scenario-Based SQL Questions

Scenario 1:

Employee Management System

Tables:

- Employees

```sql
CREATE TABLE Employees (
  EmployeeID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  DepartmentID INT,
  Salary DECIMAL(10, 2),
  HireDate DATE
);
```

- Departments

```sql
CREATE TABLE Departments (
  DepartmentID INT PRIMARY KEY,
  DepartmentName VARCHAR(50)
);
```

Questions:

Q21.

Find the second highest salary in the Employees table.

Ans.

```sql
SELECT MAX(Salary) AS SecondHighestSalary
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

Q22.

List all employees who have been hired in the last year.

Ans.

```sql
SELECT * FROM Employees
WHERE HireDate >= DATEADD(year, -1, GETDATE());
```

Q23.

Find the average salary for each department.

Ans.

```sql
SELECT d.DepartmentName, AVG(e.Salary) AS AverageSalary
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID
GROUP BY d.DepartmentName;
```

Q24.

List all departments along with the count of employees in each department.

Ans.

```sql
SELECT d.DepartmentName, COUNT(e.EmployeeID) AS EmployeeCount
FROM Departments d
LEFT JOIN Employees e ON d.DepartmentID = e.DepartmentID
GROUP BY d.DepartmentName;
```

Scenario 2:

Online Retail Database

Tables:

- Orders

```sql
CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  OrderDate DATE,
  TotalAmount DECIMAL(10, 2)
);
```

- OrderDetails

```sql
CREATE TABLE OrderDetails (
  OrderDetailID INT PRIMARY KEY,
  OrderID INT,
  ProductID INT,
  Quantity INT,
  Price DECIMAL(10, 2)
);
```

- **Products**

```sql
CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  CategoryID INT,
  Price DECIMAL(10, 2)
);
```

Questions:

Q25.

Find the top 3 most sold products in the last month.

Ans.

```sql
SELECT TOP 3 p.ProductName, SUM(od.Quantity) AS TotalSold
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Orders o ON od.OrderID = o.OrderID
WHERE o.OrderDate >= DATEADD(month, -1, GETDATE())
GROUP BY p.ProductName
ORDER BY TotalSold DESC;
```

Q26.

Calculate the total sales for each product category.

Ans.

```sql
SELECT c.CategoryName, SUM(od.Quantity * od.Price) AS TotalSales
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
JOIN Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName;
```

Q27.

Find all customers who placed orders in the last 7 days.

Ans.

```sql
SELECT DISTINCT CustomerID
FROM Orders
WHERE OrderDate >= DATEADD(day, -7, GETDATE());
```

Q28.

Calculate the average order value for each customer.

Ans.

```sql
SELECT CustomerID, AVG(TotalAmount) AS AverageOrderValue
FROM Orders
GROUP BY CustomerID;
```

Q29.

Identify the product that generated the highest revenue.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity * od.Price) AS TotalRevenue
FROM OrderDetails od
JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalRevenue DESC
LIMIT 1;
```

Q30.

List customers who have placed more than 5 orders.

Ans.

```sql
SELECT CustomerID, COUNT(OrderID) AS OrderCount
FROM Orders
GROUP BY CustomerID
HAVING COUNT(OrderID) > 5;
```

Q31.

Find the month-over-month growth rate of total sales.

Ans.

```sql
SELECT
    DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m'))) / LAG(SUM(TotalAmount), 1) OVER (ORDER BY DATE_FORMAT
    (OrderDate, '%Y-%m')) * 100 AS GrowthRate
FROM Orders
GROUP BY DATE_FORMAT(OrderDate, '%Y-%m');
```

Q32.

Identify the employees who processed the highest number of orders.

Ans.

```sql
SELECT e.EmployeeID, COUNT(o.OrderID) AS OrderCount
FROM Orders o
JOIN Employees e ON o.EmployeeID = e.EmployeeID
GROUP BY e.EmployeeID
ORDER BY OrderCount DESC
LIMIT 1;
```

Q33.

Find the products that have not been sold in the last 6 months.

Ans.

```sql
SELECT p.ProductName
FROM Products p
LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID
LEFT JOIN Orders o ON od.OrderID = o.OrderID AND o.OrderDate >= DATEADD
(month, -6, GETDATE())
WHERE o.OrderID IS NULL;
```

Q34.

List customers and the total amount they have spent.

Ans.

```sql
SELECT CustomerID, SUM(TotalAmount) AS TotalSpent
FROM Orders
GROUP BY CustomerID;
```

Q35.

Calculate the average number of items per order.

Ans.

```sql
SELECT AVG(ItemCount) AS AvgItemsPerOrder
FROM (
    SELECT OrderID, SUM(Quantity) AS ItemCount
    FROM OrderDetails
    GROUP BY OrderID
) AS OrderItemCounts;
```

Q36.

Find overlapping date ranges for product availability.

Ans.

sql Copy code

```sql
CREATE TABLE ProductAvailability (
    ProductID INT,
    StartDate DATE,
    EndDate DATE
);

SELECT a.ProductID, a.StartDate, a.EndDate, b.StartDate, b.EndDate
FROM ProductAvailability a, ProductAvailability b
WHERE a.ProductID = b.ProductID
AND a.StartDate < b.EndDate
AND b.StartDate < a.EndDate
AND a.StartDate <> b.StartDate;
```

Q37.

Retrieve the most recent order for each customer.

Ans.

```sql
SELECT CustomerID, MAX(OrderDate) AS MostRecentOrderDate
FROM Orders
GROUP BY CustomerID;
```

Q38.

List products that have been ordered more than 100 times.

Ans.

```sql
SELECT p.ProductName, SUM(od.Quantity) AS TotalOrdered
FROM Products p
JOIN OrderDetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName
HAVING SUM(od.Quantity) > 100;
```

Q39.

Create a query to find the longest consecutive sequence of orders for each customer.

Ans.

sql

Copy code

```sql
WITH ConsecutiveOrders AS (
    SELECT
        CustomerID,
        OrderDate,
        ROW_NUMBER() OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS RowNum
    FROM Orders
)
SELECT CustomerID, COUNT(*) AS LongestStreak
FROM ConsecutiveOrders
GROUP BY CustomerID, DATE_SUB(OrderDate, INTERVAL RowNum DAY)
ORDER BY LongestStreak DESC;
```

Q40.

Fetch the last N records in a table, ordered by a specific column.

Ans.

```sql
SELECT *
FROM Orders
ORDER BY OrderDate DESC
LIMIT N;
```

Q41.

Audit data changes in a SQL database.

Ans.

sql Copy code

```sql
CREATE TABLE AuditLog (
    AuditID INT PRIMARY KEY,
    TableName VARCHAR(50),
    Action VARCHAR(10),
    OldValue VARCHAR(255),
    NewValue VARCHAR(255),
    ChangeDate DATETIME,
    ChangedBy VARCHAR(50)
);

CREATE TRIGGER trgAudit
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
    INSERT INTO AuditLog (TableName, Action, OldValue, NewValue, ChangeDate,
                          ChangedBy)
    VALUES ('Orders', 'UPDATE', OLD.TotalAmount, NEW.TotalAmount, NOW(), USER());
END;
```

Q42.

Design a query to retrieve hierarchical data in a parent-child relationship.

Ans.

sql Copy code

```sql
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    ManagerID INT
);

WITH EmployeeHierarchy AS (
    SELECT EmployeeID, Name, ManagerID, 1 AS Level
    FROM Employees
    WHERE ManagerID IS NULL
    UNION ALL
    SELECT e.EmployeeID, e.Name, e.ManagerID, eh.Level + 1
    FROM Employees e
    JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID
)
SELECT * FROM EmployeeHierarchy
ORDER BY Level, ManagerID;
```

Q43.

Implement soft deletes in SQL.

Ans.

```sql
ALTER TABLE Orders ADD COLUMN IsDeleted BOOLEAN DEFAULT FALSE;

-- Mark as deleted
UPDATE Orders SET IsDeleted = TRUE WHERE OrderID = 1;

-- Retrieve non-deleted orders
SELECT * FROM Orders WHERE IsDeleted = FALSE;
```

Q44.

Write a query to calculate the year-over-year growth of sales.

Ans.

```sql
SELECT
    YEAR(OrderDate) AS Year,
    SUM(TotalAmount) AS TotalSales,
    (SUM(TotalAmount) - LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)))
    / LAG(SUM(TotalAmount), 1) OVER (ORDER BY YEAR(OrderDate)) * 100 AS YoYGrowth
FROM Orders
GROUP BY YEAR(OrderDate);
```

Q45.

Handle data migration from one database to another.

Ans.

```sql
-- Using MySQL Workbench or similar tools for data migration.
-- Example:
mysqldump -u source_user -p source_db > source_db.sql
mysql -u target_user -p target_db < source_db.sql
```

Q46.

How to implement full-text search in SQL?

Ans.

```sql
CREATE FULLTEXT INDEX idx_productname ON Products(ProductName);

SELECT * FROM Products
WHERE MATCH(ProductName) AGAINST('search term');
```

Q47.

Write a query to find the median value in a numeric column.

Ans.

```sql
SELECT AVG(value) AS MedianValue
FROM (
    SELECT value
    FROM table
    ORDER BY value
    LIMIT 2 - (SELECT COUNT(*) FROM table) % 2
    OFFSET (SELECT (COUNT(*) - 1) / 2 FROM table)
) AS MedianValues;
```

Q48.

How to perform a bulk insert in SQL?

Ans.

```sql
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES
(1, 1, '2023-01-01', 100.00),
(2, 2, '2023-01-02', 150.00),
(3, 1, '2023-01-03', 200.00);
```

Q49.

Calculate the running total of sales for each day.

Ans.

```sql
SELECT OrderDate,
       TotalAmount,
       SUM(TotalAmount) OVER (ORDER BY OrderDate) AS RunningTotal
FROM Orders;
```

Q48.

Identify customers who have not placed any orders in the last 6 months.

Ans.

```sql
SELECT CustomerID
FROM Customers
WHERE CustomerID NOT IN (
    SELECT DISTINCT CustomerID
    FROM Orders
    WHERE OrderDate >= DATEADD(month, -6, GETDATE())
);
```

BOSSCODER ACADEMY

WHY BOSSCODER?

1000+ Alumni placed at Top Product-based companies.

More than **136% hike** for every **2 out of 3** working professional.

Average package of **24LPA.**

The syllabus is most **up-to-date** and the list of problems provided covers **all important topics.**

Lavanya
Meta

Course is very well **structured** and **streamlined** to crack any **MAANG** company

Rahul .
Google

EXPLORE MORE

Q 1. What is SQL?

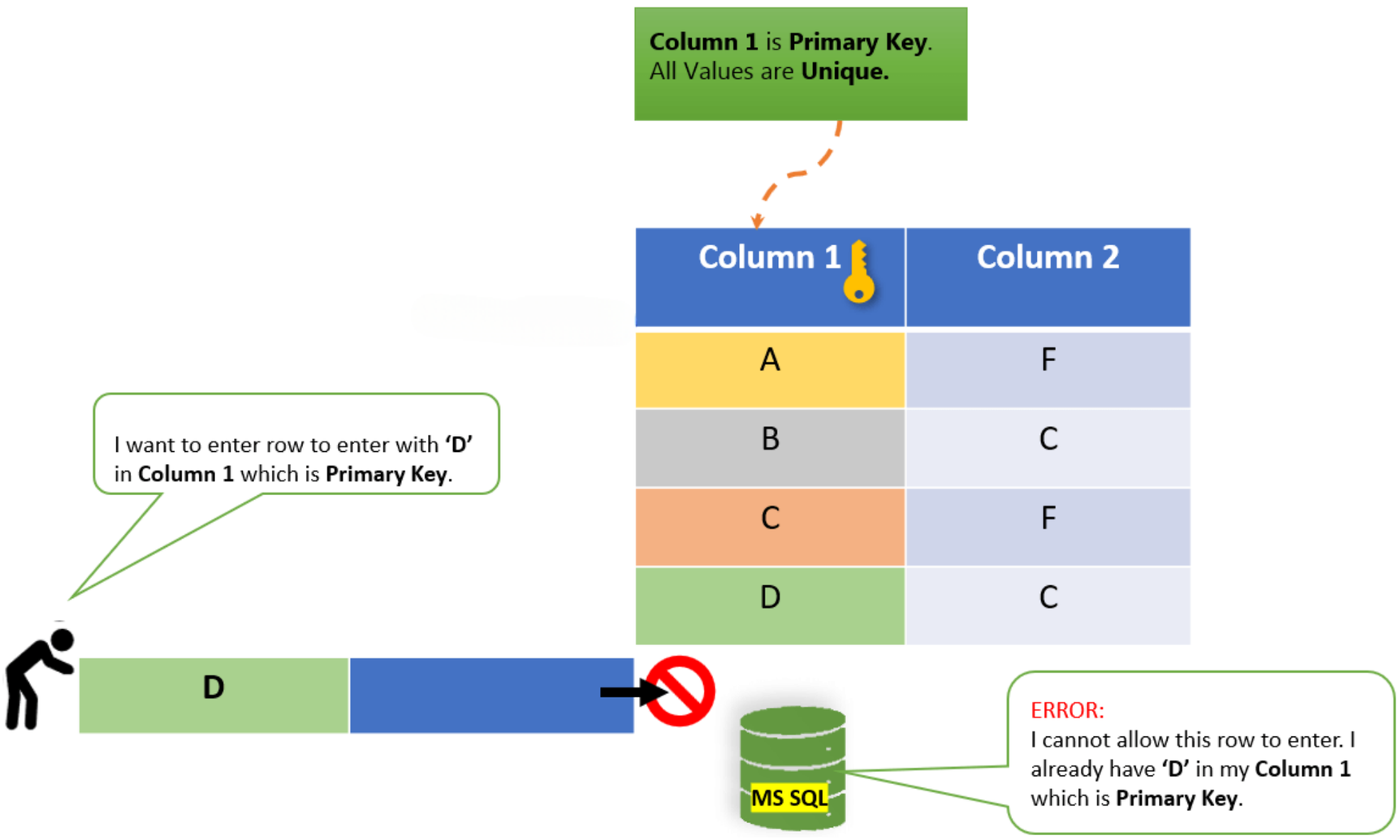
Ans: SQL stands for Structured Query Language. It is a programming language used for managing and manipulating relational databases.

Q 2. What is a database?

Ans: A database is an organized collection of data stored and accessed electronically. It provides a way to store, organize, and retrieve large amounts of data efficiently.

Q 3. What is a primary key?

Ans: A primary key is a column or combination of columns that uniquely identifies each row in a table. It enforces the entity integrity rule in a relational database.



Q 4. What is a foreign key?

Ans: A foreign key is a column or combination of columns that establishes a link between data in two tables. It ensures referential integrity by enforcing relationships between tables.

Department

EmpNo	EmpName	DepNo
1001	Sahil	101
1004	Kavish	102
1006	Aditya	103
1005	Atul	104

Foreign Key

Relationship

Employee

Primary Key

DepNo	DName	Location
101	HR	Delhi
102	Sales	Bangalore
103	Marketing Executive	Hyderabad
104	Technical Engineer	Chennai

Q 5. What is the difference between a primary key and a unique key?

Ans: A primary key is used to uniquely identify a row in a table and must have a unique value. On the other hand, a unique key ensures that a column or combination of columns has a unique value but does not necessarily identify the row.

Q 6. What is normalization?

Ans: Normalization is the process of organizing data in a database to minimize redundancy and dependency. It involves breaking down a table into smaller tables and establishing relationships between them.

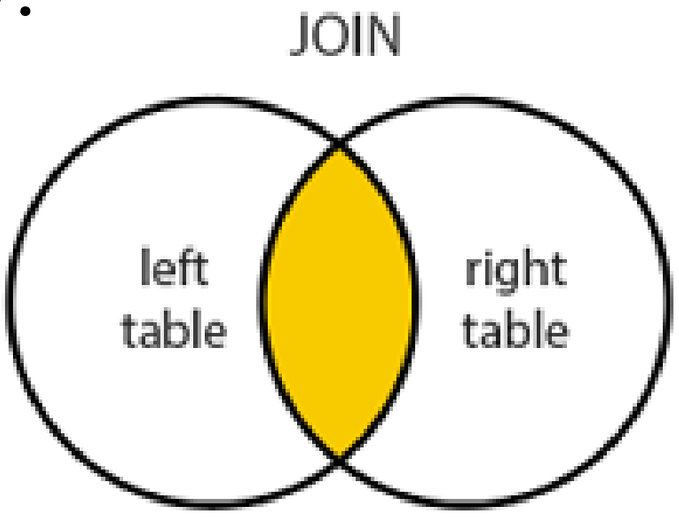
Q 7. What are the different types of normalization?

Ans: The different types of normalization are:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF) or Project-Join Normal Form (PJNF)

Q 8. What is a join in SQL?

Ans: A join is an operation used to combine rows from two or more tables based on related columns. It allows you to retrieve data from multiple tables simultaneously.



Q 9. What is the difference between DELETE and TRUNCATE in SQL?

Ans: The **DELETE statement** is used to remove specific rows from a table based on a condition. It can be rolled back and generates individual delete operations for each row.

TRUNCATE, on the other hand, is used to remove all rows from a table. It cannot be rolled back, and it is faster than DELETE as it deallocates the data pages instead of logging individual row deletions.

Q 10. What is the difference between UNION and UNION ALL?

Ans: UNION and UNION ALL are used to combine the result sets of two or more SELECT statements.

UNION removes duplicate rows from the combined result set.

whereas **UNION ALL** includes all rows, including duplicates.

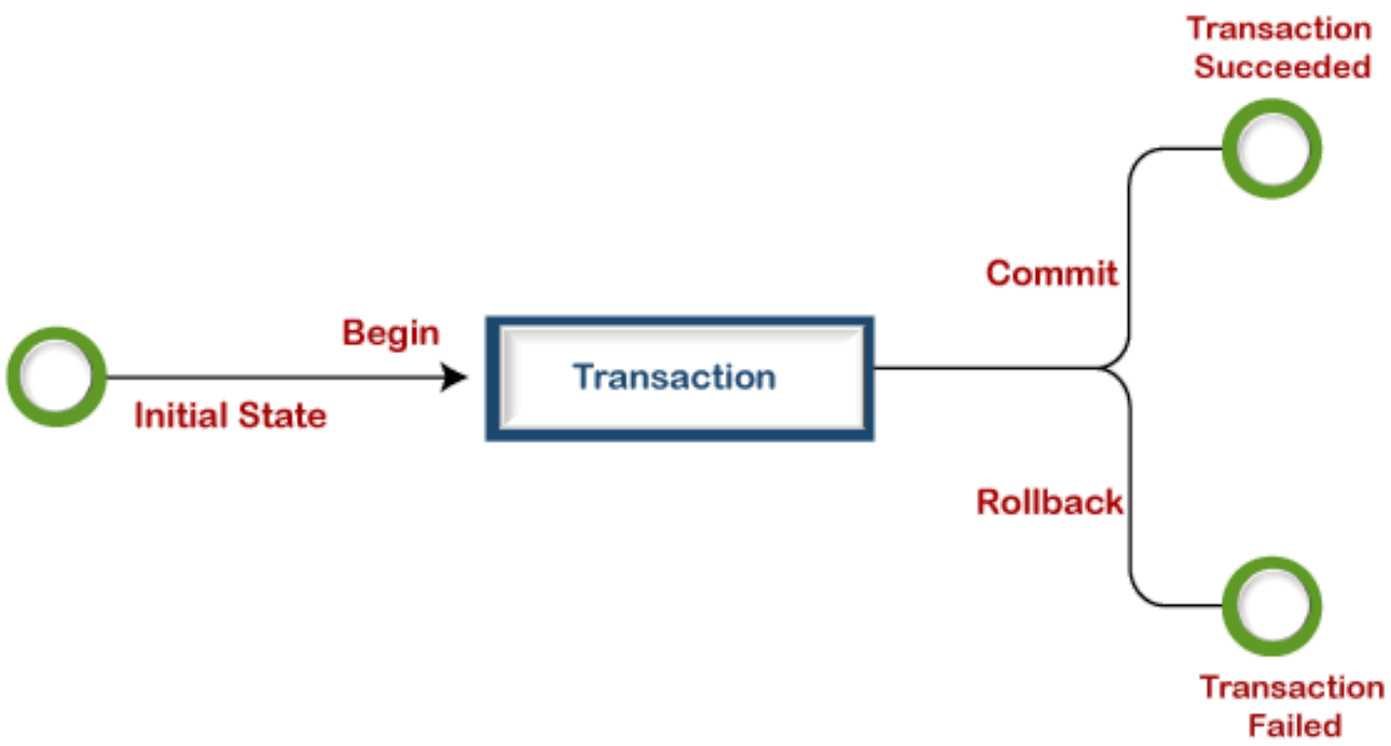
Q 11. What is the difference between the HAVING clause and the WHERE clause?

Ans: **The WHERE clause** is used to filter rows based on a condition before the data is grouped or aggregated. It operates on individual rows.

The HAVING clause, on the other hand, is used to filter grouped rows based on a condition after the data is grouped or aggregated using the GROUP BY clause.

Q 12. What is a transaction in SQL?

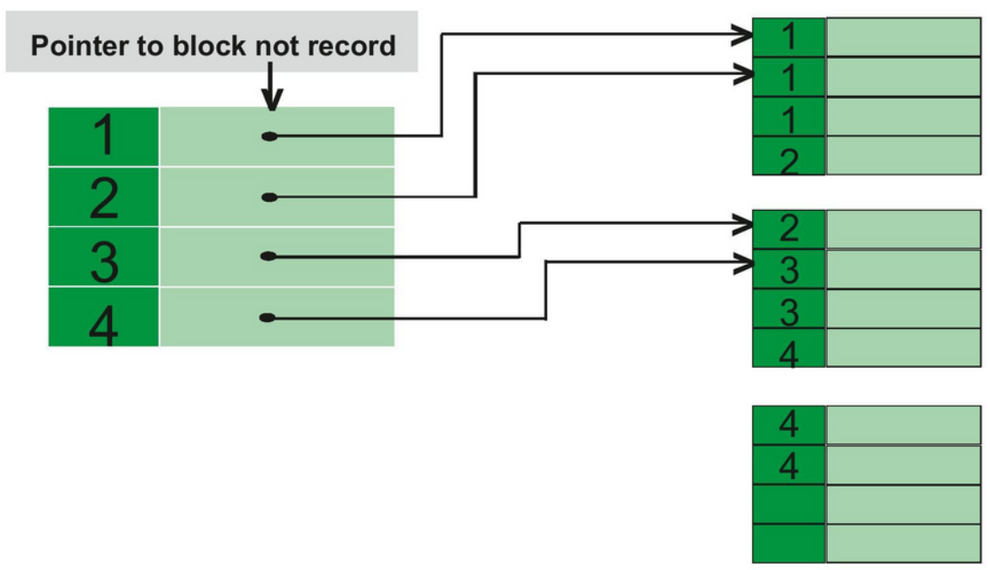
Ans: A transaction is a sequence of SQL statements that are executed as a single logical unit of work. It ensures data consistency and integrity by either committing all changes or rolling them back if an error occurs.



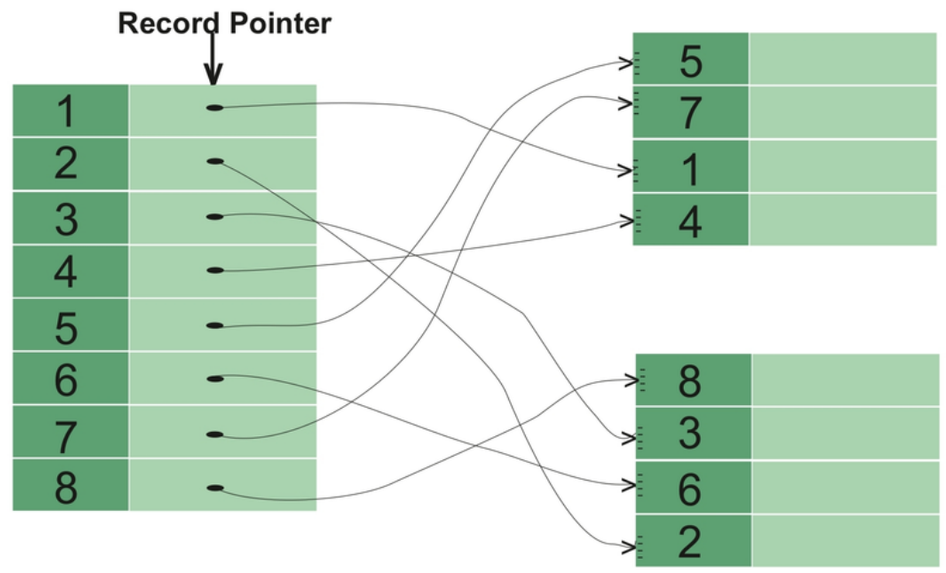
Q 13. What is the difference between a clustered and a non-clustered index?

Ans: A **clustered index** determines the physical order of data in a table. It changes the way the data is stored on disk and can be created on only one column. A table can have only one clustered index.

A **non-clustered index** does not affect the physical order of data in a table. It is stored separately and contains a pointer to the actual data. A table can have multiple non-clustered indexes.



Clustered Index



Non-Clustered Index

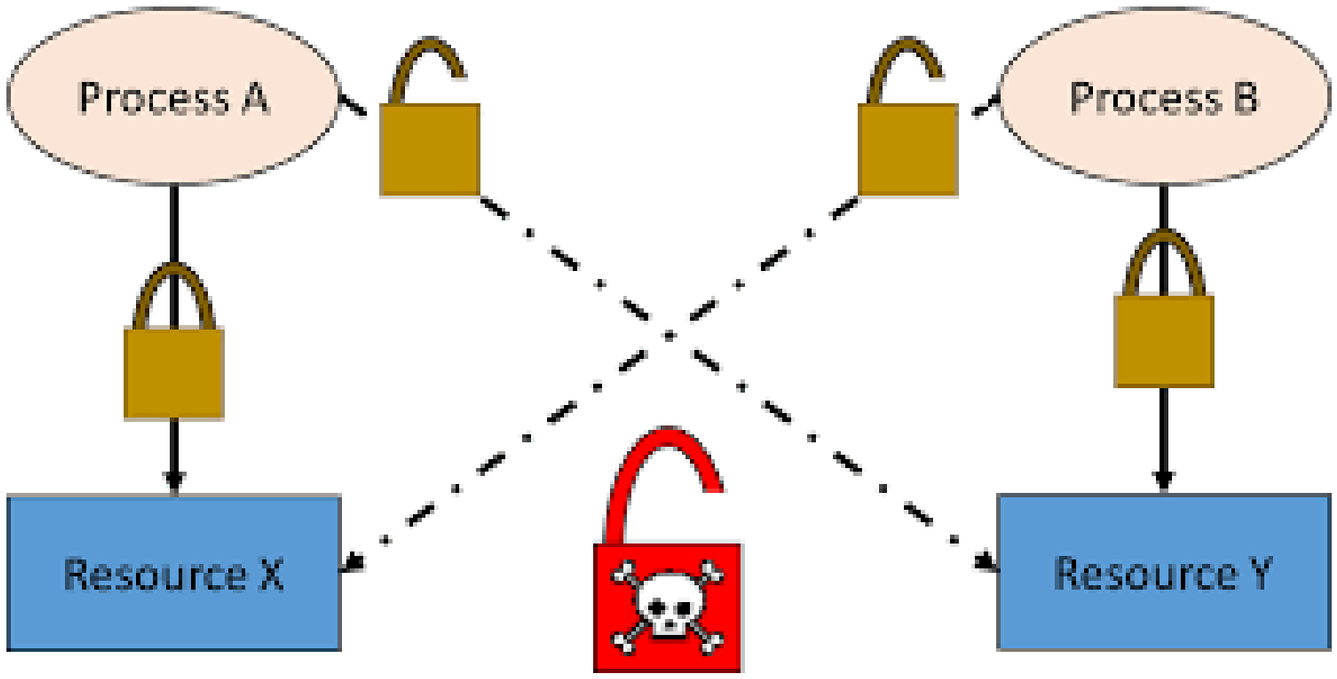
Q 14. What is ACID in the context of database transactions?

Ans: ACID stands for Atomicity, Consistency, Isolation, and Durability. It is a set of properties that guarantee reliable processing of database transactions.

- **Atomicity** ensures that a transaction is treated as a single unit of work, either all or none of the changes are applied.
- **Consistency** ensures that a transaction brings the database from one valid state to another.
- **Isolation** ensures that concurrent transactions do not interfere with each other.
- **Durability** ensures that once a transaction is committed, its changes are permanent and survive system failures.

Q 15. What is a deadlock?

Ans: A deadlock occurs when two or more transactions are waiting for each other to release resources, resulting in a circular dependency. As a result, none of the transactions can proceed, and the system may become unresponsive.



Q 16. What is the difference between a database and a schema?

Ans: A database is a container that holds multiple objects, such as tables, views, indexes, and procedures. It represents a logical grouping of related data.

A schema, on the other hand, is a container within a database that holds objects and defines their ownership. It provides a way to organize and manage database objects.

Q 17. What is the difference between a temporary table and a table variable?

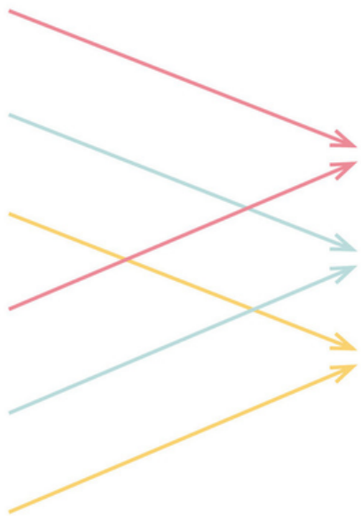
Ans: A temporary table is a table that is created and exists only for the duration of a session or a transaction. It can be explicitly dropped or is automatically dropped when the session or transaction ends.

A table variable is a variable that can store a table-like structure in memory. It has a limited scope within a batch, stored procedure, or function. It is automatically deallocated when the scope ends.

Q 18. What is the purpose of the GROUP BY clause?

Ans: The GROUP BY clause is used to group rows based on one or more columns in a table. It is typically used in conjunction with aggregate functions, such as SUM, AVG, COUNT, etc., to perform calculations on grouped data.

title	genre	qty
book 1	adventure	4
book 2	fantasy	5
book 3	romance	2
book 4	adventure	3
book 5	fantasy	3
book 6	romance	1



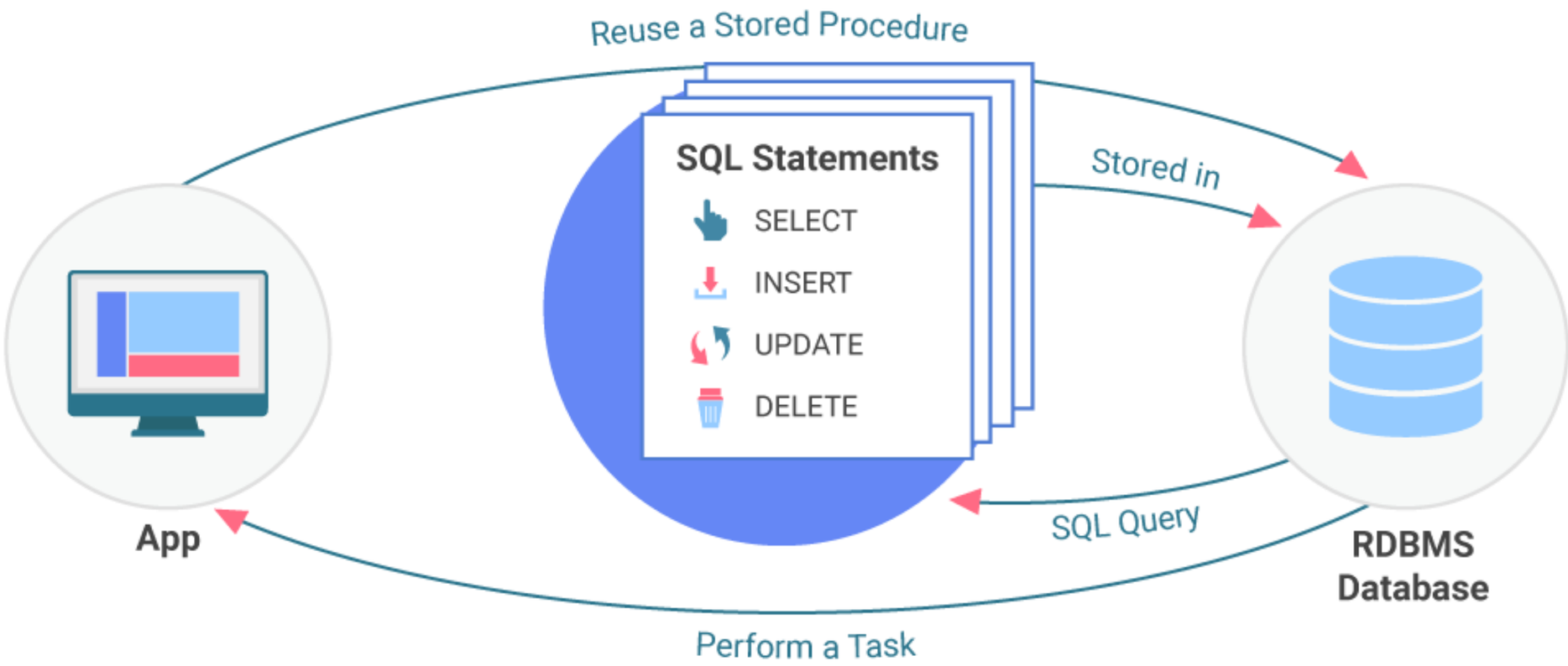
genre	total
adventure	7
fantasy	8
romance	3

Q 19. What is the difference between CHAR and VARCHAR data types?

Ans: CHAR is a fixed-length string data type, while VARCHAR is a variable-length string data type.

Q 20. What is a stored procedure?

Ans: A stored procedure is a set of SQL statements that are stored in the database and can be executed repeatedly. It provides code reusability and better performance.

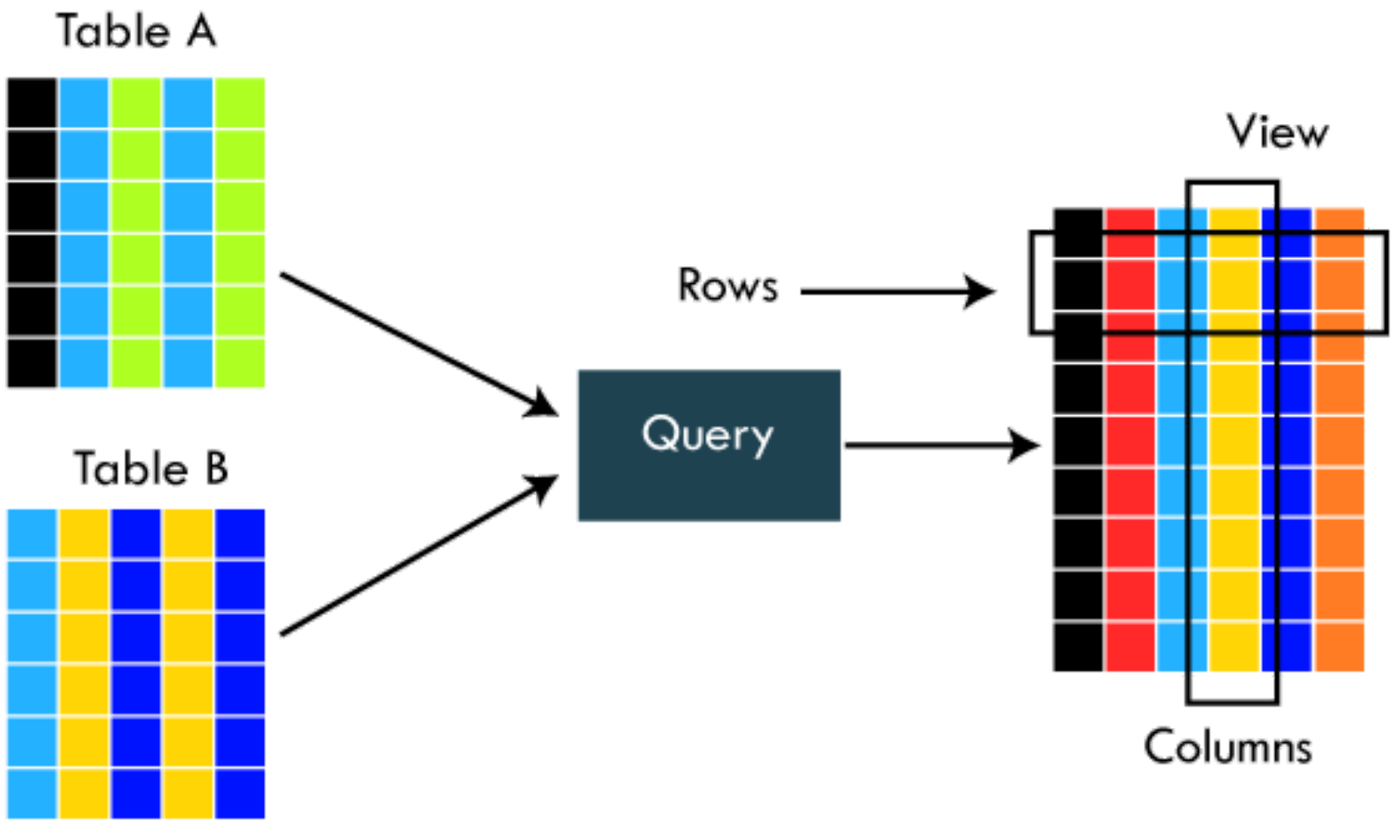


Q 21. What is a subquery?

Ans: A subquery is a query nested inside another query. It is used to retrieve data based on the result of an inner query.

Q 22. What is a view?

Ans: A view is a virtual table based on the result of an SQL statement. It allows users to retrieve and manipulate data as if



Q 23. What is the difference between a cross join and an inner join?

Ans: A cross join (Cartesian product) returns the combination of all rows from two or more tables.

An inner join returns only the matching rows based on a join condition.

Q 24. What is the purpose of the COMMIT statement?

Ans: The COMMIT statement is used to save changes made in a transaction permanently. It ends the transaction and makes the changes visible to other users.

Q 25. What is the purpose of the ROLLBACK statement?

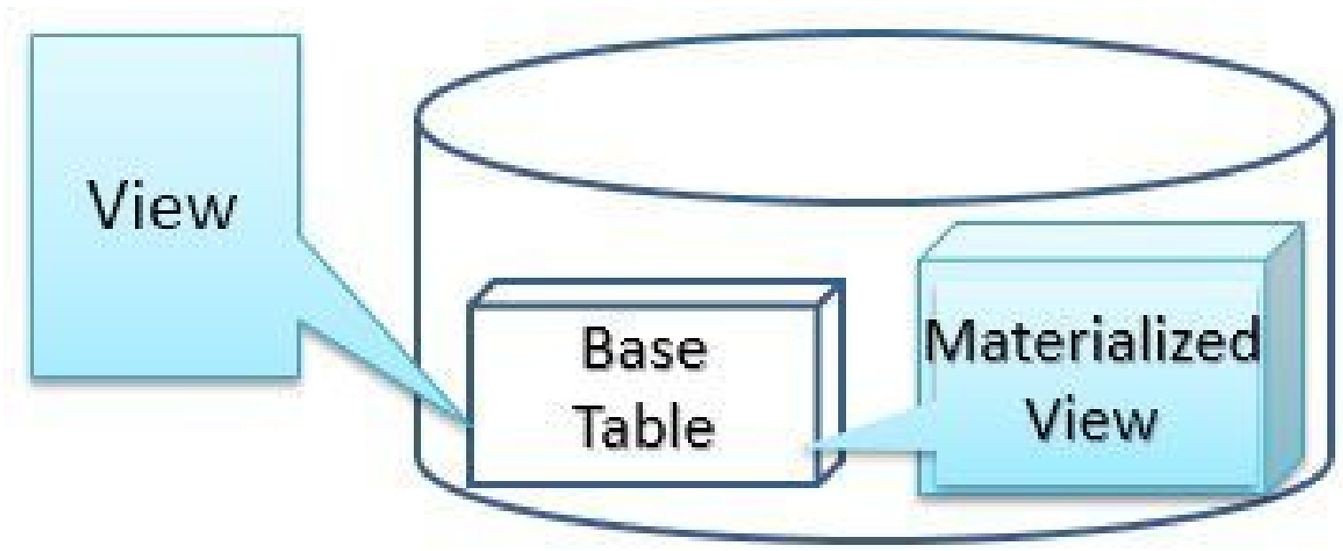
Ans: The ROLLBACK statement is used to undo changes made in a transaction. It reverts the database to its previous state before the transaction started.

Q 26. What is the purpose of the NULL value in SQL?

Ans: NULL represents the absence of a value or unknown value. It is different from zero or an empty string and requires special handling in SQL queries.

Q 27. What is the difference between a view and a materialized view?

Ans: A materialized view is a physical copy of the view's result set stored in the database, which is updated periodically. It improves query performance at the cost of data freshness.



Q 28. What is a correlated subquery?

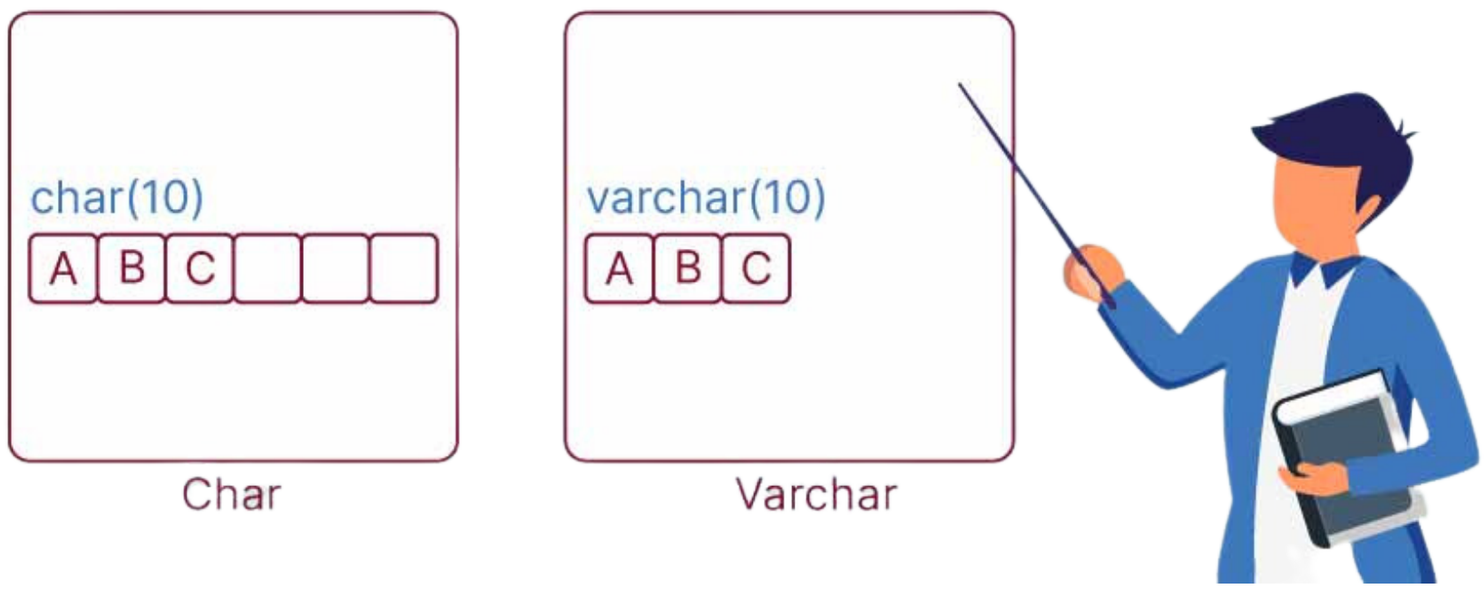
Ans: A correlated subquery is a subquery that refers to a column from the outer query. It executes once for each row processed by the outer query.

Q 29. What is the purpose of the DISTINCT keyword?

Ans: The DISTINCT keyword is used to retrieve unique values from a column or combination of columns in a SELECT statement.

Q 30. What is the difference between the CHAR and VARCHAR data types?

Ans: CHAR stores fixed-length character strings, while VARCHAR stores variable-length character strings. The storage size of CHAR is constant, while VARCHAR adjusts dynamically.



Q 31. What is the difference between the IN and EXISTS operators?

Ans: The IN operator checks for a value within a set of values or the result of a subquery. The EXISTS operator checks for the existence of rows returned by a subquery.

Q 32. What is the purpose of the TRIGGER statement?

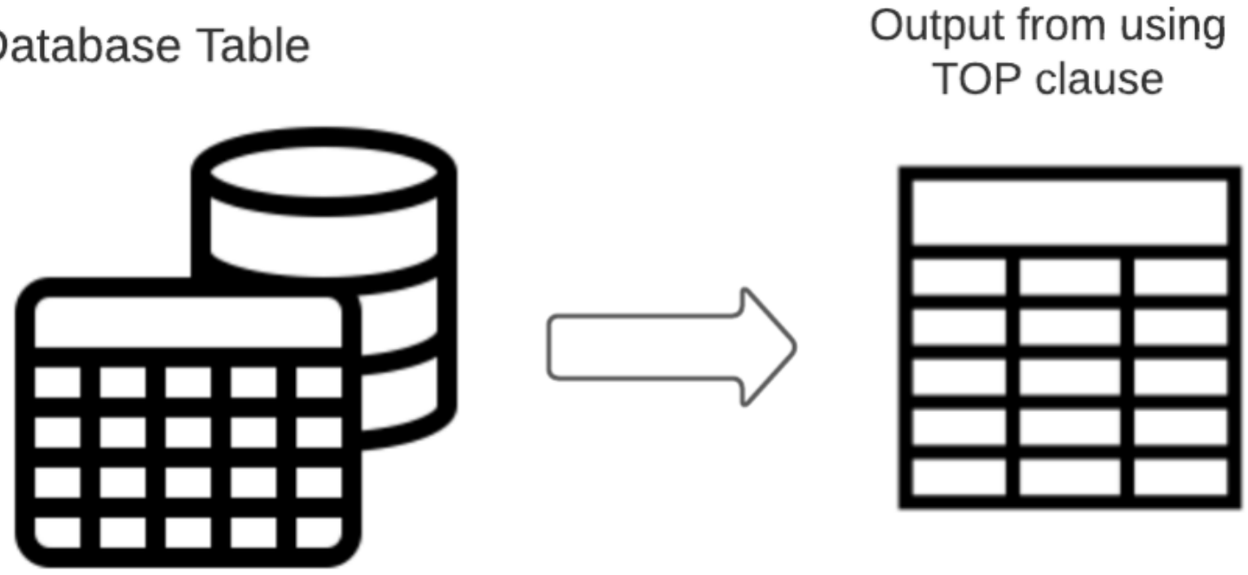
Ans: The TRIGGER statement is used to associate a set of SQL statements with a specific event in the database. It is executed automatically when the event occurs

Q 33. What is the difference between a unique constraint and a unique index?

Ans: A unique constraint ensures the uniqueness of values in one or more columns, while a unique index enforces the uniqueness and also improves query performance.

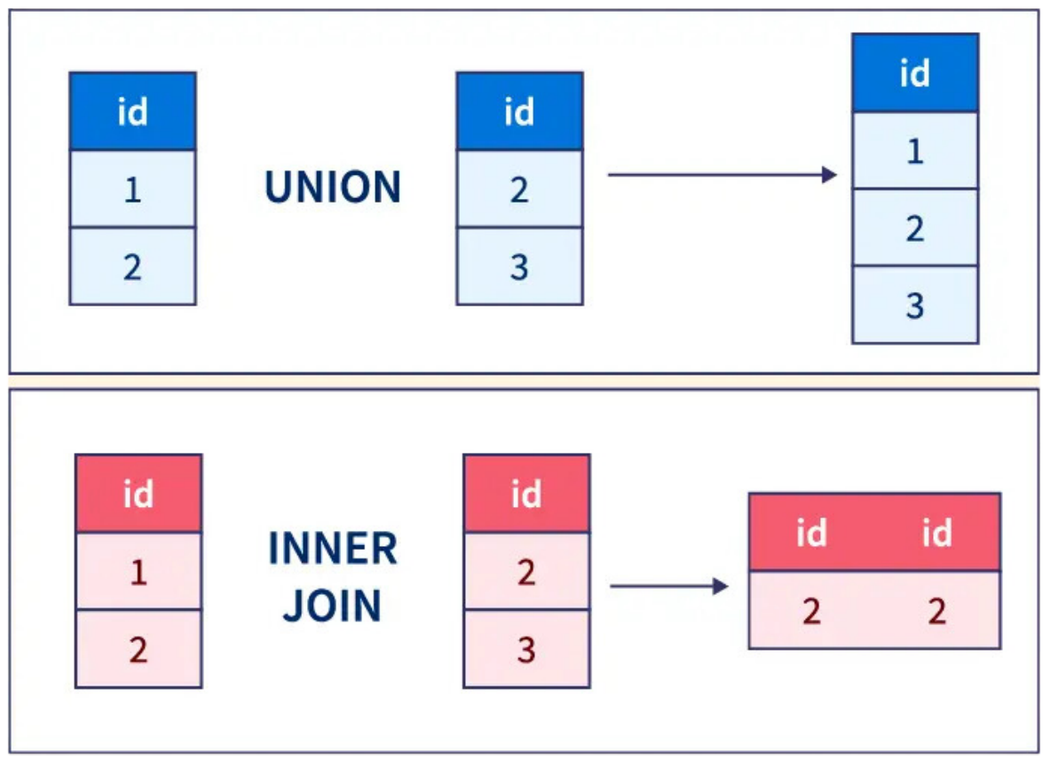
Q 34. What is the purpose of the TOP or LIMIT clause?

Ans: The TOP (in SQL Server) or LIMIT (in MySQL) clause is used to limit the number of rows returned by a query. It is often used with an ORDER BY clause.



Q 35. What is the difference between the UNION and JOIN operators?

Ans: UNION combines the result sets of two or more SELECT statements vertically, while JOIN combines columns from two or more tables horizontally based on a join condition.



Q 36. What is a data warehouse?

Ans: A data warehouse is a large, centralized repository that stores and manages data from various sources. It is designed for efficient reporting, analysis, and business intelligence purposes.

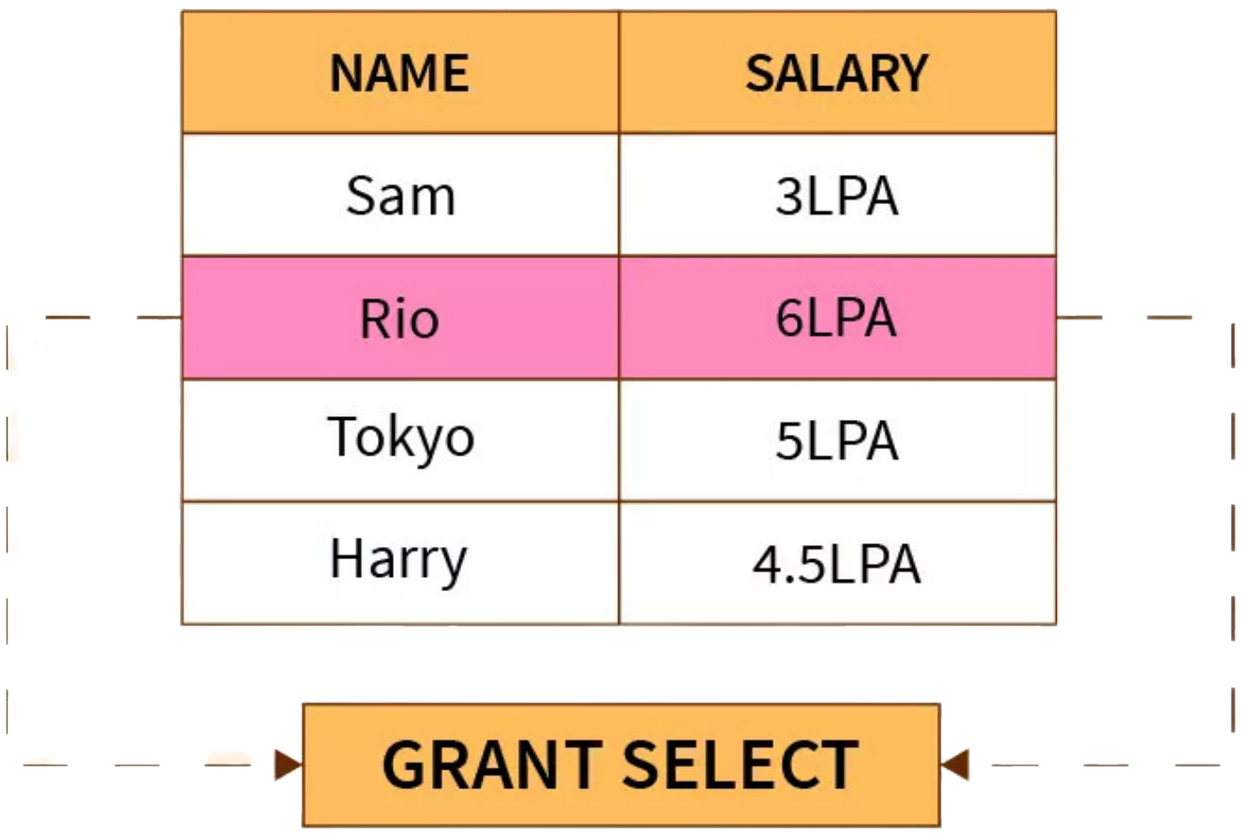
Q 37. What is the difference between a primary key and a candidate key?

Ans: A primary key is a chosen candidate key that uniquely identifies a row in a table.

A candidate key is a set of one or more columns that could potentially become the primary key.

Q 38. What is the purpose of the GRANT statement?

Ans: The GRANT statement is used to grant specific permissions or privileges to users or roles in a database.



Q 39. What is a correlated update?

Ans: A correlated update is an update statement that refers to a column from the same table in a subquery. It updates values based on the result of the subquery for each row.

Q 40. What is the purpose of the CASE statement?

Ans: The CASE statement is used to perform conditional logic in SQL queries. It allows you to return different values based on specified conditions.

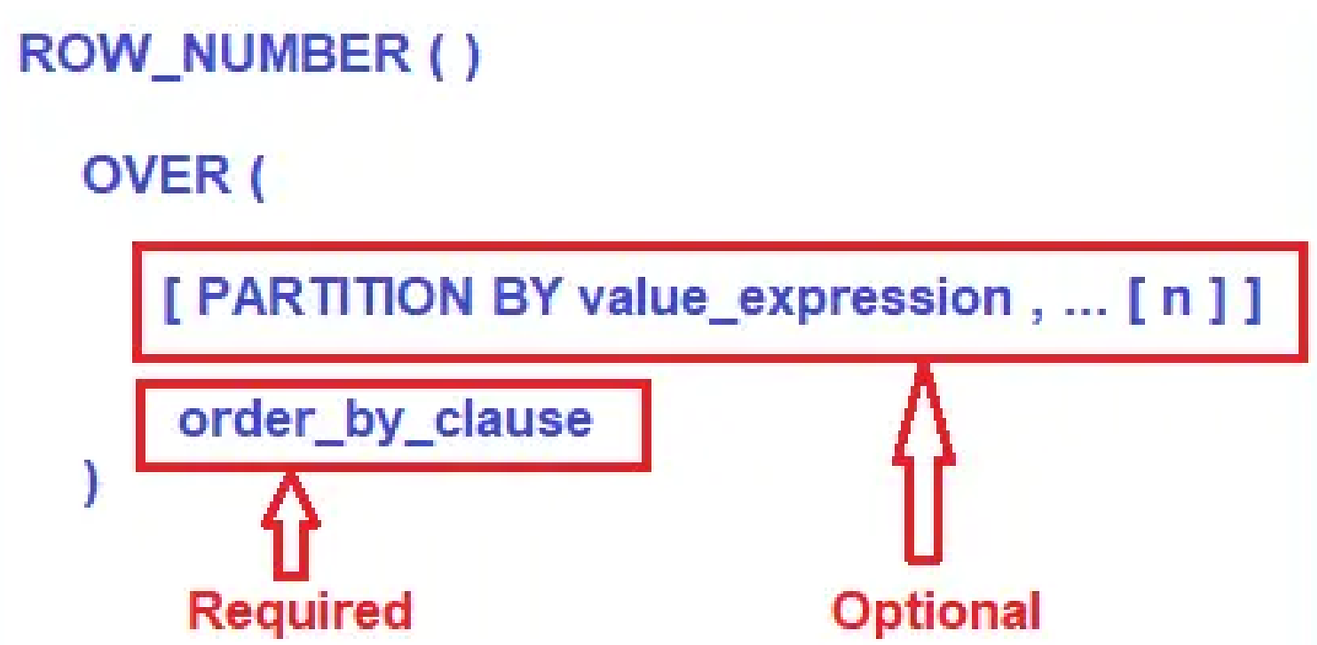
Q 41. What is the purpose of the COALESCE function?

Ans: The COALESCE function returns the first non-null expression from a list of expressions. It is often used to handle null values effectively.

Q 42. What is the purpose of the ROW_NUMBER() function?

Ans: The ROW_NUMBER() function assigns a unique incremental number to each row in the result set.

It is commonly used for pagination or ranking purposes.ll values effectively.



Q 43. What is the difference between a natural join and an inner join?

Ans: A natural join is an inner join that matches rows based on columns with the same name in the joined tables. It is automatically determined by the database.

Q 44. What is the purpose of the CASCADE DELETE constraint?

Ans: The CASCADE DELETE constraint is used to automatically delete related rows in child tables when a row in the parent table is deleted.

Q 45. What is the purpose of the ALL keyword in SQL?

Ans: The CASCADE DELETE constraint is used to automatically delete related rows in child tables when a row in the parent table is deleted.

Table: Teachers

id	name	age
1	Peter	32
2	Megan	43
3	Rose	29
4	Linda	30
5	Mary	41

Table: Students

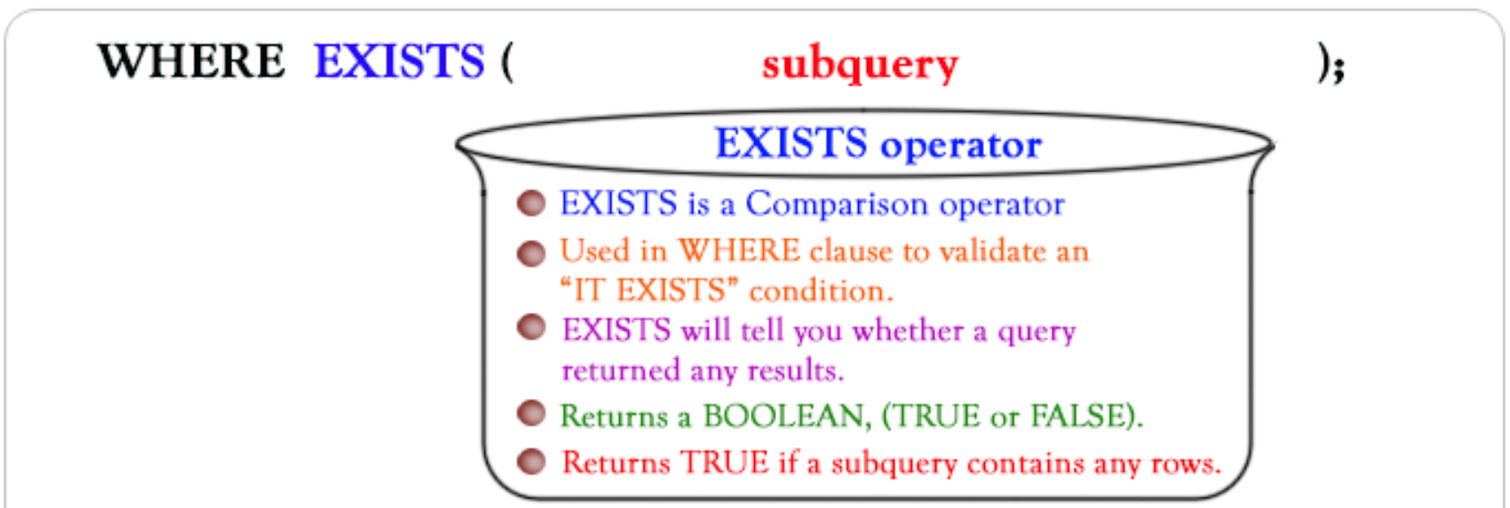
id	name	age
1	Harry	23
2	Jack	42
3	Joe	32
4	Dent	23
5	Bruce	40

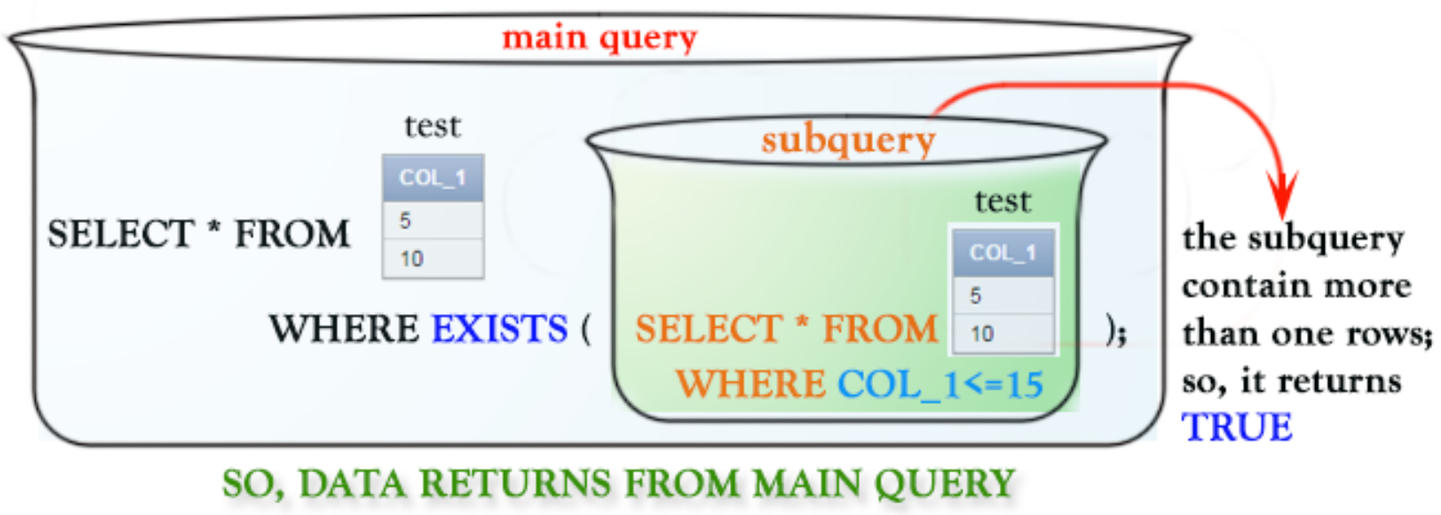
```
SELECT *
FROM Teachers
WHERE age > ALL (
   SELECT age
   FROM Students
);
```

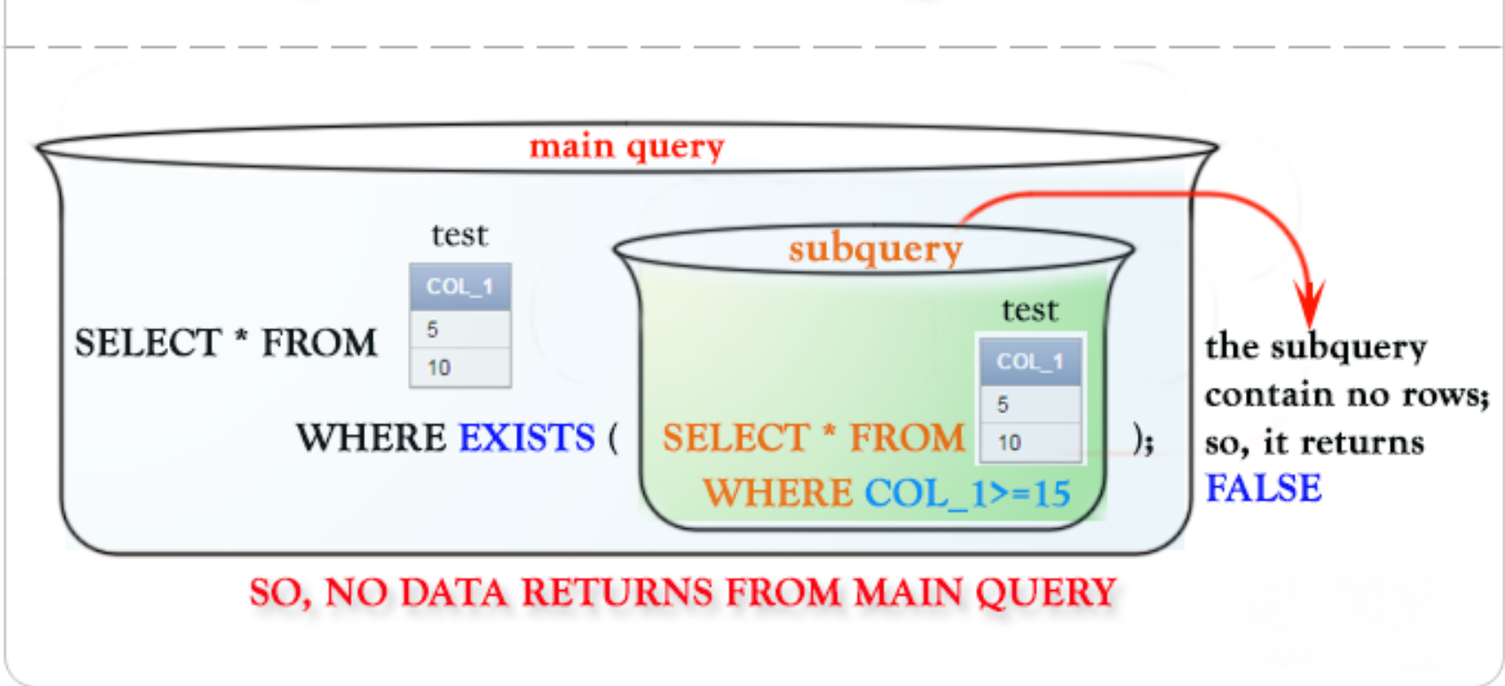
id	name	age
2	Megan	43

Q 46. What is the difference between the EXISTS and NOT EXISTS operators?

Ans: The EXISTS operator returns true if a subquery returns any rows, while the NOT EXISTS operator returns true if a subquery returns no rows.







Q 47. What is the purpose of the CROSS APPLY operator?

Ans: The CROSS APPLY operator is used to invoke a table-valued function for each row of a table expression. It returns the combined result set.

Q 48. What is a self-join?

Ans: A self-join is a join operation where a table is joined with itself. It is useful when you want to compare rows within the same table based on related columns. It requiresbined result set.

Name	Age	Location	ID
John	35	California	12698
Harry	24	Los Angeles	12699
Smith	32	Arizona	12700
Gary	45	New Jersey	12701

Ref ID	Name	Age	Location
12701	Gary	45	New Jersey
12699	Harry	24	Los Angeles
12698	Smith	32	Arizona
12700	Gary	45	New Jersey

Q 49. What is an ALIAS command?

Ans: ALIAS command in SQL is the name that can be given to any table or a column. This alias name can be referred in WHERE clause to identify a particular table or a column.

Q 50. Why are SQL functions used?

Ans: SQL functions are used for the following purposes:

- To perform some calculations on the data
- To modify individual data items
- To manipulate the output
- To format dates and numbers
- To convert the data types