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 softwares.
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 creation of table or after creationg 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);
```

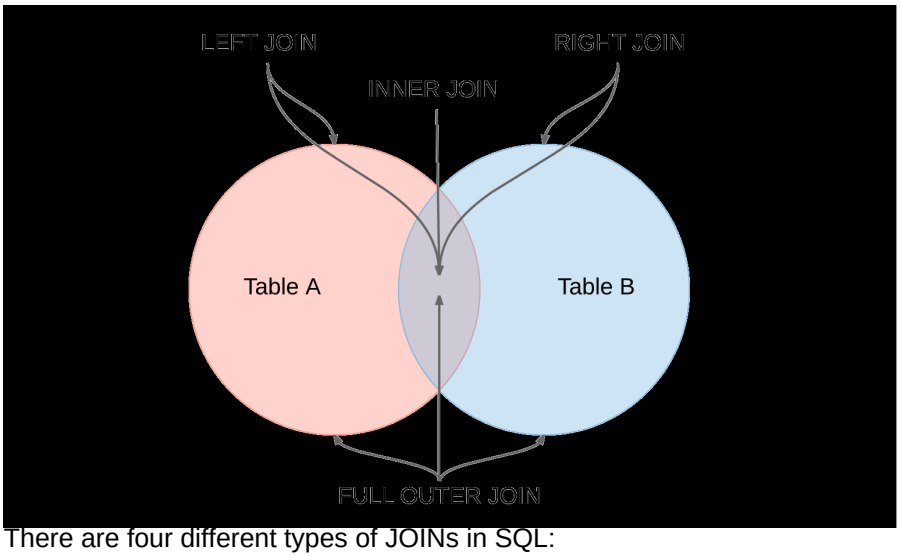
10. What is a Foreign Key?

A FOREIGN KEY comprises of single or collection of fields in a table that essentially refer 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 labelled as the child table, and the table containing the candidate key is labelled 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.

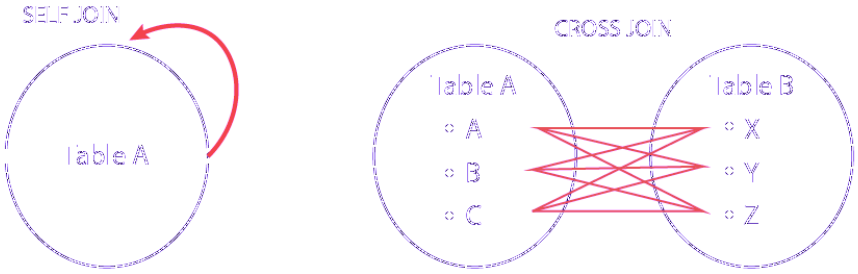
```
SELECT A.emp_id AS "Emp_ID",A.emp_name AS "Employee",
B.emp_id AS "Sup_ID",B.emp_name AS "Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

---

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 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 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 correspond 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 index 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. 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 to 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 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 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 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 conjuction 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 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?

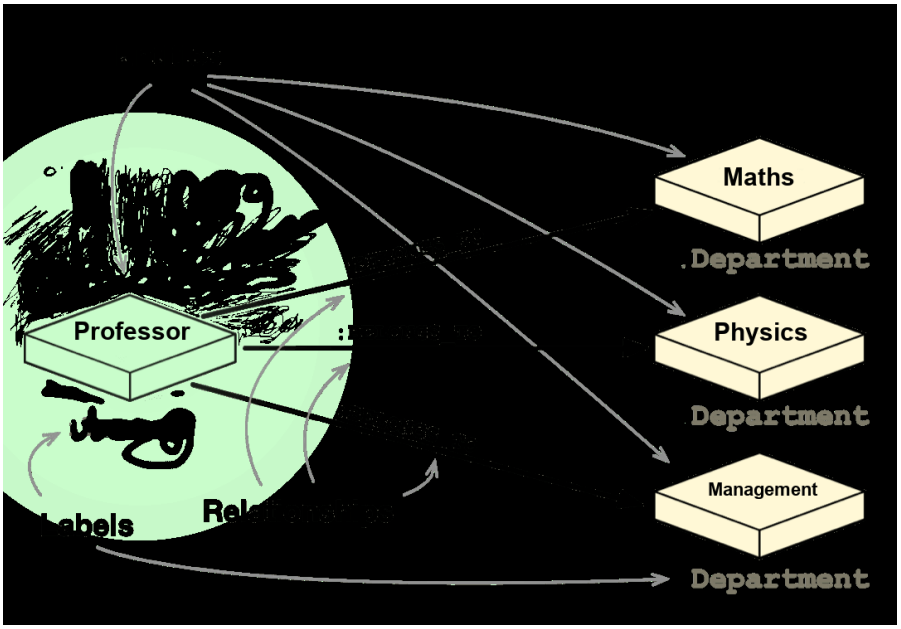
A database cursor is a control structure that allows for traversal of records in a database. Cursors, in addition, facilitates processing after traversal, such as retrieval, addition and deletion of database records. They can be viewed as a pointer to one row in a set of rows.
Working with SQL Cursor
- DECLARE a cursor after any variable declaration. The cursor declaration must always be associated with a SELECT Statement.
- Open cursor to initialize the result set. The OPEN statement must be called before fetching rows from the result set.
- FETCH statement to retrieve and move to the next row in the result set.
- Call the CLOSE statement to deactivate the cursor.
- 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 employees 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.

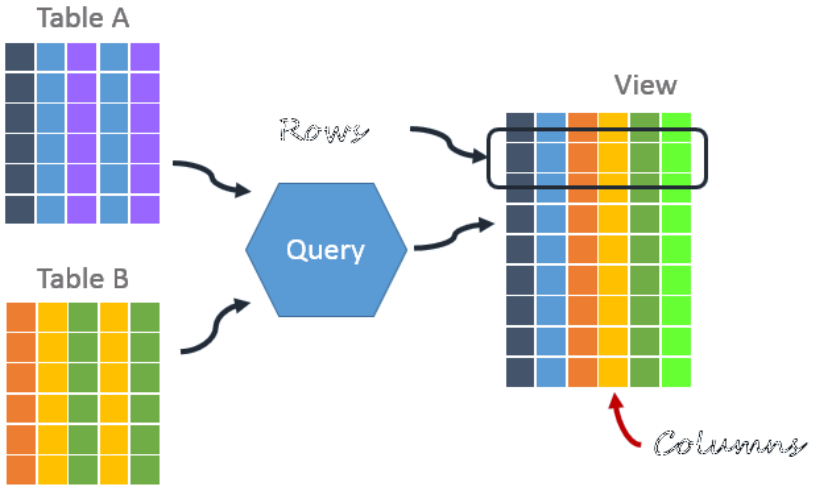
One-to-One - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
One-to-Many & Many-to-One - This is the most commonly used relationship where a record in a table is associated with multiple records 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 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 which 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 query processor by an over-normalized structure.

29. What are the various forms of Normalization?

1. 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 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.
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 values 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.

2. Second Normal Form
A relation is in second normal form if it satisfies the conditions for 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.
Consider the above example. As we can observe, Students Table in 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 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
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)

3. Third Normal Form
A relation is said to be in the third normal form, if it satisfies the conditions for 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.
Consider the Students Table in the above example. As we can observe, Students Table in 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 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)
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.

4. 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 the 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. Same cannot be said for the Books Table and there can be several books with common Book Names and 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 is 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 given length.
ROUND() - Calculates the round off integer value for a numeric field (or decimal point values).
NOW() - Returns the current data & time.
FORMAT() - Sets the format to display a collection of values.

---

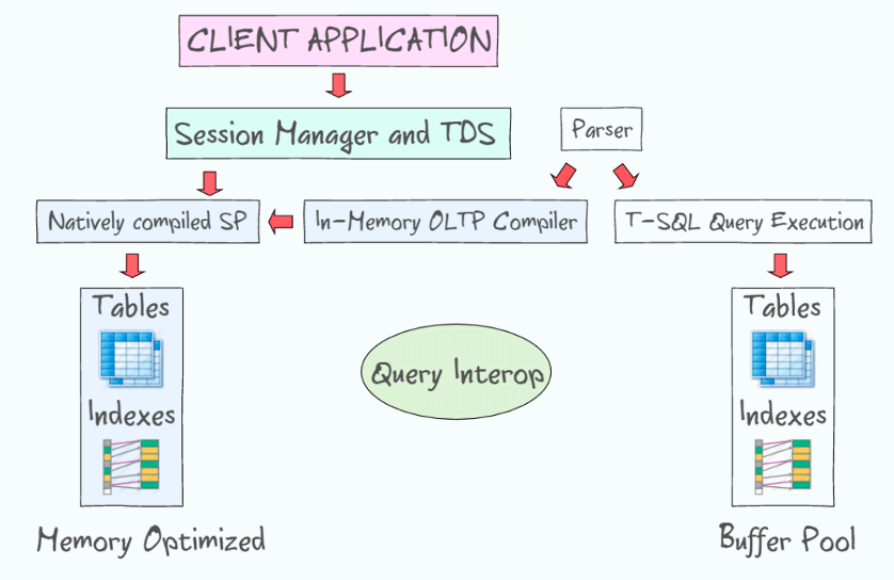
34. What is User-defined function? What are its various types?

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.

---

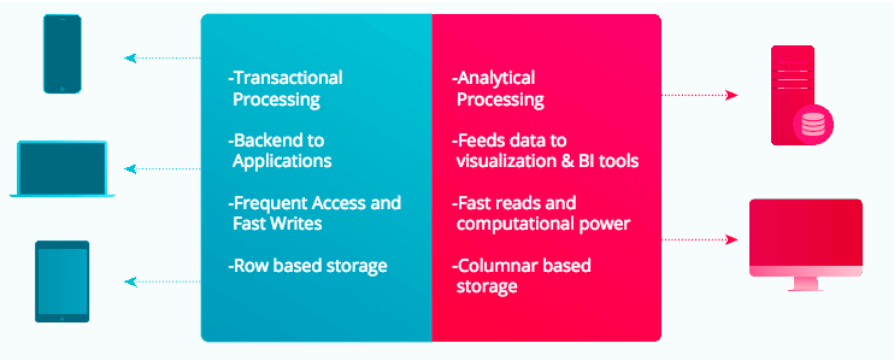
35. What is OLTP?

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



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. Number of transactions per second acts as an effective measure for such systems.
OLAP stands for Online Analytical Processing, a class of software programs which are characterized by 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.



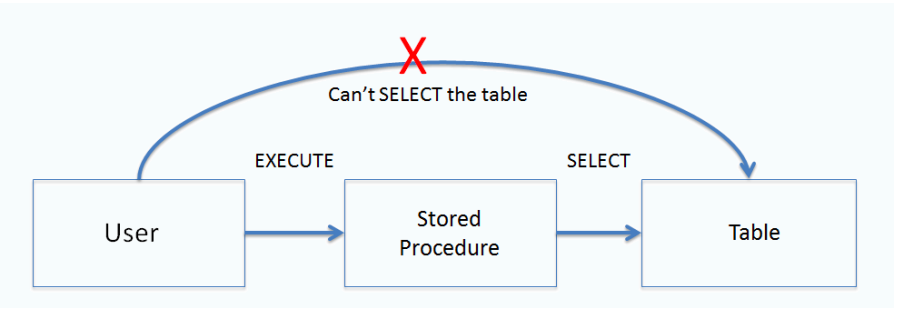
37. What is Collation? What are the different types of Collation Sensitivity?

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 which 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 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 '____'
```

42. What does UNION do? What is the difference between UNION and UNION ALL?

UNION merges the contents of two structurally-compatible tables into a single combined table. The difference between UNION and UNION ALL is that UNION will omit duplicate records whereas UNION ALL will include duplicate records.
It is important to note that the performance of UNION ALL will typically be better than UNION, since UNION requires the server to do the additional work of removing any duplicates. So, in cases where is is certain that there will not be any duplicates, or where having duplicates is not a problem, use of UNION ALL would be recommended for performance reasons.

---

43. Given the following tables:
sql> SELECT * FROM runners;
+----+--------------+
| id | name |
+----+--------------+
1	John Doe
2	Jane Doe
3	Alice Jones
4	Bobby Louis
5	Lisa Romero
+----+--------------+
sql> SELECT * FROM races;
+----+----------------+-----------+
| id | event | winner_id |
+----+----------------+-----------+
1	100 meter dash	2
2	500 meter dash	3
3	cross-country	2
4	triathalon	NULL
+----+----------------+-----------+
What will be the result of the query below?
SELECT * FROM runners WHERE id NOT IN (SELECT winner_id FROM races)
Explain your answer and also provide an alternative version of this query that will avoid the issue that it exposes.

Surprisingly, given the sample data provided, the result of this query will be an empty set. The reason for this is as follows: If the set being evaluated by the SQL NOT IN condition contains any values that are null, then the outer query here will return an empty set, even if there are many runner ids that match winner_ids in the races table.
Knowing this, a query that avoids this issue would be as follows:
SELECT * FROM runners WHERE id NOT IN (SELECT winner_id FROM races WHERE winner_id IS NOT null)
Note, this is assuming the standard SQL behavior that you get without modifying the default ANSI_NULLS setting.

---

44. Given two tables created and populated as follows:
CREATE TABLE dbo.envelope(id int, user_id int);
CREATE TABLE dbo.docs(idnum int, pageseq int, doctext varchar(100));
INSERT INTO dbo.envelope
VALUES
(1,1),
(2,2),
(3,3);
INSERT INTO
dbo.docs(idnum,pageseq) VALUES
(1,5),
(2,6),
(null,0);
What will the result be from the following query:
UPDATE docs SET doctext=pageseq
FROM docs INNER JOIN envelope
ON envelope.id=docs.idnum
WHERE EXISTS (
SELECT 1 FROM dbo.docs
WHERE id=envelope.id
);
Explain your answer.

The result of the query will be as follows:
idnum pageseq doctext
1 5 5
2 6 6
NULL 0 NULL
The EXISTS clause in the above query is a red herring. It will always be true since ID is not a member of dbo.docs. As such, it will refer to the envelope table comparing itself to itself!
The idnum value of NULL will not be set since the join of NULL will not return a result when attempting a match with any value of envelope.

---

45. Given a table TBL with a field Nmbr that has rows with the following values:
1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1
Write a query to add 2 where Nmbr is 0 and add 3 where Nmbr is 1.

update TBL
set Nmbr = case when Nmbr = 0 then Nmbr+2 else Nmbr+3 end;

---

46. What is an execution plan? When would you use it? How would you view the execution plan?

An execution plan is basically a road map that graphically or textually shows the data retrieval methods chosen by the SQL server's query optimizer for a stored procedure or ad hoc query. Execution plans are very useful for helping a developer understand and analyze the performance characteristics of a query or stored procedure, since the plan is used to execute the query or stored procedure.
In many SQL systems, a textual execution plan can be obtained using a keyword such as EXPLAIN, and visual representations can often be obtained as well. In Microsoft SQL Server, the Query Analyzer has an option called "Show Execution Plan" (located on the Query drop down menu). If this option is turned on, it will display query execution plans in a separate window when a query is run.

---

47. List and explain each of the ACID properties that collectively guarantee that database transactions are processed reliably.

ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably. They are defined as follows:
Atomicity. Atomicity requires that each transaction be “all or nothing”: if one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. An atomic system must guarantee atomicity in each and every situation, including power failures, errors, and crashes.
Consistency. The consistency property ensures that any transaction will bring the database from one valid state to another. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.
Isolation. The isolation property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. Providing isolation is the main goal of concurrency control. Depending on concurrency control method (i.e. if it uses strict - as opposed to relaxed - serializability), the effects of an incomplete transaction might not even be visible to another transaction.
Durability. Durability means that once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors. In a relational database, for instance, once a group of SQL statements execute, the results need to be stored permanently (even if the database crashes immediately thereafter). To defend against power loss, transactions (or their effects) must be recorded in a non-volatile memory.

---

48. How can you select all the even number records from a table? All the odd number records?

To select all the even number records from a table:
Select * from table where id % 2 = 0
To select all the odd number records from a table:
Select * from table where id % 2 != 0

---

49. What is the difference between the RANK() and DENSE_RANK() functions? Provide an example.

The only difference between the RANK() and DENSE_RANK() functions is in cases where there is a “tie”; i.e., in cases where multiple values in a set have the same ranking. In such cases, RANK() will assign non-consecutive “ranks” to the values in the set (resulting in gaps between the integer ranking values when there is a tie), whereas DENSE_RANK() will assign consecutive ranks to the values in the set (so there will be no gaps between the integer ranking values in the case of a tie).
For example, consider the set {25, 25, 50, 75, 75, 100}. For such a set, RANK() will return {1, 1, 3, 4, 4, 6} (note that the values 2 and 5 are skipped), whereas DENSE_RANK() will return {1,1,2,3,3,4}.

---

50. What is the difference between the WHERE and HAVING clauses?	When GROUP BY is not used, the WHERE and HAVING clauses are essentially equivalent. However, when GROUP BYis used: The WHERE clause is used to filter records from a result. The filtering occurs before any groupings are made. The HAVING clause is used to filter values from a group (i.e., to check conditions after aggregation into groups has been performed).
51. Given a table Employee having columns empName and empId, what will be the result of the SQL query below? select empName from Employee order by 2 desc;	“Order by 2” is only valid when there are at least two columns being used in select statement. However, in this query, even though the Employee table has 2 columns, the query is only selecting 1 column name, so “Order by 2” will cause the statement to throw an error while executing the above sql query.
52. What will be the output of the below query, given an Employee table having 10 records? BEGIN TRAN TRUNCATE TABLE Employees ROLLBACK SELECT * FROM Employees	This query will return 10 records as TRUNCATE was executed in the transaction. TRUNCATE does not itself keep a log but BEGIN TRANSACTION keeps track of the TRUNCATE command.
53. How do you get the last id without the max function?	In MySQL: select id from table order by id desc limit 1
54. What is the difference between IN and EXISTS?	IN: Works on List result set Doesn’t work on subqueries resulting in Virtual tables with multiple columns Compares every value in the result list Performance is comparatively SLOW for larger resultset of subquery EXISTS: Works on Virtual tables Is used with co-related queries Exits comparison when match is found Performance is comparatively FAST for larger resultset of subquery
55. How do you copy data from one table to another table ?	INSERT INTO table2 (column1, column2, column3, ...) SELECT column1, column2, column3, ... FROM table1 WHERE condition;

56. How to find a duplicate record? 1. duplicate records with one field 2. duplicate records with more than one field	duplicate records with one field SELECT name, COUNT(email) FROM users GROUP BY email HAVING COUNT(email) > 1 duplicate records with more than one field SELECT name, email, COUNT(*) FROM users GROUP BY name, email HAVING COUNT(*) > 1
57. What are triggers?	Triggers in SQL is kind of stored procedures used to create a response to a specific action performed on the table such as Insert, Update or Delete. You can invoke triggers explicitly on the table in the database. Action and Event are two main components of SQL triggers when certain actions are performed the event occurs in response to that action. CREATE TRIGGER name {BEFORE|AFTER} (event [OR..]} ON table_name [FOR [EACH] {ROW|STATEMENT}] EXECUTE PROCEDURE functionname {arguments}
58. What are different types of statements supported by SQL?	1) DDL (Data Definition Language): It is used to define the database structure such as tables. It includes three statements such as Create, Alter, and Drop. 2) DML (Data Manipulation Language): These statements are used to manipulate the data in records. Commonly used DML statements are Insert, Update, and Delete. 3) DCL (Data Control Language): These statements are used to set privileges such as Grant and Revoke database access permission to the specific user.
59. How do we use DISTINCT statement? What is its use?	The DISTINCT statement is used with the SELECT statement. If the records contain duplicate values then DISTINCT is used to select different values among duplicate records. SELECT DISTINCT column_name(s) FROM table_name;

60. What are transaction and its controls?

A transaction can be defined as the sequence task that is performed on databases in a logical manner to gain certain results. Operations performed like Creating, updating, deleting records in the database comes from transactions.
In simple word, we can say that a transaction means a group of SQL queries executed on database records.
There are 4 transaction controls such as
COMMIT: It is used to save all changes made through the transaction
ROLLBACK: It is used to roll back the transaction such as all changes made by the transaction are reverted back and database remains as before
SET TRANSACTION: Set the name of transaction
SAVEPOINT: It is used to set the point from where the transaction is to be rolled back

---

61. What are properties of the transaction?

Properties of transaction are known as ACID properties, such as
Atomicity: Ensures the completeness of all transactions performed. Checks whether every transaction is completed successfully if not then transaction is aborted at the failure point and the previous transaction is rolled back to its initial state as changes undone
Consistency: Ensures that all changes made through successful transaction are reflected properly on database
Isolation: Ensures that all transactions are performed independently and changes made by one transaction are not reflected on other
Durability: Ensures that the changes made in database with committed transactions persist as it is even after system failure

---

62. What is View in SQL?

A View can be defined as a virtual table that contains rows and columns with fields from one or more table.
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition

---

63. How we can update the view?

SQL CREATE and REPLACE can be used for updating the view.
Following query syntax is to be executed to update the created view
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition

---

64. Explain the working of SQL Privileges?

SQL GRANT and REVOKE commands are used to implement privileges in SQL multiple user environments. The administrator of the database can grant or revoke privileges to or from users of database object like SELECT, INSERT, UPDATE, DELETE, ALL etc.
GRANT Command: This command is used provide database access to user apart from an administrator.
GRANT privilege_name
ON object_name
TO {user_name|PUBLIC|role_name}
[WITH GRANT OPTION];
In above syntax WITH GRANT OPTIONS indicates that the user can grant the access to another user too.
REVOKE Command: This command is used provide database deny or remove access to database objects.
REVOKE privilege_name
ON object_name
FROM {user_name|PUBLIC|role_name};

65. How many types of Privileges are available in SQL?

There are two types of privileges used in SQL, such as
System Privilege: System privileges deal with an object of a particular type and specifies the right to perform one or more actions on it which include Admin allows a user to perform administrative tasks, ALTER ANY INDEX, ALTER ANY CACHE GROUP CREATE/ALTER/DELETE TABLE, CREATE/ALTER/DELETE VIEW etc.
Object Privilege: This allows to perform actions on an object or object of another user(s) viz. table, view, indexes etc. Some of the object privileges are EXECUTE, INSERT, UPDATE, DELETE, SELECT, FLUSH, LOAD, INDEX, REFERENCES etc.

66. What is SQL Injection?

SQL Injection is a type of database attack technique where malicious SQL statements are inserted into an entry field of database such that once it is executed the database is opened for an attacker. This technique is usually used for attacking Data-Driven Applications to have an access to sensitive data and perform administrative tasks on databases.

67. What is the difference between SQL and PL/SQL?

SQL is a structured query language to create and access databases whereas PL/SQL comes with procedural concepts of programming languages.

68. What is the use of NVL function?

NVL function is used to convert the null value to its actual value.

69. What is the Cartesian product of table?

The output of Cross Join is called as a Cartesian product. It returns rows combining each row from the first table with each row of the second table. For Example, if we join two tables having 15 and 20 columns the Cartesian product of two tables will be 15×20=300 Rows.

70. What do you mean by Subquery?	Query within another query is called as Subquery. A subquery is called inner query which returns output that is to be used by another query.
71. What do we need to check in Database Testing?	Generally, in Database Testing following thing is need to be tested Database Connectivity Constraint Check Required Application Field and its size Data Retrieval and Processing With DML operations Stored Procedures Functional flow
72. What is Database White Box Testing?	Database White Box Testing involves Database Consistency and ACID properties Database triggers and logical views Decision Coverage, Condition Coverage, and Statement Coverage Database Tables, Data Model, and Database Schema Referential integrity rules
73. What is Database Black Box Testing?	Database Black Box Testing involves Data Mapping Data stored and retrieved Use of Black Box techniques such as Equivalence Partitioning and Boundary Value Analysis (BVA)
74. What is the syntax to add a record to a table?	To add a record in a table INSERT syntax is used. INSERT into table_name VALUES (value1, value2..);
75. How do you add a column to a table?	To add another column in the table following command has been used. ALTER TABLE table_name ADD (column_name);
76. Define COMMIT?	COMMIT saves all changes made by DML statements.
77. Can we rename a column in the output of SQL query?	Yes using the following syntax we can do this. SELECT column_name AS new_name FROM table_name;
78. What is a composite primary key?	Primary key created on more than one column is called composite primary key.
79. What are the advantages of Views?	Advantages of Views: Views restrict access to the data because the view can display selective columns from the table. Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing.

80. What is schema?	A schema is a collection of database objects of a User.
81. What is CTE?	A CTE or common table expression is an expression which contains temporary result set which is defined in a SQL statement.
82. What is a Record in a Database?	A record (also called a row of data) is an ordered set of related data in a table.
83. What is the difference between Cluster and Non-Cluster Index?	Clustered Index: It is used for easy retrieval of data from the database and it is faster. One table can only have one clustered index It alters the way records are stored in a database as it sorts out rows by the column which is set to be clustered index. Non-Clustered Index: It is slower compared to the Clustered index. One table can have multiple non clustered index It doesn’t alter the way it was sorted but it creates a separate object within a table which points back to the original table rows after searching.
84. What is the difference between an inner and outer join?	An inner join returns rows when there is at least some matching data between two (or more) tables that are being compared. An outer join returns rows from both tables that include the records that are unmatched from one or both the tables.
85. What is a NULL value?	A field with a NULL value is a field with no value. A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation. Assume, there is a field in a table is optional and it is possible to insert a record without adding a value to the optional field then the field will be saved with a NULL value.
86. What is the difference between NULL value, Zero, and Blank space?	As I mentioned earlier, Null value is field with no value which is different from zero value and blank space. Null value is a field with no value. Zero is a number Blank space is the value we provide. The ASCII value of space is CHAR(32).
87. What is a CHECK constraint?	A CHECK constraint is used to limit the value that is accepted by one or more columns. ‘Age’ field should contain only the value greater than 18. CREATE TABLE EMP_DETAILS(EmpID int NOT NULL, NAME VARCHAR (30) NOT NULL, Age INT CHECK (AGE > 18), PRIMARY KEY (EmpID));

88. Define the SELECT INTO statement.

The SELECT INTO statement copies data from one table into a new table. The new table will be created with the column-names and types as defined in the old table. You can create new column names using the AS clause.