Question 1: What is a Data warehouse?

-Answer: Data warehouse refers to a central repository of data from multiple sources of information. Those data are consolidated, transformed and made available for the mining as well as online processing.

Question 2: What is a Database?

Answer: A database is a collection of information in an organized form for faster and better access, storage and manipulation. It can also be defined as a collection of tables, schema, views, and other database objects.

Question 3: What is a Table in a Database?

Answer: A table is a database object used to store records in a field in the form of columns and rows that holds data.

Question 4: What are the popular Database Management Systems in the IT Industry?

Answer: Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Sybase, MongoDB, DB2, and Microsoft Access etc.

Question 5: What is SQL?

Answer: SQL stands for Structured Query Language, and it is used to communicate with the Database. This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

Question 6: What all can we do with SQL?

Answer: We can perform the following tasks :-
a. Retrieval of data
b. Updation of data
c. Insertion of data
d. Deletion of data

Question 7: What are the different Data Types in SQL?

Answer: DATA TYPES –

- CHAR(size) – Fixed length string (size can be from 0 – 255)Default 1.
- VARCHAR(size) - Variable length string (0 – 65535)
- INT(size) – medium length integer.
- Integer(size) – equal to INT
- FLOAT(*size*, *d*) - A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter.
- DATE - A date. Format: YYYY-MM-DD.

Question 8: What is the difference between CHAR and VARCHAR ?

Answer: Difference –

CHAR	**VARCHAR**
CHAR data type helps to store characters	VARCHAR data type store variable characters
Stores value in fixed length	Store variable length long with 1 or 2 byte
Maximum of 255 character long	Maximum 65535 characters long
Uses static memory allocation	Uses dynamic memory allocation

Question 9: What are the different types of SQL commands?

Answer: SQL commands are segregated into the following types:

- DDL – Data Definition Language
- DML – Data Manipulation Language
- DQL – Data Query Language
- DCL – Data Control Language
- TCL – Transaction Control Language

DDL Commands: Data Definition Language is used to define the database structure such as tables.

- CREATE: To create databases and database objects
- ALTER: To alter existing database objects
- DROP: To drop databases and databases objects
- TRUNCATE: To remove all records from a table but not its database structure
- RENAME: To rename database objects

DML Commands Data Manipulation Language is used to manipulate the data in records. Commonly used DML commands are Update, Insert and Delete. Sometimes SELECT command is also referred as a Data Manipulation Language.

- SELECT: To select specific data from a database
- INSERT: To insert new records into a table
- UPDATE: To update existing records
- DELETE: To delete existing records from a table

DCL Commands Data Control Language is used to control the privileges by granting and revoking database access permission

- GRANT: To provide user access
- DENY: To deny permissions to users
- REVOKE: To remove user access

Question 10: What is the difference between DELETE and TRUNCATE commands?

Answer: DELETE command is used to remove rows from the table, and WHERE clause can be used for conditional set of parameters. Commit and Rollback can be performed after delete statement.
TRUNCATE removes all rows from the table. Truncate operation cannot be rolled back.

Question 11: What is the difference between DROP and TRUNCATE commands?

Answer: DROP command removes a table and it cannot be rolled back from the database whereas TRUNCATE command removes all the rows from the table.

Question 12: What is the difference between DELETE , DROP ,TRUNCATE?

Answer: Difference –

DELETE	DROP	TRUNCATE
Delete command is used selectively delete rows form table using WHERE clause.	Drop command is used to remove rows and table.	Truncate command is used remove all the rows from the table.
Data can be roll back if commit performed.	Data can't be roll back.	Data can't be roll back.
If no where condition is specified it will remove all the rows.	No need of where condition.	No need of where condition.
Delete command won't reset the AUTO_INCREMENT automatically		Truncate command reset the AUTO_INCREMENT to start value.
e.g. DELETE FROM employee WHERE id <=10;	e.g. DROP TABLE employee;	e.g. TRUNCATE TABLE employee;

Question 13: Which is faster delete or truncate ?

Answer: Truncate command is faster than the delete command **,** since no undo log is maintained and has no dependencies.

Question 14: Can we drop all columns from a table?

Answer:

1. No , We can't delete all columns from table.
2. Column with check constraint also can't be dropped. To delete that column first we need to delete the constraint then the column.

Question 15: What are the important SQL aggregate functions?

Answer:

a. AVG() – returns average value in a set.
b. COUNT() – returns number of items in a set.
c. MAX() – returns maximum value in a set.
d. MIN() – returns minimum value in a set.
e. SUM() – returns summation of all or distinct values in a set.

Question 16: What is CLAUSE?

Answer: SQL clause is defined to limit the result set by providing condition to the query. This usually filters some rows from the whole set of records.
Example –
Query that has WHERE condition.

Query that has HAVING condition.

Query that has LIKE condition.

Question 17: What is the difference between 'HAVING' CLAUSE and a 'WHERE' CLAUSE?

Answer: HAVING clause can be used only with SELECT statement.
It is usually used in a GROUP BY clause and whenever GROUP BY is not used, HAVING behaves like a WHERE clause.
Having Clause is only used with the GROUP BY function in a query
whereas WHERE Clause is applied to each row before they are a part of the GROUP BY function in a query.

WHERE CLAUSE	HAVING CLAUSE
Where is used to filter out the data before aggregation take place**.**	Having is used to filter out the data after aggregation take place.
Where cannot be used with Aggregate functions.	Having is used with Aggregate functions.
Where is used to impose condition on SELECT statement.	Having is used to impose condition on GROUP BY clause.
E.g. SELECT * FROM employee WHERE state = "INDIA" ;	E.g. SELECT *, AVG(salary) FROM employee GROUP BY department HAVING salary>=20000;
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).

.Question 18: Explain SQL LIKE Clause.

Answer: The SQL LIKE clause is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the LIKE operator.

The percent sign (%)
The underscore (_)
The percent sign represents zero, one or multiple characters. The underscore represents a single number or character.
These symbols can be used in combinations.

Question 19: What is constraint and what are the various types of constraints ?

Answer: Constraints is the way of enforcing the rules in the database to maintain integrity of the database. Integrity constraint prevents the invalid entry in the table .

Below are the some constraints used in SQL server –

- **PRIMARY KEY -**
  This constraint defines a column or combination of columns which uniquely identifies each row in the table.

  We can define constraint either while creating table or after creating table using ALTER command. The table can have only one primary key.

  E.g. :
  CREATE TABLE student (id INT `CONSTRAINT stud_id_pk PRIMARY KEY,`
  name VARCHAR(20));
  OR
  ALTER TABLE student ADD CONSTRAINT PK_student PRIMARY KEY (ID);

  NOTE ::
  We can't access the rows uniquely without having primary key constraint as without PRIMARY KEY we can have duplicate values in the table.

- **FOREIGN KEY –**

  This constraint identifies any column referencing the PRIMARY KEY in another table.

  It establishes a relationship between two columns in the same table or between different tables.

  E.g. :

  ```
  CREATE TABLE order_items
  ( order_id number(5) CONSTRAINT od_id_pk PRIMARY KEY,
  product_id number(5) CONSTRAINT pd_id_fk REFERENCES, product(product_id)
  );
  ```

- **NOT NULL –**

  This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

  E.g. :

  ```
  CREATE TABLE employee
  ( id number(5), name char(20) CONSTRAINT nm_nn NOT NULL);
  ```

- **CHECK –**

  The CHECK constraint is used to limit the value range that can be placed in a column.

  If you define a CHECK constraint on a single column it allows only certain values for this column.

  E.g. :

  ```
  CREATE TABLE Persons ( ID INT NOT NULL, FirstName varchar(255), Age INT , City varchar(255), CONSTRAINT CHK_Person  CHECK (Age>=18 AND City='Sandnes')
  );
  ```

- **UNIQUE –**

  The UNIQUE constraint ensures that all values in a column are different.

  Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

Question 20 : How we can disable a constraints ?

Answer: ALTER TABLE test DISABLE CONSTRAINT constraint_name;

Question 21: What are the cascade constraint ?

Answer: DROP TABLE test CASCADE CONSTRAINT ;

Cascade constraint column is used to drop all the referential integrity constraints that refers to a primary key or unique key .

We have to perform these step before dropping table with integrity constraints.

Question 22: What is ON DELETE CASCADE ?

Answer: A foreign key with cascade delete , means that if we delete record in parent table then it will automatically deleted from child table.

E.g :

```
CREATE TABLE order_items
( order_id number(5) CONSTRAINT od_id_pk PRIMARY KEY,
product_id number(5) CONSTRAINT pd_id_fk REFERENCES, product(product_id)
ON DELETE CASCADE );
```

Question 23: If we have two tables and they have foreign key referencing each other then how we can do insertion?

Answer: To do insertion we first need to disable the foreign key constraint. Because we can't perform insertion in tables with circular references.

Question 24: What is the difference between *UNIQUE* and *PRIMARY KEY* constraints?

Answer: here should be only one *PRIMARY KEY* in a table whereas there can be any number of *UNIQUE* Keys.
PRIMARY KEY doesn't allow *NULL* values whereas Unique key allows NULL values.

Question 25: What is a join?

Answer: This is a keyword used to query data from more tables based on the relationship between the fields of the tables. Keys play a major role when JOINs are used.

Question 26: What are the different types of joins ?

Answer: Following are the types of joins in SQL –

Consider the following tables for example :-

emp :

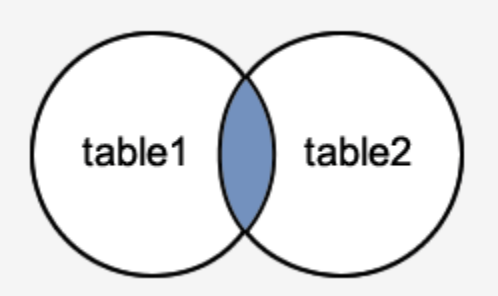
Id	name	manager
1	namrata	ram

Id	Salary
1	50000
6	55000
7	60000
4	45000
5	25000

2	nikhil	sita
3	sita	keshav
4	ram	hari
5	nita	anita

empDetails :

1. **INNER JOIN :**
   INNER JOINS return all rows from multiple tables where the join condition is met.

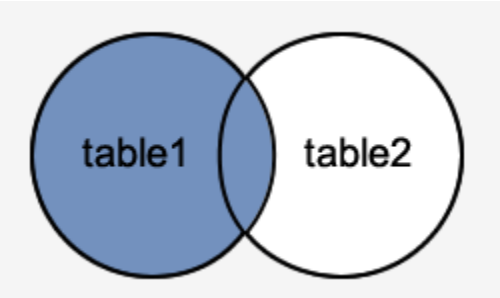


E.g. –
SELECT a. id , a. name , b.salary FROM employee a INNER JOIN empDetails b ON a. id = b. id ;
OUTPUT:

Id	name	salary
1	namrata	50000

4	ram	45000
5	nita	25000

2. **LEFT (OUTER) JOIN:**
   This type of join returns all rows from the LEFT - hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal.



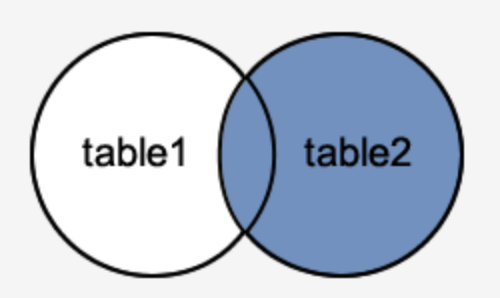
E.g. :

SELECT a. id , a. name , b.salary FROM employee a LEFT OUTER JOIN empDetails b ON a. id = b. id ;

OUTPUT :

Id	name	salary
1	namrata	50000
4	ram	45000
5	nita	25000
2	nikhil	(null)
3	sita	(null)

3. **RIGHT (OUTER) JOIN :**
   This type of join returns all rows from the RIGHT - hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal .



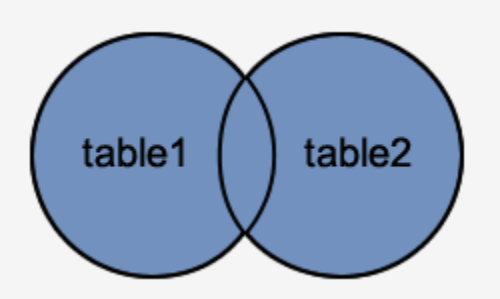
E.g. :

SELECT a. id , a. name , b.salary FROM employee a RIGHT OUTER JOIN empDetails b ON a. id = b. id;

OUTPUT :

Id	name	salary
1	namrata	50000
4	ram	45000
5	nita	25000
Null	null	55000
Null	null	60000

4. **FULL OUTER JOIN :**
   This type of join returns all rows from the **LEFT - hand** table and **RIGHT - hand** table with **NULL** in place where the join condition is not met.



E.g. :

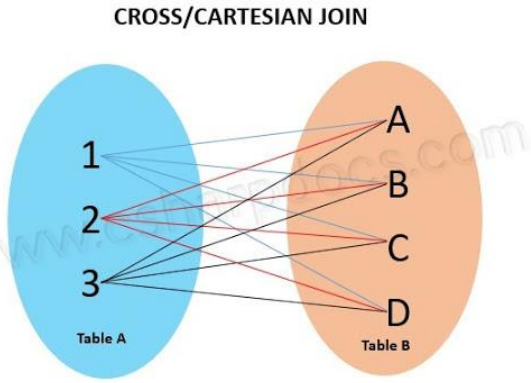
SELECT a. id , a. name , b.salary FROM employee a FULL OUTER JOIN empDetails b ON a. id = b. id;

OUTPUT:

Id	Name	salary
1	Namrata	50000
4	Ram	45000
5	Nita	25000
2	Nikhil	(null)
3	Sita	(null)

5. **CROSS JOIN :**

   A **cross join** is used when you wish to create a combination of every row from two tables. All row combinations are included in the result; this is commonly called **cross** product **join.**



   E.g. :

   SELECT * FROM emp , empDetails ;

6. **SELF JOIN :**

   A **self join** is a **join** in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To **join** a table itself means that each row of the table is combined with itself and with every other row of the table.

E.g. :
SELECT a.id , a.name as Emp_name , b.manager as Manager_Name FROM emp a INNER JOIN emp b ON
a.id = b.id ;

OUTPUT :

Id	Emp_Name	Manager_Name
1	Namrata	ram
2	Nikhil	sita
3	Sita	keshav
4	Ram	hari
5	Nita	anita

Question 27: What is the difference between inner and outer join? Explain with example.

Answer:

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).Inner join returns rows when there is at least one match in both tables.
If none of the record matches between two tables, then INNER JOIN will return a NULL set.
Below is an example of INNER JOIN and the resulting set.

Outer Join, on the other hand, will return matching rows from both tables as well as any unmatched rows from one or both the tables (based on whether it is single outer or full outer join respectively).Outer Join can be full outer or single outer.

Question 28: Full Outer Join is a combination of which of the following:-
a. Left Outer and Left Inner Join
b. Left Outer and Right Inner Join
c. Left Outer and Right Outer Join
d. Left Outer and Right Outer Join

Answer: Full Outer Join is a combination of Left Outer and Right Outer Join in SQL

Question 29: Right Outer Join is similar to:-
a. Right Inner Join
b. Left Inner Join

c. Left Outer Join
d. Right Outer Join

Answer: Right Outer Join is similar to Left Outer Join in SQL

Question 30: Explain SQL SET OPERATION.

Answer:

- **UNION :**

  The SQL Server UNION operator is used to combine the result sets of 2 or more SELECT statements. It removes duplicate rows between the various SELECT statements.

  Syntax :

  ```
  SELECT expression1, expression2, ... expression_n

  FROM tables

  [WHERE conditions]

  UNION

  SELECT expression1, expression2, ... expression_n

  FROM tables

  [WHERE conditions];
  ```

  E.g. :

  ```
  SELECT product_id

  FROM products

  UNION

  SELECT product_id

  FROM inventory;
  ```

- **UNION ALL :**

The SQL Server UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It returns all rows from the query and it does not remove duplicate rows between the various SELECT statements.

Each SELECT statement within the SQL Server UNION ALL operator must have the same number of fields in the result sets with similar data types.

Syntax :

```
SELECT expression1, expression2, ... expression_n

FROM tables

[WHERE conditions]

UNION ALL

SELECT expression1, expression2, ... expression_n

FROM tables

[WHERE conditions];
```

E.g. :

```
SELECT product_id

FROM products

UNION ALL

SELECT product_id

FROM inventory;
```

- **INTERSECT :**

  Common rows from both the tables are retrieved .

  Syntax :

```
SELECT expression1, expression2, ... expression_n

FROM tables
```

```
[WHERE conditions]

INTERSECT

SELECT expression1, expression2, ... expression_n

FROM tables

[WHERE conditions];
```

E.g. :

```
SELECT product_id

FROM products

WHERE product_id >= 50

INTERSECT

SELECT product_id

FROM inventory

WHERE quantity > 0;
```

Question 31: What is the difference between JOIN and UNION?

Answer: UNION combines the results of two or more queries into a single result set that includes all the rows that belong to all queries in the union. By using **JOINs**, you can retrieve data from two or more tables based on logical relationships between the tables

JOIN:-
a.) Joins the columns
b.)Duplicates are allowed
c.) Combines the column based on condition

UNION:-
a.) Merge the row
b.) Duplicates are not allowed
c.) Combine the result of two select statements.

Question 32: What is a view in SQL and what are its advantages?

Answer: View is virtual table based on the result set of SQL statement.When you execute view the data is pulled from the database and shown to the user.

Syntax :

```
CREATE VIEW view_name AS
 SELECT columns
 FROM tables  [WHERE conditions];
```

E.g. :

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
```

Question 33: What is force option in view ?

Answer: We can create view with errors by using force option , it means even if

the SQL statement is not valid , the view can be created.

E.g. :

```
CREATE force VIEW customer AS SELECT * FROM table_wait ;
```

In above SQL statement the table_wait is not even created still the view customer will get created.

Question 34: What happens to the view when table is dropped?

Answer: All views that are referencing to the dropped table will become invalid.

Question 35: What are the advantages of Views?

Answer: Some of the advantages of Views are

1. Views occupy no space
2. Views are used to simply retrieve the results of complicated queries that need to be executed often.
3. Views are used to restrict access to the database or to hide data complexity.

Question 36: What is a stored procedure?

Answer: Stored Procedure is a function consists of many SQL statement to access the database system. Several SQL statements are consolidated into a stored procedure and execute them whenever and wherever required.

Question 37: What is nested sub query ?

Answer: Query that appears in where clause is called as nested sub query .

E.g. :

SELECT * FROM sales WHERE id IN (SELECT id FROM customer WHERE region = “Maharashtra”);

Question 38: What is inline view ?

Answer: Query that appears in from clause is called as inline view.

E.g. :

SELECT * FROM sales s , (SELECT cust_id FROM customer WHERE region = “Maharashtra”) c WHERE s.cust_id = c.cust_id;

Question 39: What is scalar sub query ?

Answer: The sub query that appears in select statement , is called as scalar sub query.

It can return only one row and one column.

E.g. :

Select s.cust_id , s.total_amt , (select fisrtname from customer c where s.cust_id = c.cust_id) as firstname from sales s;

Question 40: What is the advantage of with clause ?

Answer: By using with clause we can execute sub query once and can reference it many times in main query.

The data is saved in temporary table , so it improves performance.

E.g. :

With dept_count AS (select deptno ,count(*) AS dept_count from emp GROUP BY deptno)

Select e.emp_name as Name , dc.dept_count as dept_count

From emp e , dept_count dc

Where e.depno = dc.depno ;

Question 41: Write co-related query and explain how it works?

Answer: Co-related query is one which the inner query requires input from outer query and the inner query execute once for each outer query.

E.g. :

Select empid , afname , salary , deptno from emp e where salary = (select avg(salary) from emp where deptno= e.depno) ;

Question 42: What is the difference between the IN and = operator?

Answer:

- **=** operator accepts only one value
- **IN** operator accepts more than 1 value.
  Example :
  - SELECT id , name FROM student WHERE id = 1;
  - SELECT id , name FROM student WHERE id IN 1,20,35;

Question 43: What is the difference between EXISTS and IN operator in Sub – queries?

Answer: The difference between IN and EXISTS depends upon how the sub query get executed .

For IN :-

The inner query is executed first and the list of values obtained as result is used by the outer query.

The inner query is executed only once.

Performance is comparatively SLOW for larger resultset of subquery

Example :

```
SELECT id , dept_id , salary FROM employee WHERE dept_id IN(SELECT
dept_id FROM department);
```

For EXIST :-

First row from the outer query is selected, then the inner query is executed , and the outer query uses this result for checking.

The inner query execution repeats as many number of times there are outer query rows.

Performance is comparatively FAST for larger resultset of subquery

Example :

```
SELECT * FROM emp WHERE EXIST ( SELECT * FROM contacts WHERE
emp.last_name = contacts.last_name AND emp.first_name =
contacts.first_name );
```

Question 44: Explain Various SQL FUNCTIONS.

Answer: SQL STRING FUNCTIONS :

- **SUBSTR() :**
  - The SUBSTRING() function extracts some characters from a string.

    Syntax :

SUBSTR(string , start , length);

E.g. :
Consider following table : emp

name
namrata
nikhil
sita

SELECT SUBSTR(name , 1, 2) AS NameSubstrings FROM emp ;

NameSubstrings
na
ni
Si

- **INSTR() :**

  - The INSTR() function returns the position of the first occurrence of a string in another string.

    This function performs a case-insensitive search.

    Syntax :

    `INSTR(string1, string2)`

    E.g. :
    SELECT name , INSTR(name , “n”) AS [First Presence] FROM emp;

name	First Presence
namrata	1
nikhil	1

- **CONCAT() :**

  - The CONCAT() function adds two or more expressions together.

    Syntax :

    CONCAT(*expression1*, *expression2*, *expression3*,...)

    E.g. :
    SELECT name ,CONCAT('city', " " ,'pincode') AS ADDRESS FROM emp;

Name	**City**	**Pincode**
Namrata	Solapur	413001
Shital	Pune	421003

    OUTPUT :

Name	**ADDRESS**
Namrata	Solapur 413001
Shital	Pune 421003

SQL NULL FUNCTIONS :

- **IFNULL() :**

  - The IFNULL() function returns a specified value if the expression is NULL.

    If the expression is NOT NULL, this function returns the expression.

    Syntax :

    IFNULL(*expression*, *alt_value*)

    E.g. :
    SELECT IFNULL(NULL , 'null value');

OUTPUT :

IFNULL(NULL , 'null value')
null value

- **ISNULL() :**

  - The ISNULL() function returns 1 or 0 depending on whether an expression is NULL.
    If expression is NULL, this function returns 1. Otherwise,
    it returns 0.

    Syntax :

    ISNULL(*expression*)

    E.g. :

    SELECT ISNULL(350) ;

    OUTPUT :

ISNULL(350)
0

Question 45: What is COALESCE function ?

Answer: COALESCE function accepts two or more parameters –

1. Returns the first non null value in the list.
2. If all values are null then , it returns null.

Syntax :

COALESCE(*val1*, *val2*, ..., *val_n*)

E.g. :

SELECT COALESCE (NULL , NULL ,'A','B');

OUTPUT :

A

- **NOTE :**
  we can not use **=** and **IN operators** with null.
  Below select statement returns 0 rows –

  Select * from customer where phone_no **=** null;
  Or
  Select * from customer where phone_no **IN** (null);

- **SOLUTION :**
  Instead of these we can use IS operator. As shown below –
  Example :

  Select * from student where name IS (NULL);

- **NOTE :**

1. Select 'tom' || null || 'joe' from table ;

   Will results in tomjoe , it ignores null.

2. All data types can have null values until you don't define NOT NULL constraint on that column.

Question 46: Explain Scalar functions.

Answer: Scalar functions are those functions which are used to return a single value based on the input. Following are the commonly used scalar functions in SQL:-

1. **UCASE()** – Convert to Upper Case
2. **LCASE(**) – Convert to Lower Case
3. **MID()** – It extracts a substring from a string.

   SELECT MID("ABCDEFGH",3,6) AS ExtractedString

   It will extract 6 characters starting from 3rd character i.e. C,D,E,F,G,H
4. **FORMAT()** – It specifies the display format
5. **LEN()** – Gives the length of a text field
6. **ROUND()** – Rounds up the decimal field in a number

Question 47: What is an Index?

Answer: An index is used to speed up the performance of queries. It makes faster retrieval of data from the table. The index can be created on one column or a group of columns.

Question 48: What are all the different types of indexes?

Answer: There are three types of indexes,

1. Unique Index:

Unique Indexes helps maintain data integrity by ensuring that no two rows of data in a table have identical key values. A unique index can be applied automatically when a primary key is defined. It ensures that the values in the index key columns are unique.

2. Clustered Index:

Clustered Index reorders the physical order of the table and search based on the key values. There will be only one clustered index per table. Each table can have only one clustered index.

3. Non-Clustered Index:

Non-Clustered Index doesn't alter the physical order of the table and maintains a logical order of the data. Each table can have max 999 non-clustered indexes.

.

Question 49: What is a relationship and its types?

Answer: Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:-
One to One Relationship.
One to Many Relationship.
Many to One Relationship.
Self-Referencing Relationship.

Question 50: What is normalization?

Answer: Normalization is the process of minimizing redundancy and dependency by organizing

fields and table of a database. The main aim of Normalization is to add, delete or modify field that can be made in a single table.

Question 51: What is Denormalization?

Answer: DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables**.**

Question 52: How many Normalization forms are there?

Answer: There are mainly 4 forms of Normalization,

First Normal Form (1NF): It removes all duplicate columns from the table. Creates a table for related data and identifies unique column values

Second Normal Form (2NF): Follows 1NF and creates and places data subsets in an individual table and defines the relationship between tables using a primary key

Third Normal Form (3NF): Follows 2NF and removes those columns which are not related through primary key

Fourth Normal Form (4NF): Follows 3NF and do not define multi-valued dependencies. 4NF also known as BCNF

Question 53: What is the difference between Local Variables and Global Variables?

Answer:

Local Variables: Local variables can be used or exist only inside the function. These variables are not used or referred by any other functions. These are not known to other functions. Variables can be created whenever that function is called.

Global Variables: Global variables can be used or exist throughout the program. Same variable

declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

Question 54: What is a NULL value?

Answer: A NULL value is neither zero nor a blank space. It represents a value that it unavailable, unknown, Not applicable, etc.
Remember zero is a number and blank space is a character

Question 55: What is the difference between NULL value, Zero, and Blank space?

OR

Is a NULL value same as zero or a blank space? If not then what is the difference?

Answer: As 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).

Question 56: What is the order of SQL query execution?

Answer:

Order	Clause	Function
1	FROM and JOIN	Select the table to extract from
2	WHERE	Filter the base data
3	GROUP BY	Aggregate the base data
4	HAVING	Filter the aggregate data
5	SELECT	Return the final data
6	DISTINCT	Apply distinct or unique condition
7	ORDER BY	Sort the final data
8	LIMIT	Apply limit on the number of rows

Question 57: What is sequences & what is need of it ?

Answer: A Sequence is database object that auto generates the continuous unique number.

Syntax :

```
CREATE SEQUENCE [schema_name.] sequence_name
   [ AS integer_type ]
   [ START WITH start_value ]
   [ INCREMENT BY increment_value ]
   [ { MINVALUE [ min_value ] } | { NO MINVALUE } ]
   [ { MAXVALUE [ max_value ] } | { NO MAXVALUE } ]
   [ CYCLE | { NO CYCLE } ]
   [ { CACHE [ cache_size ] } | { NO CACHE } ];
```

E.g. :

```
CREATE SEQUENCE item_counter
        AS INT
       START WITH 10 INCREMENT BY 10;

SELECT NEXT VALUE FOR item_counter;
OUTPUT :
Current_value
      10
SELECT NEXT VALUE FOR item_counter;
OUTPUT :
 Current_value
         20
```

Question 58: What are the different windows function in SQL SERVER ?

Answer:

Window functions operate on a set of rows and return a single aggregated value for each row. The term Window describes the set of rows in the database on which the function will operate.

The main advantage of using Window functions over regular aggregate functions is: Window functions do not cause rows to become grouped into a single output row, the rows retain their separate identities and an aggregated value will be added to each row.

Syntax :

window_function ([ALL] expression)
OVER ([PARTITION BY partition_list] [ORDER BY order_list])

Arguments :

- **window_function**
  Specify the name of the window function
- **ALL**
  ALL is an optional keyword. When you will include ALL it will count all values including duplicate ones. DISTINCT is not supported in window functions
- **expression**
  The target column or expression that the functions operates on. In other words, the name of the column for which we need an aggregated value. For example, a column containing order amount so that we can see total orders received.
- **OVER**
  Specifies the window clauses for aggregate functions.
- **PARTITION BY partition_list**
  Defines the window (set of rows on which window function operates) for window functions. We need to provide a field or list of fields for the partition after PARTITION BY clause. Multiple fields need be separated by a comma as usual. If PARTITION BY is not specified, grouping will be done on entire table and values will be aggregated accordingly.
- **ORDER BY order_list**
  Sorts the rows within each partition. If ORDER BY is not specified, ORDER BY uses the entire table.

Question 59:Explain types of windows function .

Answer: This are the types of windows function –

1. Aggregate windows function
2. Ranking windows function

1. Aggregate windows function :

Consider the following table for example : employee

```
SELECT * FROM employee;
```

Id	Name	Salary	Dept
1	amol	20000	IT
2	nisha	30000	CSE
3	namita	40000	EC
4	rita	14000	IT
5	ritesh	56000	CSE
6	monika	45000	IT
7	siya	90000	IT
8	kishore	67000	CSE
9	nikhil	95000	EC
10	ganesh	85000	EC

- **SUM() :**
  In SQL Server (Transact-SQL), the SUM function returns the summed value of an expression.

  Syntax :

  SELECT SUM (aggregate_expression)
  FROM tables
  [WHERE conditions];

  E.g. :

```
SELECT dept , SUM(salary) FROM employee GROUP BY dept HAVING sum(salary) > 20000;
```

Dept	
CSE	153000
EC	220000
IT	169000

- **MIN() :**
  In SQL Server (Transact-SQL), the MIN function returns the minimum value of an expression.
  Syntax :

  SELECT MIN(aggregate_expression)
  FROM tables
  [WHERE conditions];

  E.g. :

```
1 SELECT dept , MIN(salary)AS min_salary FROM employee GROUP BY dept HAVING sum(salary) > 20000;
2
```

Dept	Min_salary
CSE	30000
EC	40000
IT	14000

- **MAX() :**
  In SQL Server (Transact-SQL), the MAX function returns the maximum value of an expression.

  Syntax :

  SELECT MAX(aggregate_expression)
  FROM tables
  [WHERE conditions];

  E.g. :

```
SELECT dept , MAX(salary)AS max_salary FROM employee GROUP BY dept HAVING sum(salary) > 20000;
```

Dept	Max_salary
CSE	67000
EC	95000
IT	90000

- **AVG() :**
  In SQL Server (Transact-SQL), the AVG function returns the average value of an expression.
  Syntax :

  SELECT AVG (aggregate_expression)
  FROM tables
  [WHERE conditions];

  E.g. :

MS SQL | MS SQL | MS SQL

```
SELECT dept , avg(salary)AS [average salary ] FROM employee GROUP BY dept ;
```

Dept	Average salary
CSE	51000
EC	73333
IT	42250

- **COUNT() :**
  In SQL Server (Transact-SQL), the COUNT function returns the count of an expression.
  Syntax :
  SELECT COUNT (aggregate_expression)
  FROM tables
  [WHERE conditions];
  E.g. :

```
SELECT dept , COUNT(id)AS [Total employee ] FROM employee GROUP BY dept ;
```

Dept	Total employee
CSE	3
EC	3
IT	4

2. RANKING WINDOWS FUNCTIONS :

- **RANK () :**

  The RANK() function is used to give a unique rank to each record based on a specified value, for example salary, order amount etc.

  If two records have the same value then the RANK() function will assign the same rank to both records by skipping the next rank.

E.g. :

```
SELECT rank() over(ORDER BY customername) AS [RANK] ,* FROM customer;
```

Rank	Customername	Product	Amount	Vendor
1	Rajendra	books	203	Archies
1	Rajendra	pens	206	Archies
3	Raju	bag	200	Bata
4	shiv	shoes	100	Bata
5	sukesh	perfume	102	Archies

- **DENSE_RANK() :**

  Rank() function skips continuous number while ranking ,to overcome this we use dense_rank() .

  E.g. :

```
SELECT dense_rank() over(ORDER BY customername) AS [Dense RANK] ,* FROM customer;
```

Dense rank	Customername	Product	Amount	Vendor
1	Rajendra	books	203	Archies
1	Rajendra	pens	206	Archies
2	Raju	bag	200	Bata
3	shiv	shoes	100	Bata
4	sukesh	perfume	102	Archies

PARTITION BY :

The row number will be reset for each partition if PARTITION BY is specified.

E.g. :

```
SELECT row_number() over(partition BY vendor) AS [Partiton NO] , * FROM customer;
```

Partiton no	Customername	Product	Amount	Vendor
1	sukesh	perfume	102	Archies
2	Rajendra	books	203	Archies
3	Rajendra	pens	206	Archies
1	shiv	shoes	100	Bata
2	Raju	bag	200	Bata

Question 60: What is the difference between Rank() and Dense_Rank() function in SQL?

Answer: 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}.

Question 61: What is NTILE() function?

Answer: NTILE() function distributes the rows in an ordered partition into a specific number of groups. These groups are numbered. For example, NTILE(5) will divide a result set of 10 records into 5 groups with 2 record per group. If the number of records is not divided equally in the given group, the function will set more record to the starting groups and less to the following groups.

```
SELECT emp.*,
NTILE(3) over (order by salary DESC) as GeneratedRank
from Employee emp
```

This will divide the complete data set in 3 groups from top. So the GeneratedRank will be 1 for Amit and Bhargav, 2 for Chirag and Dinesh: 3 for Esha and Farhan

Question 62: What is data dictionary ?

Answer:

Data dictionary is set of read only tables that provide information(metadata) about the database .

The data dictionary contains the schema for all objects like (tables , views , indexes , cluster ,procedure , functions and so on).

The data dictionary also contains the privileges and roles granted to each user.

Question 63: What is save point ?

Answer: Save point is the marker in the transaction that allow for partial transaction.

Even if we encounter error we can rollback to a save point or all the way we can back to the beginning of transaction.

Question 64: Can we create a table based on the existing table?

OR

What is the easiest way to backup table?

Answer: Yes , Using **CTAS(Create Table AS)**

Example :

- CREATE TABLE Sales1 AS SELECT * FROM sales;

Question 65: What is a trigger?

Answer: Trigger allows you to execute a batch of SQL code when an insert, update or delete command is executed against a specific table. Actually triggers are special type of stored procedures that are defined to execute automatically in place or after data modifications.

Question 66: What is fetch and offset?

Answer:
Offset is like, “how many rows you want to chuck” and fetch is like “how many rows do you want to include”

So If my table has data of top 100 students.
OFFSET 10 FETCH 5 will get you the 11th to 15th rank student.