SQL Introduction.

1. SQL stands for Structured Query Language .

2. SQL is a standard language for accessing and manipulating databases.

SQL can execute queries against a database
SQL can retrieve data from a database
SQL can insert records in a database
SQL can update records in a database
SQL can delete records from a database
SQL can create new databases
SQL can create new tables in a database

Important Question for Interview

DBMS	RDBMS
1 . In DBMS Data are Stored as a File.	1. In RDBMS Data are stored in a Tabular Form.
2. Normalization is not Available in DBMS.	2. Normalization is Available in RDBMS.
3. It Allow one user at a time.	3. It Allows more than one user at a time.
4. Hierarchal Arrangement of data.	4. Stores Data in the form of Rows and Column.
5. It Does not Support ACID Property	5. It Support ACID Property.
6. It Does Not support Client Server Architecture.	6. It support Client Server Architecture.

1. Create Table

CREATE TABLE statement is used to Create a new table in a database. You should specify the name and data types of each column.

```
CREATE TABLE table_name(

Column1 datatype,
Column2 datatype,
Column3 datatype

);
```

```
CREATE TABLE Persons
(
P_ID int,
FirstName varchar(255),
LastName varchar(255),
Address varchar(255),
City varchar(255)
);
```

2. INSERT INTO

INSERT INTO Statement is used to insert new record in a table.

```
INSERT INTO   table_name( Column1 , Column2 , Column3 )
VALUES( value1, value2,value3)
```

```
INSERT INTO   table_name( P_ID , FirstName , LastName, Address, City)
VALUES( 1 , 'Lokesh' , 'Yadav' ,  'Street 16' , 'Walla');
```

3. AND & OR Operator

AND Operator displays a record if both the first and second condition is true.
OR Operator displays a record if either the first or the second condition is true.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

4. Where Clause

Where Clause is used to extract only those records that fulfill a specified criterion.

```
SELECT column_name from
table_name where column_name
operator value;
```

5. Order By

ORDER BY Keyword is used to sort the result set by a specified column.

1. **The ORDER BY Keyword sort the records in ascending order by default.**
2. **If you want to sort the records in a descending order, you can use DESC keyword.**

```
Select column_name from table_name order by column_name
ASC | DESC
```

6. Distinct Statement

In a table, some of the columns may contain duplicate values. This is Not a problem, sometimes you will want to list only the different values in a table.

```
SELECT DISTINCT column_name
from table_name;
```

7. Delete Statement

The DELETE Statement is used to delete rows in a table.
Note:- if you omit the where clause, all records will be deleted.
Be careful when deleting records. You cannot undo this statement.

```
DELETE FROM table_name  where
condition ;
```

8. DATE TIME

```
SELECT  NOW(), CURDATE(),
CURTIME();
```

9. Functions.

AVG() - SELECT AVG(column_name) from table_name;
COUNT() – SELECT COUNT(column_name) from table_name;
LCASE() – SELECT LCASE(column_name) from table_name;
LEN() – SELECT LEN(column_name) from table_name;
MAX() – SELECT MAX(column_name) from table_name;
MIN() – SELECT MIN(column_name) from table_name;
SUM() - SELECT SUM(*column_name*) FROM *table_name* WHERE *condition*;
ROUND() – SELECT column_name, ROUND(column_name, decimals) from table_name;
SUBSTRING() -> is used to get part of a String.
SELECT LastName , SUBSTR(FirstName,1,1) AS Initial from Persons ;
UCASE() – SELECT UCASE(column_name) from table_name;
REPLACE() – SELECT REPLACE(CustomerName , 'Brown' , 'Hello') from Orders;

10 . GROUP BY Statement.

GROUP BY Statement is used in conjunction with the aggregate functions to group the result set by one or more columns.

SELECT *column_name(s)*, FROM *table_name*
WHERE *condition* GROUP BY *column_name(s)*

SELECT Customer , SUM(OrderPrice) from Orders Group by Customers

11. HAVING Clause

THE HAVING clause was added to SQL because the Where keyword cannot be used with aggregate functions.

```
SELECT Customer,  Sum(OrderPrice) from
Orders
Group BY Customer Having
SUM(OrderPrice)<4000
```

12. ALTER TABLE STATEMENT

ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

1. **To Add a Column in a table -> (ALTER TABLE table_name ADD column_name datatypes)**
2. **To delete a column in a table -> (ALTER TABLE table_name drop column column_name)**
3. **To change the data types of a column in a table -> (ALTER TABLE table_name ALTER column column_name datatypes).**

13. ALIAS

SQL ALIAS - you can give a table or a column another name by using an alias.

Select Column_name as alias_name from table_name

14. DROP

SQL DROP DATABASE - > The DROP DATABASE Statement is used to drop an existing SQL Database.

SQL DROP TABLE - > The DROP TABLE Statement is used to drop an existing table in a Database.

```
DROP DATABASE Database_name;

DROP TABLE Table_name;
```

15. Between Operator

The Between Operator is used in a where Clause to select a range of data between two values. Begin and end value are included.

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

16. IN Operator.

IN Operator allows you to specify multiple values in a where clause. The number of values in the parenthesis can be one or more, with each values separated by comma.

```
SELECT * From Persons where LASTNAME IN ('JAMES' , 'MRIDUL');
```

17. SQL LIKE Operator

The SQL Like Operator is used in a Where CLAUSE to search for a specified Pattern in a column.

WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length.
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length.
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

18. TRUNCATE Command

The SQL TRUNCATE TABLE command is used to delete complete data from an existing table.

```
TRUNCATE TABLE  table_name;
```

19. UPDATE Command

The Update statement is used to update records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

20. Constraint

SQL Constraints are used to specify rules for the data in a table.

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY - Prevents actions that would destroy links between tables
CHECK - Ensures that the values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column if no value is specified

21. NOT NULL

By Default a table column can hold NULL Values.

THE NOT NUL Constraint enforces a column to NOT accept NULL values.

```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255) NOT NULL,
  Age int
);
```

22. SQL Unique Constraint

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

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

23. Primary Key

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key

```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  PRIMARY KEY (ID)
);
```

24. Check Constraint

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 column it will allow only certain values for this column.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

25. Foreign Key

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

```
CREATE TABLE Orders (
  OrderID int NOT NULL,
  OrderNumber int NOT NULL,
  PersonID int,
  PRIMARY KEY (OrderID),
  FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

26. JOIN OPERATION

A JOIN clause is used to combine rows from two or more tables based on a related column between them

INNER JOIN/ JOIN -> Return rows that have matching values in both tables.

LEFT JOIN -> Return all Rows from the left table, even if there are no matches in the right table.

RIGHT JOIN -> Return all Rows from the Right table , even if there are no matches in the left table.

FULL JOIN -> returns rows when there is a match in one of the tables.

27. JOIN / INNER JOIN

SELECT table1.column1, table2.column2... FROM table1 INNER JOIN table2 ON table1.common_field = table2.common_field;

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

OID	DATE	CUSTOMER_ID	AMOUNT
102	2009-10-08 00:00:00	3	3000
100	2009-10-08 00:00:00	3	1500
101	2009-11-20 00:00:00	2	1560
103	2008-05-20 00:00:00	4	2060

```
SELECT  ID, NAME, AMOUNT, DATE FROM CUSTOMERS INNER JOIN ORDERS
 ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

ID	NAME	AMOUNT	DATE
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

```
SELECT  ID, NAME, AMOUNT, DATE FROM CUSTOMERS LEFT JOIN ORDERS
  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

```
SELECT  ID, NAME, AMOUNT, DATE FROM CUSTOMERS RIGHT JOIN ORDERS
  ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

ID	NAME	AMOUNT	DATE
3	kaushik	3000	2009-10-08 00:00:00
3	kaushik	1500	2009-10-08 00:00:00
2	Khilan	1560	2009-11-20 00:00:00
4	Chaitali	2060	2008-05-20 00:00:00

28 . INCREMENT

Auto Increment allows a unique number to be generated when a new record is inserted into a table.

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

29. Top Clause.

Top Clause is used to specify the number of records to return.

```
Select * from persons limit 5 ;
```

SQL Command

DDL – Data Definition Language
DQL – Data Query Language
DML – Data Manipulation Language
DCL – Data Control Language

DDL – CREATE, DROP, ALTER, TRUNCATE.
DML - INSERT , DELETE , UPDATE
DQL - SELECT
TCL - COMMIT , SAVEPOINT, ROLLBACK,
DCL - GRANT, REVOKE

ACID Properties

A – ATOMICITY (The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all.)

C - Consistency (the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always.)

I – Isolation (It means if two operations are being performed on two different databases, they may not affect the value of one another.)

D – Durability (The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives.)