JVM Institute

Commands:SQL

1.create a table-
CREATE TABLE table_name (col_1 datatype,
col_2 datatype,
col_3 datatype);

2.Insert-
The INSERT INTO statement is used to insert new records in a table
syntax-
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
example-
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

3.Add a column to a table-
ALTER TABLE table_name
ADD column_name datatype;

4.Distinct-
The SELECT DISTINCT statement is used to return only distinct (different) values.
syntax-
SELECT DISTINCT column1, column2, ...
FROM table_name;
example-
SELECT DISTINCT Country FROM Customers;

5.Where-
The WHERE clause is used to filter records.
It is used to extract only those records that fulfill a specified condition.
syntax-
SELECT column1, column2, ...
FROM table_name
WHERE condition;
example
-SELECT * FROM Customers
WHERE Country='Mexico';

6.HAVING Clause
The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
syntax-
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);

7.Orderby
The ORDER BY keyword is used to sort the result-set in ascending or descending order.
syntax-SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
example
SELECT * FROM Products
ORDER BY Price;
SELECT * FROM Products
ORDER BY Price asc;

8.Group by
The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country"
example-
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;

9.And-
The AND operator is used to filter records based on more than one condition.
syntax-SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
example
SELECT * FROM Customers
WHERE Country = 'Mexico' AND CustomerName = 'Centro comercial Moctezuma' ;

8.OR
The WHERE clause can contain one or more OR operators.
The OR operator is used to filter records based on more than one condition.
syntax-
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
example
SELECT * FROM Customers
WHERE Country = 'usa' OR Country = 'Spain';

9.NOT
The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.
sytax-
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
example-
select * from customers
where not city = 'london';

10.NULL Value
A field with a NULL value is a field with no value.
syntax-
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
not null
syntax-
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;

11.update
The UPDATE statement is used to modify the existing records in a table
syntax-
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition.
select Customers
set City = 'Oslo';

12.DELETE
The DELETE statement is used to delete existing records in a table.
sytax-
DELETE FROM table_name WHERE condition
example-DELETE FROM Customers

13.Drop
The DROP TABLE statement is used to drop an existing table in a database.
syntax-
DROP TABLE table_name;

15.Truncate
The TRUNCATE TABLE command deletes the data inside a table, but not the table itself.
syntax-
TRUNCATE TABLE Categories;

16.Top
The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.
syntax-
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
example-
SELECT TOP 3 * FROM Customers;

17.limit
example-
SELECT * FROM Customers
LIMIT 3;

18.Aggregate function -

a)MIN() and MAX()
example-
SELECT MIN(Price)
FROM Products;
SELECT MAX(Price)
FROM Products;

b)COUNT()-
example-
SELECT COUNT(*)
FROM Products;

c)sum-
example-
SELECT SUM(Quantity)
FROM OrderDetails;

d)avg-
example-
SELECT AVG(Price)
FROM Products;

19.In
The IN operator allows you to specify multiple values in a WHERE clause.
syntax-
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
example-
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');

20.Between
The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
syntax-
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
example-
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;

21.union-
The UNION operator is used to combine the result-set of two or more SELECT statements.
Every SELECT statement within UNION must have the same number of columns
The columns must also have similar data types
The columns in every SELECT statement must also be in the same order.

-SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
and
-union all
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;

example-
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);

22.Stored Procedure-
A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
syntax-
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;

23.Sql joins:-

inner join :-
syntax-
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

left join-
syntax-
```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

right join-
syntax-
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

full outer join
syntax-
```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

self join-
syntax-
```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

24. to find the third-highest salary from the EmpPosition table
```
 SELECT TOP 1 salary
FROM(
SELECT TOP 3 salary
FROM employee_table
ORDER BY salary DESC) AS emp
ORDER BY salary ASC;
```

25.Write a query to retrieve duplicate records from a table.
```
SELECT EmpID, EmpFname, Department COUNT(*)
FROM EmployeeInfo GROUP BY EmpID, EmpFname, Department
HAVING COUNT(*) > 1;
```


26.Write a query to retrieve the last 3 records from the EmployeeInfo table
SELECT * FROM EmployeeInfo WHERE
EmpID <=3 UNION SELECT * FROM
(SELECT * FROM EmployeeInfo E ORDER BY E.EmpID DESC)
AS E1 WHERE E1.EmpID <=3;

27.Write a query to retrieve the list of employees working in the same department
Select DISTINCT E.EmpID, E.EmpFname, E.Department
FROM EmployeeInfo E, Employee E1
WHERE E.Department = E1.Department AND E.EmpID != E1.EmpID;

28.Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.
SELECT SUBSTRING(EmpLname, 1, 4) FROM EmployeeInfo;

29.select common records from two tables:-
Using the INTERSECT statement:
SELECT * FROM table_1
INTERSECT
SELECT * FROM table_1;

CASE()
this function sequentially checks the provided conditions in the WHEN clauses and returns the value from the corresponding THEN clause when the first condition is satisfied.
syntx
CASE
  WHEN condition_1 THEN value_1
  WHEN condition_2 THEN value_2
  WHEN condition_3 THEN value_3
  ...
  ELSE value
END;

30.How to select all even or all odd records in a table
SELECT * FROM table_name
WHERE MOD(ID_column, 2) = 0;

31.SELECT * FROM table_name
WHERE ID_column % 2 = 0;

33.How to find the nth highest value in a column of a table(using offset)
SELECT * FROM table_name
ORDER BY column_name DESC
LIMIT 1
OFFSET 5;

34.windowing funcation:_

1.rank
```
SELECT
   id,
   name,
   department,
   salary,
   RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS salary_rank
FROM
   employees;
```

2.dense rank-
```
SELECT
   id,
   name,
   department,
   salary,
   DENSE_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS
salary_dense_rank
FROM
   employees;
```

3.rownumber
```
SELECT
   id,
   name,
   department,
   salary,
   ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary DESC) AS row_num
FROM
   employees;
```

4.lead
```
SELECT
   id,
   name,
   department,
   salary,
   LEAD(salary, 1) OVER (PARTITION BY department ORDER BY salary DESC) AS next_salary
FROM
   employees;
```

5.lag
```
SELECT
   id,
   name,
   department,
   salary,
   LAG(salary, 1) OVER (PARTITION BY department ORDER BY salary DESC) AS prev_salary
FROM
   employees;
```

34.What is the maximum number of tables that can join in a single query?
256, check SQL Server Limits

35.Write a Query to display the date after 12 months?
```
SELECT DATEADD(mm, 2, getdate())
```

36.write a Query to display the date before 15 days?
```
SELECT DATEADD(dd, -15, getdate())
```

37.Write a Query to display the total salary of employees based on region?
```
SELECT region, SUM(salary) AS 'total_salary'
FROM employee
GROUP BY region;
```

38.Write a Query to display the number of employees working in each region?
```
SELECT region, COUNT(gender)
FROM employee
GROUP BY region;
```

39.Write a Query to display the total salary of employees based on dept_name?
```
SELECT dept_name, SUM(salary) AS 'total_sal'
FROM employee
GROUP BY dept_name
```

40.Write a Query to display the total salary of employees based on whose total salary > 12000?
```
SELECT city, SUM(salary) AS 'total_salary'
FROM employee
GROUP BY city
HAVING SUM(salary)>12000;
```

41.Write a Query to display employee details whose employee numbers are 101, 102?
```
SELECT *
FROM employee
WHERE Emp_No in (101, 102)
```

42.Write a Query to display the first record from the table?
```
SELECT TOP 1 *
FROM employee
```

43.Write a Query to display the top 3 records from the table?
SELECT TOP 3 *
FROM employee

44.Write a Query to display the last record from the table?
SELECT TOP 1 *
FROM employee
ORDER BY emp_no descending

45.index--
This creates a clustered index on the id column
CREATE CLUSTERED INDEX idx_primary_id ON employees(id);

-- Creating a non-clustered index on the department column
CREATE NONCLUSTERED INDEX idx_department ON employees(department);

-- Creating a non-clustered index on the salary column
CREATE NONCLUSTERED INDEX idx_salary ON employees(salary);

46.views:_
1.simple view-
-- Create a simple view
CREATE VIEW SimpleEmployeeView AS
SELECT EmployeeID, FirstName, LastName
FROM Employees;

-- Query the simple view
SELECT * FROM SimpleEmployeeView;

2.complex view-
SELECT * FROM ComplexEmployeeView;

3.materialized view
-- Create a materialized view
CREATE MATERIALIZED VIEW MaterializedEmployeeView AS
SELECT
  e.EmployeeID,
  e.FirstName,
  e.LastName,
  d.DepartmentName
FROM
  Employees e
JOIN
  Departments d ON e.Department = d.DepartmentName;

-- Refresh the materialized view to update its data
REFRESH MATERIALIZED VIEW MaterializedEmployeeView;

-- Query the materialized view
SELECT * FROM MaterializedEmployeeView;

47.HOW TO DELETE DUPlicate records:-

```
DELETE FROM Employees
WHERE EmployeeID NOT IN (
   SELECT MIN(EmployeeID)
   FROM Employees
   GROUP BY FirstName, LastName, Department
);
```


