Database Interview Questions

INDEX

	column as “FullName”. The first name and the last name must be separated with space.
Q37.	Write a query find number of employees whose DOB is between 02/05/1970 to 31/12/1975 and are grouped according to gender
Q38.	Write a query to fetch all the records from the EmployeeInfo table ordered by EmpLname in descending order and Department in the ascending order.
Q39.	Write a query to fetch details of employees whose EmpLname ends with an alphabet ‘A’ and contains five alphabets.
Q40.	Write a query to fetch details of all employees excluding the employees with first names, “Sanjay” and “Sonia” from the EmployeeInfo table.
Q41.	Write a query to fetch details of employees with the address as “DELHI(DEL)”
Q42.	Write a query to fetch all employees who also hold the managerial position.
Q43.	Write a query to fetch the department-wise count of employees sorted by department’s count in ascending order.
Q44.	Write a query to calculate the even and odd records from a table.
Q45.	Write a SQL query to retrieve employee details from EmployeeInfo table who have a date of joining in the EmployeePosition table.
Q46.	Write a query to retrieve two minimum and maximum salaries from the EmployeePosition table.
Q47.	Write a query to find the Nth highest salary from the table without using TOP/limit keyword.
Q48.	Write a query to retrieve duplicate records from a table.
Q49.	Write a query to retrieve the list of employees working in the same department.
Q50.	Write a query to retrieve the last 3 records from the EmployeeInfo table.

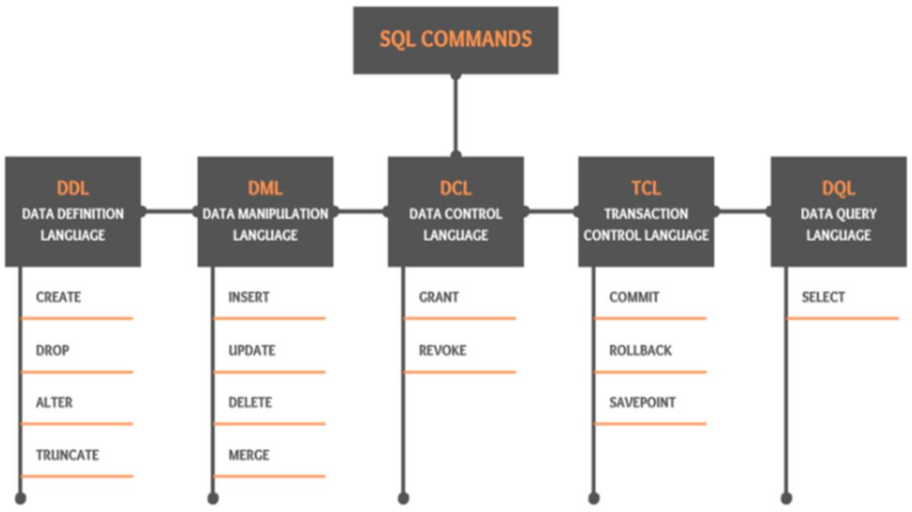
Q51.	Write a query to find the third-highest salary from the EmpPosition table
Q52.	Write a query to display the first and the last record from the EmployeeInfo table.
Q53.	Write a query to add email validation to your database
Q54.	Write a query to retrieve Departments who have less than 2 employees working in it.
Q55.	Write a query to retrieve EmpPostion along with total salaries paid for each of them.
Q56.	Write a query to fetch 50% records from the EmployeeInfo table.
Q57.	Select nth highest salary from the table.
Q58.	Write a query to select last record from the table Student.
Q59.	SELECT * FROM Student WHERE Rowid = SELECT MAX(Rowid) from Student;
Q60.	Write a query to access the first Nth records from the table student. Where N=4
Q61.	Write a query to find all even rows from the table Student.
Q62.	From the above given tables write a query to find all the students name in which students got less than 90 marks.
Q63.	Write a query to create a new table name Student_marks from the given table Student.
Q64.	Write a query to show the 3rd highest marks from the Student table.
Q65.	Write a query to view current date and time.
Q66.	Write a query to show the record of the three highest marks from the student table.
Q67.	Write a query to fetch the stu_Name and Stu_Marks of those students whose age is 20.
Q68.	Write a query to show the maximum marks of each subject.
Q69.	Write a query to show all the record of those students whose marks is greater than 82 and age is 22
Q70.	Write a query to show the record of those students whose name begins

	with the ‘m’ character.
Q71.	Write a query to show all Subject_Id along with the number of students in there.
Q72.	Write a query to fetch the values of the Stu_Name column from the Student table in the upper case.
Q73.	Write a query to show the unique values of Stu_age from the Student table.
Q74.	Write a query to view all student details from the Student table order by Stu_Name Descending.
Q75.	Write a query to show the three minimum marks from the student table.

Query Answers

Q1. What are DDL and DML languages?

- DDL stands for Data Definition Language. They include CREATE, DROP, ALTER and TRUNCATE statements.
- DDL statements are used to create, remove or modify database objects like table. You do not need to commit the changes after running DDL commands.
- CREATE statement can be used to create any database objects like tables, views, functions, procedures, triggers etc.
- DROP statement can be used to remove any database objects like tables, views, functions, procedures, triggers etc.
- ALTER statement can be used to modify the structure of a database objects.
- TRUNCATE statement can be used to remove all the data from a table at once.



DML:

DML stands for Data Manipulation Language. DML includes INSERT, UPDATE, DELETE and MERGE statements. DML statements are used to add, remove or modify data from database tables

-
- INSERT statement will add rows or records to a table.

- UPDATE statement will modify the data in the table.
- DELETE statement will remove one or multiple rows from a table.

- MERGE statement will either do an update or insert to a table based on the available data. If the data is present then it does an update. If data not present then merge will do an insert

DCL:

- DCL stands for Data Control Language. DCL includes GRANT and REVOKE statements.
- GRANT statements are used to provide access privileges to a database object to any database or schema.
- REVOKE statements are used to remove access privileges from a database object from any database or schema.

TCL:

- TCL stands for Transaction Control Language. TCL includes COMMIT, ROLLBACK and SAVEPOINT.
- COMMIT statement will permanently save any open transactions in the current session to the database. By transaction, I mean any changes done to any database table using any of the DML statements like INSERT, UPDATE, DELETE and MERGE.
- ROLLBACK statement will remove (unsave) any open transactions in the current session to the database. So all un committed transactions in the current session will be lost.
- SAVEPOINT statement can be used to create a specific pointer in your session and provide a name to this pointer. You can then either rollback or commit transactions only until this point (savepoint name) rather than committing or rollbacking all the transaction in the session.

DQL:

- **DQL** stands for Data Query Language. It includes only the SELECT statement.
- SELECT statement is used to fetch and view data from the database.

Q2.What is the difference between DELETE and TRUNCATE statement?

- Delete can be used to remove either few or all the records from a table. Whereas truncate will always remove all the records from the table. Truncate cannot have WHERE condition.

- Delete is a DML statement hence we will need to commit the transaction in order to save the changes to database. Whereas truncate is a DDL statement hence no commit is required.
- For example:
- Delete only the records from employee table where the name is ‘Thoufiq’
- DELETE FROM employee WHERE name = ‘Thoufiq’;
- COMMIT;
- delete all records from the employee table.
- DELETE FROM employee;
- COMMIT;
- delete all the records from the employee table. No commit is required here.
- TRUNCATE TABLE employee;

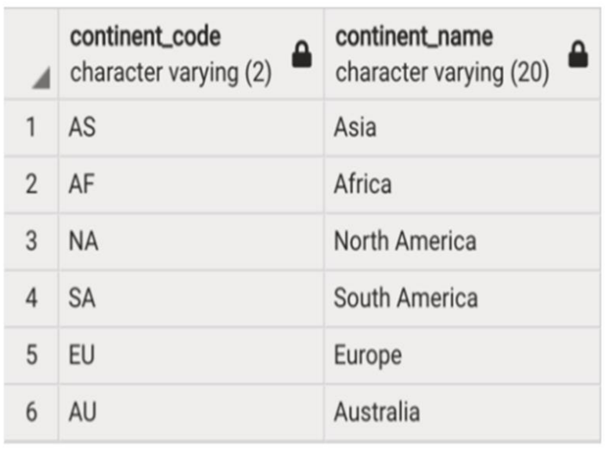
Q3.Why do we use CASE Statement in SQL? Give example

example:

We display the gender as ‘Male’ when the gender column in the employee table has value as ‘M’. And if the gender column has value as ‘F’ then we display the value as ‘Female’. If the gender column has values anything other than M or F then the query would return ‘Other’

- Select case when gender = ‘M’ then ‘Male
-                    when gender = ‘F’ then ‘Female’
-        else ‘other’
-        end as gender
- From employee

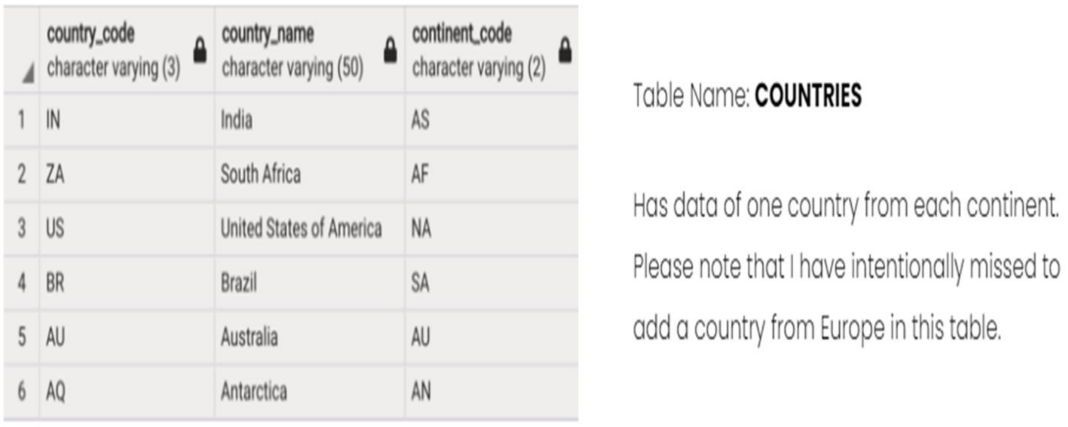
Q4.What is the difference between LEFT, RIGHT, FULL outer join and INNER join?



	continent_code character varying (2)	continent_name character varying (20)
1	AS	Asia
2	AF	Africa
3	NA	North America
4	SA	South America
5	EU	Europe
6	AU	Australia

Table Name: **CONTINENTS**

Has data of 6 continents. Please note the continent “Antarctica” is intentionally missed from this table.



	country_code character varying (3)	country_name character varying (50)	continent_code character varying (2)
1	IN	India	AS
2	ZA	South Africa	AF
3	US	United States of America	NA
4	BR	Brazil	SA
5	AU	Australia	AU
6	AQ	Antarctica	AN

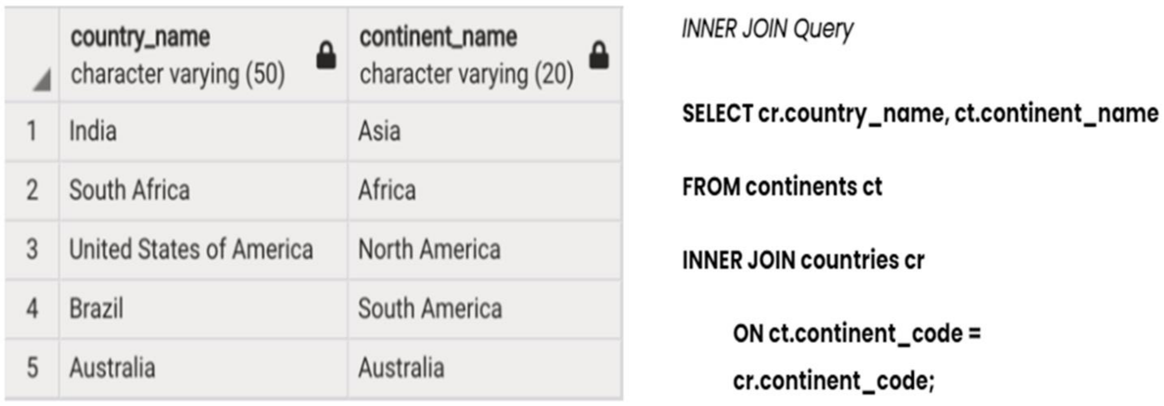
Table Name: **COUNTRIES**

Has data of one country from each continent.

Please note that I have intentionally missed to add a country from Europe in this table.

INNER JOIN:

- **INNER JOIN will fetch only those records which are present in both the joined tables.**



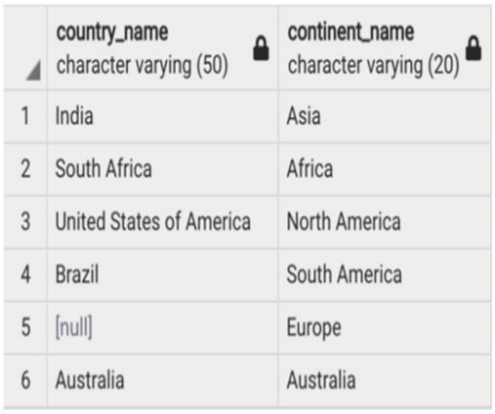
	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Africa
3	United States of America	North America
4	Brazil	South America
5	Australia	Australia

INNER JOIN Query

```
SELECT cr.country_name, ct.continent_name
FROM continents ct
INNER JOIN countries cr
    ON ct.continent_code =
    cr.continent_code;
```

LEFT JOIN

- **LEFT JOIN** will fetch all records from the left table (table placed on the left side during the join) even if those records are not present in right table (table placed on the right side during the join).



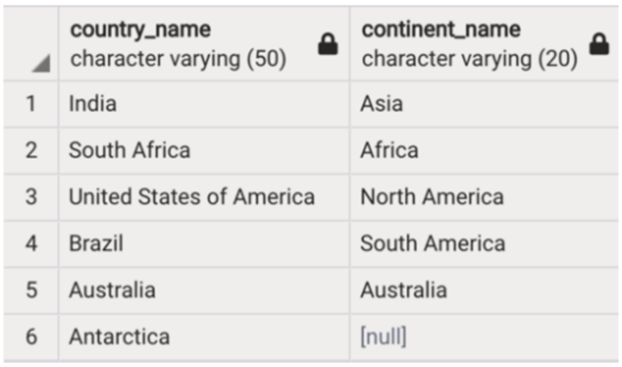
	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Africa
3	United States of America	North America
4	Brazil	South America
5	[null]	Europe
6	Australia	Australia

LEFT JOIN Query

```
SELECT cr.country_name, ct.continent_name
FROM continents ct
LEFT JOIN countries cr
    ON ct.continent_code =
    cr.continent_code;
```

RIGHT JOIN:

- **RIGHT JOIN** will fetch all records from the right table



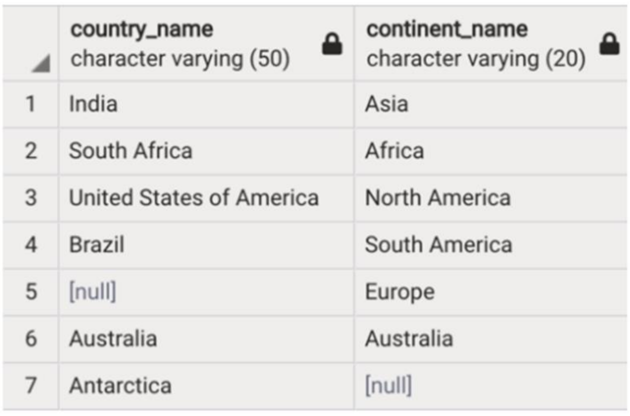
	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Africa
3	United States of America	North America
4	Brazil	South America
5	Australia	Australia
6	Antarctica	[null]

RIGHT JOIN Query

```
SELECT cr.country_name, ct.continent_name
FROM continents ct
RIGHT JOIN countries cr
    ON ct.continent_code =
    cr.continent_code;
```

FULL JOIN

- **FULL JOIN** will fetch all records from both left and right table.



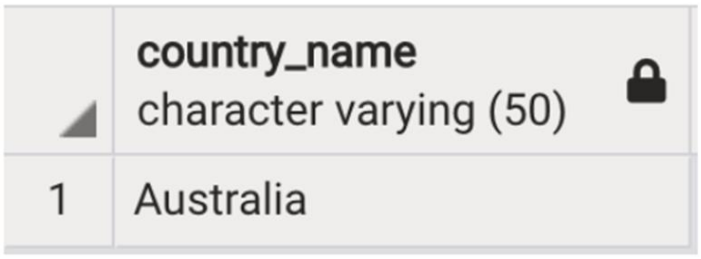
	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Africa
3	United States of America	North America
4	Brazil	South America
5	[null]	Europe
6	Australia	Australia
7	Antarctica	[null]

FULL OUTER JOIN Query

```
SELECT cr.country_name, ct.continent_name
FROM continents ct
FULL OUTER JOIN countries cr
    on ct.continent_code =
    cr.continent_code;
```

SELF JOIN

- **SELF JOIN** is when you join a table to itself.



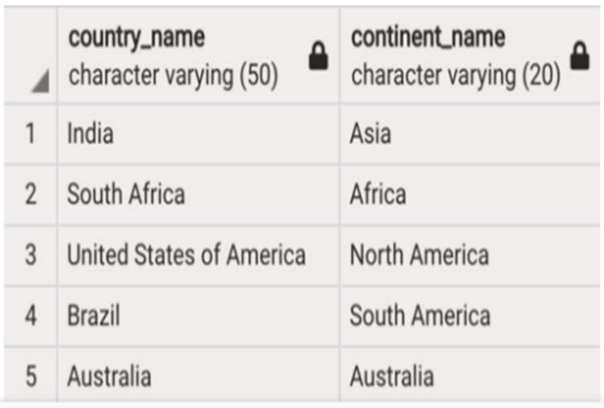
	country_name character varying (50)
1	Australia

SELF JOIN Query

```
SELECT cr1.country_name
FROM countries cr1
JOIN countries cr2
    ON cr1.country_code =
    cr2.continent_code;
```

NATURAL JOIN

- **NATURAL JOIN** is similar to INNER join but we do not need to use the ON clause during the join.



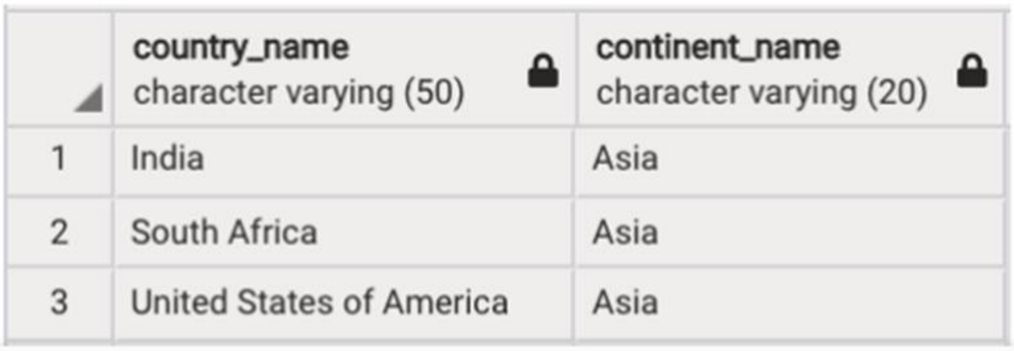
	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Africa
3	United States of America	North America
4	Brazil	South America
5	Australia	Australia

NATURAL JOIN Query

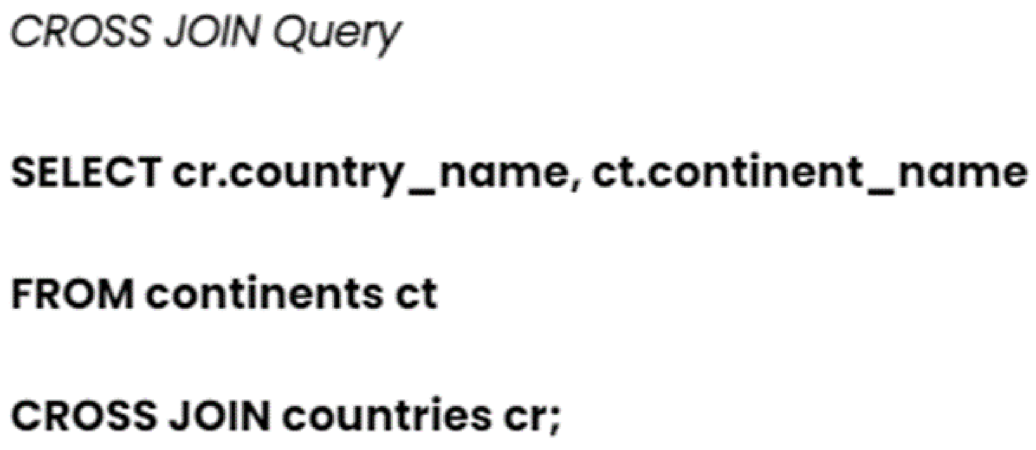
```
SELECT cr.country_name, ct.continent_name
FROM continents ct
NATURAL JOIN countries cr;
```

CROSS JOIN

- **CROSS JOIN** will join all the records from left table with all the records from right table.



	country_name character varying (50)	continent_name character varying (20)
1	India	Asia
2	South Africa	Asia
3	United States of America	Asia



CROSS JOIN Query

```
SELECT cr.country_name, ct.continent_name
FROM continents ct
CROSS JOIN countries cr;
```

Q5.What is the difference between DISTINCT and GROUP BY?

DISTINCT clause will return unique column values.

SELECT DISTINCT name FROM employee;

GROUP BY clause will group together the data based on the columns specified in group by.

SELECT name, COUNT(1) FROM employee GROUP BY name;

Q6• What are the rules to follow when using UNION operator?

•**UNION** operator can be used to combine two different SQL Queries. The output would be the result combined from both these queries. Duplicate records would not be returned.

•You can combine two queries using UNION operator if they follow the below rules:

•Both queries must return same same no of columns.

•The columns in both the queries must be in same order.

•Data type of all the columns in both the queries must be same.

Q7.What are aggregate functions? Name and explain different types of aggregate functions in SQL?

Aggregate function can be used to perform calculation on a set of values, which will then return a single value. We can use aggregate function either with GROUP BY clause or without it.

Find the sum of all the salaries from the entire employee

SELECT SUM(salary) as total_salary FROM employee;

Query will return the sum of salaries for each department in the employee table:

SELECT dept_id, SUM(salary) as total_salary_per_dept

Q8.AVG: Calculate the average from the given set of values.

- find the average salary from the entire employee table:
- SELECT AVG(salary) as average_salary FROM employee;

  query will return the average salary for each department in the employee table:

- SELECT dept_id, AVG(salary) as avg_salary_per_dept
- FROM employee
- GROUP BY dept_id;

<u>MIN</u>

- **MIN**: Find the minimum value in the given set of values.
- find the minimum salary from the entire employee table:
- SELECT MIN(salary) as min_salary FROM employee;
- Whereas the below query will return the minimum salar

  query will return the minimum salary for each department in the employee table:

- SELECT dept_id, MIN(salary) as min_salary_per_dept
- FROM employee
- GROUP BY dept_id;

<u>MAX:</u>

- **MAX**: Find the maximum value in the given set of values.

find the maximum salary from the entire employee table:

- SELECT MAX(salary) as max_salary FROM employee;
- query will return the maximum salary for each department in the employee table:
- SELECT dept_id, MAX(salary) as max_salary_per_dept
- FROM employee
- GROUP BY dept_id;

COUNT:

COUNT: Returns the number of records or rows.

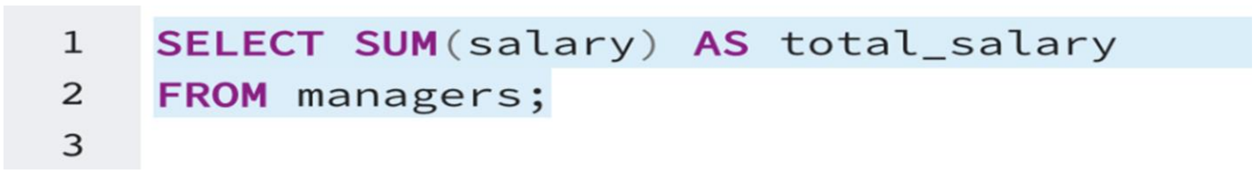
- find the total number of records (or employees) from the entire employee table:
- SELECT COUNT(emp_id) as no_of_emp FROM employee;
- query will return the maximum salary for each department in the employee table:
- SELECT dept_id, COUNT(emp_id) as no_of_emp_per_dept
- FROM employee
- GROUP BY dept_id;

Q9.Write the SQL query to get the third maximum salary of an employee from a table named employees.

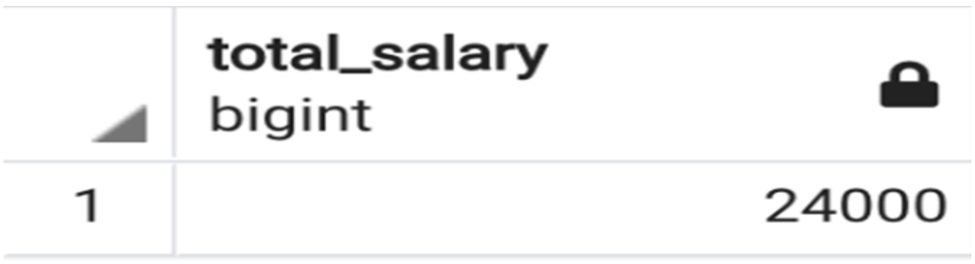
SELECT * FROM `employees` ORDER BY `salary` DESC LIMIT 1 OFFSET 2

Q10.Can we use aggregate function as window function? If yes then how do we do it?

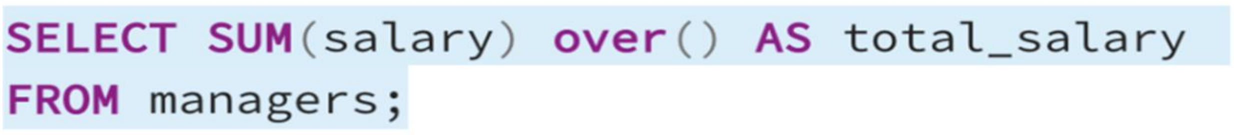
Yes, we can use aggregate function as a window function by using the OVER clause. Aggregate function will reduce the number of rows or records since they perform calculation of a set of row values to return a single value. Whereas window function does not reduce the number of records.



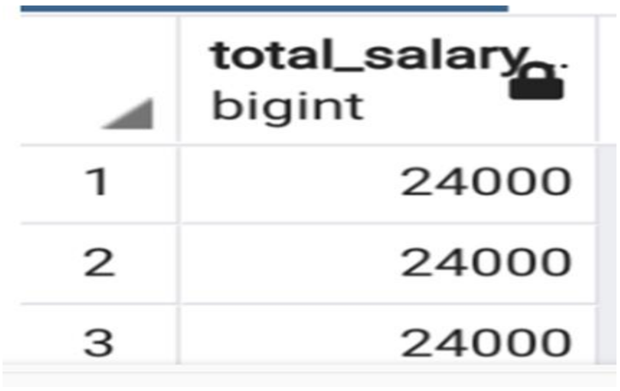
```
SELECT SUM(salary) AS total_salary
FROM managers;
```



	total_salary bigint
1	24000



```
SELECT SUM(salary) over() AS total_salary
FROM managers;
```



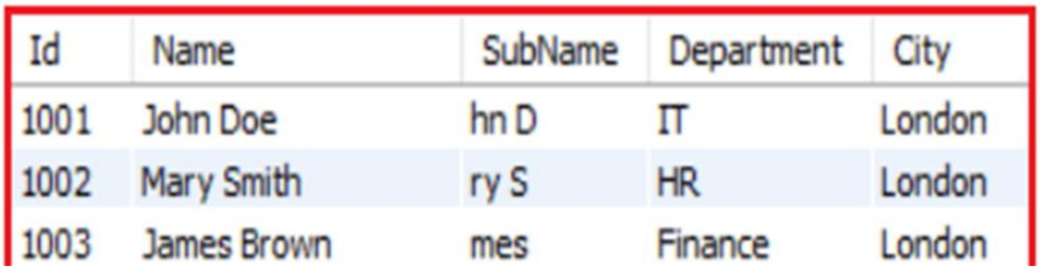
	total_salary bigint
1	24000
2	24000
3	24000

Q11.How can you convert a text into date format? Consider the How can you convert a text into date format? Consider the given text as “31-01-2021“.given text as “31-01-2021“.

SELECT STR_TO_DATE('31-01-2021','%d-%m-%Y') as date_value;

Q12.Extract a substring from the name column starting from the third position up to 4 characters.

•SELECT Id, Name, SUBSTRING(Name, 3, 4) As SubName, Department, City FROM Employee;



Id	Name	SubName	Department	City
1001	John Doe	hn D	IT	London
1002	Mary Smith	ry S	HR	London
1003	James Brown	mes	Finance	London

Q13.What are subqueries? Where can we use them?

A SELECT query statement which is placed inside another SELECT query is termed as a subquery. Subquery can also be termed as inner query.

•Example:

•SELECT * FROM continents ct

•WHERE ct.continent_code IN (SELECT cr.continent_code FROM countries cr);

Q14.Is it good to have the same subquery multiple times in your query? If no then how can you solve this?

It’s not a good practice to use the same subquery multiple times in your query. Repeating the same subquery multiple times in your query can impact the query performance (since the same query would execute multiple times) and also becomes difficult to maintain

Q15•Difference betwen WHERE and HAVING clause

WHERE clause is used to filter records from the table. We can also specify join conditions between two tables in the WHERE clause.

Whereas **HAVING** clause is used to filter records returned from the GROUP BY clause. So if a SQL query has WHERE, GROUP BY and HAVING clause then first the data gets filtered based on WHERE condition, only after this grouping of data takes place. Finally based on the conditions in HAVING clause the grouped data again gets filtered.

Q16•What are indexes? Why do we use it?

•**Index** is a database object which is applied on one or more columns of a table. When a column (or list of columns) from the table is Indexed, database creates a pointer to each value stored in that column. This significantly improves the query execution time since the database will have a more efficient way to find a particular value from the column based on its index.

Q17.What are steps you would take to tune a SQL query?

Check the SQL Query.

- Make sure all the table joins are correct and all the filter conditions are applied as intended.
- Also check for any Cartesian joins that may happen unintentionally.

Avoid any repeated sub queries by using a WITH clause.

- If using tables with huge list of columns then make sure to only fetch columns which are required for the current query.
- If required check the columns used in join conditions.

- **Check if index is created for the desired columns.**
- Make sure correct indexes are created on the desired columns. Following the correct type of indexes.
- Avoid creating unnecessary indexes.

Q18.What is the difference between primary key, unique key and foreign key

- Primary key, unique key and foreign key are constraints we can create on a table.
- **Primary Key** : When you make a column in the table as **primary key** then this column will always have unique or distinct values. Duplicate values and NULL value will not be allowed in a primary key column.
- **Foreign key** is used to create a master child kind of relationship between two tables. When we make a column in a table as foreign key, this column will then have to be referenced from another column from some other table.
- **Unique Key** : When you make a column in the table as **unique key** then this column will always have unique or distinct values. Duplicate values will not be allowed.

Q19.What is a View?

- View is a database object which is created based on a SQL Query. It’s like giving a name to the results returned from a SQL Query and storing it in the database as a view.

Q20.When can a function NOT be called from SELECT query?

If the function includes DML operations like INSERT, UPDATE, DELETE etc then it cannot be called from a SELECT query. Because SELECT statement cannot change the state of the database.

Q21.What is a trigger?

Trigger is a database object which is similar to a stored procedure which will automatically get invoked or executed when the specified event occurs in the database.

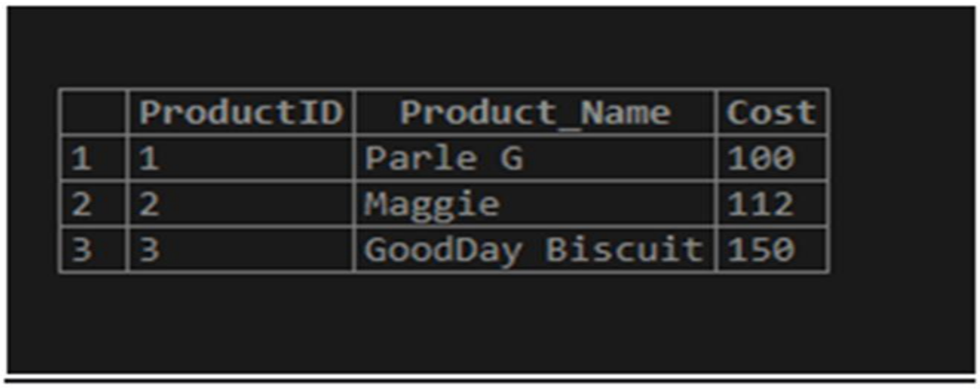
- DML triggers are invoked when a DML operation (INSERT, UPDATE, DELETE) occurs on the respective table (table on which the trigger was created).

- DDL triggers are invoked when a DDL operation (CREATE, ALTER, DROP) occurs on the respective table (table on which the trigger was created).

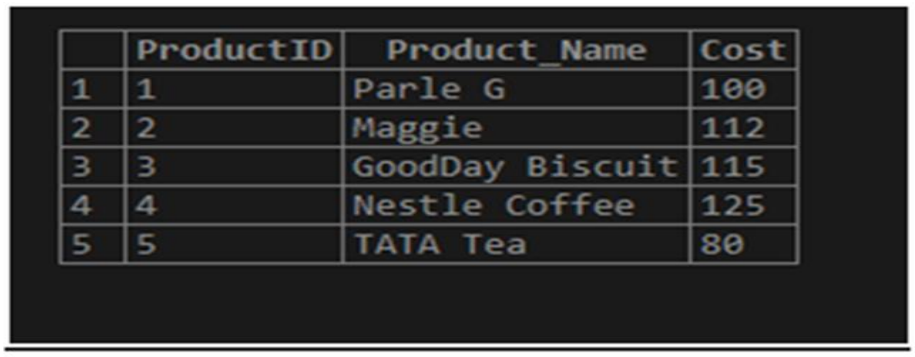
Q22.Which function can be used to fetch yesterdays date? Provide example.

- SYSDATE() will returns today’s date along with timestamp value.
- Below query would return the date and timestamp.
- SELECT DATE_SUB(SYSDATE(), INTERVAL 1 DAY) as previous_day;

Q23.Create two tables Products and Products_Info.



	ProductID	Product_Name	Cost
1	1	Parle G	100
2	2	Maggie	112
3	3	GoodDay Biscuit	150

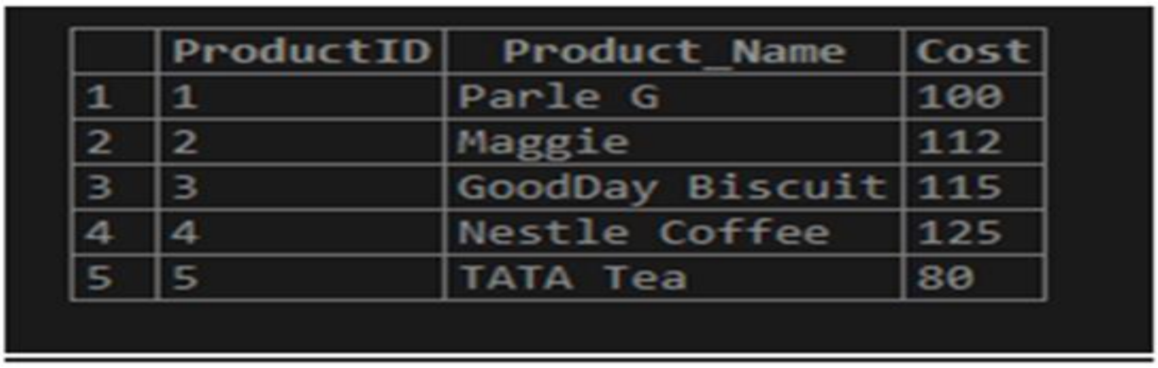


	ProductID	Product_Name	Cost
1	1	Parle G	100
2	2	Maggie	112
3	3	GoodDay Biscuit	115
4	4	Nestle Coffee	125
5	5	TATA Tea	80

•What is MERGE statement?

Merge is part of the DML commands in SQL which can be used either perform INSERT or UPDATE based on the data in the respective table.

```
MERGE Products t

USING Products_Info s

ON (s.ProductID = t.ProductID)


WHEN MATCHED

THEN UPDATE SET

t.Product_Name = s.Product_Name,

t.Cost = s.Cost

WHEN NOT MATCHED BY TARGET

THEN INSERT (ProductID, Product_Name,Cost)

VALUES(s.ProductID,s.Product_Name,s.Cost)

WHEN NOT MATCHED BY SOURCE

THEN DELETE;
```



	ProductID	Product_Name	Cost
1	1	Parle G	100
2	2	Maggie	112
3	3	GoodDay Biscuit	115
4	4	Nestle Coffee	125
5	5	TATA Tea	80

Q24.What is Normalization in a Database?

Normalization is used to minimize redundancy and dependency by organizing fields and table of a database.

- There are some rules of database normalization, which is commonly known as Normal From, and they are:
- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce-Codd normal form(BCNF)

Q25.**What is the difference between a function and a procedure?**

- **Function** should always return a value whereas for a **procedure** it's not mandatory to return a value.
- Function can be called from a SELECT query whereas procedure cannot be called from a SELECT query.
- Function is generally used to perform some calculation and return a result. Whereas procedure is generally used to implement some business logic.

Q26.What is PRAGMA AUTONOMOUS TRANSACTION?

We can declare the stored program like a procedure as a **PRAGMA AUTONOMOUS TRANSACTION** which means that any transaction committed or rolled back in this procedure will not impact any open transactions in the program from where this procedure was called from.

Q27. Write a query to fetch the EmpFname from the EmployeeInfo table in upper case and use the ALIAS name as EmpName.

```
SELECT UPPER(EmpFname) AS EmpName FROM EmployeeInfo;
```

Q28. Write a query to fetch the number of employees working in the department 'HR'.

```
SELECT COUNT(*) FROM EmployeeInfo WHERE Department = 'HR';
```

Q29. Write a query to get the current date.

You can write a query as follows in SQL Server:

```
SELECT GETDATE();
```

You can write a query as follows in MySQL:

```
SELECT SYSTDATE();
```

Q30. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.

```
SELECT SUBSTRING(EmpLname, 1, 4) FROM EmployeeInfo;
```

Q31. Write a query to fetch only the place name(string before brackets) from the Address column of EmployeeInfo table.

Using the MID function in MySQL

```
SELECT MID(Address, 0, LOCATE('(',Address)) FROM EmployeeInfo;
```

Using SUBSTRING

Q32. Write a query to create a new table which consists of data and structure copied from the other table.

Using the SELECT INTO command:

```
SELECT * INTO NewTable FROM EmployeeInfo WHERE 1 = 0;
```

Using the CREATE command in MySQL:

```
CREATE TABLE NewTable AS SELECT * FROM EmployeeInfo;
```

Q33. Write q query to find all the employees whose salary is between 50000 to 100000.

```
SELECT * FROM EmployeePosition WHERE Salary BETWEEN '50000' AND '100000';
```

Q34. Write a query to find the names of employees that begin with 'S'

```
SELECT * FROM EmployeeInfo WHERE EmpFname LIKE 'S%';
```

Q35. Write a query to fetch top N records.

By using the TOP command in SQL Server:

```
SELECT TOP N * FROM EmployeePosition ORDER BY Salary DESC;
```

By using the LIMIT command in MySQL:

```
SELECT TOP N * FROM EmployeePosition ORDER BY Salary DESC;
```

Q36. Write a query to retrieve the EmpFname and EmpLname in a single column as "FullName". The first name and the last name must be separated with space.

```
SELECT CONCAT(EmpFname, ' ', EmpLname) AS 'FullName' FROM EmployeeInfo;
```

Q37. Write a query find number of employees whose DOB is between 02/05/1970 to 31/12/1975 and are grouped according to gender

```
SELECT COUNT(*), Gender FROM EmployeeInfo WHERE DOB BETWEEN '02/05/1970 ' AND '31/12/1975' GROUP BY Gender;
```

Q38. Write a query to fetch all the records from the EmployeeInfo table ordered by EmpLname in descending order and Department in the ascending order.

To order the records in ascending and descnding order, you have to use the ORDER BY statement in SQL.

```
SELECT * FROM EmployeeInfo ORDER BY EmpFname desc, Department asc;
```

Q39. Write a query to fetch details of employees whose EmpLname ends with an alphabet 'A' and contains five alphabets.

To fetch details mathcing a certain value, you have to use the LIKE operator in SQL.

```
SELECT * FROM EmployeeInfo WHERE EmpLname LIKE '____a';
```

Q40. Write a query to fetch details of all employees excluding the employees with first names, "Sanjay" and "Sonia" from the EmployeeInfo table.

```
SELECT * FROM EmployeeInfo WHERE EmpFname NOT IN ('Sanjay','Sonia');
```

Q41. Write a query to fetch details of employees with the address as "DELHI(DEL)".

```
SELECT * FROM EmployeeInfo WHERE Address LIKE 'DELHI(DEL)%';
```

Q42. Write a query to fetch all employees who also hold the managerial position.

```
SELECT E.EmpFname, E.EmpLname, P.EmpPosition
FROM EmployeeInfo E INNER JOIN EmployeePosition P ON
E.EmpID = P.EmpID AND P.EmpPosition IN ('Manager');
```

Q43. Write a query to fetch the department-wise count of employees sorted by department's count in ascending order.

```
SELECT Department, count(EmpID) AS EmpDeptCount
FROM EmployeeInfo GROUP BY Department
ORDER BY EmpDeptCount ASC;
```

Q44. Write a query to calculate the even and odd records from a table.

To retrieve the even records from a table, you have to use the MOD() function as follows:

```
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE
MOD(rowno,2)=0;
```

Similarly, to retrieve the odd records from a table, you can write a query as follows:

```
SELECT EmpID FROM (SELECT rowno, EmpID from EmployeeInfo) WHERE
MOD(rowno,2)=1;
```

Q45. Write a SQL query to retrieve employee details from EmployeeInfo table who have a date of joining in the EmployeePosition table.

```
SELECT * FROM EmployeeInfo E
WHERE EXISTS
(SELECT * FROM EmployeePosition P WHERE E.EmpId = P.EmpId);
```

Q46. Write a query to retrieve two minimum and maximum salaries from the EmployeePosition table.

To retrieve two minimum salaries, you can write a query as below:

```
SELECT DISTINCT Salary FROM EmployeePosition E1
WHERE 2 >= (SELECTCOUNT(DISTINCT Salary)FROM EmployeePosition E2
WHERE E1.Salary >= E2.Salary) ORDER BY E1.Salary DESC;
```

To retrieve two maximum salaries, you can write a query as below:

```
SELECT DISTINCT Salary FROM EmployeePosition E1
WHERE 2 >= (SELECTCOUNT(DISTINCT Salary) FROM EmployeePosition E2
WHERE E1.Salary <= E2.Salary) ORDER BY E1.Salary DESC;
```

Q47. Write a query to find the Nth highest salary from the table without using TOP/limit keyword.

```
SELECT Salary
FROM EmployeePosition E1
WHERE N-1 = (
SELECT COUNT( DISTINCT ( E2.Salary ) )
FROM EmployeePosition E2
WHERE E2.Salary >  E1.Salary );
```

Q48. 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;
```

Q49. 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;
```

Q50. 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;
```

Q51. Write a query 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;
```

Q52. Write a query to display the first and the last record from the EmployeeInfo table.

To display the first record from the EmployeeInfo table, you can write a query as follows:

```
SELECT * FROM EmployeeInfo WHERE EmpID = (SELECT MIN(EmpID) FROM
EmployeeInfo);
```

To display the last record from the EmployeeInfo table, you can write a query as follows:

SELECT * FROM EmployeeInfo WHERE EmpID = (SELECT MAX(EmpID) FROM EmployeeInfo);

Q53. Write a query to add email validation to your database

SELECT Email FROM EmployeeInfo WHERE NOT REGEXP_LIKE(Email, ‘[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}’, ‘i’)

Q54. Write a query to retrieve Departments who have less than 2 employees working in it.

SELECT DEPARTMENT, COUNT(EmpID) as 'EmpNo' FROM EmployeeInfo GROUP BY DEPARTMENT HAVING COUNT(EmpD) < 2;

Q55. Write a query to retrieve EmpPostion along with total salaries paid for each of them.

SELECT EmpPosition, SUM(Salary) from EmployeePosition GROUP BY EmpPosition;

Q56. Write a query to fetch 50% records from the EmployeeInfo table.

SELECT *
FROM EmployeeInfo WHERE
EmpID <= (SELECT COUNT(EmpID)/2 from EmployeeInfo);

Q57. Select nth highest salary from the table.

select salary from Employee ORDER BY salary DESC limit n-1,1;

Q58. Write a query to select last record from the table Student.
SELECT * FROM Student WHERE Rowid = SELECT MAX(Rowid) from Student;

Q59. Write a query to access the first Nth records from the table student. Where N=4.
SELECT * FROM Student WHERE Rownum < = 4;

Q60. Write a query to find all even rows from the table Student.
SELECT * FROM Student WHERE MOD (Rowid,2) = 0 ;

Q61. From the above given tables write a query to find all the students name in which students got less than 90 marks.
SELECT * **FROM** Student **WHERE** Stu_Marks <90;

If we have to print subject names in which students got less than 90 marks.

```
  SELECT Student.Stu_name, Student.Stu_Marks Subject.Subject_name
 FROM Student
  JOIN Subject
      ON Student.Stu_Subject_Id = Subject.Subject_Id
      WHERE Stu_Marks <90;
```

Q62. Write a query to create a new table name Student_marks from the given table Student.
CREATE TABLE Student_Marks SELECT * FROM Student;

Q63. Write a query to show the 3rd highest marks from the Student table.
select Stu_Marks from Student ORDER BY Stu_Marks DESC limit 2,1;

Q63. Write a query to view current date and time.
SELECT GETDATE();

Q64. Write a query to show the record of the three highest marks from the student table.
SELECT (Stu_Marks) FROM (SELECT DISTINCT Stu_Marks from Student ORDER BY Stu_Marks DESC) WHERE ROWNUM<=3;

Q65. Write a query to fetch the stu_Name and Stu_Marks of those students whose age is 20.
SELECT Stu_Name, Stu_Marks FROM Student WHERE Stu_Age = 20;

Q66. Write a query to show the maximum marks of each subject.
Select Student_ID, Stu_Subject_ID, MAX(Stu_Marks) from Student group by Stu_Subject_ID;

Q67. Write a query to show all the record of those students whose marks is greater than 82 and age is 22.
SELECT * FROM Student WHERE Stu_Marks > 82 and Stu_Age = 22;

Q68. Write a query to show the record of those students whose name begins with the 'm' character.
SELECT * FROM Student WHERE Stu_Name LIKE '%m';

Q69. Write a query to show all Subject_Id along with the number of students in there.

SELECT Stu_Subject_ID COUNT(Stu_Subject_ID) as 'Number of Students' FROM Student GROUP BY Stu_Subject_ID;

Q70. Write a query to fetch the values of the Stu_Name column from the Student table in the upper case.

SELECT UPPER(Stu_Name) from Student;

Q71. Write a query to show the unique values of Stu_age from the Student table.

SELECT DISTINCT(Stu_Age) from Student;

Q72. Write a query to view all student details from the Student table order by Stu_Name Descending.

SELECT * FROM Student ORDER BY Stu_Name DESC;

Q73. Write a query to show the three minimum marks from the student table.

SELECT DISTINCT Stu_Marks FROM Student a WHERE 3 <= (SELECT COUNT(DI
STINCT Stu_Marks) FROM Student b WHERE a. Stu_Marks <= b.Stu_Marks) ORD
ER BY a.Stu_Marks DESC;

Q74. Write a query to find the average of marks of Student table.

SELECT AVG (Stu_Marks) **FROM** Student;

Q75. Write a query to add the Stu_Address column to the existing Student table:

ALTER TABLE Student **ADD** Stu_Addressvarchar (25) ;

INDEX