SQL queries:

- ALTER TABLE table_name
  ADD column_name datatype;
- SELECT column_name(s) FROM table_name WHERE column_1 = value_1 AND column_2 = value_2;
  `AND` is an operator that combines two conditions. Both conditions must be true for the row to be included in the result set.

AS

AS is a keyword in SQL that allows you to rename a column or table using an alias.

```
SELECT column_name AS 'Alias' FROM table_name;
```

AVG()

AVG() is an aggregate function that returns the average value for a numeric column.

```
SELECT AVG(column_name) FROM table_name;
```

BETWEEN

The `BETWEEN` operator is used to filter the result set within a certain range. The values can be numbers, text or dates.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value_1 AND value_2;
```

CASE

`CASE` statements are used to create different outputs (usually in the `SELECT` statement). It is SQL's way of handling if-then logic.

```
SELECT column_name,
 CASE
  WHEN condition THEN 'Result_1'
  WHEN condition THEN 'Result_2'
  ELSE 'Result_3'
 END
FROM table_name;
```

COUNT()

`COUNT()` is a function that takes the name of a column as an argument and counts the number of rows where the column is not `NULL`.

```
SELECT COUNT(column_name)
FROM table_name;
```

DELETE

`DELETE` statements are used to remove rows from a table.

```
DELETE FROM table_name
WHERE some_column = some_value;
```

GROUP BY

`GROUP BY` is a clause in SQL that is only used with aggregate functions. It is used in collaboration with the `SELECT` statement to arrange identical data into groups

It returns one record for each group.

GROUP BY queries often include aggregates: COUNT, MAX, SUM, AVG, etc.

A GROUP BY clause can group by one or more columns.

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name;
```

HAVING

`HAVING` was added to SQL because the `WHERE` keyword could not be used with aggregate functions.

```
SELECT column_name, COUNT(*)
FROM table_name
GROUP BY column_name
HAVING COUNT(*) > value;
```

ORDER BY

`ORDER BY` keyword is used to sort the result-set in ascending or descending order either alphabetically or numerically

The `ORDER BY` keyword sorts the records in ascending order by default. To sort the records in descending order, use the `DESC` keyword.

```
SELECT column_name
FROM table_name
ORDER BY column_name ASC | DESC;
```

INNER JOIN

An inner join will combine rows from different tables if the *join condition* is true.

```
SELECT column_name(s)
FROM table_1
JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

OUTER JOIN

An outer join will combine rows from different tables even if the join condition is not met. Every row in the *left* table is returned in the result set, and if the join condition is not met, then `NULL` values are used to fill in the columns from the *right* table.

```
SELECT column_name(s)
FROM table_1
LEFT JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

IS NULL / IS NOT NULL

`IS NULL` and `IS NOT NULL` are operators used with the `WHERE` clause to test for empty values.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NULL;
```

LIKE

`LIKE` is a special operator used with the `WHERE` clause to search for a specific pattern in a column.

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

LIMIT

`LIMIT` is a clause that lets you specify the maximum number of rows the result set will have.

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

MAX()

`MAX()` is a function that takes the name of a column as an argument and returns the largest value in that column.

```
SELECT MAX(column_name)
FROM table_name;
```

MIN()

`MIN()` is a function that takes the name of a column as an argument and returns the smallest value in that column.

```
SELECT MIN(column_name)
FROM table_name;
```

OR

`OR` is an operator that filters the result set to only include rows where either condition is true.

```
SELECT column_name
FROM table_name
WHERE column_name = value_1
  OR column_name = value_2;
```

ROUND()

`ROUND()` is a function that takes a column name and an integer as arguments. It rounds the values in the column to the number of decimal places specified by the integer.

```
SELECT ROUND(column_name, integer)
FROM table_name;
```

SELECT DISTINCT

`SELECT DISTINCT` specifies that the statement is going to be a query that returns unique values in the specified column(s).

```
SELECT DISTINCT column_name
FROM table_name;
```

SUM

`SUM()` is a function that takes the name of a column as an argument and returns the sum of all the values in that column.

```
SELECT SUM(column_name)
FROM table_name;
```

UPDATE

`UPDATE` statements allow you to edit rows in a table.

```
UPDATE table_name
SET some_column = some_value
WHERE some_column = some_value;
```

WHERE

`WHERE` is a clause that indicates you want to filter the result set to include only rows where the following *condition* is true.

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value;
```

WITH

`WITH` clause lets you store the result of a query in a temporary table using an alias. You can also define multiple temporary tables using a comma and with one instance of the `WITH` keyword.

The `WITH` clause is also known as **common table expression (CTE)** and subquery factoring.

```
WITH temporary_name AS (
  SELECT *
  FROM table_name)
SELECT *
FROM temporary_name
WHERE column_name operator value;
```

create an empty table from an existing table?

Example will be -.

```
Select * into studentcopy from student where 1=2;
```

Here, we are copying student table to another table with the same structure with no rows copied.

How to fetch common records from two tables?

Common records result set can be achieved by -.

```
Select studentID from student INTERSECT Select StudentID from Exam
```

How to fetch alternate records from a table?

Records can be fetched for both Odd and Even row numbers -.

To display even numbers-.

```
Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0
```

To display odd numbers-.

```
Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1
```

How to select unique records from a table?

Select unique records from a table by using DISTINCT keyword.

```
Select DISTINCT StudentID, StudentName from Student.
```

What is the command used to fetch first 5 characters of the string?

There are many ways to fetch first 5 characters of the string -.

```
Select SUBSTRING(StudentName,1,5) as studentname from student
```

```
Select LEFT(Studentname,5) as studentname from student
```

Which operator is used in query for pattern matching?

LIKE operator is used for pattern matching, and it can be used as -.

1. % - Matches zero or more characters.
2. _(Underscore) – Matching exactly one character.

Example -.

```
Select * from Student where studentname like 'a%'
```

```
Select * from Student where studentname like 'ami_'
```

EmployeeInfo Table:

EmpID	EmpFname	EmpLname	Department	Project	Address	DOB	Gender
1	Sanjay	Mehra	HR	P1	Hyderabad (HYD)	01/12/1976	M
2	Ananya	Mishra	Admin	P2	Delhi(DEL)	02/05/1968	F
3	Rohan	Diwan	Account	P3	Mumbai(BOM)	01/01/1980	M
4	Sonia	Kulkarni	HR	P1	Hyderabad (HYD)	02/05/1992	F
5	Ankit	Kapoor	Admin	P2	Delhi(DEL)	03/07/1994	M

EmployeePosition Table:

EmpID	EmpPosition	DateOfJoining	Salary
1	Manager	01/05/2019	500000
2	Executive	02/05/2019	75000
3	Manager	01/05/2019	90000
2	Lead	02/05/2019	85000
1	Executive	01/05/2019	300000

Q1. 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;
```

Q2. Write a query to fetch the number of employees working in the department ‘HR’.

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

Q3. 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();
```

Q4. Write a query to retrieve the first four characters of EmpLname from the EmployeeInfo table.

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

Q5. 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

```
SELECT SUBSTRING(Address, 1, CHARINDEX('(',Address)) FROM EmployeeInfo;
```

Q7. Write q query to find all the employees whose salary is between 50000 to 100000.

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

Q8. Write a query to find the names of employees that begin with ‘S’

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

Q9. 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 * FROM EmpPosition ORDER BY Salary DESC LIMIT N;
```

Q10. 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;
```

Q11. 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;
```

Q12. 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.

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

Q13. 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.

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

Q14. Write a query to fetch details of all employees excluding the employees with first names, "Sanjay" and "Sonia" from the EmployeeInfo table.

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

Q15. Write a query to fetch details of employees with the address as "DELHI(DEL)".

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

Q16. Write a query to fetch all employees who also hold the managerial position.

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

Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.
The STATION table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

SELECT (COUNT(CITY)-COUNT(DISTINCT CITY)) FROM STATION;

Query the two cities in **STATION** with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.Consider the above table station.

```
SELECT CITY,LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY),CITY LIMIT 1;
SELECT CITY,LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY) DESC,CITY
LIMIT 1;
```

Query the list of CITY names from **STATION** which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.Consider the above table station.

```
select distinct city from station
where left(city,1) in ('a','e','i','o','u')
and right(city, 1) in ('a','e','i','o','u');
```

Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

Input Format

Column	Type
ID	Integer
Name	String
Marks	Integer

The STUDENTS table is described as follows:

The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.

```
select Name from students where Marks>75 order by
right(Name,3),ID asc;
```

Consider P1(a,c) and P2(b,d) to be two points on a 2D plane where a ,b are the respective minimum and maximum values of Northern Latitude (LAT_N) and c,d are

the respective minimum and maximum values of Western Longitude (LONG_W) in STATION.

Query the Euclidean Distance between points and and format your answer to display decimal digits.

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

The Euclidean Distance is sqrt((a-b)^2 + (c-d)^2)

```
Select
round(sqrt(pow((min(LAT_N)-max(LAT_N)),2)+pow((min(LONG_W)-ma
x(LONG_W)),2)),4) from STATION;
```

Write a query to display three numbers in three columns.

```
Select 5,10,11;
```

Write a query to display a string “hello world”.

```
.Select ‘hello world’;
```

Write a query to display sum of two numbers 10 and 15.

```
Select 10+16;
```

Write a query to display the result of an arithmetic expression.

```
SELECT 10 + 15 - 5 * 2;
```

Write a SQL query to show all the winners in Physics for 1970 together with the winner of Economics for 1971.

```
SELECT * FROM nobel_win  WHERE (subject ='Physics' AND
year=1970) UNION (SELECT * FROM nobel_win  WHERE (subject
='Economics' AND year=1971));
```

Write a SQL query to find all the details of 1970 winners by the ordered to subject and winner name; but the list contain the subject Economics and Chemistry at last.

```
SELECT *
FROM nobel_win
WHERE year=1970
ORDER BY
 CASE
    WHEN subject IN ('Economics','Chemistry') THEN 1
    ELSE 0
 END ASC,
 subject,
 Winner;
```

Write a SQL query to find the item name and price in Rs.

```
SELECT pro_name as "Item Name", pro_price AS "Price in Rs."
    FROM item_mast;
```

Write a SQL query to find the name and price of the cheapest item(s).

```
SELECT pro_name, pro_price
   FROM item_mast
   WHERE pro_price =
    (SELECT MIN(pro_price) FROM item_mast);
```

Write a SQL query to display those customers who are neither belongs to the city New York nor grade value is more than 100.

```
SELECT * FROM customer WHERE NOT (city = 'New York' OR
grade>100);
```

Write a SQL query to display order number, purchase amount, the achieved and unachieved percentage (%) for those order which exceeds the 50% of the target value of 6000.

```
SELECT ord_no,purch_amt,
(100*purch_amt)/6000 AS "Achieved %",
(100*(6000-purch_amt)/6000) AS "Unachieved %"
FROM  orders
WHERE (100*purch_amt)/6000>50;
```

Write a SQL statement to find those salesmen with all other information and name started with other than any latter within 'A' and 'L'.

```
SELECT *
FROM salesman
WHERE name NOT BETWEEN 'A' and 'L';
```

Write a SQL statement to find those salesmen with all information whose name containing the 1st character is 'N' and the 4th character is 'l' and rests may be any character.

```
SELECT *FROM salesman WHERE name LIKE 'N__l%';
```

Write a SQL statement to find those rows from the table testtable which contain the escape character underscore (_) in its column 'col1'.

```
SELECT *FROM testtable WHERE col1 LIKE '%/_%' ESCAPE '/';
```

Write a SQL statement to find those rows from the table testtable which does not contain the character underscore (_) in its column 'col1'.

```
SELECT *FROM testtable WHERE col1 NOT LIKE '%/_%' ESCAPE '/';
```

Write a SQL statement to find those rows from the table testtable which contain the string (_/) in its column 'col1'.

```
SELECT *FROM testtable WHERE col1 LIKE '%/_//%' ESCAPE '/';
```

Write a SQL statement to find that customer with all information who does not get any grade except NULL

```
SELECT *FROM customer WHERE grade IS NULL;
```

Write a SQL statement to find that customer with all information who gets a grade except NULL value.

```
SELECT *FROM customer WHERE grade IS NOT NULL;
```

```
Or SELECT *FROM customer WHERE NOT grade IS NULL;
```

Write a SQL statement find the number of customers who gets at least a gradation for his/her performance.

```
SELECT COUNT(grade) from customer where grade is not NULL;
```

Or `SELECT COUNT (ALL grade) FROM customer;`

Write a SQL statement to find out the number of orders booked for each day and display it in such a format like "For 2001-10-10 there are 15 orders".

```
SELECT ' For',ord_date,',there are', COUNT (DISTINCT
ord_no),'orders.' FROM orders GROUP BY ord_date;
```

Write a SQL statement to find the highest purchase amount with their ID and order date, for only those customers who have highest purchase amount in a day is more than 2000.

```
SELECT customer_id,ord_date,MAX(purch_amt) FROM orders GROUP BY

customer_id,ord_date HAVING MAX(purch_amt)>2000.00;
```

Sno	Where Clause	Having Clause
1	The WHERE clause specifies the criteria which individual records must meet to be selcted by a query. It can be used without the GROUP BY clause	The HAVING clause cannot be used without the GROUP BY clause.
2	The WHERE clause selects rows before grouping.	The HAVING clause selects rows after grouping.
3	The WHERE clause cannot contain aggregate functions	The HAVING clause can contain aggregate functions.
4	WHERE clause is used to impose condition on SELECT statement as well as single row function and is used before GROUP BY clause	HAVING clause is used to impose condition on GROUP Function and is used after GROUP BY clause in the query
5	SELECT Column,AVG(Column_nmae)FROM Table_name WHERE Column > value GROUP BY Column_nmae	SELECT Columnq, AVG(Column_nmae)FROM Table_name WHERE Column > value GROUP BY Column_nmae Having column_name>or<value

Write a SQL statement that count the number of salesmen for whom a city is specified. Note that there may be spaces or no spaces in the city column if no city is specified.

```
SELECT COUNT(*) FROM salesman WHERE city IS NOT NULL;
```

Write a SQL statement to display the commission with the percent sign (%) with salesman ID, name and city columns for all the salesmen.

```
SELECT salesman_id,name,city,'%',commission*100 FROM salesman;
```

Write a query to find those customers with their name and those salesmen with their name and city who lives in the same city.

```
SELECT customer.cust_name,salesman.name, salesman.city

FROM salesman, customer WHERE salesman.city = customer.city;
```

The SQL UNION Operator

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
- The `UNION` operator selects only distinct values by default. To allow duplicate values, use `UNION ALL`:

```
SELECT 'Customer' AS Type, ContactName, City, Country

FROM Customers

UNION

SELECT 'Supplier', ContactName, City, Country

FROM Suppliers;
```

The SQL EXISTS Operator

The `EXISTS` operator is used to test for the existence of any record in a subquery.

The `EXISTS` operator returns TRUE if the subquery returns one or more records.