SQL

1. Is an acronym of the Structured Query Language and it is used for accessing and manipulating the databases.
2. Using SQL we can execute queries against a database to retrieve data from database, insert, update and delete records in a database.
3. RDBMS is the Relational Database Management System.
4. The data is RDBMS is stored in database objects called tables.
5. A table is a collection of related data entries and it has rows and columns.
6. Table is broken up into smaller entries called as fields. Field is a column in a table that is column header.
7. Record is also called as row indicates individual entry.(Horizontal entry)
8. Column is information about specific field of a record.(Vertical entry)

```
SELECT * FROM Customers;
```
➔selects all records in the customers table
SELECT TOP 6 * FROM Customers;→Selects and shows only 6 records
SELECT * FROM Customers LIMIT 2;→Selects and shows only 2 records

```
SELECT col1,col2 FROM Customers;
```
➔select particular columns from customers table

```
SELECT DISTINCT col1 FROM Customers;
```
→ select and displays only the distinct records will not show duplicate records

```
SELECT COUNT(DISTINCT col1) FROM Customers; →It displays the count of distinct values

SELECT * FROM Customers WHERE Country='Mexico'; →It displays the record which matches the condition. For text single quotes is mandatory and for numbers quotes is not required
```

= , > , < , >= , <= , != or <> , BETWEEN , LIKE , IN

```
SELECT * FROM Products WHERE Price BETWEEN 50 AND 60; will display records between 50 and 60
SELECT * FROM Customers WHERE POSTALCODE LIKE '7%';→Will dislay postal code starting with 7
SELECT * FROM Customers WHERE City IN ('Paris','London');→Will display records with city name matching paris and london
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);→Will display records whose countries match the supplier countries
```

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database

SELECT * FROM *table_name* WHERE *condition1* AND *condition2*→returns value that matches both conditions

SELECT * FROM *table_name* WHERE *condition1* or *condition2*→returns value that matches any one condition

SELECT * FROM Customers WHERE NOT Country='Germany'; →returns all value except the not condition

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

Order By:
select * from customers order by customerid; displays records in the ascending order.

select * from customers order by customerid desc; displays records in the descending order.

Insert: Used to enter new record

INSERT INTO tablename (columnname1, columnname2) VALUES ('prakash','manoharan');→Used to enter values only on the specific columns.

INSERT INTO tablename VALUES ('95', 'jerry', 'benn', 'francis', 'test', '4006', 'Norway');→User to enter values on all the columns then column name is not required

To get null records:

select * from customers where address is null;→Will display null records

select * from customers where address is not null; →Will display all records expect null records

Update:

```
UPDATE tablename SET column1 = value1 WHERE condition;→Update the values of column1 which matches the specific condition

UPDATE tablename SET column1 = value1;→Update the values of column1 for all the records since no where condition
```

DELETE:

```
DELETE FROM tablename; →Will delete all records in the table
DELETE FROM tablename where columnname=value; →Deletes records which matches the condition
```

MAXIMUM AND MINIMUM:

```
SELECT Max(CustomerId) FROM Customers; →shows the maximum value from customerid column
SELECT Min(CustomerId) FROM Customers; →shows the minimum value from customerid column
```

COUNT, AVERAGE, SUM:

```
SELECT COUNT(column_name) FROM table_name;→Shows all count
SELECT AVG(column_name) FROM table_name;→Shows average value of column mentioned
SELECT SUM(column_name) FROM table_name;→Shows sum of mentioned numeric column
```

LIKE:

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"

WILDCARD CHARACTERS IN SQL:

%	bl% finds bl, black, blue, and blob
_	h_t finds hot, hat, and hit
[]	h[oa]t finds hot and hat, but not hit
^	h[^oa]t finds hit, but not hot and hat
-	c[a-b]t finds cat and cbt

Aliases:
Aliases are used to give the temporary names.
`SELECT column_name AS alias_name FROM table_name;` →For column
`SELECT column_name(s) FROM table_name AS alias_name;` →For table

Aliases can be useful when:

- There are more than one table involved in a query
- Functions are used in the query
- Column names are big or not very readable
- Two or more columns are combined together

Join:

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
Inner join will show only matching records

select orderdetails.orderid, products.productname, orderdetails.quantity
from orderdetails inner join products on
orderdetails.productid=products.productid →Two tables

select orders.orderid, orders.customerid, customers.customername,
orders.employeeid, employees.firstname
from orders
inner join customers on orders.customerid = customers.customerid inner join
employees on orders.employeeid = employees.employeeid; →3 tables

select orders.orderid, orders.customerid, customers.customername,
orders.employeeid, employees.firstname, orders.shipperid
from orders
inner join customers on orders.customerid = customers.customerid
inner join employees on orders.employeeid = employees.employeeid
inner join shippers on orders.shipperid = shippers.shipperid; →4 tables

LEFT JOIN:
The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.
The LEFT JOIN keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

```
SELECT * FROM ORDERS LEFT JOIN CUSTOMERS ON
ORDERS.CUSTOMERID=CUSTOMERS.CUSTOMERID;
```

Right join:
The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders).

```
SELECT * FROM ORDERS right JOIN CUSTOMERS ON
ORDERS.CUSTOMERID=CUSTOMERS.CUSTOMERID;
```

FULL OUTER JOIN:

The FULL OUTER JOIN keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID;
```

SELF JOIN:

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City FROM Customers A, Customers WHERE A.CustomerID <> B.CustomerID AND A.City = B.City;

UNION: Will merger and show only distinct city records from both the tables

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers;
```

UNION ALL: Will merger and show all city records from both the tables

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers;
```