DATABASE:

```
CREATE DATABASE database_name;
```
Creates a new database.

```
USE database_name;
```
Uses the specified database

DDL COMMANDS:

```
CREATE TABLE table_name (
       column1 datatype,
       column2 datatype,
       column3 datatype);
```
The create table statement creates a new table in a database.

```
ALTER TABLE table_name ADD column_name datatype;
```
The Alter table statement is used to modify the columns of an existing table and add a new column.

```
ALTER TABLE table_name DROP COLUMN column_name ;
```
The Alter table statement is used to modify the columns of an existing table and Drop column.

```
ALTER TABLE table_name RENAME TO table_newname ;
```
Changes the table name for the existing table.

```
ALTER TABLE table_name RENAME col_name TO col_newname;
```
Renames the column names in the existing table.

```
Drop table table_name;
```
Drop deletes both structure and records in the table.

```
Truncate table table_name;
```
Truncate deletes the table but not the structure.

DML COMMANDS:

```
INSERT INTO table_name
VALUES (value1, value2);
```

```
INSERT INTO table_name (column1, column2)
     VALUES (value1, value2);
```

The Insert into statement is used to add a new record (row) to a table.

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

The delete statement is used to delete records (rows) in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2
WHERE some_column = some_value;
```

The UPDATE statement is used to edit records (rows) in a table.

DCL COMMANDS:

```
GRANT SELECT, UPDATE ON TABLE_1 TO USER_1, USER_2;
```

Used to grant a user access privileges to a database.

```
REVOKE SELECT, UPDATE ON TABLE_1 FROM USER_1, USER_2;
```

Used to revoke the permissions from a user.

TCL COMMANDS:

COMMIT; - Saves all the transactions made on a database.
ROLLBACK; - It is used to undo transactions which are not yet been saved.
SAVEPOINT savepoint_name; - Used to roll the transaction back to a certain point without having to roll back the entirety of the transaction.

DQL COMMANDS:

SELECT col1,col2.. FROM table_name;
Retrieve data from specified columns in the table

SELECT * FROM table_name;
Retrieve the data from all fields in the table.

SELECT col1,col2..FROM table_name WHERE condition;
Used to filter the records based on a particular condition.

SQL Constraints:

NOT NULL: Specifies that this column cannot store a NULL value.
UNIQUE: Specifies that this column can have only Unique values.
Primary Key: It is a field using which it is possible to uniquely identify each row in a table.
Foreign Key: It is a field using which it is possible to uniquely identify each row in some other table.
CHECK: It validates if all values in a column satisfy some particular condition or not
DEFAULT: It specifies a default value for a column when no value is specified for that field

Operators:

AND - The AND operator allows multiple conditions to be combined. Records must match both conditions.

OR - The OR operator allows multiple conditions to be combined. Records match either condition.

NOT - The NOT operator allows the negotiation of the condition.

BETWEEN - The BETWEEN operator can be used to filter by a range of values.

LIKE - The LIKE operator can be used inside of a WHERE clause to match a specified pattern.

% Wildcard - The % wildcard can be used in a LIKE operator pattern to match zero or more unspecified character(s).

_ Wildcard - The _ wildcard can be used in a LIKE operator pattern to match any single unspecified character.

IN - The IN operator is used to compare the specified value.

AS - Columns or tables can be aliased using the AS clause.

ALL - It compares a value to all the values in another set.

ANY - It compares the values in the list according to the condition.

EXIST - It is used to search for the presence of a row in a table.

SELECT column_name FROM table_name WHERE column_name IS NULL;
Column values can be NULL or have no value. These records can be matched using the IS NULL and IS NOT NULL operators.

SELECT col1, col2 FROM table_name
UNION
SELECT col1, col2 FROM table_name;
Combine rows from two queries without any duplicates.

SELECT col1, col2 FROM table_name
UNION ALL
SELECT col1, col2 FROM table_name;
Combine rows from two queries with duplicates.

SELECT col1, col2 FROM table_name
INTERSECT
SELECT col1, col2 FROM table_name;
Return the common rows of two queries.

SELECT col1, col2 FROM table_name

MINUS
SELECT col1, col2 FROM table_name;
Returns the values from the first table after removing the values from the second table.

Querying Data:

SELECT DISTINCT(column_name) FROM table_name;
Unique values of the columns are retrieved from the table.

SELECT * fROM table_name LIMIT 5;
Limit is used to limit the result set to the specified number of rows.

SELECT col1, col2 FROM table_name ORDER BY col1 ASC [DESC];
Sort the result set in ascending or descending order

SELECT col1, col2 FROM table_name ORDER BY col1 LIMIT n OFFSET offset;
Skip offset of rows and return the next n rows based on LIMIT.

SELECT col1, aggregate(col2) FROM table_name GROUP BY col1;
GROUP BY Groups rows using an aggregate function

SELECT col1, aggregate(col2) FROM table_name GROUP BY col1 HAVING condition;
Filter groups using the HAVING clause.

DESC table_name;
Describes the structure of the table.

JOINS:

SELECT col1, col2 FROM table_name t1 INNER JOIN table_name t2 ON condition;
Inner join of two tables t1 and t2

SELECT col1, col2 FROM table_name t1 LEFT JOIN table_name t2 ON condition;
Left join of two tables t1 and t2

SELECT col1, col2 FROM table_name t1 RIGHT JOIN table_name t2 ON condition;
Right join of two tables t1 and t2

SELECT col1, col2 FROM table_name t1 FULL OUTER JOIN table_name t2 ON condition;
Full outer join of two tables t1 and t2

SELECT col1, col2 FROM table_name t1 CROSS JOIN table_name t2 ON condition;
Produce a Cartesian product of rows in tables

SELECT col1, col2 FROM table_name t1 NATURAL JOIN table_name t2 ON condition;
Takes all the Key columns from t1 and tries to match with t2 columns.

AGGREGATE FUNCTIONS:

AVG() - returns the average of a list
SUM() - returns the total of a list.
COUNT() - returns the number of elements of a list.
MIN() - returns the minimum value of a list.
MAX() - returns the maximum value of a list.

CASE:

```
SELECT column_name,
CASE
WHEN Condition THEN 'output'
WHEN Condition THEN 'output'
.
.
ELSE 'output'
END 'new_colname' FROM table_name;
```
It works similarly to IF-ELSE and returns in the new column.

SUBQUERY:

```
SELECT COUNT(*) from(SELECT col1,COUNT(col2) from table_name
GROUP BY col1) AS inner_query WHERE condition;
```

First, the inner query executes later which the result is passed to the outer query and it is executed.

Advanced Aggregate functions:

over()- It is a window function used inside every analytical function.

Partition by - Creates a partition internally and later performs the specified operations.

row_number() - Provides row numbers for all the rows based on a specified column in the table.

rank() - Ranking is assigned to the rows based on a specified column. Skips the rank when it contains the same values.

dense_rank() - Ranking is assigned to the rows based on a specified column. Ranks are not skipped.

percent_rank() - Assigns the rank to the specified column within the range of 0-1.

lag()- The first value becomes NULL. Compares the current value with the previous value.

lead() - The last value becomes NULL. Compares the current value with the next value.

first_value() - Gives the first value to all rows.

last_value() - Gives the last value to all rows.

Nth value()- Gives Nth value to all rows.

NTILE() - Divides the rows to ‘n’ number of small buckets.

cume_dist() - The cumulative percentage of the records is calculated from the first row to the current row for the specified column.

VIEWS:

SELECT VIEW view_name AS SELECT * FROM table_name;
It creates a simple view.

SELECT VIEW view_name AS SELECT col1, col2 FROM table_name t1 INNER JOIN table_name t2 ON condition;
It creates a complex view

CREATE RECURSIVE VIEW view_name AS
select-statement -- anchor part
UNION [ALL] select-statement; -- recursive part
It Creates a recursive view

CREATE TEMPORARY VIEW view_name AS SELECT col1, col2 FROM table_name;
It Creates a temporary view

DROP VIEW view_name;
Delete a view

SQL Triggers:

CREATE OR MODIFY TRIGGER trigger_Name (Before | After) [Insert | Update | Delete] on [Table_Name] [for each row | for each column] [trigger_body]
Create or Modify the trigger.

DROP TRIGGER trigger_name;
Drop an already existing trigger from the table

SHOW TRIGGERS IN database_name;
Display all the triggers that are currently present in the table.

All query elements are processed in a very strict order: Query execution order.

- **FROM** - the database gets the data from tables in FROM clause and if necessary performs the JOINs,
- **WHERE** - the data are filtered with conditions specified in the WHERE clause,
- **GROUP BY** - the data are grouped by conditions specified in the WHERE clause,
- **Aggregate functions** - the aggregate functions are applied to the groups created in the GROUP BY phase,
- **HAVING** - the groups are filtered with the given condition,
- **Window functions**,
- **SELECT** - the database selects the given columns,
- **DISTINCT** - repeated values are removed,
- **UNION**/**INTERSECT**/**EXCEPT** - the database applies set operations,
- **ORDER BY** - the results are sorted,
- **OFFSET** - the first rows are skipped,
- **LIMIT/FETCH/TOP** - only the first rows are selected