Back to Top

Types of SQL Commands/subsets of SQL?

- DDL (Data Definition Language):

    - **CREATE:** Creates a new table or database.
    - **ALTER:** Modifies an existing database object.
    - **DROP:** Deletes an entire table, database, or other objects.
    - **TRUNCATE:** Removes all records from a table, deleting the space allocated for the records.

- DML (Data Manipulation Language):

    - **SELECT:** Retrieves data from the database.
    - **INSERT:** Adds new data to a table.
    - **UPDATE:** Modifies existing data within a table.
    - **DELETE:** Removes data from a table.

- DCL (Data Control Language):

    - **GRANT:** Gives users access privileges to the database.
    - **REVOKE:** Removes access privileges given with the GRANT command.

- TCL (Transaction Control Language):

- **COMMIT:** Saves all changes made in the current transaction.
- **ROLLBACK:** Restores the database to the last committed state.
- **SAVEPOINT:** Sets a savepoint within a transaction.
- **SET TRANSACTION:** Places a name on a transaction.

Show tables;

```
SHOW TABLES;
+----------------------+
| Tables_in_employee123 |
+----------------------+
| employee_table        |
+----------------------+
```

select table

```
use database_name;
```

Create table

```
CREATE TABLE table_name(
    id int NOT NULL AUTO_INCREMENT,
    name varchar(45) NOT NULL,
    occupation varchar(35) NOT NULL,
    age int NOT NULL,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE table_name
(
id int AUTO_INCREMENT primary key,
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
..
);
```

- Create table with **primary key**

```
CREATE TABLE Employee(
EmployeeID NOT NULL,
FirstName varchar(255) NOT NULL,
LastName varchar(255),
```

```
City varchar(255),
PRIMARY KEY (EmployeeID)
);
```

Insert table

```
INSERT INTO table_name
(column1,column2,column3,...)
VALUES
('value1','value2','value3',...);
```

Rename Table

```
RENAME old_table_name To new_table_name;
OR
RENAME table old_table_name TO new_table_name;
OR
ALTER TABLE old_table_name RENAME TO new_table_name;
```

RENAME column in table

- syntex

```
ALTER TABLE table_name CHANGE COLUMN old_name new_name column_definition [ FIRST |
AFTER column_name ]
```

```
ALTER TABLE employee_table CHANGE COLUMN name first_name varchar(20) NOT NULL;
```

Delete/Drop table

```
DROP TABLE table_name;
```

DROP column in table

```
ALTER TABLE employee_table DROP COLUMN unique_id4;
```

Delete row from the database

- DELETE is a DML(Data Manipulation Language) command and is used when we specify the row (tuple) that we want to remove or delete from the table or relation. The DELETE command can contain a WHERE clause.

```
delete from table_name where ID=01;
delete from table_name where ID IN(2,6);
```

Truncate

- A truncate SQL statement is used to remove all rows (complete data) from a table.
- TRUNCATE is a DDL(Data Definition Language) command and is used to delete all the rows or tuples from a table. Unlike the DELETE command, the TRUNCATE command does not contain a WHERE clause.

```
TRUNCATE TABLE table_name;
```

See the table structure:-

```
DESCRIBE employee_table;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| name       | varchar(45) | NO   |     | NULL    |                |
| occupation | varchar(35) | NO   |     | NULL    |                |
| age        | int(11)     | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

See the table structure:-

```
DESCRIBE employee_table;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| name       | varchar(45) | NO   |     | NULL    |                |
| occupation | varchar(35) | NO   |     | NULL    |                |
| age        | int(11)     | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

↑ Back to top

ALTER Table

```
# syntex
ALTER TABLE table_name
ADD new_column_name datatype
[ FIRST | AFTER column_name ];
```

- ADD a column in the table

```
ALTER TABLE employee_table ADD cus_age varchar(40) NOT NULL;

DESCRIBE employee_table;
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| id         | int(11)     | NO   | PRI | NULL    | auto_increment |
| name       | varchar(45) | NO   |     | NULL    |                |
| occupation | varchar(35) | NO   |     | NULL    |                |
| age        | int(11)     | NO   |     | NULL    |                |
| cus_age    | varchar(40) | NO   |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

- Add column after occupation

```
ALTER TABLE employee_table ADD after_occupation varchar(40) NOT NULL AFTER
occupation;

+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| name            | varchar(45) | NO   |     | NULL    |                |
| occupation      | varchar(35) | NO   |     | NULL    |                |
| after_occupation| varchar(40) | NO   |     | NULL    |                |
| age             | int(11)     | NO   |     | NULL    |                |
| cus_age         | varchar(40) | NO   |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

- Add column first occupation

```
ALTER TABLE employee_table ADD COLUMN unique_id INT(11) NOT NULL FIRST;

+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| unique_id       | int(11)     | NO   |     | NULL    |                |
| id              | int(11)     | NO   | PRI | NULL    | auto_increment |
| name            | varchar(45) | NO   |     | NULL    |                |
```

```
| occupation       | varchar(35) | NO   |     | NULL    |                |
| after_occupation | varchar(40) | NO   |     | NULL    |                |
| age              | int(11)     | NO   |     | NULL    |                |
| cus_age          | varchar(40) | NO   |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

- Add multiple columns in the table

```
ALTER TABLE employee_table
    ADD COLUMN unique_id1 INT(11) NOT NULL FIRST,
    ADD COLUMN unique_id2 INT(11) NOT NULL FIRST;


+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| unique_id2       | int(11)     | NO   |     | NULL    |                |
| unique_id1       | int(11)     | NO   |     | NULL    |                |
| unique_id        | int(11)     | NO   |     | NULL    |                |
| id               | int(11)     | NO   | PRI | NULL    | auto_increment |
| name             | varchar(45) | NO   |     | NULL    |                |
| occupation       | varchar(35) | NO   |     | NULL    |                |
| after_occupation | varchar(40) | NO   |     | NULL    |                |
| age              | int(11)     | NO   |     | NULL    |                |
| cus_age          | varchar(40) | NO   |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

- Add enum column

```
ALTER TABLE table_name ADD field_name enum('0','1') NOT NULL DEFAULT '0' after
password;
```

Change Datatype from alter cmd

```
ALTER TABLE employee_table MODIFY unique_id4 VARCHAR(40) NULL;
OR
ALTER TABLE employee_table MODIFY COLUMN unique_id4 VARCHAR(40) NULL;
```

Ques. Difference between Delete, Truncate & Drop?

Delete	Truncate	Drop
Delete is a DML command	Truncate is DDL command	
We can use where clause in delete command	We cannot use where clause with truncate	

Delete	Truncate	Drop
Delete statement is used to delete a row from a table	Truncate statement is used to remove all the row from a table	Remove table and data
You can rollback data after using delete statement	It is not possible to rollback after using TRUNCATE statement.	Can't rollback
Delete is slower	Truncate is faster	

Ques. Difference b/w DROP and TRUNCATE statements?

- **Drop Table:-**
    - Table structure will be dropped
    - Relationship will be dropped
    - Integrity constraints will be dropped
    - Access privileges will also be dropped
- **TRUNCATE Table:-**
    - On the other hand when we TRUNCATE a table, the table structure remains the same, so you will not face any of the above problems.

UPDATE

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

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

Update customer set name="mohit" where id =1;
```