Basic SQL Commands

Creating a Table

`CREATE TABLE` creates a table with the given name. You must have the `CREATE` privilege for the table.

```
CREATE TABLE tbl_name
(
create_definition
);
```

- `tbl_name`
  The table name can be specified as *`db_name.tbl_name`* to create the table in a specific database. This works regardless of whether there is a default database, assuming that the database exists. If you use quoted identifiers, quote the database and table names separately. For example, write `` `mydb`.`mytbl` ``, not `` `mydb.mytbl` ``.

- SQL commands are case-insensitive
- ; is known as delimiter (it indicates end of command)

INSERT

`INSERT` inserts new rows into an existing table.

```
INSERT into tbl_name
values (value1, value2,...);
```

INSERT values in table corresponding to mentioned column

```
INSERT INTO tbl_name (column1, column2, column3)
VALUES (value1, value2, value3);
```

Recomended: More readable as we know which value is going in which column
a. Readable
b. Flexible
c. In future, if you alter the table, if you add a column, then you don't have to modify the existing INSERT statement

INSERT multiple rows in TABLE

`INSERT` statements that use `VALUES` syntax can insert multiple rows. To do this, include multiple lists of

comma-separated column values, with lists enclosed within parentheses and separated by commas. Example:

```
INSERT INTO _tbl_name_ (a,b,c)
VALUES(1,2,3), (4,5,6), (7,8,9);
```

Each values list must contain exactly as many values as are to be inserted per row. The following statement is invalid because it contains one list of nine values, rather than three lists of three values each:

```
INSERT INTO _tbl_name_ (a,b,c) VALUES(1,2,3,4,5,6,7,8,9);
```

Where emp table has multiple columns but we are only inserting rows with values for two columns

```
INSERT into emp(empno, sal)
values ('3', 6000);
```

- otherthan empno and sal other column will store null values
- null means nothing
- zero, blank space is not equal to null
- special treatment given to null values in all RDBMS
- null is having ASCII value 0
- null ius independent of datatypes
- null value occupies only 1 Byte of storage
- if row is ending with null values, then all such columns won't occupy any space
- it's recommended that those columns that are likely to have large number of null values, should preferably be specified at the end of the table structure, to conserve on harddisk space

```
INSERT into emp
values ('4', 'Atul');           <- Will throw error
```

```
INSERT into emp
values ('4', 'Atul', null, null, null);
```

```
INSERT into emp
values ('4', 'Atul', null, null, null);
```

```
INSERT into emp
values ('5', null, 5000, null, null)
```

```
INSERT into emp(empno, sal) values
('1', 5000),
('2', 6000),
('3', 7000),
('4', 8000),
('5', 5000);
```

SELECT

`SELECT` is used to retrieve rows selected from one or more tables, and can include `UNION` statements and subqueries.
A `SELECT` statement can start with a `WITH` clause to define common table expressions accessible within the `SELECT`.
A select list consisting only of a single unqualified `*` can be used as shorthand to select all columns from all tables:

```
SELECT * FROM tbl_name;
```

Select specific columns from a table:

```
SELECT column1, column2 FROM tbl_name;
```

WHERE

```
SELECT * from emp
WHERE deptno = 10;
```

```
SELECT * from emp
WHERE sal > 2000;
```

Relational Operators with precedence :

1.
2. =

3. <
4. <=
5. != or <>
6. =

```
SELECT * from emp
WHERE sal > 2000 and sal < 3000;
```

Logical Operators with precedence :

1. NOT
2. AND
3. OR

```
SELECT * from emp
WHERE deptno = 1 OR sal > 2000 AND sal < 3000;

SELECT * from emp
WHERE (deptno = 1 OR sal > 2000) AND sal < 3000;

SELECT * from emp
WHERE sal > 2000 OR sal < 3000;

SELECT * from emp
WHERE job = 'MANAGER';
```

In Oracle :

- Queries are case-sensitive
- not user-friendly
- more secure

In MySQL :

- Queries are case-insensitive
- User-friendly
- less secure

```
SELECT * from emp
WHERE job = 'MANAGER' or job = 'CLERK';

SELECT * from emp
WHERE job = 'MANAGER' and job = 'CLERK';

SELECT ename, sal from emp;

SELECT ename, sal, sal * 12 from emp;
```

Arithmetic Operators with precedence :

1. ()
2. /
3.
    -
4.
    -
5.
    -

```
SELECT ename, sal, sal*12 from emp;
```

sal *12*
- computed column, derived column, fake column, Pseudocolumn
*- computed columns, we shall not save in the table (that's a wastage of hard-disk space); as and when required we can SELECT sal*12

Alias -
SELECT ename, sal, sal*12 as "ANNUAL" from emp;*
*SELECT ename, sal, sal*12 as annual from emp;
SELECT ename, sal, sal*12 as "annual" from emp;*
*SELECT ename, sal, sal*12 as "annual salary" from emp;
SELECT ename, sal, sal*12 as annual salary from emp;

```
SELECT ename "EMPNAME",
sal "SALARY",
sal*12 "ANNUAL"
from emp;

SELECT ename "EMPNAME",
sal "SALARY",
sal*12 "ANNUAL"
sal*12*0.4 "HRA",
sal*12*0.2 "DA",
sal*12 + sal*12*0.4 + sal*12*0.2 "NET"
from emp;
```

- you can not use alias in an expression

```
SELECT ename "EMPNAME",
sal "SALARY",
sal*12 "ANNUAL"
from emp
WHERE annual < 300000;          -> Error
```

- cannot use alias in where clause

DISTINCT
SELECT job from emp;

too suppress the duplicate values -

```
SELECT distinct job from emp;
```

- Whenever you use DISTINCT, sorting takes place internally in the Server RAM
- DISTINCT should be first command after SELECT
- sorting is one operation that slows down the working of SELECT statement

```
SELECT distinct job, ename from emp;
```

- DISTINCT will operate on all columns combined as one

```
SELECT (distinct job), ename from emp;  - Error
```

When we install MySQL, 2 users are automatically created -

root

- has DBA privileges
- create users, assign privileges, take backups, restore from the backups, performance planning/monitoring/tuning, etc.

mysql.sys

- most important user
- owner of database and owner of system tables
- startup database,shutdown database, perform recovery

To connect to MySQL Command Line Client -
Open MySQL Command Line Client
Enter password:

To connect to MySQL Workbench -

1. Open MySQL workbench
2. MySQL Connections: (Click on + sign to create a new connection)

3. Connection name: connection for root user
4. Connection method: standard (TCP/IP)
   for LAN - ipx
   - decnet
5. Hostname: DB server name or DB server IP address (localhost)
6. Port: 3306
   Oracle Port Number: 1521
7. Username: root
8. Password: (Store in vault - push button) → Click on it and enter the password - cdac
9. Default Schema: Leave blank
10. Test Connection (push button) → click on it
11. Ok
12. Ok

To login to MySQL database -

1. Click on the connection you created
2. You wil lsee an Object Navigator on LHS
3. You will see a Pull down menu and Horizontal toolbar
4. You will see query window at top
5. You will see output window below

Post Logon Commands-

- ctrl+enter to execute
- (MySQL Commands)
  show databases;
  - Shows all the databases
- to connect to a database -
  use
  use mysql

```
SELECT * from user;
```

- user is a MySQL created system table
- total of 63 system tables in MySQL
- set of system tables is known as DATA DICTIONARY (also known as CATALOG)

Creating username :
create user identified by ;

```
create user anuj@'%' identified by 'welcome123';
```

```
password is case-sensitive
username is not case-sensitive
```

Creating database
create databse ;
or
create schema;

- schema = database

```
create database mydatabase;
-- or
create schema mydatabase;
```

To grant the permissions -

- Click on server (menu at the top)
- Users and Privileges → click on it
- Select the username you created from the user accounts list on LHS
- Click on Administrative Roles tab
- Click on DBA checkbox → click on Apply push button
- Click on schema privileges tab
- Add Entry (push button) → click on it
- Selected Schema (radio button) → click on it
- click on polist
- Select creaded databse
- Ok
- Select "All" (push button) → Click on it
- Select "Grant Option" Checkbox
- Click on Apply
- Exit from MySQL workbench
- Open MySQL Workbench
- Create a new connection for user - anuj
- Default Schema: mydatabase

```
show tables;
```

```
create table emp
(
empno char(4),
ename varchar(25),
sal float,
city varchar(15),
dob date
);
```

```
desc emp;
```

(describe) → it will show structure of emp table