**CRUD**

**CREATE**

* **Insert**
* **Insert variation**
* **Insert multiple**

CREATE DABATABASE campusx

CREATE TABLE users(

user\_id INTEGER PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

)

INSERT INTO campusx.users (user\_id, name, email, password)

VALUE (NULL, 'Batman', 'batsy@gmail.com', 'bruce\_wayne')

* Or we can create a row by

INSERT INTO campusx.users

VALUE (NULL, 'Batman', 'batsy@gmail.com', 'bruce\_wayne')

* Insert multiple values

INSERT INTO campusx.users VALUES

(NULL, 'Albert', 'al@gmail.com', '5456'),

(NULL, 'Tanmay', 'Tanmay15@gmail.com', '356465'),

(NULL, 'Rohan Joshi', 'rohan@gmail.com', '098')

* How to import data in mysql workbench

**Retrieve**

* Select all

SELECT \* FROM campusx.smartphones WHERE 1

When we use WHERE 1 we mean we want to see ALL the rows and ALL the columns

* Filter cols

SELECT os,model, battery\_capacity FROM campusx.smartphones;

* Alias -> renaming using cols

SELECT os AS 'Operating System', model, battery\_capacity AS 'battery\_capacity (mAh)' FROM campusx.smartphones;

* Create expression using cols

Example 1 -

SELECT model,

SQRT(resolution\_width \* resolution\_width + resolution\_height \* resolution\_height)/screen\_size AS 'ppi value'

FROM campusx.smartphones;

Example 2 -

SELECT model, rating/10 AS 'Rating' FROM campusx.smartphones;

* Constants

SELECT model, 'smartphone' AS 'type' FROM campusx.smartphones;

This is useful in creating a column named type with all of its value as smartphone.

* Distinct(uinque) values from a col

SELECT DISTINCT(brand\_name) AS 'All Brands'

FROM campusx.smartphones;

SELECT DISTINCT(processor\_brand) AS 'All\_processors'

FROM campusx.smartphones;

SELECT DISTINCT(os) AS 'All\_OS'

FROM campusx.smartphones;

Gives out all the unique values from that column

* Distinct combos

SELECT DISTINCT brand\_name, processor\_brand

FROM campusx.smartphones;

* Filter rows WHERE clause
  + 1. Find all Samsung phones

SELECT \* FROM campusx.smartphones WHERE brand\_name = 'samsung'

* + 1. Find all phones with price > 50000

SELECT \* FROM campusx.smartphones WHERE price > 50000

* BETWEEN

1. Find all phones in the price range of 10000 and 20000

SELECT \* FROM campusx.smartphones

WHERE price > 10000 and price < 20000

This was before but for BETWEEN below is the code

SELECT \* FROM campusx.smartphones

WHERE price BETWEEN 10000 AND 20000

1. Find phones with rating > 80 and price < 25000

SELECT \* FROM campusx.smartphones

WHERE price < 25000 AND rating > 80;

1. Find all Samsung phones with ram > 8GB

SELECT \* FROM campusx.smartphones

WHERE brand\_name = 'samsung' AND ram\_capacity > 8;

1. Find all Samsung phones with snapdragon processor

SELECT \* FROM campusx.smartphones

WHERE brand\_name = 'samsung' AND processor\_brand = 'snapdragon';

* Query Execution Order
  + Find brands who sell phones with price > 50000

<https://infytq.onwingspan.com/web/en/viewer/web-module/lex_auth_01270711371270553642_shared?collectionId=lex_auth_012808459282808832527_shared&collectionType=Course&pathId=lex_auth_012758815799582720138_shared>

A close-up of a logo

Description automatically generated

"Frank John's Wicked Grave Haunts Several Dull Owls".

SELECT DISTINCT(brand\_name) FROM campusx.smartphones

WHERE price > 100000;

* IN and NOT IN

SELECT \* FROM campusx.smartphones

WHERE processor\_brand = 'snapdragon' OR

processor\_brand = 'eynos' OR

processor\_brand = 'bionic'

instead of using the above code where you need to add OR each time you want to add a new search. We can use IN and do the work.

SELECT \* FROM campusx.smartphones

WHERE processor\_brand IN ('snapdragon', 'exynos', 'bionic')

If we use NOT IN in place of IN then it will exclude all the values mentioned

SELECT \* FROM campusx.smartphones

WHERE processor\_brand NOT IN ('snapdragon', 'exynos', 'bionic')

* Update

What if we want to change the snapdragon to some other name. Suppose snapdragon 2.0

SELECT \* FROM campusx.smartphones

WHERE processor\_brand = 'snapdragon';

Example 1 -

UPDATE campusx.smartphone

SET proceccor\_brand = 'snapdragon 2.0'

WHERE processor\_brand = 'snapdragon';

Example 2-

UPDATE campusx.users

SET email = 'sajjaduddin@yahoo.com', password = '98765'

WHERE name = 'Sajjad'

* Delete

q) Delete all phones price > 200000

DELETE FROM campusx.smartphones

WHERE price > 200000;

DELETE FROM campusx.smartphones

WHERE primary\_camera\_rear > 150 AND brand\_name = 'xiaomi';

* Types of functions in SQL

A diagram of a company

Description automatically generated

* Aggregate Functions
* MAX / MIN
  + 1. Find the minimum and maximum price

SELECT MAX(price) FROM campusx.smartphones;

SELECT MIN(price) FROM campusx.smartphones;

SELECT MIN(ram\_capacity) FROM campusx.smartphones;

* + 1. Find the price of the costliest samsung phone

SELECT MAX(price) FROM campusx.smartphones

WHERE brand\_name = 'samsung'

Copy that values and make a new query

SELECT \* FROM campusx.smartphones

WHERE brand\_name = 'samsung' AND price = 110999

* AVG

\*\* Find avg rating of apple phones

SELECT AVG(rating) FROM campusx.smartphones

WHERE brand\_name = 'apple'

SELECT AVG(price) FROM campusx.smartphones

WHERE brand\_name = 'samsung'

* SUM

SELECT SUM(price) FROM campusx.smartphones

* COUNT

\*\* Find the number of oneplus phones

SELECT COUNT(\*) FROM campusx.smartphones

WHERE brand\_name = 'oneplus'

* COUNT(DISTINCT)

\*\* Find the number of brands available

SELECT COUNT(DISTINCT(brand\_name)) FROM campusx.smartphones

SELECT COUNT(DISTINCT(processor\_brand)) FROM campusx.smartphones

First we find the distinct then we count their total number.

* STD

\*\* Find std of screen sizes

SELECT STD(screen\_size) FROM campusx.smartphones;

* VAR

\*\* Find variance of Xiaomi phone prices

SELECT VARIANCE(screen\_size) FROM campusx.smartphones;

* Scalar Functions
* ABS

SELECT ABS(price - 100000) AS 'temp' FROM campusx.smartphones;

* ROUND

\*\* Round the ppi to 2 decimal place

SELECT model,

ROUND(SQRT(resolution\_width \* resolution\_width + resolution\_height \* resolution\_height)/screen\_size, 2) AS 'ppi value'

FROM campusx.smartphones;

* CEIL / FLOOR

\*\* floor/ceil the screen\_size

SELECT FLOOR(screen\_size) FROM campusx.smartphones;

SELECT CEIL(screen\_size) FROM campusx.smartphones;

-----------------------------------------------------------------------------------------------------------

Practice

\*\* Find the average battery capacity and the average primary rear camera resolution for all smartphones with a price greater than or equal to 100000

\*\* Find the average internal memory capacity of smartphones that have a refresh rate of 120 Hz or higher and a front-facing camera resolution greater than or equal to 20 megapixels.

\*\* Find the number of smartphones with 5G capabilities