QUERY EXECUTION ORDER

-- F -> J -> W -> G -> H -> S -> D -> optimize

-- FROM - JOIN- WHERE - GROUP BY - HAVING - SELECT - DISTINCT - ORDER BY

SQL DML COMMAND

```
-- CREATE DATABASE IF NOT EXISTS practice


-- Creating a table
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

```
INSERT INTO practice.users (user_id,name, email, password)
VALUES (NULL, 'sourov', 'sourov@gmail.com', '1234')


-- INSERT variation
INSERT INTO practice.users
VALUES (NULL, 'ankit', 'ankit@gmail.com', '41651')

INSERT INTO practice.users (password, name)
VALUES ('12433', 'akash')

-- INSERT multiple
INSERT INTO practice.users VALUES
(NULL, 'rishab', 'rishab@gmail.com', '43253'),
(NULL, 'rohan', 'rohan@gmail.com', '62543'),
(NULL, 'joy', 'joy@gmail.com', '835744')
```

SELECT

```
SELECT * FROM practice.users  -- (*) means all


-- SELECT all from smartphones dataset
SELECT * FROM practice.smartphones;




-- FILTER COLUMNS
SELECT model, price, rating FROM practice.smartphones


-- Filter -alias-> renaming columns
SELECT os AS 'operating_system', model,
battery_capacity AS 'battery_power'
FROM practice.smartphones


-- FITER
SELECT model, rating/10 FROM practice.smartphones
```

UNIQUE

- Find all unique brand name

```
SELECT DISTINCT(brand_name) AS 'all_brand'
FROM practice.smartphones
```

-Find all unique processor

```
SELECT DISTINCT(processor_brand) AS 'all_processor'
FROM practice.smartphones
```

-Find both brand_name and processor unique

```
SELECT DISTINCT brand_name, processor_brand
FROM practice.smartphones
```

WHERE CLAUSE

```
-- Find all samsung phones
SELECT * FROM practice.smartphones
WHERE brand_name = 'samsung'
```

```
-- find all phones with 100000<price<200000
SELECT * FROM practice.smartphones
WHERE price > 100000 AND price<200000

-- Find phone with rating>80 and price<25000
SELECT * FROM practice.smartphones
WHERE rating > 80 AND price <25000

-- Find brands who sell phones with price> 100000
SELECT DISTINCT(brand_name) FROM
practice.smartphones
WHERE price > 100000
```

BETWEEN

```
-- Find all phones with price 50000 to 70000
SELECT * FROM practice.smartphones
WHERE price BETWEEN 50000 AND 70000
```

OR

```
-- Find phones with processor brand
(snapdragon/sxynos/bionic)
SELECT * FROM practice.smartphones
WHERE processor_brand = 'snapdragon' OR
processor_brand = 'exynos' OR
processor_brand = 'bionic'
```

IN

```
SELECT * FROM practice.smartphones
WHERE processor_brand  IN
('snapdragon','exynos','bionic')
```

NOT IN

```
SELECT * FROM practice.smartphones
WHERE processor_brand  NOT IN ('dimencity', 'helio')
```

UPDATE

```
-- Change the processor name 'mediatek' by the name 'dimencity'
UPDATE practice.smartphones
SET processor_brand = 'dimencity'
WHERE processor_brand = 'mediatech'

UPDATE practice.users
SET email = 'Tanisha@gmail.com'
WHERE email = 'sourov@gmail.com'
```

DELETE

```
-- Delete smartphones whose price>200000
DELETE FROM  practice.smartphones
WHERE price > 200000
```

-- Delete samsung smartphones which primary rear camera > 150

```
DELETE FROM practice.smartphones
WHERE primary_camera_rear > 150 AND brand_name = 'samsung'
```

FUNCTIONS

Types of functions in sql

1. Builtin :(scaller, aggregate)
2. user define

-- Find the maximum price of smartphones

```
SELECT  MAX(price)
FROM practice.smartphones
```

-- Find minimum ram capacity of smartphones

```
SELECT  MIN(ram_capcity)
FROM practice.smartphones
```

```
-- Find average rating of apple smartphones
SELECT AVG(rating) FROM practice.smartphones
WHERE brand_name = 'apple'

-- Find how much Oneplus phone in this smartphone dataset
SELECT COUNT(*) FROM practice.smartphones
WHERE brand_name = 'oneplus'

-- Find how much processor brand in this smartphones dataset
SELECT COUNT(DISTINCT(processor_brand)) FROM practice.smartphones

SELECT price - 10000 AS 'temp' FROM practice.smartphones
```

-- CEIL -> highest possible without float like (5.7 -> 6)

```
SELECT CEIL(screen_size) FROM practice.smartphones
```

-- FLOOR -> lowest possible without float like(5.7 -> 5)

```
SELECT FLOOR(screen_size) FROM practice.smartphones
```

PRACTICE QUESTIONS

-- Find the average battery capacity and the average primary rear camera resolution for

-- all smartphones with a price greater than or equals to 100000

-- Find the average internal memory capacity of smartphones that have a refresh rate of

-- 120 HZ or hignt and a front_facing camera resolution greater than or equals to 20 megapixels

-- Find the number of smartphones with 5g capability

```
# Suppose you have a table 'movies' in A database :  Now someone ask you show only those movies name which contain 5 letter and starts with A .How will you find

# Ans : Use of Wild cords
SELECT * FROM movies WHERE name LIKE 'A_____'


# Find the name of movies which contain 'man' word as last

# '%man' <- man exist in last
# '%man%' -< man exist in any place
SELECT * FROM movies WHERE name LIKE '%man'



# Find the family size from titanic data and make a new column named family_type based on that family size
SELECT Name, (SibSp + Parch) AS 'size',
CASE
   WHEN (SibSp + Parch) <=2 THEN 'Small'
   WHEN (SibSp + Parch) >2 AND (SibSp + Parch)<=5 THEN 'Medium'
   ELSE 'large'
END AS 'type'
FROM interview.train

# Find Category wise Top values
# find top movie for each genre
SELECT * FROM movies
WHERE (genre, score) IN (SELECT genre, MAX(score)
                         FROM movies
                         GROUP BY genre)
```