```sql
/*
-------------- OPERATORS --------------

=> While using "WHERE", we use a lot of operators

1. Arithmetic Operators: +, -, *, /, %
2. Comparison Operators: =, !=, >, >=, <, <=
3. Logical Operators: AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY
4. Bitwise Operators: &(bitwise AND), | (bitwise OR)
*/

SELECT * FROM student WHERE (marks > 80+10 AND grade = "A"); -- showing 2 result
SELECT * FROM student WHERE (marks % 2 = 0 AND grade = "A"); -- showing 1 result
SELECT * FROM student WHERE (marks % 2 = 0 OR grade = "A"); -- showing 5 result
SELECT * FROM student WHERE marks BETWEEN 80 AND 90; -- showing 2 result (it includes 80 and 90)
SELECT * FROM student WHERE city IN ("Delhi", "Kolkata", "Mumbai", "Hyderabad"); -- showing 5 result
SELECT * FROM student WHERE city NOT IN ("Delhi", "Kolkata", "Mumbai", "Hyderabad"); -- showing 1 result


/*
---------- LIMIT Clause -------------
=> Sets an upper limit on number of (tuples) rows to be returned
SYNTAX:
    SELECT col1, col2 FROM table_name
    LIMIT number;
*/

SELECT * FROM student LIMIT 2; -- shows 2 rows
SELECT * FROM student WHERE marks > 75 LIMIT 10; -- shows 5 rows


/*
---------- ORDER BY Clause -------------
=> To sort results in ASC(ascending) or DESC (descending) order
SYNTAX:
    SELECT col1, col2 FROM table_name
    ORDER BY col_name(s) ASC;
*/

SELECT name, marks FROM student
ORDER BY city ASC; -- gives result in ascending order of city (delhi then mumbai then pune)

-- Lets find out top 3 students
SELECT * FROM student
ORDER BY marks DESC LIMIT 3; -- dhruv, bhumika, chetan


/*
------------------- Aggregate Functions ------------------------
-> Aggregate functions performs a calculation on a set of values, and return a single value

- COUNT()
- MAX()
- MIN()
- SUM()
- AVG()

=> For example: SELECT max(marks) FROM student;
*/

SELECT MAX(marks) FROM student; -- shows 96
SELECT MIN(marks) FROM student; -- shows 12
SELECT AVG(marks) FROM student; -- shows 74.3333
SELECT COUNT(rollNo) FROM student; -- shows 6
SELECT SUM(marks) FROM student; -- shows 446


/*
------------------ Group By Clause -------------------
-> Groups rows that have the same values into summary rows.
-> It collects data from multiple records and groups the result by one or more column.

-> Generally we use group by with some aggregation function.

For ex: Count number of students in each city
*/

SELECT city, COUNT(rollNo)
FROM student
GROUP BY city;
/*
-> Group the data based on cities
Pune    1
Mumbai  2
Delhi   3
*/

SELECT city, name, COUNT(rollNo)
FROM student
GROUP BY city, name;
/*
-> Group the data based on cities and name
Pune        anil        1
Mumbai      bhumika     1
Mumbai      chetan      1
Delhi       dhruv       1
Delhi       emanuel     1
Delhi       farah       1
*/

-- Lets get the average marks based on city
SELECT city, AVG(marks)
FROM student
GROUP BY city;
/*
Pune    78.0000
Mumbai  89.0000
Delhi   63.3333
*/

/*
---------------------------------------------------------------------
Practice Qs:
Qs: Write the query to find avg marks in each city in ascending order
*/

SELECT city, AVG(marks)
FROM student
GROUP BY city
ORDER BY city ASC;
/*
Delhi   63.3333
Mumbai  89.0000
Pune    78.0000
*/

SELECT city, AVG(marks)
FROM student
GROUP BY city
ORDER BY AVG(marks) ASC;
/*
Delhi   63.3333
Pune    78.0000
Mumbai  89.0000
*/
----------------------------------------------------------


/*
-----------------------------------------------------------
Practice Qs:
-> For a given table, find the total payment according to each payment method.

---------------------------------------------------------------------------------
customer id    -         customer      -      mode        -     city         -
---------------------------------------------------------------------------------
101            -     Olivia Barrett   -  Netbanking     -     Portland     -
102            -     Ethan Sinclair   -  Credit Card    -     Miami        -
103            -     Maya Hernandez   -  Credit Card    -     Seattle      -
104            -     Liam Donovan     -  Netbanking     -     Denver       -
105            -     Sophia Nguyen    -  Credit Card    -     New Orleans  -
106            -     Caleb Foster     -  Debit Card     -     Minneapolis  -
107            -     Ava Patel        -  Debit Card     -     Phoenix      -
108            -     Lucas Carter     -  Netbanking     -     Boston       -
109            -     Isabella Martinez -  Netbanking    -     Nashville    -
110            -     Jackson Brooks   -  Credit Card    -     Boston       -
---------------------------------------------------------------------------------
*/

-- Create the table
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(100),
    payment_mode VARCHAR(20),
    city VARCHAR(50)
);

-- Insert the data
INSERT INTO customers (customer_id, customer_name, payment_mode, city) VALUES
(101, 'Olivia Barrett', 'Netbanking', 'Portland'),
(102, 'Ethan Sinclair', 'Credit Card', 'Miami'),
(103, 'Maya Hernandez', 'Credit Card', 'Seattle'),
(104, 'Liam Donovan', 'Netbanking', 'Denver'),
(105, 'Sophia Nguyen', 'Credit Card', 'New Orleans'),
(106, 'Caleb Foster', 'Debit Card', 'Minneapolis'),
(107, 'Ava Patel', 'Debit Card', 'Phoenix'),
(108, 'Lucas Carter', 'Netbanking', 'Boston'),
(109, 'Isabella Martinez', 'Netbanking', 'Nashville'),
(110, 'Jackson Brooks', 'Credit Card', 'Boston');

SELECT payment_mode, COUNT(customer_name)
FROM customers
GROUP BY payment_mode;

/*
Result:
-------
Netbanking      4
Credit Card     4
Debit Card      2
*/

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

-- Q: Lets take the above student table and write a query to group students
-- according to their grade.

SELECT grade, COUNT(rollNo)
FROM student
GROUP BY grade
ORDER BY grade ASC;
/*
Result:
------
A   2
B   2
C   1
F   1
*/

/*
--------------- Having Clause ---------------
                -------------
-> Simialr to Where i.e. appliess some condition on rows.
-> Used when we want to apply any condition after grouping.

-> For ex: Count number of students in each city where max marks cross 90/
*/

SELECT city, COUNT(rollNo)
FROM student
GROUP BY city;
/*
Result:
------
Pune    1
Mumbai  2
Delhi   3
*/

-- Now, after applying having clause on this query,
SELECT city, COUNT(rollNo)
FROM student
GROUP BY city
HAVING MAX(marks) > 90;
/*
Result:
------
Mumbai  2
Delhi   3
*/

/*
--------------- General Order --------------
                ------------
SELECT column(s)
FROM table_name
WHERE condition
GROUP BY condition
HAVING condition
ORDER BY column(s) ASC;
*/

SELECT city
FROM student
WHERE grade = "A"
GROUP BY city
HAVING MAX(marks) > 90
ORDER BY city DESC;
/*
Result:
------
Mumbai
Delhi
*/

/*
---------- Table Related Queries ------------

Update (to update existing rows)
--------------------------------
SYNTAX:-
------
    UPDATE table_name
    SET col1 = val1, col2 = val2
    WHERE condition;
*/

-- Sometime SQL strict us to update any table. So in that case we have to run one command
SET SQL_SAFE_UPDATES = 0; -- mostly this problem visible in SQL Workbench

UPDATE student
SET grade = "O"
WHERE grade = "A";

SELECT * FROM student; -- here the grade "A" changed to grade "O".

-- Update student emanuel marks to 82 and grade A

UPDATE student
SET marks = 82, grade = "B"
WHERE rollNo = 105;


/*
Lets say we have to increment each student marks by 1
*/

UPDATE student
SET marks = marks + 1; -- this will increment each student marks by 1

/*
----------------- Delete ---------------
                  ------
-> To delete existing rows
SYNTAX:-
    DELETE FROM table_name
    WHERE condition;
*/

UPDATE student
SET marks = 12
WHERE rollNo = 105; -- this will again set emanuel marks to 12

-- Now we can apply a delete query
-- Delete the student whose marks < 33

DELETE FROM student
WHERE marks < 33; -- this has deleted emanuel's row
```