Tools

* MySQL engine
* PHPMyAdmin
* Install XAMPP or WAMP to use them
* SQL is case insensitive
* Write SQL commands in upper case anyways

SHOW

* SHOW - displays information contained in database and its tables
* SHOW DATABASES; - lists databases managed by server
* SHOW TABLES; - shows all tables in database
* SHOW COLUMNS FROM customers; - shows info about columns in a customers table

SELECT

* SELECT - select data from database
* Result is stored in result table; - result set
* SELECT column\_list FROM table\_name; - Selecting column\_list column from table\_name table
* You can select multiple columns at once (separated by commas)
* SELECT FirstName, LastName, City

FROM customers;

* Use \* to select all columns
* SELECT \* FROM customers;

DISTINCT

* DISTINCT is used with SELECT to select only unique records (no duplicates)
* SELECT DISTINCT City FROM customers;
* Selecting cities (no duplicates) with even id numbers is as follows
* SELECT DISTINCT city

FROM station

WHERE id%2 = 0;

COUNT

* COUNT gives number of rows
* COUNT with DISTINCT gives number of rows without duplicates
* SELECT COUNT(city)- COUNT(DISTINCT city)

FROM station;

* HackerRank: Revisiting Aggregations - Count function

SELECT COUNT(DISTINCT id)

FROM city WHERE population > 100000;

RIGHT

* Below code consider the last 3 strings
* SELECT name

FROM students

WHERE marks > 75

ORDER BY RIGHT(name,3),id;

LIMIT and OFFSET

* To fetch only subset of records, use LIMIT with SELECT
* Ex: To fetch first 5 records from customers table
* SELECT ID, FirstName, LastName, City

FROM customers LIMIT 5;

* Use OFFSET with LIMIT to mention starting point
* Ex: To fetch 4 records starting from 3rd position

SELECT ID, FirstName, LastName, City

FROM customers OFFSET 3 LIMIT 4;

* Since SQL start reading from index 0, this query starts with 4th record (id number 4)
* Ex: Offset of first row is 0
* Similar result can be achieved like this
* SELECT ID, FirstName, LastName, City

FROM customers LIMIT 3, 4;

Fully qualified names

* SELECT customers.City

FROM customers;

* This is useful when working with multiple tables that share same column names

ORDER BY

* Used with SELECT to sort data
* SELECT \* FROM customers

ORDER BY FirstName;

* Use secondary sorting using commas
* SELECT \* FROM customers

ORDER BY LastName, Age;

* Sorting is done in ascending order for numbers and alphabetically for words
* Correct syntax to combine ORDER BY with LIMIT
* SELECT \*

FROM cakes

ORDER BY calories LIMIT 3;

WHERE

* WHERE is used to extract data that fits certain criteria
* SELECT column

FROM table

WHERE condition;

* SELECT \* FROM customers

WHERE ID = 7;

* Comparison operators like =, !=, >, <, >=, <= and BETWEEN (Between inclusive range) are used
* SELECT \* FROM customers

WHERE ID BETWEEN 3 AND 7;

* Output: Data including id 3 and id 7
* SELECT firstname

FROM staff

WHERE salary BETWEEN 1500 AND 1900;

* For text values use single quotes ‘’
* SELECT ID, FirstName, LastName, City

FROM customers

WHERE City = 'New York';

* Use single quote to skip texts that include single quote
* Ex: ‘Can’’t’

Logical operators

* AND, OR, IN, NOT
* Use with WHERE to combine multiple conditions
* SELECT ID, FirstName, LastName, Age

FROM customers

WHERE Age >= 30 AND Age <= 40;

* Also, can use below query to get the same result
* SELECT ID, FirstName, LastName, Age

FROM customers

WHERE Age BETWEEN 30 AND 40;

* SELECT name

FROM films

WHERE year >= 2010 AND production = 'Marvel Studios'

ORDER BY name;

* SELECT \* FROM customers

WHERE City = 'New York' OR City = 'Chicago';

* Use parentheses when combining AND and OR
* SELECT \* FROM customers

WHERE City = 'New York'

AND (Age=30 OR Age=35);

* IN is used to compare a column with more than one value
* SELECT \*

FROM users

WHERE state = ‘NY’ OR state = ‘CA’;

* Same result can be achieved with IN operator
* SELECT \* FROM users

WHERE state IN ('NY', 'CA');

* SELECT teamname, country

FROM teams

WHERE country IN ('Spain', 'England', 'Germany');

* HackerRank:

SELECT MAX(salary\*months), COUNT(employee\_id)

FROM employee

WHERE salary\*months IN (SELECT(MAX(salary\*months)) FROM employee);

* NOT IN is used to exclude from a list
* SELECT \*

FROM customers

WHERE City NOT IN (‘New York’, ‘Los Angeles’, ‘Chicago’);

CONCAT function

* Used to concatenate 2 or more texts
* SELECT CONCAT(FirstName, ' ' , LastName)

FROM customers;

* Use AS to name the resulting concatenated column
* SELECT CONCAT(FirstName,', ', City) AS new\_column

FROM customers;

* The default column name is concat

LENGTH

* SELECT LENGTH(‘car’);
* Outputs length of a text
* Weather observation station 5 (HackerRank)

SELECT city, LENGTH(city) AS size

FROM station

ORDER BY size, city

LIMIT 1;

SELECT city, LENGTH(city) AS size

FROM station

ORDER BY size DESC, city

LIMIT 1;

Arithmetic operators

* +, -, \*, /
* Below code adds 500 to each salary
* SELECT ID, FirstName, LastName, Salary+500 AS Salary

FROM employees;

* SELECT CONCAT(firstname, ' ', lastname) AS fullname, (salary\*12 + 500\*experience) AS total

FROM staff

ORDER BY total;

FLOOR

* Rounds to nearest integer
* HackerRank: Average population

SELECT FLOOR(AVG(population))

FROM city;

CEIL

* Round to next integer

REPLACE

* Replaces character with something else
* HackerRank: The blunder

SELECT CEIL(AVG(salary) - AVG(REPLACE(salary, 0, '')))

FROM employees;

ROUND

* Rounds to specified decimal places
* HackerRank: Weather observation station 12

SELECT ROUND(SUM(lat\_n), 2), ROUND(SUM(long\_w), 2)

FROM station;

* HackerRank: Weather observation station 13

SELECT ROUND(SUM(lat\_n), 4)

FROM station

WHERE lat\_n > 38.7880 AND lat\_n < 137.2345;

* HackerRank: Weather observation station 14

SELECT ROUND(MAX(lat\_n), 4)

FROM station

WHERE lat\_n < 137.2345;

* HackerRank: Weather observation station 15

SELECT ROUND(long\_w, 4)

FROM station

WHERE lat\_n = (SELECT MAX(lat\_n) FROM station WHERE lat\_n < 137.2345);

UPPER and LOWER

* SELECT FirstName, UPPER(LastName) AS LastName

FROM employees;

SQRT

* SELECT Salary, SQRT(Salary)

FROM employees;

* Same result can be achieved using POWER with ½ exponent
* SQRT is faster than that method

AVG

* SELECT AVG(Salary)

FROM employees;

* SELECT AVG(score)

FROM sam\_grades

WHERE semester = 1;

* HackerRank: Average

SELECT AVG(population)

FROM city

WHERE district = 'california';

SUM

* SELECT SUM(Salary)

FROM employees;

* HackerRank: The sum function

SELECT SUM(population)

FROM city

WHERE district = 'california';

* HackerRank: Japan Population

SELECT SUM(population)

FROM city

WHERE countrycode = 'JPN';

Subqueries

* Query within a query
* SELECT AVG(Salary) FROM employees;
* AVG(Salary) is 3100
* SELECT FirstName, Salary

FROM employees

WHERE Salary > 3100

ORDER BY Salary DESC;

* DESC results in descending order
* ASC results in ascending order
* Single subquery can be used instead
* SELECT FirstName, Salary FROM employees

WHERE Salary > (SELECT AVG(Salary) FROM employees)

ORDER BY Salary DESC;

* Another example is as follows
* SELECT \*

FROM Foods

WHERE fatpercentage < (SELECT AVG(fatpercentage) FROM Foods);

* SELECT \*

FROM Apartments

WHERE (price > (SELECT AVG(price) FROM Apartments)) AND status = 'Not rented'

ORDER BY price;

LIKE

* Used within WHERE clause to specify search condition
* Use \_ to match single character and % to match arbitrary number of characters (including 0 characters)
* Following query is used to find FirstNames starting with A
* SELECT \*

FROM employees

WHERE FirstName LIKE 'A%';

* SELECT DISTINCT city

FROM station

WHERE (city LIKE 'A%') OR (city LIKE 'E%') OR (city LIKE 'I%') OR (city LIKE 'O%') OR (city LIKE 'U%');

* For name ending with s
* SELECT \*

FROM employees

WHERE LastName LIKE '%s';

* SELECT \*

FROM desserts

WHERE name LIKE '%Chocolate%';

* SELECT DISTINCT city

FROM station

WHERE (city LIKE '%a') OR (city LIKE '%e') OR (city LIKE '%i') OR (city LIKE '%o') OR (city LIKE '%u');

* SELECT DISTINCT city

FROM station

WHERE (city LIKE 'A%' OR city LIKE 'E%' OR city LIKE 'I%' OR city LIKE 'O%' OR city LIKE 'U%') AND

(city LIKE '%a' OR city LIKE '%e' OR city LIKE '%i' OR city LIKE '%o' OR city LIKE '%u');

NOT LIKE

* SELECT DISTINCT city

FROM station

WHERE (city NOT LIKE 'A%' AND city NOT LIKE 'E%' AND city NOT LIKE 'I%' AND city NOT LIKE 'O%' AND city NOT LIKE 'U%');

* SELECT DISTINCT city

FROM station

WHERE (city NOT LIKE '%a') AND (city NOT LIKE '%e') AND (city NOT LIKE '%i') AND (city NOT LIKE '%o') AND (city NOT LIKE '%u');

* SELECT DISTINCT city

FROM station

WHERE (city NOT LIKE 'A%' AND city NOT LIKE 'E%' AND city NOT LIKE 'I%' AND city NOT LIKE 'O%' AND city NOT LIKE 'U%') OR

(city NOT LIKE '%a' AND city NOT LIKE '%e' AND city NOT LIKE '%i' AND city NOT LIKE '%o' AND city NOT LIKE '%u');

MAX

* To find maximum value

MIN

* To find minimum value
* SELECT MIN(Salary) AS Salary

FROM employees;

* HackerRank: Population density difference

SELECT MAX(population) - MIN(population)

FROM city;

Joining tables

* Combining data from 2 or more tables (creates temporary table)
* SELECT customers.ID, customers.Name, orders.Name, orders.Amount

FROM customers, orders

WHERE customers.ID=orders.Customer\_ID

ORDER BY customers.ID;

* Fully qualified names are used to avoid confusion
* SELECT students.id, students.firstname, students.lastname, teachers.lastname AS teacher

FROM students, teachers

WHERE students.teacherid = teachers.id

ORDER BY students.id;

* Use nicknames for tables using AS to shorten code length
* SELECT ct.ID, ct.Name, ord.name, ord.Amount

FROM customers AS ct, orders AS ord

WHERE ct.ID = ord.Customers\_ID

ORDER BY ct.ID;

Types of JOIN in MySQL

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* Other join types in SQL aren’t supported by MySQL

INNER JOIN

* Equivalent to JOIN
* Returns rows when there’s a match between tables
* ON is used to specify inner join condition
* Syntax is as follows
* SELECT column(s)

FROM table1 INNER JOIN table2

ON table1.column = table2.column;

* SELECT productname, price, categoryname

FROM products INNER JOIN categories

ON products.categoryid = categories.id;

* INSERT INTO animals

VALUES

('Slim', 'Giraffe', 1);

SELECT animals.name, animals.type, countries.country

FROM animals INNER JOIN countries

ON animals.country\_id = countries.id

ORDER BY countries.country;

LEFT JOIN

* Returns all rows from left table (even with no matches with right table)
* Syntax is as follows
* SELECT table.column(s)

FROM table1 LEFT OUTER JOIN table2

ON table1.column = table2.column;

* OUTER keyword is optional
* SELECT customers.Name, items.Name

FROM customers LEFT OUTER JOIN items

ON customers.ID=items.Seller\_id;

* Results matching data from right table and all data from left table

RIGHT JOIN

* Returns all rows from right table (even with no matches with left table)
* SELECT customers.Name, items.Name

FROM customers RIGHT JOIN items

ON customers.ID=items.Seller\_id;

Set operation (UNION)

* Use UNION and UNION ALL
* UNION combines multiple datasets to a single dataset excluding duplicates
* UNION ALL does the same thing including duplicates
* UNION ALL is faster than UNION
* UNION combines result sets from 2 or more SELECT statements
* SELECT statements must have same number of columns with same data type with same order
* SELECT ID, FirstName, LastName, City

FROM First

UNION

SELECT ID, FirstName, LastName, City

FROM Second;

* If columns don’t match exactly across all queries, use NULL as follows
* SELECT FirstName, LastName, Company

FROM businessContacts

UNION  
SELECT FirstName, LastName, NULL

FROM otherContacts;

* SELECT \*

FROM NorwayChess

UNION

SELECT \*

FROM TataSteel

ORDER BY rating DESC;

* UNION ALL includes duplicates
* SELECT ID, FirstName, LastName, City

FROM First

UNION ALL

SELECT ID, FirstName, LastName, City

FROM Second;

INSERT INTO

* Used to add new rows of data to a table in the database
* Order of the values must be same order as the columns in the table
* INSERT INTO Employees

VALUES

(8, 'Anthony', 'Young', 35);

* INSERT INTO Garage

VALUES

(6, 'Mercedes-Benz', 'G 63', 2020),

(7, 'Porsche', 'Panamera', 2020 );

* Only required columns to be added can be mentioned in the query
* INSERT INTO Employees (ID, FirstName, LastName, Age)

VALUES (8, 'Anthony', 'Young', 35);

* INSERT INTO Employees (ID, FirstName, LastName)

VALUES (9, 'Samuel', 'Clark');

* Column order can be specified to user liking but inserting order should be the same
* Cells that are not added will result in default cell value or nothing

CASE - Advanced SELECT

* SELECT CASE

WHEN (A+B <= C OR B+C <= A OR C+A <= B) THEN 'Not A Triangle'

WHEN (A = B AND B = C) THEN 'Equilateral'

WHEN (A = B OR B = C OR A = C) THEN 'Isosceles'

ELSE 'Scalene'

END

FROM triangles;

UPDATE

* Allows to alter data in table
* UPDATE Employees

SET Salary=5000

WHERE ID=1;

* Update multiple columns as follows
* UPDATE Employees

SET Salary=5000, FirstName='Robert'

WHERE ID=1;

DELETE

* To remove data from table
* DELETE FROM Employees

WHERE ID=1;

SELECT \* FROM Employees;

CREAT TABLE

* Basic syntax is as follows
* CREATE TABLE table\_name

(

column\_name1 data\_type(size),

column\_name2 data\_type(size),

column\_name3 data\_type(size),

...

);

* Data types: int for whole numbers, char, varchar (variable-length character) for characters, blob, text, float(M,D): M for display length, D for number of decimals, double(M,D) for double precision floating numbers, date, datetime, timestamp, time
* Size: maximum length of column
* CREATE TABLE Users

(

UserID int,

FirstName varchar(100),

LastName varchar(100),

City varchar(100),

PRIMARY KEY(UserID)

);

* Use INSERT INTO too add data
* CREATE TABLE leaderboard

(

    place INT,

    nickname VARCHAR,

    rating INT

);

INSERT INTO leaderboard

VALUES

(1, 'Predator', 9500),

(2, 'JohnWar', 9300),

(3, 'NightWarrior', 8900);

SELECT \* FROM leaderboard;

SQL constraints

* NOT NULL: column can’t contain NULLs
* UNIQUE: no duplicates
* PRIMARY KEY
* CHECK: checking if the value is valid or not
* DEFAULT: default value
* Ex: name VARCHAR(100) NOT NULL

AUTO\_INCREMENT

* Generating unique number when new records are added
* Default starting value: 1
* UserID INT NOT NULL AUTO\_INCREMENT,

PRIMARY KEY(UserID)

* CREATE TBALE users (

id INT NOT NULL AUTO\_INCREMENT,

username VARCHAR(40) NOT NULL,

password VARCHAR(10) NOT NULL,

PRIMARY KEY(id)

);

ALTER TABLE

* Add, delete, modify columns in existing table
* Below code adds new column named DateOfBirth
* ALTER TABLE People

ADD DateOfBirth date;

* All rows will have the default value: NULL in this case
* ALTER TABLE cities

ADD AttractivePlace VARCHAR;

UPDATE cities

SET AttractivePlace='Belem Tower' WHERE name='Lisbon';

UPDATE cities

SET AttractivePlace='Plaza Mayor' WHERE name='Madrid';

UPDATE cities

SET AttractivePlace='Eiffel Tower' WHERE name='Paris';

DROP COLUMN

* Deleting column
* ALTER TABLE People

DROP COLUMN DateOfBirth;

* To delete entire table use DROP TABLE

RENAME

* Rename columns
* ALTER TABLE People

RENAME FirstName TO name;

* To rename entire table use RENAME TABLE
* RENAME TABLE people TO users;

CREATE VIEW

* Virtual table based on result set of SQL code
* Views are always updated (No need of manual update)
* Syntax is as follows
* CREATE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition;

* CREATE VIEW List AS

SELECT FirstName, Salary

FROM Employees;

* CREATE VIEW view1 AS

SELECT acc\_id, status

FROM users;

SELECT \* FROM view1;

REPLACE VIEW

* Update a view
* CREATE OR REPLACE VIEW List AS

SELECT FirstName, LastName, Salary

FROM  Employees;

DROP VIEW

* Delete a view
* DROP VIEW list;

HackerRank - The PADS

* SELECT CONCAT(name, '(', LEFT(occupation, 1), ')')

FROM occupations

ORDER BY name;

SELECT CONCAT('There are a total of ', COUNT(occupation), ' ', LOWER(occupation), 's.')

FROM occupations

GROUP BY occupation

ORDER BY COUNT(occupation), occupation;

PIVOT

* Syntax for PIVOT is as follows
* SELECT <non-pivoted column>,

[first pivoted column] AS <column name>,

[second pivoted column] AS <column name>,

...

[last pivoted column] AS <column name>

FROM

(<SELECT query that produces the data>)

AS <alias for the source query>

PIVOT

(

<aggregation function>(<column being aggregated>)

FOR

[<column that contains the values that will become column headers>]

IN ( [first pivoted column], [second pivoted column],

... [last pivoted column])

) AS <alias for the pivot table>

<optional ORDER BY clause>;

HackerRank - Occupations

* SELECT GROUP\_CONCAT(IF(occupation = 'Doctor', name, NULL)), GROUP\_CONCAT(IF(occupation = 'Professor', name, NULL)), GROUP\_CONCAT(IF(occupation = 'Singer', name, NULL)), GROUP\_CONCAT(IF(occupation = 'Actor', name, NULL))

FROM (SELECT \*, ROW\_NUMBER() OVER(PARTITION BY occupation ORDER BY name) AS custom FROM occupations) AS ord

GROUP BY custom;

HackerRank: Binary Tree Nodes

* SELECT n,

CASE WHEN p IS null THEN "Root"

WHEN N NOT IN (SELECT DISTINCT P FROM BST WHERE P IS NOT NULL) THEN "Leaf"

ELSE "Inner"

END AS NODE FROM BST

ORDER BY N;

HackerRank: New Companies

* SELECT

e.company\_code,

MIN(c.founder),

COUNT(DISTINCT e.lead\_manager\_code),

COUNT(DISTINCT e.senior\_manager\_code),

COUNT(DISTINCT e.manager\_code),

COUNT(DISTINCT e.employee\_code)

FROM employee AS e

INNER JOIN company AS c ON e.company\_code = c.company\_code

GROUP BY e.company\_code

ORDER BY e.company\_code;