* <https://www.educba.com/data-science/data-science-tutorials/sql-tutorial/>
* show databases; -> for showing all databases present;
* create database xyz;-> for creating database having name 'xyz'
* delete database xyz;-> for deleting database having name 'xyz'
* show tables from xyz; -> for showing all tables present in xyz database
* USE xyz-> for using database xyz;
* Datatypes ->

--INT - Whole Numbers

--DECIMAL(M,N) M total digits, N digits after decimal - Decimal Number - ExactValue

--VARCHAR(l) - String of text of length l

--BLOB - Binary Large Object, Stores large binary data

--DATE - YYYY-MM-DD

--TIMESTAMP - YYYY-MM-DD HH:MM:SS

* CREATE TABLE student(

attrbutename datatype constraint,

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

#OR -> PRIMARY KEY(student\_id)

name VARCHAR(20), NOT NULL, #20 is the number of characters

subject VARCHAR(20) UNIQUE DEFAULT 'anydataofmentionedtype'

);

* DROP TABLE student; ->deletes the table
* DESCRIBE xyz; ->describes table named xyz
* ALTER TABLE xyz ADD newattributename datatype constraint;
* ALTER TABLE xyz DROP attributename;
* SELECT \* FROM xyz-> shows everything from table xyz
* INSERT INTO xyz(attributename1,attributename2..)VALUES(data,data);
* INSERT INTO xyz VALUES(data,data,data..);
* **create,alter,insert/firstqueryfordatabaseone**

CREATE TABLE student(

student\_id INT PRIMARY KEY,

#OR -> PRIMARY KEY(student\_id)

name VARCHAR(20), #20 is the number of characters

subject VARCHAR(20)

);

DESCRIBE student;

#DROP TABLE student;

ALTER TABLE student ADD gradepoint DECIMAL(4,2);

ALTER TABLE student DROP gradepoint;

DESCRIBE student;

INSERT INTO student VALUES(1,'JACK','BIOLOGY');

INSERT INTO student VALUES(2,'KATE','SOCIOLOGY');

INSERT INTO student (student\_id,name)VALUES(3,'CLAIRE');

INSERT INTO student VALUES(4,'JACK','BIOLOGY');

INSERT INTO student VALUES(5,'MIKE','COMPUTER SCIENCE');

SELECT \* FROM student;

* **constraints-notnull,unique/secondqueryfordatabasetwo**

CREATE TABLE student2(

student\_id INT PRIMARY KEY, #primary key is basically not null and unique

#OR -> PRIMARY KEY(student\_id)

name VARCHAR(20) NOT NULL, #20 is the number of characters

subject VARCHAR(20) UNIQUE

);

DESCRIBE student2;

SELECT \* FROM student2;

INSERT INTO student2 VALUES(1,'JACK','BIOLOGY');

INSERT INTO student2 VALUES(2,'KATE','SOCIOLOGY');

INSERT INTO student2 (student\_id,name)VALUES(3,'CLAIRE');

INSERT INTO student2 VALUES(4,'JACK','zZOOLOGY');

INSERT INTO student2 VALUES(5,'MIKE','COMPUTER SCIENCE');

* **constraints-default/thirdqueryfordatabaseone**

CREATE TABLE student3(

student\_id INT PRIMARY KEY, #primary key is basically not null and unique

#OR -> PRIMARY KEY(student\_id)

name VARCHAR(20) NOT NULL, #20 is the number of characters

subject VARCHAR(20) UNIQUE DEFAULT 'undecided'

);

DESCRIBE student3;

INSERT INTO student3 VALUES(1,'JACK','BIOLOGY');

INSERT INTO student3 VALUES(2,'KATE','SOCIOLOGY');

INSERT INTO student3 (student\_id,name)VALUES(3,'CLAIRE');

INSERT INTO student3 VALUES(4,'JACK','zZOOLOGY');

INSERT INTO student3 VALUES(5,'MIKE','COMPUTER SCIENCE');

INSERT INTO student3 VALUES(6,'SPARROW',NULL);

SELECT \* FROM student3;

* **constraint-autoincrement/fourthqueryfordatabaseone**

CREATE TABLE student4(

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(20) NOT NULL, #20 is the number of characters

subject VARCHAR(20) UNIQUE DEFAULT 'undecided'

);

INSERT INTO student4 VALUES(1,'JACK','BIOLOGY');

INSERT INTO student4 VALUES(2,'KATE','SOCIOLOGY');

INSERT INTO student4 (name,subject)VALUES('CLAIRE','CHEMISTRY');

INSERT INTO student4(name) VALUES('JACK');

SELECT \* FROM student4;

* **update/delete-fifthqueryfordatabaseone**

SELECT \* FROM student4;

UPDATE student

SET subject='BIO' WHERE subject='BIOLOGY';

UPDATE student4

SET subject='COMPUTER SCIENCE' WHERE student\_id=4;

UPDATE student4

SET subject='BIO\_CHEMISTRY'

WHERE subject='BIOLOGY';

UPDATE student4

SET name='Tom',subject='undecided'

WHERE student\_id=1;

UPDATE student4

SET name='undecided';

DELETE FROM student

WHERE student\_id=3;

DELETE FROM student4

WHERE name='undecided' AND subject='COMPUTER SCIENCE';

DELETE FROM student4;

* **Select,From,Where,Order By, IN/Basic queriesfordatabaseone**

#SELECT, ORDERBY, ASCENDING, DESCENDING, LIMIT, WHERE

SELECT \*

FROM student;

SELECT name,subject

FROM student;

SELECT student.name,student.subject

FROM student;

SELECT student.name,student.subject

FROM student

ORDER BY name DESC;

SELECT \*

FROM student

ORDER BY name DESC,student\_id DESC

LIMIT 2;

SELECT \* FROM student

WHERE subject='BIO'

OR subject='SOCIOLOGY';

SELECT \* FROM student

WHERE subject='BIO' OR name='KATE';

--<,>,<=,>=,<>,AND,OR

SELECT \*

FROM student

WHERE subject<>'COMPUTER SCIENCE';

SELECT \*

FROM student

WHERE student\_id<=4;

SELECT \*

FROM student

WHERE student\_id<4;

SELECT \*

FROM student

WHERE student\_id<4 AND name<>'JACK';

SELECT \*

FROM student

WHERE name

IN('MIKE','KATE') AND student\_id>2;

* **CreatingCompanyDatabase**

CREATE TABLE employee (

emp\_id INT PRIMARY KEY,

first\_name VARCHAR(40),

last\_name VARCHAR(40),

birth\_date DATE,

sex VARCHAR(1),

salary INT,

super\_id INT, -- FOREIGN KEY

branch\_id INT -- FOREIGN KEY

-- add constraints later on as both branch table and employee table doesnot exist yet--

);

DESCRIBE employee;

show tables from company;

CREATE TABLE branch (

branch\_id INT PRIMARY KEY,

branch\_name VARCHAR(40),

mgr\_id INT,

mgr\_start\_date DATE,

FOREIGN KEY(mgr\_id) REFERENCES employee(emp\_id) ON DELETE SET NULL

);

ALTER TABLE employee

ADD FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE SET NULL;

ALTER TABLE employee

ADD FOREIGN KEY(super\_id) REFERENCES employee(emp\_id) ON DELETE SET NULL;

CREATE TABLE client(

client\_id INT,

client\_name VARCHAR(40),

branch\_id INT,

PRIMARY KEY(client\_id),

FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE SET NULL

);

CREATE TABLE works\_with(

emp\_id INT,

client\_id INT,

total\_sales INT,

PRIMARY KEY(emp\_id,client\_id),

FOREIGN KEY(emp\_id) REFERENCES employee(emp\_id) ON DELETE CASCADE,

FOREIGN KEY(client\_id) REFERENCES client(client\_id) ON DELETE CASCADE

);

CREATE TABLE branch\_supplier(

branch\_id INT,

supplier\_name VARCHAR(40),

supply\_type VARCHAR(40),

PRIMARY KEY(branch\_id,supplier\_name),

FOREIGN KEY(branch\_id) REFERENCES branch(branch\_id) ON DELETE CASCADE

);

* **inserting values in tables of db employees**

INSERT INTO employee VALUES(100,'DAVID','WALLACE','1967-11-17','M',25000,NULL,NULL);

INSERT INTO branch VALUES(1,'CORPORATE',100,'2006-02-09');

UPDATE employee

SET branch\_id=1

WHERE emp\_id=100;

SELECT\* FROM employee;

SELECT\* FROM branch;

INSERT INTO employee VALUES(101,'JAN','LEVINSON','1961-5-11','F',11000,100,1);

INSERT INTO branch VALUES(2,'SCRANTON',NULL,'1992-04-06');

INSERT INTO employee VALUES(102,'MICHAEL', 'SCOTT','1964-3-15','M',75000,100,2);

UPDATE branch

SET mgr\_id=102

WHERE branch\_id=2;

INSERT INTO employee VALUES(103,'ANGELA', 'MARTIN','1971-6-25','F',63000,102,2);

INSERT INTO employee VALUES(104,'KELLY', 'KAPOOR','1980-2-5','F',55000,102,2);

INSERT INTO employee VALUES(105,'STANLEY', 'HUDSON','1958-2-19','M',69000,102,2);

INSERT INTO employee VALUES(106,'JOSH', 'PORTER','1969-12-9','M',78000,100,NULL);

INSERT INTO employee VALUES(107,'ANDY', 'BERNARD','1973-7-9','M',65000,106,3);

INSERT INTO employee VALUES(108,'JIM', 'HALPERT','1978-10-9','M',71000,106,3);

INSERT INTO branch VALUES(3,'STAMFORD',106,'1998-02-13');

UPDATE employee

SET branch\_id=3

WHERE emp\_id=106;

INSERT INTO client VALUES(400,'DUNMORE HIGHSCHOOL',2);

INSERT INTO client VALUES(401,'LACKAWANA COUNTRY',2);

INSERT INTO client VALUES(402,'FEDEX',3);

INSERT INTO client VALUES(403,'JOHN DALY LAW',3);

INSERT INTO client VALUES(404,'SCRANTON WHITEPAGE',2);

INSERT INTO client VALUES(405,'TIMES NEWSPAPER',3);

INSERT INTO client VALUES(406,'FEDEX',2);

SELECT \* FROM client;

SELECT \* FROM Works\_With;

INSERT INTO Works\_With VALUES(105,400,55000);

INSERT INTO Works\_With VALUES(102,401,267000);

INSERT INTO Works\_With VALUES(108,402,22000);

INSERT INTO Works\_With VALUES(107,403,5000);

INSERT INTO Works\_With VALUES(108,403,12000);

INSERT INTO Works\_With VALUES(105,404,33000);

INSERT INTO Works\_With VALUES(107,405,26000);

INSERT INTO Works\_With VALUES(102,406,15000);

INSERT INTO Works\_With VALUES(105,406,130000);

SELECT \* FROM branch\_supplier;

INSERT INTO branch\_supplier VALUES(2,'HAMMER MALL','PAPER');

INSERT INTO branch\_supplier VALUES(2,'UNI-BALL','PAPER');

INSERT INTO branch\_supplier VALUES(3,'PATRIOT PAPER','WRITING UENSILS');

INSERT INTO branch\_supplier VALUES(2,'J.T. FORMS & LABELS','PAPER');

INSERT INTO branch\_supplier VALUES(3,'UNI-BALL','CUSTOM FORMS');

INSERT INTO branch\_supplier VALUES(3,'HAMMER MALL','PAPER');

INSERT INTO branch\_supplier VALUES(3,'STANFORD LABIES','CUSTOM FORMS');

UPDATE branch\_supplier

SET supply\_type='PAPER'

WHERE supplier\_name='PATRIOT PAPER';

UPDATE branch\_supplier

SET supply\_type='CUSTOM FORMS'

WHERE supplier\_name='J.T. FORMS & LABELS';

* **basic queries using select**

--FIND ALL EMPLOYEES

SELECT \*

FROM employee

ORDER BY salary;

-- FIND ALL EMPLOYEES IN DESCENDING

SELECT \*

FROM employee

ORDER BY salary DESC;

-- FIND ALL EMPLOYEES ORDER BY SEX THEN NAME

SELECT \*

FROM employee

ORDER BY sex, first\_name, last\_name;

-- FIND FIRST 5 EMPLOYEES

SELECT \*

FROM employee

LIMIT 5;

--FIND FIRST AND LAST NAME OF ALL EMPLOYEES

SELECT first\_name, last\_name

FROM employee

LIMIT 5;

--FIND FORENAME AND SURNAME OF ALL EMPLOYEES

SELECT first\_name AS forename, last\_name AS surname

FROM employee;

--find all gender

SELECT DISTINCT sex

FROM employee;

* **Nested queries**

--find names of all employees who have

--sold over 30000 to a single client

SELECT works\_with.emp\_id, employee.emp\_id, employee.first\_name, works\_with.total\_sales

FROM employee JOIN works\_with

ON employee.emp\_id=works\_with.emp\_id

WHERE works\_with.total\_sales>30000;

SELECT employee.emp\_id, employee.first\_name, employee.last\_name

FROM employee

WHERE employee.emp\_id IN(

SELECT works\_with.emp\_id

FROM works\_with

WHERE works\_with.total\_sales>30000

);

SELECT client\_name FROM client WHERE client.branch\_id IN(

SELECT branch\_id from branch where mgr\_id IN

(SELECT emp\_id from employee where first\_name='MICHAEL');

);

SELECT client.client\_name FROM client WHERE client.branch\_id = (

SELECT employee.branch\_id

FROM employee JOIN branch

ON employee.emp\_id=branch.mgr\_id

WHERE employee.first\_name='MICHAEL'

);

* **Functions**

--Find the number of employees

SELECT COUNT(emp\_id)

FROM employee;

SELECT COUNT(super\_id)

FROM employee;

--NUMBER OF FEMALE EMPLOYEES BORN AFTER 1970

SELECT COUNT(emp\_id)

FROM employee

WHERE sex ='F' AND

birth\_date > '1970-01-01';

--FIND AVG SALARY OF ALL EMPLOYEES

SELECT AVG(salary)

FROM employee;

--FIND AVG SALARY OF ALL FEMALE EMPLOYEES

SELECT AVG(salary)

FROM employee

WHERE sex='F' ;

--FIND SUM OF ALL EMPLOYEE

SELECT SUM(salary)

FROM employee

WHERE emp\_id<104;

--GROUP BY CLAUSE

SELECT COUNT(emp\_id) AS EMP\_COUNT,sex

FROM employee

GROUP BY sex

ORDER BY EMP\_COUNT DESC;

--FIND TOTAL SALES OF EACH SALESMAN

SELECT SUM(total\_sales),emp\_id

FROM works\_with

GROUP BY emp\_id;

* **Wildcards**

-- % = any no. of characters, \_ = one character

--FIND any clients who are an LLC

SELECT \*

FROM client

WHERE client\_name LIKE'%LLC';

--FIND any branch supplier who are in label business

SELECT \*

FROM branch\_supplier

WHERE supplier\_name LIKE'%labels';

select \* from branch\_supplier;

--FIND EMPLOYEE BORN IN OCTOBER

SELECT \*

FROM employee

WHERE birth\_date LIKE '%10%';

SELECT \*

FROM employee

WHERE birth\_date LIKE '\_\_\_\_-10-\_\_';

SELECT \*

FROM employee

WHERE birth\_date LIKE '\_\_\_\_-02%';

* **Union**

--combining results of multiple select statement into one

SELECT first\_name AS namee

FROM employee

UNION

SELECT branch\_name

FROM branch

UNION

SELECT last\_name

FROM employee;

--list of all clients and branch supplier names

SELECT client\_name, client.branch\_id

FROM client

UNION

SELECT supplier\_name, branch\_supplier.branch\_id

FROM branch\_supplier;

* **Joins**

--combine rows from 2 or more tables into a single based on a related column between them

--4 types of join

-- find all branches and name of their managers

--inner join

SELECT employee.emp\_id,employee.first\_name, branch.branch\_name

FROM employee JOIN branch

ON employee.emp\_id=branch.mgr\_id; --emp\_id and mgr\_id are related column here

--left join

SELECT employee.emp\_id,employee.first\_name, branch.branch\_name

FROM employee LEFT JOIN branch

ON employee.emp\_id=branch.mgr\_id;

--right join

SELECT employee.emp\_id,employee.first\_name, branch.branch\_name

FROM employee RIGHT JOIN branch

ON employee.emp\_id=branch.mgr\_id;

--full outer join

--both right and left join , not possible in mysql

* **on delete with foreign keys associated**

--on delete set null - on delete of the primary set the

--value foreign key dependent on it to null

select \* from employee;

select \* from branch\_supplier;

delete from employee

where emp\_id=102;

--on delete cascade deletes the entire row in which the foreign refering to the deleted primary key is present, basically it is used when foreign key is also primary key

delete from branch

where branch\_id = 2;

* **Finding Nth largest value**

SELECT emp\_id FROM employee AS e

where 2 = (

select COUNT(DISTINCT emp\_id)

FROM employee as p

WHERE e.emp\_id<=p.emp\_id

);