DROP TABLE student;

CREATE TABLE student (

adm\_no INT AUTO\_INCREMENT, -- Did not used this in creating our table

f\_name VARCHAR (20),

l\_name VARCHAR(20),

major VARCHAR (20) DEFAULT 'Undecided',

gpa DECIMAL (3,2) ,

PRIMARY KEY (adm\_no));

DESCRIBE student;

select \* from student;

ALTER TABLE student ADD COLUMN gpa DECIMAL(3,2);

ALTER TABLE student DROP COLUMN gpa;

INSERT INTO student VALUES (1, 'Protus', 'Langat', 'Economics', 3.89);

INSERT INTO student VALUES (2, 'Brian', 'Kiiza', 'Computer Science', 3.00);

INSERT INTO student VALUES (3, 'Edward', 'Xiung', 'Chemistry', 3.45);

INSERT INTO student VALUES (4, 'Linda', 'Msibi', 'Education', 3.34);

INSERT INTO student VALUES (5, 'John', 'Rowlings', 'Biology', 2.99);

INSERT INTO student VALUES (6, 'Edward', 'Xiung', 'Chemistry', 3.45);

DELETE from Student -- when you want to delete all values in the table

WHERE adm\_no = 8;

INSERT INTO student (adm\_no, f\_name, l\_name, major, gpa)

VALUES (8, 'Bai', 'Li','Geography', 4.00);

SELECT \*

FROM student

LIMIT 3;

SELECT gpa, f\_name, major

FROM student;

SELECT gpa, f\_name, major

FROM student

ORDER BY gpa;

SELECT \*

FROM student

WHERE major= 'Chemistry';

SELECT \*

FROM student

WHERE major IN ('Chemistry', 'Bio', 'Education');

SELECT gpa, major,f\_name

FROM student

ORDER BY f\_name DESC;

SELECT gpa, major,f\_name

FROM student

ORDER BY f\_name, gpa DESC

LIMIT 4;

SELECT student.gpa, student.f\_name, student.major

FROM student;

UPDATE student

SET major = 'Bio'

WHERE major = 'Biology';

UPDATE student

SET major = 'CompSci'

WHERE major = 'Computer Science';

-- <, >, >=, <=, <>, AND, OR, LIKE: comparisons

-- <> means not equal to

-- Creating Company DATABASE

CREATE TABLE employee (

emp\_id INT PRIMARY KEY,

first\_name VARCHAR(40),

last\_name VARCHAR(40),

birth\_day DATE,

sex VARCHAR(1),

salary INT,

super\_id INT,

branch\_id INT

);

SELECT \*

FROM employee;

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 PRIMARY KEY,

client\_name VARCHAR(40),

branch\_id INT,

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

);

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

-- Corporate

INSERT INTO employee VALUES(100, 'David', 'Wallace', '1967-11-17', 'M', 250000, NULL, NULL);-- because the corporate branch hasn't been created yet

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

UPDATE employee

SET branch\_id = 1

WHERE emp\_id = 100;

INSERT INTO employee VALUES(101, 'Jan', 'Levinson', '1961-05-11', 'F', 110000, 100, 1);

-- Scranton

INSERT INTO employee VALUES(102, 'Michael', 'Scott', '1964-03-15', 'M', 75000, 100, NULL);

INSERT INTO branch VALUES(2, 'Scranton', 102, '1992-04-06');

UPDATE employee

SET branch\_id = 2

WHERE emp\_id = 102;

INSERT INTO employee VALUES(103, 'Angela', 'Martin', '1971-06-25', 'F', 63000, 102, 2);

INSERT INTO employee VALUES(104, 'Kelly', 'Kapoor', '1980-02-05', 'F', 55000, 102, 2);

INSERT INTO employee VALUES(105, 'Stanley', 'Hudson', '1958-02-19', 'M', 69000, 102, 2);

-- Stamford

INSERT INTO employee VALUES(106, 'Josh', 'Porter', '1969-09-05', 'M', 78000, 100, NULL);

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

UPDATE employee

SET branch\_id = 3

WHERE emp\_id = 106;

INSERT INTO employee VALUES(107, 'Andy', 'Bernard', '1973-07-22', 'M', 65000, 106, 3);

INSERT INTO employee VALUES(108, 'Jim', 'Halpert', '1978-10-01', 'M', 71000, 106, 3);

-- BRANCH SUPPLIER

INSERT INTO branch\_supplier VALUES(2, 'Hammer Mill', 'Paper');

INSERT INTO branch\_supplier VALUES(2, 'Uni-ball', 'Writing Utensils');

INSERT INTO branch\_supplier VALUES(3, 'Patriot Paper', 'Paper');

INSERT INTO branch\_supplier VALUES(2, 'J.T. Forms & Labels', 'Custom Forms');

INSERT INTO branch\_supplier VALUES(3, 'Uni-ball', 'Writing Utensils');

INSERT INTO branch\_supplier VALUES(3, 'Hammer Mill', 'Paper');

INSERT INTO branch\_supplier VALUES(3, 'Stamford Lables', 'Custom Forms');

-- CLIENT

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, LLC', 3);

INSERT INTO client VALUES(404, 'Scranton Whitepages', 2);

INSERT INTO client VALUES(405, 'Times Newspaper', 3);

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

-- 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, 22500);

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 employee; -- find all employees

SELECT \* FROM works\_with;-- -- find all clients

SELECT \* FROM branch; -- find all branches

SELECT \* FROM client;

SELECT \* FROM branch\_supplier;

-- find all employees ordered by salary

SELECT \*

FROM employee

ORDER BY salary DESC;

-- find all employees ordered by sex then name

SELECT \*

FROM employee

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

-- find the first 5 employees in the table

SELECT \*

FROM employee

LIMIT 5;

-- Find the first and last names of all employees

SELECT first\_name, last\_name

FROM employee;

-- Find the forename and surnames names of all employees

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

FROM employee;

-- find out all the different genders

SELECT DISTINCT sex

FROM employee;

-- find the number of employees

SELECT COUNT (emp\_id)

FROM employee;

-- find the number of female employees born after 1970

SELECT COUNT (emp\_id)

FROM employee

WHERE sex = 'F' AND birth\_day > '1971-01-01';

-- find the evarage of all employee's salaries

SELECT AVG (salary)

FROM employee;

-- find the evarage of all male employee's salaries

SELECT AVG (salary)

FROM employee

WHERE sex = 'M';

-- find the evarage of all female employee's salaries

SELECT AVG (salary)

FROM employee

WHERE sex = 'F';

-- find the sum of all employee's salaries

SELECT SUM (salary)

FROM employee;

-- find out how many males and females there are-- this is aggregation

SELECT COUNT (sex), sex

FROM employee

GROUP BY sex;

-- find the total sales of each salesman (employee)

SELECT emp\_id, SUM (total\_sales)

FROM works\_with

GROUP BY emp\_id;

-- find the total amount spent by each client

SELECT client\_id, SUM (total\_sales)

FROM works\_with

GROUP BY client\_id;

-- Find all employees at branch 2

SELECT \*

FROM employee

WHERE branch\_id = 2;

Describe branch;

Describe employee;

-- Find all employee's id's and names who were born after 1969

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

FROM employee

WHERE birth\_day >= 1970-01-01;

-- Find all female employees at branch 2

SELECT \*

FROM employee

WHERE branch\_id = 2 AND sex = 'F';

-- Find all employees who are female & born after 1969 or who make over 80000

SELECT \*

FROM employee

WHERE (birth\_day >= '1970-01-01' AND sex = 'F') OR salary > 80000;

-- Find all employees born between 1970 and 1975

SELECT \*

FROM employee

WHERE birth\_day BETWEEN '1970-01-01' AND '1975-01-01';

-- Find all employees named Jim, Michael, Johnny or David

SELECT \*

FROM employee

WHERE first\_name IN ('Jim', 'Michael', 'Johnny', 'David');

-- Functions

-- Find the number of employees

SELECT COUNT(super\_id)

FROM employee;

-- Find the average of all employee's salaries

SELECT AVG(salary)

FROM employee;

-- Find the sum of all employee's salaries

SELECT SUM(salary)

FROM employee;

-- Find out how many males and females there are

SELECT COUNT(sex), sex

FROM employee

GROUP BY sex;

-- Find the total sales of each salesman

SELECT SUM(total\_sales), emp\_id

FROM works\_with

GROUP BY client\_id;

-- Find the total amount of money spent by each client

SELECT SUM(total\_sales), client\_id

FROM works\_with

GROUP BY client\_id;

-- wildcards

-- % = any # characters, \_ = one character

-- Find any client's who are an LLC

SELECT \*

FROM client

WHERE client\_name LIKE '%LLC';

-- Find any branch suppliers who are in the label business

SELECT \*

FROM branch\_supplier

WHERE supplier\_name LIKE '% Label%';

-- Find any employee born on the 10th day of the month

SELECT \*

FROM employee

WHERE birth\_day LIKE '\_\_\_\_\_10%';

-- Find any clients who are schools

SELECT \*

FROM client

WHERE client\_name LIKE '%school%';

-- union

-- find a list of all employee first name and branch names

SELECT first\_name AS Company\_Names

FROM employee

UNION

SELECT branch\_name

FROM branch

UNION

SELECT client\_name

FROM client;

-- Find a list of employee and branch names

SELECT employee.first\_name AS Employee\_Branch\_Names

FROM employee

UNION

SELECT branch.branch\_name

FROM branch;

-- Find a list of all clients & branch suppliers' names

SELECT client.client\_name AS Non\_Employee\_Entities, client.branch\_id AS Branch\_ID

FROM client

UNION

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

FROM branch\_supplier;

-- find a list of all money spent or earned by the company

DESCRIBE branch\_supplier;

SELECT salary AS Money\_Spent\_or\_Earned

FROM employee

UNION

SELECT total\_sales

FROM works\_with;

-- Joins

-- Add the extra branch

INSERT INTO branch VALUES(4, "Buffalo", NULL, NULL);

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

FROM employee

JOIN branch -- LEFT JOIN, RIGHT JOIN

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

Select \*

FROM branch;

-- Nested Querries

-- Find all clients who are handled by the branch that Michael Scott manages

-- Assume you know Michael's ID

SELECT client\_name

FROM client

WHERE branch\_id = (

SELECT branch\_id

FROM branch

WHERE mgr\_id = 102

);

-- Find names of all employees who have sold over 50,000

SELECT 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 > 50000);

-- Find all clients who are handles by the branch that Michael Scott manages

-- Assume you know Michael's ID

SELECT client.client\_id, client.client\_name

FROM client

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

FROM branch

WHERE branch.mgr\_id = 102);

-- Find all clients who are handled by the branch that Michael Scott manages

-- Assume you DONT'T know Michael's ID

SELECT client.client\_id, client.client\_name

FROM client

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

FROM branch

WHERE branch.mgr\_id = (SELECT employee.emp\_id

FROM employee

WHERE employee.first\_name = 'Michael' AND employee.last\_name ='Scott'

LIMIT 1));

-- Find the names of employees who work with clients handled by the scranton branch- THIS IS MY OWN VERSION OF THE SAME STORY

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

FROM employee

WHERE employee.emp\_id IN (SELECT works\_with.emp\_id

FROM works\_with

WHERE works\_with.client\_id IN (SELECT client.client\_id

FROM client

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

FROM branch

WHERE branch.branch\_name = 'Scranton')));

-- Find the names of employees who work with clients handled by the scranton branch

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

FROM employee

WHERE employee.emp\_id IN (

SELECT works\_with.emp\_id

FROM works\_with

)

AND employee.branch\_id = 2;

-- Find the names of all clients who have spent more than 100,000 dollars- MY OWN STORY

SELECT client\_name

FROM client;

SELECT client\_id, total\_sales

FROM works\_with

WHERE;

SELECT client\_id, SUM(total\_sales) AS totals

FROM works\_with

GROUP BY client\_id

WHERE totals >100000;

CREATE TABLE trigger\_test (message VARCHAR (100));

DELIMITER $$

CREATE

TRIGGER my\_trigger BEFORE INSERT

ON employee

FOR EACH ROW BEGIN

INSERT INTO trigger\_test VALUES('added new employee');

END$$

DELIMITER ;

INSERT INTO employee

VALUES(109, 'Oscar', 'Martinez', '1968-02-19', 'M', 69000, 106, 3);

-- Find the names of all clients who have spent more than 100,000 dollars

SELECT client.client\_name

FROM client

WHERE client.client\_id IN (

SELECT client\_id

FROM (

SELECT SUM(works\_with.total\_sales) AS totals, client\_id

FROM works\_with

GROUP BY client\_id) AS total\_client\_sales

WHERE totals > 100000

);

-- Entity Relationship ER

-- Entity- an object we want to model and store information about

-- entity->attributes->composite attributes->multi-valeud attribute, derived attrinutes

-- single line- partial participation\

DELETE TABLE employee;

SELECT NOW ();

SELECT CURRENT\_DATE();

SELECT first\_name

FROM employee

;

SELECT team\_name

WHERE team\_won IN (1,3,5,7);

DROP TABLE employee;

CREATE TABLE employee (

employee\_id INT,

first\_name VARCHAR (20),

last\_name VARCHAR (20),

birth\_date DATETIME,

gender CHAR (1),

position VARCHAR (20),

department\_id INT,

salary INT,

PRIMARY KEY(employee\_id),

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

ALTER TABLE employee

ADD COLUMN birth\_date DATETIME;

select \*

from employee;

drop table employee;

INSERT INTO employee VALUES (2002, 'Super', 'Man', '2018-11-19 10:36:46.520','M', 'Tester', 1, 75000);

INSERT INTO employee VALUES (2003, 'Jessica', 'Liyers','2018-11-18 11:36:26.400', 'F', 'Architect', 1, 60000);

INSERT INTO employee VALUES (2004, 'Bonnie', 'Adams', '2017-12-22 10:40:10.300', 'F', 'Project Manager', 1, 80000);

INSERT INTO employee VALUES (2005, 'James', 'Madison', '2017-12-30 9:30:20.100', 'M', 'Software Developer', 1, 55000);

INSERT INTO employee VALUES (2006, 'Michael', 'Greenback', '2017-11-25 7:25:14.700', 'M', 'Sales Assistant', 2, 85000);

INSERT INTO employee VALUES (2007, 'Leslie', 'Peters', '2017-10-09 8:26:14.800', 'F', 'Sales Engineer', 2, 76000);

INSERT INTO employee VALUES (2008, 'Max', 'Powers', '2017-10-09 9:40:18.900', 'M', 'Sales Representative', 2, 59000);

INSERT INTO employee VALUES (2009, 'Stacy', 'Jacobs', '2018-11-15 10:35:17.600', 'F', 'Sales Manager', 2, 730000);

INSERT INTO employee VALUES (2010, 'John', 'Henery', '2018-11-16 9:14:17.600', 'M', 'Sales Director', 2, 90000);

INSERT INTO employee VALUES (2011, 'Nancy', 'Oduor', '2018-11-17 11:16:18.600', 'F', 'Data Analyst', 2, 96000);

SELECT \*

FROM employee;

CREATE TABLE department

(department\_id INT,

department\_name VARCHAR (20));

ALTER TABLE department

ADD PRIMARY KEY(department\_id);

INSERT INTO department VALUES (1, 'IT');

INSERT INTO department VALUES (2, 'Sales');

SELECT \*

FROM department;

SELECT \*

FROM employee;

-- Return employee record with max salary

SELECT \*

FROM employee

WHERE salary = (SELECT MAX(salary)

FROM employee);

-- select highest salary in employee table

SELECT salary

FROM employee

WHERE salary = (SELECT MAX(salary)

FROM employee);

-- select 2nd highest salary in employee table

SELECT MAX(salary)

FROM employee

WHERE salary <> (SELECT MAX(salary)

FROM employee);

-- select range of employee based in ID

SELECT \*

FROM employee

WHERE employee\_id BETWEEN 2003 AND 2008;

-- return employee name, highest salary and department name

SELECT employee.first\_name, employee.last\_name, employee.salary, department.department\_name

FROM employee

JOIN department

ON employee.department\_id = department.department\_id

WHERE salary = (SELECT MAX(salary)

FROM employee);

-- return highets salary, employee name and department name for each department

SELECT employee.salary, employee.first\_name, employee.last\_name, department.department\_name

FROM employee

JOIN department

ON employee.department\_id = department.department\_id

WHERE salary IN (SELECT MAX(salary)

FROM employee

GROUP BY department\_id);

SELECT CURRENT\_DATE;

-- emongs employees, who get the highest salary

SELECT first\_name, last\_name, salary

FROM employees

WHERE salary = (SELECT MAX(salary)

FROM employees);

UPDATE employee

SET position = 'Sales Assistant'

WHERE position = 'Sales Representative';

select \* from employee;

-- Get details of an architect earning 60000

Select \*

from employee

where (position, salary) IN (SELECT position, salary

from employee

where position ='Architect' and salary = 60000);

-- which designation has exactly two employees

SELECT position, count(position)

FROM employee

group by position

having count(position) = 2;