```
DROP TABLE student;
CREATE TABLE Student (
    student_id INT auto_increment,
    name VARCHAR(20) ,
    major VARCHAR(20) DEFAULT 'UNDECIDED',
    PRIMARY KEY(student_id)
);
SELECT * FROM student;

DROP TABLE Student;

INSERT INTO Student(student_id, name) VALUES(1, 'Jack');
INSERT INTO Student VALUES(2, 'Kate', 'Sociology');
INSERT INTO student VALUES(4, 'Jack', 'Biology');
INSERT INTO Student VALUES(5, 'Mike', 'Computer Science');
INSERT INTO Student VALUE(3, NULL, "Chemistry");


UPDATE student
SET major = 'Biochemistry'
WHERE major = 'Bio'or 'Chemistry';

SELECT student.name, major
FROM student
ORDER BY major, student_id;
LIMit 2;

SELECT *
FROM student
where major = 'Chemistry' or name = 'Kate';

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

WHERE name IN ('Clair', 'Kate', 'Jack')
#simple queries, will get more complex with multiple databases
#and more specific queries

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

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
);

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),
```

```sql
  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);

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);
#circular relationship between foreign keys, for more complex database schema
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 employee;


--Find all employees ordered by salary
SELECT *
FROM employee
ORDERED BY salary DESC;
```

```
-- Find all employees ordered by sex then name
SELECT *
FROM employee
ORDERED BY sex, first_name, last_name;

--Find the first and last names of all employees
SELECT first_name, last_name
FROM employee;

-- find the forename and surbanes banes of all employees
SELECT first_name AS forename, last_name AS surname
FROM employye;

--FIND out all the ddifferent genders
SELECT DISTINCT GENDER
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_date > '1970-01-01';

--Find the average of all employees salaries
SELECT AVG(salary)
FROM employee
WHERE sex = 'M';

--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 emp_id;
```

```
--Find the total each client spent
SELECT SUM(total_sales), client_id
FROM works_with
GROUP BY client_id;

% = any character and _ is one character
--WilDCARDS
--Find any client's who are an LLC
SELECT *
FROM client
WHERE client_name LIKE '%LLC';

--find any branch supplier who are in the label business
SELECT *
FROM branch_supplier
WHERE supplier_name LIKE '% Label%';

--Find any employee born in October
SELECT *
FROM employee
WHERE birth_day LIKE '____-02%';

--Find any clients who are schools
SELECT *
FROM client
WHERE client_name LIKE '%School%';

--UNION(multiple select statements into one)
--Find a list of employee and branch names
SELECT first_name
FROM employee;

SELECT branch_name
FROM branch;


--find a list of employers and branch names
SELECT first_name #same columns
FROM  employee
UNION
SELECT  branch_name
FROM branch
UNION
```

```
select client_name
FROM client;

--Find a list of all clients & branch suppliers names
SELECT  client_name, client.branch_id
FROM CLIENT
union
select supplier_name, branch_id
from branch_supplier;

--Find a list of all money spent or earned by the company
SELECT salary
FROM employee
union
select total_sales
from works_with;

--Joins
INSERT INTO branch VALUES(4, 'Buffalo', NULL, NULL);

--Find all branches and the names of their managers
SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
JOIN branch
ON employee.emp_id = branch.mgr_id;
--combine multiple tables and columns
--4 basic type of joins
--left join and 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 IS A RIGHT AND LEFT JOIN COMBINED

--NESTED QUERIES
--FIND names of all employees who have sold over 30,000 to a single client
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 > 30000
);
```

```
--Find all clients who are handled by the branch that Micheal Scott manges and assume you know micheals ID
SELECT client.client_name
FROM client
WHERE client.branch_id = (
   SELECT branch.branch_id
   FROM branch
   WHERE branch.mgr_id = 102
   LIMIT 1
);
#combing multiple queries to find information

--On delete set null and on delete cascade
DELETE FROM employee
WHERE emp_id = 102;

SELECT * FROM branch;
#on delete cascade deletes entire row
DELETE FROM branch
WHERE branch_id = 2;

SELECT * from branch_supplier;
#useful for defining foreign key relationships

--Triggers = CREATE IN TERMINAL
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);

DELIMITER $$
CREATE
```

```
    TRIGGER my_trigger1 BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        INSERT INTO trigger_test VALUES(NEW.first_name);
    END$$
DELIMITER ;

INSERT INTO employee
VALUES(110, 'Kevin', 'Malone', '1978-02-19', 'M', 69000, 106, 3);


SELECT * FROM trigger_test;

DELIMITER $$
CREATE
    TRIGGER my_trigger2 BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        IF NEW.sex = "M" THEN
            INSERT INTO trigger_test VALUES('added male employee');
        ELSEIF NEW.sex = 'F' THEN
            INSERT INTO trigger_test VALUES('added female employee');
        ELSE
            INSERT INTO trigger_test VALUES('added other employee');
        END IF;
    END$$
DELIMITER ;

INSERT INTO employee
VALUES(111, 'Pam Beasley', )

SELECT * FROM trigger_test;

- ER DIAGRAM
--ER = Entity Relationship
--entity, attributes, primary key, composite attribute, multivalued attribute,
--derived attribute, multiple entitites, relationships, total participation, one and two lines for partial and total participation
--relationship attribute, relationship cardinality, weak entity, idenitfying relationship,

--Company Data Requirements
--Converting ER diagram to db schema
--step 1 mapping of regular entity types
--step 2 mapping of weak entity types
```

--step 3 Mapping of binary 1:1 relationship types

--step 4 mapping of binary 1:N relationship types

--step 5 mapping of binary M:N relationship types

#practice er diagram by building them and converting them into database schemas