SQL Code QUERIES and COMMANDS

*Refer to **SQL NOTES** for OUTPUTS

```sql
-- Create 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
);

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
);
--ADD branch id in employee table as a forreign key
ALTER TABLE employee
ADD FOREIGN KEY(branch_id)
REFERENCES branch(branch_id)
ON DELETE SET NULL;

--ADD supervisor ids (super_id) as a forreign key in employee table
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
);


--INSERT INFO in tables

-- 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);

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);
```

```sql
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 works_with;

--Querying the company database-----------------------------

--Find all employees
select * from employee;

--Find all clients
select * from client
limit 100;

--Find all employees ordered by salary
select * from employee
ORDER BY salary DESC;

-- Find all employees by sex then name
select * from employee
ORDER BY sex DESC, first_name ASC, last_name DESC
limit 100;

-- Find the first 5 employees in the table
select * from employee
limit 5;

--Find first and last names of all employees
SELECT first_name, last_name
FROM employee
LIMIT 50;

--Find the forenames and surnames of all employees
SELECT employee.first_name AS forename, employee.last_name AS surname
FROM employee
LIMIT 50;

--Find out all different genders
SELECT DISTINCT employee.sex
FROM employee;

--Find out all different branch ids
SELECT DISTINCT employee.branch_id
```

```sql
FROM employee;

--------FUNCTIONS---------

-- Find number of employees
SELECT COUNT(emp_id)
FROM employee;

-- Count number of employees that have supervisors
SELECT COUNT(super_id)
FROM employee;

--Find number of females employees born after 1970
SELECT COUNT(emp_id)
FROM employee
WHERE sex = 'F' AND birth_day >= '1971-01-01'
LIMIT 100;


--Find average salary of all employees who are male

SELECT AVG(salary)
FROM employee
WHERE sex = 'M';

--Find sum of all employee salary
SELECT SUM(salary)
FROM employee;

--Find how many males and howmany females there are
SELECT COUNT(sex), sex
FROM employee
GROUP BY sex;

--Find total sales of each salesman //added joins to this
SELECT works_with.emp_id, employee.first_name, sum(works_with.total_sales) AS total_sales
FROM works_with
JOIN employee ON works_with.emp_id = employee.emp_id
GROUP BY works_with.emp_id, employee.first_name;

--How much mony has each client spent
SELECT works_with.client_id, client.client_name, sum(works_with.total_sales) AS total_sales
FROM works_with
JOIN client ON works_with.client_id = client.client_id
GROUP BY works_with.client_id, client.client_name;

-----WILD CARDS-----

--- Find any clients who are an LLC
SELECT * FROM client
WHERE client_name LIKE '%LLC'; --if the client name has any number of chaacters then "LLC"
at the end then return it;

--Find any branch suppliers thats in the label business
SELECT *
FROM branch_supplier
WHERE supplier_name LIKE '% label%' OR supplier_name LIKE '% lables%' ;
```

```sql
--Find any employee born in october

select * from employee;
SELECT *
FROM employee
WHERE birth_day LIKE '____-02%' ;

--Find any client who are schools

select * from client;
SELECT *
FROM client
WHERE client_name LIKE '%school%' ;

----UNIONS-----

--Find a list of employee and baranch names
SELECT employee.first_name AS All_Names
FROM employee
UNION
SELECT branch.branch_name
From branch
UNION
SELECT client.client_name
FROM client;

--Find a list of all clients and branch supplier names and branch ids
SELECT client.client_name, client.branch_id
FROM client
UNION
SELECT branch.branch_name, branch.branch_id
From branch;

--Find a list of all money spent or earned by the company
SELECT employee.salary
FROM employee
UNION
SELECT works_with.total_sales
FROM works_with;
--essentially it just puts what ever is first above what ever is selected next into one
table (bottom join)

--------JOINS--------
INSERT INTO branch VALUES(4, 'Buffalo', NULL, NULL); --inserted for example branch with no
mgr_id

-- Find all branches and corrisponding names of managers

SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
JOIN branch -- inner join //combines rows fom employee table and branch table with repect
to shared column
ON employee.emp_id = branch.mgr_id; --columns that are in common

SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
LEFT JOIN branch -- left join combines all rows fom employee table and adds to it on the
right branch table.
```

```sql
ON employee.emp_id = branch.mgr_id; --columns that are in common

SELECT employee.emp_id, employee.first_name, branch.branch_name
FROM employee
RIGHT JOIN branch -- Right join includes all rows fom branch table and adds employee table
column to the right of it.
ON employee.emp_id = branch.mgr_id; --columns that are in common

--Full outer join combines both left and right join logic by grabbing all employees and all
branches no matter if they met a certain condition
-- (ie employee.emp_id = branch.mgr_id)
-- Not function in MySQL

-------NESTED QUERIES--------

--Find names of all employees who have sold over 30000 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
--Michael Scott manages
--Assume we know his ID

SELECT branch.branch_id
FROM branch
WHERE branch.mgr_id = 102;

SELECT client.client_name
FROM client
WHERE client.branch_id = (
    SELECT branch.branch_id
    FROM branch
    WHERE branch.mgr_id = 102
    LIMIT 1 --ensures its limited to one output
);

--Note that nested queries work from in to out so in prev example
--we first get the branch id to then find the client names

-----ON DELETE-------

--recall this table we created earlier...
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 -- if the employee
gets deleted then the mgr_id is set to NULL
);

DELETE FROM employee
```

```sql
WHERE emp_id = 102;

select * from employee;

--recall this table we created earlier...
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
);

DELETE FROM branch
WHERE branch_id = 2;

select * from branch_supplier;

-----TRIGGERS------

CREATE TABLE trigger_test(
    message VARCHAR(100)
);

---THE FOLLOWING CAN ONLY BE DONE IN COMMAND LINE NOT IN PopSQL----
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);

---THIS PART CAN BE DONE HERE (PopSQL)---
INSERT INTO employee
VALUES(109, 'Oscar', 'MArtinez', '1968-02-19', 'M', 69000, 106, 3);
select * from trigger_test

--This part is done on command line---
DELIMITER $$
CREATE
    TRIGGER my_trigger_2 BEFORE INSERT
    ON employee
    FOR EACH ROW BEGIN
        INSERT INTO trigger_test VALUES(NEW.first_name);
    END$$
DELIMITER ;
---This part is done on PopSQL
INSERT INTO employee
VALUES(110, 'Kevin', 'Malone', '1978-02-19', 'M', 69000, 106, 3);

---Conditional trigger (if else)

--THIS is done on the command line
DELIMITER $$
```

```
CREATE
    TRIGGER my_trigger_3 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');
         ELSE
               INSERT INTO trigger_test VALUES('added other employee');
         END IF;
    END$$
DELIMITER ;

--This can be done on PopSQL
INSERT INTO employee
VALUES(111, 'Pam', 'Beesly', '1988-02-19', 'F', 69000, 106, 3);

select * from trigger_test;

--In addition to BEFORE INSERT we can do BEFORE UPDATE, BEFORE DELETE or
--AFTER INSERT we can do AFTER UPDATE, AFTER DELETE
--Triggers can be deleted aswell...
DROP TRIGGER my_trigger;
```