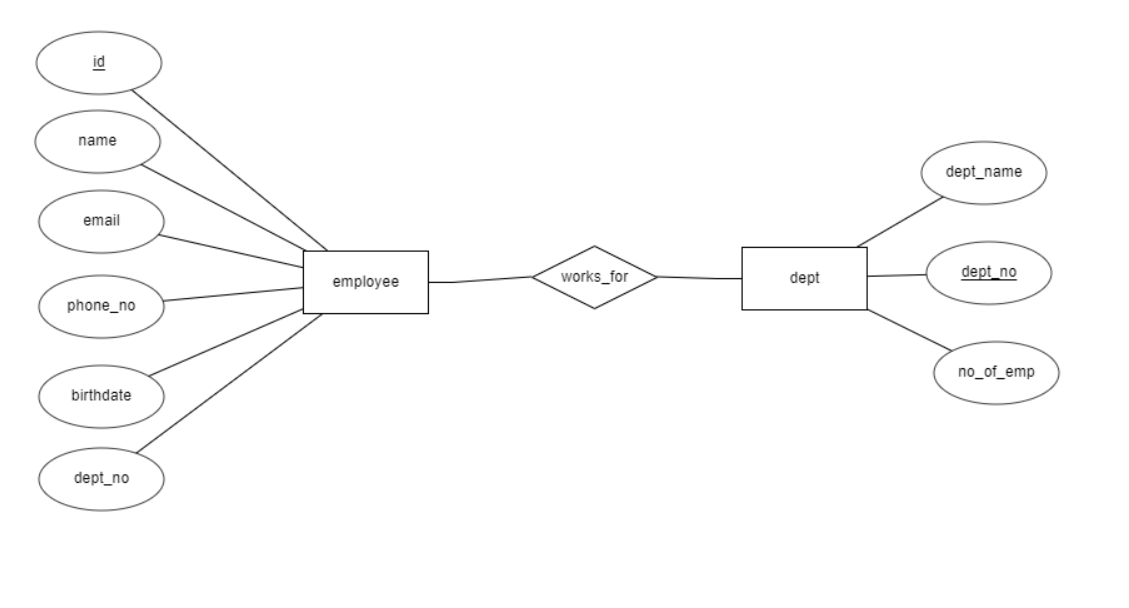
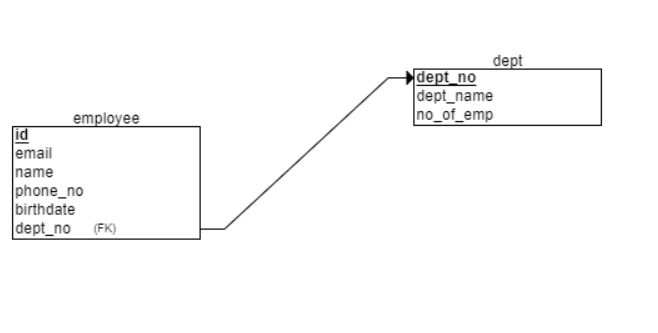
**DBMS LAB  
WEEK 9 & 10**

**PES1UG19CS592**

**Yashi Chawla**

1. Create an employee table which contains employee details and the department he works for. Create another table department consisting of dname and number of employees. Write triggers to increment or decrement the number of employees in a department table when the record in the employee table is inserted or deleted respectively.





CREATE.sql

drop database employee;

create database employee;

\c employee

CREATE TABLE dept

(

    dept\_name VARCHAR(50) NOT NULL,

    dept\_no INT NOT NULL,

    no\_of\_emp INT NOT NULL,

    PRIMARY KEY(dept\_no)

);

CREATE TABLE employee\_details

(

    email VARCHAR(50) NOT NULL,

    name VARCHAR(50) NOT NULL,

    id INT NOT NULL,

    phone\_no CHAR(10) NOT NULL,

    birthdate DATE NOT NULL,

    dept\_no INT NOT NULL,

    PRIMARY KEY(id),

    FOREIGN KEY(dept\_no) REFERENCES dept(dept\_no)

);

-- trigger to increase number of employees in the department that the insertion was made in

CREATE OR REPLACE FUNCTION increment() RETURNS TRIGGER AS

$$

BEGIN

  UPDATE dept set no\_of\_emp=no\_of\_emp+1 where new.dept\_no=dept.dept\_no;

    RETURN new;

END;

$$

language plpgsql;

CREATE TRIGGER increment\_number

     AFTER INSERT ON employee\_details

     FOR EACH ROW

     EXECUTE PROCEDURE increment();

-- trigger to decrease number of employees in the department that the deletion was made in

CREATE OR REPLACE FUNCTION decrement() RETURNS TRIGGER AS

$$

BEGIN

  UPDATE dept set no\_of\_emp=no\_of\_emp-1 where old.dept\_no=dept.dept\_no;

    RETURN new;

END;

$$

language plpgsql;

CREATE TRIGGER decrement\_number

    AFTER DELETE ON employee\_details

    FOR EACH ROW

    EXECUTE PROCEDURE decrement();

INSERT.SQL ( for testing )

\c employee

insert into dept values('CSE',1,0);

insert into dept values('EEE',2,0);

insert into dept values('Mech',3,0);

insert into employee\_details values ('abrandino0@hatena.ne.jp', 'Adaline Brandino', 1, '7524400096', '1997-02-16', 1);

insert into employee\_details values ('hflippelli1@ameblo.jp', 'Hatti Flippelli', 2, '1025817726', '1991-05-10', 2);

insert into employee\_details values ('cmacshane2@cyberchimps.com', 'Caddric MacShane', 3, '2512840590', '1999-07-22', 1);

insert into employee\_details values ('cdaldan3@odnoklassniki.ru', 'Cad Daldan', 4, '6201054879', '1996-11-26', 2);

insert into employee\_details values ('scoie4@amazon.de', 'Sylvan Coie', 5, '3143476971', '1999-03-02', 1);

insert into employee\_details values ('wmacdavitt5@1und1.de', 'Wayne MacDavitt', 6, '3144736168', '1995-05-02', 3);

insert into employee\_details values ('fgouldstraw6@miitbeian.gov.cn', 'Faustine Gouldstraw', 7, '8055483283', '1992-04-02', 2);

insert into employee\_details values ('rpalister7@huffingtonpost.com', 'Rab Palister', 8, '9171160664', '1990-12-23', 3);

insert into employee\_details values ('bjagels8@quantcast.com', 'Ber Jagels', 9, '6782063205', '1996-06-28', 2);

insert into employee\_details values ('cmalthus9@biblegateway.com', 'Cristionna Malthus', 10, '3232737530', '1996-11-28', 2);

insert into employee\_details values ('mincognaa@nature.com', 'Malchy Incogna', 11, '4779964782', '1992-12-08', 2);

insert into employee\_details values ('dburgottb@bandcamp.com', 'Deonne Burgott', 12, '1951883479', '1996-03-17', 3);

insert into employee\_details values ('dgrillsc@comcast.net', 'Dalston Grills', 13, '5016132428', '1995-08-21', 1);

insert into employee\_details values ('msaddletond@histats.com', 'Milly Saddleton', 14, '8462521055', '1997-08-09', 2);

insert into employee\_details values ('smcgrathe@microsoft.com', 'Sybille McGrath', 15, '5568667327', '1998-09-23', 2);

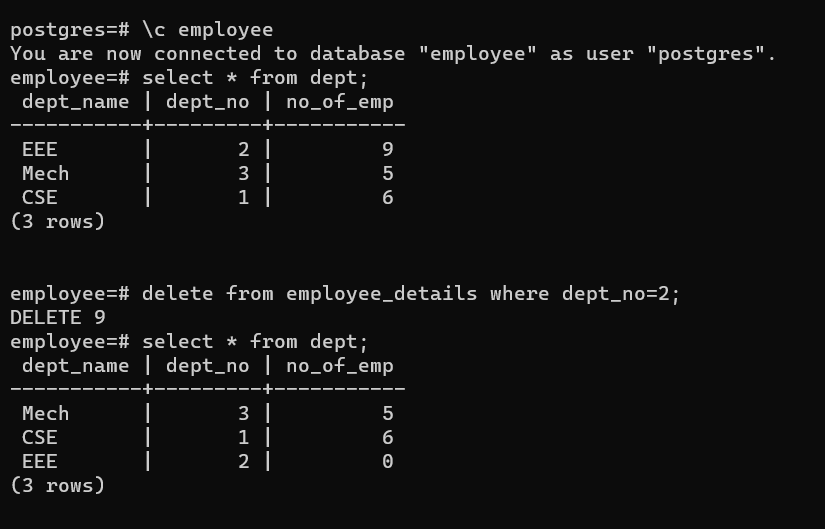
insert into employee\_details values ('uparminterf@aol.com', 'Upton Parminter', 16, '6411778628', '1995-05-09', 3);

insert into employee\_details values ('rphillcockg@eventbrite.com', 'Raquela Phillcock', 17, '4952090013', '1994-08-22', 2);

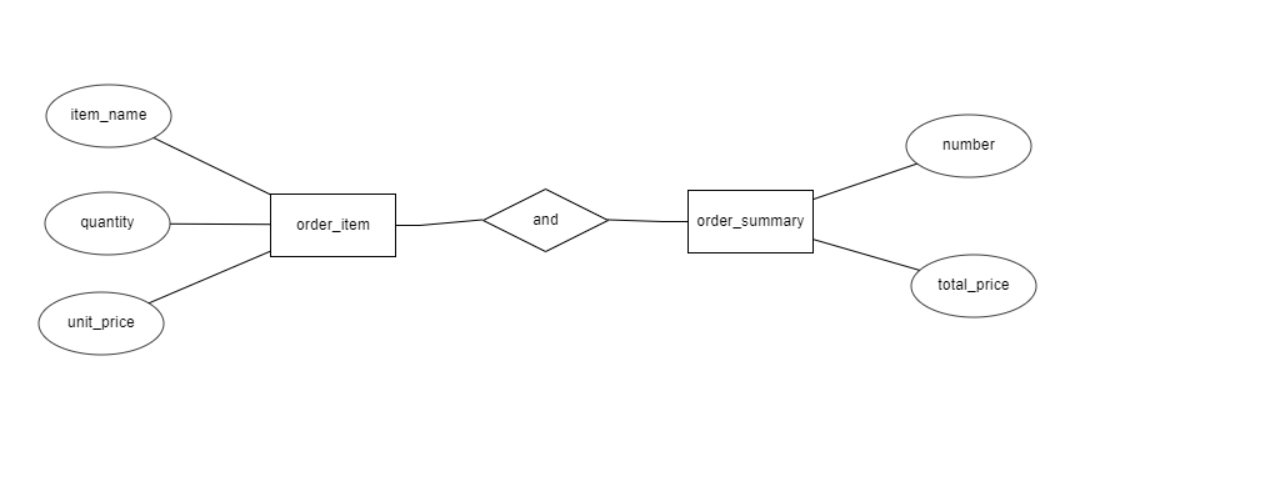
insert into employee\_details values ('sdarraghh@51.la', 'Sula Darragh', 18, '2737136188', '1992-11-26', 1);

insert into employee\_details values ('cmcgilvrayi@tmall.com', 'Cathie McGilvray', 19, '6167674075', '1992-11-12', 3);

insert into employee\_details values ('fquarlessj@soundcloud.com', 'Farleigh Quarless', 20, '8318136141', '1995-01-13', 1);



1. Create an order\_item table which contains details like name, quantity and unit price of every item purchased. Create an order summary table that contains number of items and total price. Create triggers to update entry in order summary whenever an item is inserted or deleted in the order item table.



CREATE.sql

drop database order\_details;

create database order\_details;

\c order\_details

CREATE TABLE order\_item

(

  item\_name VARCHAR(50) NOT NULL,

  quantity INT NOT NULL,

  unit\_price INT NOT NULL

);

CREATE TABLE order\_summary

(

  number\_of\_items INT NOT NULL,

  total\_price INT NOT NULL

);

CREATE OR REPLACE FUNCTION increment() RETURNS TRIGGER AS

$$

BEGIN

  UPDATE order\_summary set number\_of\_items=number\_of\_items+1;

  UPDATE order\_summary set total\_price=total\_price+new.quantity\*new.unit\_price;

    RETURN new;

END;

$$

language plpgsql;

CREATE TRIGGER increment\_number

     AFTER INSERT ON order\_item

     FOR EACH ROW

     EXECUTE PROCEDURE increment();

CREATE OR REPLACE FUNCTION decrement() RETURNS TRIGGER AS

$$

BEGIN

  UPDATE order\_summary set number\_of\_items=number\_of\_items-1;

  UPDATE order\_summary set total\_price=total\_price-old.quantity\*old.unit\_price;

  RETURN new;

END;

$$

language plpgsql;

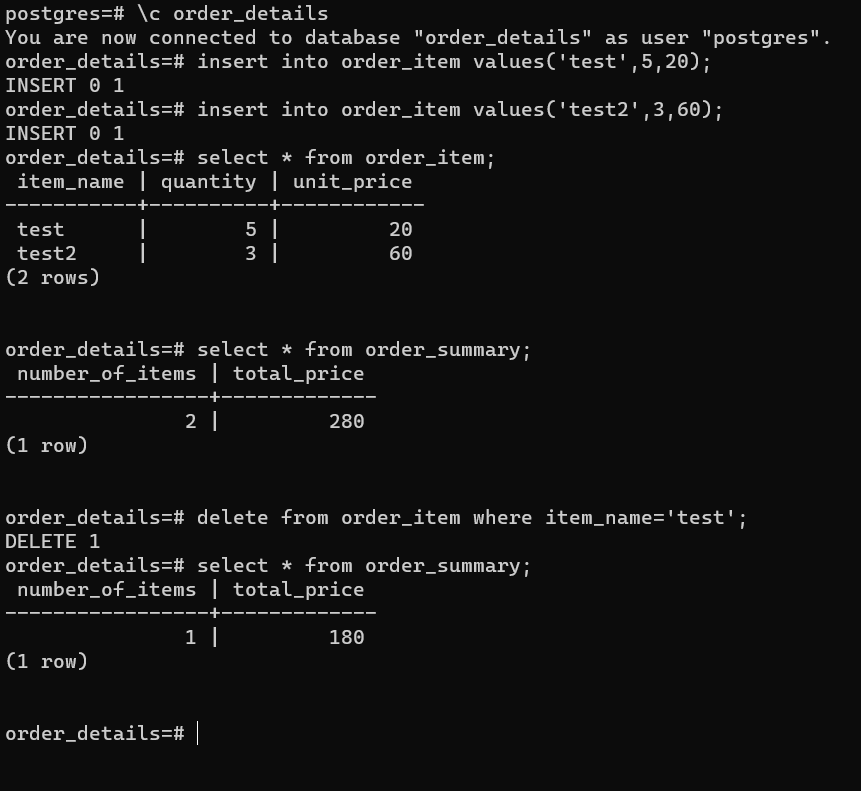
CREATE TRIGGER decrement\_number

     AFTER DELETE ON order\_item

     FOR EACH ROW

     EXECUTE PROCEDURE decrement();

insert into order\_summary values(0,0);



note: the instruction pdf does not specify the exact structure of the order\_summary table. It could be implemented in two ways.

1. Each row would have number of items of a particular product (being redundant to the quantity in the order\_item table) and the total price would be the quanity\*unit price for each item.
2. It is a one row table, where number of items is the total number of items present in the order\_item table and total price is the total summation of all item’s quantity into their unit price.

The 2nd has been implemented here.