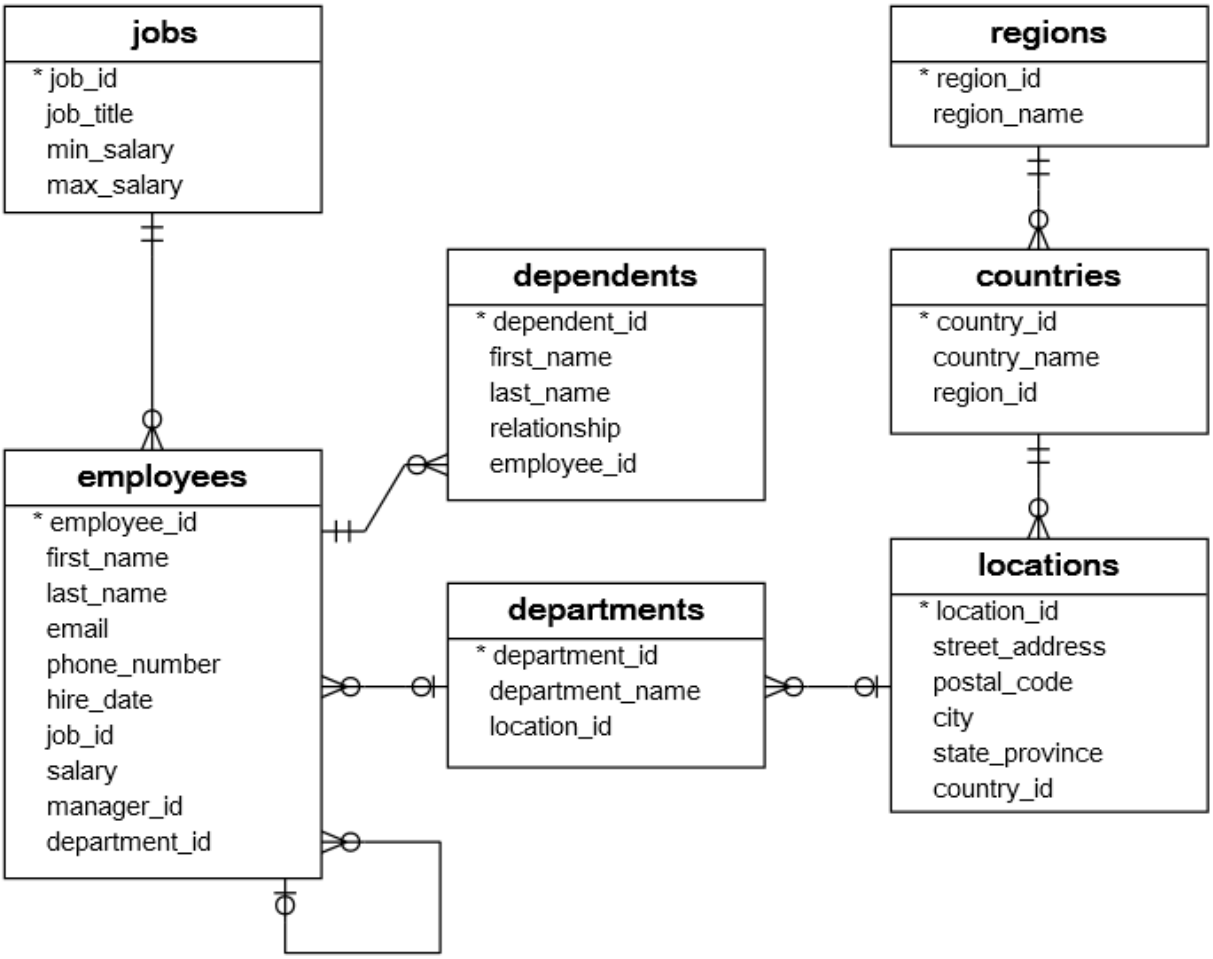
QUESTIONS



Overview of Project:--

The HR sample database has seven tables:

1. The employees table stores the data of employees.
2. The jobs table stores the job data including job title and salary range.
3. The departments table stores department data.
4. The dependents table stores the employee's dependents.
5. The locations table stores the location of the departments of the company.

6. The countries table stores the data of countries where the company is doing business.
7. The regions table stores the data of regions such as Asia, Europe, America, and the Middle East and Africa. The countries are grouped into regions.

The following picture shows the table names and their records.

Table	**Rows**
employees	40
dependents	30
departments	11
jobs	11
locations	7
countries	25
regions	4

Create the tables given below or else you can Copy the below table and paste it into your SSMS(SQL Server Management Studio)

```
CREATE TABLE regions (

    region_id INT IDENTITY(1,1) PRIMARY KEY,

    region_name VARCHAR (25) DEFAULT NULL
```

```
);

CREATE TABLE countries (

	country_id CHAR (2) PRIMARY KEY,

	country_name VARCHAR (40) DEFAULT NULL,

	region_id INT NOT NULL,

	FOREIGN KEY (region_id) REFERENCES regions (region_id) ON DELETE CASCADE
ON UPDATE CASCADE

);

CREATE TABLE locations (

	location_id INT IDENTITY(1,1) PRIMARY KEY,

	street_address VARCHAR (40) DEFAULT NULL,

	postal_code VARCHAR (12) DEFAULT NULL,

	city VARCHAR (30) NOT NULL,

	state_province VARCHAR (25) DEFAULT NULL,

	country_id CHAR (2) NOT NULL,

	FOREIGN KEY (country_id) REFERENCES countries (country_id) ON DELETE
CASCADE ON UPDATE CASCADE

);

CREATE TABLE jobs (

	job_id INT IDENTITY(1,1) PRIMARY KEY,

	job_title VARCHAR (35) NOT NULL,

	min_salary DECIMAL (8, 2) DEFAULT NULL,

	max_salary DECIMAL (8, 2) DEFAULT NULL
```

```
);

CREATE TABLE departments (

	department_id INT IDENTITY(1,1) PRIMARY KEY,

	department_name VARCHAR (30) NOT NULL,

	location_id INT DEFAULT NULL,

	FOREIGN KEY (location_id) REFERENCES locations (location_id) ON DELETE
CASCADE ON UPDATE CASCADE

);



CREATE TABLE employees (

	employee_id INT IDENTITY(1,1) PRIMARY KEY,

	first_name VARCHAR (20) DEFAULT NULL,

	last_name VARCHAR (25) NOT NULL,

	email VARCHAR (100) NOT NULL,

	phone_number VARCHAR (20) DEFAULT NULL,

	hire_date DATE NOT NULL,

	job_id INT NOT NULL,

	salary DECIMAL (8, 2) NOT NULL,

	manager_id INT DEFAULT NULL,

	department_id INT DEFAULT NULL,

	FOREIGN KEY (job_id) REFERENCES jobs (job_id) ON DELETE CASCADE ON
UPDATE CASCADE,
```

```
    FOREIGN KEY (department_id) REFERENCES departments (department_id) ON
DELETE CASCADE ON UPDATE CASCADE,

    FOREIGN KEY (manager_id) REFERENCES employees (employee_id)

);
```

```
CREATE TABLE dependents (

    dependent_id INT IDENTITY(1,1) PRIMARY KEY,

    first_name VARCHAR (50) NOT NULL,

    last_name VARCHAR (50) NOT NULL,

    relationship VARCHAR (25) NOT NULL,

    employee_id INT NOT NULL,

    FOREIGN KEY (employee_id) REFERENCES employees (employee_id) ON DELETE
CASCADE ON UPDATE CASCADE

);
```

After Creating All tables insert the given records and execute it in your SSMS

/*Data for the table regions */

```
INSERT INTO regions(region_id,region_name)

VALUES (1,'Europe');

INSERT INTO regions(region_id,region_name)

VALUES (2,'Americas');

INSERT INTO regions(region_id,region_name)

VALUES (3,'Asia');

INSERT INTO regions(region_id,region_name)

VALUES (4,'Middle East and Africa');
```

/*Data for the table countries */

```
INSERT INTO countries(country_id,country_name,region_id)

VALUES ('AR','Argentina',2);
```

```
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('AU','Australia',3);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('BE','Belgium',1);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('BR','Brazil',2);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('CA','Canada',2);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('CH','Switzerland',1);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('CN','China',3);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('DE','Germany',1);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('DK','Denmark',1);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('EG','Egypt',4);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('FR','France',1);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('HK','HongKong',3);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('IL','Israel',4);
INSERT INTO countries(country_id,country_name,region_id)
```

```
VALUES ('IN','India',3);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('IT','Italy',1);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('JP','Japan',3);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('KW','Kuwait',4);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('MX','Mexico',2);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('NG','Nigeria',4);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('NL','Netherlands',1);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('SG','Singapore',3);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('UK','United Kingdom',1);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('US','United States of America',2);
INSERT INTO countries(country_id,country_name,region_id)
 VALUES ('ZM','Zambia',4);
INSERT INTO countries(country_id,country_name,region_id)
VALUES ('ZW','Zimbabwe',4);
```

/*Data for the table locations */

```
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
 VALUES (1400,'2014 Jabberwocky Rd','26192','Southlake','Texas','US');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
 VALUES (1500,'2011 Interiors Blvd','99236','South San Francisco','California','US');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
 VALUES (1700,'2004 Charade Rd','98199','Seattle','Washington','US');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
VALUES (1800,'147 Spadina Ave','M5V 2L7','Toronto','Ontario','CA');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
VALUES (2400,'8204 Arthur St',NULL,'London',NULL,'UK');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
VALUES (2500,'Magdalen Centre, The Oxford Science Park','OX9 9ZB','Oxford','Oxford','UK');
INSERT INTO locations(location_id,street_address,postal_code,city,state_province,country_id)
 VALUES (2700,'Schwanthalerstr. 7031','80925','Munich','Bavaria','DE');
```

/*Data for the table jobs */

```
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (1,'Public Accountant',4200.00,9000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (2,'Accounting Manager',8200.00,16000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (3,'Administration Assistant',3000.00,6000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (4,'President',20000.00,40000.00);
```

```
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (5,'Administration Vice President',15000.00,30000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (6,'Accountant',4200.00,9000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (7,'Finance Manager',8200.00,16000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (8,'Human Resources Representative',4000.00,9000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (9,'Programmer',4000.00,10000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (10,'Marketing Manager',9000.00,15000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (11,'Marketing Representative',4000.00,9000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (12,'Public Relations Representative',4500.00,10500.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (13,'Purchasing Clerk',2500.00,5500.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (14,'Purchasing Manager',8000.00,15000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (15,'Sales Manager',10000.00,20000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (16,'Sales Representative',6000.00,12000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
```

```
 VALUES (17,'Shipping Clerk',2500.00,5500.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
 VALUES (18,'Stock Clerk',2000.00,5000.00);
INSERT INTO jobs(job_id,job_title,min_salary,max_salary)
VALUES (19,'Stock Manager',5500.00,8500.00);
```

/*Data for the table departments */

```
INSERT INTO departments(department_id,department_name,location_id)
 VALUES (1,'Administration',1700);
INSERT INTO departments(department_id,department_name,location_id)
 VALUES (2,'Marketing',1800);
INSERT INTO departments(department_id,department_name,location_id)
 VALUES (3,'Purchasing',1700);
INSERT INTO departments(department_id,department_name,location_id)
 VALUES (4,'Human Resources',2400);
INSERT INTO departments(department_id,department_name,location_id)
VALUES (5,'Shipping',1500);
INSERT INTO departments(department_id,department_name,location_id)
 VALUES (6,'IT',1400);
INSERT INTO departments(department_id,department_name,location_id)
```

```
VALUES (7,'Public Relations',2700);

INSERT INTO departments(department_id,department_name,location_id)

VALUES (8,'Sales',2500);

INSERT INTO departments(department_id,department_name,location_id)

 VALUES (9,'Executive',1700);

INSERT INTO departments(department_id,department_name,location_id)

VALUES (10,'Finance',1700);

INSERT INTO departments(department_id,department_name,location_id)

VALUES (11,'Accounting',1700);
```

/*Data for the table employees */

```
INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (100,'Steven','King','steven.king@sqltutorial.org','515.123.4567','1987-06-17',4,24000.00,NULL,9);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (101,'Neena','Kochhar','neena.kochhar@sqltutorial.org','515.123.4568','1989-09-21',5,17000.00,100,9);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (102,'Lex','De Haan','lex.de haan@sqltutorial.org','515.123.4569','1993-01-13',5,17000.00,100,9);
```

```
INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (103,'Alexander','Hunold','alexander.hunold@sqltutorial.org','590.423.4567','1990-01-03',9,9000.00,102,6);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (104,'Bruce','Ernst','bruce.ernst@sqltutorial.org','590.423.4568','1991-05-21',9,6000.00,103,6);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (105,'David','Austin','david.austin@sqltutorial.org','590.423.4569','1997-06-25',9,4800.00,103,6);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (106,'Valli','Pataballa','valli.pataballa@sqltutorial.org','590.423.4560','1998-02-05',9,4800.00,103,6);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (107,'Diana','Lorentz','diana.lorentz@sqltutorial.org','590.423.5567','1999-02-07',9,4200.00,103,6);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (108,'Nancy','Greenberg','nancy.greenberg@sqltutorial.org','515.124.4569','1994-08-17',7,12000.00,101,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)
```

```
 VALUES (109,'Daniel','Faviet','daniel.faviet@sqltutorial.org','515.124.4169','1994-08-
16',6,9000.00,108,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (110,'John','Chen','john.chen@sqltutorial.org','515.124.4269','1997-09-28',6,8200.00,108,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (111,'Ismael','Sciarra','ismael.sciarra@sqltutorial.org','515.124.4369','1997-09-
30',6,7700.00,108,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (112,'Jose Manuel','Urman','jose manuel.urman@sqltutorial.org','515.124.4469','1998-03-
07',6,7800.00,108,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (113,'Luis','Popp','luis.popp@sqltutorial.org','515.124.4567','1999-12-07',6,6900.00,108,10);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (114,'Den','Raphaely','den.raphaely@sqltutorial.org','515.127.4561','1994-12-
07',14,11000.00,100,3);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)

 VALUES (115,'Alexander','Khoo','alexander.khoo@sqltutorial.org','515.127.4562','1995-05-
18',13,3100.00,114,3);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_i
d,department_id)
```

VALUES (116,'Shelli','Baida','shelli.baida@sqltutorial.org','515.127.4563','1997-12-24',13,2900.00,114,3);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (117,'Sigal','Tobias','sigal.tobias@sqltutorial.org','515.127.4564','1997-07-24',13,2800.00,114,3);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (118,'Guy','Himuro','guy.himuro@sqltutorial.org','515.127.4565','1998-11-15',13,2600.00,114,3);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (119,'Karen','Colmenares','karen.colmenares@sqltutorial.org','515.127.4566','1999-08-10',13,2500.00,114,3);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (120,'Matthew','Weiss','matthew.weiss@sqltutorial.org','650.123.1234','1996-07-18',19,8000.00,100,5);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (121,'Adam','Fripp','adam.fripp@sqltutorial.org','650.123.2234','1997-04-10',19,8200.00,100,5);

INSERT INTO employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

VALUES (122,'Payam','Kaufling','payam.kaufling@sqltutorial.org','650.123.3234','1995-05-01',19,7900.00,100,5);

```
INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (123,'Shanta','Vollman','shanta.vollman@sqltutorial.org','650.123.4234','1997-10-10',19,6500.00,100,5);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (126,'Irene','Mikkilineni','irene.mikkilineni@sqltutorial.org','650.124.1224','1998-09-28',18,2700.00,120,5);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (145,'John','Russell','john.russell@sqltutorial.org',NULL,'1996-10-01',15,14000.00,100,8);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (146,'Karen','Partners','karen.partners@sqltutorial.org',NULL,'1997-01-05',15,13500.00,100,8);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (176,'Jonathon','Taylor','jonathon.taylor@sqltutorial.org',NULL,'1998-03-24',16,8600.00,100,8);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (177,'Jack','Livingston','jack.livingston@sqltutorial.org',NULL,'1998-04-23',16,8400.00,100,8);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (178,'Kimberely','Grant','kimberely.grant@sqltutorial.org',NULL,'1999-05-24',16,7000.00,100,8);
```

```
INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (179,'Charles','Johnson','charles.johnson@sqltutorial.org',NULL,'2000-01-04',16,6200.00,100,8);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (192,'Sarah','Bell','sarah.bell@sqltutorial.org','650.501.1876','1996-02-04',17,4000.00,123,5);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (193,'Britney','Everett','britney.everett@sqltutorial.org','650.501.2876','1997-03-
03',17,3900.00,123,5);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (200,'Jennifer','Whalen','jennifer.whalen@sqltutorial.org','515.123.4444','1987-09-
17',3,4400.00,101,1);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (201,'Michael','Hartstein','michael.hartstein@sqltutorial.org','515.123.5555','1996-02-
17',10,13000.00,100,2);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (202,'Pat','Fay','pat.fay@sqltutorial.org','603.123.6666','1997-08-17',11,6000.00,201,2);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (203,'Susan','Mavris','susan.mavris@sqltutorial.org','515.123.7777','1994-06-
07',8,6500.00,101,4);
```

```
INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (204,'Hermann','Baer','hermann.baer@sqltutorial.org','515.123.8888','1994-06-07',12,10000.00,101,7);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (205,'Shelley','Higgins','shelley.higgins@sqltutorial.org','515.123.8080','1994-06-07',2,12000.00,101,11);

INSERT INTO
employees(employee_id,first_name,last_name,email,phone_number,hire_date,job_id,salary,manager_id,department_id)

 VALUES (206,'William','Gietz','william.gietz@sqltutorial.org','515.123.8181','1994-06-07',1,8300.00,205,11);
```

/*Data for the table dependents */

```
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)

VALUES (1,'Penelope','Gietz','Child',206);

INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)

VALUES (2,'Nick','Higgins','Child',205);

INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)

VALUES (3,'Ed','Whalen','Child',200);

INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)

VALUES (4,'Jennifer','King','Child',100);

INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)

VALUES (5,'Johnny','Kochhar','Child',101);

INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
```

```
VALUES (6,'Bette','De Haan','Child',102);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (7,'Grace','Faviet','Child',109);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (8,'Matthew','Chen','Child',110);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (9,'Joe','Sciarra','Child',111);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (10,'Christian','Urman','Child',112);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (11,'Zero','Popp','Child',113);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (12,'Karl','Greenberg','Child',108);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (13,'Uma','Mavris','Child',203);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (14,'Vivien','Hunold','Child',103);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (15,'Cuba','Ernst','Child',104);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
 VALUES (16,'Fred','Austin','Child',105);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (17,'Helen','Pataballa','Child',106);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
 VALUES (18,'Dan','Lorentz','Child',107);
```

```
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (19,'Bob','Hartstein','Child',201);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (20,'Lucille','Fay','Child',202);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
 VALUES (21,'Kirsten','Baer','Child',204);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (22,'Elvis','Khoo','Child',115);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (23,'Sandra','Baida','Child',116);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
 VALUES (24,'Cameron','Tobias','Child',117);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (25,'Kevin','Himuro','Child',118);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (26,'Rip','Colmenares','Child',119);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (27,'Julia','Raphaely','Child',114);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (28,'Woody','Russell','Child',145);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (29,'Alec','Partners','Child',146);
INSERT INTO dependents(dependent_id,first_name,last_name,relationship,employee_id)
VALUES (30,'Sandra','Taylor','Child',176);
```

Based on Above Database Solve the following Task:--

TASK 1:→

1)WRITE A QUERY FOR SELECT STATEMENTS :-

Syntax of SELECT STATEMENT:-

```
SELECT
  select_list
FROM
  table_name;
```

A. To get data from all the rows and columns in the employees table:
B. select data from the employee id, first name, last name, and hire date of all rows in the employees table:
C. to get the first name, last name, salary, and new salary:
D. Increase the salary two times and named as New_SALARY from employees table

2)WRITE A QUERY FOR ORDER BY STATEMENTS :-

Syntax of ORDER BY Statements:-

```
SELECT
   select_list
FROM
   table_name
ORDER BY
   sort_expression1 [ASC | DESC],
 sort_expression 2[ASC | DESC];
```

A. returns the data from the employee id, first name, last name, hire date, and salary column of the employees table:
B. to sort employees by first names in alphabetical order:
C. to sort the employees by the first name in ascending order and the last name in descending order:
D. to sort employees by salary from high to low:
E. to sort the employees by values in the hire_date column from:
F. sort the employees by the hire dates in descending order:

3)WRITE A QUERY FOR DISTINCT STATEMENTS :-

Syntax of DISTINCT Statements:-

```
SELECT DISTINCT
   column1, column2, ...
FROM
```

table1;

A. selects the salary data from the salary column of the employees table and sorts them from high to low:
B. select unique values from the salary column of the employees table:
C. selects the job id and salary from the employees table:
D. to remove the duplicate values in job id and salary:
E. returns the distinct phone numbers of employees:

4)WRITE A QUERY FOR TOP N STATEMENTS :-

Syntax of TOP N Statements(N=Will be any nos)

```
SELECT TOP N
  column_list
FROM
  table1
ORDER BY column_list
```

A. returns all rows in the employees table sorted by the first_name column.
B. to return the first 5 rows in the result set returned by the SELECT clause:
C. to return five rows starting from the 4th row:
D. gets the top five employees with the highest salaries.
E. to get employees who have the 2nd highest salary in the company

5)WRITE A QUERY FOR WHERE CLAUSE and COMPARISON OPERATORS :-

Syntax of WHERE CLAUSE and COMPARISON OPERATORS:--

```
SELECT
   column1, column2, ...
FROM
   table_name
WHERE
   condition;
```

The WHERE clause appears immediately after the FROM clause. The WHERE clause contains one or more logical expressions that evaluate each row in the table. If a row that causes the condition evaluates to true, it will be included in the result set; otherwise, it will be excluded.

Note that SQL has three-valued logic which are TRUE, FALSE, and UNKNOWN. It means that if a row causes the condition to evaluate to FALSE or NULL, the row will not be returned.

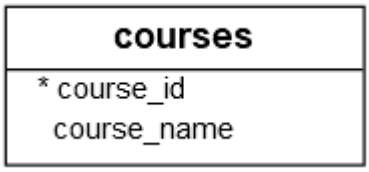
Note that the logical expression that follows the WHERE clause is also known as a predicate. You can use various operators to form the row selection criteria used in the WHERE clause.

Operator	**Meaning**
=	Equal to
<> (!=)	Not equal to
<	Less than
>	Greater than

Operator	Meaning
<=	Less than or equal
>=	Greater than or equal

A. query finds employees who have salaries greater than 14,000 and sorts the results sets based on the salary in descending order.
B. query finds all employees who work in the department id 5.
C. query finds the employee whose last name is Chen
D. To get all employees who joined the company after January 1st, 1999
E. to find the employees who joined the company in 1999,
F. statement finds the employee whose last name is Himuro
G. the query searches for the string Himuro in the last_name column of the employees table.
H. to find all employees who do not have phone numbers:
I. returns all employees whose department id is not 8.
J. finds all employees whose department id is not eight and ten.
K. to find the employees whose salary is greater than 10,000,
L. finds employees in department 8 and have the salary greater than 10,000:
M. the statement below returns all employees whose salaries are less than 10,000:
N. finds employees whose salaries are greater than or equal 9,000:
O. finds employees whose salaries are less than or equal to 9,000:

6)WRITE A QUERY FOR:-



A. adds a new column named credit_hours to the courses table.
B. adds the fee and max_limit columns to the courses table and places these columns after the course_name column.
C. changes the attribute of the fee column to NOT NULL.
D. to remove the fee column of the courses table
E. removes the max_limit and credit_hours of the courses table.

6)WRITE A QUERY FOR:-

SQL foreign key constraint

A foreign key is a column or a group of columns that enforces a link between the data in two tables. In a foreign key reference, the primary key column (or columns) of the first table is referenced by the column (or columns) of the second table. The column (or columns) of the second table becomes the foreign key.

You use the FOREIGN KEY constraint to create a foreign key when you create or alter table. Let's take a simple example to get a better understanding.

SQL FOREIGN KEY constraint examples

See the following projects and project_assignments tables:

```
CREATE TABLE projects (
  project_id INT PRIMARY KEY,
  project_name VARCHAR(255),
  start_date DATE NOT NULL,
  end_date DATE NOT NULL
);

CREATE TABLE project_milestones(
  milestone_id INT PRIMARY KEY,
  project_id  INT,
```

```
    milestone_name VARCHAR(100)
);
```

Each project may have zero or more milestones while one milestone must belong to one and only one project. The application that uses these tables must ensure that for each row in the project_milestones table there exists the corresponding row in the projects table. In other words, a milestone cannot exist without a project.

Unfortunately, users may edit the database using client tool or if there is a bug in the application, a row might be added to the project_milestones table that does not correspond to any row in the projects table. Or user may delete a row in the projects table, leaving orphaned rows in the project_milestones table. This causes the application not to work properly.

Write a Query

A. to add an SQL FOREIGN KEY constraint to the project_milestones table to enforce the relationship between the projects and project_milestones tables.

B. Suppose the project_milestones already exists without any predefined foreign key and you want to define a FOREIGN KEY constraint for the project_id column so write a Query to add a FOREIGN KEY constraint to existing table

TASK 2:→

Logical Operators and Special Operators

A logical operator allows you to test for the truth of a condition ,a logical operator returns a value of true, false, or unknown.

The following table illustrates the SQL logical operators:

Operator	**Meaning**
AND	Return true if both expressions are true

Operator	Meaning
NOT	Reverse the result of any other Boolean operator.
OR	Return true if either expression is true

OTHER SPECIAL OPERATORS:--

ANY	Return true if any one of the comparisons is true.
BETWEEN	Return true if the operand is within a range
EXISTS	Return true if a subquery contains any rows
IN	Return true if the operand is equal to one of the value in a list
LIKE	Return true if the operand matches a pattern
ALL	Return true if all comparisons are true

1)WRITE A QUERY FOR LOGICAL OPERATORS and OTHER ADVANCED OPERATORS:-

Part 1:-

A. finds all employees whose salaries are greater than 5,000 and less than 7,000:
B. finds employees whose salary is either 7,000 or 8,000:
C. finds all employees who do not have a phone number:
D. finds all employees whose salaries are between 9,000 and 12,000.
E. finds all employees who work in the department id 8 or 9.
F. finds all employees whose first name starts with the string jo
G. finds all employees with the first names whose the second character is h

H. finds all employees whose salaries are greater than all salaries of employees in the department 8:

Part 2:-

A. finds all employees whose salaries are greater than the average salary of every department:
B. finds all employees who have dependents:
C. to find all employees whose salaries are between 2,500 and 2,900:
D. to find all employees whose salaries are not in the range of 2,500 and 2,900:
E. to find all employees who joined the company between January 1, 1999, and December 31, 2000:
F. to find employees who have not joined the company from January 1, 1989 to December 31, 1999:
G. to find employees who joined the company between 1990 and 1993:

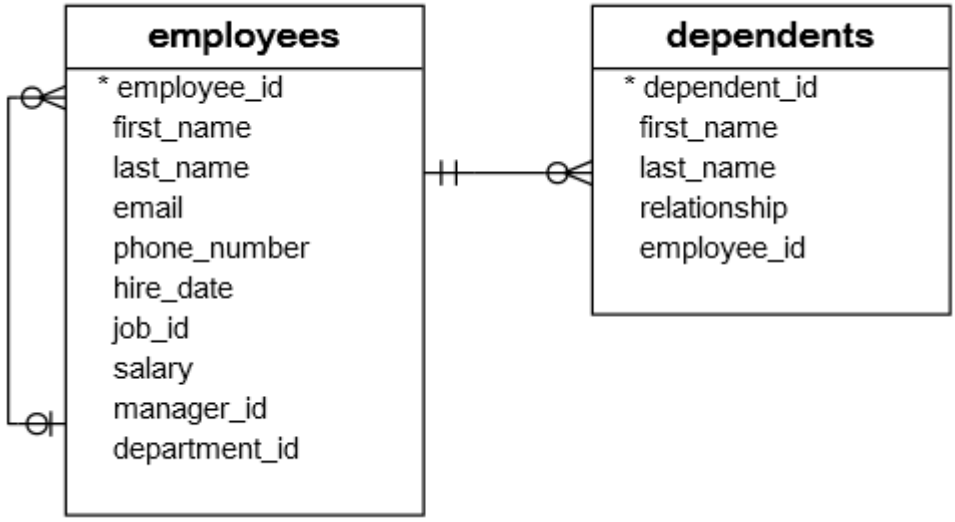
Part 3:-

A. to find all employees whose first names start with Da
B. to find all employees whose first names end with er
C. to find employees whose last names contain the word an:
D. retrieves employees whose first names start with Jo and are followed by at most 2 characters:
E. to find employees whose first names start with any number of characters and are followed by at most one character:
F. to find all employees whose first names start with the letter S but not start with Sh:

Part 4:-

A. retrieves all employees who work in the department id 5.
B. To get the employees who work in the department id 5 and with a salary not greater than 5000.
C. statement gets all the employees who are not working in the departments 1, 2, or 3.
D. retrieves all the employees whose first names do not start with the letter D.
E. to get employees whose salaries are not between 5,000 and 1,000.

Part 5:-

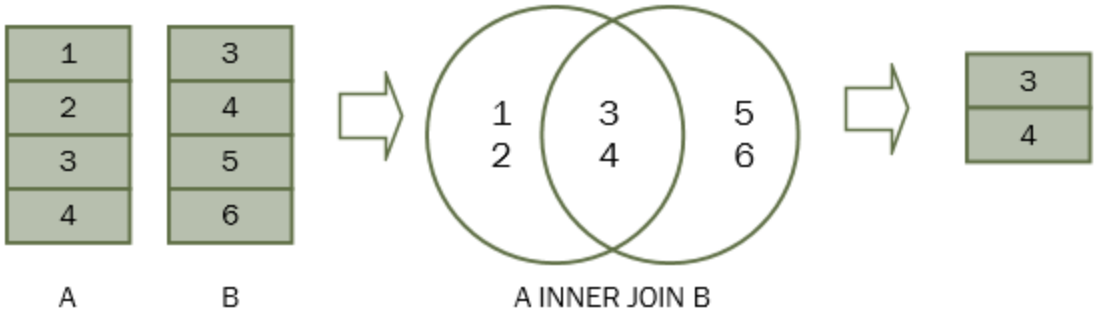


A. Write a query to get the employees who do not have any dependents by above image
B. To find all employees who do not have the phone numbers
C. To find all employees who have phone numbers

TASK 3:→

JOINS:-

SQL INNER JOIN clause



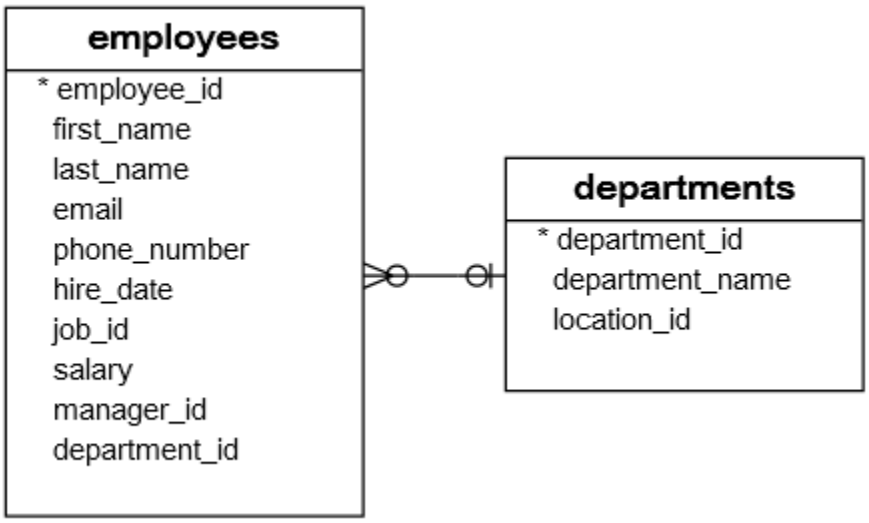
For each row in table A, the inner join clause finds the matching rows in table B. If a row is matched, it is included in the final result set.

Suppose the columns in the A and B tables are a and b. The following statement illustrates the inner join clause:

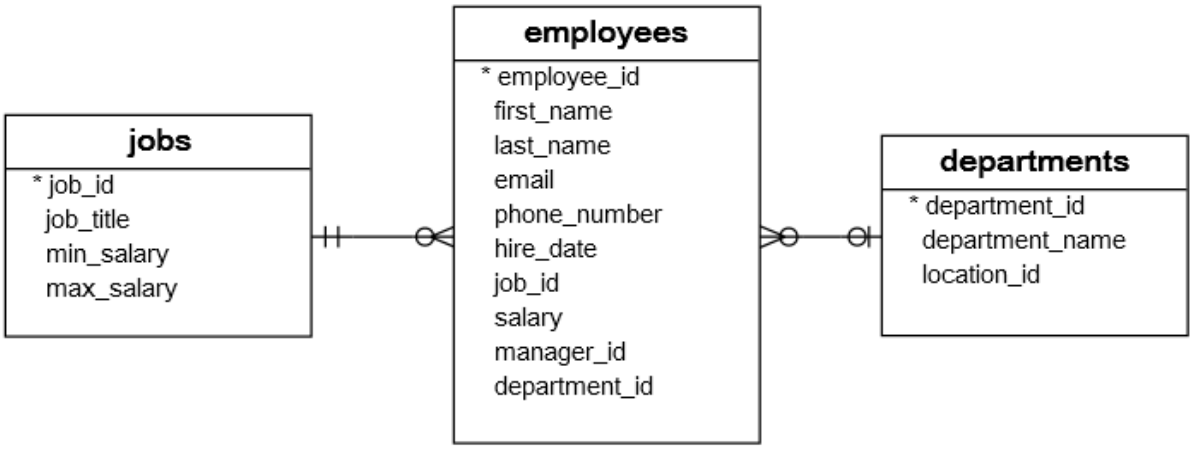
```
SELECT a

FROM A

INNER JOIN B ON b = a;
```

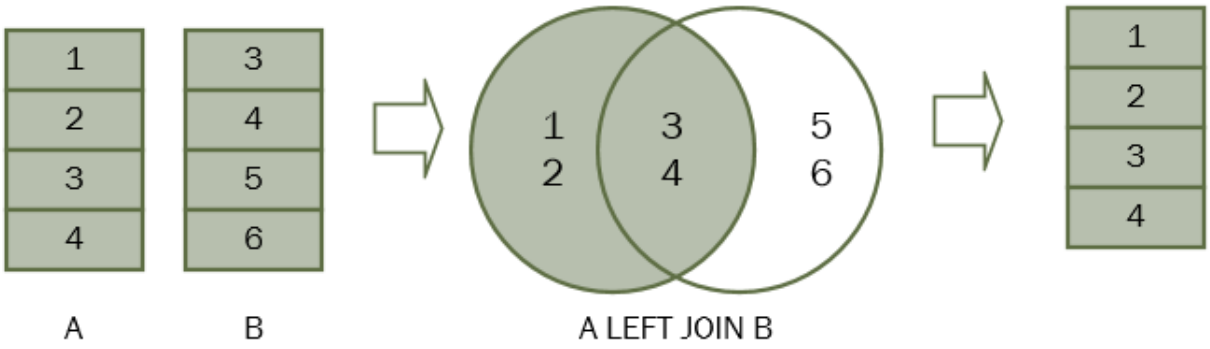


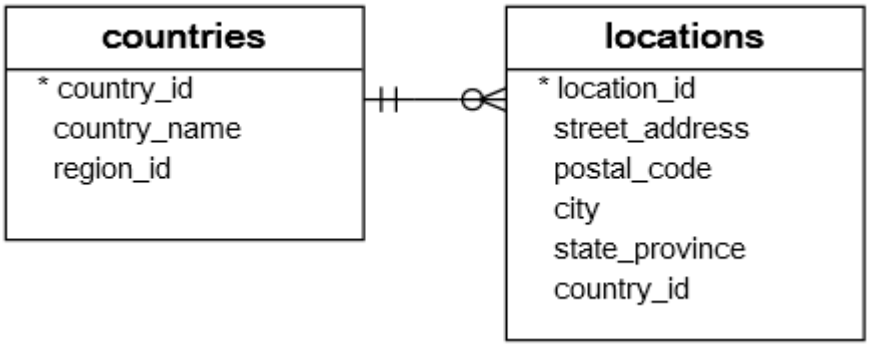
1) Write a Query to
   A. To get the information of the department id 1,2, and 3
   B. To get the information of employees who work in the department id 1, 2 and 3



Write a Query to get the first name, last name, job title, and department name of employees who work in department id 1, 2, and 3.

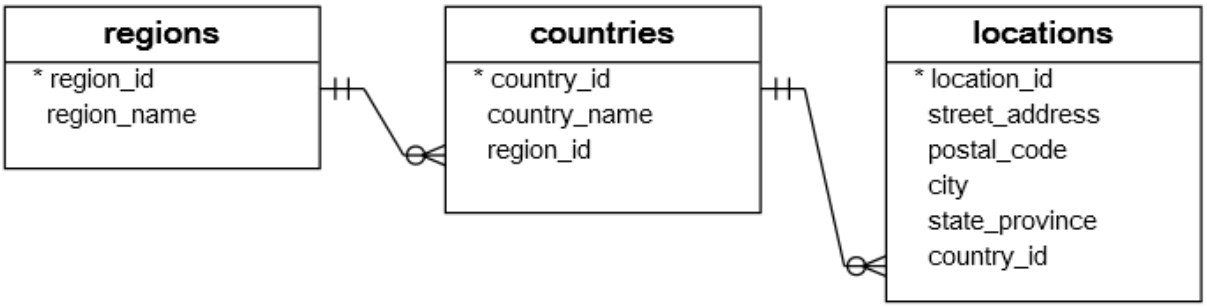
SQL LEFT JOIN clause





Write a Query :--

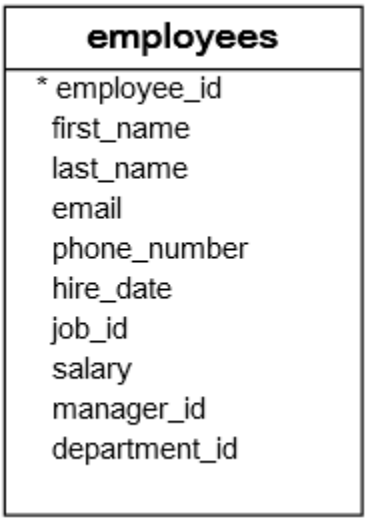
A. To query the country names of US, UK, and China
B. query retrieves the locations located in the US, UK and China:
C. To join the countries table with the locations table
D. to find the country that does not have any locations in the locations table



Write a query to join 3 tables: regions, countries, and locations

SQL self-join

```
SELECT
    column1,
    column2,
    column3,
     ...
FROM
    table1 A
INNER JOIN table1 B ON B.column1 = A.column2;
```



Questions:-

The manager_id column specifies the manager of an employee. Write a query statement to joins the employees table to itself to query the information of who reports to whom.

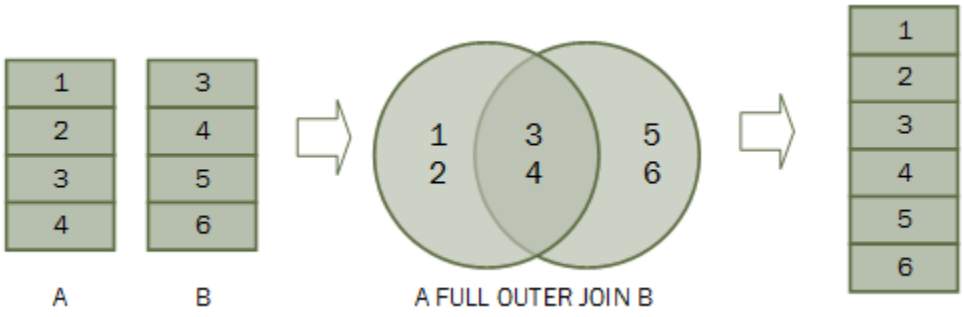
	employee	manager
▶	Bruce Ernst	Alexander Hunold
	David Austin	Alexander Hunold
	Valli Pataballa	Alexander Hunold
	Diana Lorentz	Alexander Hunold
	Alexander Khoo	Den Raphaely
	Shelli Baida	Den Raphaely
	Sigal Tobias	Den Raphaely
	Guy Himuro	Den Raphaely
	Karen Colmenares	Den Raphaely

The president does not have any manager. In the employees table, the manager_id of the row that contains the president is NULL.

Because the inner join clause only includes the rows that have matching rows in the other table, therefore the president did not show up in the result set of the query above.

Now write a Query To include the president in the result set:-

SQL FULL OUTER JOIN clause



Let's take an example of using the FULL OUTER JOIN clause to see how it works.

First, create two new tables: baskets and fruits for the demonstration. Each basket stores zero or more fruits and each fruit can be stored in zero or one basket.

```
CREATE  TABLE  fruits (
        fruit_id  INT PRIMARY KEY,
        fruit_name  VARCHAR (255) NOT NULL,
        basket_id  INTEGER
);
CREATE TABLE baskets (
        basket_id  INT PRIMARY KEY,
        basket_name  VARCHAR (255) NOT NULL
);
```

Second, insert some sample data into the baskets and fruits tables.

```
INSERT  INTO  baskets  (basket_id, basket_name)
VALUES
        (1, 'A'),
        (2, 'B'),
        (3, 'C');
```

```
INSERT  INTO  fruits (
        fruit_id,
        fruit_name,
        basket_id
)
VALUES
        (1, 'Apple', 1),
        (2, 'Orange', 1),
        (3, 'Banana', 2),
        (4, 'Strawberry', NULL);
```

Question:-

A. Write a query to returns each fruit that is in a basket and each basket that has a fruit, but also returns each fruit that is not in any basket and each basket that does not have any fruit.

B. Write a query to find the empty basket, which does not store any fruit
C. Write a query which fruit is not in any basket

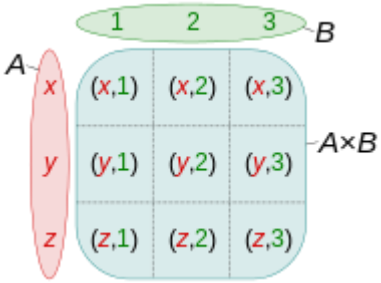
SQL CROSS JOIN clause

A cross join is a join operation that produces the Cartesian product of two or more tables.

In Math, a Cartesian product is a mathematical operation that returns a product set of multiple sets.

For example, with two sets A {x,y,z} and B {1,2,3}, the Cartesian product of A x B is the set of all ordered pairs (x,1), (x,2), (x,3), (y,1) (y,2), (y,3), (z,1), (z,2), (z,3).

The following picture illustrates the Cartesian product of A and B:



Similarly, in SQL, a Cartesian product of two tables A and B is a result set in which each row in the first table (A) is paired with each row in the second table (B). Suppose the A table has n rows and the B table has m rows, the result of the cross join of the A and B tables have n x m rows.

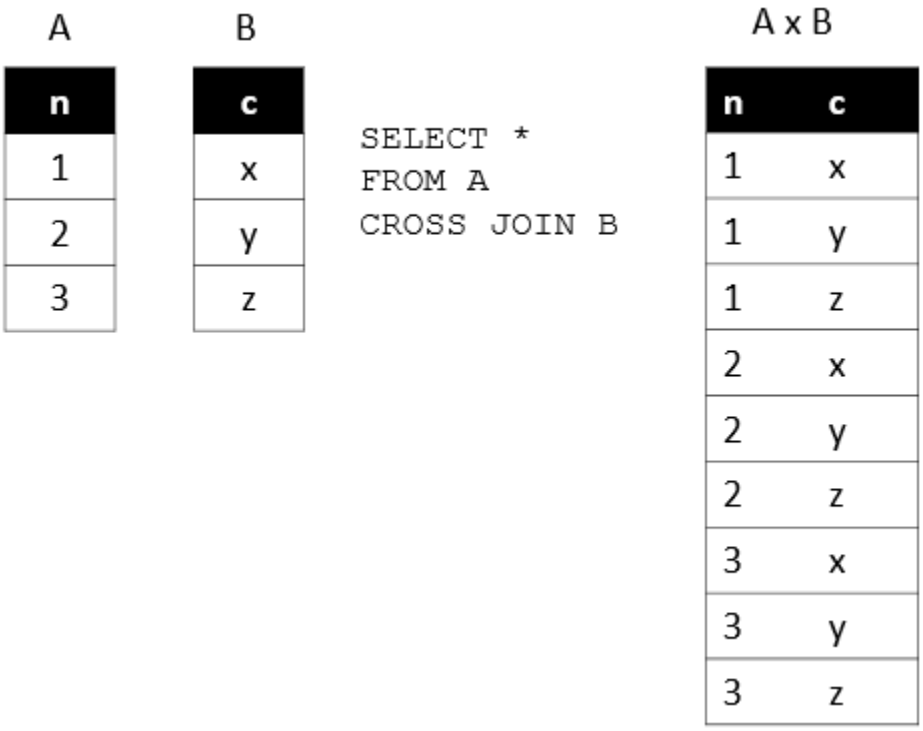
The following illustrates syntax of the CROSS JOIN clause:

```
SELECT column_list

FROM A

CROSS JOIN B;
```

The following picture illustrates the result of the cross join between the table A and table B. In this illustration, the table A has three rows 1, 2 and 3 and the table B also has three rows x, y and z. As the result, the Cartesian product has nine rows:



We will create two new tables for the demonstration of the cross join:

- sales_organization table stores the sale organizations.
- sales_channel table stores the sales channels.

The following statements create the sales_organization and sales_channel tables:

```
CREATE TABLE sales_organization (
        sales_org_id INT PRIMARY KEY,
        sales_org VARCHAR (255)
);

CREATE TABLE sales_channel (
        channel_id INT PRIMARY KEY,
        channel VARCHAR (255)
);
```

Suppose the company has two sales organizations that are Domestic and Export, which are in charge of sales in the domestic and international markets.

The following statement inserts two sales organizations into the sales_organization table:

```
INSERT INTO sales_organization (sales_org_id, sales_org)
VALUES
        (1, 'Domestic'),
        (2, 'Export');
```

The company can distribute goods via various channels such as wholesale, retail, eCommerce, and TV shopping. The following statement inserts sales channels into the sales_channel table:

```
INSERT INTO sales_channel (channel_id, channel)
VALUES
        (1, 'Wholesale'),
        (2, 'Retail'),
        (3, 'eCommerce'),
        (4, 'TV Shopping');
```

Question:--

Write a Query To find the all possible sales channels that a sales organization

TASK 4:→

SQL GROUP BY clause

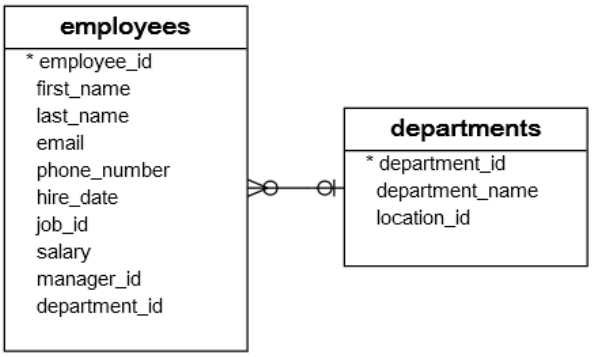
The GROUP BY is an optional clause of the SELECT statement. The GROUP BY clause allows you to group rows based on values of one or more columns. It returns one row for each group.

The following shows the basic syntax of the GROUP BY clause:

```
SELECT
        column1,
        column2,
        aggregate_function(column3)
FROM
        table_name
GROUP BY
        column1,
        column2;
```

In practice, you often use the GROUP BY clause with an aggregate function such as MIN, MAX, AVG, SUM, or COUNT to calculate a measure that provides the information for each group.

We will use the employees and departments tables to demonstrate how the GROUP BY clause works.



Questions:-

Write a Query

A. to group the values in department_id column of the employees table:
B. to count the number of employees by department:
C. returns the number of employees by department
D. to sort the departments by headcount:
E. to find departments with headcounts are greater than 5:
F. returns the minimum, maximum and average salary of employees in each department.
G. To get the total salary per department,
H. groups rows with the same values both department_id and job_id columns in the same group then return the rows for each of these groups

SQL HAVING clause

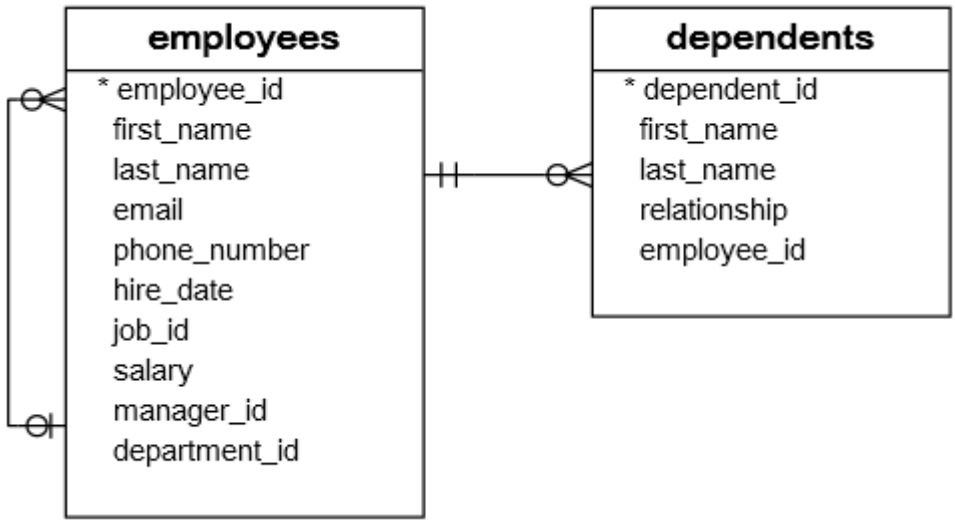
To specify a condition for groups, you use the HAVING clause.

The HAVING clause is often used with the GROUP BY clause in the SELECT statement. If you use a HAVING clause without a GROUP BY clause, the HAVING clause behaves like the WHERE clause

The following illustrates the syntax of the HAVING clause:

```
SELECT
```

```
    column1,
    column2,
    AGGREGATE_FUNCTION (column3)
FROM
    table1
GROUP BY
    column1,
    column2
HAVING
    group_condition;
```



Questions:-

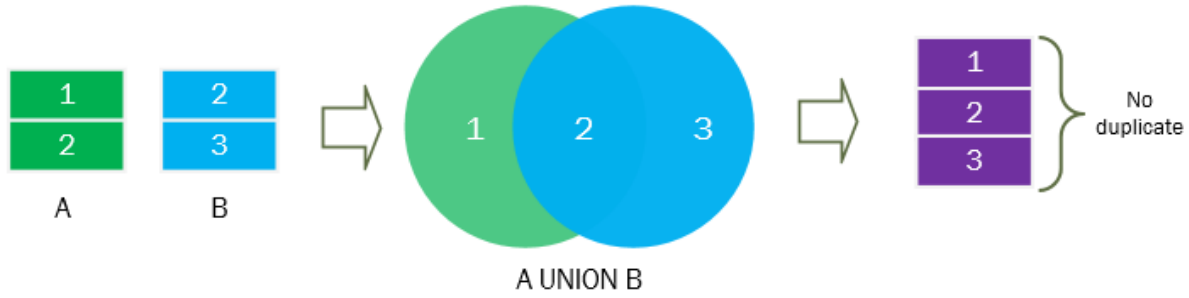
Write a Query

A. To get the managers and their direct reports, and to group employees by the managers and to count the direct reports.
B. To find the managers who have at least five direct reports
C. calculates the sum of salary that the company pays for each department and selects only the departments with the sum of salary between 20000 and 30000.
D. To find the department that has employees with the lowest salary greater than 10000
E. To find the departments that have the average salaries of employees between 5000 and 7000

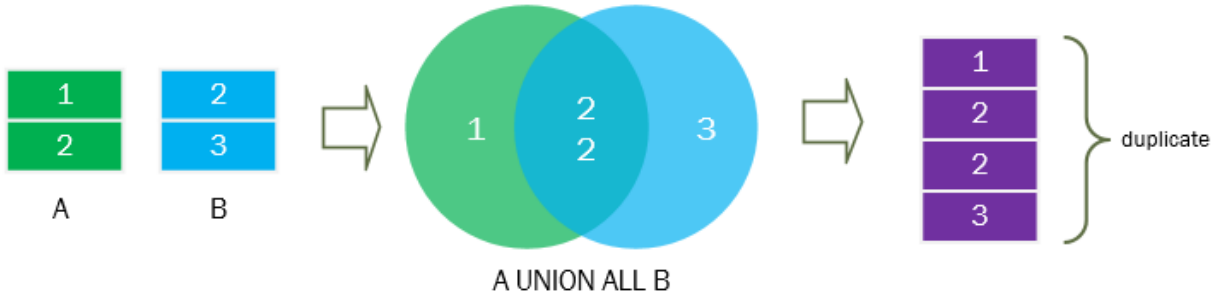
TASK 5 (Other Queries)

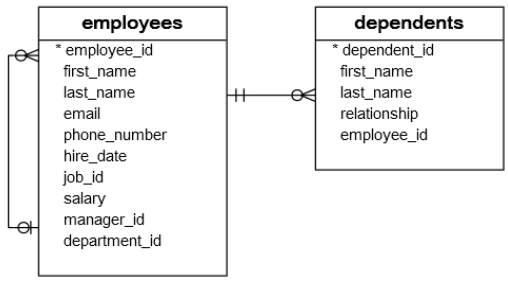
1)SQL UNION operator

Suppose, we have two result sets A(1,2) and B(2,3). The following picture illustrates A UNION B:



And the following picture illustrates A UNION ALL B





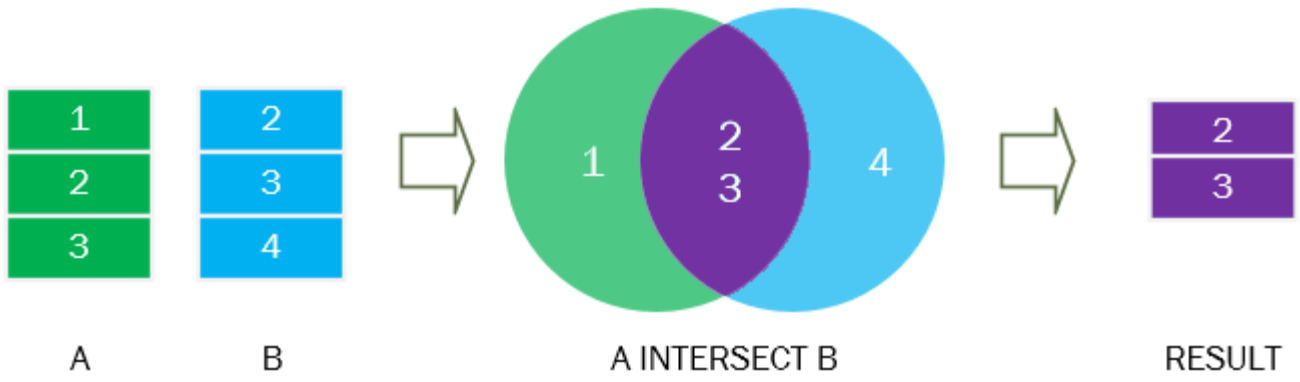
Quetsion:-

Write a Query to combine the first name and last name of employees and dependents

2)SQL INTERSECT operator

Suppose, we have two tables: A(1,2) and B(2,3).

The following picture illustrates the intersection of A & B tables.

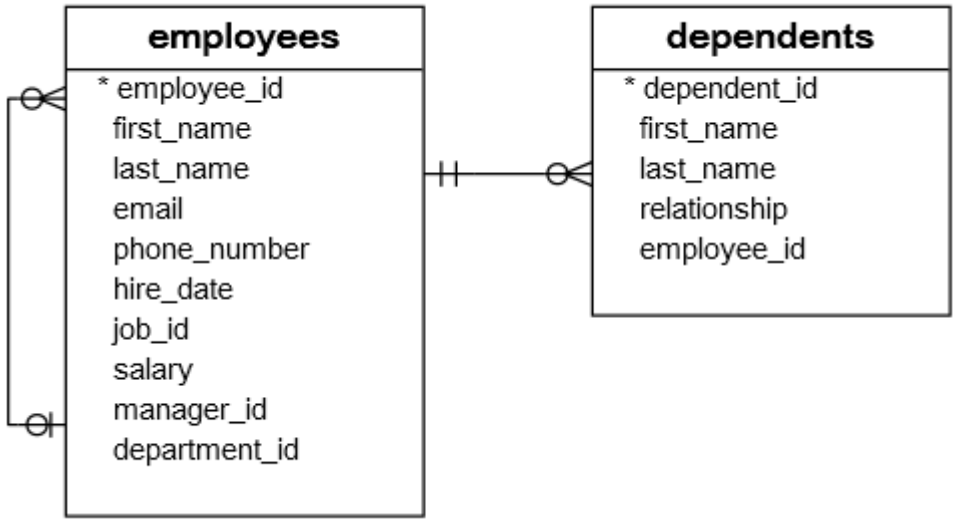


Question :-

Write a Query to Applies the INTERSECT operator to the A and B tables and sorts the combined result set by the id column in descending order.

3)SQL EXISTS operator

We will use the employees and dependents tables in the sample database for the demonstration.



Write a Query

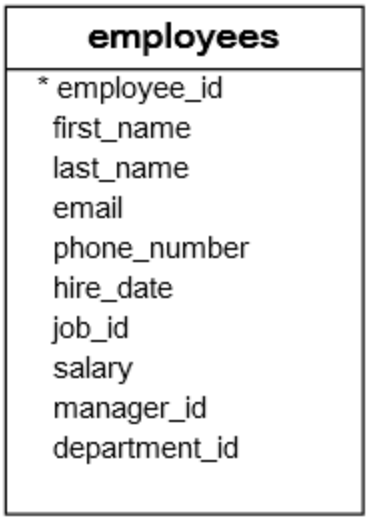
A. finds all employees who have at least one dependent.

B . finds employees who do not have any dependents:

4) SQL CASE expression

```
CASE expression
WHEN when_expression_1 THEN
        result_1
WHEN when_expression_2 THEN
        result_2
WHEN when_expression_3 THEN
        result_3
...
ELSE
        else_result
END
```

Let’s take a look at the employees table.

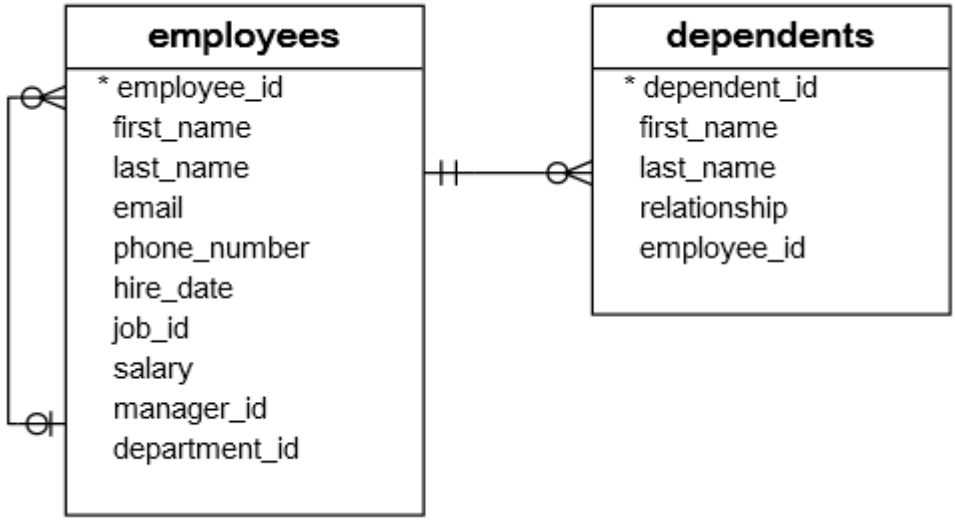


Questions:-

A. Suppose the current year is 2000. How to use the simple CASE expression to get the work anniversaries of employees by

B. Write a Query If the salary is less than 3000, the CASE expression returns “Low”. If the salary is between 3000 and 5000, it returns “average”. When the salary is greater than 5000, the CASE expression returns “High”.

5) SQL UPDATE statement



Suppose the employee id 192 Sarah Bell changed her last name from Bell to Lopez and you need to update her record in the employees table.

	employee_id	first_name	last_name
▶	192	Sarah	Bell

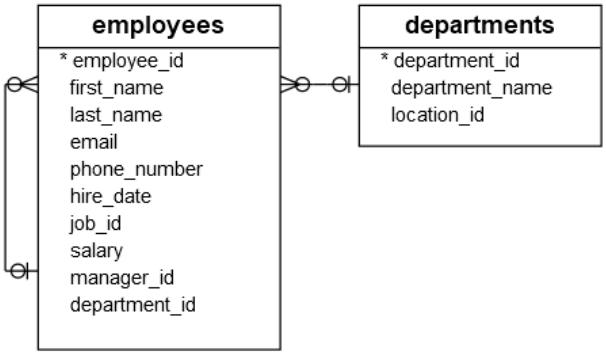
Write a Query to update Sarah's last name from Bell to Lopez

How to make sure that the last names of children are always matched with the last name of parents in the employees table,

FINAL TASK (Advanced Queries)

SQL SUBQUERY

Consider the following employees and departments tables from the sample database



Suppose you have to find all employees who locate in the location with the id 1700. You might come up with the following solution.

First, find all departments located at the location whose id is 1700:

```
SELECT
```

```
  *
FROM
  departments
WHERE
  location_id = 1700;
```

department_id	department_name	location_id
1	Administration	1700
3	Purchasing	1700
9	Executive	1700
10	Finance	1700
11	Accounting	1700

Second, find all employees that belong to the location 1700 by using the department id list of the previous query:

```
SELECT
  employee_id, first_name, last_name
FROM
  employees
WHERE
  department_id IN (1 , 3, 8, 10, 11)
ORDER BY first_name , last_name;
```

employee_id	first_name	last_name
115	Alexander	Khoo
179	Charles	Johnson
109	Daniel	Faviet
114	Den	Raphaely
118	Guy	Himuro
111	Ismael	Sciarra
177	Jack	Livingston
200	Jennifer	Whalen
110	John	Chen
145	John	Russell

This solution has two problems. To start with, you have looked at the departments table to check which department belongs to the location 1700. However, the original question was not referring to any specific departments; it referred to the location 1700.

Because of the small data volume, you can get a list of department easily

.

However, in the real system with high volume data, it might be problematic .Another problem was that you have to revise the queries whenever you want to find employees who locate in a different location

A much better solution to this problem is to use a subquery. By definition, a subquery is a query nested inside another query such as SELECT, INSERT, UPDATE, or DELETE statement. In this tutorial, we are focusing on the subquery used with the SELECT statement.

Question:-

Write a Query :-

A. *Combine Above two queries using subquery inorder find all departments located at the location whose id is 1700 and find all employees that belong to the location 1700 by using the department id list of the previous query*

B. to find all employees who do not locate at the location 1700:

C. finds the employees who have the highest salary:

D. finds all employees who salaries are greater than the average salary of all employees:

E. finds all departments which have at least one employee with the salary is greater than 10,000:

F. finds all departments that do not have any employee with the salary greater than 10,000:

G. to find the lowest salary by department:

H. finds all employees whose salaries are greater than the lowest salary of every department:
I. finds all employees whose salaries are greater than or equal to the highest salary of every department

J. returns the average salary of every department

K. to calculate the average of average salary of departments :

L. finds the salaries of all employees, their average salary, and the difference between the salary of each employee and the average salary.

THANK YOU