HR DATABASE MANAGEMENT SYSTEM

BY

SACHIN GUNJAL

INTRODUCTION

The **HR Database Management System (HRDMS)** project is a comprehensive initiative designed to model and implement an adaptable, enterprise-grade solution for managing human resources data. Harnessing the capabilities of SQL and MS SQL Server, the HRDMS demonstrates best practices in relational database design, efficient data manipulation, and robust reporting. The project replicates the operational complexity of large organizations by integrating data across geographies, departments, job roles, and employee dependents.

PROJECT SCOPE AND OBJECTIVES

The scope encompasses every stage of the database lifecycle, including:

- **Schema Design:** Creating a fully normalized database that mitigates data redundancy and ensures referential integrity.
- **Data Modeling:** Defining clear relationships between core HR entities such as employees, jobs, departments, dependents, locations, countries, and regions.
- **Implementation:** Translating the data model into efficient SQL code for table creation, constraints, and population with realistic datasets.
- **Query Development:** Building and refining a variety of SQL queries, from basic retrievals to advanced data aggregations, to simulate HR analytics and operational reports.
- **Maintenance Activities:** Handling updates, data consistency enforcement, and schema changes as typical in a real-world HRIS environment1.

KEY FEATURES

- **Multi-Table Structure:** The system consists of seven interconnected tables, each responsible for a specific aspect of HR data management:
  - Employees
  - Jobs
  - Departments
  - Dependents
  - Locations
  - Countries
  - Regions
- **Referential Integrity:** Strategic usage of primary and foreign keys preserves the integrity of organizational data and supports business rules such as cascading updates and deletes.
- **Comprehensive Data Operations:** The platform supports a wide range of SQL operations, including:
  - Data creation (DDL and DML operations)
  - Complex SELECT statements with JOINs, subqueries, filtering, sorting, and grouping
  - Advanced queries for departmental analysis, salary trends, and managerial hierarchies
  - Aggregation and reporting to support HR decision-making
- **Scalability and Flexibility:** The HRDMS is designed to accommodate organizational growth (e.g., new regions, departments, or employees) with minimal impact on the existing database structure, ensuring long-term usability and adaptability.
- **Data Security and Consistency:** Implementation of constraints ensures only valid data is stored, reducing errors and enhancing the reliability of reports extracted from the system.

REAL-WORLD RELEVANCE

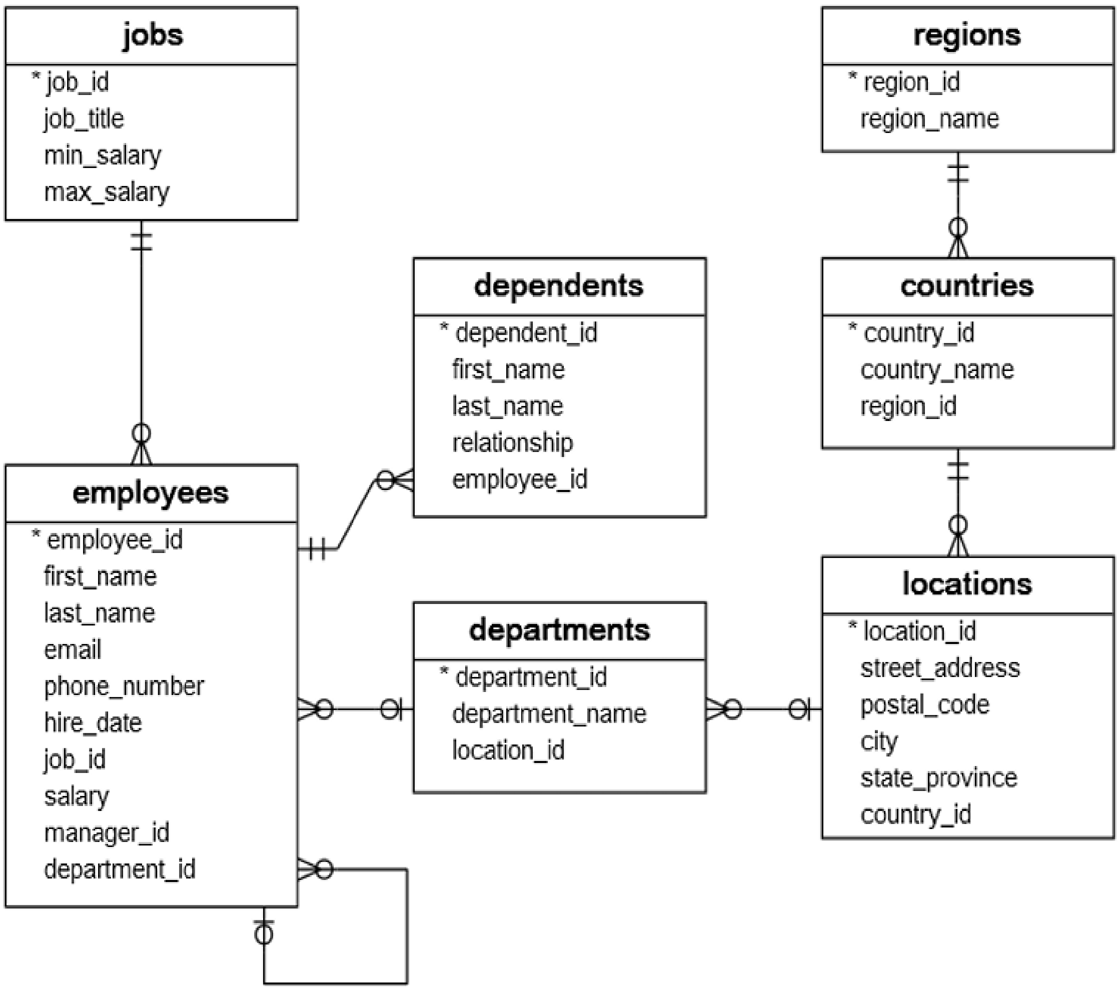
Through this project, users gain exposure to novel HR data challenges encountered in medium to large organizations, such as:

- Tracking and managing employee information in global, multi-site environments
- Supporting organizational hierarchy with manager-employee relationships (self-joins)
- Monitoring dependents for employee benefits administration
- Empowering HR departments with reliable analytics, trend identification, and automated reporting for informed strategic planning

TECHNOLOGIES USED

- **SQL (Structured Query Language)**
- **MS SQL Server Management Studio (SSMS)**

ER DIAGRAM


jobs
* job_id
job_title
min_salary
max_salary
regions
* region_id
region_name
dependents
* dependent_id
first_name
last_name
relationship
employee_id
countries
* country_id
country_name
region_id
employees
* employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
manager_id
department_id
departments
* department_id
department_name
location_id
locations
* location_id
street_address
postal_code
city
state_province
country_id

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.

CREATE TABLE AND INSERT DATA

```
CREATE DATABASE HR_Management_System_1
```

REGIONS TABLE:

```
CREATE TABLE Regions(

Region_Id INT PRIMARY KEY NOT NULL,

Region_Name VARCHAR(50) DEFAULT NULL)


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');
```

COUNTRY TABLE:

```
CREATE TABLE Countries(

Country_Id CHAR(3) PRIMARY KEY NOT NULL,

Country_Name VARCHAR(50) DEFAULT NULL,
Region_Id INT NOT NULL,

FOREIGN KEY (Region_Id) REFERENCES Regions(Region_Id)

ON DELETE CASCADE ON UPDATE CASCADE)


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);
```

LOCATIONS TABLE:

```
CREATE TABLE Locations(

Location_ID INT PRIMARY KEY NOT NULL,

Street_Address VARCHAR(100) DEFAULT NULL,

Postal_code VARCHAR(10) DEFAULT NULL,

City VARCHAR(15) NOT NULL,

State_Province VARCHAR(20) DEFAULT NULL,

Country_Id CHAR(3) NOT NULL,

FOREIGN KEY (Country_Id) REFERENCES Countries(Country_Id)

ON DELETE CASCADE ON UPDATE CASCADE

)


ALTER TABLE Locations

ALTER Column City VARCHAR(30);


INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_id)

VALUES (1400,'2014 Jabberwocky Rd','26192','Southlake','Texas','US');

INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_id)

VALUES (1500,'2011 Interiors Blvd','99236','South San
Francisco','California','US');

INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_id)

VALUES (1700,'2004 Charade Rd','98199','Seattle','Washington','US');

INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_id)
```

```
VALUES (1800,'147 Spadina Ave','M5V 2L7','Toronto','Ontario','CA');

INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_id)

VALUES (2400,'8204 Arthur St',NULL,'London',NULL,'UK');

INSERT INTO
locations(location_id,street_address,postal_code,city,state_province,co
untry_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,co
untry_id)

VALUES (2700,'Schwanthalerstr. 7031','80925','Munich','Bavaria','DE');


SELECT * from Locations
```

DEPARTMENT TABLE:

```
CREATE TABLE Departments(

Department_ID INT PRIMARY KEY NOT NULL,

Department_Name VARCHAR(50) DEFAULT NULL,

Location_ID INT NOT NULL,

FOREIGN KEY (Location_ID) REFERENCES Locations(Location_ID)

ON DELETE CASCADE ON UPDATE CASCADE

)


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);
```

JOBS TABLE:

```
CREATE TABLE Jobs(
Job_ID INT PRIMARY KEY NOT NULL,
Job_Title VARCHAR(30) DEFAULT NULL,
Min_Salary DECIMAL(8,2) DEFAULT NULL,
Max_Salary DECIMAL(8,2) DEFAULT NULL)
```

```
ALTER TABLE JOBS

ALTER COLUMN Job_Title VARCHAR(50)


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);
```

EMPLOYEES TABLE:

```
CREATE TABLE Employees(
Employee_ID INT PRIMARY KEY NOT NULL,
First_Name VARCHAR(25) DEFAULT NULL,
Last_Name VARCHAR(25) NOT NULL,
Email VARCHAR(25) 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 NOT 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)

)


ALTER TABLE EMPLOYEEs

ALTER COLUMN Email VARCHAR(100)


ALTER TABLE Employees

ALTER COLUMN Manager_Id INT DEFAULT NULL;


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','590.423.4
567','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','19
97-06-25',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','590.423.456
0','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','515.124.456
9','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_id,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_id,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_id,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_id,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_id,department_id)

VALUES
(114,'Den','Raphaely','den.raphaely@sqltutorial.org','515.127.4561','19
94-12-07',14,11000.00,100,3);

INSERT INTO

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

VALUES
(116,'Shelli','Baida','shelli.baida@sqltutorial.org','515.127.4563','19
97-12-24',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','19
97-07-24',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','515.127.4
566','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','650.501.287
6','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','515.123.444
4','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','19
94-06-07',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','19
94-06-07',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','515.123.808
0','1994-06-07',2,12000.00,101,11);
```

DEPENDENTS TABLE:

```
CREATE TABLE Dependents(

Dependent_Id INT PRIMARY KEY NOT NULL,

First_Name VARCHAR(25) DEFAULT NULL,

Last_Name VARCHAR(25) DEFAULT NULL,

Relationship VARCHAR(25) DEFAULT NULL,

Employee_id INT NOT NULL,

FOREIGN KEY (employee_id) REFERENCES employees (employee_id)

ON DELETE CASCADE ON UPDATE CASCADE)


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

VALUES (1,'Penelope','Gietz','Child',120);

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)
```

```sql
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);


SELECT * FROM Employees

SELECT * FROM Dependents

SELECT * FROM  Departments

SELECT * FROM Locations

SELECT * FROM Regions

SELECT * FROM Countries

SELECT * FROM Jobs
```

TASK 1

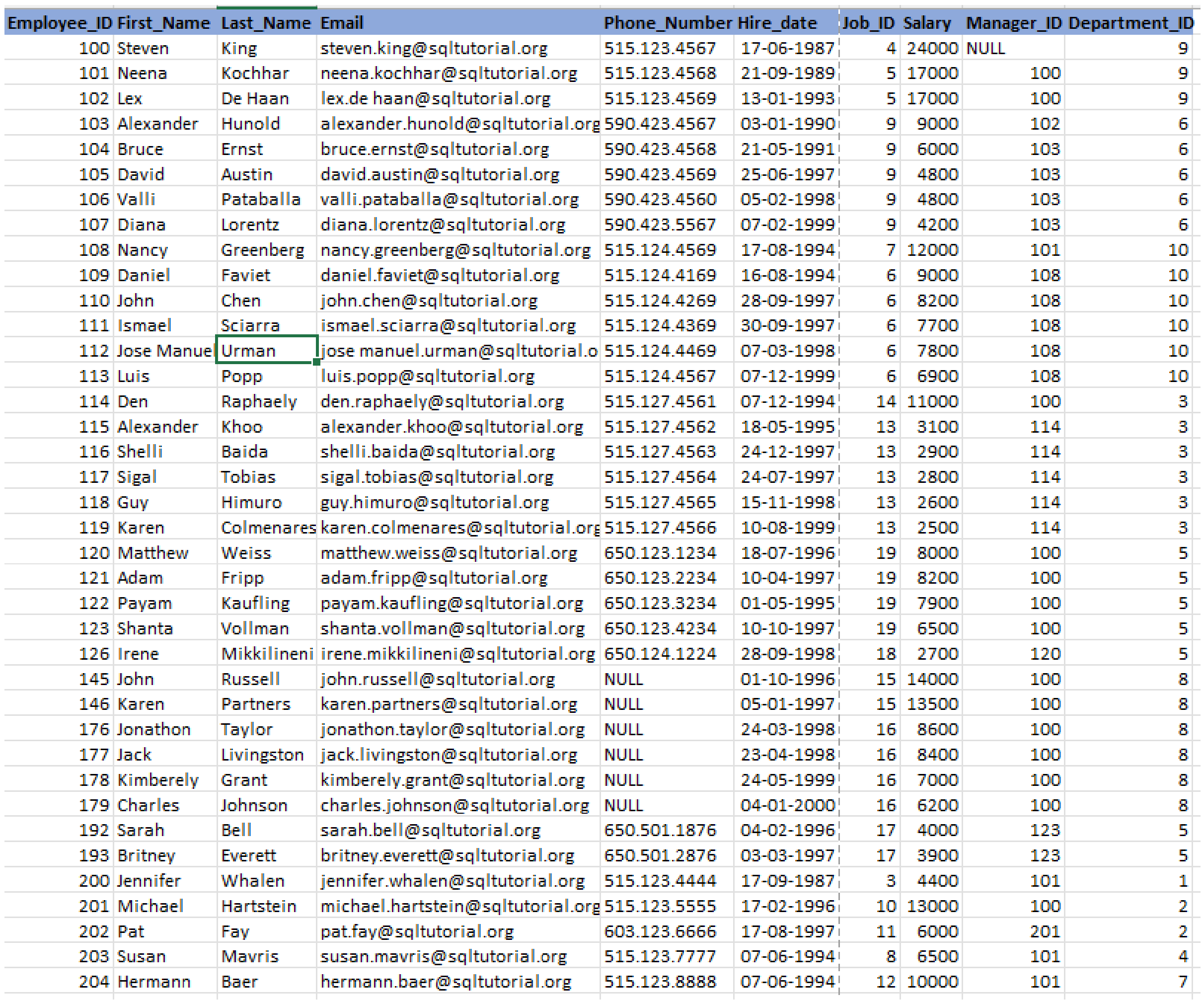
1)WRITE A QUERY FOR SELECT STATEMENTS :-

Syntax of SELECT STATEMENT:-

```
SELECT
        select_list
FROM
        table_name;
```

A. To get data from all the rows and columns in the employees table:

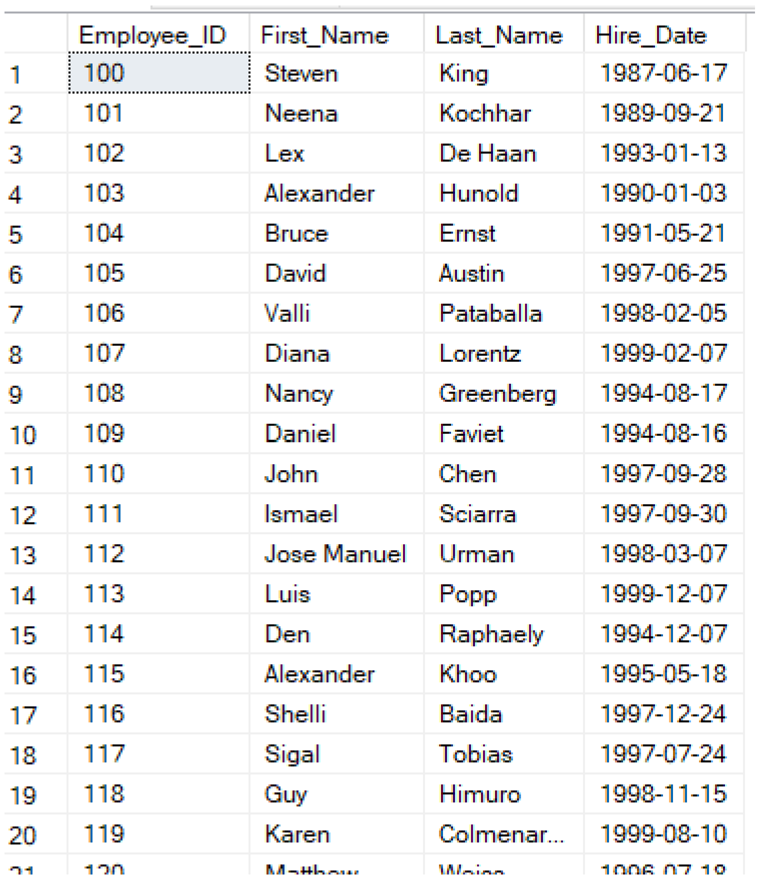
```
SELECT * FROM Employees
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_date	Job_ID	Salary	Manager_ID	Department_ID
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	17-06-1987	4	24000	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	21-09-1989	5	17000	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	13-01-1993	5	17000	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	03-01-1990	9	9000	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	21-05-1991	9	6000	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	25-06-1997	9	4800	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	05-02-1998	9	4800	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	07-02-1999	9	4200	103	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	17-08-1994	7	12000	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	16-08-1994	6	9000	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	28-09-1997	6	8200	108	10
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	30-09-1997	6	7700	108	10
112	Jose Manue	Urman	jose manuel.urman@sqltutorial.o	515.124.4469	07-03-1998	6	7800	108	10
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	07-12-1999	6	6900	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	07-12-1994	14	11000	100	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	18-05-1995	13	3100	114	3
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	24-12-1997	13	2900	114	3
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	24-07-1997	13	2800	114	3
118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	15-11-1998	13	2600	114	3
119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	10-08-1999	13	2500	114	3
120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	18-07-1996	19	8000	100	5
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	10-04-1997	19	8200	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	01-05-1995	19	7900	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	10-10-1997	19	6500	100	5
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	28-09-1998	18	2700	120	5
145	John	Russell	john.russell@sqltutorial.org	NULL	01-10-1996	15	14000	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	05-01-1997	15	13500	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	24-03-1998	16	8600	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	23-04-1998	16	8400	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	24-05-1999	16	7000	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	04-01-2000	16	6200	100	8
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	04-02-1996	17	4000	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	03-03-1997	17	3900	123	5
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	17-09-1987	3	4400	101	1
201	Michael	Hartstein	michael.hartstein@sqltutorial.org	515.123.5555	17-02-1996	10	13000	100	2
202	Pat	Fay	pat.fay@sqltutorial.org	603.123.6666	17-08-1997	11	6000	201	2
203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	07-06-1994	8	6500	101	4
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	07-06-1994	12	10000	101	7

B. select data from the employee id, first name, last name, and hire date of all rows in the employees table:

```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date FROM Employees
```

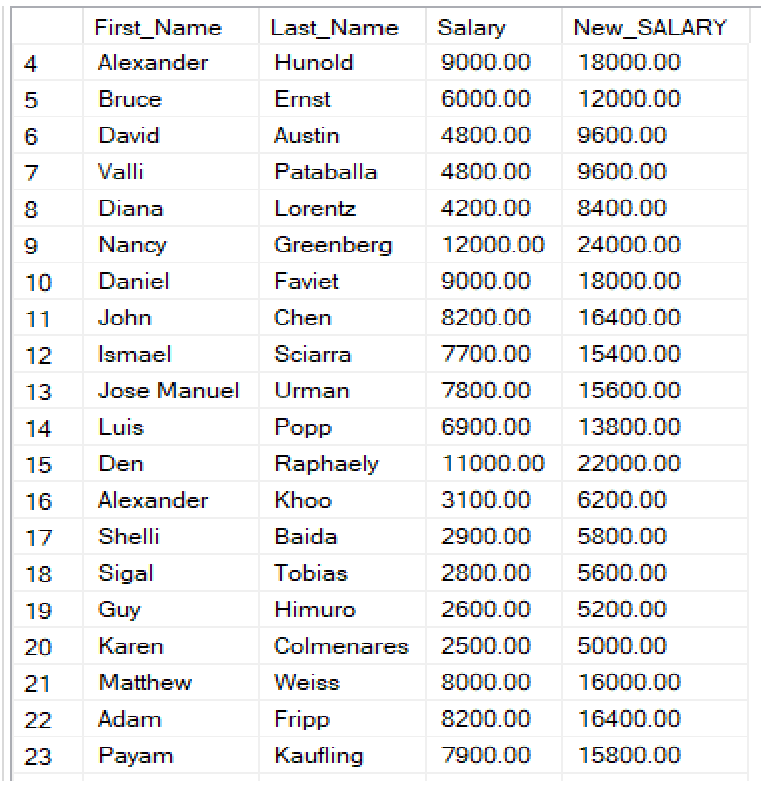


	Employee_ID	First_Name	Last_Name	Hire_Date
1	100	Steven	King	1987-06-17
2	101	Neena	Kochhar	1989-09-21
3	102	Lex	De Haan	1993-01-13
4	103	Alexander	Hunold	1990-01-03
5	104	Bruce	Ernst	1991-05-21
6	105	David	Austin	1997-06-25
7	106	Valli	Pataballa	1998-02-05
8	107	Diana	Lorentz	1999-02-07
9	108	Nancy	Greenberg	1994-08-17
10	109	Daniel	Faviet	1994-08-16
11	110	John	Chen	1997-09-28
12	111	Ismael	Sciarra	1997-09-30
13	112	Jose Manuel	Urman	1998-03-07
14	113	Luis	Popp	1999-12-07
15	114	Den	Raphaely	1994-12-07
16	115	Alexander	Khoo	1995-05-18
17	116	Shelli	Baida	1997-12-24
18	117	Sigal	Tobias	1997-07-24
19	118	Guy	Himuro	1998-11-15
20	119	Karen	Colmenar...	1999-08-10
21	120	Matthew	Weiss	1996-07-18

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

```
SELECT First_Name, Last_Name, Salary, Salary*2 as New_SALARY FROM
Employees
```



	First_Name	Last_Name	Salary	New_SALARY
4	Alexander	Hunold	9000.00	18000.00
5	Bruce	Ernst	6000.00	12000.00
6	David	Austin	4800.00	9600.00
7	Valli	Pataballa	4800.00	9600.00
8	Diana	Lorentz	4200.00	8400.00
9	Nancy	Greenberg	12000.00	24000.00
10	Daniel	Faviet	9000.00	18000.00
11	John	Chen	8200.00	16400.00
12	Ismael	Sciarra	7700.00	15400.00
13	Jose Manuel	Urman	7800.00	15600.00
14	Luis	Popp	6900.00	13800.00
15	Den	Raphaely	11000.00	22000.00
16	Alexander	Khoo	3100.00	6200.00
17	Shelli	Baida	2900.00	5800.00
18	Sigal	Tobias	2800.00	5600.00
19	Guy	Himuro	2600.00	5200.00
20	Karen	Colmenares	2500.00	5000.00
21	Matthew	Weiss	8000.00	16000.00
22	Adam	Fripp	8200.00	16400.00
23	Payam	Kaufling	7900.00	15800.00

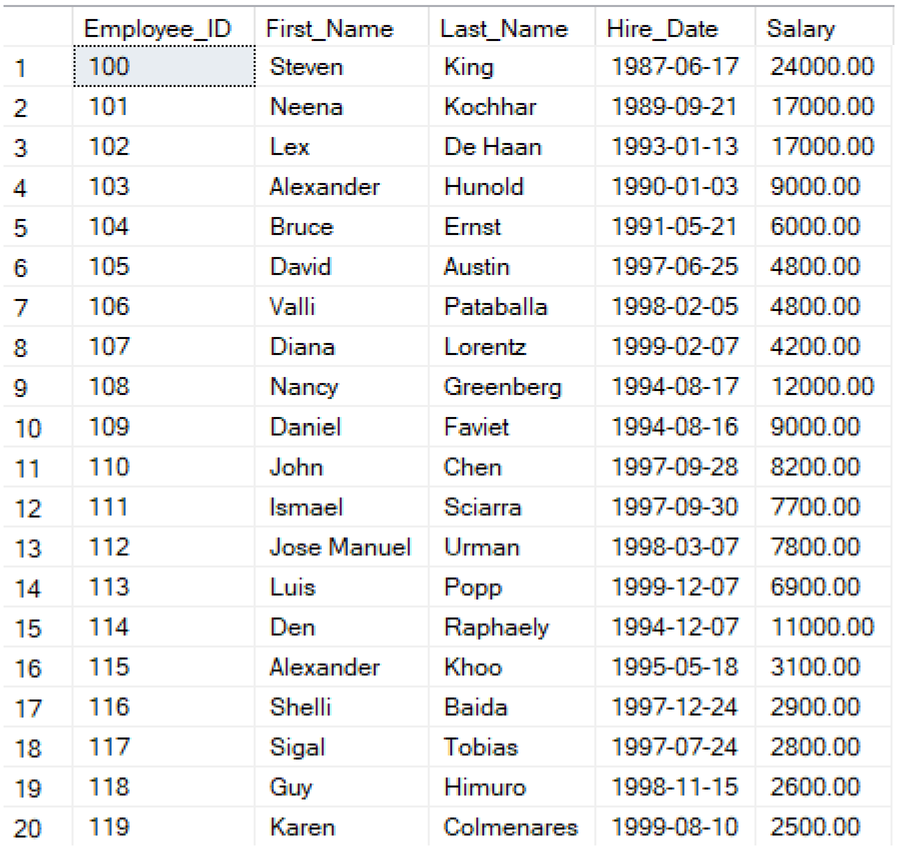
2)WRITE A QUERY FOR ORDER BY STATEMENTS :-

Syntax of ORDER BY Statements:-

```
SELECT
        select_list
FROM
        table_name
ORDER BY
        sort_expression1 [ASC | DESC],
        _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:

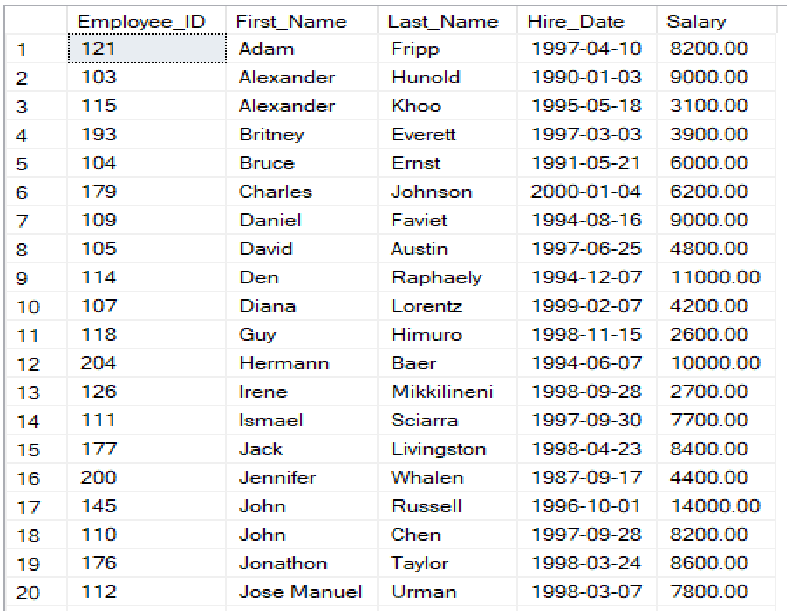
```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary
FROM Employees
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	100	Steven	King	1987-06-17	24000.00
2	101	Neena	Kochhar	1989-09-21	17000.00
3	102	Lex	De Haan	1993-01-13	17000.00
4	103	Alexander	Hunold	1990-01-03	9000.00
5	104	Bruce	Ernst	1991-05-21	6000.00
6	105	David	Austin	1997-06-25	4800.00
7	106	Valli	Pataballa	1998-02-05	4800.00
8	107	Diana	Lorentz	1999-02-07	4200.00
9	108	Nancy	Greenberg	1994-08-17	12000.00
10	109	Daniel	Faviet	1994-08-16	9000.00
11	110	John	Chen	1997-09-28	8200.00
12	111	Ismael	Sciarra	1997-09-30	7700.00
13	112	Jose Manuel	Urman	1998-03-07	7800.00
14	113	Luis	Popp	1999-12-07	6900.00
15	114	Den	Raphaely	1994-12-07	11000.00
16	115	Alexander	Khoo	1995-05-18	3100.00
17	116	Shelli	Baida	1997-12-24	2900.00
18	117	Sigal	Tobias	1997-07-24	2800.00
19	118	Guy	Himuro	1998-11-15	2600.00
20	119	Karen	Colmenares	1999-08-10	2500.00

B. to sort employees by first names in alphabetical order:

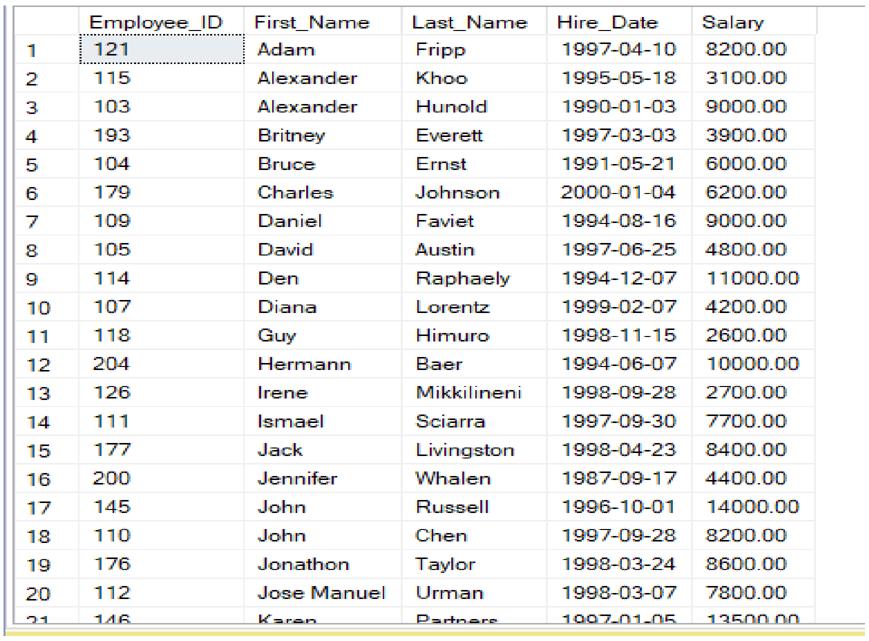
```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary
 FROM Employees
 ORDER BY First_Name
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	121	Adam	Fripp	1997-04-10	8200.00
2	103	Alexander	Hunold	1990-01-03	9000.00
3	115	Alexander	Khoo	1995-05-18	3100.00
4	193	Britney	Everett	1997-03-03	3900.00
5	104	Bruce	Ernst	1991-05-21	6000.00
6	179	Charles	Johnson	2000-01-04	6200.00
7	109	Daniel	Faviet	1994-08-16	9000.00
8	105	David	Austin	1997-06-25	4800.00
9	114	Den	Raphaely	1994-12-07	11000.00
10	107	Diana	Lorentz	1999-02-07	4200.00
11	118	Guy	Himuro	1998-11-15	2600.00
12	204	Hermann	Baer	1994-06-07	10000.00
13	126	Irene	Mikkilineni	1998-09-28	2700.00
14	111	Ismael	Sciarra	1997-09-30	7700.00
15	177	Jack	Livingston	1998-04-23	8400.00
16	200	Jennifer	Whalen	1987-09-17	4400.00
17	145	John	Russell	1996-10-01	14000.00
18	110	John	Chen	1997-09-28	8200.00
19	176	Jonathon	Taylor	1998-03-24	8600.00
20	112	Jose Manuel	Urman	1998-03-07	7800.00

C. to sort the employees by the first name in ascending order and the last name in descending order:

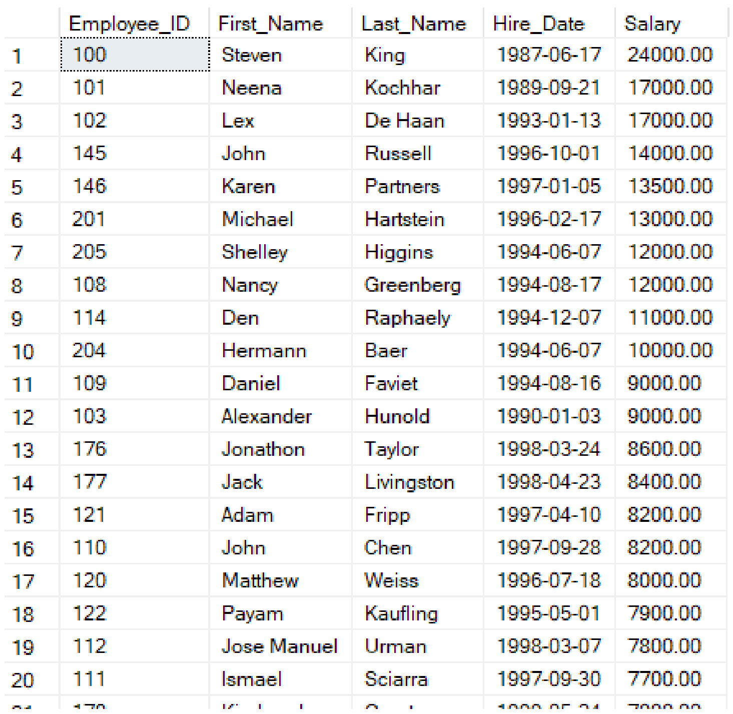
```
 SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary

 FROM Employees

 ORDER BY First_Name ASC, Last_Name DESC
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	121	Adam	Fripp	1997-04-10	8200.00
2	115	Alexander	Khoo	1995-05-18	3100.00
3	103	Alexander	Hunold	1990-01-03	9000.00
4	193	Britney	Everett	1997-03-03	3900.00
5	104	Bruce	Ernst	1991-05-21	6000.00
6	179	Charles	Johnson	2000-01-04	6200.00
7	109	Daniel	Faviet	1994-08-16	9000.00
8	105	David	Austin	1997-06-25	4800.00
9	114	Den	Raphaely	1994-12-07	11000.00
10	107	Diana	Lorentz	1999-02-07	4200.00
11	118	Guy	Himuro	1998-11-15	2600.00
12	204	Hermann	Baer	1994-06-07	10000.00
13	126	Irene	Mikkilineni	1998-09-28	2700.00
14	111	Ismael	Sciarra	1997-09-30	7700.00
15	177	Jack	Livingston	1998-04-23	8400.00
16	200	Jennifer	Whalen	1987-09-17	4400.00
17	145	John	Russell	1996-10-01	14000.00
18	110	John	Chen	1997-09-28	8200.00
19	176	Jonathon	Taylor	1998-03-24	8600.00
20	112	Jose Manuel	Urman	1998-03-07	7800.00
21	146	Karen	Partners	1997-01-05	13500.00

D. to sort employees by salary from high to low:

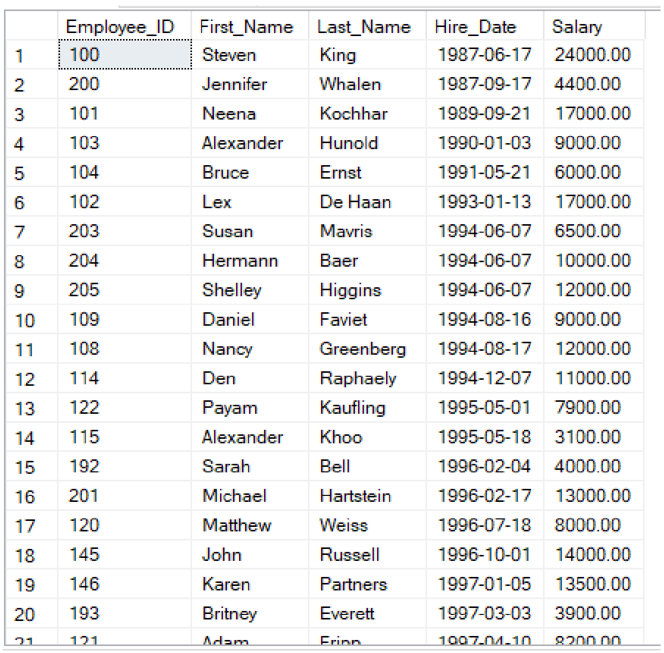
```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary

 FROM Employees

 ORDER BY Salary DESC
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	100	Steven	King	1987-06-17	24000.00
2	101	Neena	Kochhar	1989-09-21	17000.00
3	102	Lex	De Haan	1993-01-13	17000.00
4	145	John	Russell	1996-10-01	14000.00
5	146	Karen	Partners	1997-01-05	13500.00
6	201	Michael	Hartstein	1996-02-17	13000.00
7	205	Shelley	Higgins	1994-06-07	12000.00
8	108	Nancy	Greenberg	1994-08-17	12000.00
9	114	Den	Raphaely	1994-12-07	11000.00
10	204	Hermann	Baer	1994-06-07	10000.00
11	109	Daniel	Faviet	1994-08-16	9000.00
12	103	Alexander	Hunold	1990-01-03	9000.00
13	176	Jonathon	Taylor	1998-03-24	8600.00
14	177	Jack	Livingston	1998-04-23	8400.00
15	121	Adam	Fripp	1997-04-10	8200.00
16	110	John	Chen	1997-09-28	8200.00
17	120	Matthew	Weiss	1996-07-18	8000.00
18	122	Payam	Kaufling	1995-05-01	7900.00
19	112	Jose Manuel	Urman	1998-03-07	7800.00
20	111	Ismael	Sciarra	1997-09-30	7700.00

E. to sort the employees by values in the hire_date column from:

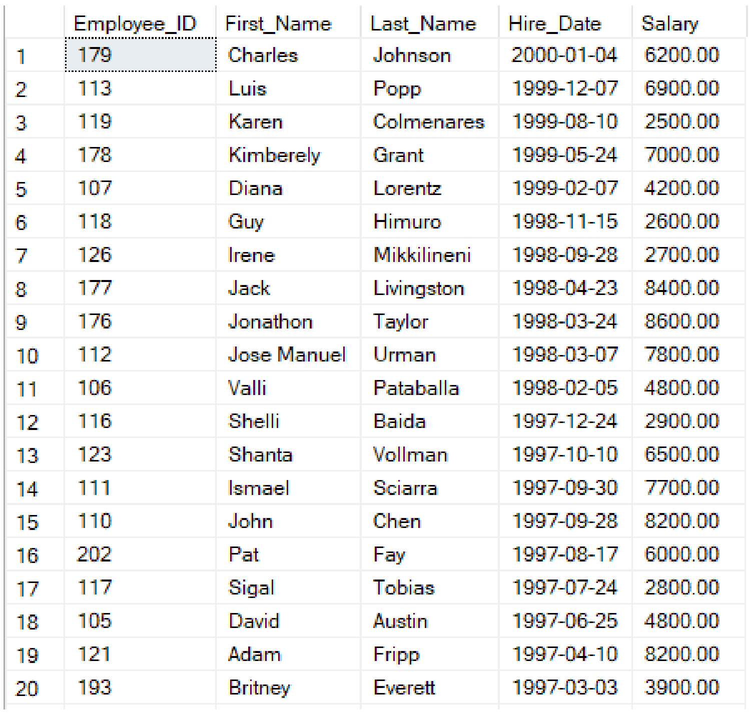
```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary

 FROM Employees

 ORDER BY Hire_Date ASC
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	100	Steven	King	1987-06-17	24000.00
2	200	Jennifer	Whalen	1987-09-17	4400.00
3	101	Neena	Kochhar	1989-09-21	17000.00
4	103	Alexander	Hunold	1990-01-03	9000.00
5	104	Bruce	Ernst	1991-05-21	6000.00
6	102	Lex	De Haan	1993-01-13	17000.00
7	203	Susan	Mavris	1994-06-07	6500.00
8	204	Hermann	Baer	1994-06-07	10000.00
9	205	Shelley	Higgins	1994-06-07	12000.00
10	109	Daniel	Faviet	1994-08-16	9000.00
11	108	Nancy	Greenberg	1994-08-17	12000.00
12	114	Den	Raphaely	1994-12-07	11000.00
13	122	Payam	Kaufling	1995-05-01	7900.00
14	115	Alexander	Khoo	1995-05-18	3100.00
15	192	Sarah	Bell	1996-02-04	4000.00
16	201	Michael	Hartstein	1996-02-17	13000.00
17	120	Matthew	Weiss	1996-07-18	8000.00
18	145	John	Russell	1996-10-01	14000.00
19	146	Karen	Partners	1997-01-05	13500.00
20	193	Britney	Everett	1997-03-03	3900.00

F. sort the employees by the hire dates in descending order:

```
SELECT Employee_ID, First_Name, Last_Name, Hire_Date, Salary
 FROM Employees
 ORDER BY Hire_Date DESC
```



	Employee_ID	First_Name	Last_Name	Hire_Date	Salary
1	179	Charles	Johnson	2000-01-04	6200.00
2	113	Luis	Popp	1999-12-07	6900.00
3	119	Karen	Colmenares	1999-08-10	2500.00
4	178	Kimberely	Grant	1999-05-24	7000.00
5	107	Diana	Lorentz	1999-02-07	4200.00
6	118	Guy	Himuro	1998-11-15	2600.00
7	126	Irene	Mikkilineni	1998-09-28	2700.00
8	177	Jack	Livingston	1998-04-23	8400.00
9	176	Jonathon	Taylor	1998-03-24	8600.00
10	112	Jose Manuel	Urman	1998-03-07	7800.00
11	106	Valli	Pataballa	1998-02-05	4800.00
12	116	Shelli	Baida	1997-12-24	2900.00
13	123	Shanta	Vollman	1997-10-10	6500.00
14	111	Ismael	Sciarra	1997-09-30	7700.00
15	110	John	Chen	1997-09-28	8200.00
16	202	Pat	Fay	1997-08-17	6000.00
17	117	Sigal	Tobias	1997-07-24	2800.00
18	105	David	Austin	1997-06-25	4800.00
19	121	Adam	Fripp	1997-04-10	8200.00
20	193	Britney	Everett	1997-03-03	3900.00

3)WRITE A QUERY FOR DISTINCT STATEMENTS :-

Syntax of DISTINCT Statements:-

SELECT DISTINCT

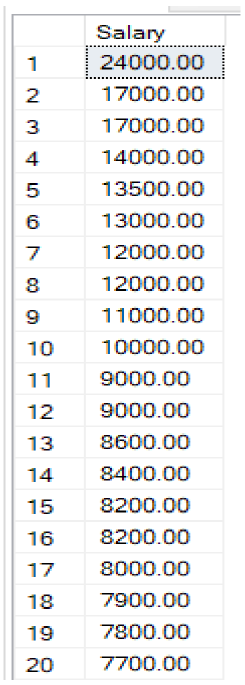
column1, column2, ...

FROM

table1;

A. selects the salary data from the salary column of the employees table and sorts them from high to low:

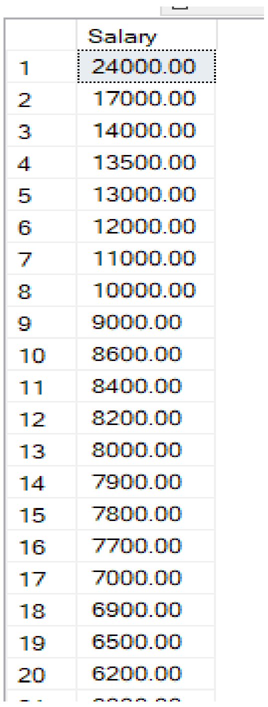
```
SELECT Salary
 FROM Employees
 ORDER BY Salary DESC
```



	Salary
1	24000.00
2	17000.00
3	17000.00
4	14000.00
5	13500.00
6	13000.00
7	12000.00
8	12000.00
9	11000.00
10	10000.00
11	9000.00
12	9000.00
13	8600.00
14	8400.00
15	8200.00
16	8200.00
17	8000.00
18	7900.00
19	7800.00
20	7700.00

B. select unique values from the salary column of the employees table:

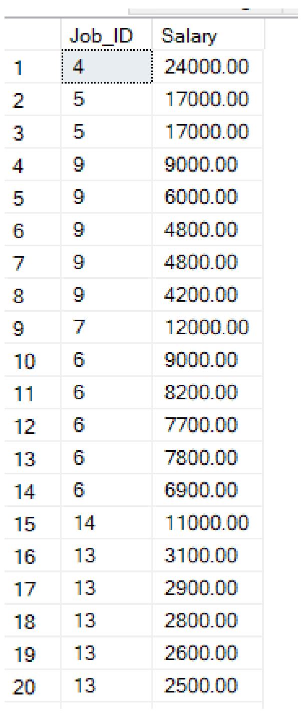
```
SELECT DISTINCT Salary

 FROM Employees

 ORDER BY Salary DESC
```



	Salary
1	24000.00
2	17000.00
3	14000.00
4	13500.00
5	13000.00
6	12000.00
7	11000.00
8	10000.00
9	9000.00
10	8600.00
11	8400.00
12	8200.00
13	8000.00
14	7900.00
15	7800.00
16	7700.00
17	7000.00
18	6900.00
19	6500.00
20	6200.00

C. selects the job id and salary from the employees table:

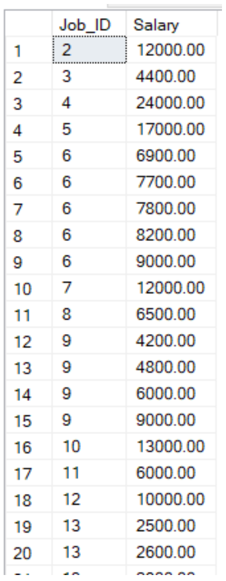
```
SELECT Job_ID, Salary
 FROM Employees
```



	Job_ID	Salary
1	4	24000.00
2	5	17000.00
3	5	17000.00
4	9	9000.00
5	9	6000.00
6	9	4800.00
7	9	4800.00
8	9	4200.00
9	7	12000.00
10	6	9000.00
11	6	8200.00
12	6	7700.00
13	6	7800.00
14	6	6900.00
15	14	11000.00
16	13	3100.00
17	13	2900.00
18	13	2800.00
19	13	2600.00
20	13	2500.00

D. to remove the duplicate values in job id and salary:

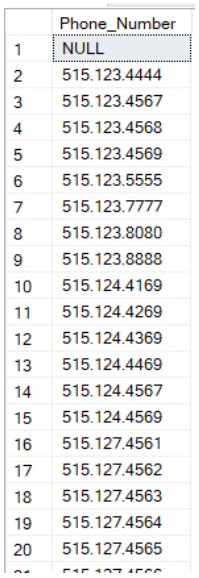
```
SELECT  DISTINCT Job_ID, Salary
 FROM Employees
```



	Job_ID	Salary
1	2	12000.00
2	3	4400.00
3	4	24000.00
4	5	17000.00
5	6	6900.00
6	6	7700.00
7	6	7800.00
8	6	8200.00
9	6	9000.00
10	7	12000.00
11	8	6500.00
12	9	4200.00
13	9	4800.00
14	9	6000.00
15	9	9000.00
16	10	13000.00
17	11	6000.00
18	12	10000.00
19	13	2500.00
20	13	2600.00

E. returns the distinct phone numbers of employees:

```
SELECT  DISTINCT Phone_Number FROM Employees
```



	Phone_Number
1	NULL
2	515.123.4444
3	515.123.4567
4	515.123.4568
5	515.123.4569
6	515.123.5555
7	515.123.7777
8	515.123.8080
9	515.123.8888
10	515.124.4169
11	515.124.4269
12	515.124.4369
13	515.124.4469
14	515.124.4567
15	515.124.4569
16	515.127.4561
17	515.127.4562
18	515.127.4563
19	515.127.4564
20	515.127.4565
21	515.127.4566

4)WRITE A QUERY FOR TOP N STATEMENTS :-

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

SELECT TOP N

column_list

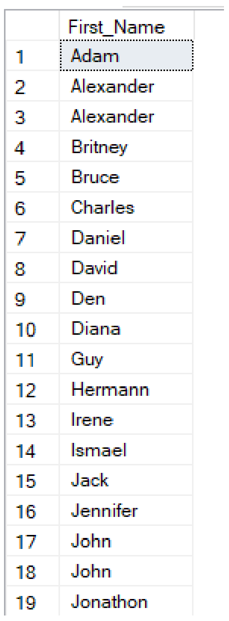
FROM

table1

ORDER BY column_list

A. returns all rows in the employees table sorted by the first_name column.

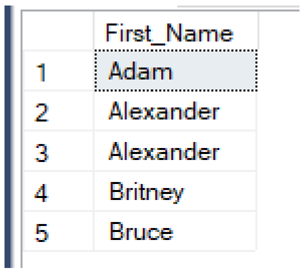
```
SELECT First_Name FROM Employees

ORDER BY First_Name
```



	First_Name
1	Adam
2	Alexander
3	Alexander
4	Britney
5	Bruce
6	Charles
7	Daniel
8	David
9	Den
10	Diana
11	Guy
12	Hermann
13	Irene
14	Ismael
15	Jack
16	Jennifer
17	John
18	John
19	Jonathon

B. to return the first 5 rows in the result set returned by the SELECT clause:

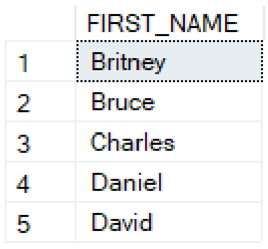
```
SELECT TOP 5 First_Name FROM Employees

ORDER BY First_Name
```



	First_Name
1	Adam
2	Alexander
3	Alexander
4	Britney
5	Bruce

C. to return five rows starting from the 4th row:

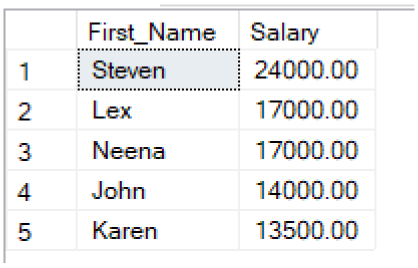
```
SELECT FIRST_NAME  FROM Employees

ORDER BY First_Name

OFFSET 3 ROWS FETCH NEXT 5 ROWS ONLY;
```



	FIRST_NAME
1	Britney
2	Bruce
3	Charles
4	Daniel
5	David

D. gets the top five employees with the highest salaries.

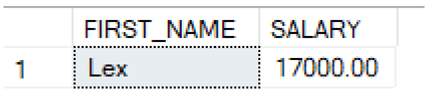
```
SELECT TOP 5 First_Name, Salary  FROM Employees

ORDER BY Salary DESC
```



	First_Name	Salary
1	Steven	24000.00
2	Lex	17000.00
3	Neena	17000.00
4	John	14000.00
5	Karen	13500.00

E. to get employees who have the 2nd highest salary in the company

```
SELECT FIRST_NAME, SALARY FROM Employees

ORDER BY Salary DESC

OFFSET 1 ROW FETCH NEXT 1 ROW ONLY
```



	FIRST_NAME	SALARY
1	Lex	17000.00

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.

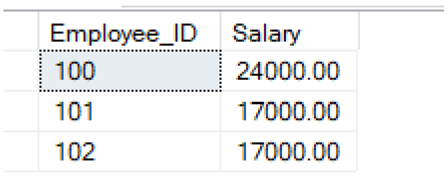
Note that SQL has three-valued logic which are TRUE, FALSE, and UNKNOWN. It means that if a row causes the condition to evaluate to FALSE or NULL, the row will not be returned.

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
<=	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.

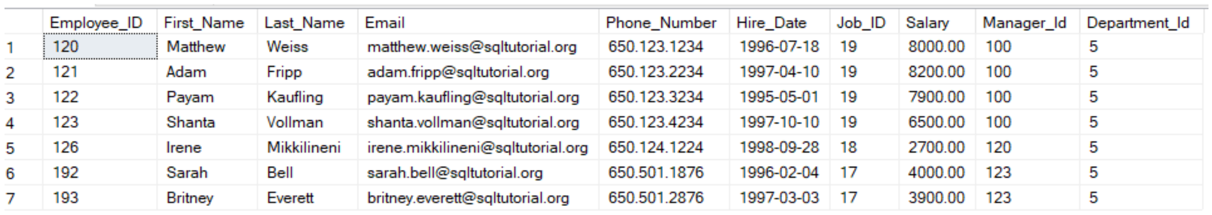
```
SELECT Employee_ID, Salary FROM EMPLOYEES

WHERE SALARY > 14000

ORDER BY SALARY DESC
```



Employee_ID	Salary
100	24000.00
101	17000.00
102	17000.00

B. query finds all employees who work in the department id 5.

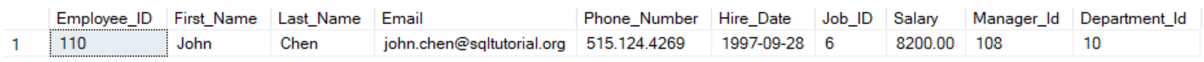
```
SELECT * FROM EMPLOYEES

WHERE DEPARTMENT_ID = 5
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
2	121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
3	122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
4	123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
5	126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
6	192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
7	193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5

C. query finds the employee whose last name is Chen

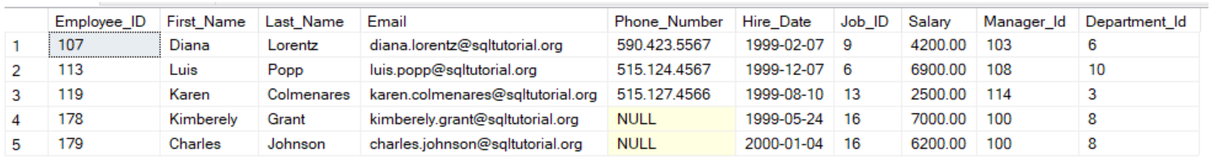
```
SELECT * FROM EMPLOYEES

 WHERE LAST_NAME LIKE 'Chen%'
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10

D. To get all employees who joined the company after January 1st, 1999

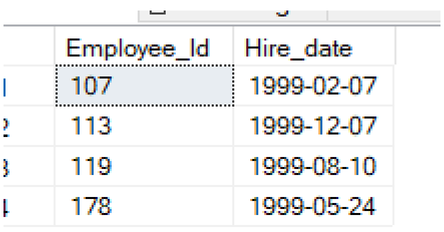
```
SELECT * FROM EMPLOYEES

WHERE hire_date > '1999-01-01'
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
2	113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
3	119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3
4	178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
5	179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

E. to find the employees who joined the company in 1999,

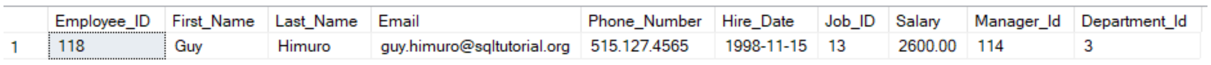
```
SELECT Employee_Id, Hire_date FROM EMPLOYEES

WHERE YEAR(hire_date) = 1999
```



	Employee_Id	Hire_date
	107	1999-02-07
	113	1999-12-07
	119	1999-08-10
	178	1999-05-24

F. statement finds the employee whose last name is Himuro

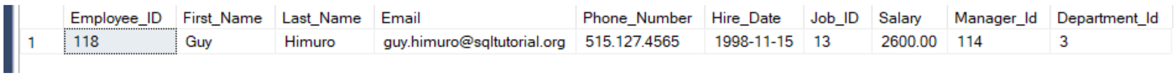
```
SELECT * FROM EMPLOYEES

 WHERE LAST_NAME LIKE 'Himuro'
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3

G. the query searches for the string Himuro in the last_name column of the employees table.

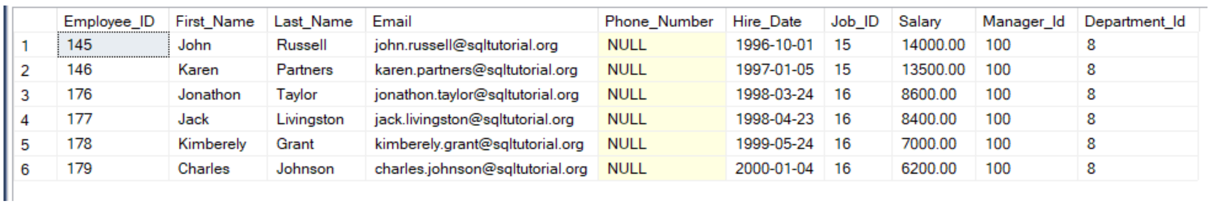
```
SELECT * FROM EMPLOYEES

 WHERE LAST_NAME LIKE '%Himuro%'
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3

H. to find all employees who do not have phone numbers:

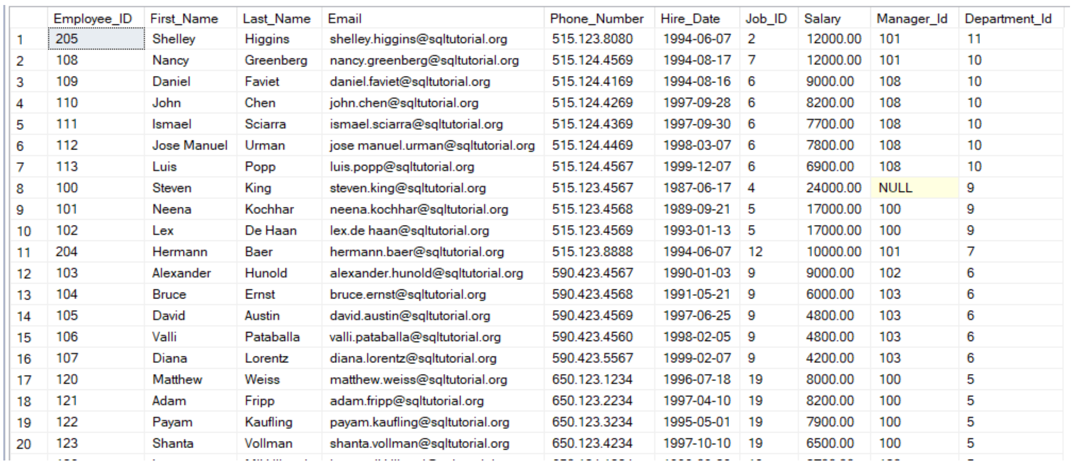
```
SELECT * FROM EMPLOYEES

 WHERE Phone_Number IS NULL
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
2	146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
3	176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
4	177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
5	178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
6	179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

I. returns all employees whose department id is not 8.

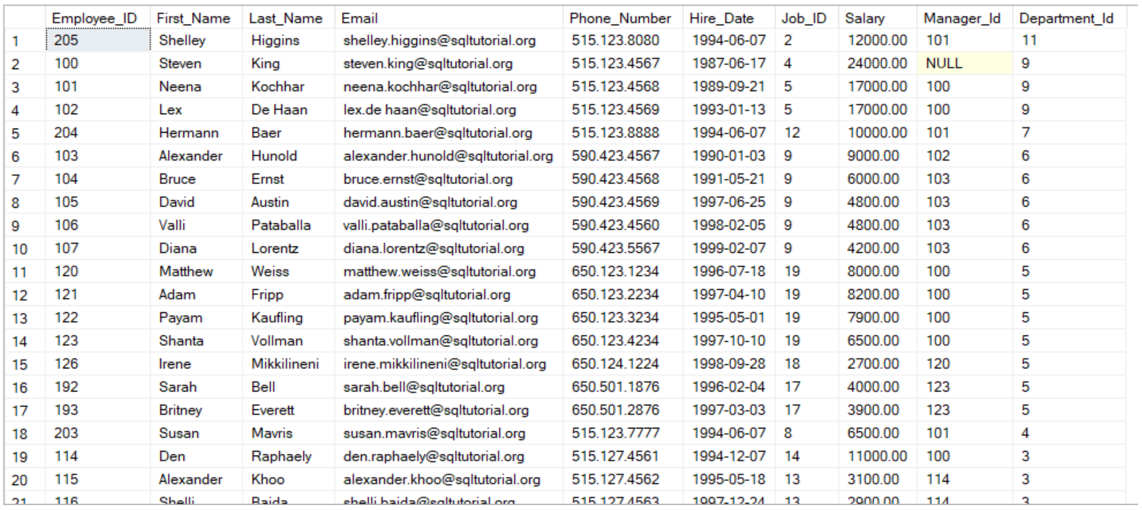
```
 SELECT * FROM EMPLOYEES

  WHERE Department_Id <> 8

  ORDER BY Department_Id DESC
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11
2	108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
3	109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
4	110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
5	111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
6	112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
7	113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
8	100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
9	101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
10	102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
11	204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
12	103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
13	104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
14	105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
15	106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
16	107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
17	120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
18	121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
19	122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
20	123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5

J. finds all employees whose department id is not eight and ten.

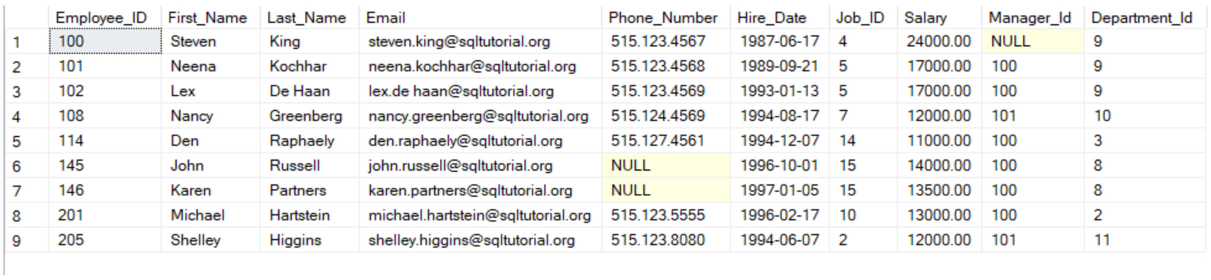
```
SELECT * FROM EMPLOYEES

 WHERE Department_Id <> 8

 AND

 Department_Id <> 10

 ORDER BY Department_Id DESC
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11
2	100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
3	101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
4	102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
5	204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
6	103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
7	104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
8	105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
9	106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
10	107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
11	120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
12	121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
13	122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
14	123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
15	126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
16	192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
17	193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5
18	203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	1994-06-07	8	6500.00	101	4
19	114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
20	115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
21	116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3

K. to find the employees whose salary is greater than 10,000,

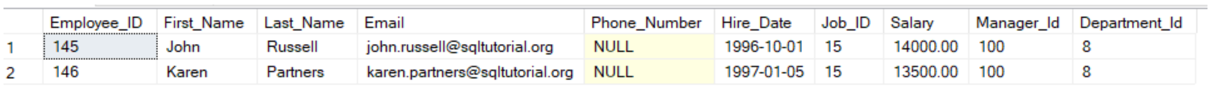
```
SELECT * FROM EMPLOYEES

WHERE Salary > 10000
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
2	101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
3	102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
4	108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
5	114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
6	145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
7	146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
8	201	Michael	Hartstein	michael.hartstein@sqltutorial.org	515.123.5555	1996-02-17	10	13000.00	100	2
9	205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11

L. finds employees in department 8 and have the salary greater than 10,000:

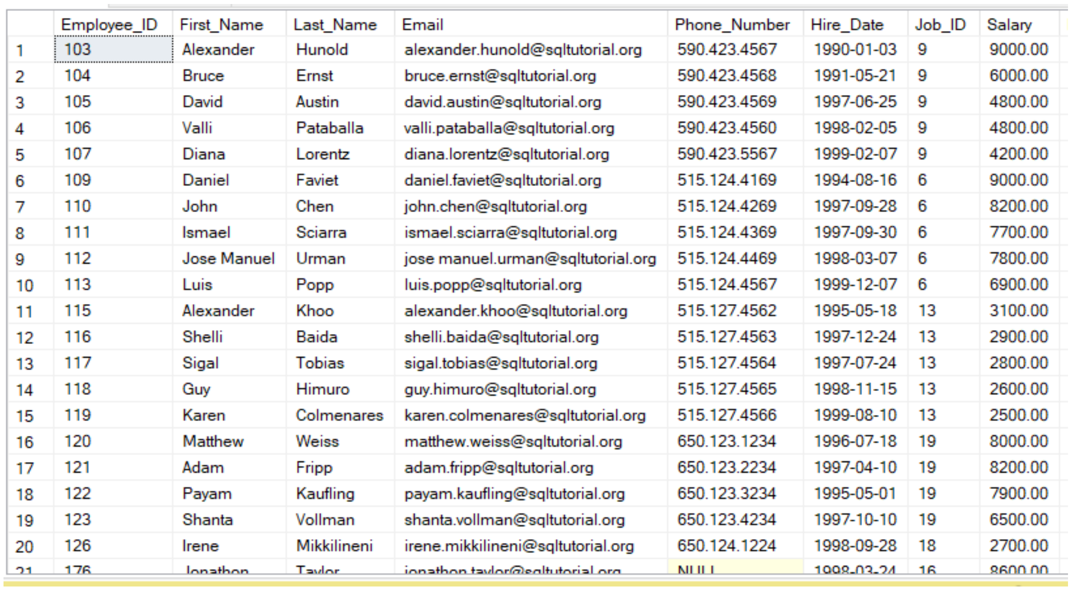
```
SELECT * FROM EMPLOYEES

WHERE DEPARTMENT_ID = 8

AND

Salary > 10000
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
2	146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8

M. the statement below returns all employees whose salaries are less than 10,000:

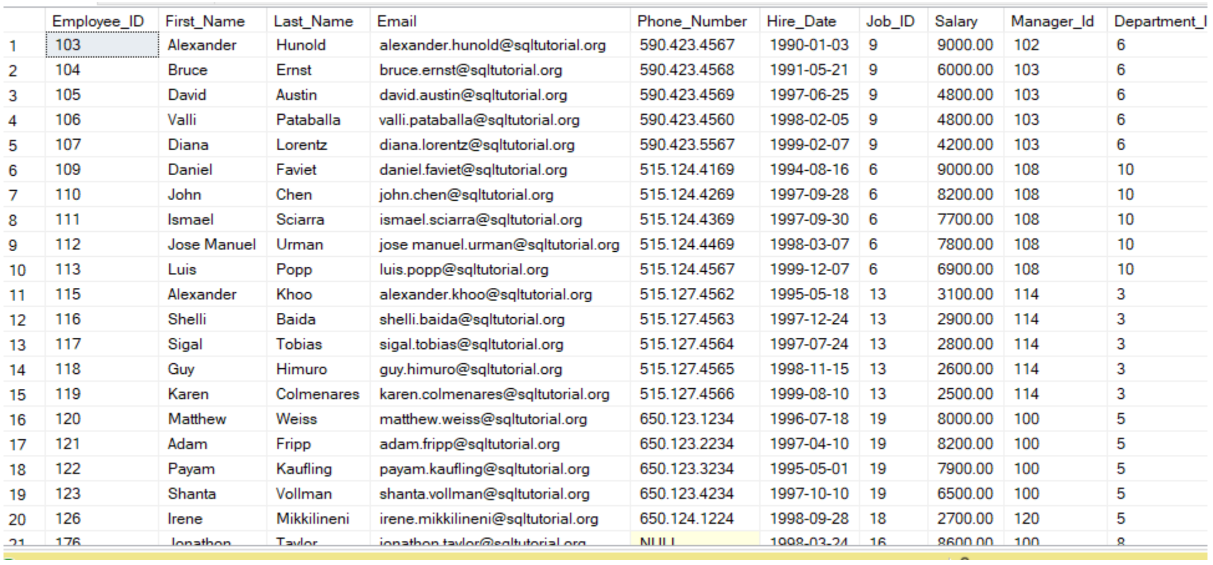
```
SELECT * FROM EMPLOYEES

WHERE Salary < 10000
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary
1	103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00
2	104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00
3	105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00
4	106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00
5	107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00
6	109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00
7	110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00
8	111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00
9	112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00
10	113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00
11	115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00
12	116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00
13	117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00
14	118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00
15	119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00
16	120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00
17	121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00
18	122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00
19	123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00
20	126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00
21	176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00

N. finds employees whose salaries are greater than or equal 9,000:

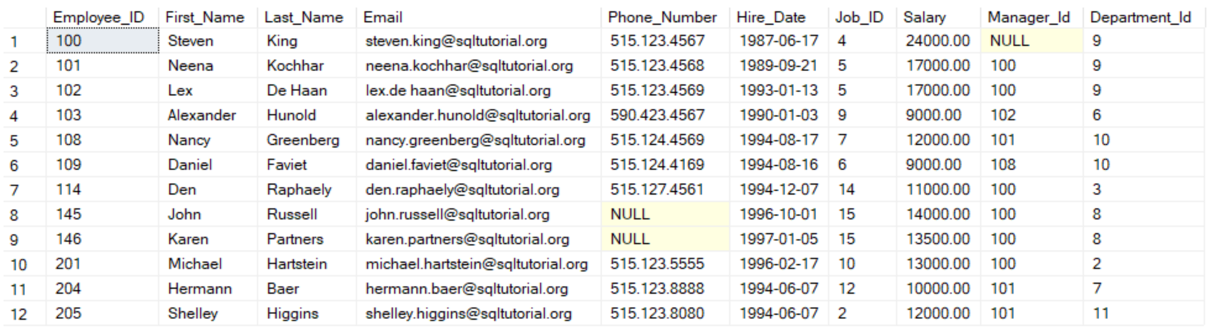
```
SELECT * FROM EMPLOYEES

WHERE Salary <= 9000
```



	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_I
1	103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
2	104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
3	105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
4	106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
5	107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
6	109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
7	110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
8	111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
9	112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
10	113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
11	115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
12	116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3
13	117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3
14	118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3
15	119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3
16	120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
17	121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
18	122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
19	123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
20	126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
21	176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8

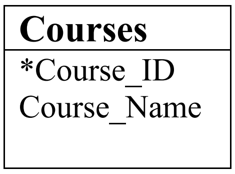
O. finds employees whose salaries are less than or equal to 9,000:

```
SELECT * FROM EMPLOYEES

WHERE Salary >= 9000
```

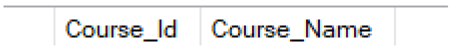


	Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
1	100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
2	101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
3	102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
4	103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
5	108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
6	109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
7	114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
8	145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
9	146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
10	201	Michael	Hartstein	michael.hartstein@sqltutorial.org	515.123.5555	1996-02-17	10	13000.00	100	2
11	204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
12	205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11

6)WRITE A QUERY FOR:-



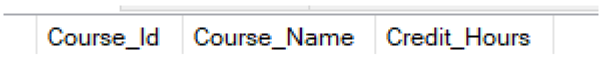
```
CREATE TABLE Courses(

Course_Id INT PRIMARY KEY NOT NULL,

Course_Name VARCHAR(100) NOT NULL)


SELECT * FROM Courses
```



	Course_Id	Course_Name

A. adds a new column named credit_hours to the courses table.

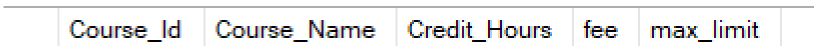
```
ALTER TABLE Courses

ADD Credit_Hours INT


SELECT * FROM Courses
```



	Course_Id	Course_Name	Credit_Hours

B. adds the fee and max_limit columns to the courses table and places these columns after the course_name column.

```
ALTER TABLE Courses
ADD fee DECIMAL(10,2),
max_limit INT ;

SELECT * FROM Courses
```



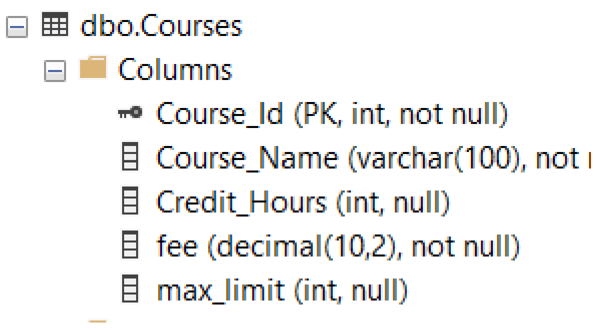
	Course_Id	Course_Name	Credit_Hours	fee	max_limit

We cannot control column order in SQL Server.

Columns will be added at the end of the table.

C. changes the attribute of the fee column to NOT NULL.

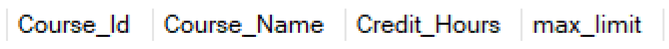
```
ALTER TABLE COURSES
ALTER COLUMN FEE DECIMAL(10,2) NOT NULL
```



```
dbo.Courses
  Columns
    Course_Id (PK, int, not null)
    Course_Name (varchar(100), not ı
    Credit_Hours (int, null)
    fee (decimal(10,2), not null)
    max_limit (int, null)
```

D. to remove the fee column of the courses table

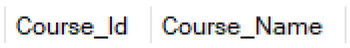
```
ALTER TABLE COURSES
DROP COLUMN FEE
SELECT * FROM Courses
```



Course_Id	Course_Name	Credit_Hours	max_limit

E. removes the max_limit and credit_hours of the courses table.

```
ALTER TABLE COURSES

DROP COLUMN Max_Limit, Credit_hours

SELECT * FROM Courses
```



Course_Id	Course_Name

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) );
```

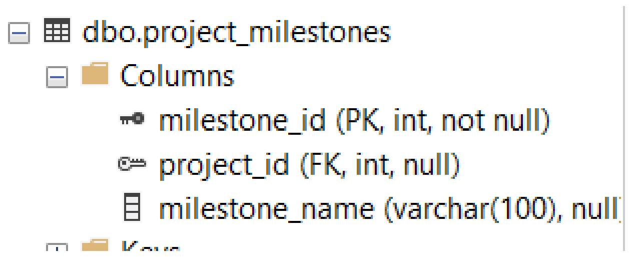
Each project may have zero or more milestones while one milestone must belong to one and only one project. The application that uses these tables must ensure that for each row in the project_milestones table there exists the corresponding row in the projects table. In other words, a milestone cannot 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.

```
ALTER TABLE PROJECT_MILESTONES

ADD FOREIGN KEY (PROJECT_ID) REFERENCES projects (project_id)
```



```
dbo.project_milestones
  Columns
    milestone_id (PK, int, not null)
    project_id (FK, int, null)
    milestone_name (varchar(100), null
```

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

```
ALTER TABLE project_milestones

ADD CONSTRAINT fk_project

FOREIGN KEY (project_id) REFERENCES projects(project_id)
```

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:

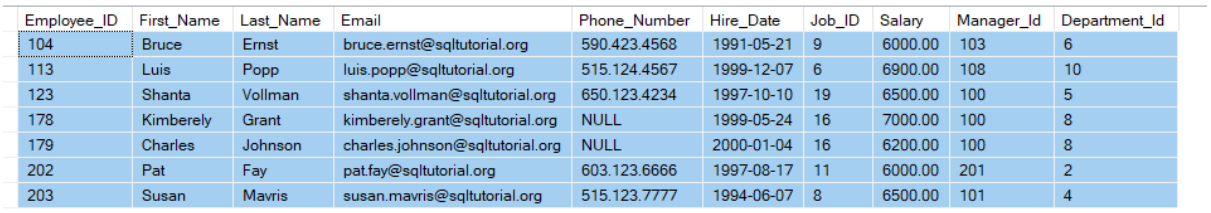
Operator	Meaning
AND	Return true if both expressions are true
NOT	Reverse the result of any other Boolean operator.
OR	Return true if either expression is true
ANY	Return true if any one of the comparisons is true.
BETWEEN	Return true if the operand is within a range
EXISTS	Return true if a subquery 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:

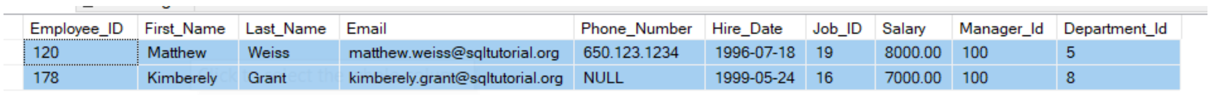
```
SELECT * FROM EMPLOYEES

WHERE SALARY BETWEEN 5000 AND 7000
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
202	Pat	Fay	pat.fay@sqltutorial.org	603.123.6666	1997-08-17	11	6000.00	201	2
203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	1994-06-07	8	6500.00	101	4

B. finds employees whose salary is either 7,000 or 8,000:

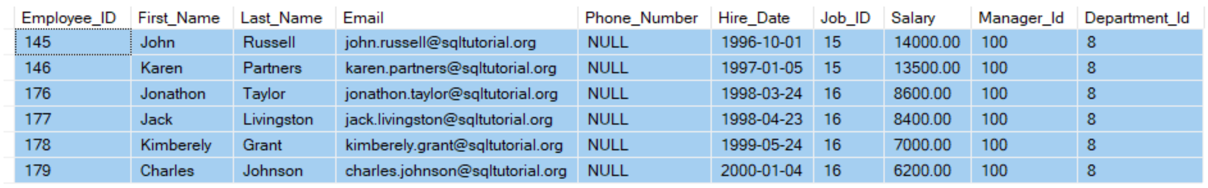
```
SELECT * FROM Employees

WHERE Salary IN (7000, 8000);
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8

C. finds all employees who do not have a phone number:

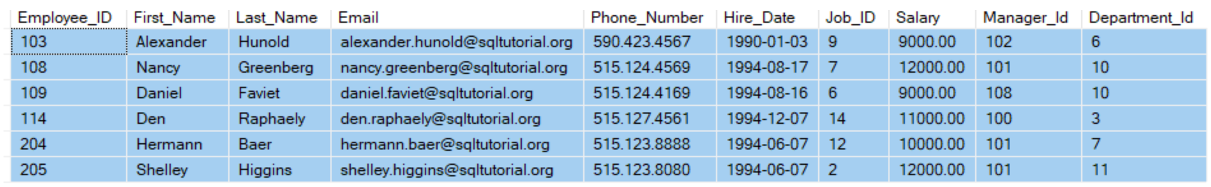
```
SELECT * FROM Employees

WHERE PHONE_NUMBER IS NULL
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

D. finds all employees whose salaries are between 9,000 and 12,000.

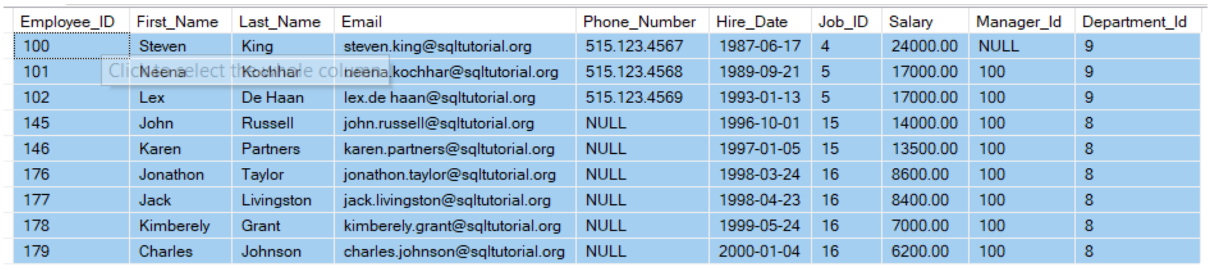
```
SELECT * FROM EMPLOYEES

WHERE SALARY BETWEEN 9000 AND 12000
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11

E. finds all employees who work in the department id 8 or 9.

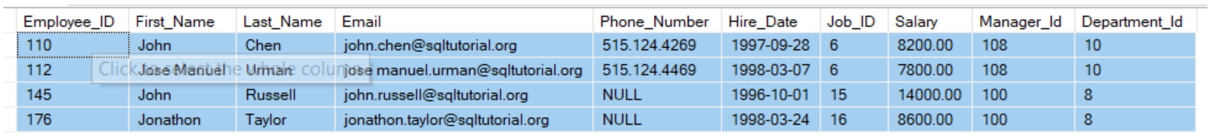
```
SELECT * FROM Employees

WHERE DEPARTMENT_ID IN (8, 9);
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

F. finds all employees whose first name starts with the string jo

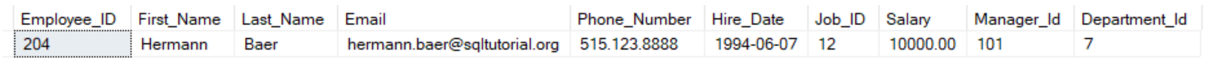
```
SELECT * FROM Employees

WHERE FIRST_NAME LIKE 'JO%'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8

G. finds all employees with the first names whose the second character is h

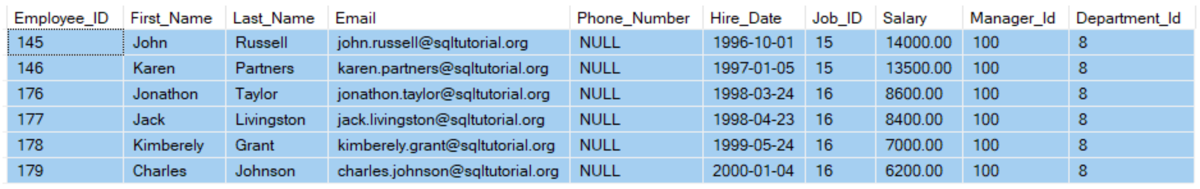
```
SELECT * FROM Employees

WHERE FIRST_NAME LIKE 'H%'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7

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

```
SELECT * FROM Employees

WHERE DEPARTMENT_ID = 8

ORDER BY SALARY DESC
```

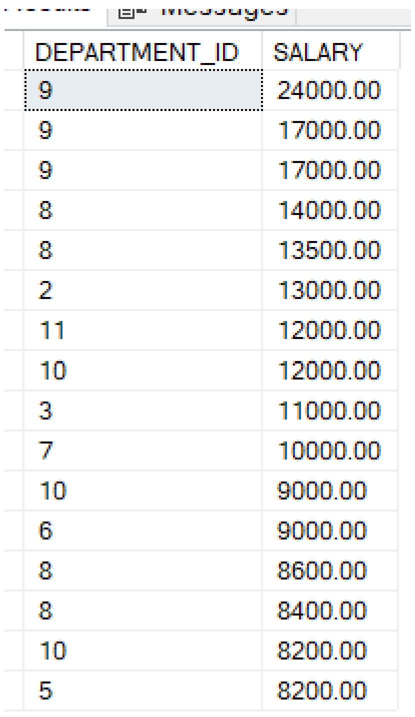


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

Part 2:-

A. finds all employees whose salaries are greater than the average salary of every department:

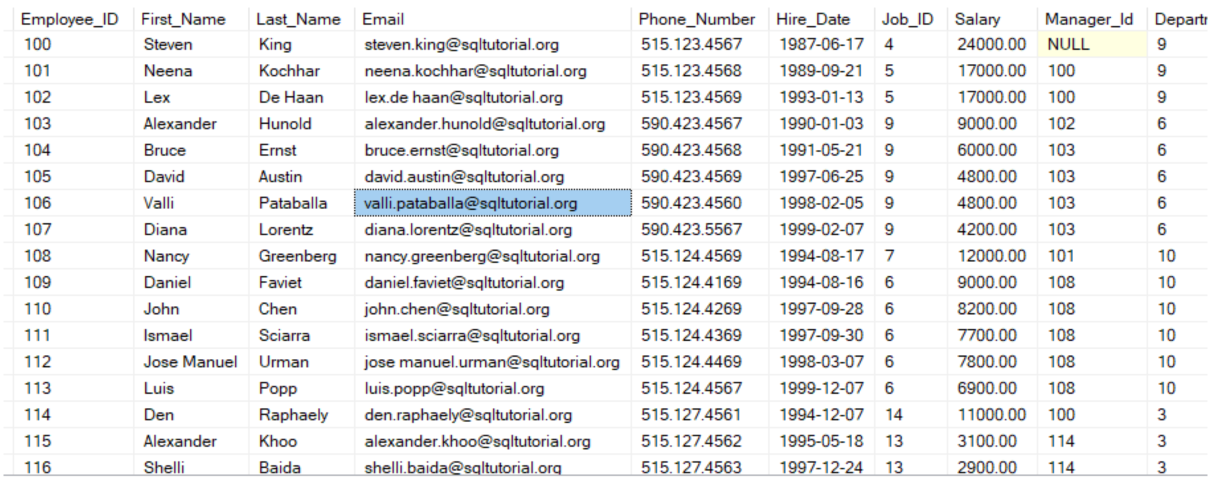
```
SELECT DEPARTMENT_ID,SALARY FROM EMPLOYEES

WHERE SALARY > (

SELECT AVG(SALARY) FROM EMPLOYEES)

ORDER BY SALARY DESC
```



DEPARTMENT_ID	SALARY
9	24000.00
9	17000.00
9	17000.00
8	14000.00
8	13500.00
2	13000.00
11	12000.00
10	12000.00
3	11000.00
7	10000.00
10	9000.00
6	9000.00
8	8600.00
8	8400.00
10	8200.00
5	8200.00

B. finds all employees who have dependents:

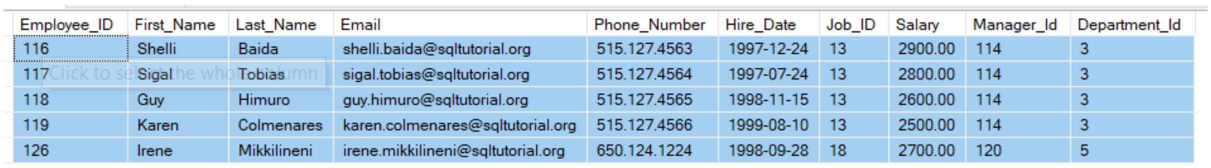
```
SELECT DISTINCT e.*

FROM employees e

INNER JOIN

dependents d

ON

e.employee_id = d.employee_id;
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Departr
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3

C. to find all employees whose salaries are between 2,500 and 2,900:

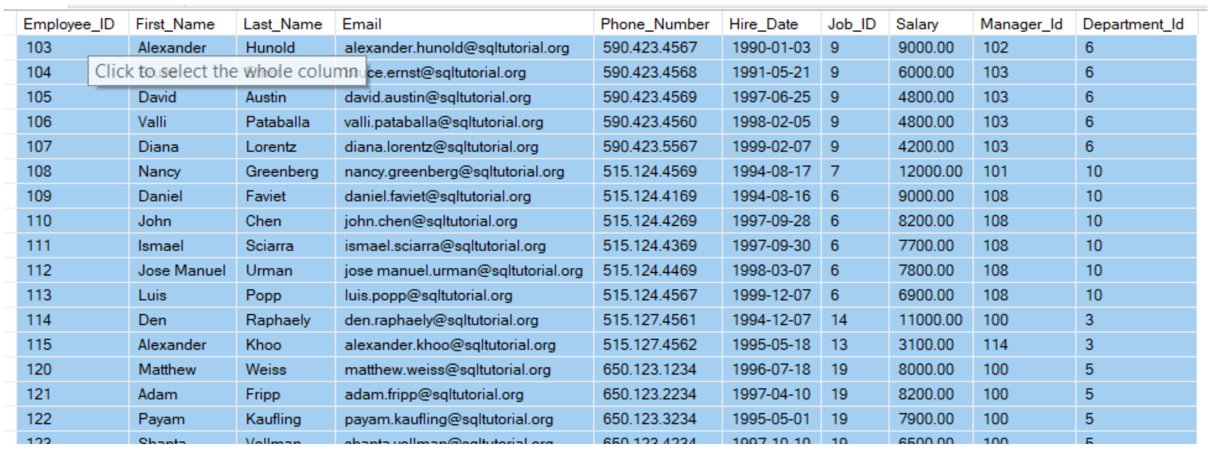
```
SELECT * FROM EMPLOYEES

WHERE SALARY BETWEEN 2500 AND 2900
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3
118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3
119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5

D. to find all employees whose salaries are not in the range of 2,500 and 2,900:

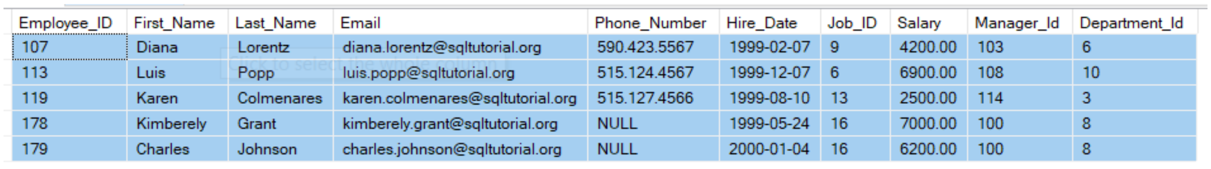
```
SELECT * FROM EMPLOYEES

WHERE SALARY NOT BETWEEN 2500 AND 2900
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	[illegible]	[illegible]	ce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	[illegible]	6500.00	100	5

E. to find all employees who joined the company between January 1, 1999, and December 31, 2000:

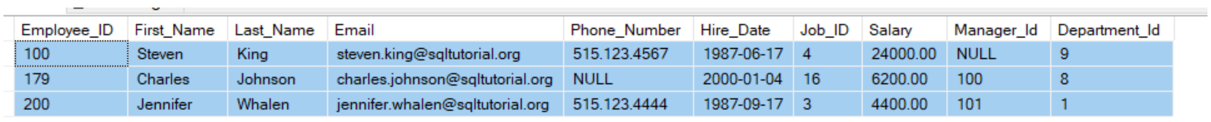
```
SELECT * FROM EMPLOYEES

WHERE HIRE_DATE BETWEEN '1999-01-01' AND '2000-12-31'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

F. to find employees who have not joined the company from January 1, 1989 to December 31, 1999:

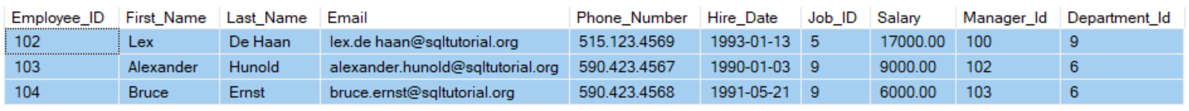
```
SELECT * FROM EMPLOYEES

WHERE HIRE_DATE NOT BETWEEN '1989-01-01' AND '1999-12-31'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	1987-09-17	3	4400.00	101	1

G. to find employees who joined the company between 1990 and 1993:

```
SELECT * FROM EMPLOYEES

WHERE YEAR(HIRE_DATE) BETWEEN 1990 AND 1993
```

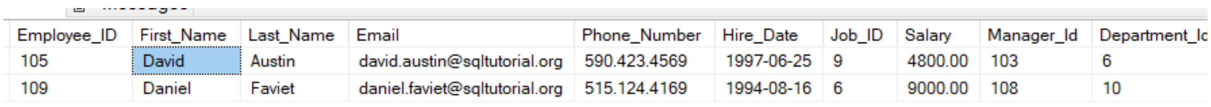


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6

Part 3:-

A. to find all employees whose first names start with Da

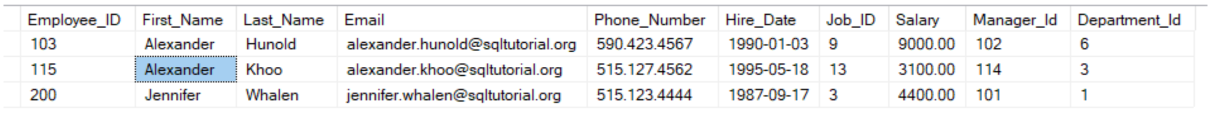
```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME LIKE 'DA%'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Ic
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10

B. to find all employees whose first names end with er

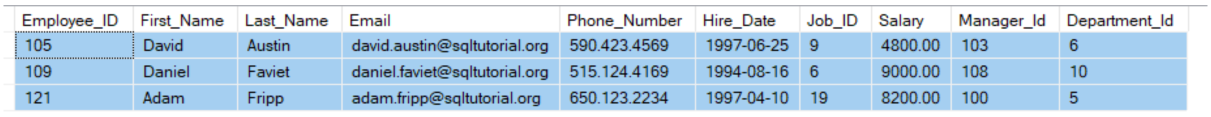
```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME LIKE '%ER'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	1987-09-17	3	4400.00	101	1

C. to find employees whose last names contain the word an:

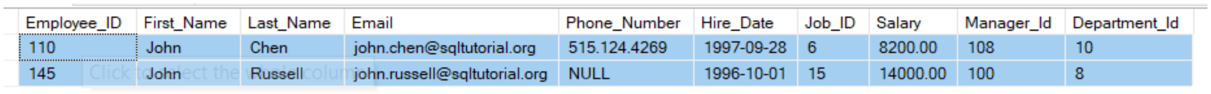
```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME LIKE '%DA%'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5

D. retrieves employees whose first names start with Jo and are followed by at most 2 characters:

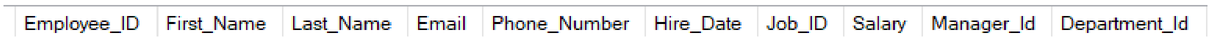
```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME LIKE 'Jo__'
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8

E. to find employees whose first names start with any number of characters and are followed by at most one character:

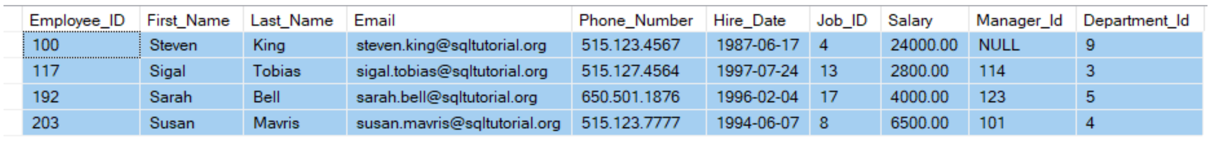
```
SELECT * FROM employees

WHERE LEN(first_name) <= 2;
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id

F. to find all employees whose first names start with the letter S but not start with Sh:

```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME LIKE 'S%'

AND

first_name NOT LIKE 'Sh%'
```

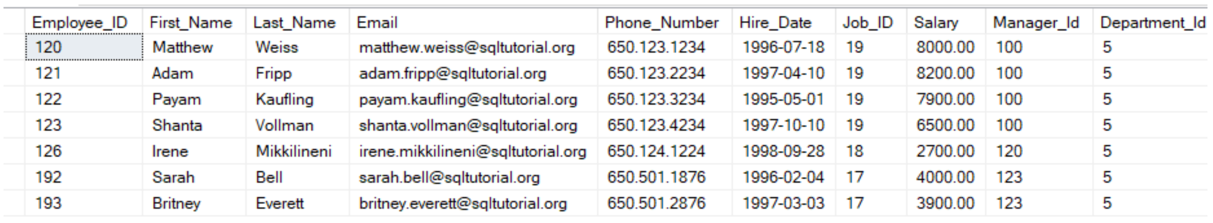


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	1994-06-07	8	6500.00	101	4

Part 4:-

A. retrieves all employees who work in the department id 5.

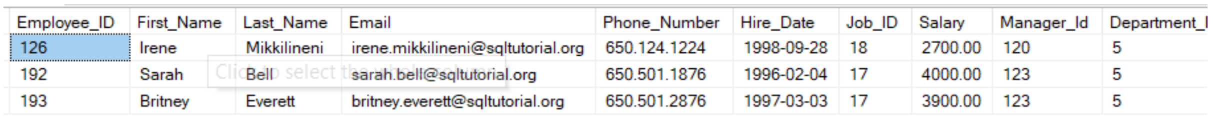
```
SELECT * FROM EMPLOYEES

WHERE department_id = 5
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5

B. To get the employees who work in the department id 5 and with a salary not greater than 5000.

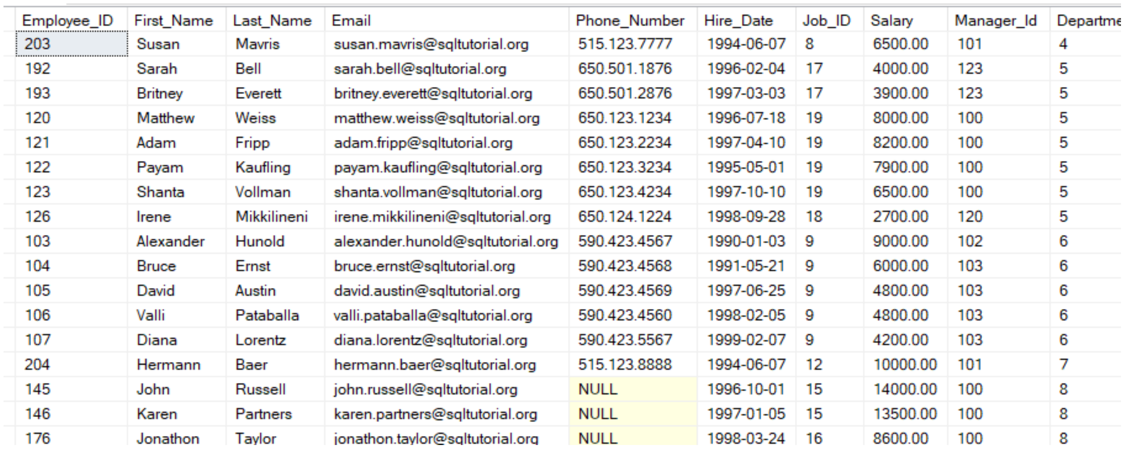
```
SELECT * FROM EMPLOYEES

WHERE department_id = 5

AND

SALARY !> 5000
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_I
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5

C. statement gets all the employees who are not working in the departments 1, 2, or 3.

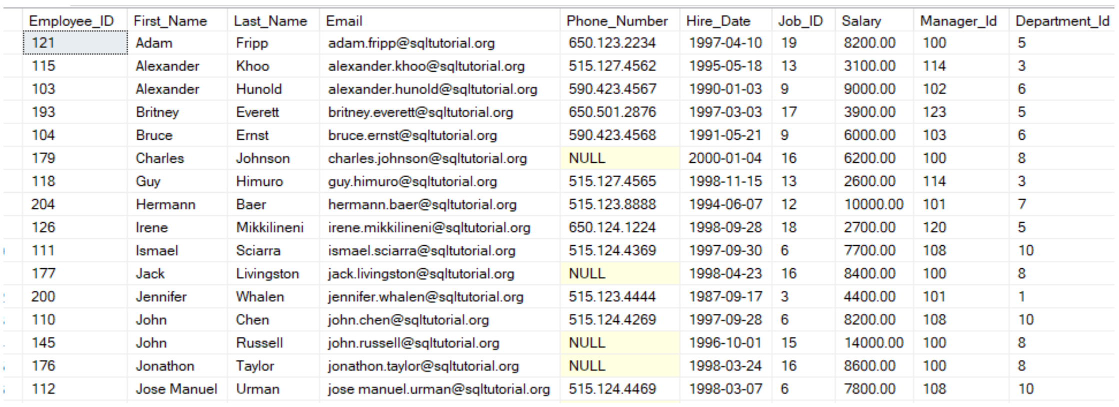
```
SELECT * FROM EMPLOYEES

WHERE department_id NOT IN (1,2,3)

ORDER BY DEPARTMENT_ID
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Departme
203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	1994-06-07	8	6500.00	101	4
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5
120	Matthew	Weiss	matthew.weiss@sqltutorial.org	650.123.1234	1996-07-18	19	8000.00	100	5
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8

D. retrieves all the employees whose first names do not start with the letter D.

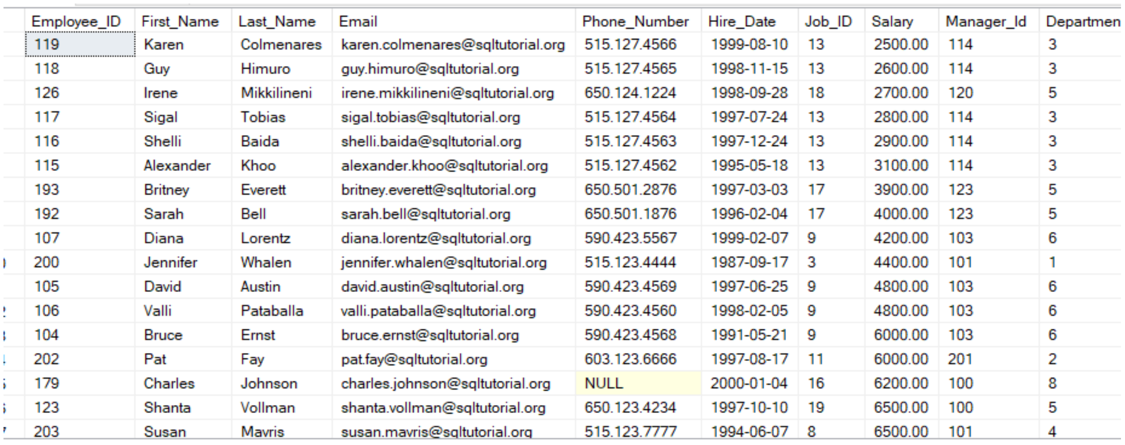
```
SELECT * FROM EMPLOYEES

WHERE FIRST_NAME NOT LIKE 'D%'

ORDER BY FIRST_NAME ASC
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	1987-09-17	3	4400.00	101	1
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10

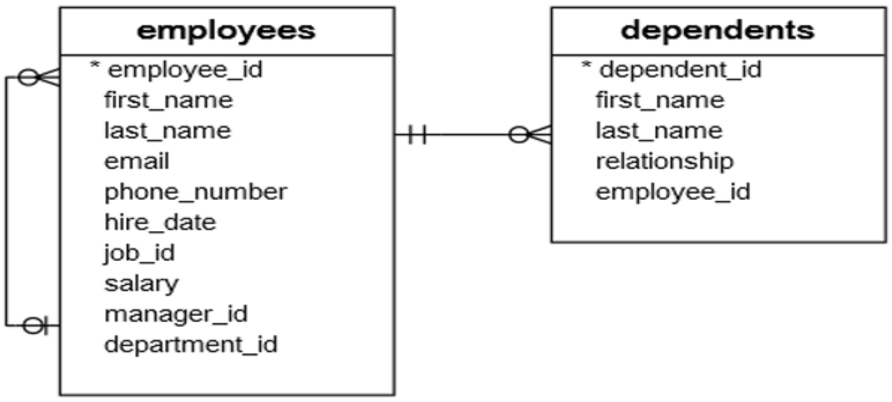
E. to get employees whose salaries are not between 5,000 and 1,000.

```
SELECT * FROM EMPLOYEES

WHERE SALARY NOT BETWEEN 5000 AND 1000

ORDER BY SALARY ASC
```



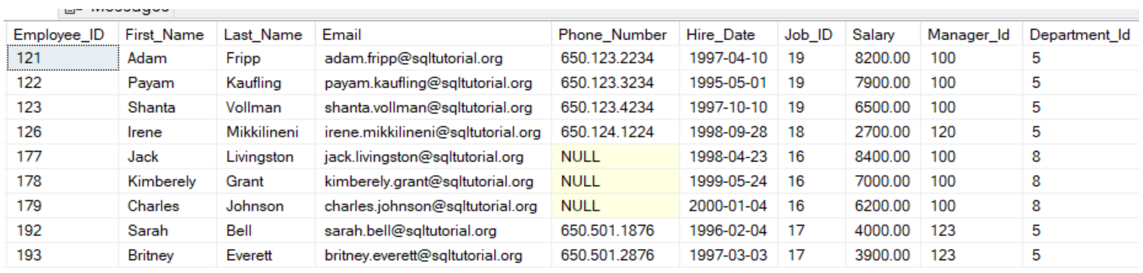
Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Departmen
119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3
118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	1987-09-17	3	4400.00	101	1
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
202	Pat	Fay	pat.fay@sqltutorial.org	603.123.6666	1997-08-17	11	6000.00	201	2
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
203	Susan	Mavris	susan.mavris@sqltutorial.org	515.123.7777	1994-06-07	8	6500.00	101	4

Part 5:-



A. Write a query to get the employees who do not have any dependents by above image

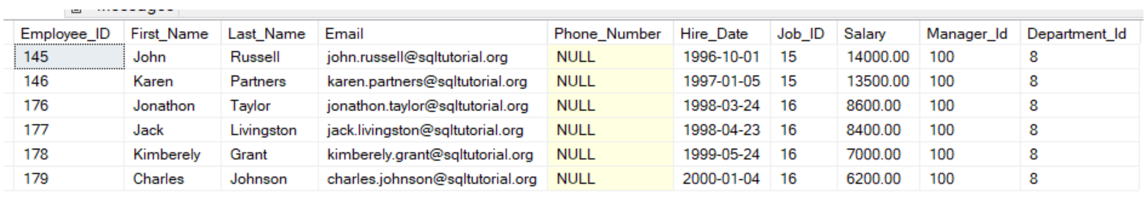
```
SELECT * FROM Employees E

LEFT JOIN

Dependents D

ON

E.Employee_ID = D.Employee_id

WHERE d.employee_id IS NULL;
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5

B. To find all employees who do not have the phone numbers

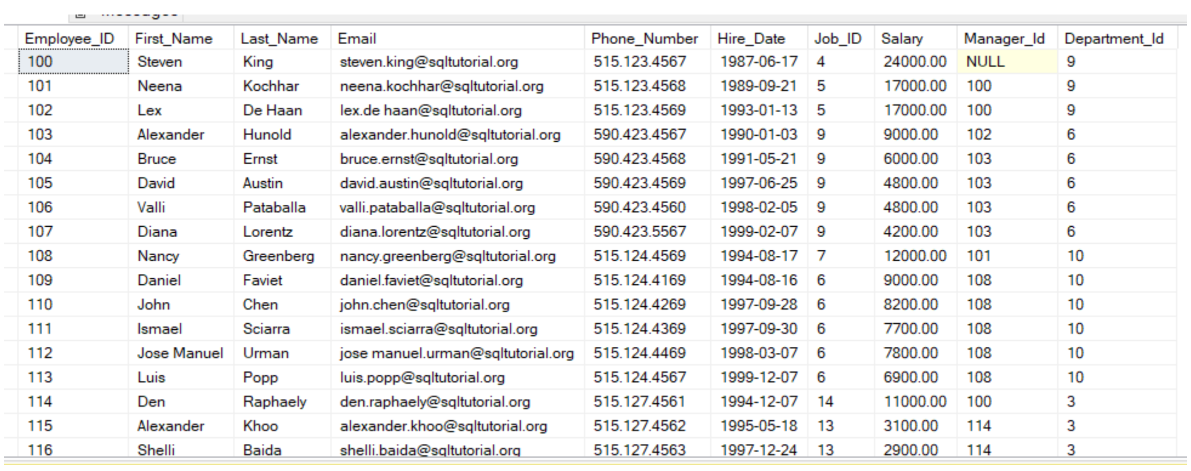
```
SELECT * FROM EMPLOYEES

WHERE Phone_Number IS NULL
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8

C. To find all employees who have phone numbers

```
SELECT * FROM EMPLOYEES

WHERE Phone_Number IS NOT NULL
```

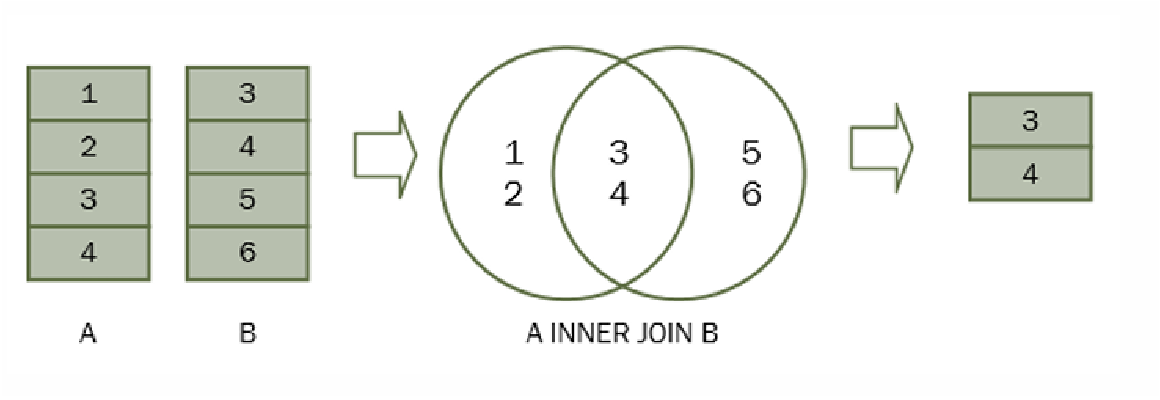


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3

TASK 3

JOINS:-

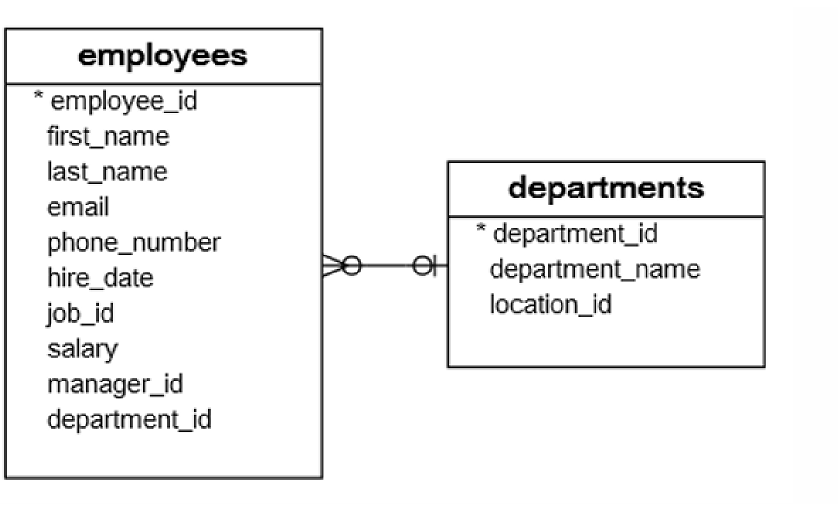
SQL INNER JOIN clause



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:

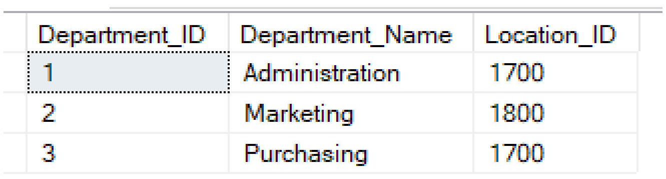
```
SELECT a
FROM A
INNER JOIN B
ON b = a;
```



1) Write a Query to

A. To get the information of the department id 1,2, and 3

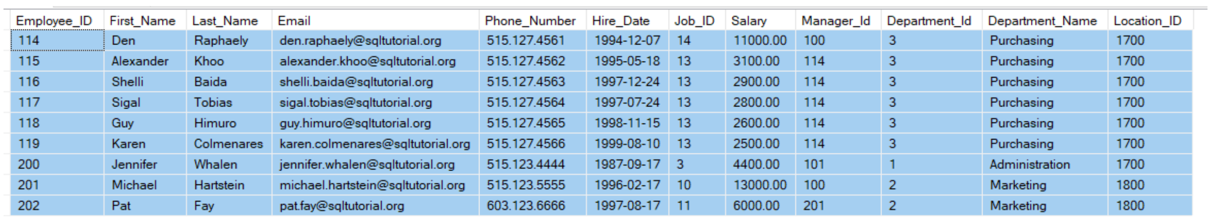
```
SELECT *

FROM departments

WHERE department_id IN (1, 2, 3);
```



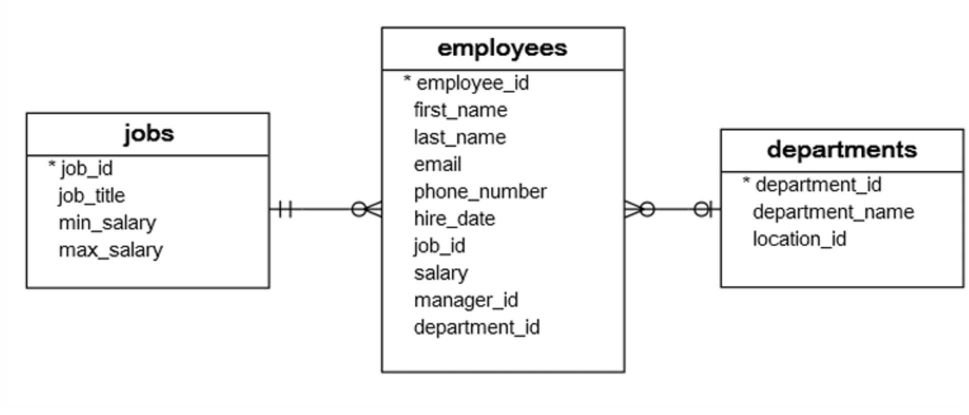
Department_ID	Department_Name	Location_ID
1	Administration	1700
2	Marketing	1800
3	Purchasing	1700

B. To get the information of employees who work in the department id 1, 2 and 3

```
SELECT E.*, D.Department_Name, D.Location_ID FROM Employees E

INNER JOIN

Departments D

ON

E.Department_Id = D.Department_ID

WHERE D.Department_ID IN (1,2,3)
```

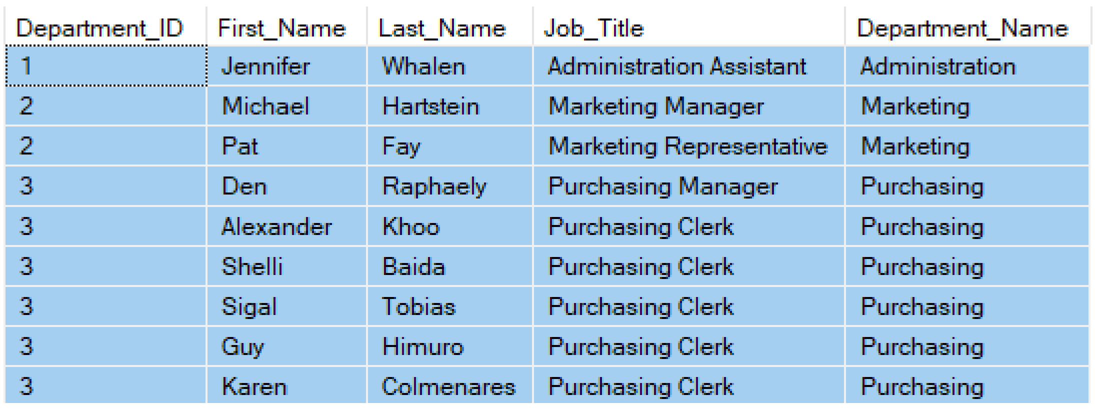


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id	Department_Name	Location_ID
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3	Purchasing	1700
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3	Purchasing	1700
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3	Purchasing	1700
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3	Purchasing	1700
118	Guy	Himuro	guy.himuro@sqltutorial.org	515.127.4565	1998-11-15	13	2600.00	114	3	Purchasing	1700
119	Karen	Colmenares	karen.colmenares@sqltutorial.org	515.127.4566	1999-08-10	13	2500.00	114	3	Purchasing	1700
200	Jennifer	Whalen	jennifer.whalen@sqltutorial.org	515.123.4444	1987-09-17	3	4400.00	101	1	Administration	1700
201	Michael	Hartstein	michael.hartstein@sqltutorial.org	515.123.5555	1996-02-17	10	13000.00	100	2	Marketing	1800
202	Pat	Fay	pat.fay@sqltutorial.org	603.123.6666	1997-08-17	11	6000.00	201	2	Marketing	1800



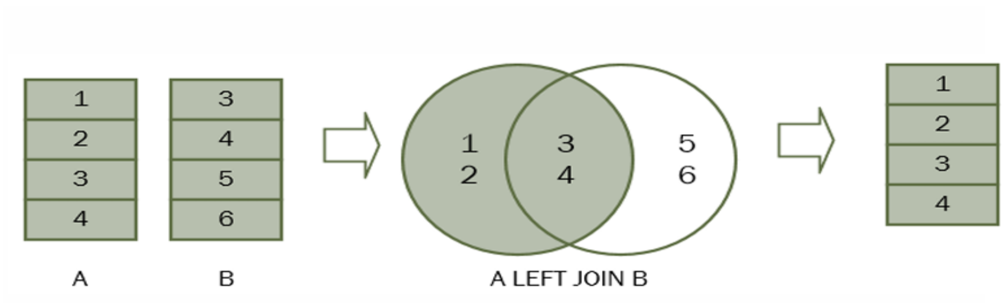
Write a Query to get the first name, last name, job title, and department name of employees who work in department id 1, 2, and 3.

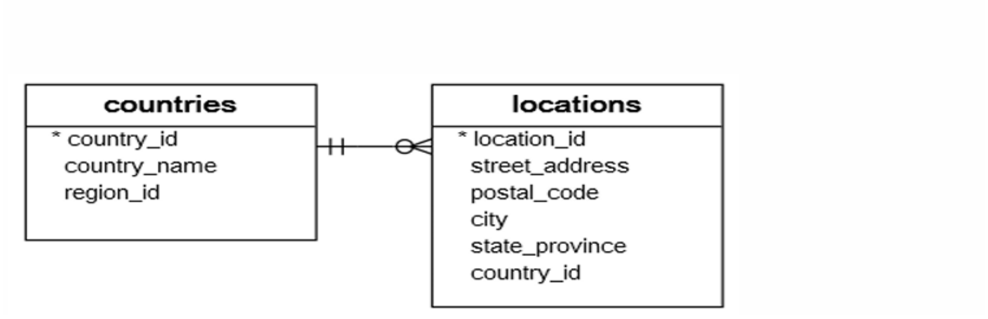
```
     SELECT D.Department_ID,E.First_Name, E.Last_Name, J.Job_Title, 
D.Department_Name

     FROM Departments D

     INNER JOIN Employees E

     ON D.Department_ID = E.Department_Id

     INNER JOIN Jobs J

     ON E.Job_ID = J.Job_ID

     WHERE D.Department_ID IN (1,2,3)

     ORDER BY D.Department_ID
```



Department_ID	First_Name	Last_Name	Job_Title	Department_Name
1	Jennifer	Whalen	Administration Assistant	Administration
2	Michael	Hartstein	Marketing Manager	Marketing
2	Pat	Fay	Marketing Representative	Marketing
3	Den	Raphaely	Purchasing Manager	Purchasing
3	Alexander	Khoo	Purchasing Clerk	Purchasing
3	Shelli	Baida	Purchasing Clerk	Purchasing
3	Sigal	Tobias	Purchasing Clerk	Purchasing
3	Guy	Himuro	Purchasing Clerk	Purchasing
3	Karen	Colmenares	Purchasing Clerk	Purchasing

SQL LEFT JOIN clause

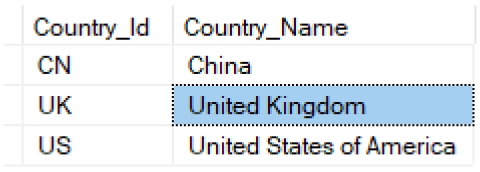




Write a Query :--

A. To query the country names of US, UK, and China

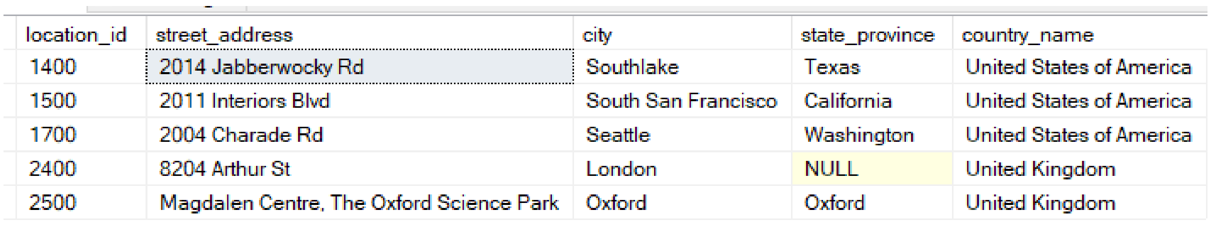
```
SELECT Country_Id, Country_Name FROM Countries

WHERE Country_ID IN ('US', 'UK','CN')
```



Country_Id	Country_Name
CN	China
UK	United Kingdom
US	United States of America

B. query retrieves the locations located in the US, UK and China:

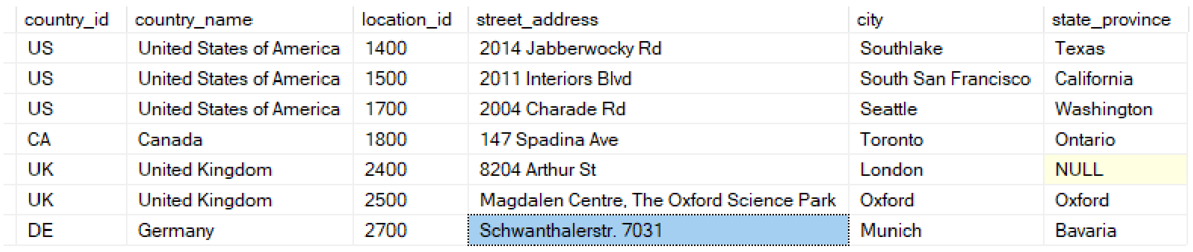
```
SELECT l.location_id, l.street_address, l.city, l.state_province,
c.country_name

FROM locations l

LEFT JOIN countries c ON l.country_id = c.country_id

WHERE c.country_id IN ('US', 'UK', 'CN');
```



location_id	street_address	city	state_province	country_name
1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
1500	2011 Interiors Blvd	South San Francisco	California	United States of America
1700	2004 Charade Rd	Seattle	Washington	United States of America
2400	8204 Arthur St	London	NULL	United Kingdom
2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

C. To join the countries table with the locations table

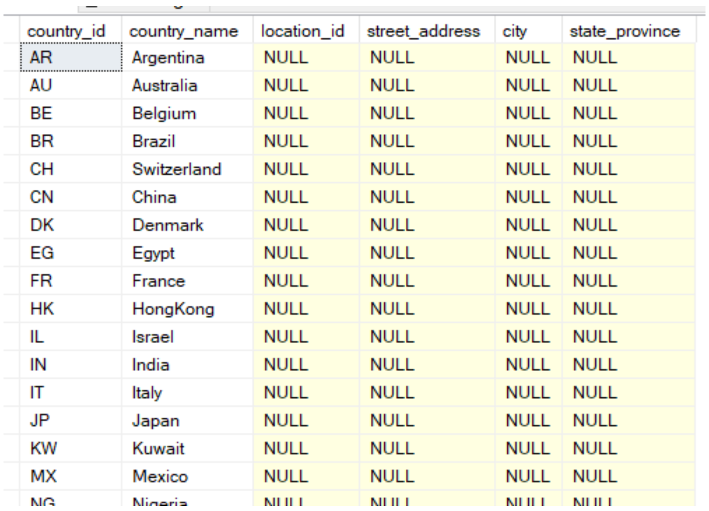
```
SELECT c.country_id, c.country_name, l.location_id, l.street_address,
l.city, l.state_province

FROM countries c

LEFT JOIN locations l ON c.country_id = l.country_id;
```



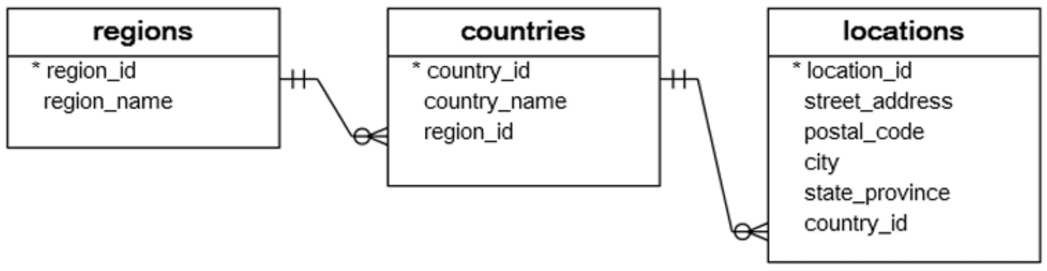
country_id	country_name	location_id	street_address	city	state_province
US	United States of America	1400	2014 Jabberwocky Rd	Southlake	Texas
US	United States of America	1500	2011 Interiors Blvd	South San Francisco	California
US	United States of America	1700	2004 Charade Rd	Seattle	Washington
CA	Canada	1800	147 Spadina Ave	Toronto	Ontario
UK	United Kingdom	2400	8204 Arthur St	London	NULL
UK	United Kingdom	2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford
DE	Germany	2700	Schwanthalerstr. 7031	Munich	Bavaria

D. to find the country that does not have any locations in the locations table

```
SELECT c.country_id, c.country_name, l.location_id, l.street_address,
l.city, l.state_province

FROM countries c

LEFT JOIN locations l ON c.country_id = l.country_id;

WHERE l.location_id IS NULL
```

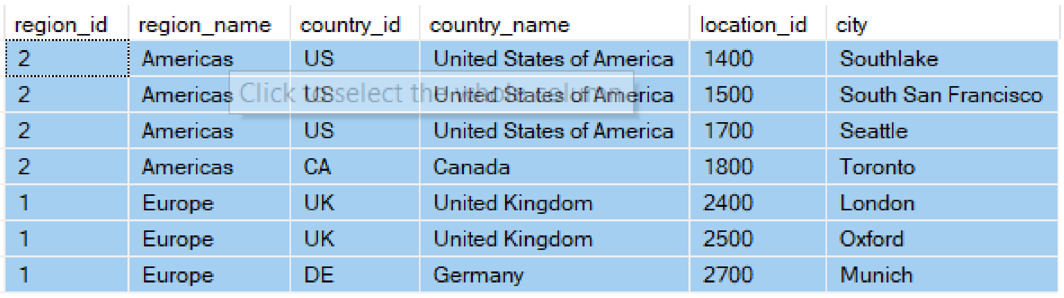


country_id	country_name	location_id	street_address	city	state_province
AR	Argentina	NULL	NULL	NULL	NULL
AU	Australia	NULL	NULL	NULL	NULL
BE	Belgium	NULL	NULL	NULL	NULL
BR	Brazil	NULL	NULL	NULL	NULL
CH	Switzerland	NULL	NULL	NULL	NULL
CN	China	NULL	NULL	NULL	NULL
DK	Denmark	NULL	NULL	NULL	NULL
EG	Egypt	NULL	NULL	NULL	NULL
FR	France	NULL	NULL	NULL	NULL
HK	HongKong	NULL	NULL	NULL	NULL
IL	Israel	NULL	NULL	NULL	NULL
IN	India	NULL	NULL	NULL	NULL
IT	Italy	NULL	NULL	NULL	NULL
JP	Japan	NULL	NULL	NULL	NULL
KW	Kuwait	NULL	NULL	NULL	NULL
MX	Mexico	NULL	NULL	NULL	NULL
NG	Nigeria	NULL	NULL	NULL	NULL



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

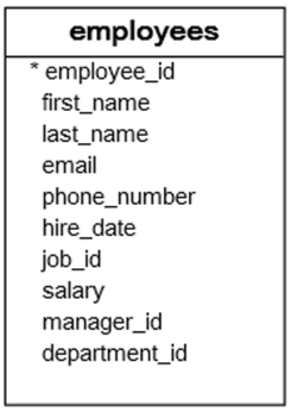
```
SELECT R.region_id, R.region_name, C.country_id, C.country_name,
       L.location_id, L.city
FROM regions R
INNER JOIN countries C
    ON R.region_id = C.region_id
INNER JOIN locations L
    ON C.country_id = L.country_id;
```



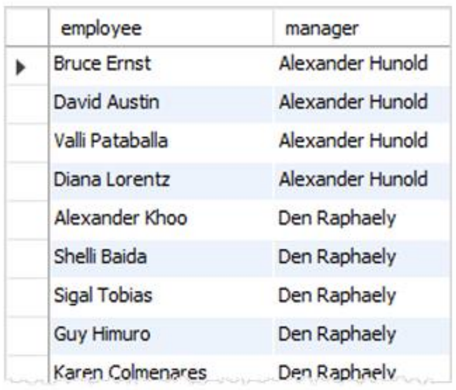
region_id	region_name	country_id	country_name	location_id	city
2	Americas	US	United States of America	1400	Southlake
2	Americas	US	United States of America	1500	South San Francisco
2	Americas	US	United States of America	1700	Seattle
2	Americas	CA	Canada	1800	Toronto
1	Europe	UK	United Kingdom	2400	London
1	Europe	UK	United Kingdom	2500	Oxford
1	Europe	DE	Germany	2700	Munich

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.



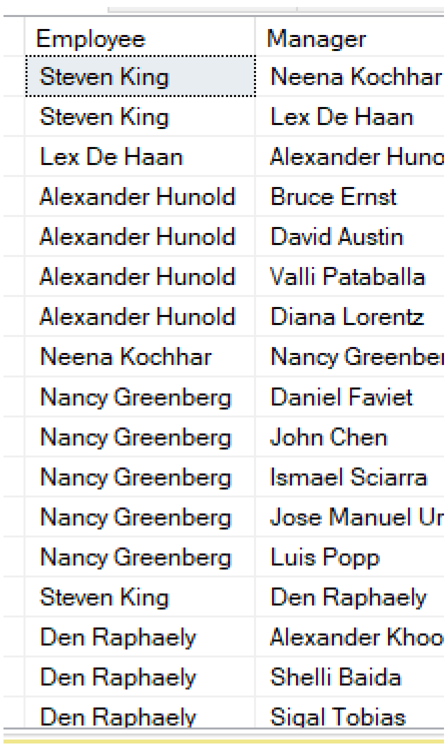
employee	manager
Bruce Ernst	Alexander Hunold
David Austin	Alexander Hunold
Valli Pataballa	Alexander Hunold
Diana Lorentz	Alexander Hunold
Alexander Khoo	Den Raphaely
Shelli Baida	Den Raphaely
Sigal Tobias	Den Raphaely
Guy Himuro	Den Raphaely
Karen Colmenares	Den Raphaely

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:-

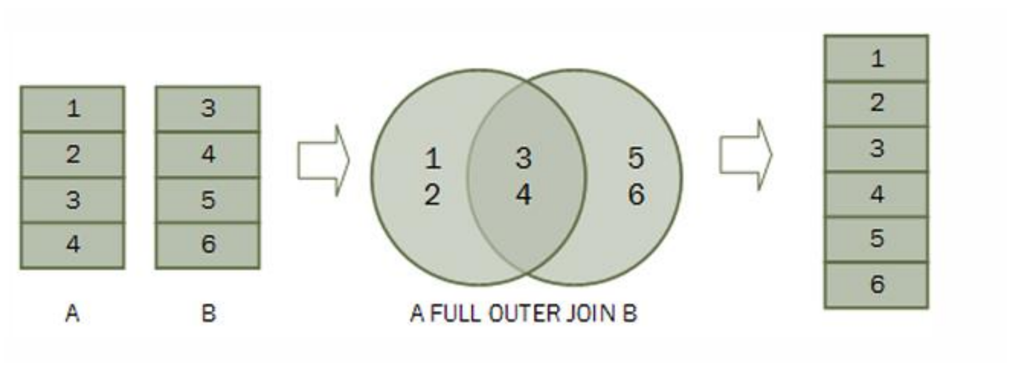
```
SELECT CONCAT(E.First_Name,' ', E.Last_Name) AS Employee,
    CONCAT(M.First_Name,' ', M.Last_Name) AS Manager
    FROM Employees E
    INNER JOIN
    EMPLOYEES M
    ON E.EMPLOYEE_ID = M.MANAGER_ID
```



Employee	Manager
Steven King	Neena Kochhar
Steven King	Lex De Haan
Lex De Haan	Alexander Huno
Alexander Hunold	Bruce Ernst
Alexander Hunold	David Austin
Alexander Hunold	Valli Pataballa
Alexander Hunold	Diana Lorentz
Neena Kochhar	Nancy Greenbe
Nancy Greenberg	Daniel Faviet
Nancy Greenberg	John Chen
Nancy Greenberg	Ismael Sciarra
Nancy Greenberg	Jose Manuel Ur
Nancy Greenberg	Luis Popp
Steven King	Den Raphaely
Den Raphaely	Alexander Khoo
Den Raphaely	Shelli Baida
Den Raphaely	Sigal Tobias

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 );
```

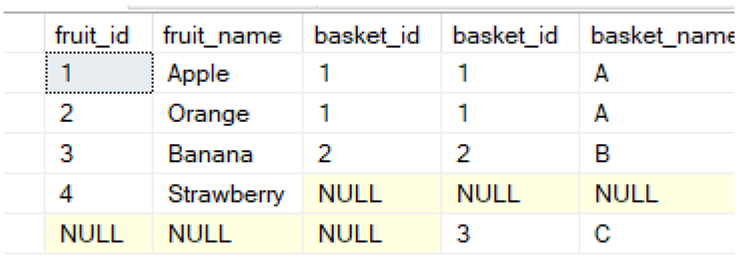
Second, insert some sample data into the baskets and fruits tables.

```
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.

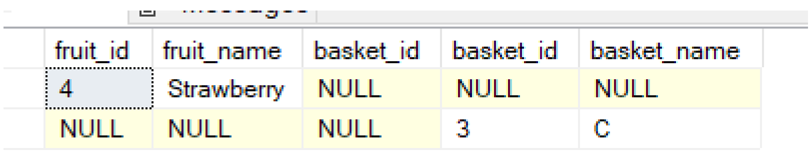
```
SELECT * FROM FRUITS F

FULL OUTER JOIN

BASKETS B

ON

F.BASKET_ID = B.BASKET_ID
```



fruit_id	fruit_name	basket_id	basket_id	basket_name
1	Apple	1	1	A
2	Orange	1	1	A
3	Banana	2	2	B
4	Strawberry	NULL	NULL	NULL
NULL	NULL	NULL	3	C

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

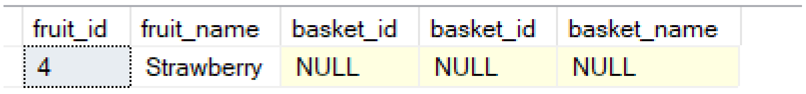
```
SELECT F.*, B.* FROM FRUITS F

FULL OUTER JOIN

BASKETS B

ON

F.BASKET_ID = B.BASKET_ID

WHERE F.BASKET_ID IS NULL
```



fruit_id	fruit_name	basket_id	basket_id	basket_name
4	Strawberry	NULL	NULL	NULL
NULL	NULL	NULL	3	C

C. Write a query which fruit is not in any basket

```
SELECT F.*, B.* FROM FRUITS F

FULL OUTER JOIN

BASKETS B

ON

F.BASKET_ID = B.BASKET_ID

WHERE B.BASKET_Id IS NULL
```



fruit_id	fruit_name	basket_id	basket_id	basket_name
4	Strawberry	NULL	NULL	NULL

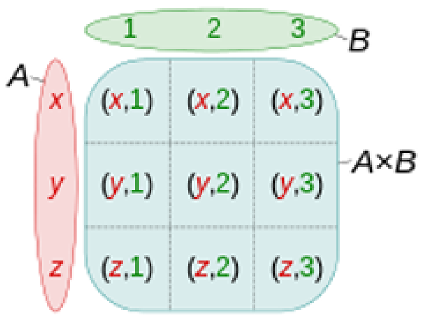
SQL CROSS JOIN clause

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:

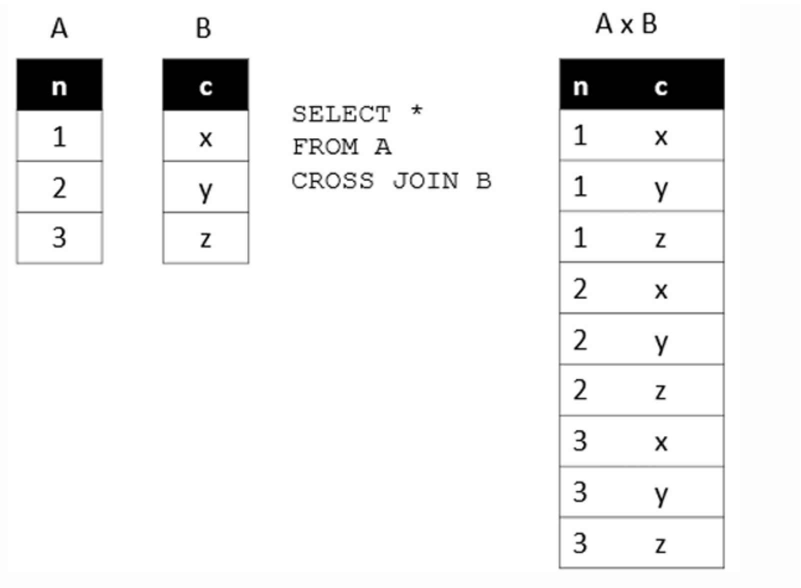


Similarly, in SQL, a Cartesian product of two tables A and B is a result set in which each row in the first table (A) is paired with each row in the second table (B). Suppose the A table has n rows and the B table has m rows, the result of the cross join of the A and B tables have n x m rows.

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:



A

n
1
2
3

B

c
x
y
z

```
SELECT *
FROM A
CROSS JOIN B
```

A x B

n	c
1	x
1	y
1	z
2	x
2	y
2	z
3	x
3	y
3	z

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');
```

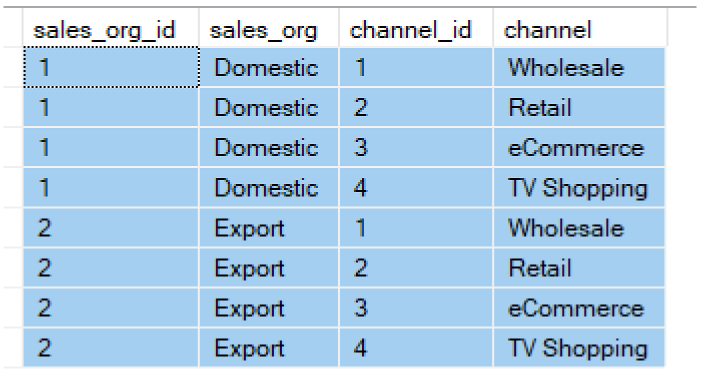
The company can distribute goods via various channels such as wholesale, retail, eCommerce, and TV shopping.

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

```
SELECT * FROM sales_organization

CROSS JOIN  Sales_Channel
```



sales_org_id	sales_org	channel_id	channel
1	Domestic	1	Wholesale
1	Domestic	2	Retail
1	Domestic	3	eCommerce
1	Domestic	4	TV Shopping
2	Export	1	Wholesale
2	Export	2	Retail
2	Export	3	eCommerce
2	Export	4	TV Shopping

TASK 4

SQL GROUP BY clause

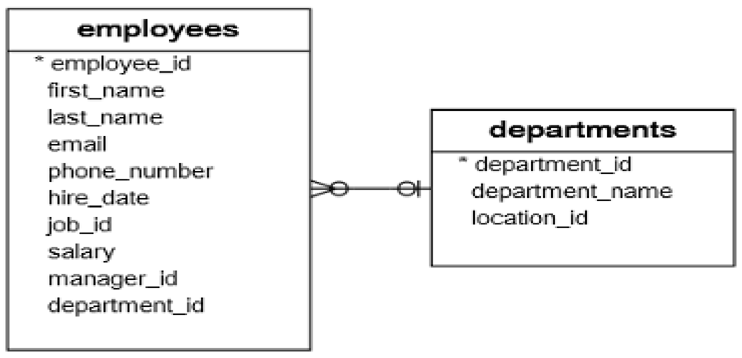
The GROUP BY is an optional clause of the SELECT statement. The GROUP BY clause allows you to group rows based on values of one or more columns. It returns one row for each group.

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.

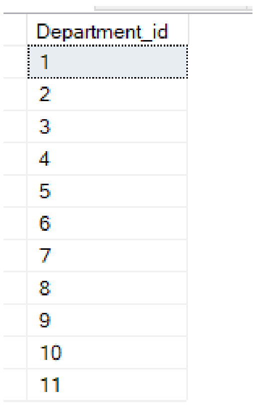
We will use the employees and departments tables to demonstrate how the GROUP BY clause works.



Questions

A. to group the values in department_id column of the employees table:

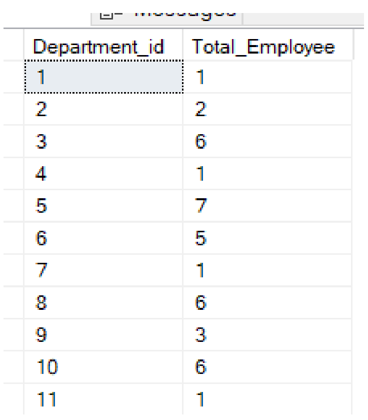
```
SELECT Department_id

FROM EMPLOYEES

GROUP BY Department_ID
```



	Department_id
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11

B. to count the number of employees by department:

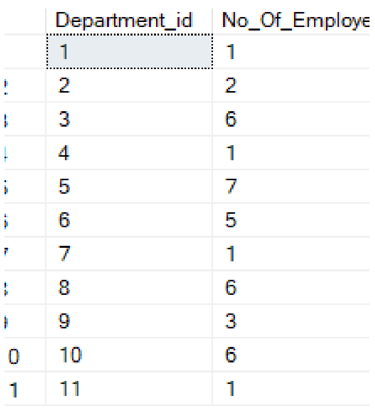
```
SELECT Department_id, COUNT(*) AS Total_Employee

FROM EMPLOYEES

GROUP BY Department_ID
```



	Department_id	Total_Employee
	1	1
	2	2
	3	6
	4	1
	5	7
	6	5
	7	1
	8	6
	9	3
	10	6
	11	1

C. returns the number of employees by department

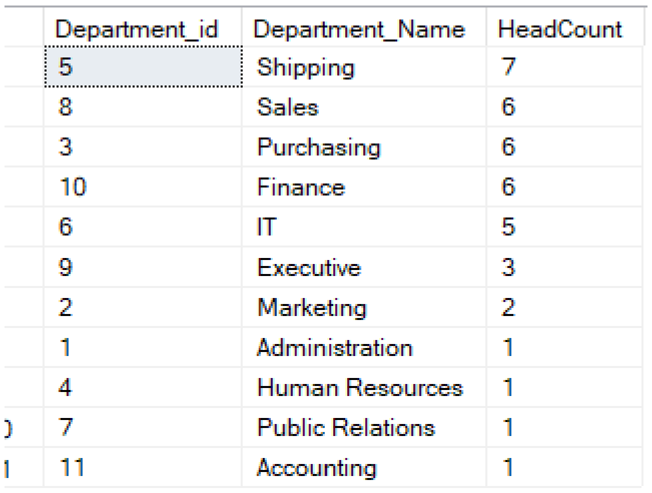
```
SELECT Department_id, COUNT(*) AS No_Of_Employee

FROM EMPLOYEES

GROUP BY Department_ID
```



Department_id	No_Of_Employe
1	1
2	2
3	6
4	1
5	7
6	5
7	1
8	6
9	3
10	6
11	1

D. to sort the departments by headcount:

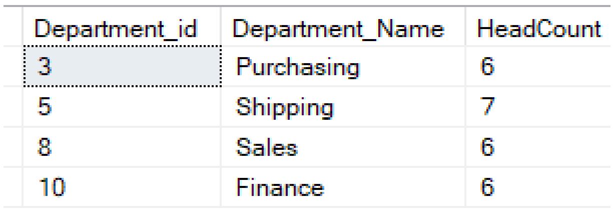
```
SELECT D.Department_id, D.Department_Name,

COUNT(e.employee_ID) AS HeadCount

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name

ORDER BY HeadCount DESC
```



Department_id	Department_Name	HeadCount
5	Shipping	7
8	Sales	6
3	Purchasing	6
10	Finance	6
6	IT	5
9	Executive	3
2	Marketing	2
1	Administration	1
4	Human Resources	1
7	Public Relations	1
11	Accounting	1

E. to find departments with headcounts are greater than 5:

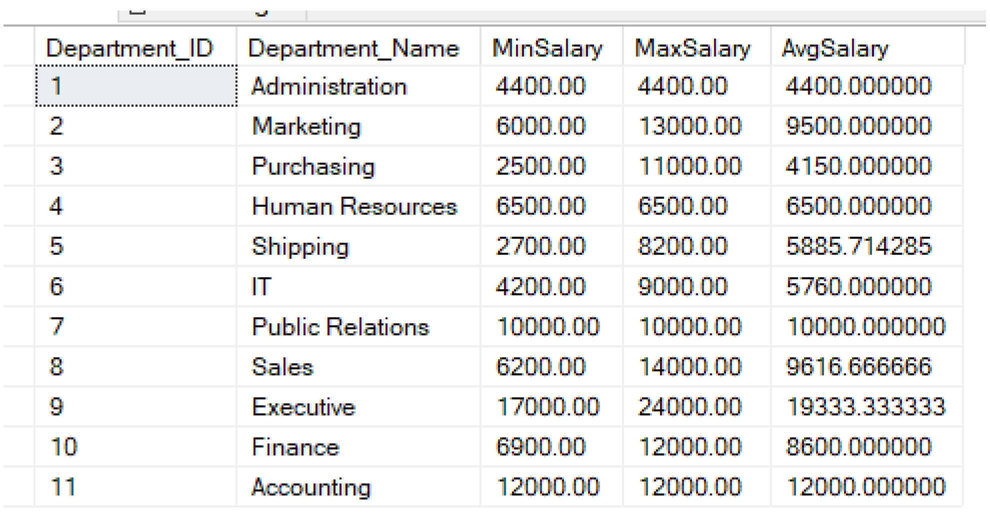
```
SELECT D.Department_id, D.Department_Name,

COUNT(e.employee_ID) AS HeadCount

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name

HAVING COUNT(e.employee_ID) > 5
```



Department_id	Department_Name	HeadCount
3	Purchasing	6
5	Shipping	7
8	Sales	6
10	Finance	6

F. returns the minimum, maximum and average salary of employees in each department.

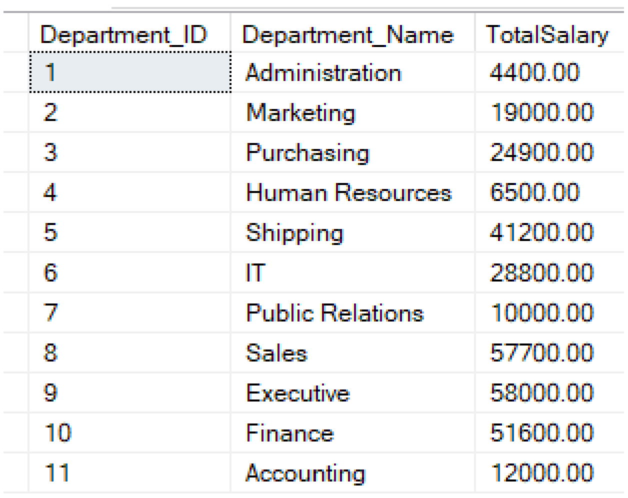
```
SELECT D.Department_ID, D.Department_Name, MIN(E.Salary) AS
MinSalary,

MAX(E.Salary) AS MaxSalary, AVG(E.Salary) AS AvgSalary

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name
```



	Department_ID	Department_Name	MinSalary	MaxSalary	AvgSalary
	1	Administration	4400.00	4400.00	4400.000000
	2	Marketing	6000.00	13000.00	9500.000000
	3	Purchasing	2500.00	11000.00	4150.000000
	4	Human Resources	6500.00	6500.00	6500.000000
	5	Shipping	2700.00	8200.00	5885.714285
	6	IT	4200.00	9000.00	5760.000000
	7	Public Relations	10000.00	10000.00	10000.000000
	8	Sales	6200.00	14000.00	9616.666666
	9	Executive	17000.00	24000.00	19333.333333
	10	Finance	6900.00	12000.00	8600.000000
	11	Accounting	12000.00	12000.00	12000.000000

G. To get the total salary per department,

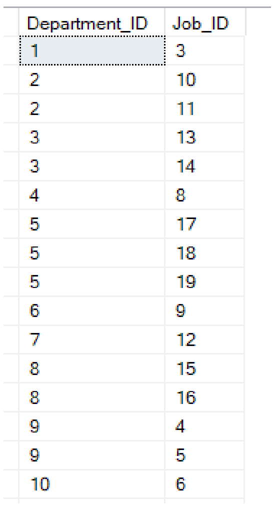
```
SELECT D.Department_ID, D.Department_Name, SUM(E.Salary) AS
TotalSalary

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name
```



	Department_ID	Department_Name	TotalSalary
	1	Administration	4400.00
	2	Marketing	19000.00
	3	Purchasing	24900.00
	4	Human Resources	6500.00
	5	Shipping	41200.00
	6	IT	28800.00
	7	Public Relations	10000.00
	8	Sales	57700.00
	9	Executive	58000.00
	10	Finance	51600.00
	11	Accounting	12000.00

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

```
SELECT Department_ID, Job_ID

FROM EMPLOYEES

GROUP BY DEPARTMENT_ID, JOB_ID;
```



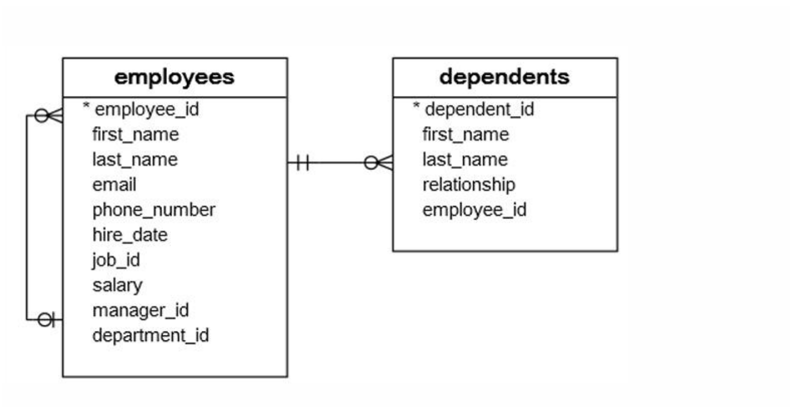
Department_ID	Job_ID
1	3
2	10
2	11
3	13
3	14
4	8
5	17
5	18
5	19
6	9
7	12
8	15
8	16
9	4
9	5
10	6

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:

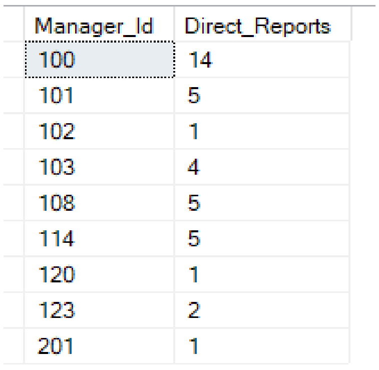
```
SELECT
        column1, column2,
        AGGREGATE_FUNCTION (column3)
FROM table1
GROUP BY
        column1, column2
HAVING
       group_condition;
```



Questions:-

A. To get the managers and their direct reports, and to group employees by the managers and to count the direct reports.

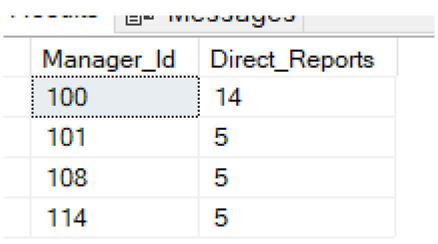
```
SELECT Manager_Id, Count(*) Direct_Reports

FROM Employees

where manager_id  IS NOT NULL

GROUP BY Manager_Id
```



Manager_Id	Direct_Reports
100	14
101	5
102	1
103	4
108	5
114	5
120	1
123	2
201	1

B. To find the managers who have at least five direct reports

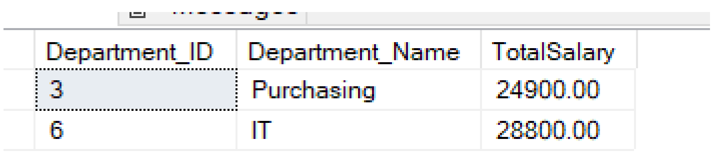
```
SELECT Manager_Id, Count(*) Direct_Reports

FROM Employees

where manager_id  IS NOT NULL

GROUP BY Manager_iD

HAVING COUNT(*) >= 5
```



	Manager_Id	Direct_Reports
	100	14
	101	5
	108	5
	114	5

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.

```
SELECT D.Department_ID, D.Department_Name, SUM(E.Salary) AS
TotalSalary

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name

HAVING SUM(E.Salary) BETWEEN 20000 AND 30000
```

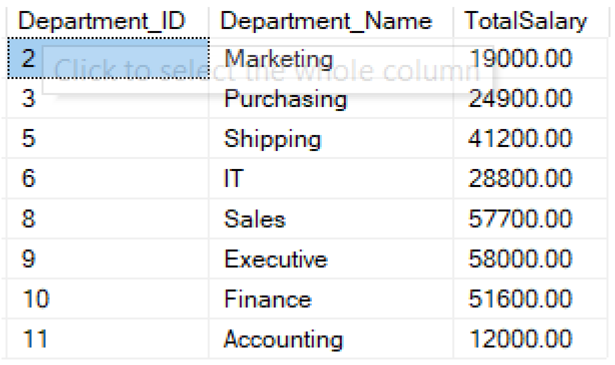


	Department_ID	Department_Name	TotalSalary
	3	Purchasing	24900.00
	6	IT	28800.00

D. To find the department that has employees with the lowest salary greater than 10000

```
SELECT D.Department_ID, D.Department_Name, SUM(E.Salary) AS
TotalSalary

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON
```

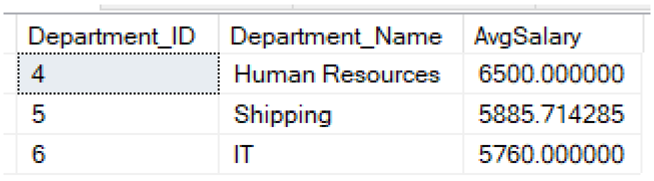
```
E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name

HAVING SUM(E.Salary) > 10000
```



Department_ID	Department_Name	TotalSalary
2	Marketing	19000.00
3	Purchasing	24900.00
5	Shipping	41200.00
6	IT	28800.00
8	Sales	57700.00
9	Executive	58000.00
10	Finance	51600.00
11	Accounting	12000.00

E. To find the departments that have the average salaries of employees between 5000 and 7000

```
SELECT D.Department_ID, D.Department_Name, AVG(E.Salary) AS
TotalSalary

FROM EMPLOYEES E

RIGHT JOIN

Departments D

ON

E.Department_ID = D.Department_ID

GROUP BY D.Department_ID, D.Department_Name

HAVING AVG(E.Salary) BETWEEN 5000 AND 7000
```



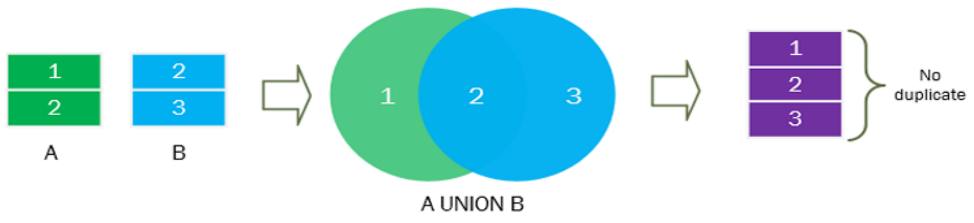
Department_ID	Department_Name	AvgSalary
4	Human Resources	6500.000000
5	Shipping	5885.714285
6	IT	5760.000000

TASK 5

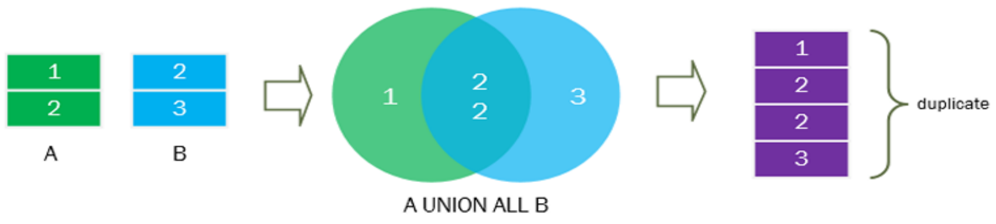
(Other Queries)

1)SQL UNION operator

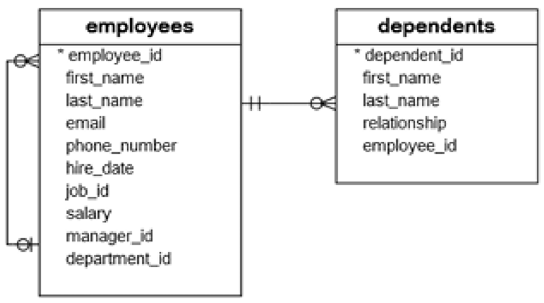
Suppose, we have two result sets A(1,2) and B(2,3). The following picture illustrates A UNION B:



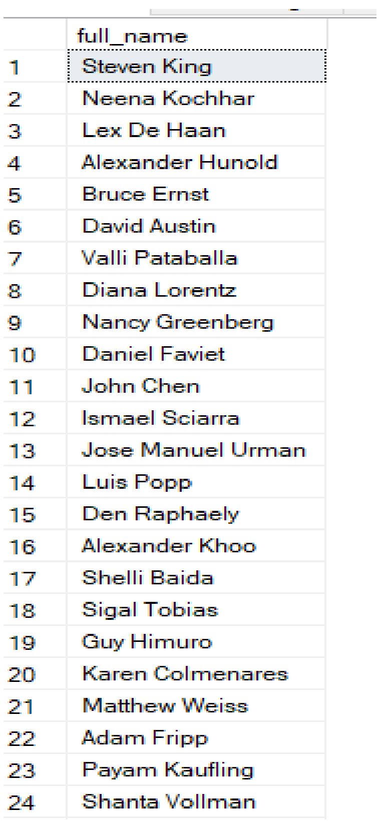
And the following picture illustrates A UNION ALL B



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



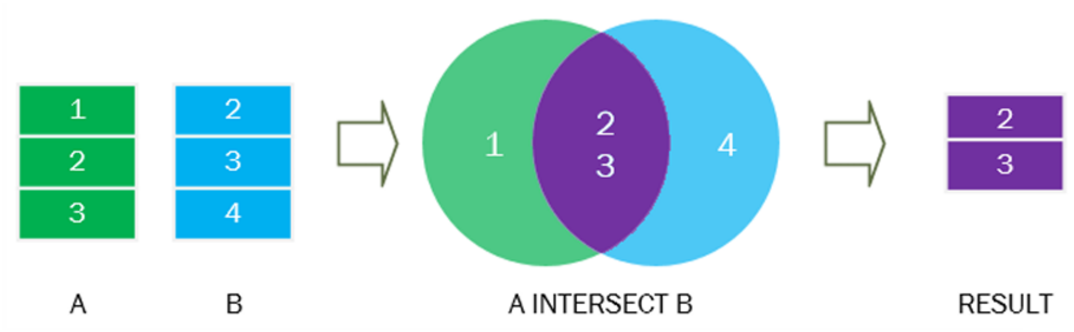
```
SELECT  first_name + ' ' + last_name AS full_name

FROM employees

UNION ALL

SELECT

 first_name + ' ' + last_name AS full_name

FROM dependents;
```



```
    full_name
1   Steven King
2   Neena Kochhar
3   Lex De Haan
4   Alexander Hunold
5   Bruce Ernst
6   David Austin
7   Valli Pataballa
8   Diana Lorentz
9   Nancy Greenberg
10  Daniel Faviet
11  John Chen
12  Ismael Sciarra
13  Jose Manuel Urman
14  Luis Popp
15  Den Raphaely
16  Alexander Khoo
17  Shelli Baida
18  Sigal Tobias
19  Guy Himuro
20  Karen Colmenares
21  Matthew Weiss
22  Adam Fripp
23  Payam Kaufling
24  Shanta Vollman
```

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.

```
SELECT id FROM A

INTERSECT

SELECT id FROM B

ORDER BY id DESC;
```

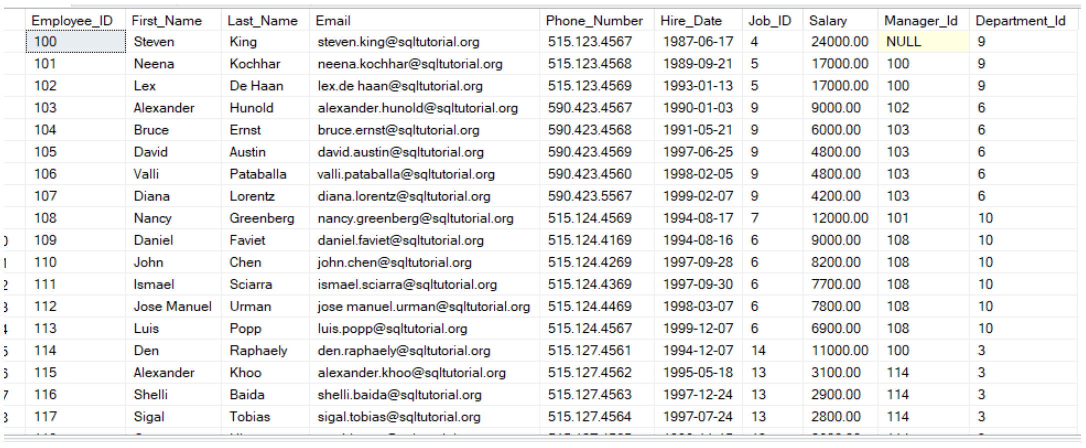
3)SQL EXISTS operator

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.

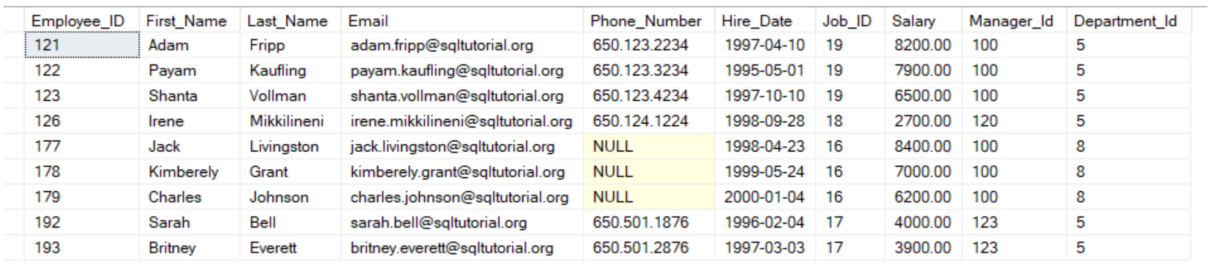
```
SELECT *
FROM employees e
WHERE EXISTS (
  SELECT 1
  FROM dependents d
  WHERE d.employee_id = e.employee_id );
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
104	Bruce	Ernst	bruce.ernst@sqltutorial.org	590.423.4568	1991-05-21	9	6000.00	103	6
105	David	Austin	david.austin@sqltutorial.org	590.423.4569	1997-06-25	9	4800.00	103	6
106	Valli	Pataballa	valli.pataballa@sqltutorial.org	590.423.4560	1998-02-05	9	4800.00	103	6
107	Diana	Lorentz	diana.lorentz@sqltutorial.org	590.423.5567	1999-02-07	9	4200.00	103	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7700.00	108	10
112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	7800.00	108	10
113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	6900.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
115	Alexander	Khoo	alexander.khoo@sqltutorial.org	515.127.4562	1995-05-18	13	3100.00	114	3
116	Shelli	Baida	shelli.baida@sqltutorial.org	515.127.4563	1997-12-24	13	2900.00	114	3
117	Sigal	Tobias	sigal.tobias@sqltutorial.org	515.127.4564	1997-07-24	13	2800.00	114	3

B . finds employees who do not have any dependents:

```
SELECT *
FROM employees e
WHERE NOT EXISTS (
  SELECT 1
  FROM dependents d
  WHERE d.employee_id = e.employee_id );
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
122	Payam	Kaufling	payam.kaufling@sqltutorial.org	650.123.3234	1995-05-01	19	7900.00	100	5
123	Shanta	Vollman	shanta.vollman@sqltutorial.org	650.123.4234	1997-10-10	19	6500.00	100	5
126	Irene	Mikkilineni	irene.mikkilineni@sqltutorial.org	650.124.1224	1998-09-28	18	2700.00	120	5
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
178	Kimberely	Grant	kimberely.grant@sqltutorial.org	NULL	1999-05-24	16	7000.00	100	8
179	Charles	Johnson	charles.johnson@sqltutorial.org	NULL	2000-01-04	16	6200.00	100	8
192	Sarah	Bell	sarah.bell@sqltutorial.org	650.501.1876	1996-02-04	17	4000.00	123	5
193	Britney	Everett	britney.everett@sqltutorial.org	650.501.2876	1997-03-03	17	3900.00	123	5

4) SQL CASE expression

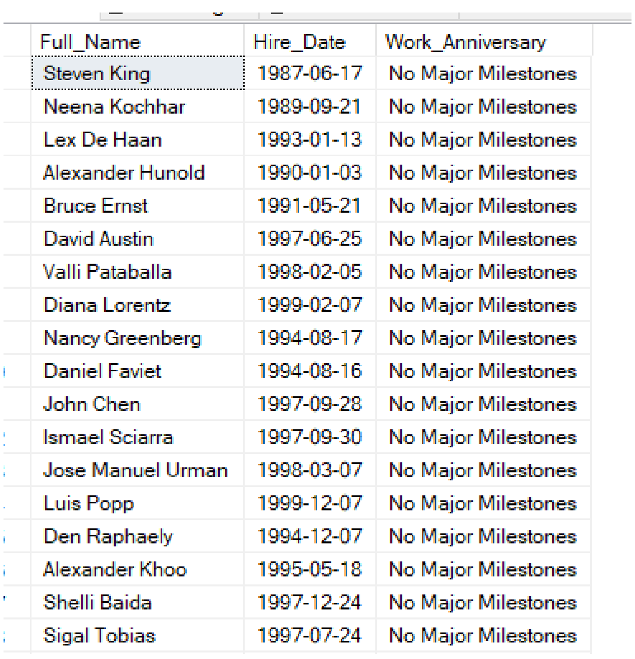
```
CASE expression
WHEN when_expression_1 THEN
     result_1
WHEN when_expression_2 THEN
    result_2
WHEN when_expression_3 THEN
    result_3
...
ELSE
    else_result
END
```

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

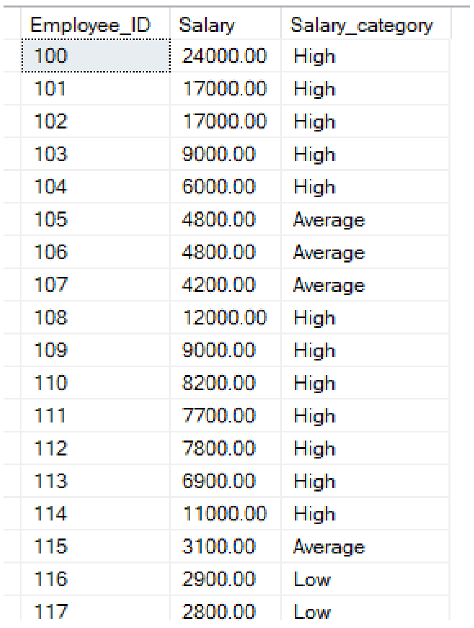
```
SELECT CONCAT(First_Name, ' ', Last_Name) AS Full_Name, Hire_Date,
                    CASE DATEPART(YEAR, 2020-01-01) - DATEPART(YEAR,
HIRE_DATE)
                      WHEN 1 THEN '1 Year Anniversary'
                      WHEN 5 THEN '5 Year Anniversary'
                      WHEN 10 THEN '10 Year Anniversary'
                      ELSE 'No Major Milestones'
                    END AS Work_Anniversary
FROM EMPLOYEES
```



Full_Name	Hire_Date	Work_Anniversary
Steven King	1987-06-17	No Major Milestones
Neena Kochhar	1989-09-21	No Major Milestones
Lex De Haan	1993-01-13	No Major Milestones
Alexander Hunold	1990-01-03	No Major Milestones
Bruce Ernst	1991-05-21	No Major Milestones
David Austin	1997-06-25	No Major Milestones
Valli Pataballa	1998-02-05	No Major Milestones
Diana Lorentz	1999-02-07	No Major Milestones
Nancy Greenberg	1994-08-17	No Major Milestones
Daniel Faviet	1994-08-16	No Major Milestones
John Chen	1997-09-28	No Major Milestones
Ismael Sciarra	1997-09-30	No Major Milestones
Jose Manuel Urman	1998-03-07	No Major Milestones
Luis Popp	1999-12-07	No Major Milestones
Den Raphaely	1994-12-07	No Major Milestones
Alexander Khoo	1995-05-18	No Major Milestones
Shelli Baida	1997-12-24	No Major Milestones
Sigal Tobias	1997-07-24	No Major Milestones

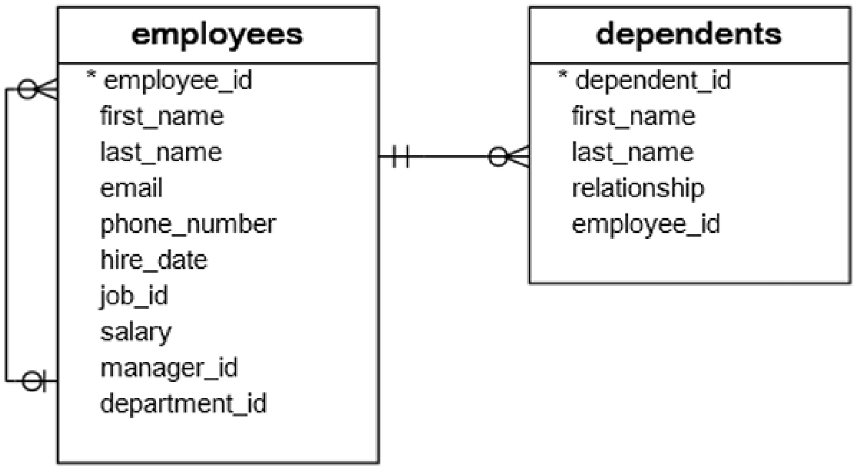
B. Write a Query If the salary is less than 3000, the CASE expression returns "Low". If the salary is between 3000 and 5000, it returns "average". When the salary is greater than 5000, the CASE expression returns "High".

```
SELECT Employee_ID, Salary,

                    CASE

                      WHEN Salary<3000 THEN 'Low'

                      WHEN Salary BETWEEN 3000 and 5000 THEN
'Average'

                      WHEN Salary>5000 THEN 'High'

                      ELSE 'Other'

                    END AS Salary_category

FROM EMPLOYEES
```

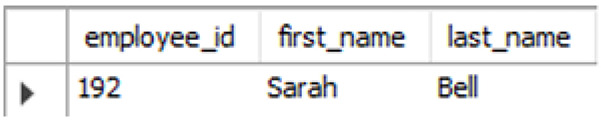


Employee_ID	Salary	Salary_category
100	24000.00	High
101	17000.00	High
102	17000.00	High
103	9000.00	High
104	6000.00	High
105	4800.00	Average
106	4800.00	Average
107	4200.00	Average
108	12000.00	High
109	9000.00	High
110	8200.00	High
111	7700.00	High
112	7800.00	High
113	6900.00	High
114	11000.00	High
115	3100.00	Average
116	2900.00	Low
117	2800.00	Low

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.

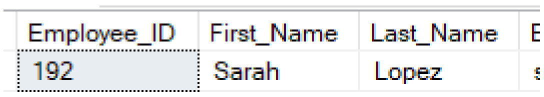


employee_id	first_name	last_name
192	Sarah	Bell

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,

```
UPDATE  Employees

SET last_name = 'Lopez'

WHERE Employee_ID = 192


SELECT * FROM Employees

WHERE Employee_ID = 192
```



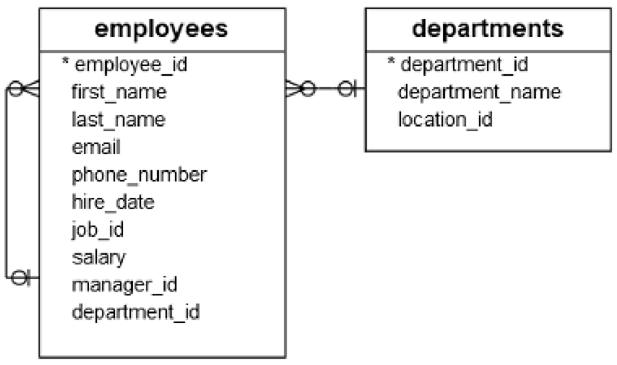
Employee_ID	First_Name	Last_Name	E
192	Sarah	Lopez	s

FINAL TASK

(ADVANCED QUERIES)

SQL SUBQUERY

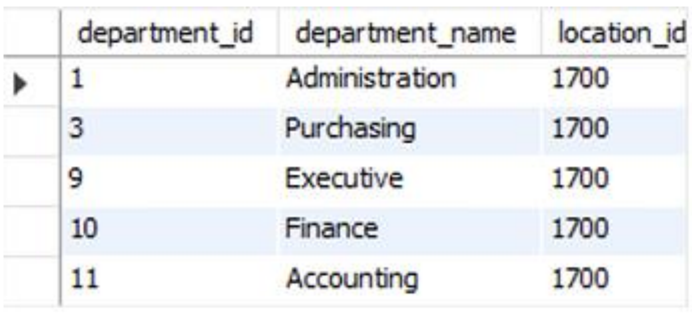
Consider the following employees and departments tables from the sample database



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:

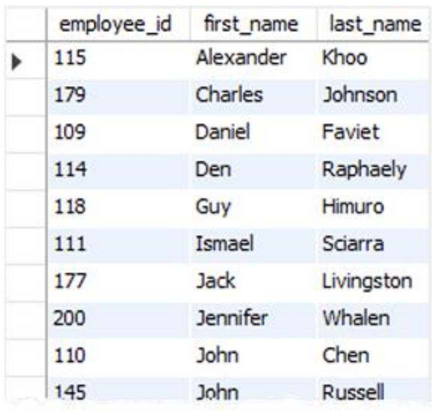
```
SELECT * FROM departments

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

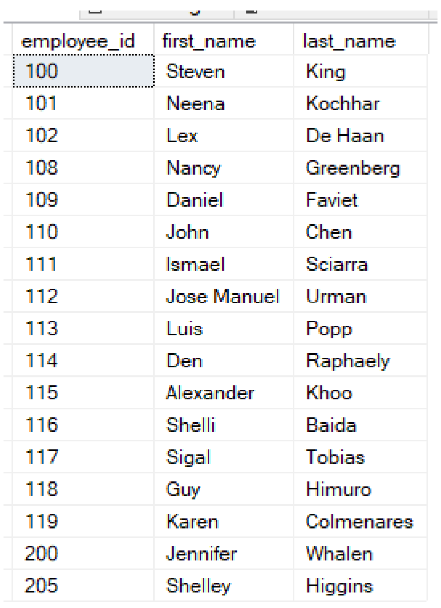
A much better solution to this problem is to use a subquery. By definition, a subquery is a query nested inside another query such as SELECT, INSERT, UPDATE, or DELETE statement. In this tutorial, we are focusing on the subquery used with the SELECT statement.

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

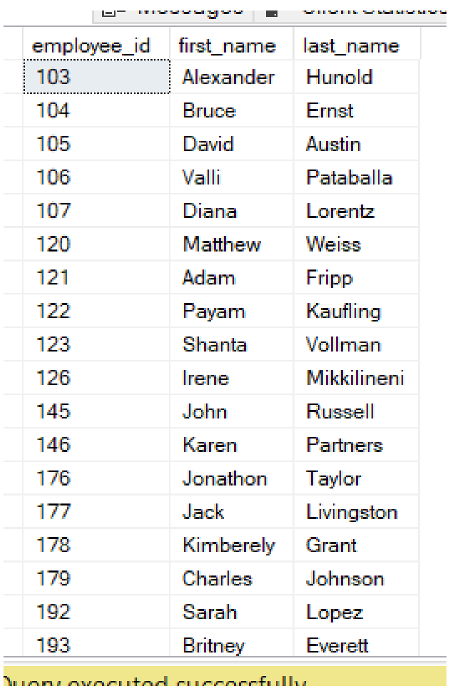
```
SELECT  employee_id, first_name, last_name

FROM  employees

WHERE Department_ID IN ( SELECT  Department_ID

FROM

departments

WHERE location_id = 1700 )
```



employee_id	first_name	last_name
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
108	Nancy	Greenberg
109	Daniel	Faviet
110	John	Chen
111	Ismael	Sciarra
112	Jose Manuel	Urman
113	Luis	Popp
114	Den	Raphaely
115	Alexander	Khoo
116	Shelli	Baida
117	Sigal	Tobias
118	Guy	Himuro
119	Karen	Colmenares
200	Jennifer	Whalen
205	Shelley	Higgins

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

```
SELECT  employee_id, first_name, last_name

FROM  employees

WHERE Department_ID IN ( SELECT  Department_ID
FROM

departments

WHERE location_id != 1700 )
```

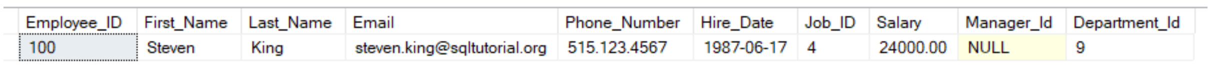


employee_id	first_name	last_name
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Pataballa
107	Diana	Lorentz
120	Matthew	Weiss
121	Adam	Fripp
122	Payam	Kaufling
123	Shanta	Vollman
126	Irene	Mikkilineni
145	John	Russell
146	Karen	Partners
176	Jonathon	Taylor
177	Jack	Livingston
178	Kimberely	Grant
179	Charles	Johnson
192	Sarah	Lopez
193	Britney	Everett

Query executed successfully

C. finds the employees who have the highest salary:

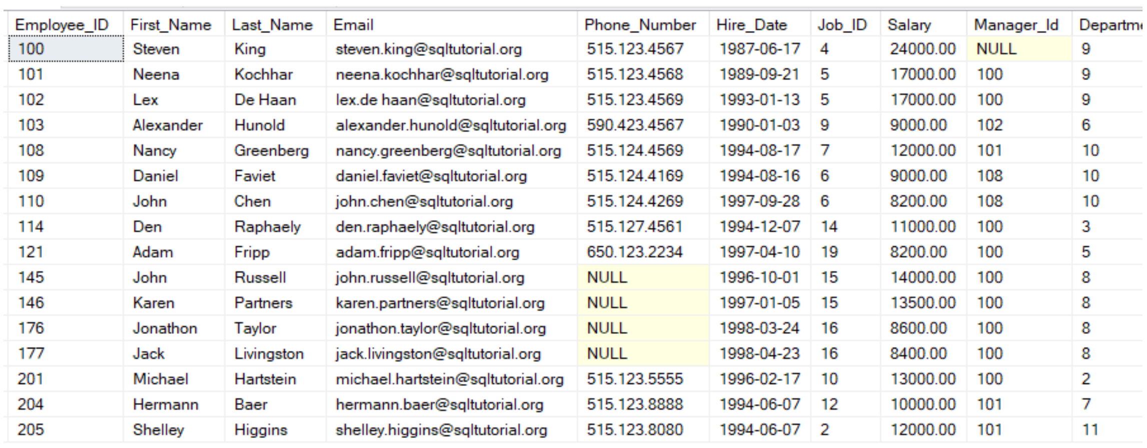
```
SELECT * FROM Employees

WHERE SALARY = (SELECT MAX(SALARY) as MaxSalary FROM Employees)
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9

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

```
SELECT * FROM Employees

WHERE SALARY >

(SELECT AVG(SALARY) FROM Employees)
```

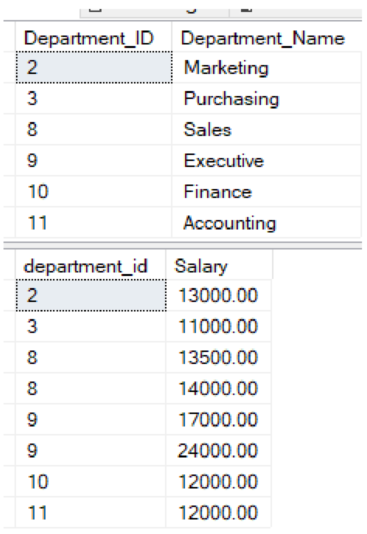


Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Departm
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9
101	Neena	Kochhar	neena.kochhar@sqltutorial.org	515.123.4568	1989-09-21	5	17000.00	100	9
102	Lex	De Haan	lex.de haan@sqltutorial.org	515.123.4569	1993-01-13	5	17000.00	100	9
103	Alexander	Hunold	alexander.hunold@sqltutorial.org	590.423.4567	1990-01-03	9	9000.00	102	6
108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12000.00	101	10
109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9000.00	108	10
110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8200.00	108	10
114	Den	Raphaely	den.raphaely@sqltutorial.org	515.127.4561	1994-12-07	14	11000.00	100	3
121	Adam	Fripp	adam.fripp@sqltutorial.org	650.123.2234	1997-04-10	19	8200.00	100	5
145	John	Russell	john.russell@sqltutorial.org	NULL	1996-10-01	15	14000.00	100	8
146	Karen	Partners	karen.partners@sqltutorial.org	NULL	1997-01-05	15	13500.00	100	8
176	Jonathon	Taylor	jonathon.taylor@sqltutorial.org	NULL	1998-03-24	16	8600.00	100	8
177	Jack	Livingston	jack.livingston@sqltutorial.org	NULL	1998-04-23	16	8400.00	100	8
201	Michael	Hartstein	michael.hartstein@sqltutorial.org	515.123.5555	1996-02-17	10	13000.00	100	2
204	Hermann	Baer	hermann.baer@sqltutorial.org	515.123.8888	1994-06-07	12	10000.00	101	7
205	Shelley	Higgins	shelley.higgins@sqltutorial.org	515.123.8080	1994-06-07	2	12000.00	101	11

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

```
SELECT Department_ID, Department_Name FROM Departments

WHERE  Department_ID IN (

SELECT Department_ID FROM Employees

WHERE SALARY > 10000)
```

```
SELECT DISTINCT department_id, Salary

FROM employees

WHERE salary > 10000;
```

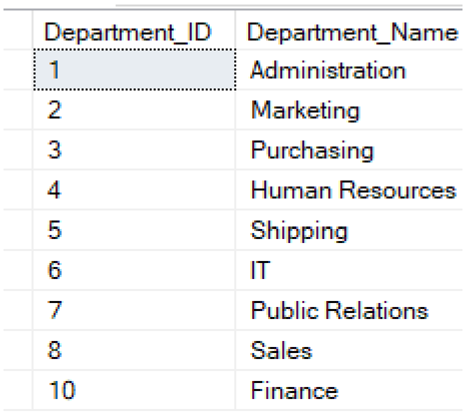


Department_ID	Department_Name
2	Marketing
3	Purchasing
8	Sales
9	Executive
10	Finance
11	Accounting

department_id	Salary
2	13000.00
3	11000.00
8	13500.00
8	14000.00
9	17000.00
9	24000.00
10	12000.00
11	12000.00

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

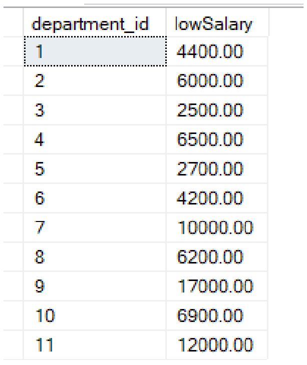
```
SELECT Department_ID, Department_Name FROM Departments

WHERE  Department_ID IN (

SELECT Department_ID FROM Employees

WHERE SALARY !> 10000)
```



	Department_ID	Department_Name
	1	Administration
	2	Marketing
	3	Purchasing
	4	Human Resources
	5	Shipping
	6	IT
	7	Public Relations
	8	Sales
	10	Finance

G. to find the lowest salary by department:

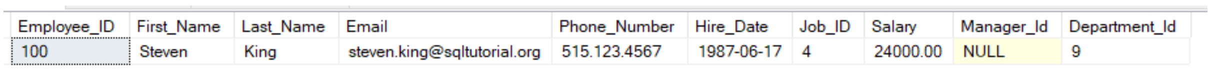
```
SELECT DISTINCT department_id, MIN(Salary) as lowSalary

FROM employees

GROUP BY Department_ID
```



	department_id	lowSalary
	1	4400.00
	2	6000.00
	3	2500.00
	4	6500.00
	5	2700.00
	6	4200.00
	7	10000.00
	8	6200.00
	9	17000.00
	10	6900.00
	11	12000.00

H. finds all employees whose salaries are greater than the lowest salary of every department:

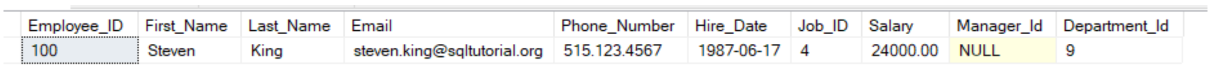
```
SELECT * FROM employees
WHERE salary > ALL (
  SELECT MIN(salary)
  FROM employees
  GROUP BY department_id )
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9

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

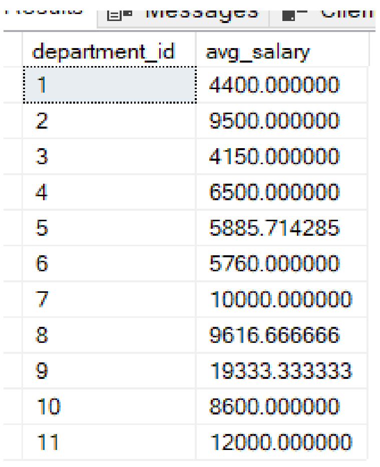
```
SELECT * FROM employees
WHERE salary >= ALL (
  SELECT MAX(salary)
  FROM employees
  GROUP BY department_id );
```



Employee_ID	First_Name	Last_Name	Email	Phone_Number	Hire_Date	Job_ID	Salary	Manager_Id	Department_Id
100	Steven	King	steven.king@sqltutorial.org	515.123.4567	1987-06-17	4	24000.00	NULL	9

J. returns the average salary of every department

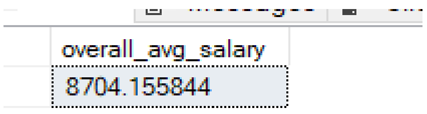
```
SELECT department_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY department_id;
```



department_id	avg_salary
1	4400.000000
2	9500.000000
3	4150.000000
4	6500.000000
5	5885.714285
6	5760.000000
7	10000.000000
8	9616.666666
9	19333.333333
10	8600.000000
11	12000.000000

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

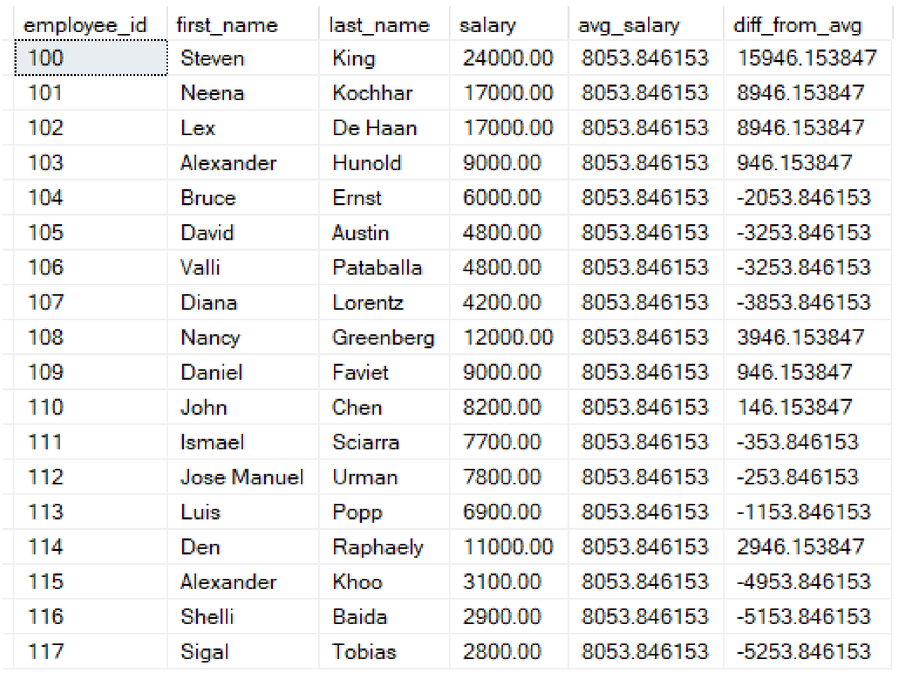
```
SELECT AVG(dept_avg) AS overall_avg_salary
FROM (
  SELECT AVG(salary) AS dept_avg
  FROM employees
  GROUP BY department_id
) AS avg_per_dept;
```



overall_avg_salary
8704.155844

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

```
SELECT employee_id, first_name, last_name, salary,
  (SELECT AVG(salary) FROM employees) AS avg_salary,
  salary - (SELECT AVG(salary) FROM employees) AS diff_from_avg
FROM employees;
```



employee_id	first_name	last_name	salary	avg_salary	diff_from_avg
100	Steven	King	24000.00	8053.846153	15946.153847
101	Neena	Kochhar	17000.00	8053.846153	8946.153847
102	Lex	De Haan	17000.00	8053.846153	8946.153847
103	Alexander	Hunold	9000.00	8053.846153	946.153847
104	Bruce	Ernst	6000.00	8053.846153	-2053.846153
105	David	Austin	4800.00	8053.846153	-3253.846153
106	Valli	Pataballa	4800.00	8053.846153	-3253.846153
107	Diana	Lorentz	4200.00	8053.846153	-3853.846153
108	Nancy	Greenberg	12000.00	8053.846153	3946.153847
109	Daniel	Faviet	9000.00	8053.846153	946.153847
110	John	Chen	8200.00	8053.846153	146.153847
111	Ismael	Sciarra	7700.00	8053.846153	-353.846153
112	Jose Manuel	Urman	7800.00	8053.846153	-253.846153
113	Luis	Popp	6900.00	8053.846153	-1153.846153
114	Den	Raphaely	11000.00	8053.846153	2946.153847
115	Alexander	Khoo	3100.00	8053.846153	-4953.846153
116	Shelli	Baida	2900.00	8053.846153	-5153.846153
117	Sigal	Tobias	2800.00	8053.846153	-5253.846153

LEARNING OUTCOMES

By completing the HRDMS, participants will:

- Master the principles of data normalization and relationship management in SQL databases
- Develop expertise in transforming business requirements into resilient database solutions
- Enhance their proficiency in writing and optimizing advanced SQL queries for real-world HR analytics
- Acquire skills necessary for database maintenance, troubleshooting, and evolution in a live business context

ACTIONABLE

This HR database project offers several actionable insights for real-world HR and business operations. By using SQL queries on well-structured tables like employees, departments, jobs, and dependents, one can quickly retrieve key information such as department-wise salary expenses, employee distribution across locations, or managers with large teams. These insights can help HR teams make informed decisions like identifying departments with overstaffing, underpaid or overpaid employees, or locations that may require more recruitment. Additionally, by using subqueries and joins instead of hardcoded department IDs or salary values, the system becomes scalable and dynamic. This means HR managers or analysts won't need to modify the SQL code every time there is a data change — the system will always fetch up-to-date insights based on real-time data.

METHODOLOGIES

The project was built using **Microsoft SQL Server** as the core RDBMS, and all operations were executed using **SQL Server Management Studio (SSMS)**. The database schema was designed using standard **relational database design principles**, with primary and foreign key constraints to ensure data integrity. SQL was used in a modular way — using **Data Definition Language (DDL)** commands to create the structure (tables and relationships) and **Data Manipulation Language (DML)** for inserting and querying data. Throughout the project, different SQL features were utilized, including JOIN clauses (INNER, LEFT, FULL OUTER, SELF JOIN), GROUP BY and HAVING clauses for aggregation, WHERE filters for conditions, CASE expressions for logic-based outputs, and SUBQUERIES for dynamic filtering. These methodologies ensured that the project was not only functionally complete but also aligned with best practices in database development and querying.

APPROACHES

The approach was structured in phases. Initially, a **normalized schema** was created consisting of seven main tables — employees, jobs, departments, locations, countries, regions, and dependents. Each table had appropriate foreign key references to maintain consistency and logical relations between data entities. After designing the schema, the tables were populated with realistic sample data. Next, a wide range of queries were executed to answer specific business questions, such as identifying top-performing employees, calculating total salary costs by department, or listing employees without dependents. Special attention was given to performance and maintainability — wherever possible, **subqueries** and **dynamic filters** were used instead of hardcoded values. This made the queries reusable and less prone to errors when the dataset changes. The goal was to simulate real-time HR decision-making using SQL alone, without relying on external software.

INSIGHTS

Several practical insights emerged from the project. For example, by grouping employees by department and aggregating salaries, it became easy to spot which departments were cost-intensive or under-utilized. Using conditional CASE statements helped classify salaries into categories like "Low", "Average", and "High", which is useful for employee compensation reviews. The use of EXISTS and NOT EXISTS queries helped identify employees who had or did not have dependents, which can assist in determining eligibility for family-related benefits. Similarly, using subqueries to compare employee salaries against departmental averages or extremes revealed income disparities or outliers. These types of queries replicate the kinds of insights that real HR teams would rely on for decisions related to promotions, hiring, retention, and compliance.

CONCLUSIONS

This project demonstrates how a structured, well-normalized SQL database can serve as the backbone for effective HR data management and decision-making. SQL, though traditionally considered a backend skill, proved to be incredibly powerful in extracting insights, applying logic, and summarizing data for business needs. One of the major takeaways from this project is that using SQL's full potential — including JOINS, CASE, GROUP BY, HAVING, and SUBQUERIES — allows analysts and HR professionals to avoid manual errors and repetitive tasks. Instead of relying on Excel filters or manual calculations, dynamic queries can provide consistent and fast results. Going forward, integrating this SQL-based HR system with data visualization tools like Power BI or Tableau would enhance interpretability and make it easier for non-technical stakeholders to engage with the data. Overall, the project is a strong example of how structured data and thoughtful queries can lead to intelligent and actionable business outcomes.

THANK

YOU