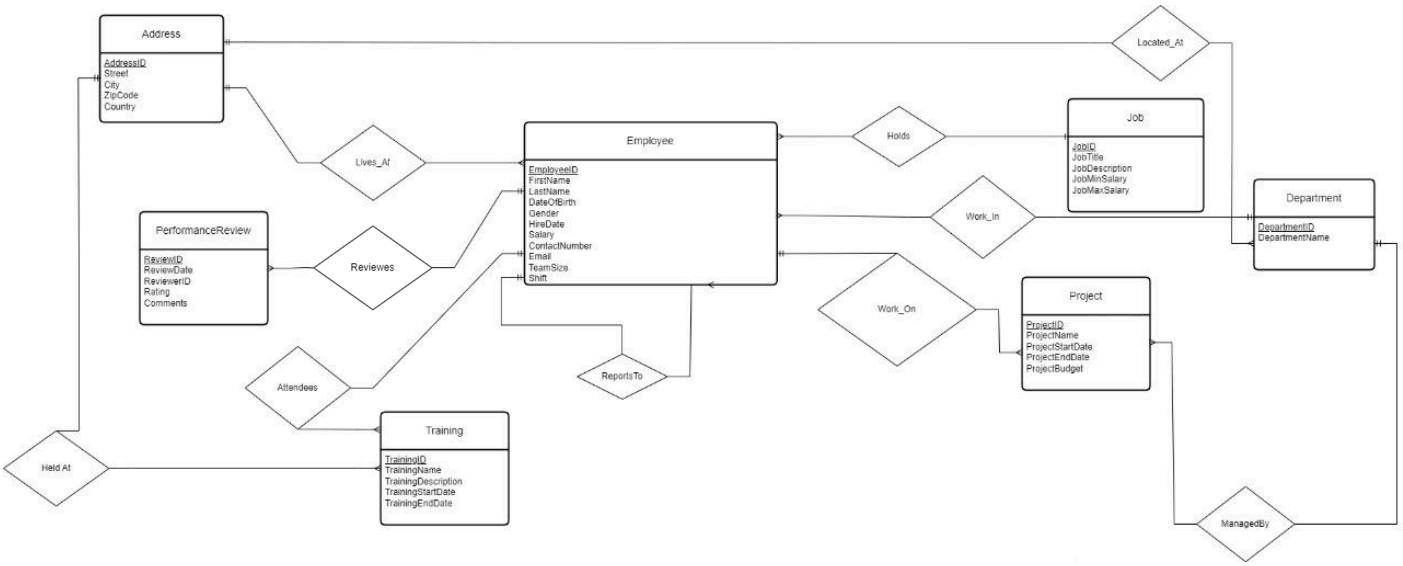
DBMS Project Submission - Report
Group-3

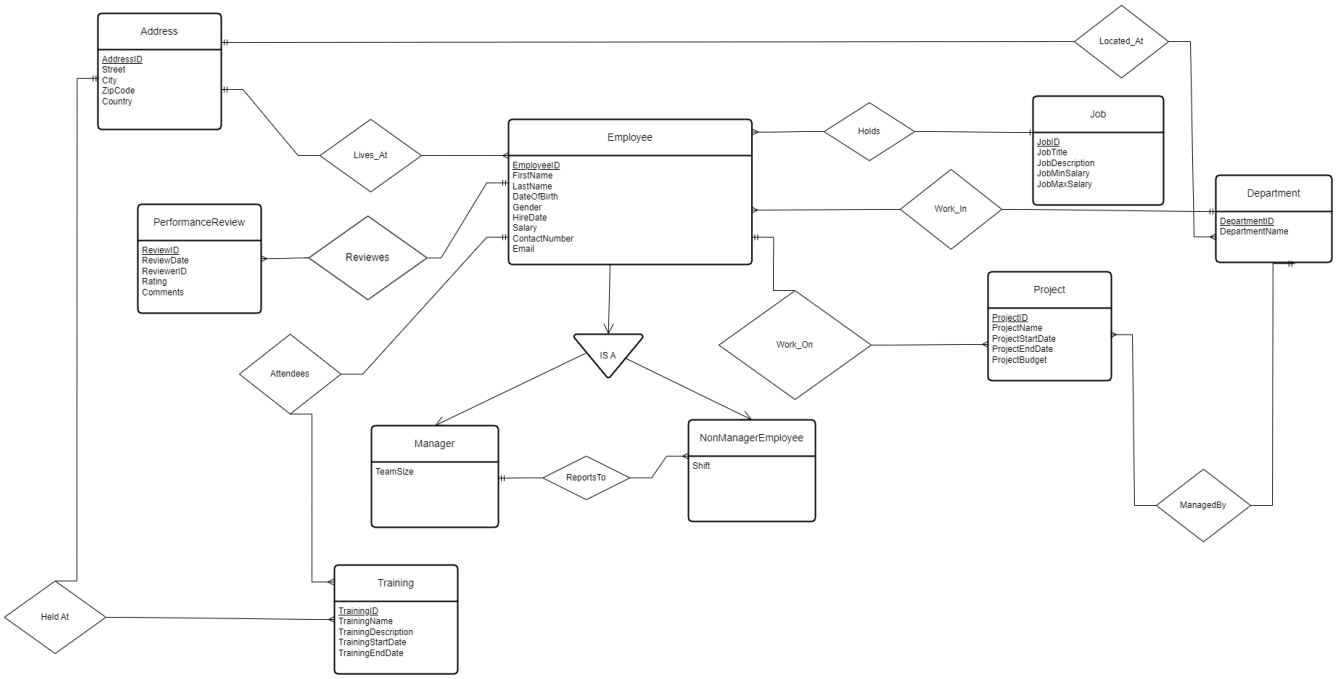
Group members:
AM.EN.U4AIE21002 - Abhishek Ashok Kumar
AM.EN.U4AIE21004 - Adith S. Biju
AM.EN.U4AIE21016 - Anugrah Nambiar
AM.EN.U4AIE21027 - Eric Oommen Mathew
AM.EN.U4AIE21054 - Rithesh R

Abstract
The Human Resource Management System is a database management system (DBMS) project designed to automate various human resources and administrative tasks. The HRMS integrates various functions of human resource management into a unified system, encompassing employee records, training and development, job and project assignments, and performance evaluations. The project will involve designing a relational database schema, developing SQL queries for data manipulation, and creating a user-friendly interface. The system should ensure data integrity, security, and privacy. It can be developed using DBMS software like postgresQL Server, and a programming language for the interface such as Python. This HRMS project will provide practical experience in database design, SQL programming, and software development, and it will solve real-world problems in HR management.

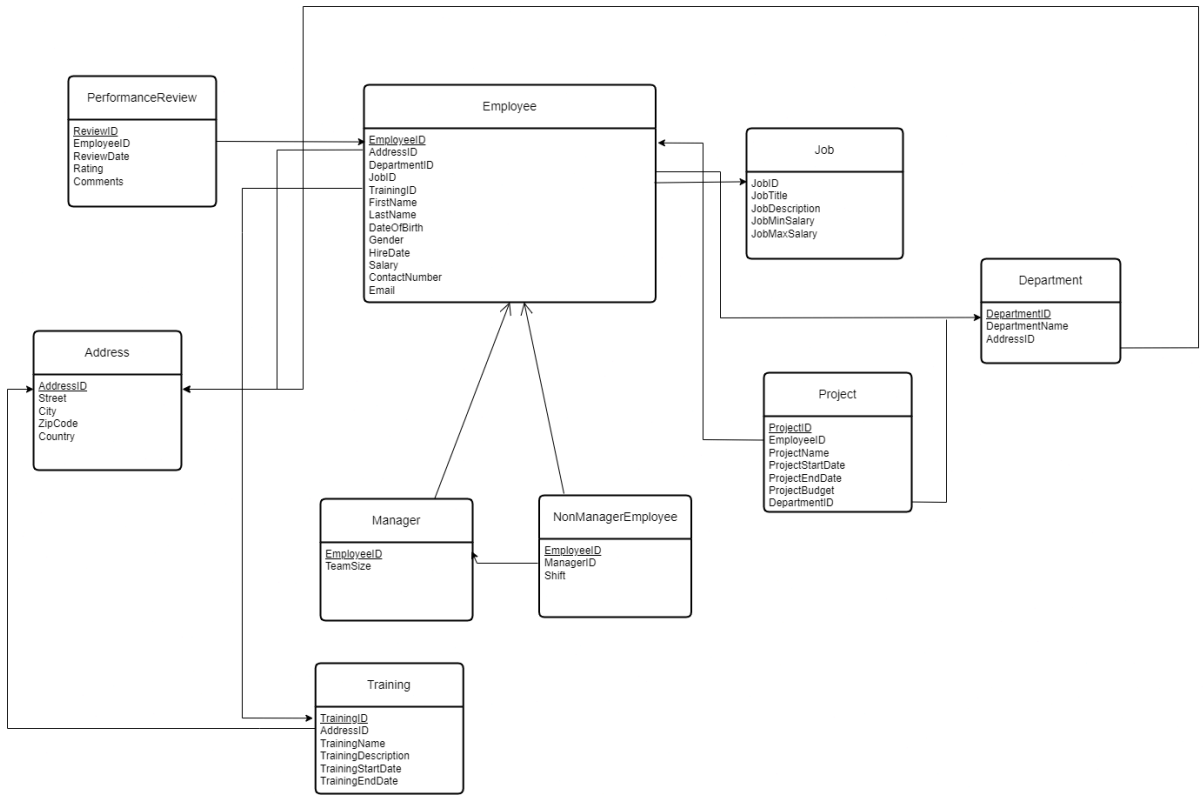
1. ER DIAGRAM



2. EXTENDED ER DIAGRAM



3. SCHEMA DIAGRAM



4. RELATIONAL SCHEMA

Employee (<u>EmployeeID</u>, AddressID (FK), DepartmentID (FK), JobID (FK), TrainingID (FK), FirstName, LastName, DateOfBirth, Gender, HireDate, Salary, ContactNumber, Email)

Manager (<u>EmployeeID</u>, TeamSize)

NonManagerEmployee (<u>EmployeeID</u>, ManagerID, Shift)

Address (<u>AddressID</u>, Street, City, ZipCode, Country)

Training (<u>TrainingID</u>, AddressID, TrainingName, TrainingDescription, TrainingStartDate, TrainingEndDate)

PerformanceReview (<u>ReviewID</u>, EmployeeID, ReviewDate, Rating, Comments)

Job (<u>JobID</u>, JobTitle, JobDescription, JobMinSalary, JobMaxSalary)

Department (<u>DepartmentID</u>, DepartmentName, AddressID)

Project (<u>ProjectID</u>, EmployeeID, ProjectName, ProjectStartDate, ProjectEndDate, ProjectBudget, DepartmentID)

5.NORMALIZATION

Universal Table

EmployeeID	FirstName	LastName	DateOfBirth	Gender	HireDate	Salary	ContactNumber

Email	MangerID	TeamSize	Shift	ReviewID	ReviewDate	Rating	Comments	AddressID	Address

TrainingID	TrainingName	TrainingDescription	TrainingStartDate	TrainingEndDate	JobId	JobTitle

JobDescription	JobMinSalary	JobMaxSalary	DepartmentID	DepartmentName	ProjectID	ProjectName

ProjectStartDate	ProjectEndDate	ProjectBudget

1NF Tables

We have reduced Address Attribute into Street,city,ZipCode and Country Attribute

EmployeeID	FirstName	LastName	DateOfBirth	Gender	HireDate	Salary	ContactNumber

Email	MangerID	TeamSize	Shift	ReviewID	ReviewDate	Rating	Comments	AddressID	Street	City	ZipCode

Country	TrainingID	TrainingName	TrainingDescription	TrainingStartDate	TrainingEndDate	JobId	JobTitle

JobDescription	JobMinSalary	JobMaxSalary	DepartmentID	DepartmentName	ProjectID	ProjectName

ProjectStartDate	ProjectEndDate	ProjectBudget

Primary key: (EmployeeID, TrainingID, JobID, ProjectID, ReviewID)

2NF TABLES

It should not consist of partial dependency.

Employee Table

EmployeeID (PK)	FirstName	LastName	DateOfBirth	Gender	HireDate	Salary	ContactNumber	Email

AddressID	Street	City	ZipCode	Country

Manager Table

EmployeeID (PK)(FK)	TeamSize

NonManager Table

EmployeeID(PK)(FK)	ManagerID(FK to Manager.Employee)	Shift

Review Table

ReviewID (PK)	EmployeeID (FK)	ReviewDate	Rating	Comments

Project Table

ProjectID(PK)	ProjectName	ProjectStartDate	ProjectEndDate	ProjectBudget

EmployeeProject Table

EmployeeID (FK)	ProjectID (FK)

Department Table

DepartmentID (PK)	DepartmentName

Employee Department Table

EmployeeID (FK)	DepartmentID (FK)

Job Table

JobID (PK)	JobTitle	JobDescription	JobMinSalary	JobMaxSalary

EmployeeJob Table

EmployeeID (FK)	JobID (FK)

Training Table

TrainingID (PK)	TrainingName	TrainingDescription	TrainingStartDate	TrainingEndDate

Employee Training Table

EmployeeID (FK)	TrainingID (FK)

3NF

Removing any transitive partial dependency like in employee table using employee id we can find Address id and using Address id we can find street,city,zip code and country.
So we split it into an Employee and Address Table.

Employee Table

EmployeeID (PK)	FirstName	LastName	DateOfBirth	Gender	HireDate	Salary	ContactNumber	Email	AddressID(FK)

Address Table

AddressID	Street	City	ZipCode	Country

Manager Table

EmployeeID (PK)(FK)	TeamSize

NonManager Table

EmployeeID(PK)(FK)	ManagerID(FK to Manager.Employee)	Shift

Review Table

ReviewID (PK)	EmployeeID (FK)	ReviewDate	Rating	Comments

Project Table

ProjectID(PK)	ProjectName	ProjectStartDate	ProjectEndDate	ProjectBudget

EmployeeProject Table

EmployeeID (FK)	ProjectID (FK)

Department Table

DepartmentID (PK)	DepartmentName

Employee Department Table

EmployeeID (FK)	DepartmentID (FK)

Job Table

JobID (PK)	JobTitle	JobDescription	JobMinSalary	JobMaxSalary

EmployeeJob Table

EmployeeID (FK)	JobID (FK)

Training Table

TrainingID (PK)	TrainingName	TrainingDescription	TrainingStartDate	TrainingEndDate

Employee Training Table

EmployeeID (FK)	TrainingID (FK)

Sql queries

Table Creation:

```
CREATE TABLE Address (
   AddressID INT PRIMARY KEY,
   Street VARCHAR(255) NOT NULL,
   City VARCHAR(100) NOT NULL,
   ZipCode VARCHAR(20) NOT NULL,
   Country VARCHAR(50) NOT NULL
);

CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   FirstName VARCHAR(100) NOT NULL,
   LastName VARCHAR(100) NOT NULL,
   DateOfBirth DATE NOT NULL,
   Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),
   HireDate DATE NOT NULL,
   Salary DECIMAL(10, 2) CHECK (Salary > 0),
   ContactNumber VARCHAR(20) NOT NULL,
   Email VARCHAR(100) NOT NULL UNIQUE,
   AddressID INT,
   FOREIGN KEY (AddressID) REFERENCES Address(AddressID)
);

CREATE TABLE Manager (
   EmployeeID INT PRIMARY KEY,
   TeamSize INT CHECK (TeamSize >= 0),
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

CREATE TABLE NonManager (
   EmployeeID INT PRIMARY KEY,
   Shift VARCHAR(20) NOT NULL,
   ManagerID INT,
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
   FOREIGN KEY (ManagerID) REFERENCES Manager(EmployeeID)
);

CREATE TABLE Review (
   ReviewID INT PRIMARY KEY,
   EmployeeID INT,
   ReviewDate DATE NOT NULL,
   Rating INT CHECK (Rating BETWEEN 1 AND 5),
   Comments TEXT,
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
```

```
);

CREATE TABLE Training (
    TrainingID INT PRIMARY KEY,
    TrainingName VARCHAR(100) NOT NULL,
    TrainingDescription TEXT,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    CHECK (EndDate >= StartDate)
);

CREATE TABLE EmployeeTraining (
    EmployeeID INT,
    TrainingID INT,
    PRIMARY KEY (EmployeeID, TrainingID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (TrainingID) REFERENCES Training(TrainingID)
);

CREATE TABLE Job (
    JobID INT PRIMARY KEY,
    JobTitle VARCHAR(100) NOT NULL,
    JobDescription TEXT,
    JobMinSalary DECIMAL(10, 2) CHECK (JobMinSalary > 0),
    JobMaxSalary DECIMAL(10, 2) CHECK (JobMaxSalary >= JobMinSalary)
);

CREATE TABLE EmployeeJob (
    EmployeeID INT,
    JobID INT,
    PRIMARY KEY (EmployeeID, JobID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (JobID) REFERENCES Job(JobID)
);

CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(100) NOT NULL
);

CREATE TABLE EmployeeDepartment (
    EmployeeID INT,
    DepartmentID INT,
    PRIMARY KEY (EmployeeID, DepartmentID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

```
CREATE TABLE Project (
    ProjectID INT PRIMARY KEY,
    ProjectName VARCHAR(100) NOT NULL,
    ProjectStartDate DATE NOT NULL,
    ProjectEndDate DATE NOT NULL,
    ProjectBudget DECIMAL(15, 2) CHECK (ProjectBudget > 0),
    CHECK (ProjectEndDate >= ProjectStartDate)
);

CREATE TABLE EmployeeProject (
    EmployeeID INT,
    ProjectID INT,
    PRIMARY KEY (EmployeeID, ProjectID),
    FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID),
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);
```

Queries

i. Aggregate functions, Group by…having

```
-- Get average salary of employees in each city who earn more than 50000
SELECT City, AVG(Salary) AS AvgSalary
FROM Employee E
JOIN Address A ON E.AddressID = A.AddressID
GROUP BY City
HAVING AVG(Salary) > 50000;
```

	city character varying (100)	avgsalary numeric
1	Bengaluru	69200.000000000000
2	Chennai	70000.000000000000
3	Kolkata	61000.000000000000

ii. Order by

```
-- Get all employees ordered by their hire date in descending order
SELECT EmployeeID, FirstName, LastName, HireDate
FROM Employee
ORDER BY HireDate DESC;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	hiredate date
1	9	Amit	Kumar	2016-02-25
2	5	Suman	Gupta	2015-12-01
3	7	Kiran	Rao	2013-11-14
4	2	Priya	Nair	2012-08-20
5	3	Anjali	Verma	2011-03-18
6	6	Rohan	Desai	2011-01-09
7	10	Rita	Shah	2010-09-17
8	1	Rahul	Sharma	2010-06-15
9	4	Vikram	Patel	2009-04-22
10	8	Meera	Iyer	2008-07-19

iii. Join, Outer Join

```
-- Get list of employees and their assigned projects, including employees with no projects
SELECT E.EmployeeID, E.FirstName, E.LastName, P.ProjectName
FROM Employee E
LEFT JOIN EmployeeProject EP ON E.EmployeeID = EP.EmployeeID
LEFT JOIN Project P ON EP.ProjectID = P.ProjectID;
```

	employeeid integer	firstname character varying (100)	lastname character varying (100)	projectname character varying (100)
1	1	Rahul	Sharma	Alpha
2	2	Priya	Nair	Beta
3	3	Anjali	Verma	Gamma
4	4	Vikram	Patel	Delta
5	5	Suman	Gupta	Epsilon
6	6	Rohan	Desai	Zeta
7	7	Kiran	Rao	Eta
8	8	Meera	Iyer	Theta
9	9	Amit	Kumar	Iota
10	10	Rita	Shah	Kappa

iv. Query having Boolean operators

```
-- Find employees who are either Managers or earn a salary greater than 70000
SELECT E.EmployeeID, E.FirstName, E.LastName
FROM Employee E
LEFT JOIN Manager M ON E.EmployeeID = M.EmployeeID
WHERE M.EmployeeID IS NOT NULL OR E.Salary > 70000;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)
1	1	Rahul	Sharma
2	2	Priya	Nair
3	3	Anjali	Verma
4	4	Vikram	Patel
5	5	Suman	Gupta
6	6	Rohan	Desai
7	7	Kiran	Rao
8	8	Meera	Iyer
9	9	Amit	Kumar
10	10	Rita	Shah

v. Query having arithmetic operators

```
SELECT (ProjectBudget*2) AS newBudget FROM Project;
```

```
HumanResourceManagement=# SELECT (ProjectBudget*2) AS newBudget FROM Project;
 newbudget
-----------
 200000.00
 100000.00
 150000.00
 120000.00
 180000.00
 160000.00
 140000.00
 170000.00
 190000.00
 110000.00
(10 rows)
```

vi. A search query using string operators

```
-- Find employees with ‘Rahul’ in their first name
SELECT EmployeeID, FirstName, LastName
FROM Employee
WHERE FirstName LIKE '%Rahul%';
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)
1	1	Rahul	Sharma

vii. Usage of to_char, extract

```
-- Get the month and year from the hire date of employees
SELECT EmployeeID, FirstName, LastName, TO_CHAR(HireDate, 'Month YYYY') AS
HireMonthYear
FROM Employee;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	hiremonthyear text
1	1	Rahul	Sharma	June 2010
2	2	Priya	Nair	August 2012
3	3	Anjali	Verma	March 2011
4	4	Vikram	Patel	April 2009
5	5	Suman	Gupta	December 2015
6	6	Rohan	Desai	January 2011
7	7	Kiran	Rao	November 2013
8	8	Meera	Iyer	July 2008
9	9	Amit	Kumar	February 2016
10	10	Rita	Shah	September 2010

```
-- Extract year from hire date and count employees hired each year
SELECT EXTRACT(YEAR FROM HireDate) AS HireYear, COUNT(*) AS EmployeeCount
FROM Employee
GROUP BY EXTRACT(YEAR FROM HireDate);
```

	hireyear numeric	employeecount bigint
1	2011	2
2	2012	1
3	2016	1
4	2010	2
5	2015	1
6	2013	1
7	2008	1
8	2009	1

viii. Between, IN, Not between, Not IN

```
-- Get employees with salaries between 50000 and 100000
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee
WHERE Salary BETWEEN 50000 AND 100000;
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	salary numeric (10,2)
1	1	Rahul	Sharma	65000.00
2	2	Priya	Nair	70000.00
3	3	Anjali	Verma	72000.00
4	4	Vikram	Patel	68000.00
5	5	Suman	Gupta	55000.00
6	6	Rohan	Desai	80000.00
7	7	Kiran	Rao	60000.00
8	8	Meera	Iyer	75000.00
9	9	Amit	Kumar	58000.00
10	10	Rita	Shah	66000.00

```
-- Get employees working in specified departments
SELECT E.EmployeeID, E.FirstName, E.LastName
FROM Employee E
JOIN EmployeeDepartment ED ON E.EmployeeID = ED.EmployeeID
WHERE ED.DepartmentID IN (1, 2, 3);
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)
1	1	Rahul	Sharma
2	2	Priya	Nair
3	3	Anjali	Verma

```
-- Get employees not hired between two dates
SELECT EmployeeID, FirstName, LastName, HireDate
FROM Employee
WHERE HireDate NOT BETWEEN '2020-01-01' AND '2021-01-01';
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	hiredate date
1	1	Rahul	Sharma	2010-06-15
2	2	Priya	Nair	2012-08-20
3	3	Anjali	Verma	2011-03-18
4	4	Vikram	Patel	2009-04-22
5	5	Suman	Gupta	2015-12-01
6	6	Rohan	Desai	2011-01-09
7	7	Kiran	Rao	2013-11-14
8	8	Meera	Iyer	2008-07-19
9	9	Amit	Kumar	2016-02-25
10	10	Rita	Shah	2010-09-17

ix. Set operations

```
-- Get all employees who are either managers or assigned to projects, removing duplicates
SELECT EmployeeID, FirstName, LastName
FROM Employee
WHERE EmployeeID IN (SELECT EmployeeID FROM Manager)
UNION
SELECT E.EmployeeID, E.FirstName, E.LastName
FROM Employee E
JOIN EmployeeProject EP ON E.EmployeeID = EP.EmployeeID;
```

	employeeid integer	firstname character varying (100)	lastname character varying (100)
1	9	Amit	Kumar
2	3	Anjali	Verma
3	2	Priya	Nair
4	5	Suman	Gupta
5	8	Meera	Iyer
6	4	Vikram	Patel
7	10	Rita	Shah
8	7	Kiran	Rao
9	6	Rohan	Desai
10	1	Rahul	Sharma

```
-- Get employees who are managers but not assigned to any projects
SELECT E.EmployeeID, E.FirstName, E.LastName
FROM Employee E
JOIN Manager M ON E.EmployeeID = M.EmployeeID
WHERE E.EmployeeID NOT IN (SELECT EmployeeID FROM EmployeeProject);
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)

x. Subquery using EXISTS / NOT EXISTS, ANY, ALL

```
-- Get employees who have attended any training
SELECT E.EmployeeID, E.FirstName, E.LastName
FROM Employee E
WHERE EXISTS (SELECT 1 FROM EmployeeTraining ET WHERE E.EmployeeID =
ET.EmployeeID);
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)
1	1	Rahul	Sharma
2	2	Priya	Nair
3	3	Anjali	Verma
4	4	Vikram	Patel
5	5	Suman	Gupta
6	6	Rohan	Desai
7	7	Kiran	Rao
8	8	Meera	Iyer
9	9	Amit	Kumar
10	10	Rita	Shah

```
-- Get employees who have not been reviewed
SELECT EmployeeID, FirstName, LastName
FROM Employee E
WHERE NOT EXISTS (SELECT 1 FROM Review R WHERE E.EmployeeID =
R.EmployeeID);
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)

```
-- Get employees whose salary is greater than all employees in department 1
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employee
WHERE Salary > ALL (SELECT Salary FROM Employee E JOIN EmployeeDepartment ED
ON E.EmployeeID.EmployeeID WHERE ED.DepartmentID = 1);
```

	employeeid [PK] integer	firstname character varying (100)	lastname character varying (100)	salary numeric (10,2)
1	2	Priya	Nair	70000.00
2	3	Anjali	Verma	72000.00
3	4	Vikram	Patel	68000.00
4	6	Rohan	Desai	80000.00
5	8	Meera	Iyer	75000.00
6	10	Rita	Shah	66000.00