TASK 3

SQL DB

Request 0:

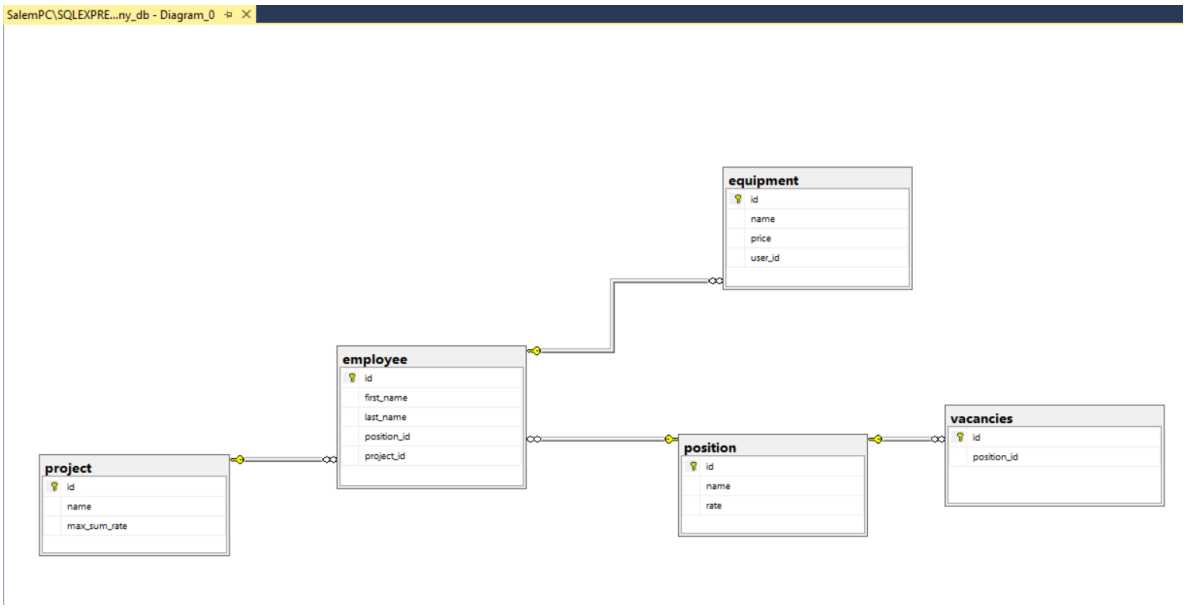
Write requests to generate a database with the specified structure and fill it with initial data (requests with the creation of tables + requests to create at least 5 rows per table).

```
CREATE TABLE project
(
id INT IDENTITY(1,1) PRIMARY KEY,
name nvarchar(MAX) NOT NULL,
max_sum_rate DECIMAL(10,2) NOT NULL
);

CREATE TABLE employee
(
id INT IDENTITY(1,1) PRIMARY KEY,
first_name nvarchar(MAX) NOT NULL,
last_name nvarchar(MAX) NOT NULL,
position_id int,
project_id int
);

CREATE TABLE position
(
id INT IDENTITY(1,1) PRIMARY KEY,
name nvarchar(MAX) NOT NULL,
rate decimal(10, 2) NOT NULL
);

CREATE TABLE equipment
(
id INT IDENTITY(1,1) PRIMARY KEY,
name nvarchar(MAX) NOT NULL,
price DECIMAL(10, 2) NOT NULL,
user_id int NOT NULL
);

CREATE TABLE vacancies
(
id INT IDENTITY(1,1) PRIMARY KEY,
position_id INT NOT NULL
);
```

Initialization of data:

```sql
INSERT INTO project (name, max_sum_rate) VALUES
('Bank of America', 10000),
('Vance Refrigeration', 20000),
('Project X', 30000),
('Apple', 35000),
('Dunder Mifflin', 25000);

INSERT INTO employee (first_name, last_name, position_id, project_id) VALUES
('Isaac', 'Amortegui', 5, 2),
('Salem', 'Amortegui', 3, 4),
('Laura', 'Rico', 5, 1),
('Michael', 'Lavander', 1, NULL),
('Michael', 'Scott', 2, 4);
('Dwight', 'Schrutte', NULL, NULL),
('Falcao', 'Garcia', 2, NULL),
('John', 'Doe', 5, NULL),
('Alex', 'Smith', 3, 2),
('Slim', 'Shady', 2, 4),
('William', 'Smith', 1, 5);


INSERT INTO position ( name, rate) VALUES
('QA Auto', 500),
('Developer', 390.50),
('Project Manager', 800),
('Business Analyst', 600),
('Junior QA', 200.8),
('Senior QA', 400);

INSERT INTO equipment (name, price, user_id) VALUES
('View Sonic Monitor', 20, 4),
('HP Laptop', 500, 4),
('Mouse', 5, 1),
('Mechanical Keyboard', 200, 2),
('iPad', 600, 5),
('CPU', 500, 2);

INSERT INTO vacancies (position_id) VALUES
(2),
(5),
(1),
(3),
(4);
```

Relations Diagram



Request 1:

Display all the employees without a project:

```
SELECT *
FROM EMPLOYEE
WHERE project_id IS NULL;
```

Results Messages

	id	first_name	last_name	position_id	project_id
1	4	Michael	Lavander	1	NULL
2	10	Dwight	Schrutte	NULL	NULL
3	11	Falcao	Garcia	2	NULL
4	12	John	Doe	5	NULL

Another way, only showing Employee Names, given that the displayed don't have a project assigned:

```
SELECT first_name + ' ' + last_name AS Employee_Without_Project
FROM Employee
WHERE project_id IS NULL;
```

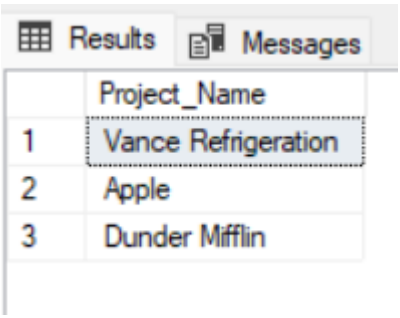
Results Messages

	Employee_Without_Project
1	Michael Lavander
2	Dwight Schrutte
3	Falcao Garcia
4	John Doe

Request 2:

Display the names of projects where the total monthly salary of employees is higher than the monthly project budget:

```
SELECT project.name AS Project_Name
FROM Project
INNER JOIN Employee ON project.id = project_id
INNER JOIN Position ON position_id = position.id
GROUP BY project.id, project.name, project.max_sum_rate
HAVING SUM(position.rate) > project.max_sum_rate
```



Results | Messages

	Project_Name
1	Vance Refrigeration
2	Apple
3	Dunder Mifflin

*I had to update some of the position rates and project budgets so I could display matching data.

Request 3:

Display the names of employees (first and last name in one string), project names from projects where the total monthly salary of employees is higher than the monthly budget of the project.

```
SELECT
  CONCAT(employee.first_name, ' ', employee.last_name) AS Employee_Name,
  Project.name AS Project_Name,
  SUM(Position.rate) AS Total_Monthly_Salary,
  Project.max_sum_rate AS Project_Monthly_Budget
FROM
  Employee
  JOIN Position ON Employee.position_id = Position.id
  JOIN Project ON Employee.project_id = Project.id
GROUP BY
  CONCAT(Employee.first_name, ' ', Employee.last_name), Project.name,
Project.max_sum_rate
HAVING
  SUM(Position.rate) > Project.max_sum_rate;
```

Results | Messages

	Employee_Name	Project_Name	Total_Monthly_Salary	Project_Monthly_Budget
1	Alex Smith	Vance Refrigeration	15000.00	10500.00
2	Salem Amortegui	Apple	15000.00	10500.00

Request 4:

Display the names of employees from projects where the total monthly salary of employees + their cost working equipment divided by 12 (total monthly project costs) above the monthly budget of the project. Also display the name of the project and the difference between the monthly budget of the project and monthly expenses in the final table.

```
SELECT CONCAT(Employee.first_name, ' ', Employee.last_name) AS Employee_Name,
       Project.name AS Project_Name,
       (Project.max_sum_rate - ((SUM(Position.rate) + (SUM(Equipment.price))/12)))
AS Difference
FROM Employee
INNER JOIN Position ON Employee.position_id = Position.id
INNER JOIN Project ON Employee.project_id = Project.id
LEFT JOIN Equipment ON Employee.id = Equipment.user_id
GROUP BY Employee.id, Employee.first_name, Employee.last_name, Project.name,
Project.max_sum_rate
HAVING SUM(Position.rate) + COALESCE(SUM(Equipment.price), 0) > Project.max_sum_rate
ORDER BY Project_Name, Employee_Name;
```

Results | Messages

	Employee_Name	Project_Name	Difference
1	Michael Scott	Apple	450.00
2	Salem Amortegui	Apple	-4516.67
3	Alex Smith	Vance Refrigeration	NULL

Trigger:

Create a trigger that, when creating a project, adds a new vacancy for Project Manager (in the trigger it is impossible to use the hardcode ID of the position of the PM, find it with a request):

```
CREATE TRIGGER tr_add_project_manager_vacancy
ON Project
AFTER INSERT
AS
BEGIN
    DECLARE @project_manager_position_id INT;
    SELECT @project_manager_position_id = id FROM Position WHERE name = 'Project
Manager';

    INSERT INTO Vacancies (position_id)
    VALUES (@project_manager_position_id);
END;
```