1. ***Write a SQL query to find the names and salaries of the employees that take the minimal salary in the company. Use a nested SELECT statement.***

```
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary =
       (SELECT MIN(Salary) FROM Employees)
```

2. ***Write a SQL query to find the names and salaries of the employees that have a salary that is up to 10% higher than the minimal salary for the company.***

```
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary <=
       (SELECT MIN(Salary) * 1.1 FROM Employees)
```

3. ***Write a SQL query to find the full name, salary and department of the employees that take the minimal salary in their department. Use a nested SELECT statement.***

```
SELECT
       FirstName,
       LastName,
       d.Name AS [Department Name],
       Salary
FROM Employees e
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
WHERE Salary =
  (SELECT MIN(Salary) FROM Employees s
   WHERE s.DepartmentID = e.DepartmentID)
```

4. ***Write a SQL query to find the average salary in the department #1.***

```
SELECT
       DepartmentID,
       AVG(Salary) [Average Salary]
FROM Employees
WHERE DepartmentID = 1
GROUP BY DepartmentID
```

5. ***Write a SQL query to find the average salary in the "Sales" department.***

```
SELECT
       d.Name [Department Name],
       AVG(e.Salary) [Average Salary]
FROM Employees e
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
WHERE d.Name = 'Sales'
GROUP BY d.Name
```

6. ***Write a SQL query to find the number of employees in the "Sales" department.***

```
SELECT
       d.Name [Department Name],
       COUNT(*) [Employees Count]
FROM Employees e
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
WHERE d.Name = 'Sales'
GROUP BY d.Name
```

7. ***Write a SQL query to find the number of all employees that have manager.***

```
SELECT
COUNT(ManagerID) [Employees With Manager Count]
FROM Employees
```

8. **Write a SQL query to find the number of all employees that have no manager.**

```
SELECT
COUNT(EmployeeID) [Employees Without Manager Count]
FROM Employees
WHERE ManagerID IS NULL
```

9. ***Write a SQL query to find all departments and the average salary for each of them.***

```
SELECT
       d.Name [Department Name],
       AVG(Salary) [Average Salary]
FROM Employees e
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
GROUP BY d.Name
```

10. ***Write a SQL query to find the count of all employees in each department and for each town.***

```
SELECT
       d.Name [Department Name],
       t.Name [Town Name],
       COUNT(e.EmployeeID) [Employees Count]
FROM Employees e
       INNER JOIN Addresses a
              ON e.AddressID = a.AddressID
       INNER JOIN Towns t
              ON a.TownID = t.TownID
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
GROUP BY d.Name, t.Name
ORDER BY t.Name
```

11. ***Write a SQL query to find all managers that have exactly 5 employees. Display their first name and last name.***

```
SELECT m.FirstName, m.LastName
FROM Employees m
       INNER JOIN Employees e
              ON m.EmployeeID = e.ManagerID
GROUP BY m.FirstName, m.LastName
HAVING COUNT(e.EmployeeID) = 5
```

12. ***Write a SQL query to find all employees along with their managers. For employees that do not have manager display the value "(no manager)".***

```
SELECT
e.FirstName + ' ' + e.LastName AS [Employee Name],
COALESCE(m.FirstName + ' ' + m.LastName, 'No manager') AS [Manager Name]
FROM Employees e
       LEFT OUTER JOIN Employees m
              ON e.ManagerID= m.EmployeeID
```

13. ***Write a SQL query to find the names of all employees whose last name is exactly 5 characters long. Use the built-in LEN(str) function.***

```
SELECT
e.FirstName + ' ' + e.LastName AS [Employee Name]
FROM Employees e
WHERE LEN(e.LastName) = 5
```

14. ***Write a SQL query to display the current date and time in the following format "day.month.year hour:minutes:seconds:milliseconds". Search in Google to find how to format dates in SQL Server.***

```
SELECT CONVERT(VARCHAR(24), GETDATE(), 113)
```

15. ***Write a SQL statement to create a table Users. Users should have username, password, full name and last login time. Choose appropriate data types for the table fields. Define a primary key column with a primary key constraint. Define the primary key column as identity to facilitate inserting records. Define unique constraint to avoid repeating usernames. Define a check constraint to ensure the password is at least 5 characters long.***

```
CREATE TABLE Users (
       UserId int IDENTITY,
       Username nvarchar(50) NOT NULL,
       Password nvarchar(50) NOT NULL,
       FullName nvarchar(50),
       LastLogin datetime NOT NULL,
       CONSTRAINT PK_Users PRIMARY KEY (UserId),
       CONSTRAINT UK_Username UNIQUE (Username),
       CONSTRAINT CHK_Password CHECK (LEN(Password) >= 5)
)
```

16. ***Write a SQL statement to create a view that displays the users from the Users table that have been in the system today. Test if the view works correctly.***

```
CREATE VIEW [Users Logged In Today] AS
SELECT *
FROM Users
WHERE DATEDIFF(day, LastLogin, GETDATE()) = 0
```

17. ***Write a SQL statement to create a table Groups. Groups should have unique name (use unique constraint). Define primary key and identity column.***

```
CREATE TABLE Groups (
       GroupId int IDENTITY,
       Name nvarchar(50) NOT NULL,
       CONSTRAINT PK_GroupId PRIMARY KEY (GroupId),
       CONSTRAINT UK_Name UNIQUE (Name)
)
```

18. ***Write a SQL statement to add a column GroupID to the table Users. Fill some data in this new column and as well in the Groups table. Write a SQL statement to add a foreign key constraint between tables Users and Groups tables.***

```
ALTER TABLE Users
ADD GroupId int NOT NULL
GO
INSERT INTO Users(Username, Password, FullName, LastLogind, GroupId)
VALUES
       ('Ivan', 12345, 'Ivan Dimitrov', GETDATE(), 1),
       ('Ceko', 12345, 'Ceko Cekov', GETDATE(), 2),
       ('Dimitur', 12345, 'Dimitur Ivanov', GETDATE(), 3)
GO
ALTER TABLE Users
ADD CONSTRAINT FK_Users_Groups
       FOREIGN KEY (GroupId)
       REFERENCES Groups(GroupId)
GO
```

19. ***Write SQL statements to insert several records in the Users and Groups tables.***

```
INSERT INTO Groups (Name)
VALUES
       ('Group One'),
       ('Group Two'),
       ('Group Three')
GO
INSERT INTO Users(Username, Password, FullName, LastLogind, GroupId)
VALUES
       ('Ivan', 12345, 'Ivan Dimitrov', GETDATE(), 1),
       ('Ceko', 12345, 'Ceko Cekov', GETDATE(), 2),
       ('Dimitur', 12345, 'Dimitur Ivanov', GETDATE(), 3)
GO
```

20. ***Write SQL statements to update some of the records in the Users and Groups tables.***

```
UPDATE Groups
SET Name = 'Group Four'
WHERE GroupId = 3
GO
UPDATE Users
SET Username = 'Pesho', Password = 678910, FullName = 'Pesho Petrov', LastLogind
= GETDATE(), GroupId = 3
WHERE UserId = 2
GO
```

21. Write SQL statements to delete some of the records from the Users and Groups tables.

```
DELETE FROM Groups
WHERE GroupId = 3
GO
DELETE FROM Users
WHERE UserId = 2
GO
```

22. Write SQL statements to insert in the Users table the names of all employees from the Employees table. Combine the first and last names as a full name. For username use the first letter of the first name + the last name (in lowercase). Use the same for the password, and NULL for last login time.

```
INSERT INTO Users
(FullName, Username, Password, LastLogin, GroupId)
SELECT
FirstName + ' ' + LastName,
SUBSTRING(FirstName, 1, 3) + LOWER(LastName),
SUBSTRING(FirstName, 1, 3) + LOWER(LastName),
NULL,
1
FROM Employees
```

23. Write a SQL statement that changes the password to NULL for all users that have not been in the system since 10.03.2010.

```
UPDATE Users
SET Password = NULL
WHERE LastLogin < CONVERT(varchar(24), '10.03.2010', 113)
```

24. Write a SQL statement that deletes all users without passwords (NULL password).

```
DELETE FROM Users
WHERE Password IS NULL
```

25. Write a SQL query to display the average employee salary by department and job title.

```
SELECT
       d.Name AS [Department Name],
       e.JobTitle,
       AVG(e.Salary) AS [Average Salary]
FROM Employees e
       INNER JOIN Departments d
              ON e.DepartmentID = d.DepartmentID
GROUP BY d.Name, e.JobTitle
```

26. ***Write a SQL query to display the minimal employee salary by department and job title along with the name of some of the employees that take it.***

```
SELECT
        d.Name AS [Department Name],
        e.JobTitle,
        MIN(e.Salary) AS [Min Salary],
        MIN(e.FirstName) + ' and ' + MAX(e.FirstName) AS [Some Employees Taking
That Salary]
FROM Employees e
        INNER JOIN Departments d
                ON e.DepartmentID = d.DepartmentID
GROUP BY d.Name, e.JobTitle
```

27. ***Write a SQL query to display the town where maximal number of employees work.***

```
SELECT
        t.Name AS [Town Name],
        COUNT(e.EmployeeID) AS [Number Of Employees]
FROM Employees e
        INNER JOIN Addresses a
                ON e.AddressID = a.AddressID
        INNER JOIN Towns t
                ON a.TownID = t.TownID
GROUP BY t.Name
ORDER BY [Number Of Employees] DESC
```

28. ***Write a SQL query to display the number of managers from each town.***

```
SELECT
        t.Name,
        COUNT(*) as [Number of managers]
FROM Towns t
        JOIN Addresses a
                ON t.TownID = a.TownID
        JOIN Employees e
                ON a.AddressID = e.AddressID
WHERE e.EmployeeID IN
        (SELECT DISTINCT ManagerID FROM Employees)
GROUP BY t.Name
ORDER BY [Number of managers] DESC
```

29. ***Write a SQL to create table WorkHours to store work reports for each employee (employee id, date, task, hours, comments). Don't forget to define identity, primary key and appropriate foreign key.***
Issue few SQL statements to insert, update and delete of some data in the table.
Define a table WorkHoursLogs to track all changes in the WorkHours table with triggers.
For each change keep the old record data, the new record data and the command (insert / update / delete).

30. ***Start a database transaction, delete all employees from the 'Sales' department along with all dependent records from the pother tables. At the end rollback the transaction.***

```
BEGIN TRAN DeleteEmployees

DECLARE @id int, @managerId int
SET @id = (SELECT DepartmentID FROM Departments
WHERE Name = 'Sales')
SET @managerId = (SELECT ManagerID
FROM Departments WHERE DepartmentID = @id)

DELETE FROM Employees
WHERE DepartmentID = @id AND EmployeeID != @managerId

ROLLBACK TRAN DeleteEmployees
```

31. ***Start a database transaction and drop the table EmployeesProjects. Now how you could restore back the lost table data?***

```
BEGIN TRAN

DROP TABLE EmployeesProjects

ROLLBACK TRAN
```