**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 e.FirstName + ' ' + e.LastName as Name, e.Salary

FROM [TelerikAcademy].[dbo].Employees e

WHERE Salary =

(SELECT MAX(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 e.FirstName + ' ' + e.LastName as Name, e.Salary

FROM [TelerikAcademy].[dbo].Employees e

WHERE Salary =

(SELECT MIN(Salary)

FROM Employees e)

**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 e.FirstName + ' ' + e.LastName as Name, e.Salary, e.DepartmentID, d.DepartmentID, d.Name as [Department Name]

FROM TelerikAcademy.dbo.Employees e INNER JOIN

TelerikAcademy.dbo.Departments d

ON d.DepartmentID = e.DepartmentID

WHERE Salary =

(SELECT MIN(Salary) FROM Employees

WHERE DepartmentID = e.DepartmentID)

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

--SELECT e.DepartmentID, d.Name, SUM(Salary) as [Average salary]

--FROM Employees e

-- JOIN Departments d

-- ON d.DepartmentID = e.DepartmentID

-- GROUP BY e.DepartmentID, d.Name

SELECT

AVG(Salary) [Average Salary]

FROM Employees

WHERE DepartmentID = 1

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

SELECT

AVG(Salary) [Average Salary]

FROM Employees e

WHERE DepartmentID = '2'

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

SELECT COUNT(EmployeeID) as [Employees in Sales department]

FROM Employees

WHERE DepartmentID = '2'

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

SELECT COUNT(ManagerID) as [Employees with Manager]

FROM Employees

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

SELECT COUNT(\*) as [Employees with Manager]

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, AVG(Salary) as [Average salary]

FROM TelerikAcademy.dbo.Departments d INNER JOIN TelerikAcademy.dbo.Employees e

ON e.DepartmentID = d.DepartmentIDSELECT SELECT

GROUP BY Name

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

SELECT Count(\*) as [Ppl in department], e.DepartmentID

FROM TelerikAcademy.dbo.Employees e

GROUP BY DepartmentID

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

SELECT e.ManagerID, COUNT(e.EmployeeID) as [Employees managed], m.FirstName, m.LastName

FROM TelerikAcademy.dbo.Employees e INNER JOIN Employees m

ON e.ManagerID = m.ManagerID

GROUP BY e.ManagerID, 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 CASE WHEN e.ManagerID IS NULL THEN 'no manager' ELSE CONVERT(nvarchar(50), e.ManagerID) END AS ManagerID, e.EmployeeID, e.FirstName, e.LastName

--FROM TelerikAcademy.dbo.Employees e

--GROUP BY e.FirstName, e.LastName, e.ManagerID, e.EmployeeID

SELECT ISNULL(CONVERT(nvarchar(50), e.ManagerID), 'no manager') as ManagerID, e.EmployeeID, e.FirstName, e.LastName

FROM TelerikAcademy.dbo.Employees e

GROUP BY e.FirstName, e.LastName, e.ManagerID, e.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, LEN(e.LastName) as [Last name - 5 characters]

FROM [TelerikAcademy].[dbo].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 SYSDATETIME() as [Current datetime] Default

SELECT FORMAT (GETDATE(), 'dd.mm.yyyy HH:m:ss:mmm', 'en-US' ) AS [Current datime]

**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 NOT NULL IDENTITY PRIMARY KEY,

Username nvarchar(50) NOT NULL UNIQUE,

PasswordUser nvarchar(15) NOT NULL CHECK (LEN(PasswordUser) >= 5),

FirstName nvarchar(30) NOT NULL,

LastName nvarchar(30) NOT NULL,

LastLogin DateTime NOT NULL,

)

1. **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 [Last logged] AS

SELECT FirstName, LastLogin

FROM MyUsers.dbo.Users

WHERE LastLogin = '2015-01-12'

1. **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(

ID INT NOT NULL IDENTITY PRIMARY KEY,

Name nvarchar(30) NOT NULL UNIQUE

)

1. **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

ALTER TABLE Users

ADD FOREIGN KEY (GroupID)

REFERENCES Groups(ID)

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

INSERT INTO Groups (Name)

VALUES ('Grandpa')ALERT

INSERT INTO Groups (Name)

VALUES ('Alcender')

INSERT INTO Groups (Name)

VALUES (‘Thief’)

INSERT INTO Users (Username, PasswordUser, FirstName, LastName, LastLogin, GroupID)

VALUES ('CeckaTra', 'naceciparolata', 'Cecka', 'Borichkova', '2015/2/2', 2)

INSERT INTO Users (Username, PasswordUser, FirstName, LastName, LastLogin, GroupID)

VALUES ('Asencho', 'naciparolata', 'Asen', 'Balabanov', '2011/2/2', 6)

INSERT INTO Users (Username, PasswordUser, FirstName, LastName, LastLogin, GroupID)

VALUES ('DrinkCode', 'hollysh1t', 'Maina', 'Gospodinov', '2001/2/2', 5)

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

UPDATE Users

SET Username = 'Sanchez', FirstName = 'Asenzhez'

WHERE Username = 'Asencho'

UPDATE Users

SET Username = 'SASHETO', FirstName = 'SashoWE'

WHERE FirstName = 'Sashko'

UPDATE Users

SET Username = 'CodeBetterCoffe'

WHERE Username = 'DrinkCode'

UPDATE Groups

SET Name = 'Lepka'

WHERE Name = 'Donisnik'

UPDATE Groups

SET Name = 'stalker'

WHERE Name = 'Slushalka'

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

DELETE FROM Groups

WHERE Name = 'Grandpa'

Conflict occurred due to FK\_KEY

1. **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.**

SET IDENTITY\_INSERT MyUsers.dbo.Users OFF

INSERT INTO MyUsers.dbo.Users ( FirstName, Username, PasswordUser, LastLogin)

SELECT TOP 1 e.FirstName + ' ' + e.LastName as [Full name], SUBSTRING(e.FirstName, 1, 1) + LOWER(e.LastName) as [Username], SUBSTRING(e.FirstName, 1, 1) + LOWER(e.LastName) as [Password] , NULL

FROM TelerikAcademy.dbo.Employees e

1. **Write a SQL statement that changes the password to NULL for all users that have not been in the system since 10.03.2010.**
2. **Write a SQL statement that deletes all users without passwords (NULL password).**
3. **Write a SQL query to display the average employee salary by department and job title.**

SELECT e.DepartmentID, e.JobTitle, d.Name as [Department Name], AVG(Salary) as [Average department salary]

FROM Employees e INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY e.DepartmentID, d.Name, e.JobTitle

ORDER BY DepartmentID

1. **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 e.DepartmentID, e.JobTitle, d.Name as [Department Name], MIN(Salary) as [Minimal department salary]

FROM Employees e INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY e.DepartmentID, d.Name, e.JobTitle

ORDER BY DepartmentID

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

SELECT COUNT(e.ManagerID) AS [Employees], t.Name

FROM TelerikAcademy.dbo.Employees e INNER JOIN TelerikAcademy.dbo.Addresses a

ON E.AddressID = a.AddressID

INNER JOIN TelerikAcademy.dbo.Towns t

ON a.TownID = t.TownID

INNER JOIN TelerikAcademy.dbo.Towns n

ON t.Name = n.Name

GROUP BY t.Name

HAVING COUNT(e.ManagerID) > 40

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

SELECT t.Name AS [City], COUNT(\*) AS [Managers]

FROM TelerikAcademy.dbo.Employees e INNER JOIN TelerikAcademy.dbo.Addresses a

ON e.AddressID = a.AddressID

INNER JOIN

TelerikAcademy.dbo.Towns t

ON a.TownID = t.TownID

INNER JOIN TelerikAcademy.dbo.Towns n

ON t.Name = n.Name

GROUP BY t.Name

1. **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.**

CREATE TABLE WorkHours

(

EployeeID INT NOT NULL IDENTITY PRIMARY KEY,

DateT DATE NOT NULL,

Task TEXT NOT NULL,

HoursT TIME NOT NULL,

Comments TEXT NOT NULL

)

INSERT INTO WorkHours (DateT, Task, HoursT, Comments)

VALUES ('2014/12/12', 'Design patter for Project X', '2014/12/12 21:00:00', 'Excellent employee')

INSERT INTO WorkHours (DateT, Task, HoursT, Comments)

VALUES ('2013/08/15', 'Fire lazy employee', '2013/08/15 14:25:33', 'Good worker')

INSERT INTO WorkHours (DateT, Task, HoursT, Comments)

VALUES ('2012/05/05', 'Meet with new clients', '2012/05/05 17:05:13', 'Very good worker')

UPDATE WorkHours

SET Task = N'Meet with mayor Kolov'

WHERE Task LIKE N'Fire lazy employee'

UPDATE WorkHours

SET Task = 'Go buy glasses in Japan'

WHERE Task LIKE 'Meet with new clients'

**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).**