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.

USE TelerikAcademy

SELECT e.FirstName + ' ' + e.LastName AS Name, e.Salary FROM Employees e

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.

USE TelerikAcademy

SELECT e.FirstName + ' ' + e.LastName AS Name, e.Salary FROM Employees e

WHERE Salary <

(SELECT MIN(Salary) FROM Employees) \* 1.1

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.

USE TelerikAcademy

SELECT d.Name, MIN(Salary) as [Minimal Salary For Department]

FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID

GROUP BY d.Name

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

USE TelerikAcademy

SELECT AVG(Salary) AS [Avarage Salary For Department #1]

FROM Employees е

WHERE DepartmentID = 1

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

USE TelerikAcademy

SELECT AVG(e.Salary) AS [Avarage Salary For Department]

FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

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

USE TelerikAcademy

SELECT AVG(e.Salary) AS [Avarage Salary For Sales Department]

FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

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

USE TelerikAcademy

SELECT COUNT(e.ManagerID) AS [Employees that have manager]

FROM Employees e

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

USE TelerikAcademy

SELECT COUNT(\*) AS [Employees that have no manager]

FROM Employees e

WHERE e.ManagerID IS NULL

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

USE TelerikAcademy

SELECT AVG(Salary) AS [Avarage Salary], d.Name AS [Department Name]

FROM Employees e JOIN Departments d ON e.DepartmentID = d.DepartmentID

GROUP BY d.Name ORDER BY [Avarage Salary] DESC

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

USE TelerikAcademy

SELECT dep.Name AS [Department], twn.Name AS [Town], COUNT(EmployeeID) AS [Employees Count] FROM Departments dep

JOIN Employees emp

ON dep.DepartmentID = emp.DepartmentID

JOIN Addresses adr

ON adr.AddressID = emp.AddressID

JOIN Towns twn

ON adr.TownID = twn.TownID

GROUP BY twn.Name, dep.Name

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

USE TelerikAcademy

SELECT man.FirstName, man.LastName FROM Employees emp

JOIN Employees man

ON emp.ManagerID = man.EmployeeID

GROUP BY man.FirstName, man.LastName

HAVING COUNT(man.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)".

USE TelerikAcademy

SELECT emp.FirstName + ' ' + emp.LastName AS [Employee Name],

COALESCE (man.FirstName + ' ' + man.LastName, 'No Manager') AS [Manager Name]

FROM Employees emp

LEFT JOIN Employees man

ON emp.ManagerID = man.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.

USE TelerikAcademy

SELECT emp.FirstName + ' ' + emp.LastName AS [Employee Name]

FROM Employees emp

WHERE LEN(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.

USE TelerikAcademy

SELECT CONVERT(varchar(10), GETDATE(), 104) + ' ' +

CONVERT(varchar(12),GETDATE(), 114)

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.

CREATE TABLE Users(

UserId int

IDENTITY(1, 1)

PRIMARY KEY CLUSTERED,

UserName varchar(20) NOT NULL

UNIQUE NONCLUSTERED,

Password varchar(20) NOT NULL

CHECK(LEN(Password) > 5),

FirstName varchar(20) NOT NULL,

LastName varchar(20) NOT NULL,

LastLogin DATETIME)

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.

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.

ALTER VIEW [Users View] AS

SELECT UserName, FirstName, LastName FROM Users

WHERE LastLogin = CONVERT(nvarchar(10), GETDATE(), 120)

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

PRIMARY KEY CLUSTERED

IDENTITY,

Name varchar(20) NOT NULL

UNIQUE)

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

FOREIGN KEY (GroupId) REFERENCES Groups(GroupId)

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

INSERT INTO dbo.Groups(Name)

VALUES('SecretGroup'),

('NotSoSecretGroup')

INSERT INTO Users(UserName, Password, FirstName, LastName, LastLogin, GroupId)

VALUES ('Киро', 'Пешо', 'Гошо', 'Иван, GETDATE(), 10),

('Пенка', 'Станка', 'Ганка', 'Луканка?', GETDATE(), 11)

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

UPDATE Users

SET Password = '12345Пенка'

WHERE UserName = 'Пенка'

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

DELETE FROM Users

WHERE UserName = 'Пенка'

DELETE FROM Groups

WHERE Name = 'SecretGroup'

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(UserName, Password, FirstName, LastName, LastLogin, GroupId)

SELECT CONVERT(nvarchar(20),emp.FirstName+emp.LastName), '12345Пенка', FirstName, LastName, GETDATE(), 14

FROM Employees emp

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(DATE, '2010.03.10', 102)

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.

USE TelerikAcademy

SELECT emp.JobTitle, dep.Name AS [Department name], AVG(emp.Salary) AS [Average Salary]

FROM Employees emp

JOIN Departments dep

ON dep.DepartmentID = emp.DepartmentID

GROUP BY emp.JobTitle, dep.Name

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.

USE TelerikAcademy

SELECT emp.FirstName + ' ' + emp.LastName AS [Full Name],

emp.JobTitle, dep.Name AS [Department name], emp.Salary

FROM Employees emp

JOIN Departments dep

ON dep.DepartmentID = emp.DepartmentID

WHERE emp.Salary =

(SELECT MIN(em.Salary) FROM Employees em

WHERE em.DepartmentID = emp.DepartmentID AND em.JobTitle = emp.JobTitle)

ORDER BY [Department name]

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

SELECT TOP(1) twn.Name FROM Employees emp

JOIN Addresses adr

ON emp.AddressID = adr.AddressID

JOIN Towns twn

ON adr.TownID = twn.TownID

GROUP BY twn.Name

ORDER BY COUNT(\*) DESC

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

SELECT twn.Name AS Town, COUNT(DISTINCT emp.ManagerID) AS Managers FROM Employees emp

JOIN Employees manag

ON emp.ManagerID = manag.EmployeeID

JOIN Addresses adr

ON manag.AddressID = adr.AddressID

JOIN Towns twn

ON twn.TownID = adr.TownID

GROUP BY twn.Name

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.

CREATE TABLE WorkHours

(Id int PRIMARY KEY

IDENTITY(1, 1),

EmployeeId int NOT NULL

FOREIGN KEY REFERENCES Employees(EmployeeId),

[Date] date NOT NULL,

Task nvarchar(20) NOT NULL,

[Hours] int NOT NULL,

Comments text)

INSERT INTO WorkHours(EmployeeId, Date, Task, Hours, Comments)

VALUES (111, GETDATE(), 'Write Code', 2, 'NOW!')

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

GO

Then =>

ROLLBACK TRAN

32. Find how to use temporary tables in SQL Server. Using temporary tables backup all records from EmployeesProjects and restore them back after dropping and re-creating the table.

INSERT INTO #TemporaryTable

SELECT \* FROM EmployeesProjects

DROP TABLE EmployeesProjects

CREATE TABLE EmployeesProjects(

EmployeeID int NOT NULL,

ProjectID int NOT NULL,

CONSTRAINT PK\_EmployeesProjects PRIMARY KEY(EmployeeID, ProjectID),

CONSTRAINT FK\_EP\_Employee FOREIGN KEY(EmployeeID) REFERENCES Employees(EmployeeID),

CONSTRAINT FK\_EP\_Project FOREIGN KEY(ProjectID) REFERENCES Projects(ProjectID)

)

INSERT INTO EmployeesProjects

SELECT \*

FROM #TemporaryTable