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

GO

SELECT FirstName, LastName, Salary

FROM Employees

WHERE Salary =

(SELECT MIN(Salary) FROM Employees)

1. **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

GO

SELECT FirstName, LastName, Salary

FROM Employees

WHERE Salary <=

(SELECT MIN(Salary) + (MIN(Salary) \* 0.1) FROM Employees)

ORDER BY Salary

1. **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

GO

SELECT e.FirstName, e.LastName, e.Salary AS MinimalSalary, d.Name

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE e.Salary =

(SELECT MIN(Salary) FROM Employees

WHERE DepartmentID = d.DepartmentID)

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

USE TelerikAcademy

**GO**

**SELECT TOP 1 DepartmentID, AVG(Salary)**

FROM Employees

GROUP BY DepartmentID

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

USE TelerikAcademy

GO

SELECT AVG(e.Salary) [Average Salary in Sales]

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

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

USE TelerikAcademy

GO

SELECT COUNT(\*) NumberOfEmployee

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

WHERE d.Name = 'Sales'

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

USE TelerikAcademy

GO

SELECT COUNT(\*) Cnt FROM Employees

WHERE ManagerID IS NOT NULL

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

USE TelerikAcademy

GO

SELECT COUNT(\*) Cnt FROM Employees

WHERE ManagerID IS NULL

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

USE TelerikAcademy

GO

SELECT d.DepartmentID, d.Name, AVG(e.Salary) AS AverageSalary

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID=d.DepartmentID

GROUP BY d.DepartmentID, d.Name

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

SELECT COUNT(e.EmployeeID) AS EmpCount, d.Name AS DeptName, t.Name AS Town

FROM Employees e

INNER JOIN Departments d ON e.DepartmentID = d.DepartmentID

INNER JOIN Addresses a ON a.AddressID = e.AddressID

INNER JOIN Towns t ON t.TownID = a.TownID

GROUP BY d.Name, t.Name

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

SELECT em.FirstName + ' ' + em.LastName AS ManagerName

FROM Employees e

INNER JOIN Employees em

ON e.ManagerID = em.EmployeeID

GROUP BY em.FirstName, em.LastName

HAVING COUNT(\*) = 5

1. **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 COALESCE(em.FirstName + ' ' + em.LastName, 'no manager') AS ManagerName, e.FirstName + ' ' + e.LastName AS EmployeeName

FROM Employees e

LEFT JOIN Employees em

ON e.ManagerID = em.EmployeeID

1. **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 LastName

FROM Employees

WHERE LEN(LastName) = 5

1. **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(10), GETDATE(), 104) + ' ' + CONVERT(VARCHAR(12), GETDATE(), 114)

1. **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.**

USE TelerikAcademy

GO

CREATE TABLE Users(

UserID int IDENTITY,

UserName nvarchar(50) NOT NULL,

UserPassword nvarchar(150) NOT NULL,

FullName nvarchar(100) NOT NULL,

LastLogIn datetime,

CONSTRAINT PK\_Users PRIMARY KEY(UserID),

CONSTRAINT UN\_UserName UNIQUE(UserName),

CONSTRAINT CH\_Password CHECK(LEN(UserPassword) > 4)

)

Go

INSERT Users (UserName, UserPassword, FullName, LastLogIn) VALUES('Gosho', 'qwerty', 'GoshoTosho', GETDATE())

GO

INSERT Users (UserName, UserPassword, FullName, LastLogIn) VALUES('Gosho2', 'qwerty', 'GoshoTosho', GETDATE())

GO

INSERT Users (UserName, UserPassword, FullName, LastLogIn) VALUES('Gosho3', 'qwerty', 'GoshoTosho', GETDATE())

GO

INSERT Users (UserName, UserPassword, FullName, LastLogIn) VALUES('Gosho4', 'qwerty', 'GoshoTosho', GETDATE())

GO

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 [Today User Registrations] AS

SELECT \*

FROM Users

WHERE DAY(LastLogIn) = DAY(GETDATE())

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(

GroupID int IDENTITY,

GroupName nvarchar(50) NOT NULL,

CONSTRAINT PK\_Groups PRIMARY KEY(GroupID),

CONSTRAINT UN\_GroupName UNIQUE(GroupName)

)

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.**

--create new column in users--

ALTER TABLE Users ADD GroupID int

--fill group table—

GO

INSERT Groups (GroupName) VALUES ('Administrators')

INSERT Groups (GroupName) VALUES ('CommonUsers')

INSERT Groups (GroupName) VALUES ('Guests')

GO

--Load Some GroupIDs in Users--

UPDATE Users SET GroupID = 1 WHERE UserID = 1

UPDATE Users SET GroupID = 3 WHERE UserID = 2

UPDATE Users SET GroupID = 2 WHERE UserID = 3

UPDATE Users SET GroupID = 1 WHERE UserID = 4

--Add foreignKey constraints—

ALTER TABLE Users

ADD CONSTRAINT FK\_Users\_Groups

FOREIGN KEY (GroupID)

REFERENCES Groups(GroupID)

GO

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

ALREADY DONE UP

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

UPDATE Users SET UserName = 'Kolio' WHERE UserID = 1

UPDATE Groups SET GroupName = 'Bok' WHERE GroupID = 3

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

DELETE FROM Users

WHERE UserID = 2

GO

DELETE FROM Groups

WHERE GroupID = 2

GO

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.**

INSERT INTO Users (UserName, UserPassword, FullName, LastLogIn)

SELECT LEFT(e.FirstName, 1) + LOWER(e.LastName) AS UserName,

LEFT(e.FirstName, 1) + LOWER(e.LastName) AS UserPassword,

e.FirstName + ' ' + e.LastName AS FullName,

NULL AS LastLogIn

FROM Employees e

WHERE (1 + LEN(e.LastName)) >=5 AND LEFT(LOWER(e.FirstName),1) + LOWER(e.LastName) <> 'ahill'

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.**

UPDATE Users SET UserPassword = NULL

WHERE CAST(LastLogin AS date) <= CONVERT(datetime, '10.03.2010',104)

GO

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

DELETE FROM Users

WHERE UserPassword IS NULL

GO

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

SELECT d.Name, e.JobTitle, AVG(e.Salary) AS AverageSalary

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID = e.DepartmentID

GROUP BY d.Name, e.JobTitle

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 MIN(e.Salary) AS MinSalary, d.Name AS DepName, MIN(e.FirstName + ' ' + e.LastName) AS FullName

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID = e.DepartmentID

GROUP BY e.JobTitle, d.Name

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

SELECT TOP 1 t.Name, COUNT(\*) AS Citizens

FROM Employees e

INNER JOIN Addresses a

ON a.AddressID = e.AddressID

INNER JOIN Towns t

ON t.TownID = a.TownID

GROUP BY t.Name

ORDER BY COUNT(\*) DESC

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

SELECT t.Name as Town, COUNT(e.ManagerID) AS ManagersCount

FROM Employees e

JOIN Addresses a ON e.AddressID = a.AddressID

join Towns t ON t.TownID = a.TownID

WHERE e.EmployeeID in (SELECT DISTINCT ManagerID FROM Employees)

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.**

**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).**

USE TelerikAcademy

GO

CREATE TABLE WorkHours

(

WorkHourEntryID int IDENTITY,

EntryDate date NOT NULL,

Task nvarchar(100) NOT NULL,

EntryHours int NOT NULL,

EntryComments nvarchar(200),

EmployeeID int NOT NULL,

CONSTRAINT PK\_WorkHourEntryID PRIMARY KEY(WorkHourEntryID),

CONSTRAINT CC\_Hours CHECK (EntryHours>0 AND EntryHours<=24)

)

GO

ALTER TABLE [dbo].[WorkHours] WITH CHECK ADD CONSTRAINT [FK\_WorkHours\_Employees] FOREIGN KEY([EmployeeID])

REFERENCES [dbo].[Employees] ([EmployeeID])

GO

CREATE TABLE WorkHoursLogs

(

LogID int IDENTITY,

WorkHourEntryIDOld int,

EntryDateOld date,

TaskOld nvarchar(100),

EntryHoursOld int,

EntryCommentsOld nvarchar(200),

EmployeeIDOld int,

WorkHourEntryIDNew int,

EntryDateNew date,

TaskNew nvarchar(100),

EntryHoursNew int,

EntryCommentsNew nvarchar(200),

EmployeeIDNew int,

CommandType nvarchar(10),

CONSTRAINT PK\_LogID PRIMARY KEY(LogID),

)

GO

CREATE TRIGGER tr\_WorkHoursInsert ON WorkHours FOR INSERT

AS

INSERT INTO WorkHoursLogs (WorkHourEntryIDOld,EntryDateOld,TaskOld,EntryHoursOld,EntryCommentsOld,

EmployeeIDOld,WorkHourEntryIDNew,EntryDateNew,TaskNew,EntryHoursNew,EntryCommentsNew,

EmployeeIDNew,CommandType)

SELECT NULL, NULL, NULL, NULL, NULL, NULL,

i.WorkHourEntryID, i.EntryDate, i.Task, i.EntryHours, i.EntryComments, i.EmployeeID,'insert'

FROM WorkHours w INNER JOIN inserted i on w.WorkHourEntryID = i.WorkHourEntryID

GO

CREATE TRIGGER tr\_WorkHoursUpdate ON WorkHours FOR UPDATE

AS

INSERT INTO WorkHoursLogs (WorkHourEntryIDOld,EntryDateOld,TaskOld,EntryHoursOld,EntryCommentsOld,

EmployeeIDOld,WorkHourEntryIDNew,EntryDateNew,TaskNew,EntryHoursNew,EntryCommentsNew,

EmployeeIDNew,CommandType)

SELECT d.WorkHourEntryID, d.EntryDate, d.Task, d.EntryHours, d.EntryComments, d.EmployeeID,

i.WorkHourEntryID, i.EntryDate, i.Task, i.EntryHours, i.EntryComments, i.EmployeeID,'update'

FROM deleted d INNER JOIN inserted i on d.WorkHourEntryID = i.WorkHourEntryID

GO

CREATE TRIGGER tr\_WorkHoursDelete ON WorkHours FOR DELETE

AS

INSERT INTO WorkHoursLogs (WorkHourEntryIDOld,EntryDateOld,TaskOld,EntryHoursOld,EntryCommentsOld,

EmployeeIDOld,WorkHourEntryIDNew,EntryDateNew,TaskNew,EntryHoursNew,EntryCommentsNew,

EmployeeIDNew,CommandType)

SELECT d.WorkHourEntryID, d.EntryDate, d.Task, d.EntryHours, d.EntryComments, d.EmployeeID,

NULL, NULL, NULL, NULL, NULL, NULL,'delete'

FROM deleted d

GO

INSERT INTO WorkHours (EntryDate, Task, EntryHours, EntryComments, EmployeeID)

VALUES (GETDATE(), 'Slacking', 8, 'bad work', 1),

(CONVERT(date, '20140511', 112), 'Task1', 4, 'bad work', 3),

(CONVERT(date, '20140713', 112), 'Task2', 4, 'good work', 1),

(CONVERT(date, '20140611', 112), 'Task3', 12, 'more work', 2),

(CONVERT(date, '20140530', 112), 'Task4', 5, 'good work' , 4)

GO

UPDATE WorkHours

SET EntryHours = 8, Task ='hablahabla', EntryComments = 'no homework anymore'

WHERE EmployeeID=1

GO

DELETE FROM WorkHours

WHERE EmployeeID = 2

1. **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

ALTER TABLE Departments DROP [FK\_Departments\_Employees]

DELETE FROM Employees

WHERE DepartmentID = (SELECT DepartmentID FROM Departments WHERE Name = 'Sales')

DELETE FROM Departments

WHERE Name = 'Sales'

ROLLBACK TRAN

GO

1. **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

GO

1. **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.**

CREATE TABLE #TemporaryTable(

EmployeeID int NOT NULL,

ProjectID int NOT NULL

)

INSERT INTO #TemporaryTable

SELECT EmployeeID, ProjectID

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 EmployeeID, ProjectID

FROM #TemporaryTable

CREATE DATABASE MyProjectDataBase

USE MyProjectDataBase

GO

CREATE TABLE Persons(

PersonID int IDENTITY,

FirstName nvarchar(50) NOT NULL,

LastName nvarchar(50) NOT NULL,

SSN nvarchar(10) NOT NULL

CONSTRAINT PK\_PersonID PRIMARY KEY(PersonID)

)

CREATE TABLE Accounts(

AccountID int IDENTITY,

PersonID int

CONSTRAINT PK\_AccountID PRIMARY KEY(AccountID)

)