# Advanced SQL Homework

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 FirstName + ' ' + LastName AS [Full Name], 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

SELECT FirstName + ' ' + LastName AS [Full Name], Salary

FROM Employees

WHERE Salary < (SELECT (MIN(Salary) \* 1.1) 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

SELECT FirstName + ' ' + LastName AS [Full Name], Salary, DepartmentID

FROM Employees e

WHERE Salary = (SELECT MIN(Salary) FROM Employees

WHERE DepartmentID = e.DepartmentID)

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

USE TelerikAcademy

SELECT AVG(Salary) AS [Average Department 1 Salary]

FROM Employees

WHERE DepartmentID = 1

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

USE TelerikAcademy

SELECT AVG(Salary) AS [Average Department 'Sales' Salary]

FROM Employees e

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

SELECT COUNT(\*) AS [Workers in Sales department]

FROM Employees e

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

SELECT COUNT(\*) AS [Count of workers with manager]

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

SELECT COUNT(\*) AS [Count of workers with manager]

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

SELECT DepartmentID, AVG(Salary) AS [Average Salary]

FROM Employees

GROUP BY DepartmentID

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

USE TelerikAcademy

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

FROM Employees e

JOIN Addresses a

ON e.AddressID = a.AddressID

JOIN Towns t

ON a.TownID = t.TownID

GROUP BY e.DepartmentID, t.Name

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

USE TelerikAcademy

SELECT m.FirstName, m.LastName, COUNT(m.EmployeeID) AS [Employees count]

FROM Employees e

JOIN Employees m

ON e.ManagerID = m.EmployeeID

GROUP BY m.FirstName, m.LastName

HAVING COUNT(m.EmployeeID) = 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)".**

USE TelerikAcademy

SELECT e.FirstName + ' ' + e.LastName AS Employee,

ISNULL(m.FirstName + ' ' + m.LastName, 'no manager')

FROM Employees e

LEFT OUTER JOIN Employees m

ON e.ManagerID = m.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.**

USE TelerikAcademy

SELECT FirstName, 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(24),GETDATE(), 113) AS [Current time]

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

CREATE TABLE Users (

UserID int IDENTITY,

Username nvarchar(50) NOT NULL UNIQUE,

Password nvarchar(50) CHECK(LEN(Password) >= 5),

FullName nvarchar(50) NOT NULL,

LastLoginName datetime NOT NULL,

CONSTRAINT PK\_Users PRIMARY KEY(UserID)

)

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.**

USE TelerikAcademy

GO

CREATE VIEW TodayUsers AS

SELECT \*

FROM Users

WHERE DAY(GETDATE() - LastLoginTime) = 1

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

USE TelerikAcademy

GO

CREATE TABLE Groups(

GroupID int IDENTITY PRIMARY KEY,

Name nvarchar(50) 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.**

USE TelerikAcademy

GO

ALTER TABLE Users

ADD GroupID int

CONSTRAINT FK\_Users\_Groups

FOREIGN KEY(GroupID)

REFERENCES Groups(GroupID)

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

USE TelerikAcademy

GO

INSERT INTO TelerikAcademy.dbo.Users (Username, Password, FullName, LastLoginTime, GroupID)

VALUES ('Ivaylo', '543123', 'Ivaylo Kenov', GETDATE(), 2),

('Doncho', 'Bourgas', 'Doncho Minkov', GETDATE(), 1),

('Niki', 'telerikacademy', 'Nikolay Kostov', GETDATE(),3)

INSERT INTO Groups (Name)

VALUES ('Designers'),

('Testers'),

('Developers')

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

USE TelerikAcademy

GO

UPDATE Users

SET Password = 'IvayloKenov'

WHERE Username = 'Ivaylo'

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

USE TelerikAcademy

GO

DELETE FROM Users

WHERE Username = 'Ivaylo'

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.**

USE TelerikAcademy

GO

INSERT INTO Users

SELECT

LOWER(LEFT(e.FirstName, 3) + e.LastName) AS [Username], /\*take 3 chars to avoid duplication\*/

LOWER('password\_' + LEFT(e.FirstName, 3) + e.LastName) AS [Password],

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

NULL AS [LastLoginTime],

1 AS [GroupID]

FROM 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.**

USE TelerikAcademy

GO

UPDATE Users

SET LastLoginTime = NULL

WHERE LastLoginTime < '10.03.2010'

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

USE TelerikAcademy

GO

DELETE FROM Users

WHERE Password IS NULL

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

USE TelerikAcademy

SELECT DepartmentID, JobTitle, AVG(Salary) AS [Average Salary]

FROM Employees

GROUP BY DepartmentID, 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.**

USE TelerikAcademy

SELECT DepartmentID, JobTitle,

FirstName + ' ' + LastName AS [Full Name],

MIN(Salary) AS [Average Salary]

FROM Employees

GROUP BY DepartmentID, JobTitle, FirstName, LastName

ORDER BY DepartmentID

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

USE TelerikAcademy

SELECT TOP 1 t.Name AS Town, COUNT(t.TownID) AS [Employees Count]

FROM Employees e

JOIN Addresses a

ON e.AddressID = a.AddressID

JOIN Towns t

ON a.TownID = t.TownID

GROUP BY t.TownID, t.Name

ORDER BY [Employees Count] DESC

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

USE TelerikAcademy

SELECT t.Name AS [Town], COUNT(ManagerID) AS [Manager Count]

FROM Employees e

JOIN Addresses a

ON e.AddressID = a.AddressID

JOIN Towns t

ON a.TownID = t.TownID

GROUP BY t.Name, e.ManagerID

ORDER BY [Manager Count] DESC

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

CREATE TABLE WorkHours(

WorkHoursID int IDENTITY PRIMARY KEY,

EmployeeID int NOT NULL,

Date datetime,

Task nvarchar(250),

Hours int,

Comments nvarchar(250),

CONSTRAINT FK\_WorkHours\_Employees

FOREIGN KEY(EmployeeID)

REFERENCES Employees(EmployeeID)

)

GO

INSERT INTO WorkHours (EmployeeID, Date, Task, Hours)

VALUES (12, GETDATE(), 'No tasks', 8),

(1, GETDATE(), 'Work', 12),

(4, GETDATE(), 'Manage', 10),

(122, GETDATE(), 'Code', 8),

(31, GETDATE(), 'Code', 8)

GO

UPDATE WorkHours

SET Hours = 6

WHERE EmployeeID = 4

GO

UPDATE WorkHours

SET Hours = 9

WHERE Task = 'Code'

GO

DELETE FROM WorkHours

WHERE Task = 'No tasks'

GO

CREATE TABLE WorkHoursLog(

LogID INT IDENTITY PRIMARY KEY,

ExecutedCommand nvarchar(20) NULL,

WorkHoursID INT FOREIGN KEY REFERENCES WorkHours(WorkHoursID),

OldEmployeeID INT,

OldDate datetime,

OldTask nvarchar(250),

OldHours INT,

OldComments nvarchar(250),

NewEmployeeID INT,

NewDate datetime,

NewTask nvarchar(250),

NewHours INT,

NewComments nvarchar(250)

)

GO

CREATE TRIGGER utr\_WorkHoursDelete

ON WorkHours

FOR DELETE

AS

INSERT INTO WorkHoursLog

SELECT 'DELETE', \*, NULL, NULL, NULL, NULL, NULL

FROM deleted

GO

CREATE TRIGGER utr\_WorkHoursInsert

ON WorkHours

FOR INSERT

AS

INSERT INTO WorkHoursLog

SELECT 'INSERT', WorkHoursID,NULL, NULL, NULL, NULL, NULL,

EmployeeID, Date, Task, Hours, Comments

FROM inserted

GO

CREATE TRIGGER utr\_WorkHoursUpdate

ON WorkHours

FOR UPDATE

AS

INSERT INTO WorkHoursLog

SELECT 'UPDATE', d.WorkHoursID, d.EmployeeID, d.Date, d.Task, d.Hours, d.Comments,

i.EmployeeID, i.Date, i.Task, i.Hours, i.Comments

FROM inserted i, deleted d

GO

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.**

USE TelerikAcademy

BEGIN TRAN

DELETE FROM Employees

WHERE DepartmentID IN (SELECT DepartmentID FROM Departments

WHERE Name = 'Sales')

ROLLBACK TRAN

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

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.**

USE TelerikAcademy

CREATE TABLE #TempTable (

EmployeeID INT,

ProjectID INT

)

INSERT INTO #TempTable

SELECT ep.EmployeeID,

ep.ProjectID

FROM EmployeesProjects ep

DROP TABLE EmployeesProjects

CREATE TABLE EmployeesProjects (

EmployeeID INT,

ProjectID INT

)