**Advanced SQL**

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, e.Salary

FROM Employees e

WHERE e.Salary =

(SELECT MIN(innerE.Salary) FROM Employees innerE)

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 e.FirstName, e.LastName, e.Salary

FROM Employees e

WHERE e.Salary <=

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

ORDER BY e.Salary DESC

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 e.FirstName + ' ' + e.LastName AS 'Full Name', e.Salary, d.Name AS 'Department'

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID= d.DepartmentID

WHERE e.Salary =

(SELECT MIN(innerEmpl.Salary) FROM Employees innerEmpl

WHERE innerEmpl.DepartmentID= e.DepartmentID)

ORDER BY e.Salary

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

USE TelerikAcademy

SELECT e.DepartmentID, d.Name AS 'Departement Name', AVG(e.Salary) AS 'Department Average Salary'

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID= e.DepartmentID

GROUP BY e.DepartmentID, d.Name

HAVING e.DepartmentID = 1

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

USE TelerikAcademy

SELECT e.DepartmentID, d.Name AS 'Departement Name', AVG(e.Salary) AS 'Department Average Salary'

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID= e.DepartmentID

GROUP BY e.DepartmentID, d.Name

HAVING d.Name = 'Sales'

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

SELECT e.DepartmentID, d.Name AS 'Departement Name',COUNT(e.EmployeeID) AS 'Emplyees'

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID= e.DepartmentID

GROUP BY e.DepartmentID, d.Name

HAVING d.Name = 'Sales'

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

USE TelerikAcademy

SELECT e.DepartmentID, d.Name AS 'Departement Name',COUNT(e.EmployeeID) AS 'Employ Count'

FROM Employees e

INNER JOIN Departments d

ON d.DepartmentID = e.DepartmentID

WHERE e.ManagerID IS NOT NULL

GROUP BY e.DepartmentID, d.Name

ORDER BY e.DepartmentID

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

USE TelerikAcademy

SELECT COUNT(e.EmployeeID) AS 'Employes Without Manager'

FROM Employees e

WHERE e.ManagerID IS NULL

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

USE TelerikAcademy

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

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.

USE TelerikAcademy

SELECT e.DepartmentID,d.Name, t.Name, COUNT(e.EmployeeID) AS 'Employe Count'

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

INNER JOIN Addresses a

ON e.AddressID = a.AddressID

INNER JOIN Towns t

ON t.TownID = a.TownID

GROUP BY e.DepartmentID, d.Name, a.TownID, t.Name

ORDER BY e.DepartmentID

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 'Employes'

FROM Employees e

INNER JOIN Employees m

ON e.ManagerID = m.EmployeeID

GROUP BY m.EmployeeID, m.FirstName, m.LastName

HAVING COUNT(m.EmployeeID)= 5

ORDER BY m.FirstName, m.LastName

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 'Employe Name',ISNULL(m.FirstName +' '+m.LastName,'no manager') AS 'Manager Name'

FROM Employees e

LEFT 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-inLEN(str) function.

SELECT e.FirstName, e.LastName, LEN(e.LastName) AS 'Last Name Len'

FROM Employees e

WHERE LEN(e.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.

USE TelerikAcademy

SELECT GETDATE() AS 'Standart',

REPLACE(CONVERT(VARCHAR(10), GETDATE(), 104)+' ' + RIGHT(CONVERT(VARCHAR(26), GETDATE(), 109),14),'PM','') AS 'day.month.year hour:minutes:seconds:milliseconds',

CONVERT(VARCHAR(24), GETDATE(), 113) AS 'My try'

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.

CREATE TABLE Users (

UserID int IDENTITY,

UserName NVARCHAR(100) UNIQUE NOT NULL,

[Password] NVARCHAR(100) CHECK (DATALENGTH([Password]) >= 5) NOT NULL,

Name NVARCHAR(50),

LoginTime DATETIME NOT NULL

CONSTRAINT PK\_Users PRIMARY KEY(UserID)

)

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 Users] AS

SELECT \*

FROM Users u

WHERE DATEPART(DAY,u.LoginTime) = DATEPART(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,

Name nvarchar(100) UNIQUE NOT NULL,

CONSTRAINT PK\_Groups PRIMARY KEY(GroupID)

)

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

CONSTRAINT FK\_Users\_Groups

FOREIGN KEY (GroupID)

REFERENCES Groups(GroupID)

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

INSERT INTO Groups(Name)

VALUES('OOP'),

('HQC'),

('DB');

INSERT INTO Users(Username, Password, LoginTime,Name ,GroupID)

VALUES('John123','123456', GETDATE(),'John',1),

('John223','123456', GETDATE(),'John2',1),

('Jane3','123456', GETDATE(),'Jane',3);

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

UPDATE Users

SET Name = 'New John'

WHERE UserID = 2;

UPDATE Groups

SET Name= 'DATABASES'

WHERE Name = 'DB';

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

DELETE FROM Users

WHERE Username LIKE 'user%'

DELETE FROM Groups

WHERE GroupID = 2

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,Password, Name)

SELECT DISTINCT

LOWER(LEFT(e.FirstName,3) + e.LastName) AS Username,

LOWER(LEFT(e.FirstName,3) + e.LastName) ,

e.FirstName+ ' '+ e.LastName

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.

UPDATE Users

SET Password= 'NULL'

WHERE LoginTime < '2010-10-03'

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

DELETE Users

WHERE Password= 'NULL'

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

SELECT d.Name AS 'Departement', 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

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

MIN(e.Salary) AS 'Min Salary',

e.JobTitle,

d.Name AS 'Departement'

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID= d.DepartmentID

GROUP BY d.Name, e.JobTitle,e.Salary, e.FirstName, e.LastName

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

SELECT TOP 1 t.Name, COUNT(e.EmployeeID) AS 'Emloyees'

FROM Employees e

INNER JOIN Addresses a

ON e.AddressID= a.AddressID

INNER JOIN Towns t

ON t.TownID = a.TownID

GROUP BY t.Name

ORDER BY COUNT(e.EmployeeID) DESC

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

SELECT t.Name, COUNT(DISTINCT e.ManagerID) AS 'Managers'

FROM Employees e

INNER JOIN Employees m

ON e.ManagerID = m.EmployeeID

INNER JOIN Addresses a

ON e.AddressID= a.AddressID

INNER JOIN Towns t

ON t.TownID = a.TownID

WHERE m.ManagerID IS NOT NULL

GROUP BY t.Name

ORDER BY COUNT(m.ManagerID) 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).

CREATE TABLE WorkHours(

WorkHourID INT IDENTITY,

EmployeeID INT NOT NULL,

[Date] DATE NOT NULL,

Task NVARCHAR(150) NOT NULL,

[Hours] INT NOT NULL,

Comments NVARCHAR(1000),

CONSTRAINT PK\_WorkHourID PRIMARY KEY(WorkHourID),

CONSTRAINT FK\_WorkHours\_Employees

FOREIGN KEY (EmployeeID)

REFERENCES Employees(EmployeeID)

)

GO

INSERT INTO WorkHours(EmployeeID, [Date], Task, [Hours], Comments)

VALUES(15,GETDATE(),'Task1',15,'Just do it!'),

(35,GETDATE(),'Task2',35,'Do it again!'),

(5,GETDATE(),'Task2',5,'And again!'),

(17,GETDATE(),'Task3',105,'And again!')

GO

CREATE TABLE WorkHoursLogs (

WorkHoursLogID INT IDENTITY,

WorkHourID INT,

EmployeeID INT NOT NULL,

[Date] DATETIME,

Task NVARCHAR(150),

[Hours] INT,

Comments NVARCHAR(1000),

Command nvarchar(30) NOT NULL,

CONSTRAINT PK\_WorkHoursLogs PRIMARY KEY(WorkHoursLogID),

CONSTRAINT FK\_WorkHoursLogs\_Employees FOREIGN KEY(EmployeeID)

REFERENCES Employees(EmployeeID)

)

GO

CREATE TRIGGER TR\_WorhoursInsert ON WorkHours FOR INSERT

AS

INSERT INTO WorkHoursLogs(WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, Command)

SELECT WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, 'INSERT'

FROM inserted

GO

CREATE TRIGGER TR\_WorhoursUpdate ON WorkHours FOR UPDATE

AS

INSERT INTO WorkHoursLogs(WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, Command)

SELECT WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, 'UPDATE'

FROM inserted

GO

CREATE TRIGGER TR\_WorhoursDelete ON WorkHours FOR DELETE

AS

INSERT INTO WorkHoursLogs(WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, Command)

SELECT WorkHourID, EmployeeID, [Date], Task, [Hours], Comments, 'DELETE'

FROM deleted

GO

INSERT INTO WorkHours(EmployeeID, [Date], Task, [Hours], Comments)

VALUES

(10, GETDATE(),'Test 1', 2, 'Test insert'),

(11, GETDATE(),'Test 2', 4, 'Test insert')

UPDATE WorkHours

SET Task = 'Test Udate'

WHERE EmployeeID = 11

DELETE FROM WorkHours

WHERE Task= 'Test again'

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 CONSTRAINT FK\_Departments\_Employees

DELETE FROM Employees

WHERE DepartmentID =

(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.

BEGIN TRAN

CREATE TABLE #TempTable (EmployeeID int, ProjectID int)

SELECT EmployeeID, ProjectID

FROM EmployeesProjects

DROP TABLE EmployeesProjects;

CREATE TABLE EmployeesProjects(EmployeeID int, ProjectID int)

SELECT EmployeeID, ProjectID

FROM #TempTable

ROLLBACK TRAN