# 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.

* 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.

* SELECT FirstName, LastName, Salary

FROM Employees

WHERE

(Salary > (SELECT MIN(Salary) FROM Employees) AND

Salary <= (SELECT MIN(Salary) \* 1.1 From Employees))

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.

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

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.

* SELECT AVG(Salary) [Average Salary]

FROM Employees

WHERE DepartmentID = 1

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

* SELECT AVG(Salary) [Average Salary]

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.

* SELECT COUNT(\*) [Employees Count]

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.

* SELECT COUNT(\*) [Employees Count]

FROM Employees

WHERE ManagerID IS NOT NULL

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

* SELECT COUNT(\*) [Employees Count]

FROM Employees

WHERE ManagerID IS NULL

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

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

FROM Employees e

INNER JOIN Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY d.Name

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

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

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.

* SELECT FirstName, LastName

FROM Employees m

WHERE 5 =

(

SELECT COUNT (\*)

FROM Employees e

WHERE e.ManagerID = m.EmployeeID

)

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 e.FirstName + ' ' + e.LastName AS [Employee Full Name], ISNULL(m.FirstName +' '+ m.LastName, 'no manager') AS [Manager Full Name]

FROM Employees m

RIGHT OUTER JOIN Employees e

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.

* SELECT FirstName, LastName

FROM Employees

WHERE 5 = LEN(LastName)

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(50), GETDATE(), 13) AS [DATE]

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(20) UNIQUE NOT NULL,

Password nvarchar(25) CHECK(LEN([Password]) > 4),

Fullname nvarchar(50) NOT NULL,

LastLoginTime datetime

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.

* CREATE VIEW UsersToday AS

SELECT Username, DAY(GETDATE() - LastLoginTime) AS DayDifference

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.

* CREATE TABLE Groups (

GroupID int IDENTITY,

Name nvarchar(25) UNIQUE,

CONSTRAINT PK\_Groups PRIMARY KEY (GroupID)

)

GO

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 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

VALUES (N'Gamers'), (N'Programmers')

GO

* INSERT INTO Users VALUES

(N'pesho', N'12345', N'Peter Petrov', CAST(N'2014-08-25 00:00:00.000' AS DateTime), 1),

(N'gosho', N'12345', N'George Georgiev', CAST(N'2014-08-24 00:00:00.000' AS DateTime), 2),

(N'gesho', N'12345', N'Gesho Geshev', CAST(N'2014-08-26 00:00:00.000' AS DateTime), 1)

GO

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

* UPDATE Groups

SET Name = 'Mathematicians'

WHERE GroupID = 1

* UPDATE Users

SET Password = 'secret'

WHERE Username = 'gosho'

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

* DELETE FROM Users Where GroupID = 2

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], Fullname)

SELECT LOWER(LEFT(FirstName, 3) + LastName),

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

(FirstName + ' ' + LastName)

FROM Employees

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 LastLoginTime < CONVERT(datetime, '10-03-2010')

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

* DELETE FROM Users

WHERE [Password] IS NULL

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

* SELECT AVG(e.Salary) AS [Average Salary], e.JobTitle, d.Name AS [Department Name]

FROM Employees e

JOIN Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY e.JobTitle, d.Name

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 [Full Name],

e.Salary AS [Minimum Salary], e.JobTitle,

d.Name AS [Department Name]

FROM Employees e

JOIN Departments d

ON e.DepartmentID = d.DepartmentID

GROUP BY e.JobTitle, d.Name, e.Salary, e.FirstName + ' ' + e.LastName, e.DepartmentID

HAVING e.Salary =

(

SELECT MIN(Salary)

FROM Employees

WHERE JobTitle = e.JobTitle AND DepartmentID = e.DepartmentID

)

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

* SELECT TOP(1) t.Name, COUNT(e.EmployeeID) AS [Working Employees Number]

FROM Towns t

JOIN Addresses a

ON t.TownID = a.TownID

JOIN Employees e

ON e.AddressID = a.AddressID

Group By t.Name

ORDER BY [Working Employees Number] DESC

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

* SELECT COUNT(DISTINCT m.EmployeeID), t.Name

FROM Employees m

INNER JOIN Employees e

ON e.ManagerID = m.EmployeeID

INNER JOIN Addresses a

ON a.AddressID = m.AddressID

INNER JOIN Towns t

ON a.TownID = t.TownID

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).

* Don’t know triggers yet!

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

DELETE FROM Employees

WHERE DepartmentID IN

(SELECT DepartmentID 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

)

GO

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\_EmployeesProjects\_Employees FOREIGN KEY(EmployeeID) REFERENCES Employees(EmployeeID),

CONSTRAINT FK\_EmployeesProjects\_Projects FOREIGN KEY(ProjectID) REFERENCES Projects(ProjectID)

)

INSERT INTO EmployeesProjects

SELECT EmployeeID, ProjectID

FROM #TemporaryTable