IST 659: Lab 8

1) Add an employee using your own name and create a project assignment for your self using existing project id.

SQL:

```
/*Inserting new entries and displaying employee and ProjectAssignment Table
*/

INSERT INTO employee VALUES (9999, 'Farees', 'Patel', 80000);
INSERT INTO projectAssignment VALUES (9999, 1002, 'developer');
SELECT * FROM employee
SELECT * FROM projectAssignment
```

Output:

	employeeID	employeeFName	employeeLName	employeeSalary
1	9999	Farees	Patel	80000
2	11111	James	Smith	30000
3	11112	Ada	Zack	40000
4	11113	Ben	Yale	50000
5	11114	Callen	Wales	60000
6	11115	Dale	Veller	70000
7	11116	Ethan	Miller	80000
8	11117	Fanny	Niel	90000

	employeeID	projectID	projectRole
1	9999	1002	developer
2	11111	1001	manager
3	11112	1001	PHP programmer
4	11113	1001	Oracle DBA
5	11113	1002	test engineer
6	11114	1001	quality assurance
7	11114	1002	quality assurance
8	11115	1001	test engineer

2) Write a scalar function that returns the average salary of the Employees

SQL:

```
/*Creating a function a scalar function to return the average salary of the
employees in the employee table */

CREATE FUNCTION average_salary()
RETURNS DECIMAL(10,2)
AS
BEGIN
DECLARE @ret INT;
SELECT @ret=AVG(employeeSalary)
FROM employee
RETURN @ret;
END;
```

```
/*Displaying the average salary of the employees */

SELECT dbo.average_salary() AS 'AVERAGE SALARY';
```

Output:

	AVERAGE SALARY
1	62500.00

3) Write a table-valued function that returns the Projects given an EmployeeID as a parameter and
   a. Show the function created
   b. return the results for your own project

SQL:

```
/*Creating a table-valued function to pass the EmployeeID and returning the
projects assigned to the given employee*/

CREATE FUNCTION employee_projects(@EmployeeID INT)
RETURNS TABLE
AS
RETURN
(SELECT e.employeeID AS 'Employee ID', pa.projectID  AS 'Projects
Assigned', p.projectDesc AS 'Project Name', p.projectStartDate,
p.projectDuration
FROM employee e
INNER JOIN projectAssignment pa ON e.employeeID = pa.employeeID
INNER JOIN project p ON pa.projectID=p.projectID
WHERE e.employeeID = @EmployeeID
);

SELECT * FROM dbo.employee_projects(9999);
```

Output:

	Employee ID	Projects Assigned	Project Name	projectStartDate	projectDuration
1	9999	1002	new website	2010-01-01 00:00:00.000	six months

4) Alter the Employee table to add a new column called ‘Num of Projects’ which can be **INTEGER** data type. Write a procedure that updates employee table with the total projects assigned to each employee (to the newly created column)

SQL:

```
/*Alter the employee table to add an additional column and displaying employee
Table */

alter table employee add NumOfProjects INT;
SELECT * FROM employee
```

Output:

	employeeID	employeeFName	employeeLName	employeeSalary	NumOfProjects
1	9999	Farees	Patel	80000	NULL
2	11111	James	Smith	30000	NULL
3	11112	Ada	Zack	40000	NULL
4	11113	Ben	Yale	50000	NULL
5	11114	Callen	Wales	60000	NULL
6	11115	Dale	Veller	70000	NULL
7	11116	Ethan	Miller	80000	NULL
8	11117	Fanny	Niel	90000	NULL

SQL:

```
/*Creating procedure to update the new NumofProjects column by the number
of projects assigned to every given employee */

CREATE PROCEDURE every_employee_total_projects
AS
BEGIN
UPDATE employee
SET NumOfProjects =Projectnum.NumOfProjects
FROM
(SELECT e.employeeID, count(pa.projectID) AS 'NumOfProjects'
FROM employee e
INNER JOIN projectAssignment pa ON e.employeeID = pa.employeeID
INNER JOIN project p ON p.projectID = pa.projectID
GROUP BY e. employeeID) AS Projectnum
WHERE employee.employeeID = projectnum.employeeID
END;

EXEC every_employee_total_projects

SELECT * FROM employee
```

Output:

	employeeID	employeeFName	employeeLName	employeeSalary	NumOfProjects
1	9999	Farees	Patel	80000	1
2	11111	James	Smith	30000	1
3	11112	Ada	Zack	40000	1
4	11113	Ben	Yale	50000	2
5	11114	Callen	Wales	60000	2
6	11115	Dale	Veller	70000	2
7	11116	Ethan	Miller	80000	1
8	11117	Fanny	Niel	90000	2

5) Create a trigger that can update the num of projects whenever a new project is assigned to an employee.
   a. Test the trigger with the below insert

      INSERT INTO projectAssignment VALUES (11114, 1003, ‘quality assurance');
      INSERT INTO projectAssignment VALUES (11115, 1003,'test engineer');

SQL:

```
/*Creating a trigger to update the employee and ProjectAssignment Table for any inserting or updating activity */

CREATE TRIGGER projectAssignmentTrigger
ON projectAssignment
FOR INSERT, UPDATE
AS
IF @@ROWCOUNT >=1
BeGIN
UPDATE employee
SET NumOfProjects =Projectnum.NumOfProjects
FROM
(SELECT e.employeeID, count(pa.projectID) AS 'NumOfProjects'
FROM employee e
INNER JOIN projectAssignment pa ON e.employeeID = pa.employeeID
INNER JOIN project p ON p.projectID=pa.projectID
GROUP BY e.employeeID) AS Projectnum, inserted, projectAssignment

WHERE employee.employeeID= projectnum.employeeID
AND inserted.projectID= projectAssignment.projectID
END;

SELECT * FROM employee

INSERT INTO projectAssignment VALUES (11115, 1003,'test engineer');
INSERT INTO projectAssignment VALUES (11114, 1003, 'quality assurance');

SELECT * FROM employee
```

Prior Output:

	employeeID	employeeFName	employeeLName	employeeSalary	NumOfProjects
1	9999	Farees	Patel	80000	1
2	11111	James	Smith	30000	1
3	11112	Ada	Zack	40000	1
4	11113	Ben	Yale	50000	2
5	11114	Callen	Wales	60000	2
6	11115	Dale	Veller	70000	2
7	11116	Ethan	Miller	80000	1
8	11117	Fanny	Niel	90000	2

Final Output:

	employeeID	employeeFName	employeeLName	employeeSalary	NumOfProjects
1	9999	Farees	Patel	80000	1
2	11111	James	Smith	30000	1
3	11112	Ada	Zack	40000	1
4	11113	Ben	Yale	50000	2
5	11114	Callen	Wales	60000	3
6	11115	Dale	Veller	70000	3
7	11116	Ethan	Miller	80000	1
8	11117	Fanny	Niel	90000	2