Lab Homework 2: SQL Programming

# Learning Objectives

In this exercise, you will practice programming in Microsoft SQL Server:

* Use built in SQL Server functions
* Write stored procedures and user-defined functions
* Create views and table valued functions

# Setup

To complete this homework or follow along with the instructor’s class demos from your coursework, you must first complete the steps in this setup section. For this exercise we require the SQL Server instance in the Docker environment. For more details on the how’s and why’s of managing the database environments used in this course, consult Lab 1. Instructions for this assignment:

1. Open the PowerShell prompt on Windows or the Terminal on MacOS.
2. Type:  
   cd adv-db-labs  
   to change the working directory to the repository folder. If you are in the correct spot, your command prompt should have adv-db-labs in it, for instance: PS adv-db-labs>
3. Change into the mssql folder, type:  
   cd mssql  
   if you are in the correct folder, your command prompt should have mssql in it.
4. Bring up the MSSQL environment, type:  
   docker-compose up -d
5. Check to make sure the environment is running, type:  
   docker-compose ps

The state of the **mssql** should be **Up** on **port 1433**.

1. You are now ready to connect to the running instance. Open **SQL Server Management Studio** (on Windows) or **SQL Server Operations Studio** (if you’re on a Mac), a.k.a the SQL Client. Use the SQL Client to connect to SQL server with:
   1. Server name: **localhost**
   2. Authentication: **SQL Server Authentication**
   3. Login: **sa**
   4. Password: **SU2Orange!**
2. After you’ve connected, open a new query window by pressing CTRL+n.
3. You are ready to begin the lab!

# Exercises

Complete each of the following exercises. If you are unsure how to accomplish the task, please consult the coursework videos where there are explanations and demos.

1. Use built in SQL functions to write an SQL Select statement on **fudgemart\_products** which derives a **product\_category** column by extracting the last word in the product name. For example
   1. for a product named ‘Leather Jacket’ the product category would be ‘Jacket’
   2. for a product named ‘Straight Claw Hammer’ the category would be ‘Hammer’

Your select statement should include product id, product name, product category and product department.

1. Write a user defined function called **f\_total\_vendor\_sales** which calculates the sum of the wholesale price \* quantity of all products sold for that vendor. There should be one number associated with each vendor id, which is the input into the function. Demonstrate the function works by executing an SQL select statement over all vendors calling the function.
2. Write a stored procedure called **p\_write\_vendor** which when given a required vendor name, phone and optional website, will look up the vendor by name first. If the vendor exists, it will update the phone and website. If the vendor does not exist, it will add the info to the table. Write code to demonstrate the procedure works by executing the procedure twice so that it adds a new vendor and then updates that vendor’s information.
3. Create a view based on the logic you completed in question 1 or 2. Your SQL script should be programmed so that the entire script works every time, dropping the view if it exists, and then re-creating it.
4. Write a table valued function **f\_employee\_timesheets** which when provided an employee\_id will output the employee id, name, department, payroll date, hourly rate on the timesheet, hours worked, and gross pay (hourly rate times hours worked).

# Turning it in

Take your copy and paste each of the solutions to the exercises into the submission template file included with this assignment. Make sure your name and SU email are at the top and turn in your work through the course learning management system.

# Tear-Down

When you are finished with the homework you should stop the environment:

1. From the terminal window where you typed docker-compose up -d type in the following:  
   docker-compose stop