CSIS05I

Database Systems II

Lab (10)

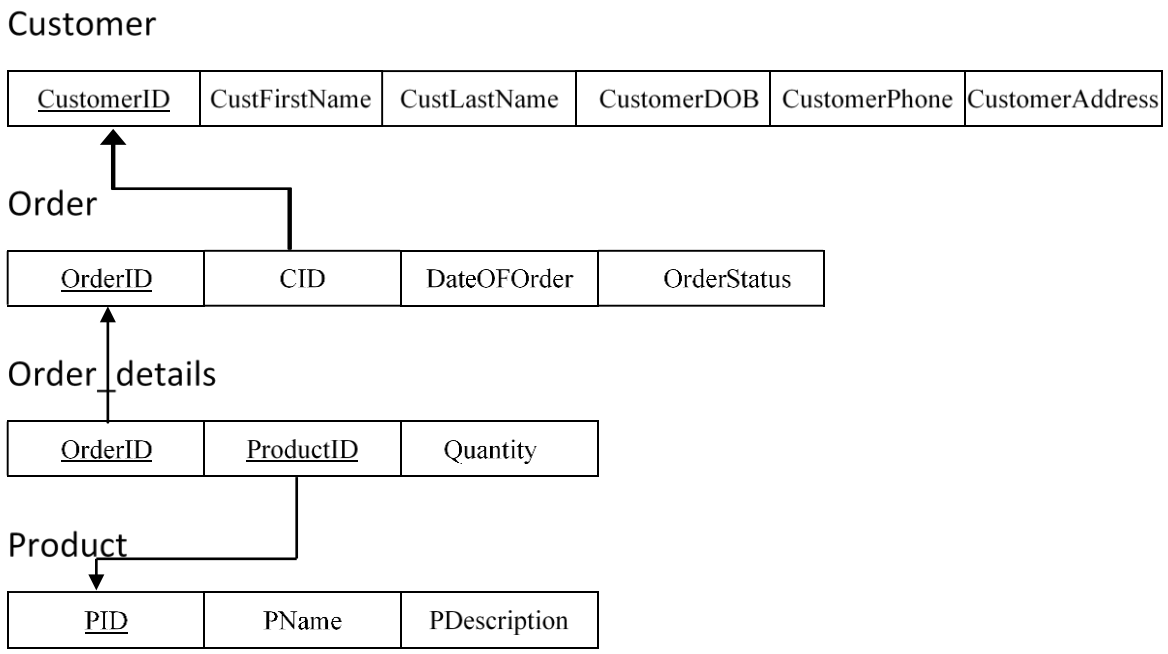
SQL Functions, Database Security and Jobs

Lab (10)

The Overview

In the previous lab, we mentioned the simple functions and the creation syntax for it. Through this lab, we will introduce the SQL functions in a more complex way. In addition to that, we will introduce new concepts which are: SQL jobs, and DCL of database.

Store Schema:



SQL Functions:

As we discussed in the previous lab, there are two function types, the scalar function and the table-valued function. The scalar function can take multiple inputs and returns only one output. While the table-valued functions return data of a table type (stores a result set for later processing).

Example 1:

Create a function that checks the number of orders made, if the number of orders exceeds 2, then the function outputs the following message 'target is reached'. Else, the function outputs 'target is not reached'. Display the output under 'OrdersTarget' alias.

```
CREATE or alter FUNCTION NumOfOrders()
RETURNS varchar(100)
AS
BEGIN
Declare @TotalNumOfOrders int
Declare @message varchar(100)

select @TotalNumOfOrders= Count(OrderID)
              FROM Orderr
              If (@TotalNumOfOrders< 2)
              select @message = 'target is not reached'
else
              select @message = 'target is reached'
              RETURN @message
END


Select dbo.NumOfOrders() As 'Orders Target';
```

Example 2:

Create a function that retrieve the Top 2 records in the Customer using a function.

```
CREATE or alter FUNCTION CustomerTopTwoRecords()
RETURNS table
AS
RETURN select top (2) * From Customer;

Select * from CustomerTopTwoRecords();
```

Note that: *the SQL SELECT TOP statement is used to retrieve records from one or more tables in a database and limit the number of records returned based on a fixed value or percentage.*

Example 3:

Create a function that displays today's date to the user.

```
create function gettodaysdate()
returns date
AS
BEGIN
declare @thedate date
select @thedate = getdate()
return @thedate
end

Select dbo.gettodaysdate() as 'Todays Date';
```

Example 4:

Create a function that retrieves a product's information based on the Product Id inserted by the user. If null value is passed to the function, all products and their information will be inserted in new table called 'products list'. Otherwise, the row of Product ID that user entered will be inserted only in the 'products list' table.

```
CREATE or alter FUNCTION product_info (@PRODUCT_ID Int)
RETURNS @ProductsList Table
(Product_Id int, Product_Dsp varchar(150), Product_Name varchar(100) )
AS
BEGIN
IF @PRODUCT_ID IS NULL
BEGIN

INSERT INTO @ProductsList (Product_Id, Product_Dsp, Product_Name)
SELECT PID, Pdescription, PName
FROM Product

END

ELSE
BEGIN

INSERT INTO @ProductsList (Product_Id, Product_Dsp, Product_Name)
SELECT PID, Pdescription, PName
FROM Product
WHERE PID = @PRODUCT_ID

END
RETURN
END
GO

Select * from product_info(NULL);
```

Data control language (DCL)

- **Data control language (DCL)** is a subset of the Structured Query Language. Database administrators use DCL to configure security access to relational databases. It complements the data definition language (DDL), which adds and deletes database objects, and the data manipulation language (DML), which retrieves, inserts, and modifies the contents of a database.

- **DCL** consists of statements / commands such as **GRANT and REVOKE** that control security and concurrent access to table data. They mainly deal with the rights, permissions, and other controls of the database system.

- In other words, **GRANT and REVOKE** are types of DCL commands that are used to assign permission to the users to perform a different task. The GRANT command is used for permitting the users while the REVOKE command is used for removing the authorization.

1. Grant

SQL Grant is SQL Data Control Language command and is used to enforce security in a multiuser database environment. SQL Grant is used to provide permissions to the users on the database objects.

General syntax:

Grant privilegeName on objectName To {userName/Public}
[with Grant Option]

Note that:

- ✓ *PrivilegeName is the access right or permission that is granted to the user like SELECT, INSERT, UPDATE, etc.*
- ✓ *ObjectName is the name of a database object like Table, View, Stored Procedure or Function.*
- ✓ *UserName is the name of the user to whom the permission is granted.*
- ✓ *"With Grant Option" allows the user to grant the permission to the other user and it is optional.*

Example 5:

Grant select privilege to the customer table, where the user name in your local host is Ahmed.

```
grant select on Customer to [ahmed];
```

Example 6:

Grant select privilege to the customer table.

```
grant select on Customer to public;
```

Note that: *the keyword PUBLIC is used to specify all users. When PUBLIC is specified, the privileges or roles affect all current and future users.*

Example 7:

Grant select privilege to users on customer table knowing that they can propagate that privilege.

```
grant select on Customer to public with grant option;
```

Example 8:

Grant insert privilege on product table, knowing that it can be propagated.

```
GRANT INSERT ON Product TO public with GRANT OPTION;
```

Note that: *you should use the WITH GRANT option carefully because for example if you GRANT specific privilege on a specific table to user1 using the WITH GRANT option, then user1 can GRANT this privilege on this table to another user like user2. Later, if you REVOKE this privilege on the table from user1, still user2 will have the privilege on this table.*

Example 9:

Create a view that shows all customers that live in 'New Cairo', then make a privilege on it to all the users who can propagate it later.

```
create view [Cairo Customer]
As
select Custfirstname, custlastname
from Customer
where CustomerAddress = 'New Cairo';

Select * from [Cairo Customer];

GRANT SELECT ON [Cairo Customer] TO public with grant option;
```

Example 10:

Grant multiple privileges to the users on product table.

```
GRANT SELECT, INSERT, DELETE, UPDATE ON Product TO public;
```

OR

```
GRANT All ON Product TO public;
```

Granting Execute to a function or a procedure:

General syntax:

```
Grant Execute on FunctionName/procedureName to User;
```

Example 11:

Create a function that gets maximum quantity of orders, then grant execute to public.

```
CREATE FUNCTION Get_Maximum_Quantity()
RETURNS int
AS
BEGIN
Declare @Max_Quantity int
select @Max_Quantity = MAX (Quantity)
From Order_Details
RETURN @Max_Quantity
END

grant execute on dbo.Get_Maximum_Quantity to public;
```

2. Revoke

Revoke command withdraw user privileges on database objects if any granted. It does operations opposite to the grant command. This DCL command removes permissions if any granted to the users on database objects.

General syntax:

```
REVOKE privilegeName ON ObjectName FROM User;
```

Example 12:

Revoke on table order.

```
REVOKE SELECT ON Orderr FROM public;
```

Example 13:

Revoke Execute on Get_Maximum_Quantity function.

```
REVOKE EXECUTE ON dbo.Get_Maximum_Quantity FROM public;
```

Example 14:

Revoke Select, Insert, Delete and update on table Order_Details from Public.

```
REVOKE SELECT, INSERT, DELETE, UPDATE ON Order_Details FROM Public;
```

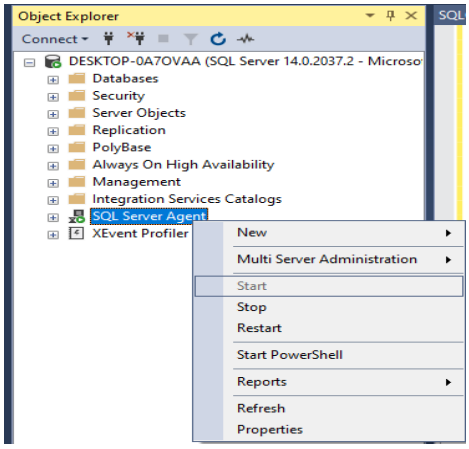
SQL Jobs:

Jobs: A SQL Agent job is the specified action or series of actions/steps that the SQL server agent will execute.

Schedulers: The SQL Agent schedule specifies when a job will run. Multiple jobs can have the same schedule.

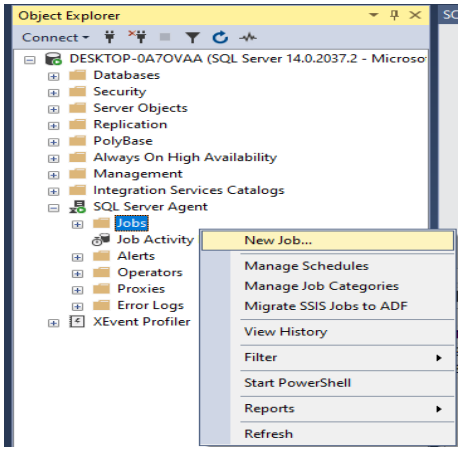
Creating and scheduling SQL Server jobs:

1- Start the SQL Server Agent (Right-Click on the SQL Server Agent menu in Object Explorer then click START).

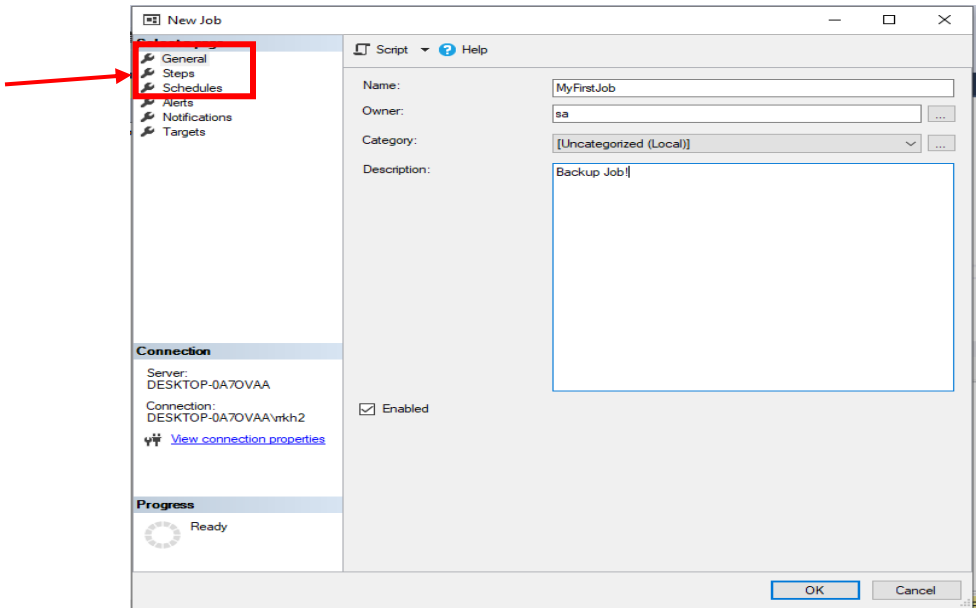


2- Expand the "SQL Server Agent" Then you will find “Jobs Folder”.

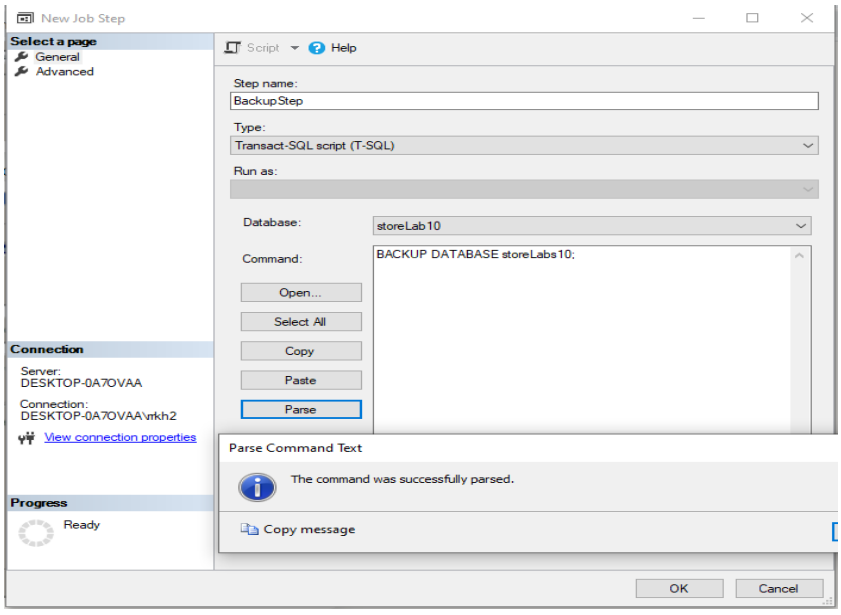
3- Right-Click on "Jobs" and Click "New Job..."



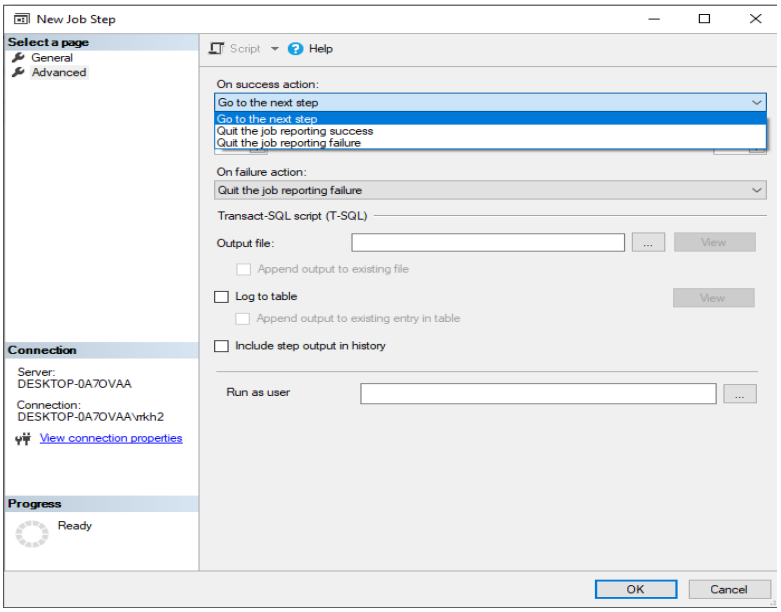
4- In the new job creation screen, give the name of the Job, select Owner and add a description of the created job.



5- Navigate to the "Steps" page (on the left side of the window) and click "New", provide the Step Name and choose the Type as “Transact-SQL script (T-SQL)". Then Choose your Database Name, then write the SQL script that the job will run in the “Command” window, then click “Parse” button to make sure that your SQL code will be successfully executed without any syntax error.

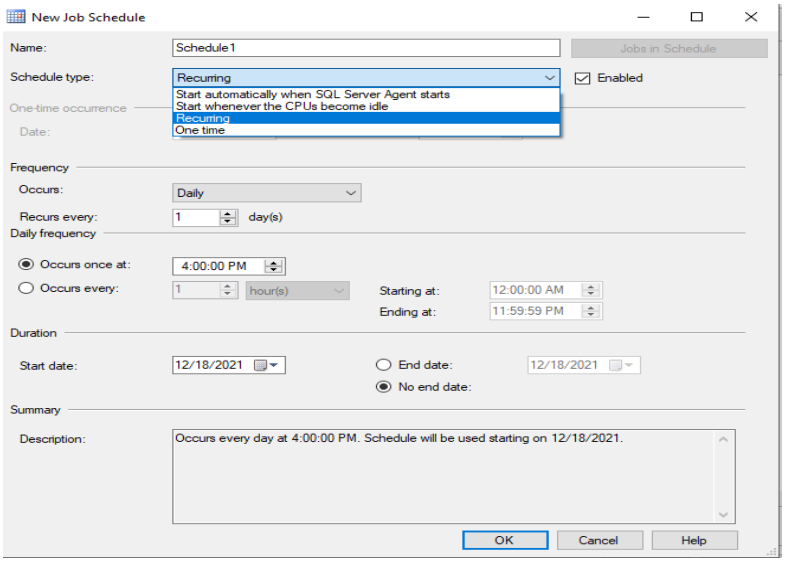


6- In the "Advanced" page of steps window, you can set the advance job step to choose the required actions on success or failure cases of the job, then click “Ok”.



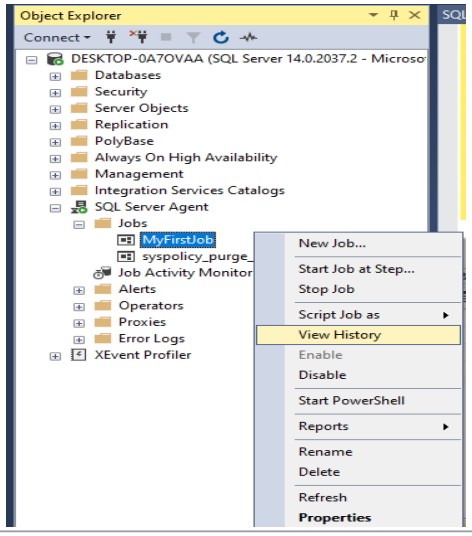
7- Now you can set the schedule to this job by navigating to the "Schedules" page or you may choose to run this job manually.

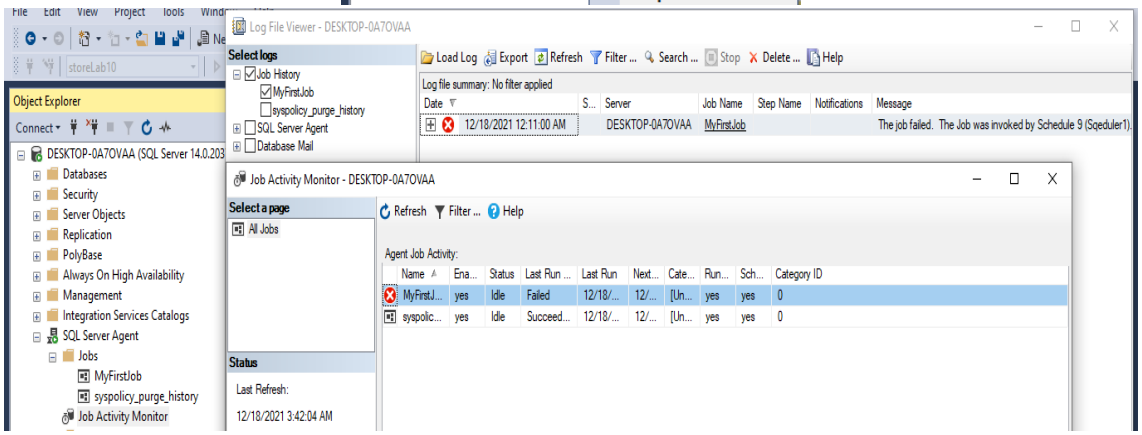
8- Provide the Schedule Name then choose the Schedule Type, Frequency, Duration, Start and End Date and Time.



9- Once you click "OK" and come out of the wizard, your new job will appear in the Jobs list and based on the schedule set, the job will automatically execute provided that the job is enabled and the SQL Server agent is in running state.

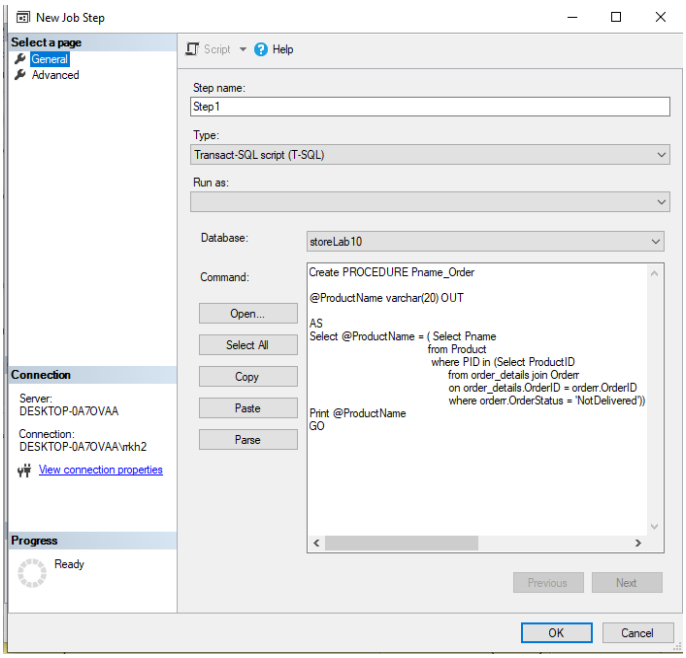
10- Finally, you can view the log activity or the history of the job; Right-Click on the Job Name in the Jobs list then choose “View History” or directly Double-Click on the “job Activity Monitor” as shown below.

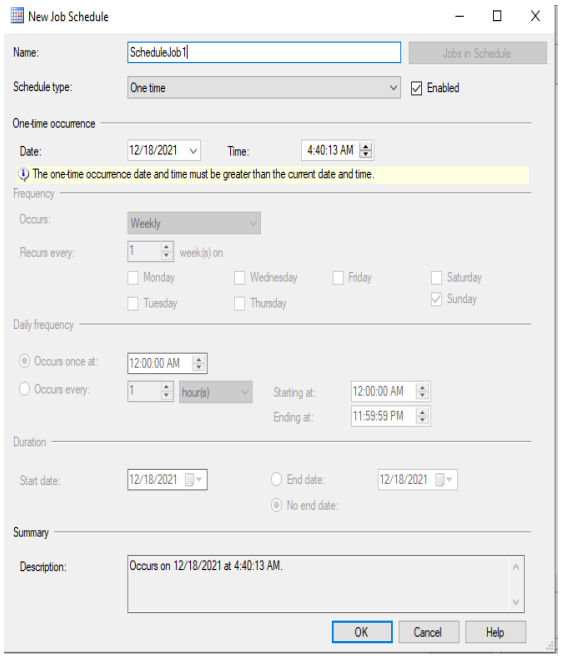




Example 15:

Create a Job that schedule a stored procedure that prints the product name where the order status is 'Not Delivered' using nested queries. Schedule this job to run just once.





SQL Command:

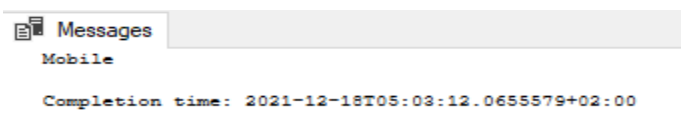
```
Create PROCEDURE Pname_Order

@ProductName varchar(20) OUT

AS
Select @ProductName = ( Select Pname
                        from Product
                        where PID in (Select ProductID
                                      from order_details join Orderr
                                      on order_details.OrderID = orderr.OrderID
                                      where orderr.OrderStatus = 'NotDelivered'));
Print @ProductName
GO
```

To Test the Stored Procedure created inside the job:

```
Execute Pname_Order OUT
```



Exercise:

1. Create a function named ‘average’ that calculates the average of two numbers taken from the user as an input. Display the output in float, under the alias of Average Point.
2. Grant multiple Privileges to the users on the Product table.
3. Grant a privilege to public on the ‘average’ function.
4. Revoke Select, Insert, Delete and update on table Customer from Public.
5. Create a Job that schedule a function that retrieves the minimum quantity of a product, schedule this job to run recurrently on daily basis.