CONTENTS

INTRODUCTION

The tasks for this session are to:

- Explore views, triggers and stored procedures for the set scenario of a stationary supply application.
- Create views, triggers and stored procedures for the three given scenarios.
- Explore transactions.
- Apply the use of transactions to the creation of a stored procedure
- Create transactions for the three given scenarios.

For the purposes of these exercises you are to use Azure Data Studio as the tool for viewing your remote database instance.

VIEWS, TRIGGERS AND STORED PROCEDURES

Using your server database and the tables (orders, products, etc) you created last week, your tasks for this week are to create a number of views, triggers and stored procedures. Start with downloading *Unit 2.sql and running through the commands. Make sure you understand what they are doing.

Run the sql – having edited it to apply to your database and following that edit the sql commands to make amendments. You should create views for minimum stock order and invoices, to create triggers for auditing changes to the prices of products and to create stored procedures for sales per customer, to create an invoice for a customer and to create a picking list for the warehouse for a particular date.

It is entirely possible you will not be able to complete all the activities here. Ensure you complete the rest in your self-study time.

Activity 1

Download the sql file for this Unit. Using Azure Data Studio, and the connection to your Microsoft SQL database, first edit (to use your database) and then run the given SQL script so that it creates a view, trigger and stored procedure in your database. (NOTE: You will need to have completed activity 1 from last week for this to work.) Using the side bar in Azure Studio look for where the Views are stored. Are they what you would expect? Annotate the sql file so that you label important concepts that are demonstrated there. Use the checklist at the side to help you consider what is important to note.

Modify the .sql file so that you alter the view to add in the customer Details. Think about how you alter existing objects. Think about how the customer details might be linked (this was part of the exercise you were to finish off last week). Think about what you might display in your view.

- ✓ What does “sum” do?
- ✓ How would you use it?
- ✓ What else do you need to have for it to work?
- ✓ How are the tables joined?

Activity 2

Create new views, triggers and stored procedures to account for the following:
* A view to see the Sales per customer
* A view to see the number of orders per month.

> Here you will need to consider the aggregate function of Count. Think about what it is you need to count.
> Also look for how to extract both the YEAR and the MONTH from a date.
>
> Remember to use GROUP BY

* A trigger for highlighting the minimum quantity of stock ready for reordering. Think about the data you would need for this. Consider either adding a new column to your products table, or even consider having a stock order table. You will need to know what quantity to use for your minimum order – for the purposes of this exercise you may choose your own value that seems sensible. An example might be a minimum quantity of 50 for pens. As an extension you might consider creating a view for your re-ordering details.
* A trigger that saves price change data into an audit table. For each time a price is changed on a product, the old price and date change are saved in a separate table.

> When thinking about an audit table you need to record key aspects of the transaction. You need to know the date and time it was changed, you need to know what it was changed from. The what it was changed to will be recorded in the main table. You will need to create an audit table that not only replicates the data in the table you are concerned with, but also adds the columns for this additional detail to be stored.
>
> You will need to reference the temporary "inserted" table that the trigger holds for incoming values.

* A stored procedure that brings back the invoice information for a specific order.

> The core information in an invoice is that the customer knows the details of the order it is referring to. This will be the order number, the order details relating to that number and a final total. The order details will show the name of the product.

* A stored procedure that brings the amount of sales per customer. You may need to remind yourself here how to do subqueries within the select statement.
* A stored procedure that brings the items to be packed and sent on a particular date. You will need to know the customer name – but we shall assume that an address is somewhere else on the system and we shall ignore that detail for the moment. All you need to have an output for is the customer name, the order number and the quantities of what products go into that order. Look up how you use SQL to gather details for given dates.

Activity 3

For each of the scenarios below identify the views to be created. You are to write a collection of create view statements to add to the three separate applications. Remember to use the name of the view to identify which scenario it belongs to.

Scenario 1 : Medical trial. The basic information was given to in Workbook 1. However, when thinking about the reporting side of the application the end-users want to see the following:

- A data summary grouped by patient. This would show the patient ID, the total number of falls and the total number of injuries.
- A summary per injury type. This would include the total number of patients per injury type.
- A summary grouped per day. This would have the total number of falls, total number of injuries for each day recorded.

Scenario 2 : Placements Information. The basic information was given to you last week. Views will help with the following reporting:

- A summary view grouped by student with the placement provider and the contact details.
- A list view of students showing start and end dates in order of earliest finish. NOTE: Dealing with dates in SQL is challenging. Please refer to the Useful links section for further guidance.

Scenario 3 : Hostel booking application. The basic information was given last week. Views to provide the following reporting are required:

- A list of rooms which have unoccupied beds.
- A list of volunteers that will arrive in the next week.
- A list of booked volunteers, their room allocations for today

TRANSACTIONS

Using your database server and the tables (orders, products etc) you created previously, your tasks for this week are to incorporate transactions into stored procedures. Start with downloading *Unit 3.sql and running through the commands. Make sure you understand what they are doing.

Run the sql – having edited it to apply to your database and following that- edit the sql commands to make amendments. You will be creating a stored procedure that will place orders for customers so long as there is enough stock to fulfil the order. Within that stored procedure you will use a transaction so that if there is not enough stock, you can roll back the entries.

It is entirely possible you will not be able to complete all the activities here. Ensure you complete the rest in your self-study time.

Activity 1

Download the sql file for this Unit. Use Azure Data Studio as previously instructed to view the file. Annotate the .sql file so that you label the important concepts you need to remember. Work out what each section is doing and why.

Activity 2

This activity steps through adding a new order into the database. Use your existing socem1 database and tables from previous activities and connect using Azure Data Studio.

- Create a new stored procedure to add an order for a customer.
- The stored procedure should use transactions to ensure that all the steps are completed – or rolled back if an error occurs.
- If the Orders table does not have an Identity type for it's primary key, you will need to find the last figure for the OrdersID. Look up how to use Max.

Modify the .sql file so that you alter the view to add in the customer Details. Think about how you alter existing objects. Think about how the customer details might be linked (this was part of the exercise you were to finish off last week). Think about what you might display in your view.

- To successfully complete this task you will need to learn how to use variables in SQL Server (see insert notes)

> Variables in SQL are like other variables in the coding languages you have come across already. They have a special notation – using the @ symbol and first you must declare them with their data type.
>
> DECLARE @Error NVARCHAR(Max);
>
> Using variables in SQL can be done either by setting them directly
> SET @Error = 'An error has occurred';
>
> OR by using a SELECT statement to fill it from a table
> DECLARE @Price money;
> SET @Price = Price FROM Products WHERE ProductID = 2;

- As you saw in the example SQL, use Try and Catch blocks.
-

Activity 3

For each of the scenarios below identify where transactions would be used and write out the pseudocode for them.

Scenario 1 : Medical trial. The basic information was given to you last week. However, when thinking about the reporting side of the application the end-users want to see the following:

- A data summary grouped by patient. This would show the patient ID, the total number of falls and the total number of injuries.
- A summary per injury type. This would include the total number of patients per injury type.
- A summary grouped per day. This would have the total number of falls, total number of injuries for each day recorded.

Scenario 2 : Placements Information. The basic information was given to you last week. Views will help with the following reporting:
• A summary view grouped by student with the placement provider and the contact details.
• A list view of students showing start and end dates in order of earliest finish. NOTE: Dealing with dates in SQL is challenging. Please refer to the Useful links section for further guidance.

Scenario 3 : Hostel booking application. The basic information was given last week. Views to provide the following reporting are required:
• A list of rooms which have unoccupied beds.
• A list of volunteers that will arrive in the next week.
• A list of booked volunteers, their room allocations for today

USEFUL LINKS

Conventions used in SQL
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql?view=sql-server-2017
Dates in SQL Server
https://docs.microsoft.com/en-us/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql?view=sql-server-2017
More Dates
https://www.w3schools.com/sql/sql_dates.asp
Using the YEAR command
https://www.w3schools.com/sql/func_sqlserver_year.asp
Using the MONTH command
https://www.w3schools.com/sql/func_sqlserver_month.asp
Tips for dealing with dates
https://www.mssqltips.com/sqlservertip/5206/sql-server-datetime-best-practices/
Subqueries in SQL
http://www.sqltutorial.org/sql-subquery/
SQL Identity – how to retrieve the value of the last inserted row
https://docs.microsoft.com/en-us/sql/t-sql/functions/identity-transact-sql?view=sql-server-2017