**INFO 6210 Data Management and Database Design**

**Final Project 6**

**Database for Handling Online and in-store Sales at Liquor Shop**

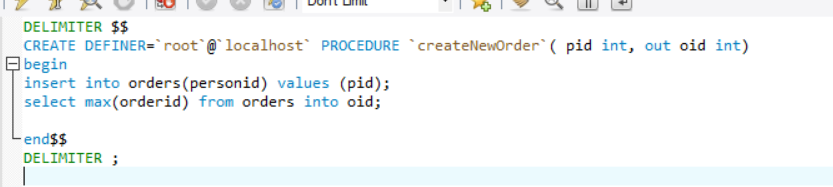
A liquor shop deals with a lot of products including different types of spirits and non-alcoholic commodities. A topic like this gives me the chance to explore lots of analytics. I am looking at helping customers buy liquor online as well and instore and I have written different stored procedures to place an order via the respective procurement medium.

To purchase alcohol in USA an individual must be 21 and above. Well, when this needs to be validated during an in-store sale, it shall be done physically. However, during the online procurement my stored procedure warns the Database Operator that this order should not be placed.

For me to place an order the DB operator must call three procedures. They are as follows;

**1) createNewOrder(PersonID, @OrderID);**

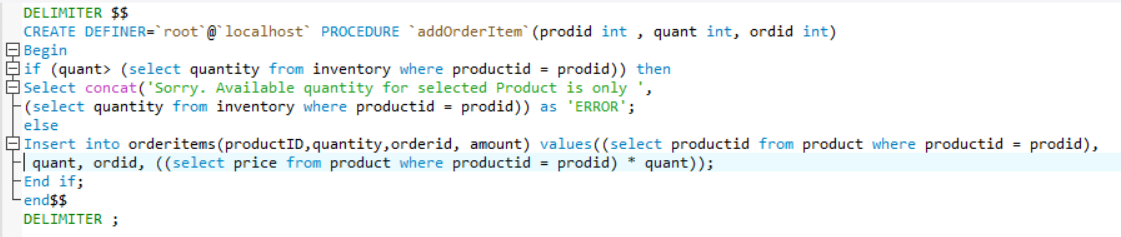
This function creates a new entry in the orders table and gives back the order id of this newly created table. Procedure Definition is as follows.

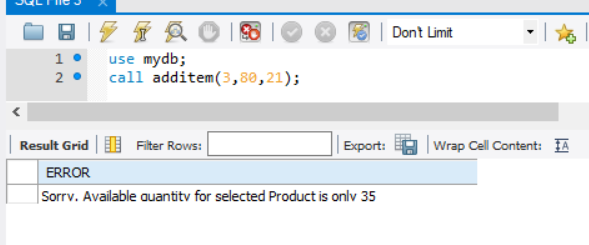


**2) additem(ProductID, Quantity needed, @OrderID);**

This procedure adds into the orderitem table with the orderID received from the previous procedure. Using this function:

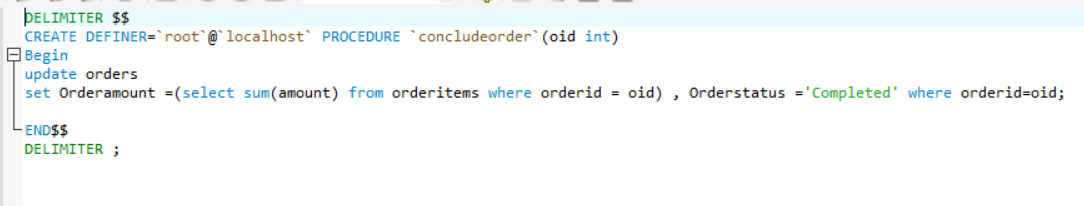
1. We validate if inventory has available quantity
2. We can how many ever products we want to a single order.





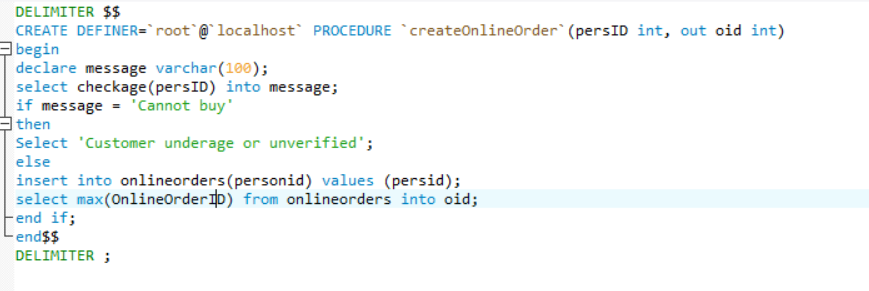
**3) conclude order(@OrderID);**

This procedure traverses through the orderItems table and calculates the sum of all items that belong to a certain Order.

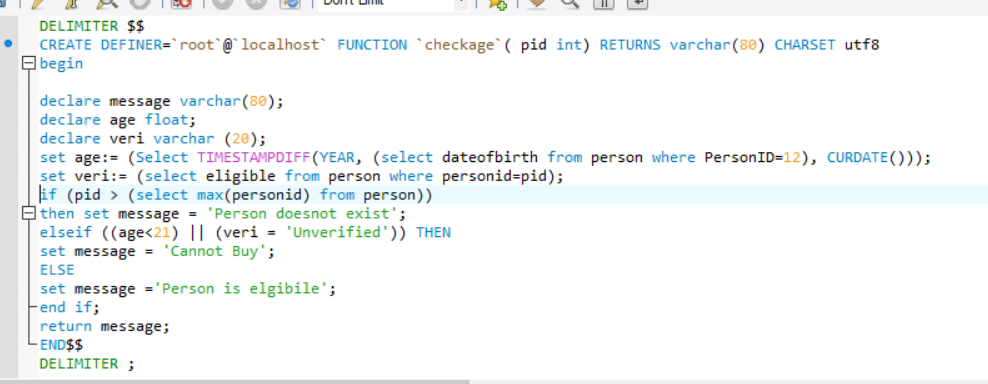


**4) createNewOnlineOrde**

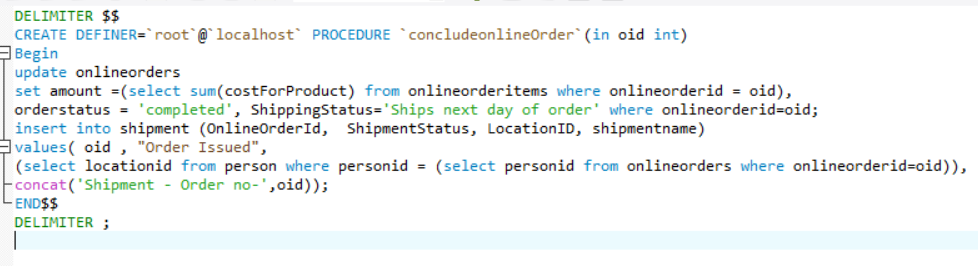
This procedure first checks the age of the customer and then alerts the DBO that he/she is underage or unverified. As we have online system lets imagine a scenario where I sign up for the service, send them a picture of my ID proving that I’m 21+ and then my status is verified. This age is verified using the checkage() UDF.



**5) checkage()**



**Similarly, I have included other functions for online order for adding orderitems. Kindly find the concludeOnlineOrder() below.**



**8) take\_backup()**

This procedure allows me to take a backup of all the data in the current DB.

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `take\_Backup`()

block1: BEGIN

DECLARE tab\_name char(50);

DECLARE q varchar(1500);

DECLARE done INTEGER DEFAULT 0;

DECLARE cursorBackupTable CURSOR FOR SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE table\_schema='mydb' and table\_type='BASE TABLE';

DROP DATABASE IF EXISTS mydb\_backup;

CREATE DATABASE mydb\_backup;

open cursorBackupTable;

block2: begin

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

cur\_loop:Loop

FETCH cursorBackupTable into tab\_name;

IF done= 1 THEN LEAVE cur\_loop;

END IF;

SET @q=CONCAT('DROP TABLE IF EXISTS mydb\_backup.',tab\_name);

PREPARE stmt FROM @q;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET @q=CONCAT('CREATE TABLE mydb\_backup.',tab\_name,' AS SELECT \* FROM mydb.',tab\_name,' WHERE 1=1');

PREPARE stmt FROM @q;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END LOOP cur\_loop;

END block2;

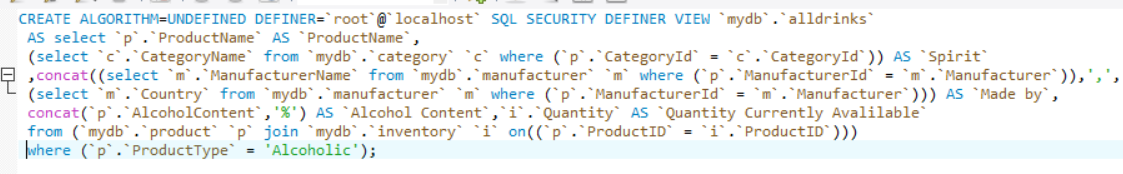
close cursorBackupTable;

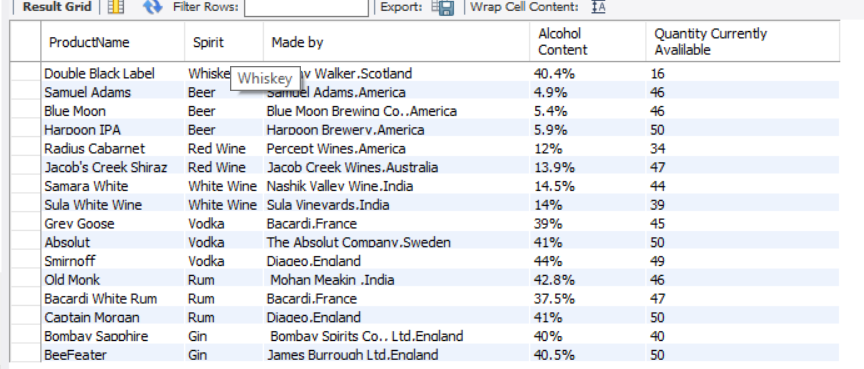
END block1$$

DELIMITER ;

**Now that we are done with Stored procedures kindly find the views below:**

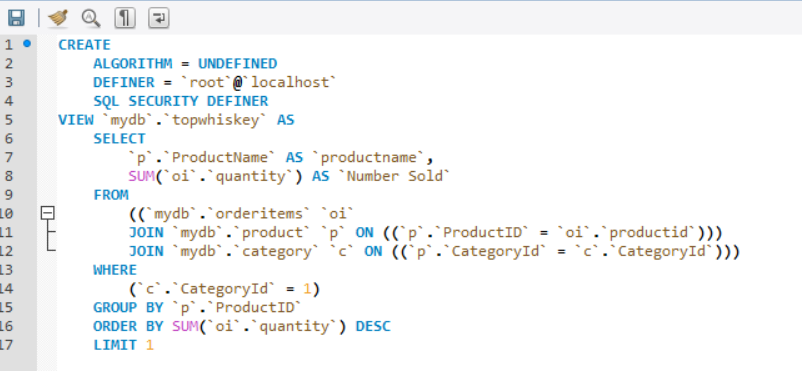
1) allDrinks;

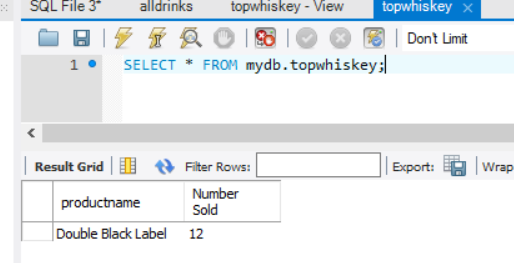




2) TopWhiskey

This Whiskey gives me the name of that whiskey that is the most sold joining three tables.





**We can now take a look at all the triggers created within the solution:**

1) sale\_ANSI: This trigger reduces the quantity once a sale is made. A similar trigger is written when an online sale is made.

2)Updateofavailibilty:

Once an order is canceled the quantity is updated in the inventory. A similar trigger is written when an online sale is made