

**JSC “Kazakh British Technical University”**

**School of Mathematic and Cybernetics**

    Analysis of Data Bases

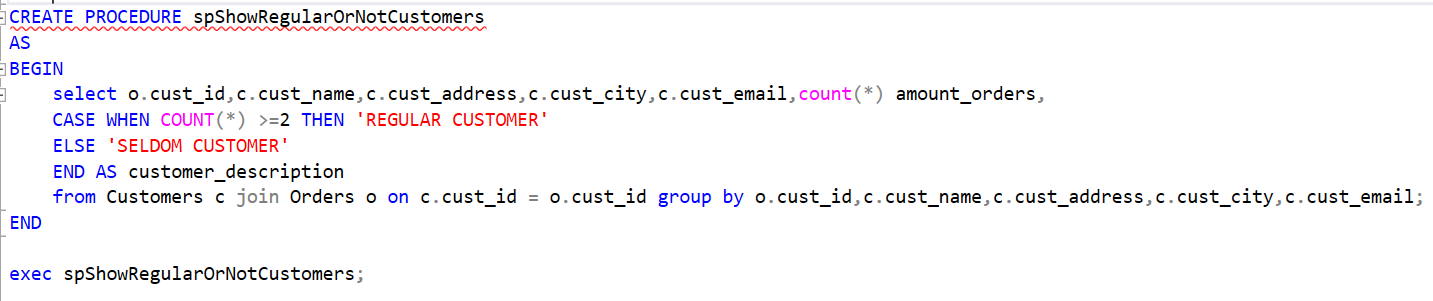
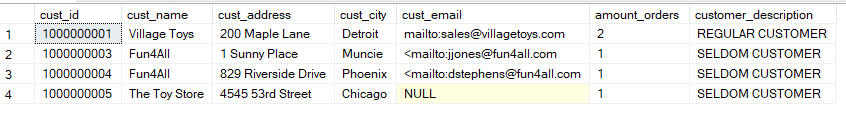
**Laboratory Work #10**

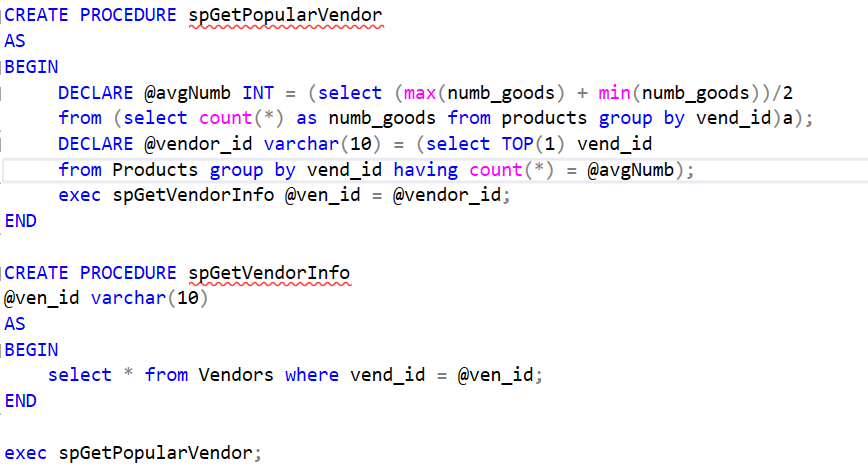
**Prepared by: Maratuly Temirbolat**

**Almaty 2021**

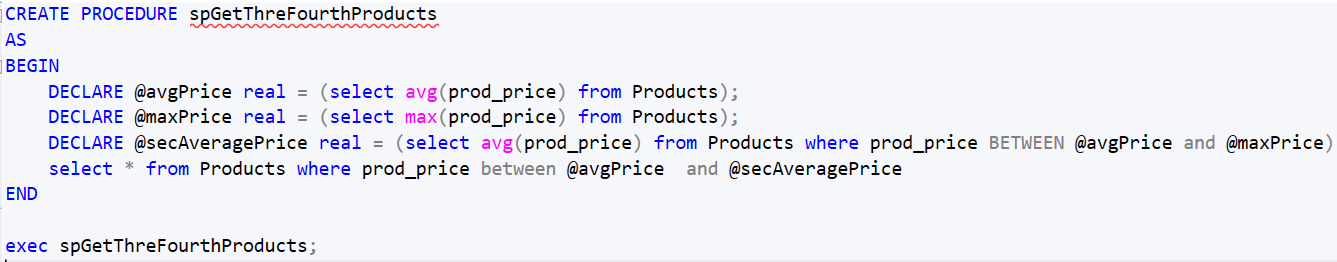
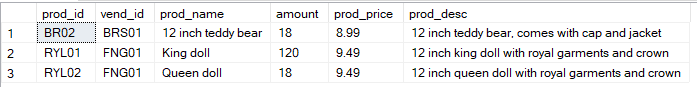
**The task of the Laboratory Work is to create queries using 5 Stored Procedures, 7 Triggers, 8 Functions.**

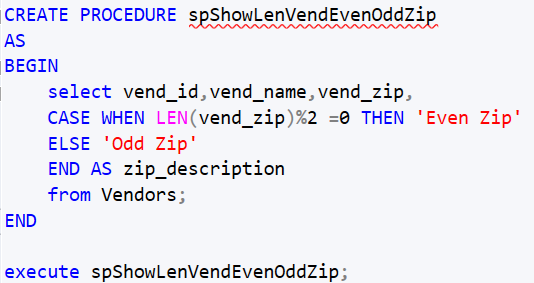
**Queries with 5 Procedures:**

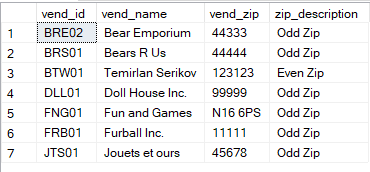
1. Create a stored procedure that shows the customers and regularities of their visits to the market. The customer is said to be ‘REGULAR CUSTOMER’ if he/she bought at least 2 things, otherwise ‘SELDOM CUSTOMER’. Show all the info with this description.
2. Investigate a stored procedure that illustrates the most popular seller among all of the them. Use two procedures if it is necessary then show all the info about this vendor.



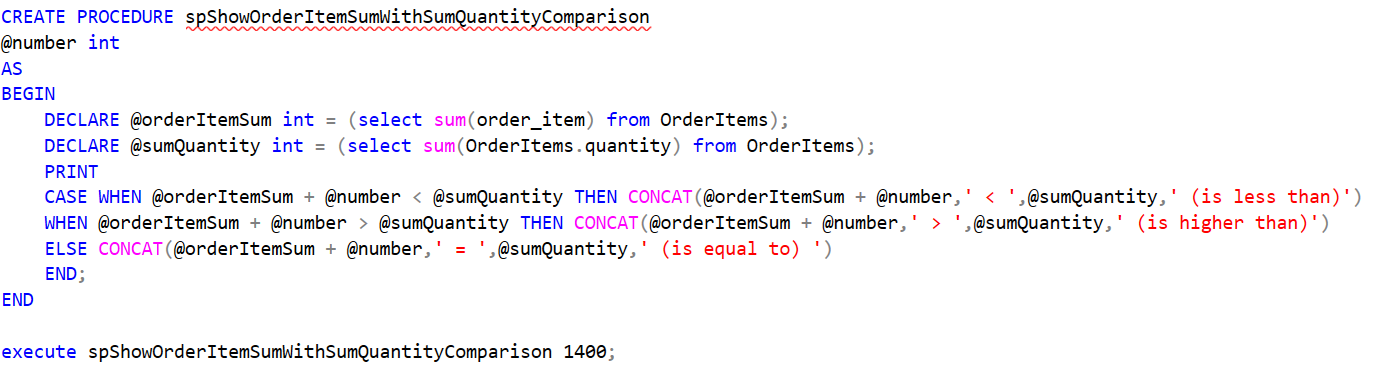


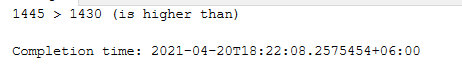
1. Provide a stored procedure that shows all the products whoes price is located between average price and second higher average price of the products. The second higher average price is located exactly in the middle of the mean value and max value price of the products.
2. Create a stored procedure that shows all the id, name, zip as well as description of all the vendors. The description must indicate whether or not the length of the zip ODD or Even.



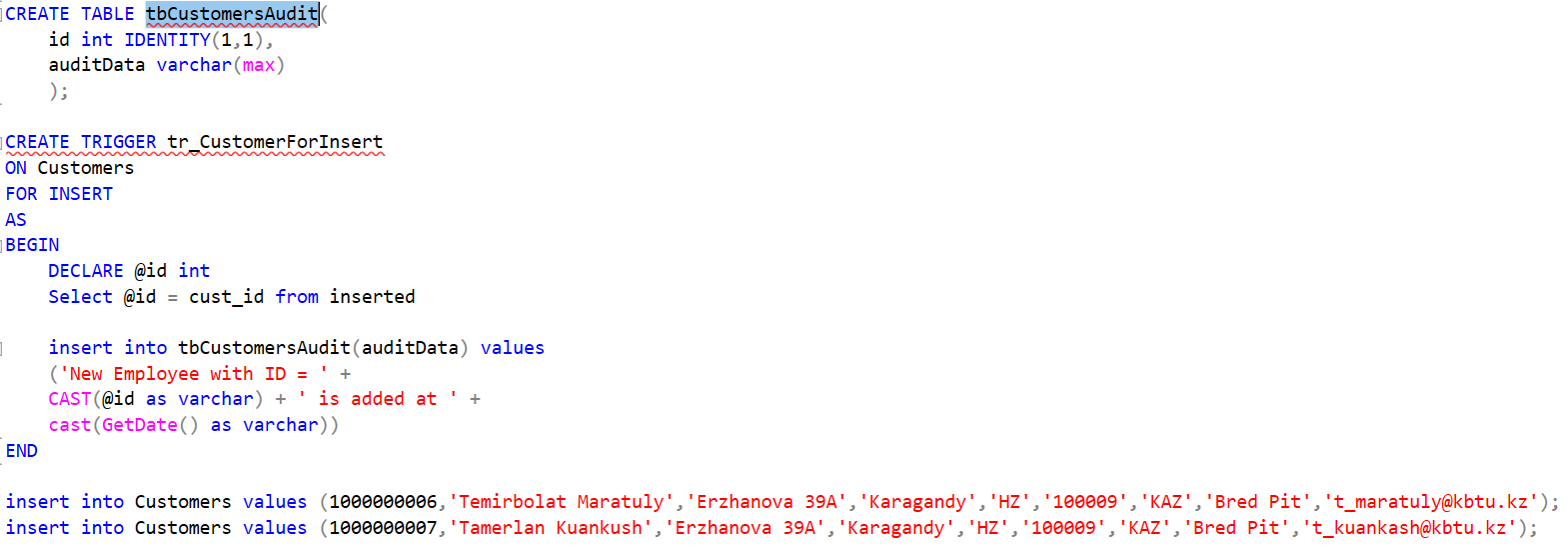


1. Create a stored procedure which takes any integer number and prints the sum of the whole items + given number less,equal or higher than the total sum of the whole quantities of the items. See solution below:



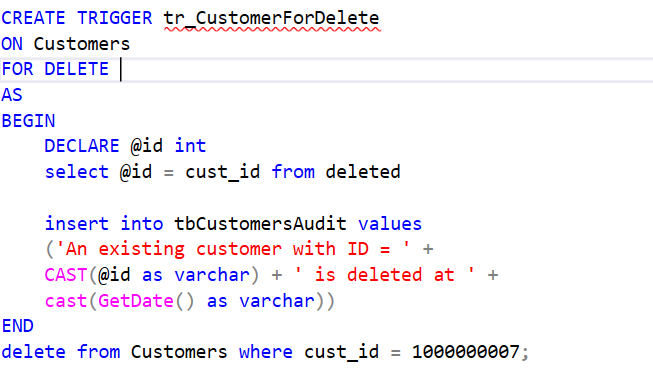


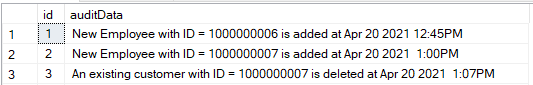
**7 queries for Triggers (Very Interesting)** 😊

1. Create a new table and call it tbCustomersAudit with 2 columns: id int with identity starting from 1 as well having step 1 and auditData varchar(max). As you finished, create a trigger that would add the information about Customer who was inserted into the Customers table with his/her id and time when he was added.

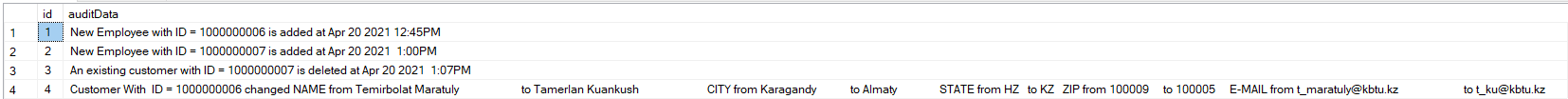


1. Do about the same procedure as in 6-th exercise. However, you have to do it with DELETE Part and write the information down about activities into recently created table. Type “An existing customer with ID = … is deleted at … (TODAYS DATE)”.



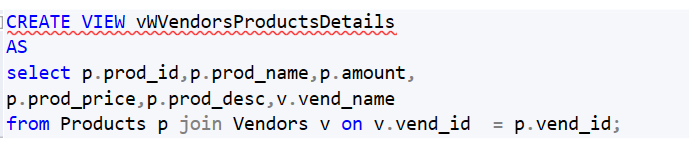


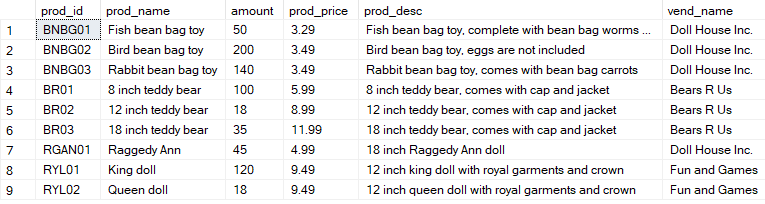
1. Again. Create a Trigger that is responsible for indicating the information that customers changed about themselves and add this notification into the table that was create in 6-th exercise. The information would be too long because of the number of given attributes in the CUSTOMERS table.

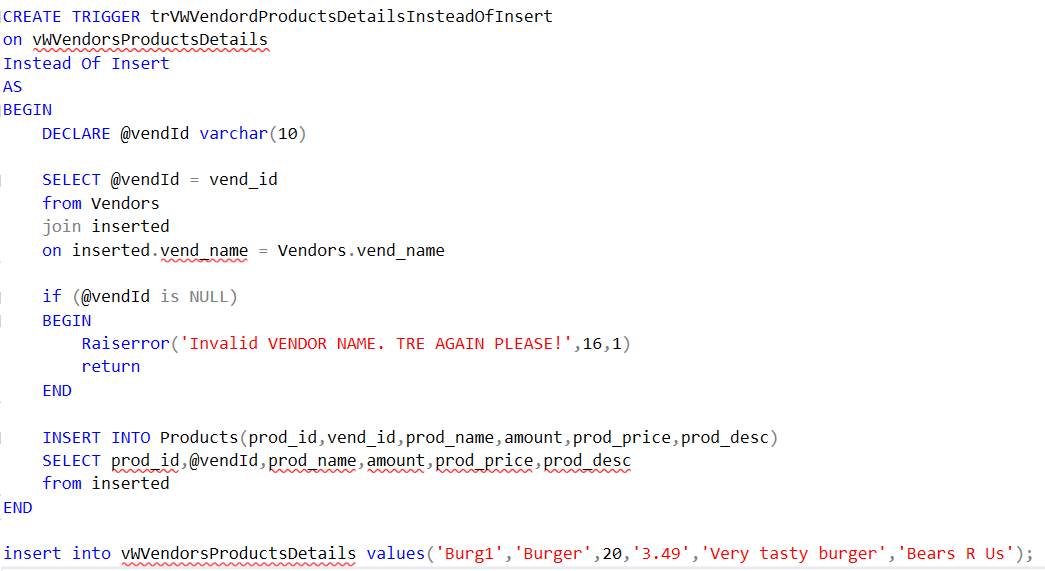
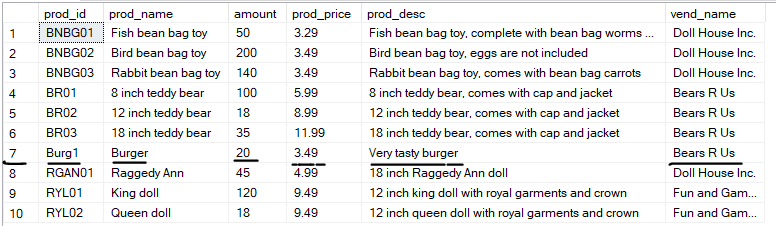


1. Wow, the previous task was quite unexpected but what if we additionally create a VIEW (virtual table) which contains the join(combination) of products and vendors. We need to use this combination as one table make some manipulations for it. If we write insert into nameOfTheView values we would obtain a mistake because it consists of the multiple tables. So, for this purpose we use TRIGGER. However, we create it with INSTEAD of Insert to imitate the insert procedure.

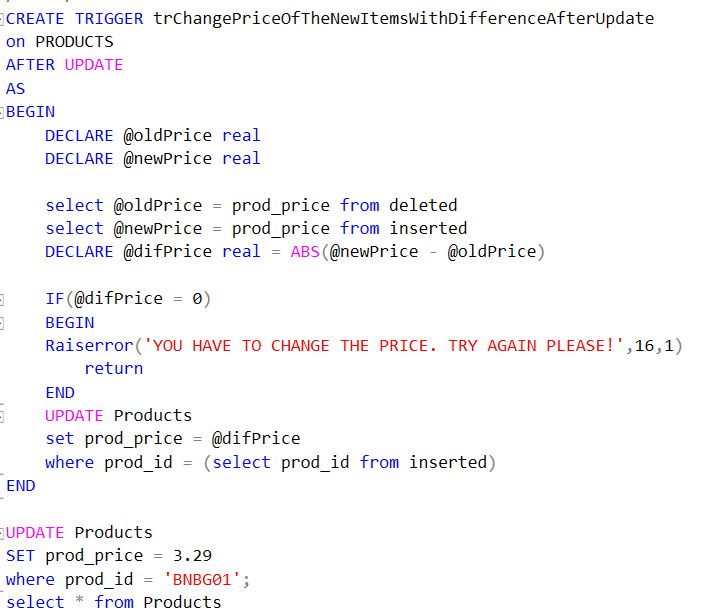
CREATED VIEW:

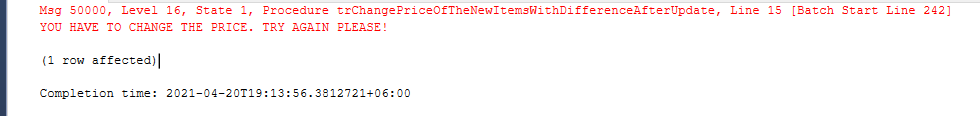




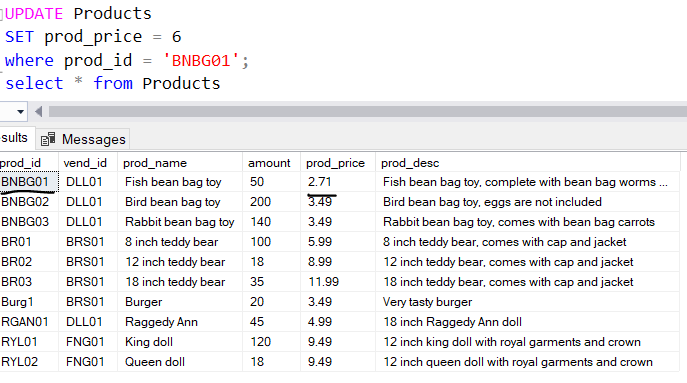
Create a TRIGGER FOR INSERT:

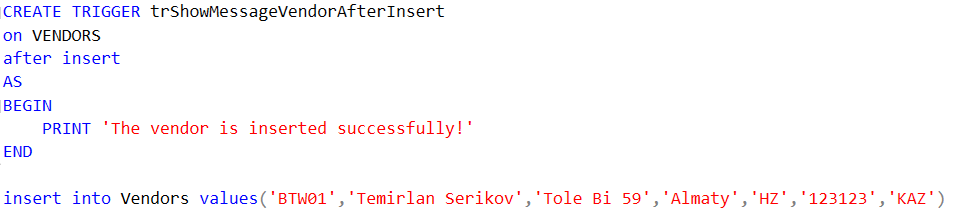
1. Create a Trigger that works if we update the PRICE of the PRODUCTS otherwise mistake appears. We need to assign the difference between new and old prices as new price for those products that were influenced by changes.

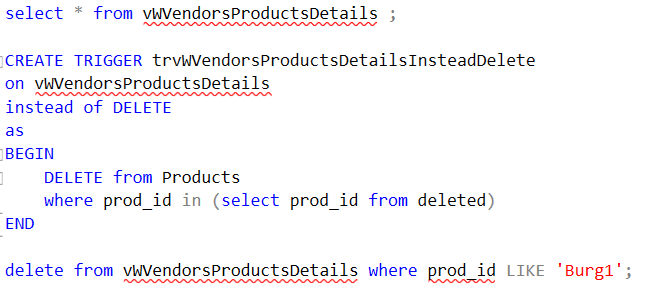


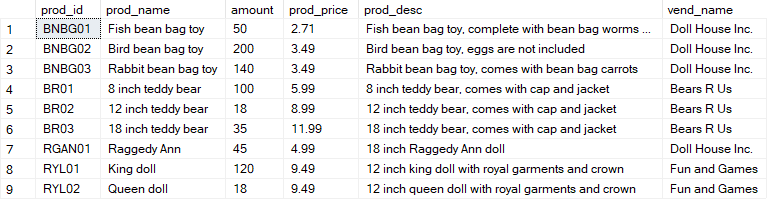
If we apply the same new price :

Otherwise:

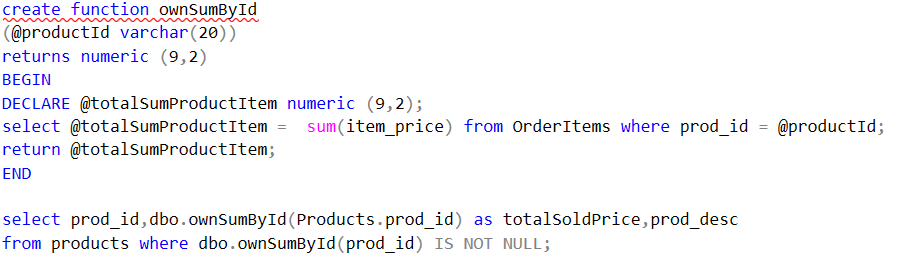


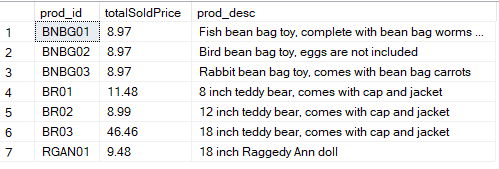
1. Oh, since there are no efforts and imagination to produce a cool query let’s create something basic. Write a Trigger that would show ‘The vendor is inserted successfully!’ if there was inserted a new Seller into the table.
2. Well, you need to use the created previously VIEW and this time investigate a TRIGGER that would imitate the working principle of DELETE for the VIEW since mistakes appears if we try to do it. We will delete recently inserted burger product.

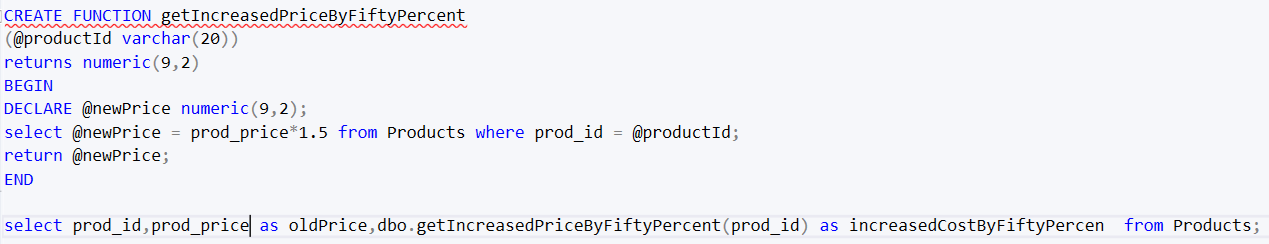
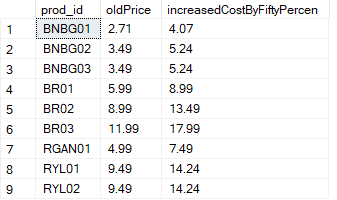
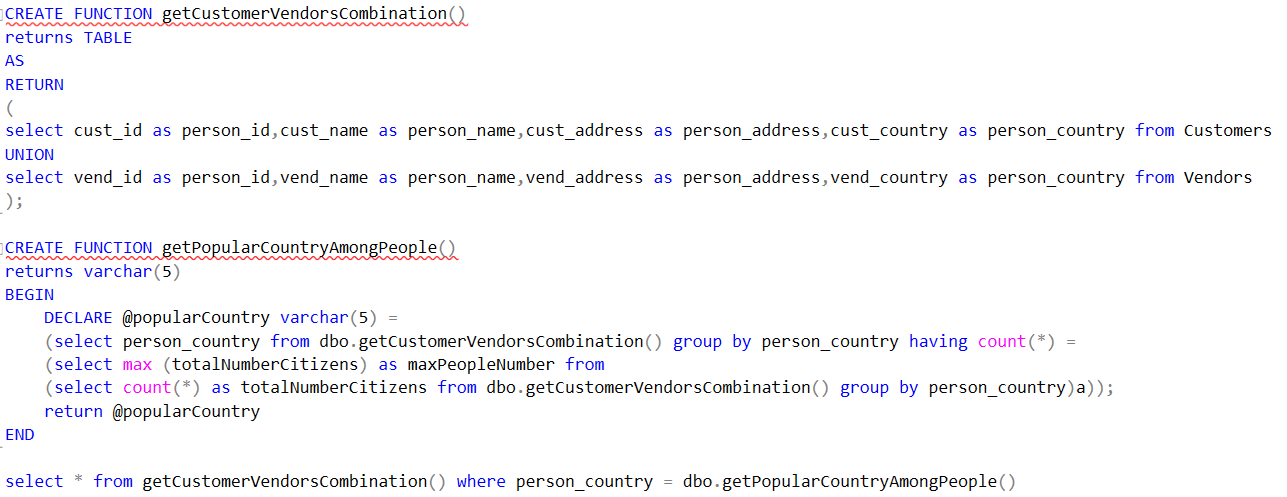




**8 QUERIES FOR FUNCTIONS  
 (Also very interesting)**

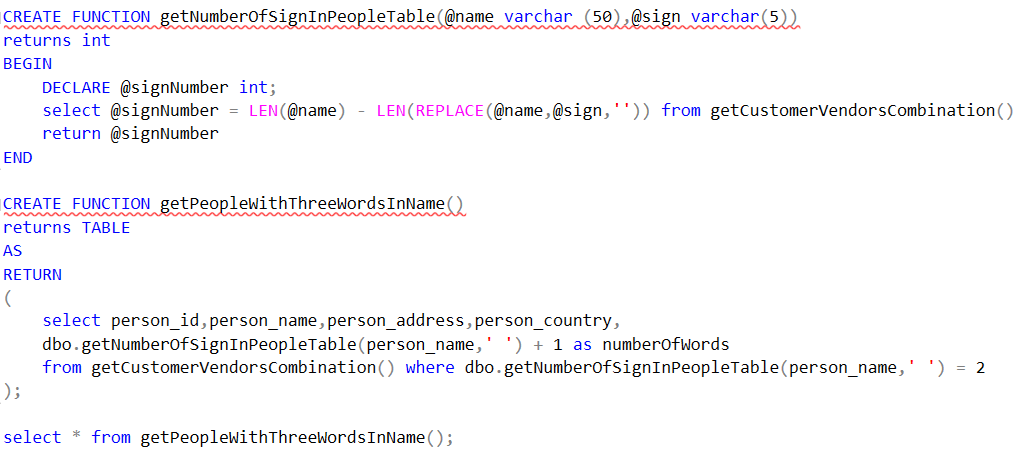
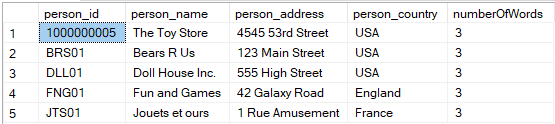
1. Create a function that would replace the real sum for the prices of the products and return this value.

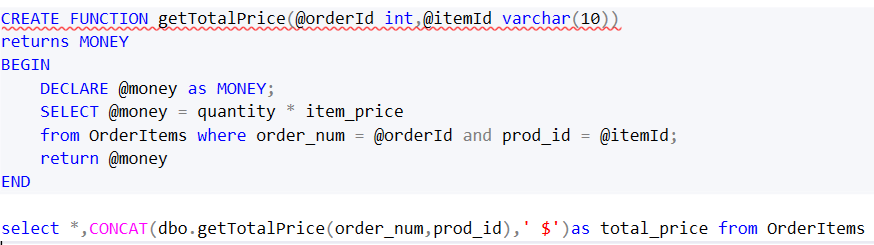


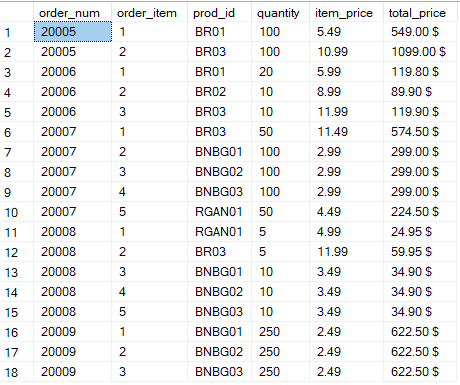
1. Write the function that would increase a price of the products by 50 percent and returns this new price.
2. Write to functions where the first is responsible for the union of customers and vendors and it is returned as a table while the second function needs to take this new table from the first function. The second function has to return the most popular country among people. Show the people who live in the most popular Country.

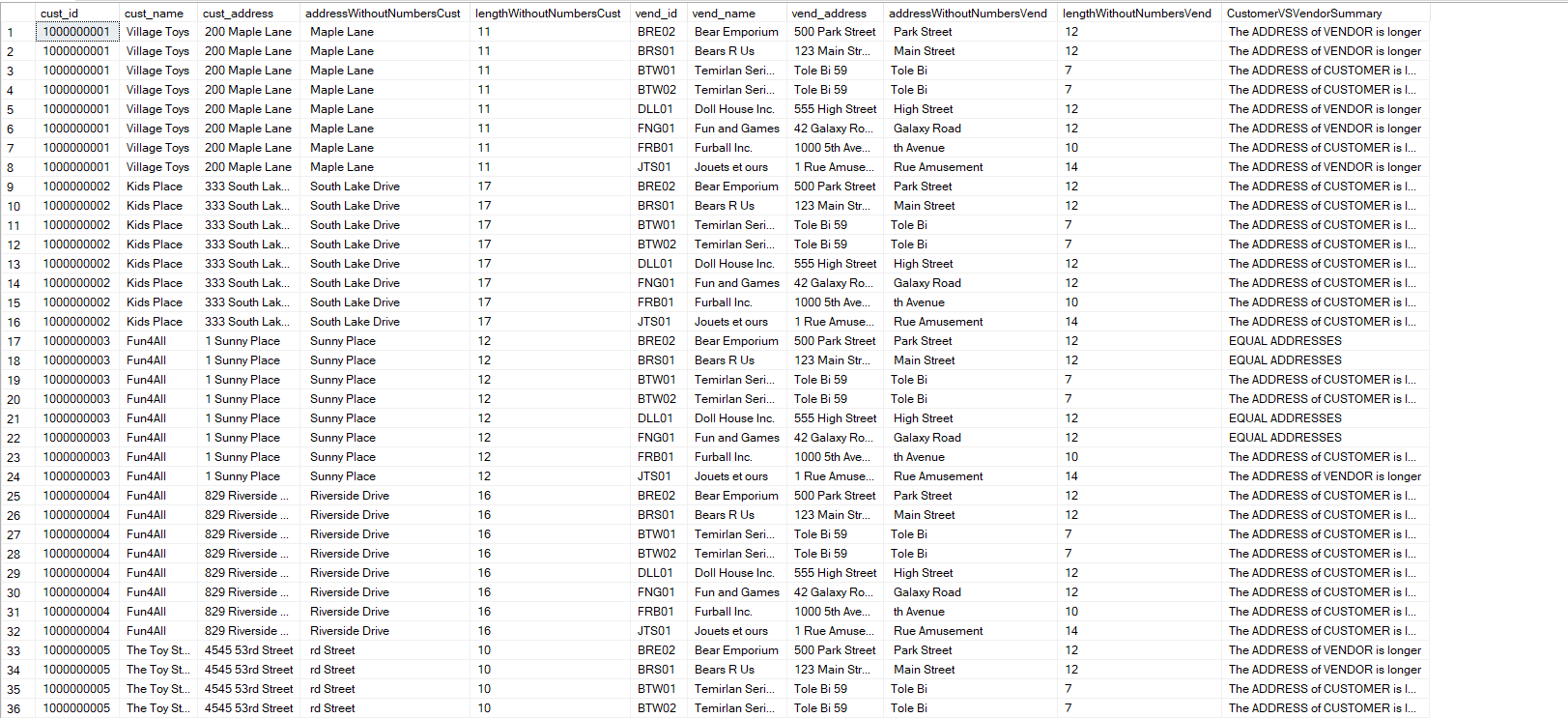
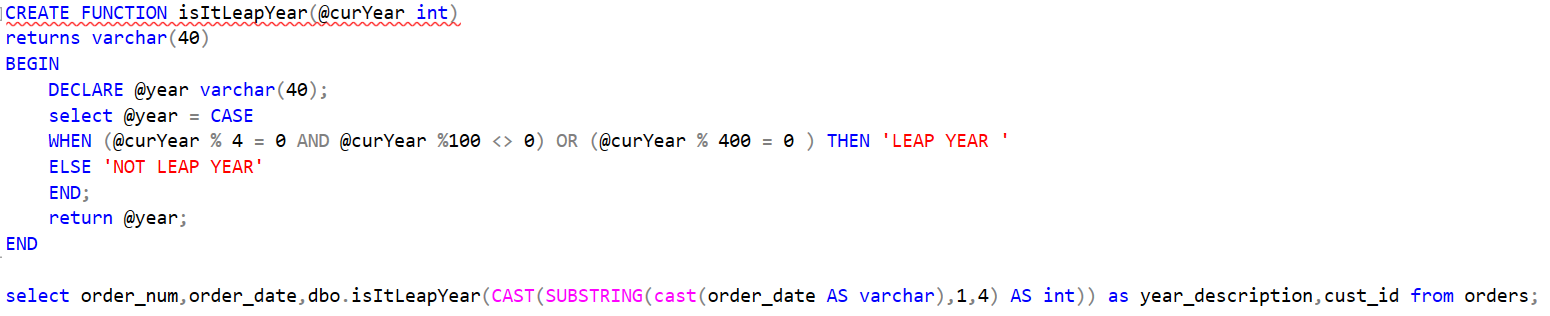


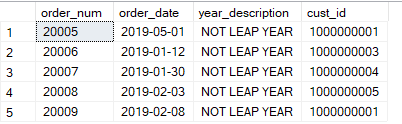
1. Write the functions that finally return a table with those people whose names consists of 3 words. Then show the whole information of these people.



1. Write a function that would return the total price that person needs to pay. Just multiply the quantity of items by cost of one. Use the type MONEY with dollars sign to make it more realistic.



1. Write two functions where one will return the address for the supplied person without any figures while the second function would use it and compare the lengths of the addresses between customers and vendors and SHOWS The ADDRESS of CUSTOMER is longer if the length of the customer’s address without numbers is longer than vendor’s , The ADDRESS of VENDOR is longer if vendor’s if bigger or EQUAL ADDRESSES if they are the same. Compare all the customers to the vedors using CARTESIAN PRODUCT.
2. Write a function that would check the taken year and return ‘LEAP YEAR’ or ‘NOT LEAP YEAR’.



1. Write functions that together would produce the sum of the figures that are inside of the ZIP column but there problem can appear since there are some rows which have not only numbers but letters too. For example, the zip = N16 6PS would produce 13 (1 + 6 + 6). Produce the solution for all the Zip numbers among the whole people. You can create a view where you would store al the people (Customers + Vendors).

