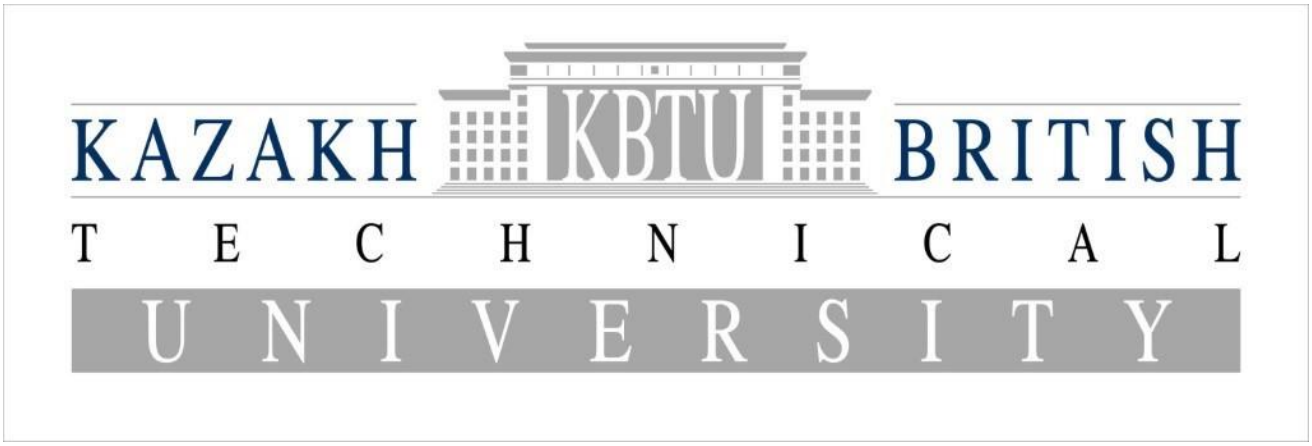

JSC "Kazakh British Technical University"
School of Mathematic and Cybernetics

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.

```
CREATE PROCEDURE spShowRegularOrNotCustomers
AS
BEGIN
    select o.cust_id,c.cust_name,c.cust_address,c.cust_city,c.cust_email,count(*) amount_orders,
    CASE WHEN COUNT(*) >=2 THEN 'REGULAR CUSTOMER'
    ELSE 'SELDOM CUSTOMER'
    END AS customer_description
    from Customers c join Orders o on c.cust_id = o.cust_id group by o.cust_id,c.cust_name,c.cust_address,c.cust_city,c.cust_email;
END

exec spShowRegularOrNotCustomers;
```

	cust_id	cust_name	cust_address	cust_city	cust_email	amount_orders	customer_description
1	1000000001	Village Toys	200 Maple Lane	Detroit	mailto:sales@villagetoys.com	2	REGULAR CUSTOMER
2	1000000003	Fun4All	1 Sunny Place	Muncie	<mailto:jjones@fun4all.com	1	SELDOM CUSTOMER
3	1000000004	Fun4All	829 Riverside Drive	Phoenix	<mailto:dstephens@fun4all.com	1	SELDOM CUSTOMER
4	1000000005	The Toy Store	4545 53rd Street	Chicago	NULL	1	SELDOM CUSTOMER

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.

```
CREATE PROCEDURE spGetPopularVendor
AS
BEGIN
     DECLARE @avgNumb INT = (select (max(numb_goods) + min(numb_goods))/2
     from (select count(*) as numb_goods from products group by vend_id)a);
     DECLARE @vendor_id varchar(10) = (select TOP(1) vend_id
     from Products group by vend_id having count(*) = @avgNumb);
     exec spGetVendorInfo @ven_id = @vendor_id;
END

CREATE PROCEDURE spGetVendorInfo
@ven_id varchar(10)
AS
BEGIN
    select * from Vendors where vend_id = @ven_id;
END

exec spGetPopularVendor;
```

	vend_id	vend_name	vend_address	vend_city	vend_state	vend_zip	vend_country
1	BRS01	Bears R Us	123 Main Street	Bear Town	MI	44444	USA

3. 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.

```
CREATE PROCEDURE spGetThreFourthProducts
AS
BEGIN
    DECLARE @avgPrice real = (select avg(prod_price) from Products);
    DECLARE @maxPrice real = (select max(prod_price) from Products);
    DECLARE @secAveragePrice real = (select avg(prod_price) from Products where prod_price BETWEEN @avgPrice and @maxPrice)
    select * from Products where prod_price between @avgPrice  and @secAveragePrice
END

exec spGetThreFourthProducts;
```

	prod_id	vend_id	prod_name	amount	prod_price	prod_desc
1	BR02	BRS01	12 inch teddy bear	18	8.99	12 inch teddy bear, comes with cap and jacket
2	RYL01	FNG01	King doll	120	9.49	12 inch king doll with royal garments and crown
3	RYL02	FNG01	Queen doll	18	9.49	12 inch queen doll with royal garments and crown

4. 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.

```
CREATE PROCEDURE spShowLenVendEvenOddZip
AS
BEGIN
    select vend_id,vend_name,vend_zip,
    CASE WHEN LEN(vend_zip)%2 =0 THEN 'Even Zip'
    ELSE 'Odd Zip'
    END AS zip_description
    from Vendors;
END

execute spShowLenVendEvenOddZip;
```

	vend_id	vend_name	vend_zip	zip_description
1	BRE02	Bear Emporium	44333	Odd Zip
2	BRS01	Bears R Us	44444	Odd Zip
3	BTW01	Temirlan Serikov	123123	Even Zip
4	DLL01	Doll House Inc.	99999	Odd Zip
5	FNG01	Fun and Games	N16 6PS	Odd Zip
6	FRB01	Furball Inc.	11111	Odd Zip
7	JTS01	Jouets et ours	45678	Odd Zip

5. 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:

```
CREATE PROCEDURE spShowOrderItemSumWithSumQuantityComparison
@number int
AS
BEGIN
    DECLARE @orderItemSum int = (select sum(order_item) from OrderItems);
    DECLARE @sumQuantity int = (select sum(OrderItems.quantity) from OrderItems);
    PRINT
    CASE WHEN @orderItemSum + @number < @sumQuantity THEN CONCAT(@orderItemSum + @number,' < ',@sumQuantity,' (is less than)')
    WHEN @orderItemSum + @number > @sumQuantity THEN CONCAT(@orderItemSum + @number,' > ',@sumQuantity,' (is higher than)')
    ELSE CONCAT(@orderItemSum + @number,' = ',@sumQuantity,' (is equal to) ')
    END;
END

execute spShowOrderItemSumWithSumQuantityComparison 1400;
```

```
1445 > 1430 (is higher than)

Completion time: 2021-04-20T18:22:08.2575454+06:00
```

7 queries for Triggers (Very Interesting) 😊

6. 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.

```
CREATE TABLE tbCustomersAudit(
    id int IDENTITY(1,1),
    auditData varchar(max)
    );

CREATE TRIGGER tr_CustomerForInsert
ON Customers
FOR INSERT
AS
BEGIN
    DECLARE @id int
    Select @id = cust_id from inserted

    insert into tbCustomersAudit(auditData) values
    ('New Employee with ID = ' +
    CAST(@id as varchar) + ' is added at ' +
    cast(GetDate() as varchar))
END

insert into Customers values (1000000006,'Temirbolat Maratuly','Erzhanova 39A','Karagandy','HZ','100009','KAZ','Bred Pit','t_maratuly@kbtu.kz');
insert into Customers values (1000000007,'Tamerlan Kuankush','Erzhanova 39A','Karagandy','HZ','100009','KAZ','Bred Pit','t_kuankash@kbtu.kz');
```

	id	auditData
1	1	New Employee with ID = 1000000006 is added at Apr 20 2021 12:45PM
2	2	New Employee with ID = 1000000007 is added at Apr 20 2021 1:00PM

7. 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)”.

```
CREATE TRIGGER tr_CustomerForDelete
ON Customers
FOR DELETE
AS
BEGIN
    DECLARE @id int
    select @id = cust_id from deleted

    insert into tbCustomersAudit values
    ('An existing customer with ID = ' +
    CAST(@id as varchar) + ' is deleted at ' +
    cast(GetDate() as varchar))
END
delete from Customers where cust_id = 1000000007;
```

	id	auditData
1	1	New Employee with ID = 1000000006 is added at Apr 20 2021 12:45PM
2	2	New Employee with ID = 1000000007 is added at Apr 20 2021 1:00PM
3	3	An existing customer with ID = 1000000007 is deleted at Apr 20 2021 1:07PM

8. 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.

```
CREATE TRIGGER tr_CustomerUpdate
on Customers
FOR UPDATE
AS
BEGIN
    DECLARE @Id int
    DECLARE @oldName varchar(40), @newName varchar(40)
    DECLARE @oldAddress varchar(40), @newAddress varchar(40)
    DECLARE @oldCity varchar(20), @newCity varchar(20)
    DECLARE @oldState varchar(4), @newState varchar(4)
    DECLARE @oldZip varchar(10), @newZip varchar(10)
    DECLARE @oldCountry varchar(5), @newCountry varchar(5)
    DECLARE @oldContact varchar(40), @newContact varchar(40)
    DECLARE @oldEmail varchar(40), @newEmail varchar(40)

    DECLARE @finalString varchar(max)

    select * into #TempTable from inserted

    WHILE(EXISTS(select cust_id from #TempTable))
    BEGIN
        SET @finalString = ''
        SELECT TOP 1 @Id = cust_id, @newName = cust_name, @newAddress  = cust_address, @newCity = cust_city, @newState = cust_state,
        @newZip = cust_zip, @newCountry = cust_country, @newContact = cust_contact,@newEmail = cust_email from #TempTable

        SELECT @oldName = cust_name,@oldAddress = cust_address,@oldCity = cust_city,@oldState = cust_state,
        @oldZip = cust_zip, @oldCountry = cust_country, @oldContact = cust_contact, @oldEmail = cust_email from deleted where cust_id = @Id

        SET @finalString = 'Customer With  ID = ' + CAST(@Id as varchar) + ' changed'
        if (@oldName <> @newName )
                SET @finalString = @finalString + ' NAME from ' + @oldName + ' to ' + @newName
        if (@oldAddress <> @newAddress)
                SET @finalString = @finalString + ' ADDRESS from ' + @oldAddress + ' to '+ @newAddress
        if (@oldCity <> @newCity)
                SET @finalString = @finalString + ' CITY from ' + @oldCity + ' to ' + @newCity
        if (@oldState <> @newState)
                SET @finalString = @finalString + ' STATE from ' + @oldState + ' to ' + @newState
        if (@oldZip <> @newZip)
                SET @finalString = @finalString + ' ZIP from ' + @oldZip + ' to ' + @newZip
        if (@oldCountry <> @newCountry)
                SET @finalString = @finalString + ' COUNTRY from ' + @oldCountry + ' to ' + @newCountry
        if (@oldContact <> @newContact)
                SET @finalString = @finalString + ' CONTACT from ' + @oldContact + ' to ' + @newContact
        if (@oldEmail <> @newEmail)
                SET @finalString = @finalString + ' E-MAIL from ' + @oldEmail + ' to ' + @newEmail

        insert into tbCustomersAudit values (@finalString)

        Delete from #TempTable where cust_id = @Id
    END
END

UPDATE Customers SET cust_name = 'Tamerlan Kuankush', cust_city = 'Almaty', cust_state = 'KZ', cust_zip = '100005',cust_email = 't_ku@kbtu.kz' where cust_id = 1000000006;
```

	id	auditData
1	1	New Employee with ID = 1000000006 is added at Apr 20 2021 12:45PM
2	2	New Employee with ID = 1000000007 is added at Apr 20 2021 1:00PM
3	3	An existing customer with ID = 1000000007 is deleted at Apr 20 2021 1:07PM
4	4	Customer With ID = 1000000006 changed NAME from Temirbolat Maratuly to Tamerlan Kuankush CITY from Karagandy to Almaty STATE from HZ to KZ ZIP from 100009 to 100005 E-MAIL from t_maratuly@kbtu.kz to t_ku@kbtu.kz

9. 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 VIEW vWVendorsProductsDetails
AS
select p.prod_id,p.prod_name,p.amount,
p.prod_price,p.prod_desc,v.vend_name
from Products p join Vendors v on v.vend_id  = p.vend_id;
```

	prod_id	prod_name	amount	prod_price	prod_desc	vend_name
1	BNBG01	Fish bean bag toy	50	3.29	Fish bean bag toy, complete with bean bag worms ...	Doll House Inc.
2	BNBG02	Bird bean bag toy	200	3.49	Bird bean bag toy, eggs are not included	Doll House Inc.
3	BNBG03	Rabbit bean bag toy	140	3.49	Rabbit bean bag toy, comes with bean bag carrots	Doll House Inc.
4	BR01	8 inch teddy bear	100	5.99	8 inch teddy bear, comes with cap and jacket	Bears R Us
5	BR02	12 inch teddy bear	18	8.99	12 inch teddy bear, comes with cap and jacket	Bears R Us
6	BR03	18 inch teddy bear	35	11.99	18 inch teddy bear, comes with cap and jacket	Bears R Us
7	RGAN01	Raggedy Ann	45	4.99	18 inch Raggedy Ann doll	Doll House Inc.
8	RYL01	King doll	120	9.49	12 inch king doll with royal garments and crown	Fun and Games
9	RYL02	Queen doll	18	9.49	12 inch queen doll with royal garments and crown	Fun and Games

Create a TRIGGER FOR INSERT:

```
CREATE TRIGGER trVWVendordProductsDetailsInsteadOfInsert
on vWVendorsProductsDetails
Instead Of Insert
AS
BEGIN
    DECLARE @vendId varchar(10)

    SELECT @vendId = vend_id
    from Vendors
    join inserted
    on inserted.vend_name = Vendors.vend_name

    if (@vendId is NULL)
    BEGIN
        Raiserror('Invalid VENDOR NAME. TRE AGAIN PLEASE!',16,1)
        return
    END

    INSERT INTO Products(prod_id,vend_id,prod_name,amount,prod_price,prod_desc)
    SELECT prod_id,@vendId,prod_name,amount,prod_price,prod_desc
    from inserted
END

insert into vWVendorsProductsDetails values('Burg1','Burger',20,'3.49','Very tasty burger','Bears R Us');
```

	prod_id	prod_name	amount	prod_price	prod_desc	vend_name
1	BNBG01	Fish bean bag toy	50	3.29	Fish bean bag toy, complete with bean bag worms ...	Doll House Inc.
2	BNBG02	Bird bean bag toy	200	3.49	Bird bean bag toy, eggs are not included	Doll House Inc.
3	BNBG03	Rabbit bean bag toy	140	3.49	Rabbit bean bag toy, comes with bean bag carrots	Doll House Inc.
4	BR01	8 inch teddy bear	100	5.99	8 inch teddy bear, comes with cap and jacket	Bears R Us
5	BR02	12 inch teddy bear	18	8.99	12 inch teddy bear, comes with cap and jacket	Bears R Us
6	BR03	18 inch teddy bear	35	11.99	18 inch teddy bear, comes with cap and jacket	Bears R Us
7	Burg1	Burger	20	3.49	Very tasty burger	Bears R Us
8	RGAN01	Raggedy Ann	45	4.99	18 inch Raggedy Ann doll	Doll House Inc.
9	RYL01	King doll	120	9.49	12 inch king doll with royal garments and crown	Fun and Gam...
10	RYL02	Queen doll	18	9.49	12 inch queen doll with royal garments and crown	Fun and Gam...

10. 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.

```
CREATE TRIGGER trChangePriceOfTheNewItemsWithDifferenceAfterUpdate
on PRODUCTS
AFTER UPDATE
AS
BEGIN
    DECLARE @oldPrice real
    DECLARE @newPrice real

    select @oldPrice = prod_price from deleted
    select @newPrice = prod_price from inserted
    DECLARE @difPrice real = ABS(@newPrice - @oldPrice)

    IF(@difPrice = 0)
    BEGIN
    Raiserror('YOU HAVE TO CHANGE THE PRICE. TRY AGAIN PLEASE!',16,1)
        return
    END
    UPDATE Products
    set prod_price = @difPrice
    where prod_id = (select prod_id from inserted)
END

UPDATE Products
SET prod_price = 3.29
where prod_id = 'BNBG01';
select * from Products
```

If we apply the same new price :

```
Msg 50000, Level 16, State 1, Procedure trChangePriceOfTheNewItemsWithDifferenceAfterUpdate, Line 15 [Batch Start Line 242]
YOU HAVE TO CHANGE THE PRICE. TRY AGAIN PLEASE!

(1 row affected)

Completion time: 2021-04-20T19:13:56.3812721+06:00
```

Otherwise:

```
UPDATE Products
SET prod_price = 6
where prod_id = 'BNBG01';
select * from Products
```

sults | Messages

prod_id	vend_id	prod_name	amount	prod_price	prod_desc
BNBG01	DLL01	Fish bean bag toy	50	2.71	Fish bean bag toy, complete with bean bag worms ...
BNBG02	DLL01	Bird bean bag toy	200	3.49	Bird bean bag toy, eggs are not included
BNBG03	DLL01	Rabbit bean bag toy	140	3.49	Rabbit bean bag toy, comes with bean bag carrots
BR01	BRS01	8 inch teddy bear	100	5.99	8 inch teddy bear, comes with cap and jacket
BR02	BRS01	12 inch teddy bear	18	8.99	12 inch teddy bear, comes with cap and jacket
BR03	BRS01	18 inch teddy bear	35	11.99	18 inch teddy bear, comes with cap and jacket
Burg1	BRS01	Burger	20	3.49	Very tasty burger
RGAN01	DLL01	Raggedy Ann	45	4.99	18 inch Raggedy Ann doll
RYL01	FNG01	King doll	120	9.49	12 inch king doll with royal garments and crown
RYL02	FNG01	Queen doll	18	9.49	12 inch queen doll with royal garments and crown

11. 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.

```
CREATE TRIGGER trShowMessageVendorAfterInsert
on VENDORS
after insert
AS
BEGIN
    PRINT 'The vendor is inserted successfully!'
END

insert into Vendors values('BTW01','Temirlan Serikov','Tole Bi 59','Almaty','HZ','123123','KAZ')
```

```
The vendor is inserted successfully!

(1 row affected)
```

12. 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.

```
select * from vWVendorsProductsDetails ;

CREATE TRIGGER trvWVendorsProductsDetailsInsteadDelete
on vWVendorsProductsDetails
instead of DELETE
as
BEGIN
    DELETE from Products
    where prod_id in (select prod_id from deleted)
END

delete from vWVendorsProductsDetails where prod_id LIKE 'Burg1';
```

	prod_id	prod_name	amount	prod_price	prod_desc	vend_name
1	BNBG01	Fish bean bag toy	50	2.71	Fish bean bag toy, complete with bean bag worms ...	Doll House Inc.
2	BNBG02	Bird bean bag toy	200	3.49	Bird bean bag toy, eggs are not included	Doll House Inc.
3	BNBG03	Rabbit bean bag toy	140	3.49	Rabbit bean bag toy, comes with bean bag carrots	Doll House Inc.
4	BR01	8 inch teddy bear	100	5.99	8 inch teddy bear, comes with cap and jacket	Bears R Us
5	BR02	12 inch teddy bear	18	8.99	12 inch teddy bear, comes with cap and jacket	Bears R Us
6	BR03	18 inch teddy bear	35	11.99	18 inch teddy bear, comes with cap and jacket	Bears R Us
7	RGAN01	Raggedy Ann	45	4.99	18 inch Raggedy Ann doll	Doll House Inc.
8	RYL01	King doll	120	9.49	12 inch king doll with royal garments and crown	Fun and Games
9	RYL02	Queen doll	18	9.49	12 inch queen doll with royal garments and crown	Fun and Games

8 QUERIES FOR FUNCTIONS
(Also very interesting)

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

```
create function ownSumById
(@productId varchar(20))
returns numeric (9,2)
BEGIN
DECLARE @totalSumProductItem numeric (9,2);
select @totalSumProductItem =  sum(item_price) from OrderItems where prod_id = @productId;
return @totalSumProductItem;
END

select prod_id,dbo.ownSumById(Products.prod_id) as totalSoldPrice,prod_desc
from products where dbo.ownSumById(prod_id) IS NOT NULL;
```

	prod_id	totalSoldPrice	prod_desc
1	BNBG01	8.97	Fish bean bag toy, complete with bean bag worms ...
2	BNBG02	8.97	Bird bean bag toy, eggs are not included
3	BNBG03	8.97	Rabbit bean bag toy, comes with bean bag carrots
4	BR01	11.48	8 inch teddy bear, comes with cap and jacket
5	BR02	8.99	12 inch teddy bear, comes with cap and jacket
6	BR03	46.46	18 inch teddy bear, comes with cap and jacket
7	RGAN01	9.48	18 inch Raggedy Ann doll

14. Write the function that would increase a price of the products by 50 percent and returns this new price.

```
CREATE FUNCTION getIncreasedPriceByFiftyPercent
(@productId varchar(20))
returns numeric(9,2)
BEGIN
DECLARE @newPrice numeric(9,2);
select @newPrice = prod_price*1.5 from Products where prod_id = @productId;
return @newPrice;
END

select prod_id,prod_price as oldPrice,dbo.getIncreasedPriceByFiftyPercent(prod_id) as increasedCostByFiftyPercen  from Products;
```

	prod_id	oldPrice	increasedCostByFiftyPercen
1	BNBG01	2.71	4.07
2	BNBG02	3.49	5.24
3	BNBG03	3.49	5.24
4	BR01	5.99	8.99
5	BR02	8.99	13.49
6	BR03	11.99	17.99
7	RGAN01	4.99	7.49
8	RYL01	9.49	14.24
9	RYL02	9.49	14.24

15. 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.

```
CREATE FUNCTION getCustomerVendorsCombination()
returns TABLE
AS
RETURN
(
select cust_id as person_id,cust_name as person_name,cust_address as person_address,cust_country as person_country from Customers
UNION
select vend_id as person_id,vend_name as person_name,vend_address as person_address,vend_country as person_country from Vendors
);

CREATE FUNCTION getPopularCountryAmongPeople()
returns varchar(5)
BEGIN
    DECLARE @popularCountry varchar(5) =
    (select person_country from dbo.getCustomerVendorsCombination() group by person_country having count(*) =
    (select max (totalNumberCitizens) as maxPeopleNumber from
    (select count(*) as totalNumberCitizens from dbo.getCustomerVendorsCombination() group by person_country)a));
    return @popularCountry
END

select * from getCustomerVendorsCombination() where person_country = dbo.getPopularCountryAmongPeople()
```

	person_id	person_name	person_address	person_country
1	1000000001	Village Toys	200 Maple Lane	USA
2	1000000002	Kids Place	333 South Lake Drive	USA
3	1000000003	Fun4All	1 Sunny Place	USA
4	1000000004	Fun4All	829 Riverside Drive	USA
5	1000000005	The Toy Store	4545 53rd Street	USA
6	BRE02	Bear Empori...	500 Park Street	USA
7	BRS01	Bears R Us	123 Main Street	USA
8	DLL01	Doll House I...	555 High Street	USA
9	FRB01	Furball Inc.	1000 5th Avenue	USA

16. 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.

```
CREATE FUNCTION getNumberOfSignInPeopleTable(@name varchar (50),@sign varchar(5))
returns int
BEGIN
    DECLARE @signNumber int;
    select @signNumber = LEN(@name) - LEN(REPLACE(@name,@sign,'')) from getCustomerVendorsCombination()
    return @signNumber
END

CREATE FUNCTION getPeopleWithThreeWordsInName()
returns TABLE
AS
RETURN
(
    select person_id,person_name,person_address,person_country,
    dbo.getNumberOfSignInPeopleTable(person_name,' ') + 1 as numberOfWords
    from getCustomerVendorsCombination() where dbo.getNumberOfSignInPeopleTable(person_name,' ') = 2
);

select * from getPeopleWithThreeWordsInName();
```

	person_id	person_name	person_address	person_country	numberOfWords
1	1000000005	The Toy Store	4545 53rd Street	USA	3
2	BRS01	Bears R Us	123 Main Street	USA	3
3	DLL01	Doll House Inc.	555 High Street	USA	3
4	FNG01	Fun and Games	42 Galaxy Road	England	3
5	JTS01	Jouets et ours	1 Rue Amusement	France	3

17. 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.

```
CREATE FUNCTION getTotalPrice(@orderId int,@itemId varchar(10))
returns MONEY
BEGIN
    DECLARE @money as MONEY;
    SELECT @money = quantity * item_price
    from OrderItems where order_num = @orderId and prod_id = @itemId;
    return @money
END

select *,CONCAT(dbo.getTotalPrice(order_num,prod_id),' $')as total_price from OrderItems
```

	order_num	order_item	prod_id	quantity	item_price	total_price
1	20005	1	BR01	100	5.49	549.00 $
2	20005	2	BR03	100	10.99	1099.00 $
3	20006	1	BR01	20	5.99	119.80 $
4	20006	2	BR02	10	8.99	89.90 $
5	20006	3	BR03	10	11.99	119.90 $
6	20007	1	BR03	50	11.49	574.50 $
7	20007	2	BNBG01	100	2.99	299.00 $
8	20007	3	BNBG02	100	2.99	299.00 $
9	20007	4	BNBG03	100	2.99	299.00 $
10	20007	5	RGAN01	50	4.49	224.50 $
11	20008	1	RGAN01	5	4.99	24.95 $
12	20008	2	BR03	5	11.99	59.95 $
13	20008	3	BNBG01	10	3.49	34.90 $
14	20008	4	BNBG02	10	3.49	34.90 $
15	20008	5	BNBG03	10	3.49	34.90 $
16	20009	1	BNBG01	250	2.49	622.50 $
17	20009	2	BNBG02	250	2.49	622.50 $
18	20009	3	BNBG03	250	2.49	622.50 $

18. 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.

```
CREATE FUNCTION getAddressesWithoutNumbers(@address varchar(50))
returns varchar(50)
BEGIN
    DECLARE @newAddress varchar(50) = '';
    DECLARE @size int = len(@address);
    DECLARE @cnt int = 1;
    WHILE (@cnt<=@size)
    BEGIN
        if(SUBSTRING(@address,@cnt,1) NOT LIKE '[0123456789]%' AND @cnt <= @size)
            SET @newAddress = CONCAT(@newAddress,SUBSTRING(@address,@cnt,1))
        SET @cnt = @cnt + 1
    END
    return @newAddress;
END

CREATE FUNCTION compareCustomersVendorsAddresses(@custAddress varchar(50),@vendAddress varchar(50))
returns varchar(70)
BEGIN
    DECLARE @description varchar(70);
    select  @description =
    CASE WHEN LEN(dbo.getAddressesWithoutNumbers(@custAddress)) > LEN(dbo.getAddressesWithoutNumbers(@vendAddress))
    THEN 'The ADDRESS of CUSTOMER is longer'
    WHEN LEN(dbo.getAddressesWithoutNumbers(@custAddress)) < LEN(dbo.getAddressesWithoutNumbers(@vendAddress))
    THEN 'The ADDRESS of VENDOR is longer'
    ELSE 'EQUAL ADDRESSES'
    END;
    return @description;
END

select CUSTOMERS.cust_id,CUSTOMERS.cust_name,CUSTOMERS.cust_address,dbo.getAddressesWithoutNumbers(cust_address) as addressWithoutNumbersCust,
LEN(dbo.getAddressesWithoutNumbers(cust_address)) as lengthWithoutNumbersCust
,VENDORS.vend_id,VENDORS.vend_name,VENDORS.vend_address,dbo.getAddressesWithoutNumbers(vend_address) as addressWithoutNumbersVend,
LEN(dbo.getAddressesWithoutNumbers(vend_address)) as lengthWithoutNumbersVend,
dbo.compareCustomersVendorsAddresses(cust_address,vend_address) as CustomerVSVendorSummary from Customers,Vendors;
```

	cust_id	cust_name	cust_address	addressWithoutNumbersCust	lengthWithoutNumbersCust	vend_id	vend_name	vend_address	addressWithoutNumbersVend	lengthWithoutNumbersVend	CustomerVSVendorSummary
1	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	BRE02	Bear Emporium	500 Park Street	Park Street	12	The ADDRESS of VENDOR is longer
2	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	BRS01	Bears R Us	123 Main Str...	Main Street	12	The ADDRESS of VENDOR is longer
3	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	BTW01	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
4	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	BTW02	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
5	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	DLL01	Doll House Inc.	555 High Street	High Street	12	The ADDRESS of VENDOR is longer
6	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	FNG01	Fun and Games	42 Galaxy Ro...	Galaxy Road	12	The ADDRESS of VENDOR is longer
7	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	FRB01	Furball Inc.	1000 5th Ave...	th Avenue	10	The ADDRESS of CUSTOMER is l...
8	1000000001	Village Toys	200 Maple Lane	Maple Lane	11	JTS01	Jouets et ours	1 Rue Amuse...	Rue Amusement	14	The ADDRESS of VENDOR is longer
9	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	BRE02	Bear Emporium	500 Park Street	Park Street	12	The ADDRESS of CUSTOMER is l...
10	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	BRS01	Bears R Us	123 Main Str...	Main Street	12	The ADDRESS of CUSTOMER is l...
11	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	BTW01	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
12	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	BTW02	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
13	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	DLL01	Doll House Inc.	555 High Street	High Street	12	The ADDRESS of CUSTOMER is l...
14	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	FNG01	Fun and Games	42 Galaxy Ro...	Galaxy Road	12	The ADDRESS of CUSTOMER is l...
15	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	FRB01	Furball Inc.	1000 5th Ave...	th Avenue	10	The ADDRESS of CUSTOMER is l...
16	1000000002	Kids Place	333 South Lak...	South Lake Drive	17	JTS01	Jouets et ours	1 Rue Amuse...	Rue Amusement	14	The ADDRESS of CUSTOMER is l...
17	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	BRE02	Bear Emporium	500 Park Street	Park Street	12	EQUAL ADDRESSES
18	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	BRS01	Bears R Us	123 Main Str...	Main Street	12	EQUAL ADDRESSES
19	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	BTW01	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
20	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	BTW02	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
21	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	DLL01	Doll House Inc.	555 High Street	High Street	12	EQUAL ADDRESSES
22	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	FNG01	Fun and Games	42 Galaxy Ro...	Galaxy Road	12	EQUAL ADDRESSES
23	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	FRB01	Furball Inc.	1000 5th Ave...	th Avenue	10	The ADDRESS of CUSTOMER is l...
24	1000000003	Fun4All	1 Sunny Place	Sunny Place	12	JTS01	Jouets et ours	1 Rue Amuse...	Rue Amusement	14	The ADDRESS of VENDOR is longer
25	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	BRE02	Bear Emporium	500 Park Street	Park Street	12	The ADDRESS of CUSTOMER is l...
26	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	BRS01	Bears R Us	123 Main Str...	Main Street	12	The ADDRESS of CUSTOMER is l...
27	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	BTW01	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
28	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	BTW02	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
29	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	DLL01	Doll House Inc.	555 High Street	High Street	12	The ADDRESS of CUSTOMER is l...
30	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	FNG01	Fun and Games	42 Galaxy Ro...	Galaxy Road	12	The ADDRESS of CUSTOMER is l...
31	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	FRB01	Furball Inc.	1000 5th Ave...	th Avenue	10	The ADDRESS of CUSTOMER is l...
32	1000000004	Fun4All	829 Riverside ...	Riverside Drive	16	JTS01	Jouets et ours	1 Rue Amuse...	Rue Amusement	14	The ADDRESS of CUSTOMER is l...
33	1000000005	The Toy St...	4545 53rd Street	rd Street	10	BRE02	Bear Emporium	500 Park Street	Park Street	12	The ADDRESS of VENDOR is longer
34	1000000005	The Toy St...	4545 53rd Street	rd Street	10	BRS01	Bears R Us	123 Main Str...	Main Street	12	The ADDRESS of VENDOR is longer
35	1000000005	The Toy St...	4545 53rd Street	rd Street	10	BTW01	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...
36	1000000005	The Toy St...	4545 53rd Street	rd Street	10	BTW02	Temirlan Seri...	Tole Bi 59	Tole Bi	7	The ADDRESS of CUSTOMER is l...

19. Write a function that would check the taken year and return 'LEAP YEAR' or 'NOT LEAP YEAR'.

```
CREATE FUNCTION isItLeapYear(@curYear int)
returns varchar(40)
BEGIN
    DECLARE @year varchar(40);
    select @year = CASE
    WHEN (@curYear % 4 = 0 AND @curYear %100 <> 0) OR (@curYear % 400 = 0 ) THEN 'LEAP YEAR '
    ELSE 'NOT LEAP YEAR'
    END;
    return @year;
END

select order_num,order_date,dbo.isItLeapYear(CAST(SUBSTRING(cast(order_date AS varchar),1,4) AS int)) as year_description,cust_id from orders;
```

	order_num	order_date	year_description	cust_id
1	20005	2019-05-01	NOT LEAP YEAR	1000000001
2	20006	2019-01-12	NOT LEAP YEAR	1000000003
3	20007	2019-01-30	NOT LEAP YEAR	1000000004
4	20008	2019-02-03	NOT LEAP YEAR	1000000005
5	20009	2019-02-08	NOT LEAP YEAR	1000000001

20. 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).

```
CREATE FUNCTION getCorrectZipNumbers(@oldZip varchar(20))
returns varchar(20)
BEGIN
    DECLARE @sizeZip int = len(@oldZip);
    DECLARE @cnt int = 1;
    DECLARE @newZip varchar(20) = '';
    WHILE (@cnt <= @sizeZip)
    BEGIN
        if(SUBSTRING(@oldZip,@cnt,1) LIKE '[0123456789]%')
            SET @newZip = CONCAT(@newZip,SUBSTRING(@oldZip,@cnt,1))
        SET @cnt = @cnt + 1
    END
    return @newZip;
END

CREATE FUNCTION sumOfZipNumbers(@curZip varchar(20))
returns int
BEGIN
    DECLARE @sumNumbers int = 0;
    DECLARE @newZip varchar(20) = dbo.getCorrectZipNumbers(@curZip);
    DECLARE @counter int = 1;
    DECLARE @newZipSize int = len(@newZip);
    while (@counter <= @newZipSize)
    BEGIN
        SET @sumNumbers = @sumNumbers + CAST(SUBSTRING(@newZip,@counter,1) AS tinyint)
        SET @counter = @counter + 1;
    END
    return @sumNumbers;
END

CREATE VIEW people AS
select cust_id as person_id,cust_name as person_name,cust_zip as person_zip from Customers
union
select vend_id as person_id,vend_name as person_name,vend_zip as person_zip from Vendors;

select *, dbo.sumOfZipNumbers(CAST(people.person_zip as varchar)) from people;
```

	person_id	person_name	person_zip	(No column name)
1	1000000001	Village Toys	44444	20
2	1000000002	Kids Place	43333	16
3	1000000003	Fun4All	42222	12
4	1000000004	Fun4All	88888	40
5	1000000005	The Toy Store	54545	23
6	1000000006	Tamerlan K...	100005	6
7	BRE02	Bear Empori...	44333	17
8	BRS01	Bears R Us	44444	20
9	BTW01	Temirlan Se...	123123	12
10	BTW02	Temirlan Se...	123123	12
11	DLL01	Doll House I...	99999	45
12	FNG01	Fun and Ga...	N16 6PS	13
13	FRB01	Furball Inc.	11111	5
14	JTS01	Jouets et ours	45678	30