**SQL ASSIGNMENT – 4**

* **Creating Stored Procedure for calculating average freight value**

1. ALTER PROC spGetAvgFreight

@AvgFreight real output

AS

BEGIN

SELECT @AvgFreight = AVG(o.freight)

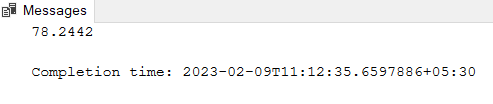
from orders as o

END

Declare @AvgFreight real

EXECUTE spGetAvgFreight @AvgFreight OUTPUT

print @AvgFreight



* **Creating Stored Procedure For Business Rule**

CREATE PROC spBeforeInsertUpdate

@customer\_id char(5),

@freight real = NULL

AS

BEGIN

IF NOT EXISTS(select @customer\_id from customers where customer\_id = @customer\_id)

BEGIN

RAISERROR(N'Given Customer ID % Does not Exist!', 16,1, @customer\_id)

RETURN

END

Declare @AvgFreight real

EXECUTE spGetAvgFreight @AvgFreight OUTPUT

IF @freight is NULL

BEGIN

select @freight = freight from orders where customer\_id = @customer\_id

END

IF @freight > @AvgFreight

BEGIN

RAISERROR(N'Given Freight exceeds Average Freight, Command will be Cancelled!', 16,1)

END

ELSE

BEGIN

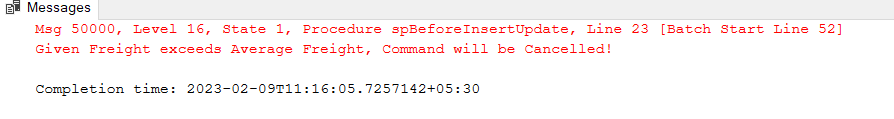
PRINT('Given freight is accepted Business Rule Value.')

END

END

spBeforeInsertUpdate QUICK

* **Customer “QUICK” Freight Value Exceeds the Average Freight Value**



1. CREATE PROC spGetEmployeeSalesByCountry

AS

BEGIN

select e.first\_name as Employee, round(sum((o.unit\_price \* o.quantity) - (o.unit\_price \* o.quantity \* o.discount)) ,2) as "Total Sale" , ord.ship\_country as Country

from employees as e

join orders as ord

on ord.employee\_id = e.employee\_id

join order\_details as o

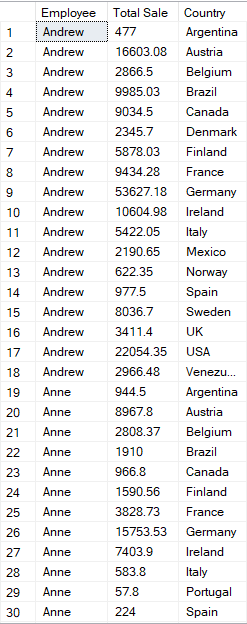
on o.order\_id = ord.order\_id

group by ord.ship\_country,e.first\_name

order by e.first\_name

END

EXECUTE spGetEmployeeSalesByCountry



1. CREATE PROC spGetSalesByYear

AS

BEGIN

select DATENAME(year,ord.shipped\_date) as Year, round(sum((o.unit\_price \* o.quantity) - (o.unit\_price \* o.quantity \* o.discount)) ,2) as "Total Sale"

from order\_details as o

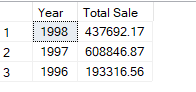
join orders as ord

on o.order\_id = ord.order\_id and ord.shipped\_date is not null

group by DATENAME(year,ord.shipped\_date)

END

EXECUTE spGetSalesByYear



1. CREATE PROC spGetSalesByCategory

AS

BEGIN

select c.CategoryName, round(sum((o.unit\_price \* o.quantity) - (o.unit\_price \* o.quantity \* o.discount)) ,2) as "Total Sale"

from order\_details as o

join Products as p

on o.product\_id = p.ProductID

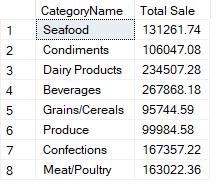
join Categories as c

on c.CategoryID = p.CategoryID

group by c.CategoryName

END

EXECUTE spGetSalesByCategory



1. CREATE PROC spGetTenMostExpensiveProducts

AS

BEGIN

select top 10 p.ProductName, p.UnitPrice

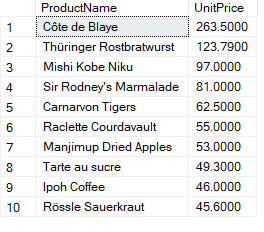
from Products as p

group by p.ProductName, p.UnitPrice

order by p.UnitPrice desc

END

EXECUTE spGetTenMostExpensiveProducts



1. CREATE PROC spInsertOrderDetails

@order\_id smallint,

@product\_id int,

@unit\_price real,

@quantity smallint,

@discount real

AS

BEGIN

IF EXISTS(SELECT \* from order\_details where order\_id = @order\_id and product\_id = @product\_id)

BEGIN

RAISERROR(N'Record is already exists with Order ID %d and Product Id %d',16,1, @order\_id, @product\_id)

return

END

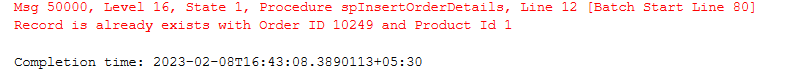
insert into order\_details values(@order\_id, @product\_id, @unit\_price, @quantity, @discount)

print('Record inserted Successfully!')

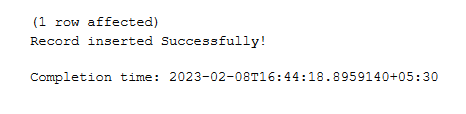
END

EXECUTE spInsertOrderDetails 10249,1, 11,2,0

* **Validations**
* If user enter Existed values



* If user enter valid values



1. CREATE PROC spUpdateOrderDetails

@order\_id smallint,

@product\_id int,

@column\_name varchar(20),

@new\_value real

AS

BEGIN

IF NOT EXISTS(SELECT \* from order\_details where order\_id = @order\_id and product\_id = @product\_id)

BEGIN

RAISERROR(N'Record is not exists with Order ID %d and Product Id %d',16,1, @order\_id, @product\_id)

return

END

DECLARE @query nvarchar(500)

set @query = 'Update order\_details

set' + QUOTENAME(@column\_name) + '=' + cast(@new\_value as varchar(20)) +

'where order\_id = '+ cast(@order\_id as varchar(20)) +'and product\_id =' + cast(@product\_id as varchar(20))

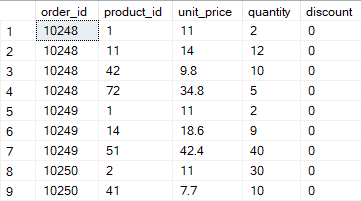
EXECUTE sp\_executesql @query

print('Record inserted Successfully!')

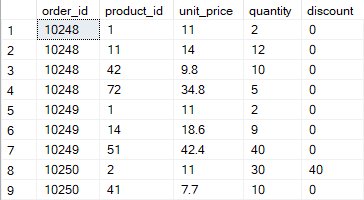
END

EXECUTE spUpdateOrderDetails 10250,2, 'discount',40

* **Before Executing Stored Procedures**



* **After Executing Stored Procedures**

****