**Assignment 4**

* **Queries**

**1)**

**Procedure to find average freight: -**

CREATE PROCEDURE usp\_GetAverageCustomerFreight(

@customer\_id char(5),

@average\_freight real = 0 output

)

AS

BEGIN

SELECT @average\_freight = AVG(freight)

FROM Customers c

JOIN Orders o ON c.customer\_id = o.customer\_id

WHERE c.customer\_id = @customer\_id

END

**Insert trigger to add business rule: -**

CREATE OR ALTER TRIGGER tr\_Orders\_InsteadOfInsert

ON Orders

INSTEAD OF INSERT

AS

BEGIN

DECLARE @freight real

DECLARE @customer\_id char(5)

DECLARE @order\_id smallint

SELECT @customer\_id = customer\_id, @freight = freight, @order\_id = order\_id

FROM inserted

IF NOT EXISTS(SELECT customer\_id FROM Customers WHERE customer\_id = @customer\_id)

BEGIN

RAISERROR('Customer id does not exists...', 16, 1)

RETURN

END

DECLARE @average\_freight real

EXECUTE usp\_GetAverageCustomerFreight @customer\_id, @average\_freight out

PRINT(N'Average freight is ' + CAST(@average\_freight as varchar(20)))

IF(@average\_freight < @freight)

BEGIN

RAISERROR(N'Freight value is exceeding average freight value for customer id "%s"', 16, 1, @customer\_id)

RETURN

END

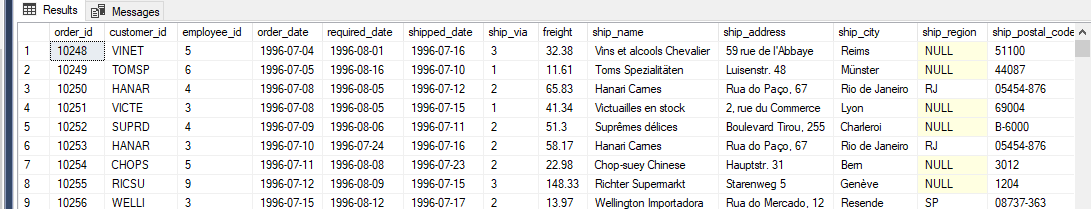
INSERT INTO Orders

SELECT \* FROM inserted

PRINT('Record inserted successfully...!')

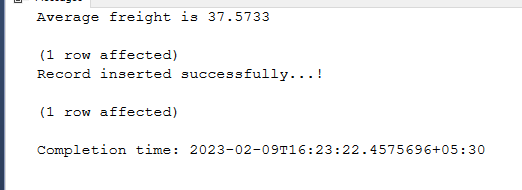
END

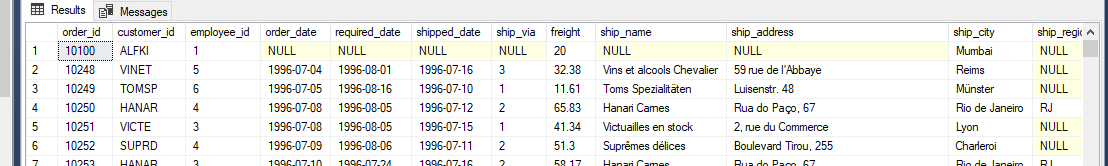
**Before inserting: -**



**After insert query: -**

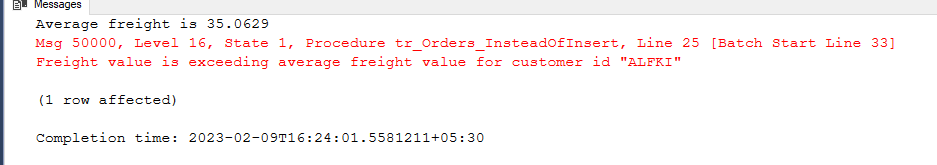
INSERT INTO Orders(order\_id, customer\_id, employee\_id, freight, ship\_city) VALUES(10100, 'ALFKI', 1, 20, 'Mumbai')





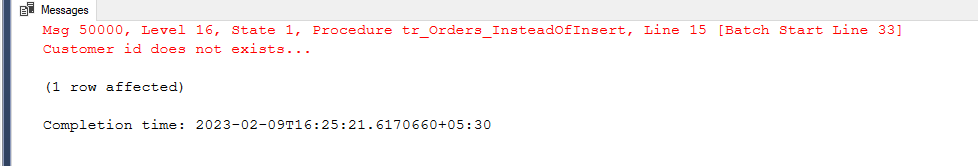
**For high freight value: -**

INSERT INTO Orders(order\_id, customer\_id, employee\_id, freight, ship\_city) VALUES(10100, 'ALFKI', 1, 200, 'Mumbai')



**For invalid customer id: -**

INSERT INTO Orders(order\_id, customer\_id, employee\_id, freight, ship\_city) VALUES(10101, 'AABCD', 1, 200, 'Mumbai')



**Update trigger for business rule: -**

CREATE OR ALTER TRIGGER tr\_Orders\_InsteadOfUpdate

ON Orders

INSTEAD OF UPDATE

AS

BEGIN

DECLARE @freight real = 0

DECLARE @customer\_id char(5)

DECLARE @order\_id smallint;

SELECT @customer\_id = customer\_id, @freight = freight, @order\_id = order\_id

FROM inserted

IF NOT EXISTS(SELECT order\_id FROM Orders WHERE order\_id = @order\_id)

BEGIN

RAISERROR(N'Order id does not exists...', 16, 1)

RETURN

END

DECLARE @average\_freight real

EXECUTE usp\_GetAverageCustomerFreight @customer\_id, @average\_freight out

PRINT(N'Average freight is ' + CAST(@average\_freight as varchar(20)))

IF(@average\_freight < @freight)

BEGIN

RAISERROR(N'Freight value is exceeding average freight value for customer id "%s"', 16, 1, @customer\_id)

RETURN

END

DECLARE @employee\_id smallint, @order\_date date, @required\_date date, @shipped\_date date, @ship\_via smallint,

@ship\_name varchar(40), @ship\_address varchar(60), @ship\_city varchar(15),

@ship\_region varchar(15), @ship\_postal\_code varchar(10), @ship\_country varchar(15);

SELECT @order\_id = i.order\_id,

@customer\_id = i.customer\_id,

@employee\_id = i.employee\_id,

@order\_date = i.order\_date,

@required\_date = i.required\_date,

@shipped\_date = i.shipped\_date,

@ship\_via = i.ship\_via,

@freight = i.freight,

@ship\_name = i.ship\_name,

@ship\_address = i.ship\_address,

@ship\_city = i.ship\_city,

@ship\_region = i.ship\_region,

@ship\_postal\_code = i.ship\_postal\_code,

@ship\_country = i.ship\_country

FROM inserted i

UPDATE Orders

SET customer\_id = @customer\_id,

employee\_id = @employee\_id,

order\_date = @order\_date,

required\_date = @required\_date,

shipped\_date = @shipped\_date,

ship\_via = @ship\_via,

freight = @freight,

ship\_name = @ship\_name,

ship\_address = @ship\_address,

ship\_city = @ship\_city,

ship\_region = @ship\_region,

ship\_postal\_code = @ship\_postal\_code,

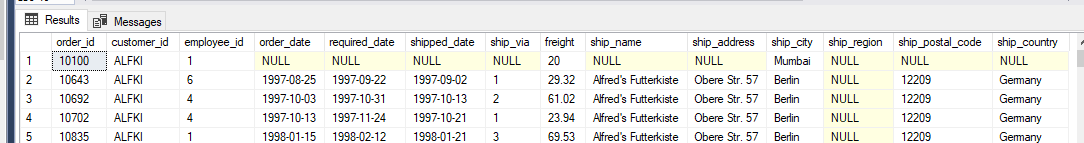
ship\_country = @ship\_country

WHERE order\_id = @order\_id

PRINT('Record updated successfully...!')

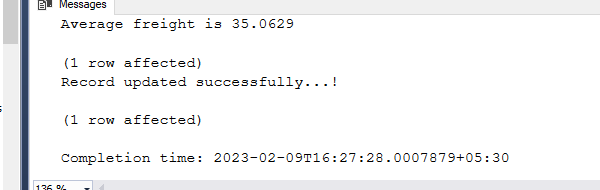
END

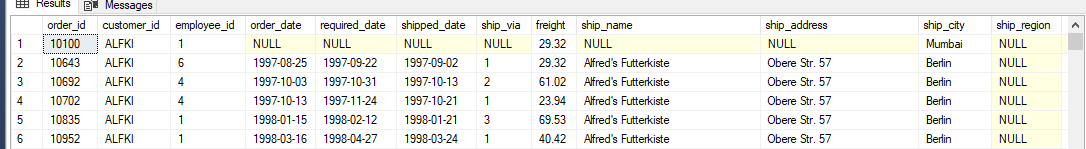
**Before updating: -**



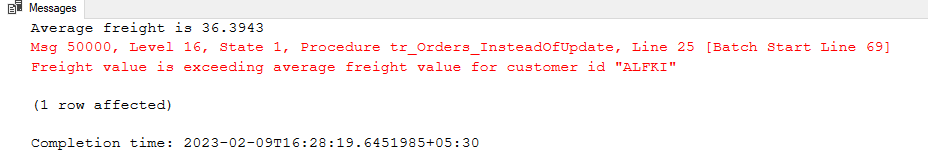
**After update query: -**

UPDATE Orders SET freight = 29.32 where order\_id = 10100

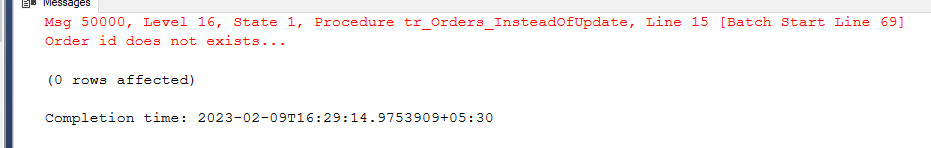




**For high freight value: -**



**For invalid order id: -**

****

**2)**

CREATE PROCEDURE usp\_GetEmployeesSalesByCountry

AS

BEGIN

SELECT e.employee\_id AS 'ID',

e.first\_name + ' ' + e.last\_name AS 'Employee Name',

o.ship\_country AS 'Country',

ROUND(SUM(od.unit\_price \* od.quantity - (od.unit\_price \* od.quantity \* od.discount)), 2) AS 'Total Sale'

FROM Employees e

JOIN Orders o ON e.employee\_id = o.employee\_id

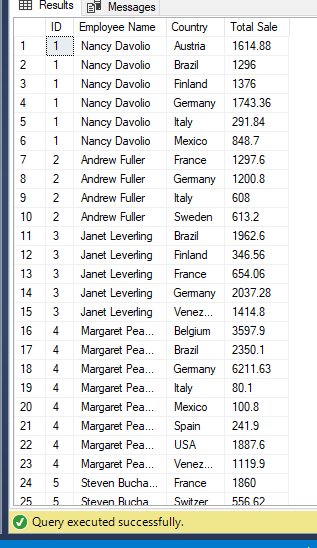
JOIN OrderDetails od ON o.order\_id = od.order\_id

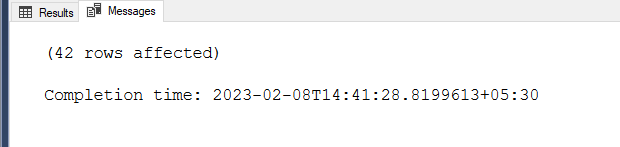
GROUP BY e.employee\_id, e.first\_name, e.last\_name, o.ship\_country

ORDER BY e.employee\_id

END

EXECUTE usp\_GetEmployeesSalesByCountry





**3)**

CREATE PROCEDURE usp\_GetSalesByYear

AS

BEGIN

SELECT DATENAME(Year, o.shipped\_date) AS 'Year',

ROUND(SUM(od.unit\_price \* od.quantity - (od.unit\_price \* od.quantity \* od.discount)), 2) AS 'Total Sale'

FROM Orders o

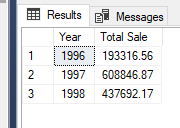
JOIN OrderDetails od ON o.order\_id = od.order\_id AND o.shipped\_date IS NOT NULL

GROUP BY DATENAME(Year, o.shipped\_date)

ORDER BY DATENAME(Year, o.shipped\_date)

END

EXECUTE usp\_GetSalesByYear



**4)**

CREATE PROCEDURE usp\_GetSalesByCategory

AS

BEGIN

SELECT c.CategoryID, c.CategoryName,

ROUND(SUM(od.unit\_price \* od.quantity - (od.unit\_price \* od.quantity \* od.discount)), 2) AS 'Total Sale'

FROM Categories c

JOIN Products p ON P.CategoryID = c.CategoryID

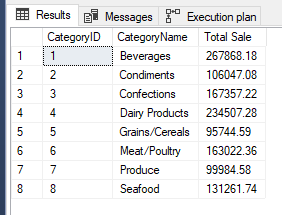
JOIN OrderDetails od ON od.product\_id = p.ProductID

GROUP BY c.CategoryID, c.CategoryName

ORDER BY c.CategoryName

END

EXECUTE usp\_GetSalesByCategory



**5)**

CREATE PROCEDURE usp\_Get10MostExpensiveProducts

AS

BEGIN

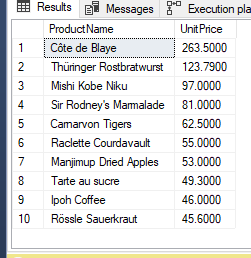
SELECT TOP 10 ProductName, UnitPrice

FROM Products

ORDER BY UnitPrice DESC

END

EXECUTE usp\_Get10MostExpensiveProducts



**6)**

CREATE PROCEDURE usp\_InsertOrderDetails (

@order\_id smallint,

@product\_id int,

@unit\_price real,

@quantity smallint,

@discount real

)

AS

BEGIN

IF EXISTS (SELECT order\_id FROM OrderDetails WHERE order\_id = @order\_id AND product\_id = @product\_id)

BEGIN

RAISERROR (N'Record already exist with order id %d and product id %d...!', 11, 1, @order\_id, @product\_id)

RETURN

END

IF NOT EXISTS (SELECT order\_id FROM Orders WHERE order\_id = @order\_id)

BEGIN

RAISERROR (N'Record does not exist with order id %d...!', 11, 2, @order\_id)

RETURN

END

IF NOT EXISTS (SELECT ProductID FROM Products WHERE ProductID = @product\_id)

BEGIN

RAISERROR (N'Record does not exist with product id %d...!', 11, 3, @product\_id)

RETURN

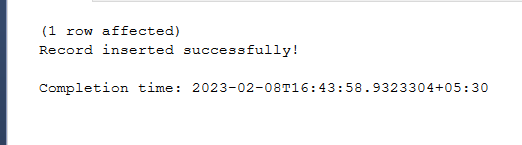
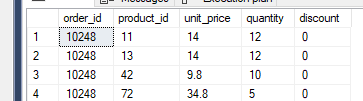
END

INSERT INTO OrderDetails VALUES (@order\_id, @product\_id, @unit\_price, @quantity, @discount)

PRINT('Record inserted successfully!')

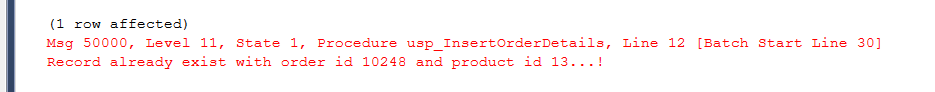
END

EXECUTE usp\_InsertOrderDetails 10248, 13, 14, 12, 0

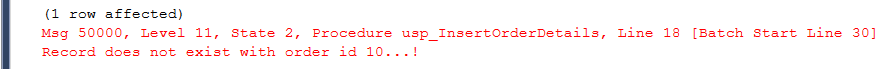
 

**Validations:-**

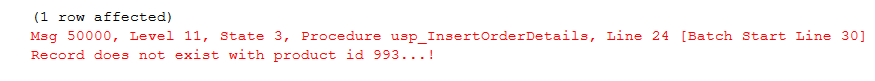
**If entry already exists in order details table**



**If order id doesn’t exist in order table**

****

**If product id doesn’t exist in product table**



**7)**

CREATE PROCEDURE usp\_UpdateOrderDetails (

@order\_id smallint,

@product\_id int,

@column\_name varchar(20),

@new\_value real

)

AS

BEGIN

IF NOT EXISTS (SELECT order\_id FROM OrderDetails WHERE order\_id = @order\_id AND product\_id = @product\_id)

BEGIN

RAISERROR (N'Record does not exist having order id %d or product id %d...!', 11, 1, @order\_id, @product\_id)

RETURN

END

DECLARE @query nvarchar(1000)

SET @query = 'UPDATE OrderDetails

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))

PRINT(@QUERY)

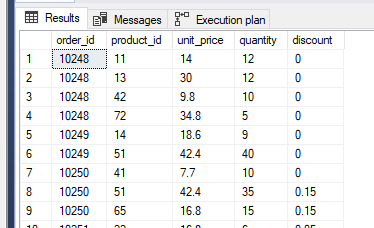
EXEC sp\_executesql @query;

PRINT('Record updated successfully!')

END

EXECUTE usp\_UpdateOrderDetails 10248, 13, 'discount', 0.10

**Before executing query:-**



**After running query:-**

