--The Project\_I made by Kishan Thakkar.

**Transaction exercise 1: Ordering a item**

**Write Transact-SQL code that places an order on an in-store item and verifies that the requested quantity is available.**

go

create trigger Checks on orders for insert as

declare @qty int

declare @stock int

declare @id\_article int

begin

print 'The system is Checking on the request! Please Wait! '

select @id\_article = id\_article from inserted

select @qty = qty from inserted

select @stock = stock from article where @id\_article = id\_article

if (@stock - @qty > 0 )

begin

update article set stock = (stock - @qty) where id\_article = @id\_article

print ' The order you have asked for is in Stock and It will placed soon! Thank You! '

end

else if (@stock - @qty < 0)

begin

ROLLBACK TRANSACTION

print ' We are very Sorry!!!We do not have the enough stock!!'

end

end

select \* from article

select \* from orders

insert into orders (id\_article, qty) values(1, 2)

**Transaction exercise 2: Airport reservations**

**Write Transact-SQL code that reserves 3 seats on flight AF714. The a TRANSACTION.\**

go

create trigger ticket on T\_RESERVATION for insert as

declare @vol\_ref varchar(50)

declare @cli\_id varchar(10)

declare @places int

declare @seats int

begin

print 'searching for the seat availability!!'

select @cli\_id = CLI\_ID from inserted

select @vol\_ref = vol\_ref from inserted

select @places = places from inserted

select @seats = VOL\_PLACES\_LIBRES from T\_VOL\_AVION where vol\_ref=@vol\_ref

if (@seats - @places > 0 )

begin

update T\_VOL\_AVION set VOL\_PLACES\_LIBRES= (VOL\_PLACES\_LIBRES- @places) where vol\_ref = @vol\_ref

print 'Congratulations! The reservation for your seats is confirmed! Have a safe journey!'

end

else if (@iseats- @places < 0)

begin

ROLLBACK TRANSACTION

print 'We are so Sorry..! There is not enough seats for your reservations.'

end

end

select \* from T\_VOL\_AVION

select \* from T\_RESERVATION

insert into T\_RESERVATION (cli\_id,vol\_ref,places) values(1,'AF714',3)

# Question 3) Procedure

1. Write the stored procedure “**StoreSales**” that displays the average sales, the average amount per order, the average of these amounts for a store provided as a parameter.

Execution Example :

### EXEC StoreSales '7067'

**Total of order from store 7067 :**

**OrderNumber total**

**-------------------- --------------------- D4482 109,50**

**P2121 1376,80**

**Average amount of orders from the store 7067: $ 743.15**

1. Modify the stored procedure, to make the store number optional. If the user does not provide a store number, display the average of all store sales combined. If the store number does not exist, display a suitable message.
2. Write another version of the procedure that returns the average value instead of displaying it with a PRINT or a SELECT. Use a return value in case the store number does not exist.

IF EXISTS (SELECT \* FROM sysobjects WHERE name='StoreSales')

DROP PROC StoreSales

GO

CREATE PROCEDURE StoreSales @stor\_id char(5) = NULL AS

declare @avg\_price float

declare @nblinesfound int

select sa.ord\_num , avg (price\*qty) as total into #temp from sales sa Join titles t on sa.title\_id=t.title\_id where sa.stor\_id=@stor\_id group by sa.ord\_num

print 'The Total number of orders from the store' + @stor\_id +':'

Select te.ord\_num as order\_number , avg(total) as Total from #temp te join sales sa on te.ord\_num=sa.ord\_num where sa.stor\_id=@stor\_id group by te.ord\_num

Select @avg\_price= avg(total) from #temp

print' The total Average amount of orders from the store 7067 are as below: '+ convert(varchar,@avg\_price)

--b)

if @stor\_id is null

begin

print ('The all stores sales combined average is:')

select avg(price\*qty) as 'Average' from sales sa join titles te on sa.title\_id = te.title\_id join stores st on sa.stor\_id=st.stor\_id

return

end

select @nblinesfound= count(\*)

from stores where stor\_id = @stor\_id

if @nbLinesFound=0

begin

print 'This ' + @stor\_id +' store\_id does not consist any store in the database.'

return

end

EXEC StoreSales '7067'

IF EXISTS (SELECT \* FROM sysobjects WHERE name='StoreSales')

DROP PROC StoreSales

GO

--c)

CREATE PROCEDURE StoreSales2 @stor\_id char(5) = NULL AS

declare @avg\_price float

begin

print ('The all stores sales combined average is:')

select @avg\_price = 'average' select @avg\_price = avg(t.price\*t.qty) as 'Average' from sales sa join titles t on sa.title\_id = t.title\_id join stores st on sa.stor\_id=st.stor\_id

returns average

end

# Question 4) View

Create a view showing the sales per year of each publisher, the related year, and each publisher's respective total sales (across all years).

create view publishers\_view AS

select p.pub\_name as 'Publisher\_Name', year(t.pubdate) as 'Year\_Related', avg(t.ytd\_sales) as 'Average\_Sales\_Per\_Year' , sum(t.ytd\_sales) as 'Respective\_Total\_Sales'

From publishers p JOIN titles t

ON p.pub\_id = t.pub\_id Group by p.pub\_name, year(t.pubdate)

select \* from publishers\_view