-- ERROR HANDLING IN SQL/ WE WILL CREATE A DATABASE THAT WILL UPDATE THE STOCK OF AN ONLINE STORE

SELECT \* FROM tblProduct

SELECT \* FROM tblProductSales

insert into tblProductSales values(2, 1, 10)

--LET'S CREATE A STORE PROCEDOURE THAT WILL FIRST CHECK THE STOCK AVAILABLE

ALTER PROC spSellProduct

@UserProductId int,

@QuantityToSell int

AS

BEGIN

--SO HERE WE WILL DECLARE ANOTHER VARIABLE IN WHICH WE WILL PASS THE 'QTYAvailable' FROM tblProduct

DECLARE @StockAvailable int

SELECT @StockAvailable = QtyAvailable from tblProduct where ProductID = @UserProductId

--NOW WE HAVE TO CHECK IF THE QUANTITY DEMANDED IS AVAILABE IN THE STOCK, IF NOT TELL IT BY RAISING AN ERROR

if(@StockAvailable< @QuantityToSell)

begin

raiserror('The stock available is not enough for the quantity demanded', 16 ,1) --ERROR 16 MEANS THE CUSTOMER CAN SOLVE IT

end

--OTHERWISE,IF THE STOCK WE HAVE is ENOUGH TO SELL FIRST 'update tblProduct' AND SUBSTRACT 'QTYDEMANDED'FROM 'QTY TO SELL'

else

begin

BEGIN TRY

begin transaction

update tblProduct set QtyAvailable = (QtyAvailable - @QuantityToSell)

where ProductID = @UserProductId

--end I THIN THIS IS WHERE I MADE THE MISTAKE

-- NOW WEED NEED TO PUT SIGNIFY THE NUMBER OF PRODUCTS SOLD INTO THE 'tblProductSales' TO LET PEOPLE KNOW HOW

-- MANY PRODUCT WERE SOLD ANY TIME A TRANSACTION IS MADE.

--BUT ProducSalesID IS NOT AN IDENTITY COLULMN SO WE WILL HAVE TO ENTER THE VALUES MANUALLY

--SO HERE WE ARE MANUALLY COMPUTIING THE VALUE OF THE PRIMAKEY COLUMN

DECLARE @MAXProducSalesID int

SELECT @MAXProducSalesID = CASE WHEN

--MAX (@MAXProducSalesID) IS NULL -- here is the mistake

MAX(ProductSalesID) IS NULL -- where are rother looking

THEN 0 ELSE MAX (ProductSalesID) end -- for the maximum value of the 'ProductSales' Column

-- NOW WE INCREAMENT THE VALUE by 1 to avoid Primary Key violation AND INSERT ALL IN tblProductSales

FROM tblProductSales

--SET @MAXProducSalesID = @MAXProducSalesID + 1

INSERT INTO tblProductSales VALUES ( @MAXProducSalesID, @UserProductId, @QuantityToSell)

commit tran

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

SELECT

ERROR\_NUMBER() as ErrorNumber, --tell us where the error happened

ERROR\_MESSAGE() as ErroMessage,

ERROR\_PROCEDURE() as ErrorProcedure,

ERROR\_SEVERITY() as ErrorSevierity,

ERROR\_LINE() as ErrorLine,

ERROR\_STATE() as ErrorState

END CATCH

END

END

--NOW LET'S TEST OUR STORE PROCEDURE ENTERING VALUES IN ITS PARAMETERS

EXECUTE spSellProduct 1, 10 -- no parentheses here/ so we want to buy 10 laptops / let's check what happe

