ASSIGNMENT-02(WEEK-2)

ANIKET SAROJ

Bennett University

CT\_CSI\_SQ\_4791

Database Used: AdventureWorks2022

1. CREATE PROCEDURE INSERTINTODETAILS

@ORDERID INT,

@PRODUCTID INT,

@LISTPRICE MONEY = NULL,

@ORDERQTY SMALLINT,

@UNITPRICEDISCOUNT REAL = 0

AS

BEGIN

DECLARE @REORDERPOINT SMALLINT;

DECLARE @STOCKLEVEL SMALLINT;

DECLARE @ACTUALPRICE MONEY;

BEGIN TRANSACTION;

SELECT

@ACTUALPRICE = ListPrice,

@STOCKLEVEL = SafetyStockLevel,

@REORDERPOINT = ReorderPoint

FROM PRODUCTION.Product

WHERE ProductID = @PRODUCTID;

IF @ACTUALPRICE IS NULL

BEGIN

PRINT 'PRODUCT MISSING';

ROLLBACK TRANSACTION;

RETURN;

END

IF @LISTPRICE IS NULL

SET @LISTPRICE = @ACTUALPRICE;

IF @ORDERQTY > @STOCKLEVEL

BEGIN

PRINT 'NUMBER OF STOCKS ARE NOT AVAILABLE';

ROLLBACK TRANSACTION;

RETURN;

END

INSERT INTO SALES.SalesOrderDetails (OrderID, ProductID, UnitPrice, OrderQty, UnitPriceDiscount)

VALUES (@ORDERID, @PRODUCTID, @LISTPRICE, @ORDERQTY, @UNITPRICEDISCOUNT);

IF @@ROWCOUNT = 0

BEGIN

PRINT 'ORDER NOT PLACED. TRY AGAIN';

ROLLBACK TRANSACTION;

RETURN;

END

UPDATE PRODUCTION.Product

SET SafetyStockLevel = SafetyStockLevel - @ORDERQTY

WHERE ProductID = @PRODUCTID

IF (@STOCKLEVEL - @ORDERQTY) < @REORDERPOINT

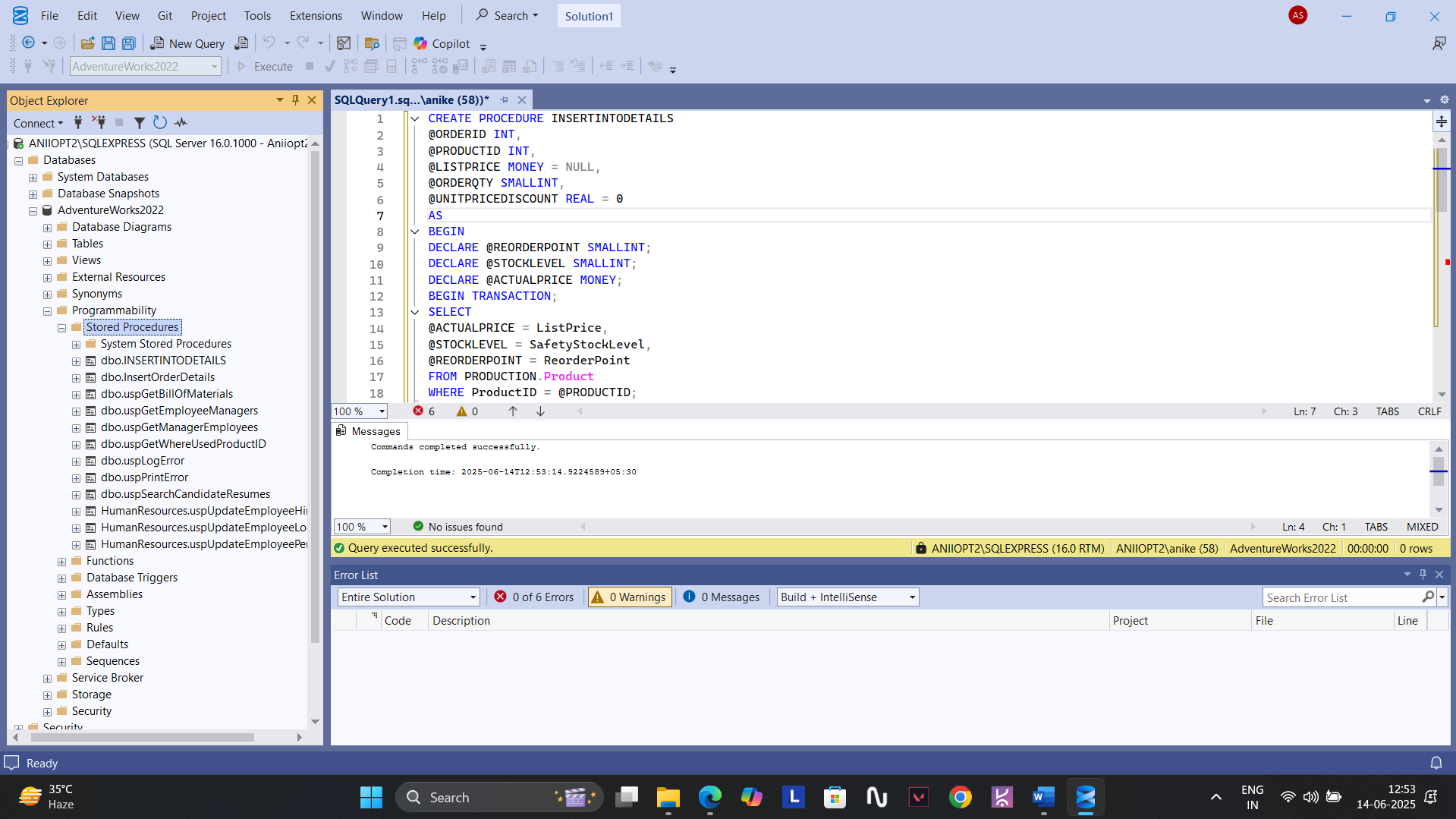
BEGIN

PRINT 'QUANTITY OF THE STOCKS IS BELOW THE REORDER LEVEL.';

END

COMMIT TRANSACTION;

END;



2.

CREATE PROC UPDATEORDERDETAILS

@ORDERID INT,

@PRODUCTID INT,

@LISTPRICE MONEY=NULL,

@ORDERQTY SMALLINT = NULL,

@UNITPRICEDISCOUNT REAL=NULL

AS

BEGIN

DECLARE @OLDQTY SMALLINT;

DECLARE @OLDPRICE MONEY;

DECLARE @OLDDISCOUNT REAL;

DECLARE @NEWQTY SMALLINT;

DECLARE @AVAILABLESTOCKS SMALLINT;

DECLARE @REORDERPOINT SMALLINT;

BEGIN TRY

BEGIN TRANSACTION;

SELECT

@OLDQTY=ORDERQTY,

@OLDPRICE=UnitPrice,

@OLDDISCOUNT=UNITPRICEDISCOUNT

FROM SALES.SalesOrderDetail

WHERE SALESORDERID=@ORDERID AND PRODUCTID=@PRODUCTID;

IF @OLDQTY IS NULL

BEGIN

PRINT 'ORDER DETAIL IS MISSING.';

ROLLBACK TRANSACTION

RETURN;

END

SET @NEWQTY =ISNULL(@ORDERQTY,@OLDQTY);

SET @LISTPRICE=ISNULL(@LISTPRICE,@OLDPRICE);

SET @UNITPRICEDISCOUNT=ISNULL(@UNITPRICEDISCOUNT,@OLDDISCOUNT);

SELECT @AVAILABLESTOCKS=SAFETYSTOCKLEVEL,@REORDERPOINT=REORDERPOINT FROM PRODUCTION.Product WHERE PRODUCTID=@PRODUCTID;

SET @AVAILABLESTOCKS=@AVAILABLESTOCKS+@OLDQTY

IF @NEWQTY>@AVAILABLESTOCKS

BEGIN

PRINT 'NOT ENOUGH STOCKS.';

ROLLBACK TRANSACTION;

RETURN;

END

UPDATE SALES.SalesOrderDetail

SET

UnitPrice=@LISTPRICE,

ORDERQTY=@NEWQTY,

UNITPRICEDISCOUNT=@UNITPRICEDISCOUNT

WHERE SALESORDERID=@ORDERID AND PRODUCTID=@PRODUCTID;

UPDATE PRODUCTION.Product

SET SAFETYSTOCKLEVEL=@AVAILABLESTOCKS-@NEWQTY

WHERE PRODUCTID=@PRODUCTID

IF(@AVAILABLESTOCKS-@NEWQTY)<@REORDERPOINT

BEGIN

PRINT 'STOCKS ARE BELOW THE REORDER LEVEL.';

END

COMMIT TRANSACTION;

PRINT 'ORDER UPDATED.';

END TRY

BEGIN CATCH

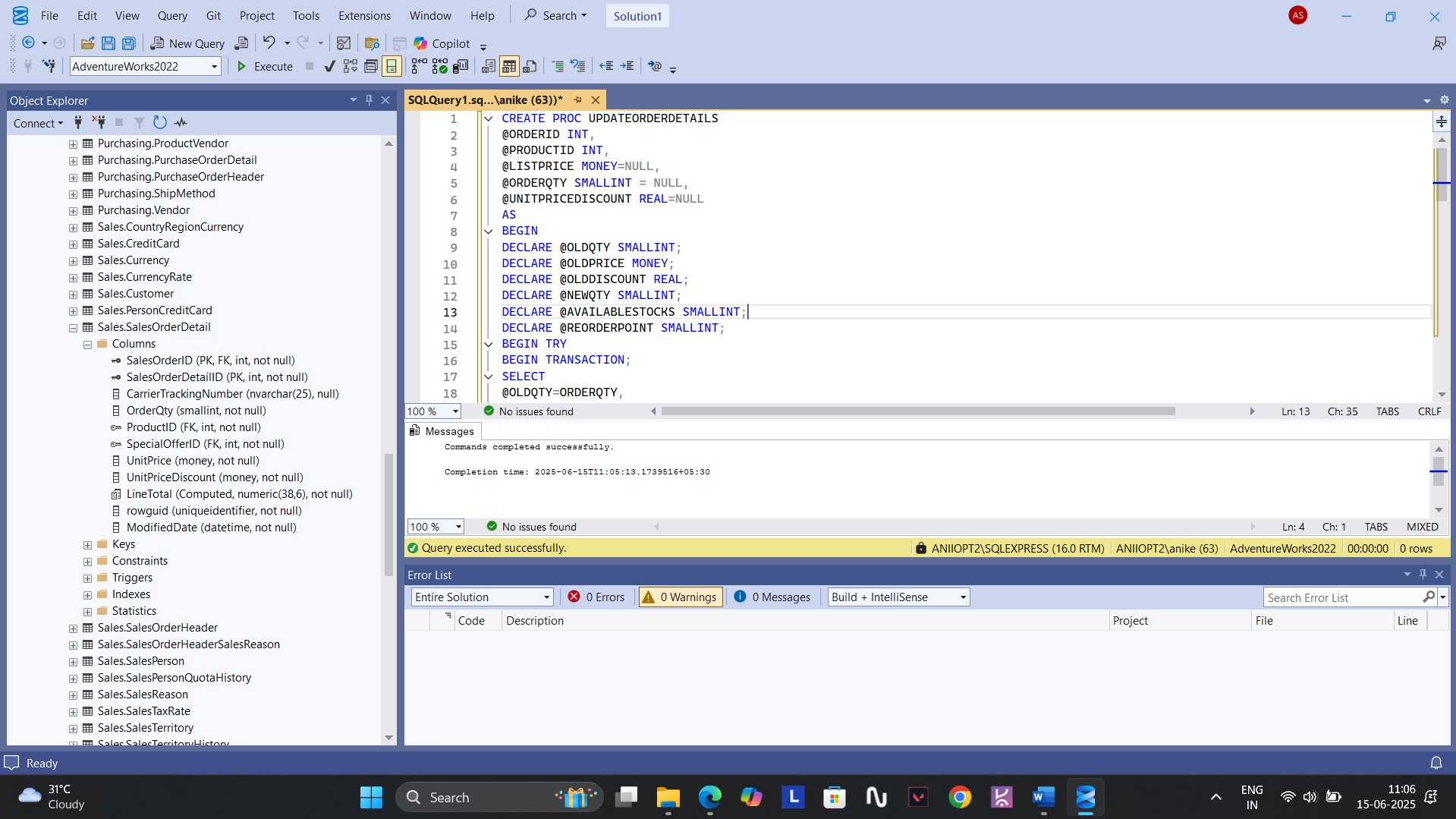
IF @@TRANCOUNT>0

ROLLBACK TRANSACTION

PRINT 'ERROR.';

END CATCH

END;



3. CREATE PROC GETORDERDETAILS

@ORDERID INT

AS

BEGIN

IF NOT EXISTS (

SELECT 1 FROM SALES.SalesOrderDetail

WHERE SALESORDERID=@ORDERID

)

BEGIN

PRINT 'THE ORDERID'+CAST(@ORDERID AS VARCHAR(10))+ 'DOES NOT EXIST';

RETURN 1;

END

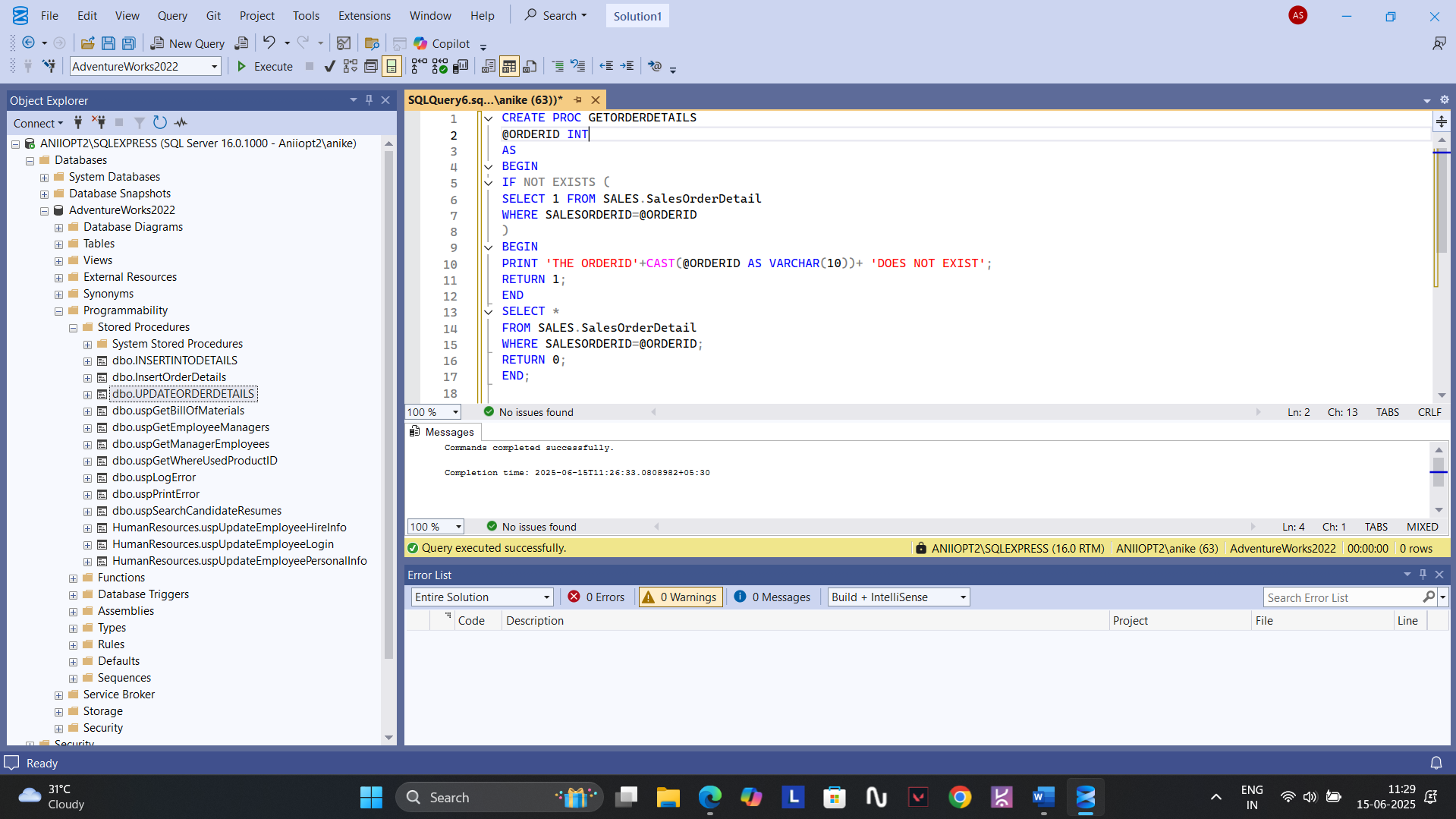
SELECT \*

FROM SALES.SalesOrderDetail

WHERE SALESORDERID=@ORDERID;

RETURN 0;

END;



4. CREATE PROC DELETEORDERDETAILS

@ORDERID INT,

@PRODUCTID INT

AS

BEGIN

IF NOT EXISTS(SELECT 1 FROM SALES.SalesOrderDetail WHERE SALESORDERID=@ORDERID)

BEGIN

PRINT 'ERROR: ORDERID'+ CAST(@ORDERID AS VARCHAR(10))+' DOES NOT EXIST.';

RETURN -1;

END

IF NOT EXISTS(SELECT 1 FROM SALES.SalesOrderDetail WHERE SALESORDERID = @ORDERID AND PRODUCTID = @PRODUCTID)

BEGIN

PRINT 'ERROR: PRODUCTID'+CAST(@PRODUCTID AS VARCHAR(10))+' DOES NOT EXIST IN ORDERID'+CAST(@ORDERID AS VARCHAR(10))+'.';

RETURN -1;

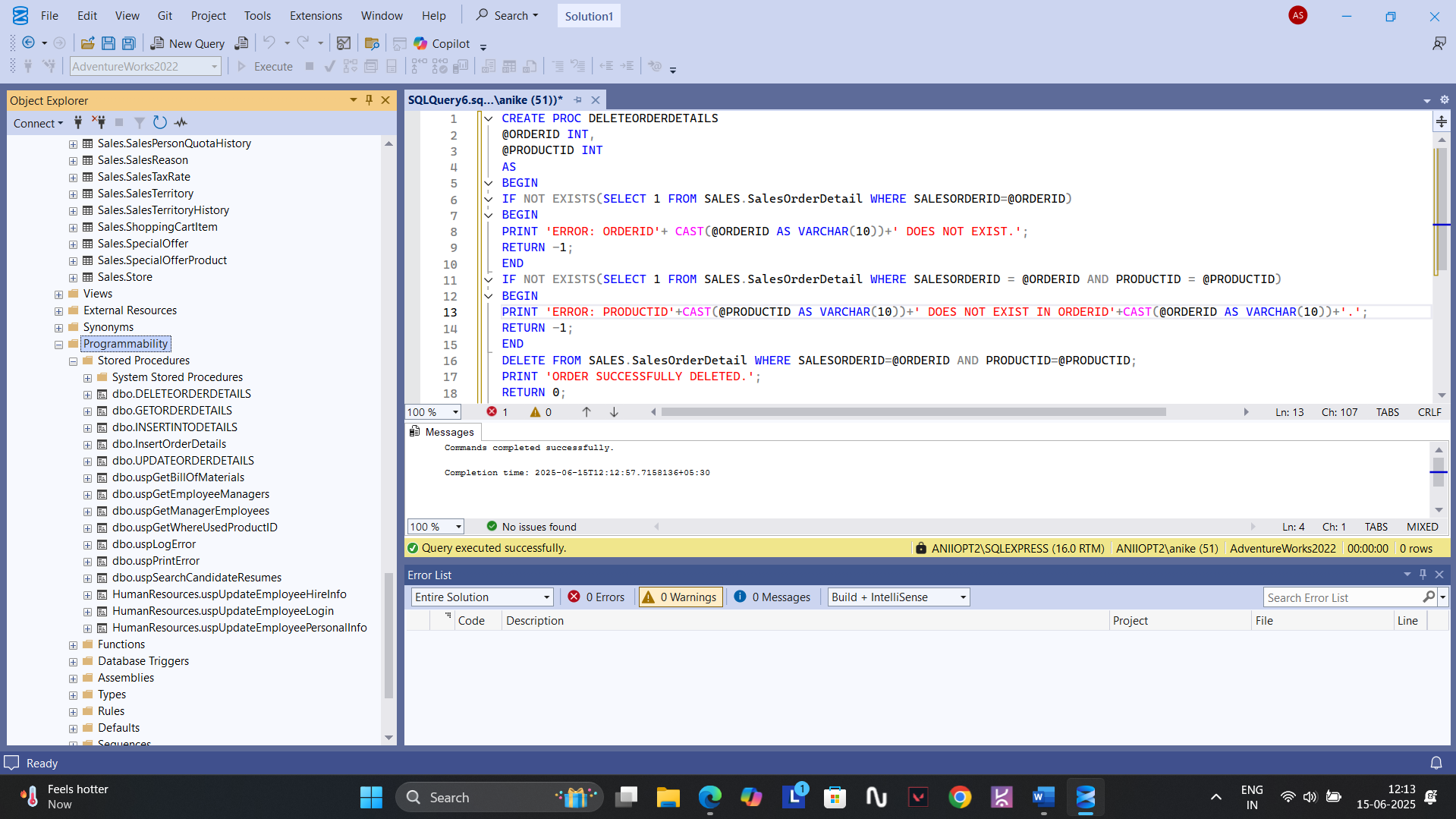
END

DELETE FROM SALES.SalesOrderDetail WHERE SALESORDERID=@ORDERID AND PRODUCTID=@PRODUCTID;

PRINT 'ORDER SUCCESSFULLY DELETED.';

RETURN 0;

END;



FUNCTIONS

1. CREATE FUNCTION DATEFORMAT\_MMDDYYYY(@INPUTDATE DATETIME)

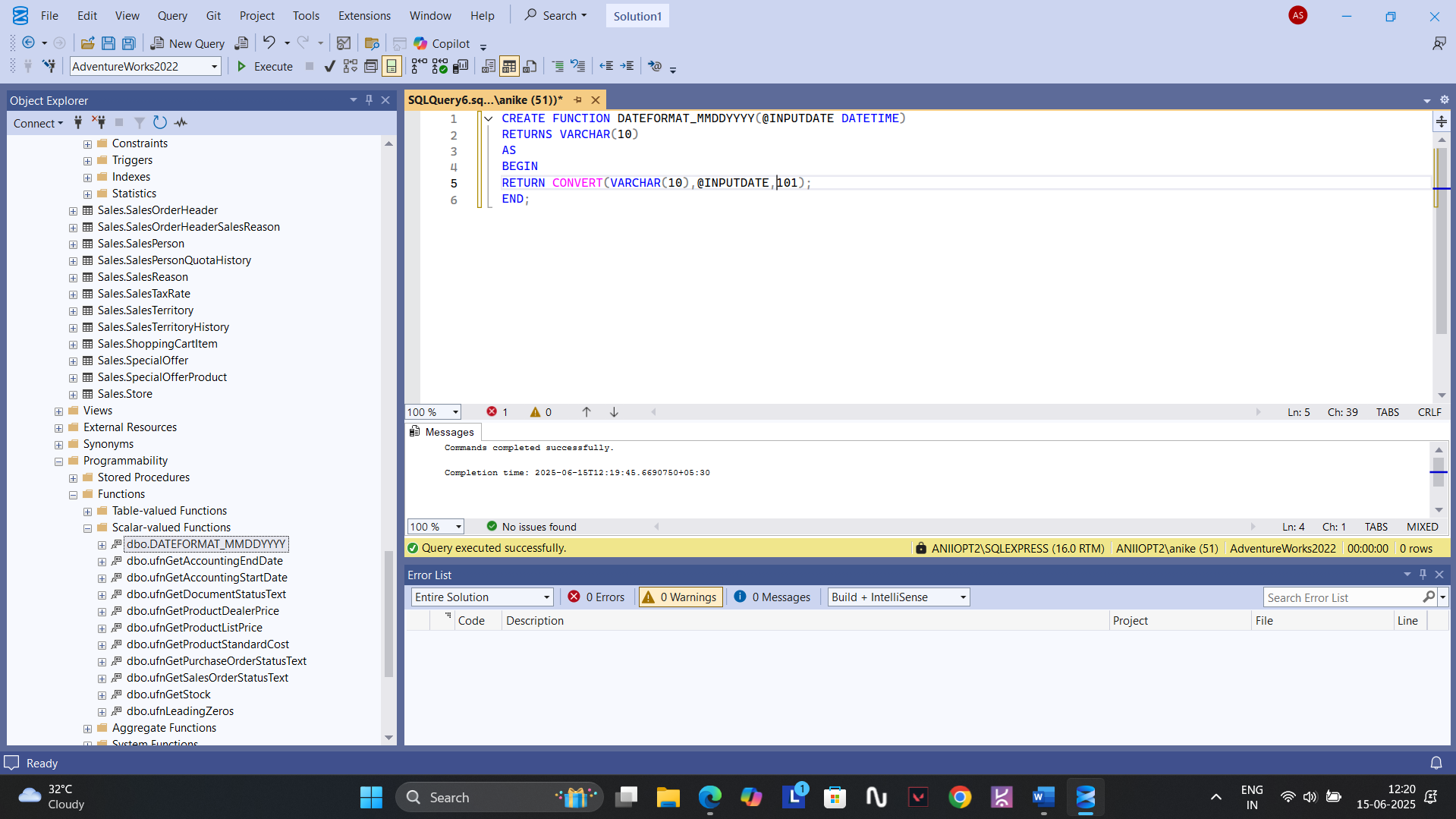
RETURNS VARCHAR(10)

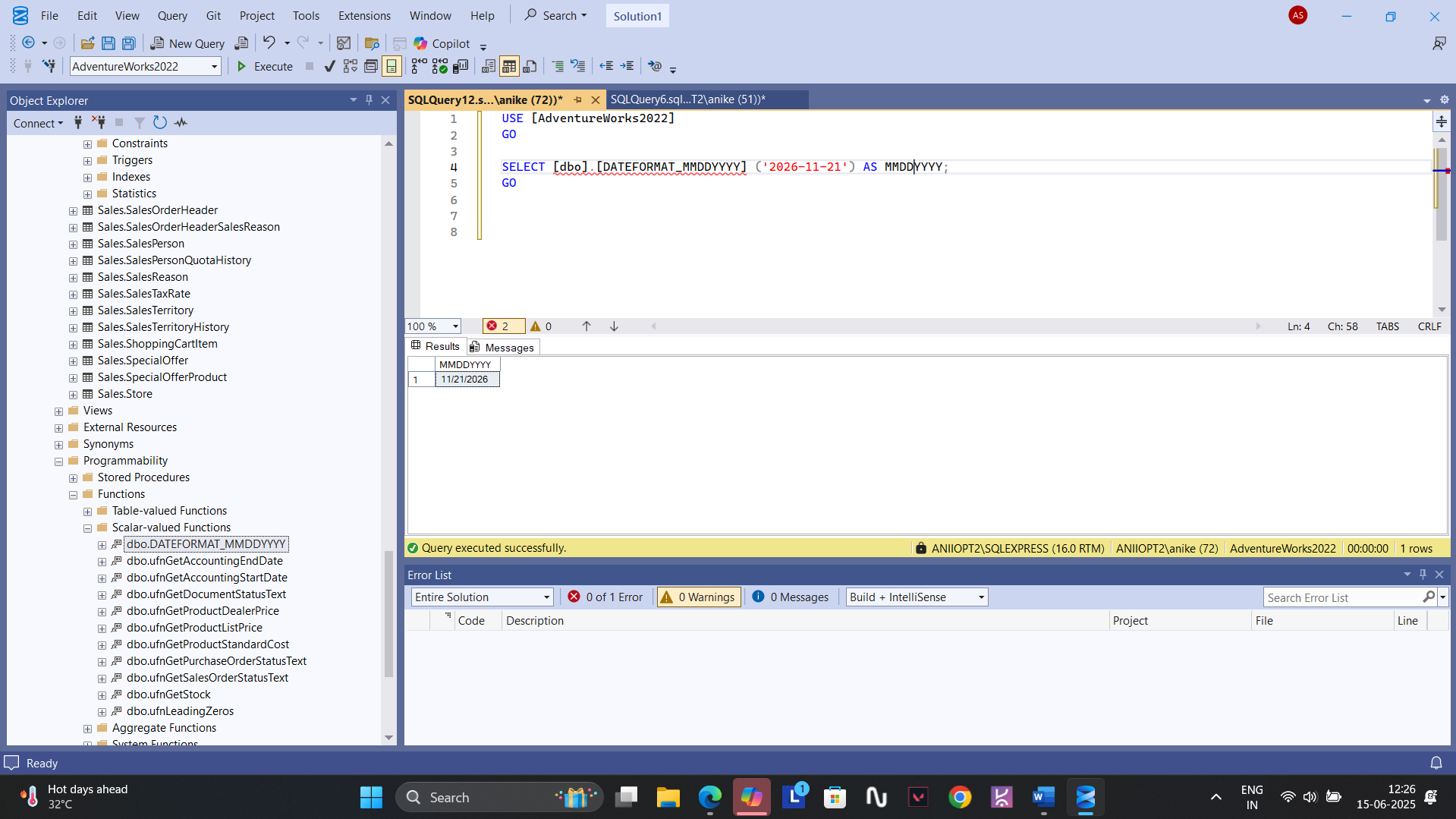
AS

BEGIN

RETURN CONVERT(VARCHAR(10),@INPUTDATE,101);

END;





2. CREATE FUNCTION DATEFORMAT\_YYYYMMDD(@INPUTDATE DATETIME)

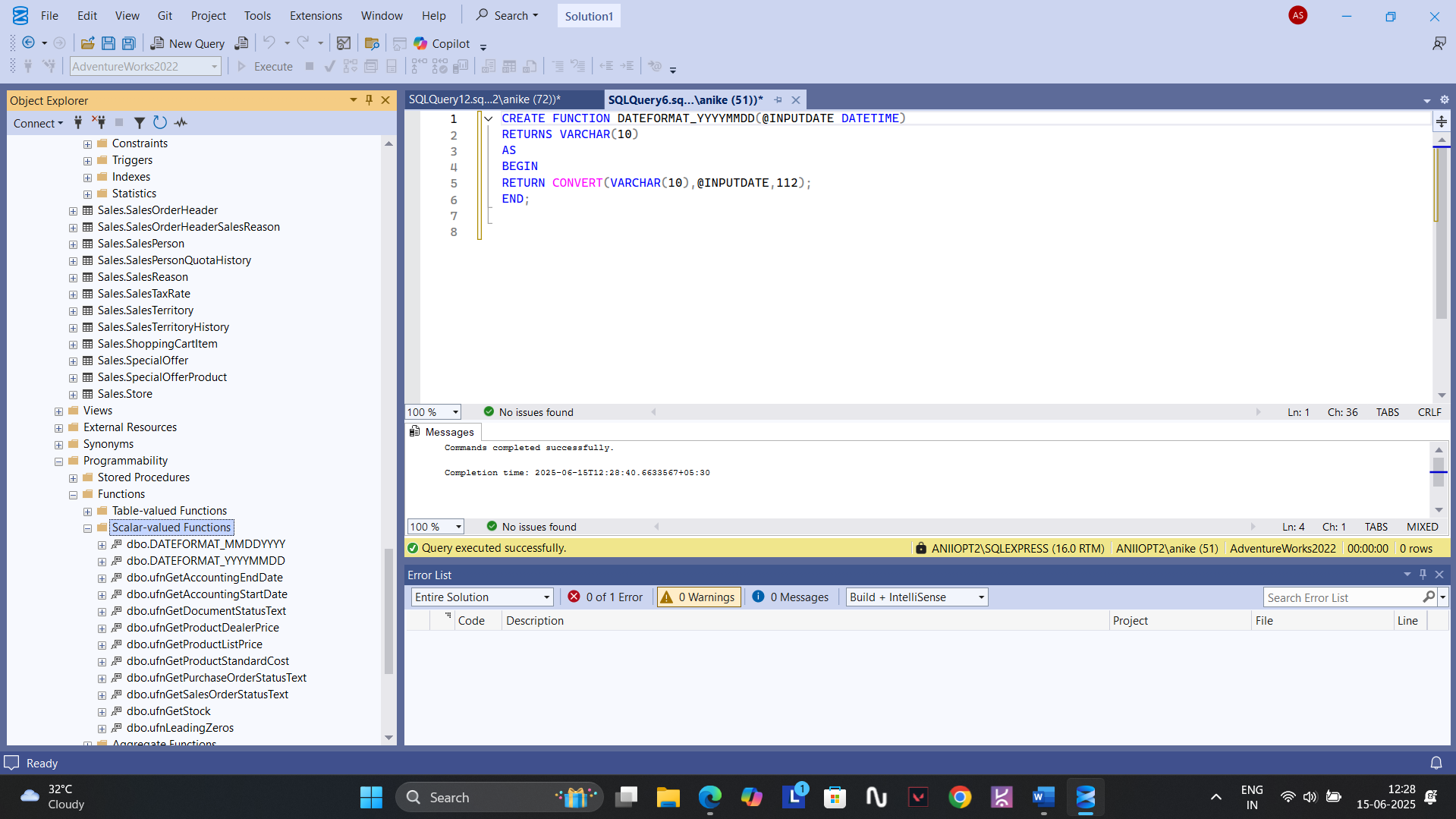
RETURNS VARCHAR(10)

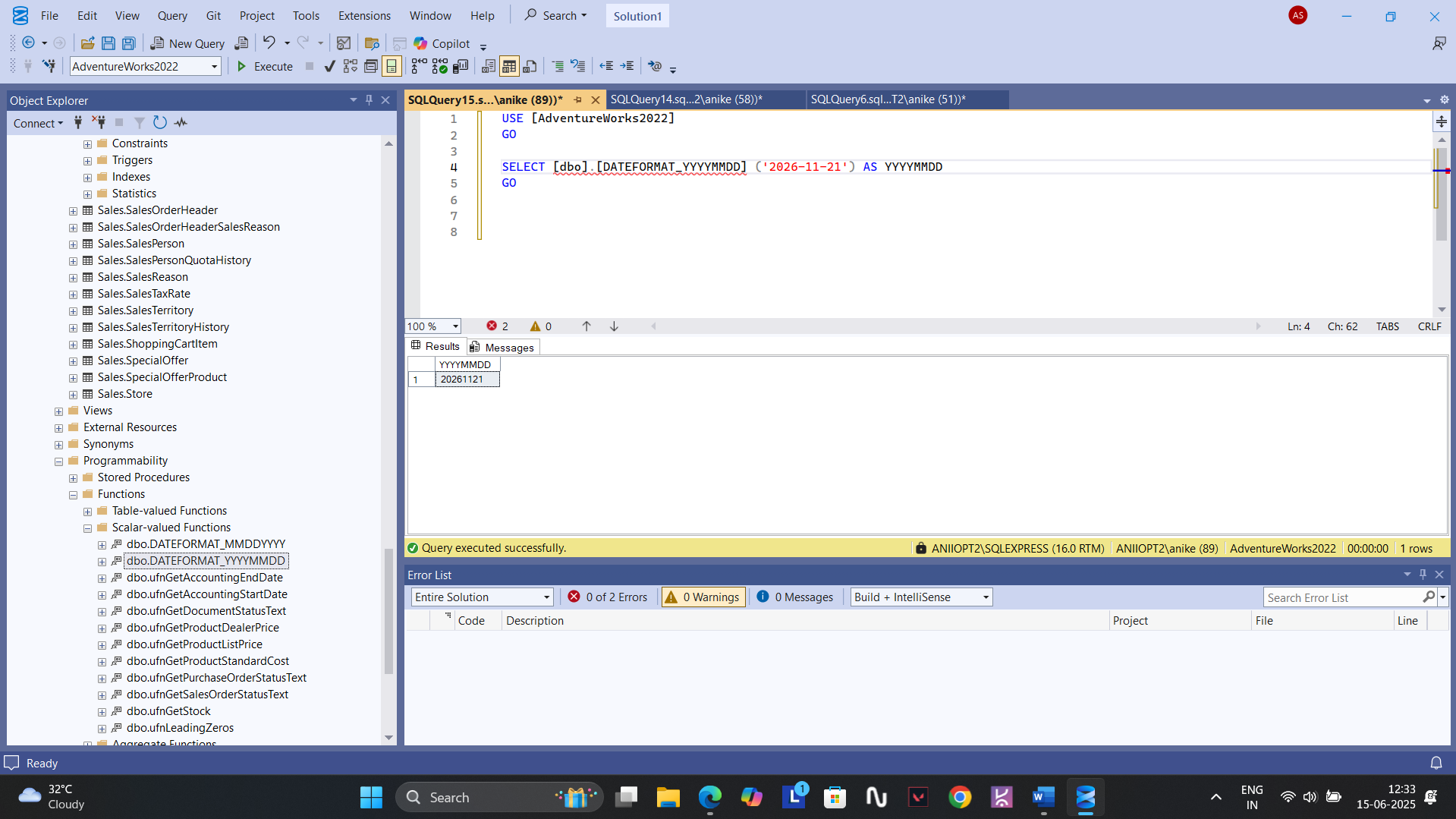
AS

BEGIN

RETURN CONVERT(VARCHAR(10),@INPUTDATE,112);

END;





VIEWS

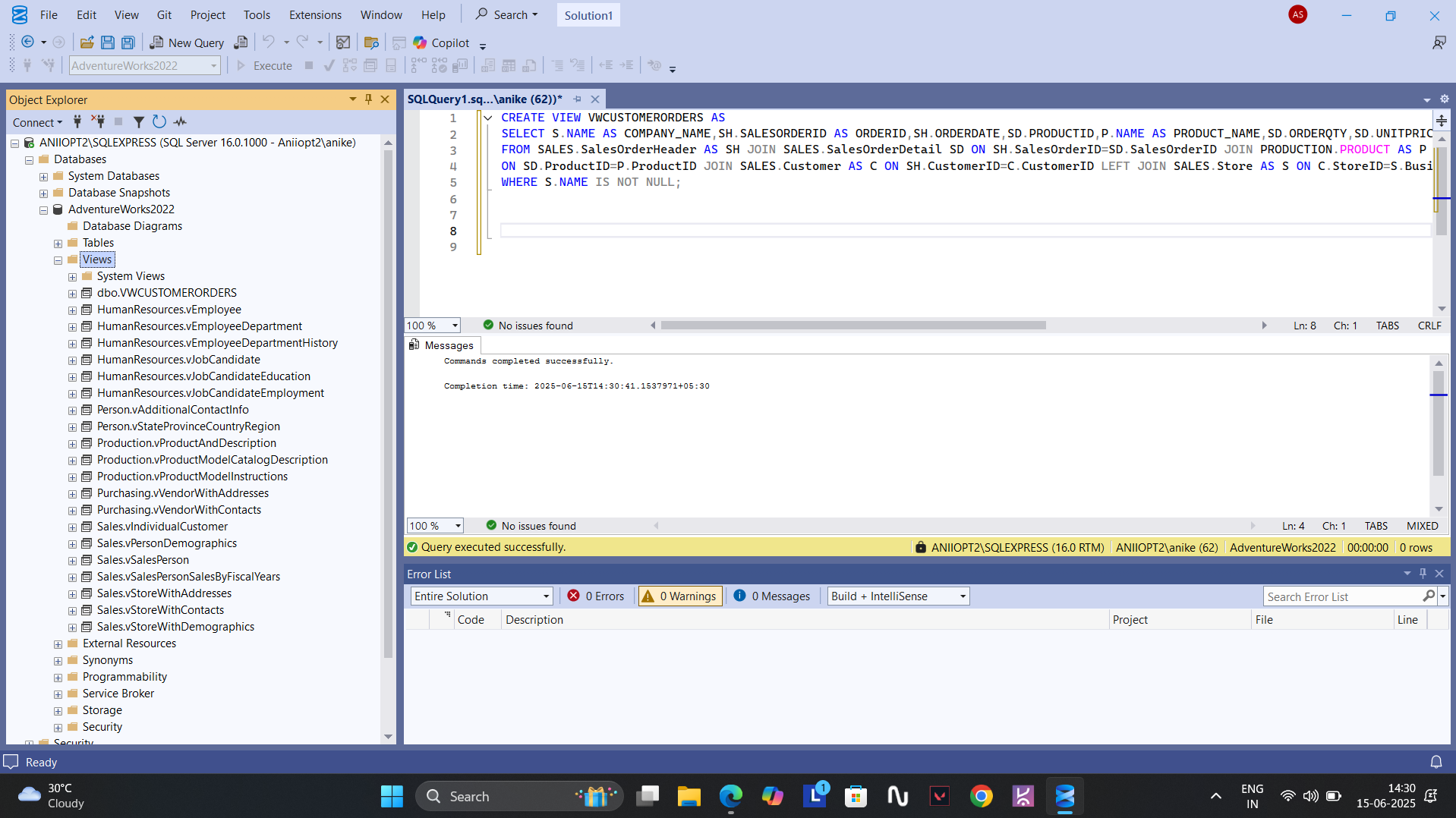
1. CREATE VIEW VWCUSTOMERORDERS AS

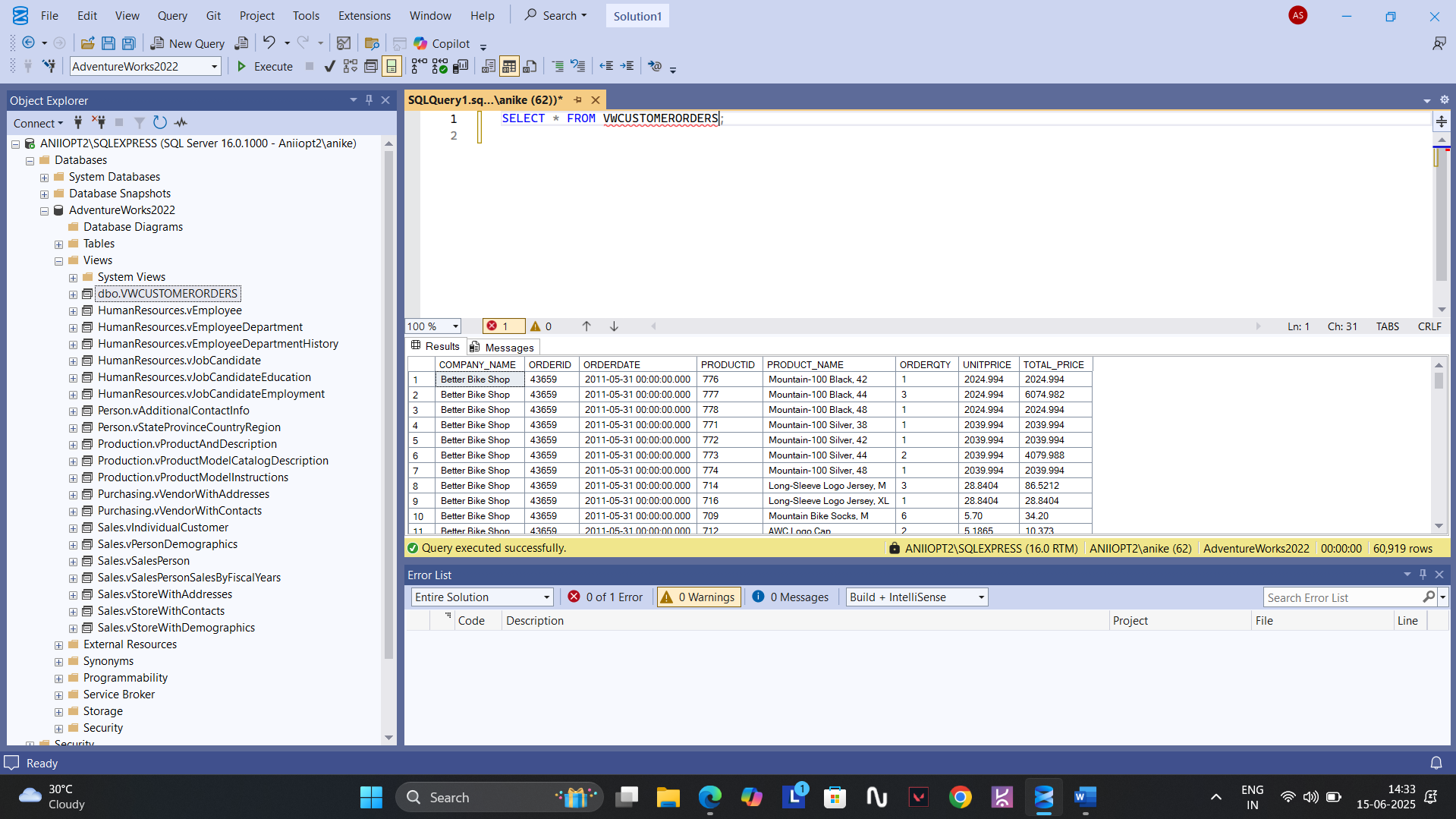
SELECT S.NAME AS COMPANY\_NAME,SH.SALESORDERID AS ORDERID,SH.ORDERDATE,SD.PRODUCTID,P.NAME AS PRODUCT\_NAME,SD.ORDERQTY,SD.UNITPRICE,(SD.ORDERQTY\*SD.UNITPRICE) AS TOTAL\_PRICE

FROM SALES.SalesOrderHeader AS SH JOIN SALES.SalesOrderDetail SD ON SH.SalesOrderID=SD.SalesOrderID JOIN PRODUCTION.PRODUCT AS P

ON SD.ProductID=P.ProductID JOIN SALES.Customer AS C ON SH.CustomerID=C.CustomerID LEFT JOIN SALES.Store AS S ON C.StoreID=S.BusinessEntityID

WHERE S.NAME IS NOT NULL;





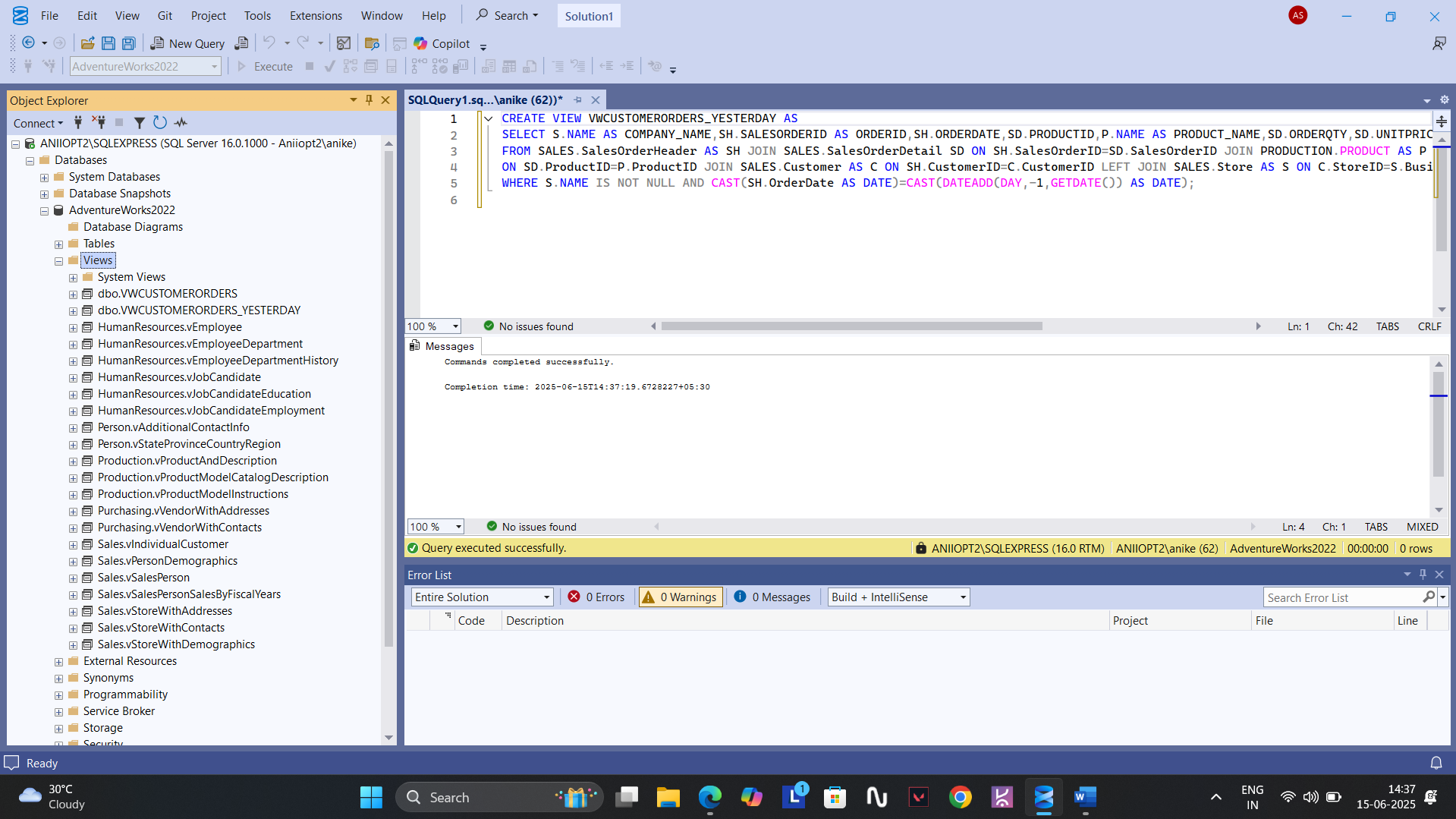
2. CREATE VIEW VWCUSTOMERORDERS\_YESTERDAY AS

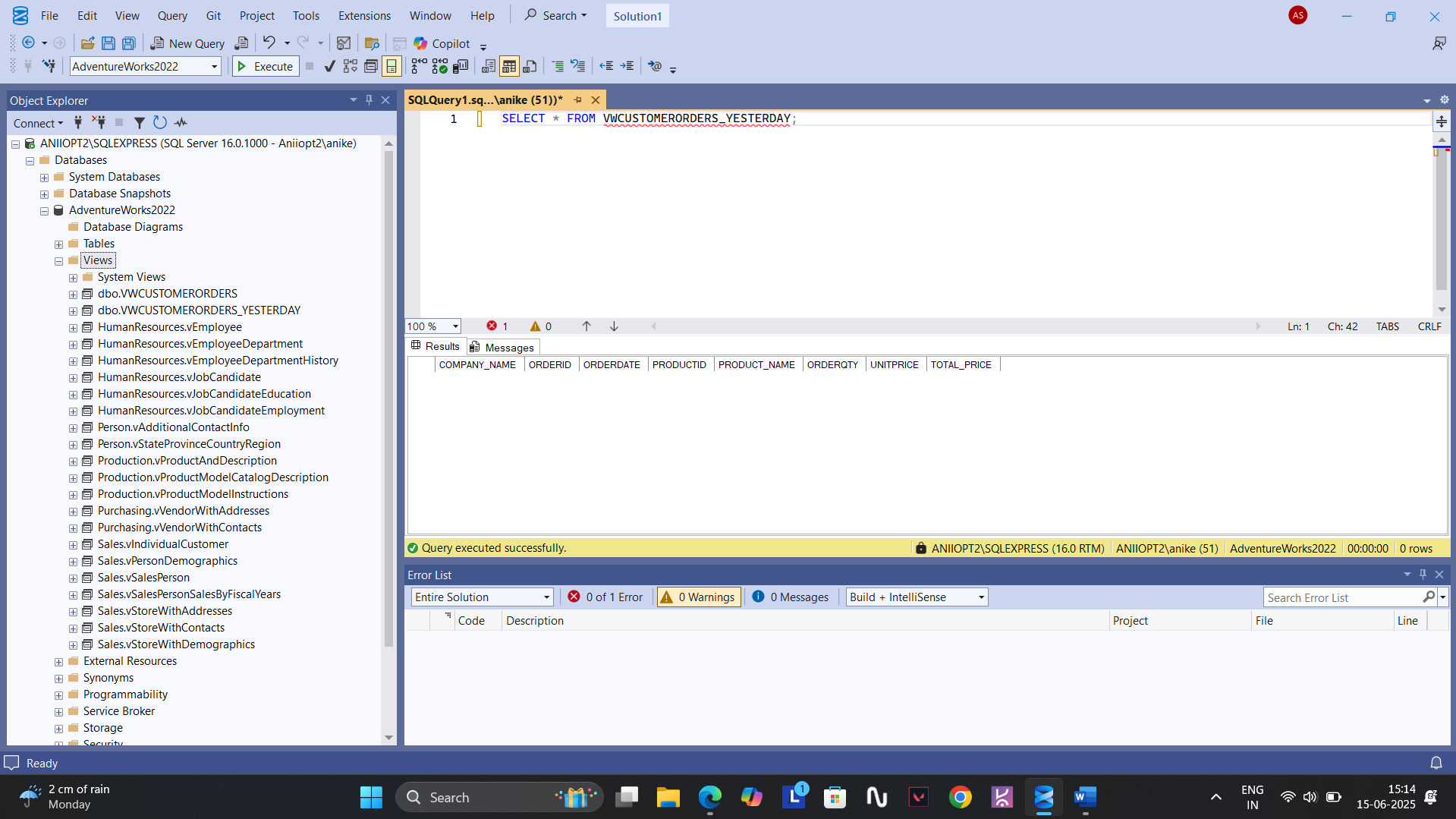
SELECT S.NAME AS COMPANY\_NAME,SH.SALESORDERID AS ORDERID,SH.ORDERDATE,SD.PRODUCTID,P.NAME AS PRODUCT\_NAME,SD.ORDERQTY,SD.UNITPRICE,(SD.ORDERQTY\*SD.UNITPRICE) AS TOTAL\_PRICE

FROM SALES.SalesOrderHeader AS SH JOIN SALES.SalesOrderDetail SD ON SH.SalesOrderID=SD.SalesOrderID JOIN PRODUCTION.PRODUCT AS P

ON SD.ProductID=P.ProductID JOIN SALES.Customer AS C ON SH.CustomerID=C.CustomerID LEFT JOIN SALES.Store AS S ON C.StoreID=S.BusinessEntityID

WHERE S.NAME IS NOT NULL AND CAST(SH.OrderDate AS DATE)=CAST(DATEADD(DAY,-1,GETDATE()) AS DATE);





3. CREATE VIEW MYPRODUCTS AS

SELECT P.PRODUCTID,P.NAME AS PRODUCT\_NAME,P.STANDARDCOST AS UNITPRICE,P.SIZE AS ORDER\_QUANTITY,PC.NAME AS CATEGORY\_NAME

,V.NAME AS COMPANY\_NAME FROM PRODUCTION.Product AS P JOIN PRODUCTION.PRODUCTSUBCATEGORY AS PSG ON P.PRODUCTSUBCATEGORYID=PSG.PRODUCTSUBCATEGORYID

JOIN PRODUCTION.PRODUCTCATEGORY AS PC ON PSG.PRODUCTCATEGORYID=PC.PRODUCTCATEGORYID JOIN PURCHASING.ProductVendor AS PV ON P.PRODUCTID=PV.PRODUCTID

JOIN PURCHASING.VENDOR AS V ON PV.BusinessEntityID=V.BusinessEntityID WHERE P.DiscontinuedDate IS NULL;

