UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

INFORME DE LABORATORIO N3 U3 "UTILIZANDO FUNCIONES DE AGREGACIÓN, OFFSET Y WINDOW RANKING"

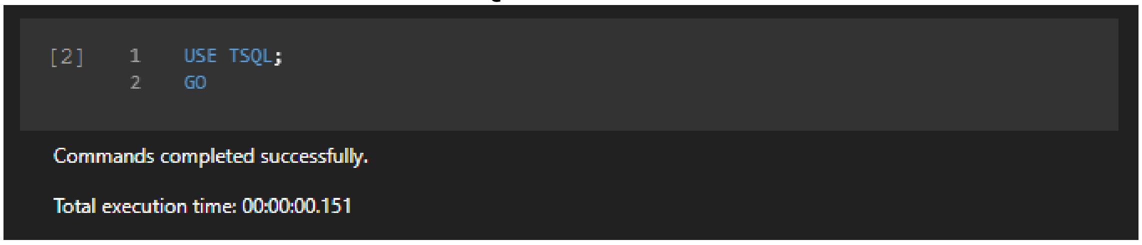
Curso: Base de Datos II

Docente: Ing. Patrick Cuadros

GOMEZ QUIROZ, YUMIN YHULYÑO (2015052385)

Tacna – Perú
2021

1.1 ABRIR LA BASE DE DATOS TSQL:

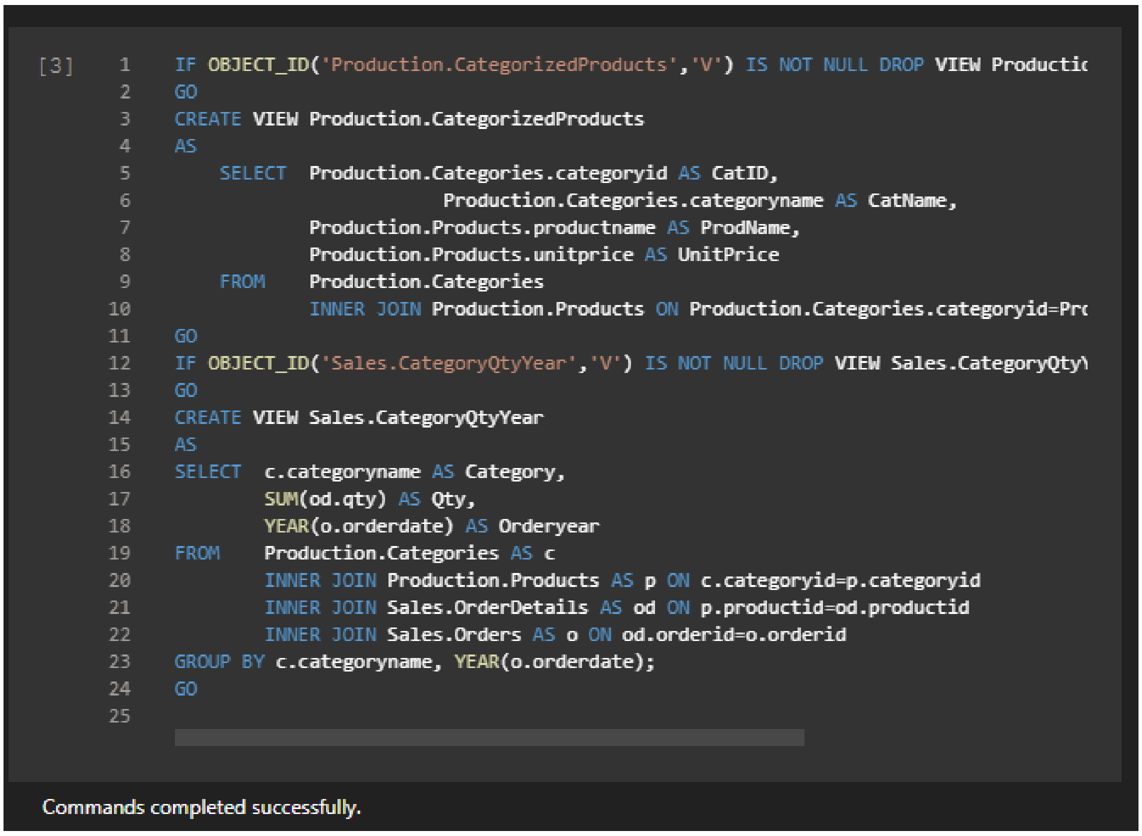


```
[2]
USE TSQL;
GO
```

```
Commands completed successfully.

Total execution time: 00:00:00.151
```

1.2 PARA CREAR UNA VENTANA CON OVER PRIMERO CREAR DOS VISTAS:

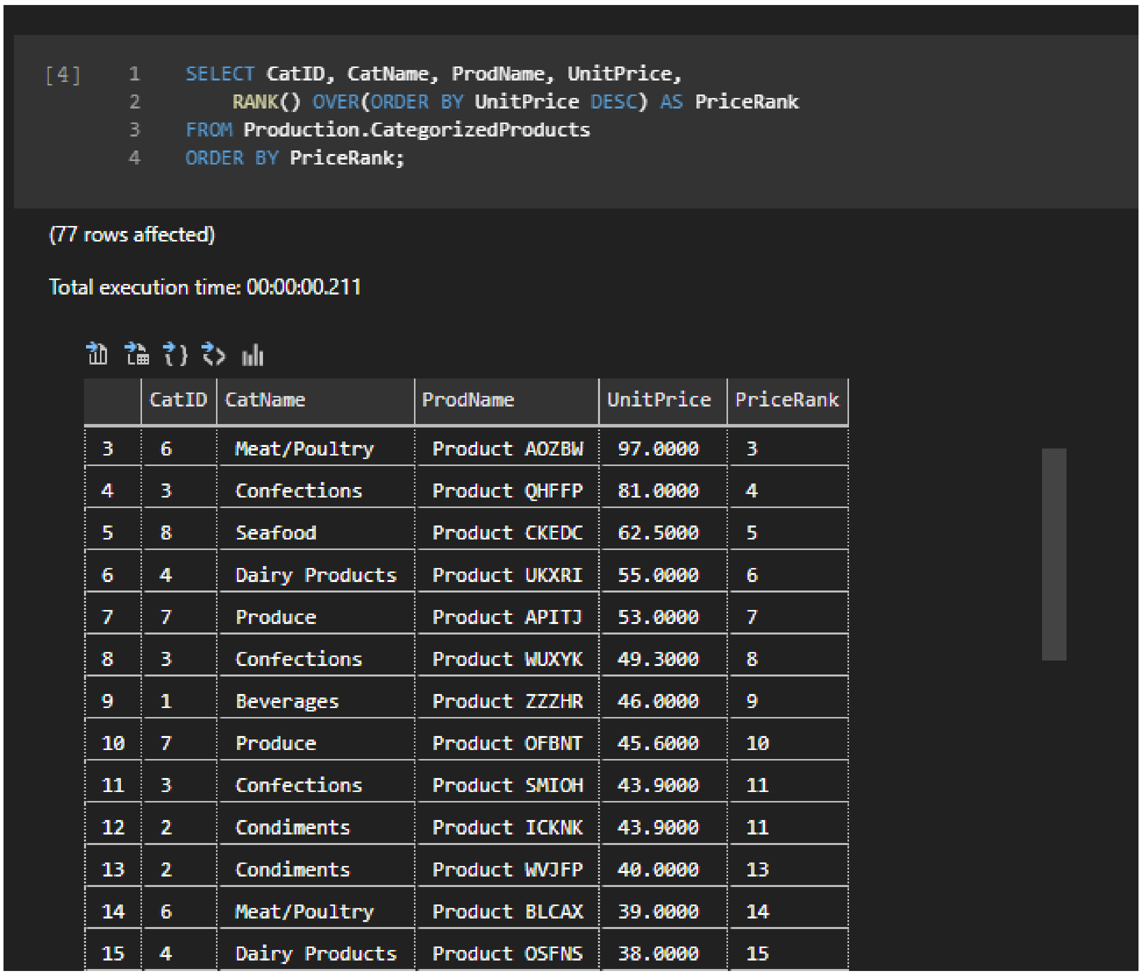


```
[3]
IF OBJECT_ID('Production.CategorizedProducts','V') IS NOT NULL DROP VIEW Productic
GO
CREATE VIEW Production.CategorizedProducts
AS
    SELECT  Production.Categories.categoryid AS CatID,
                  Production.Categories.categoryname AS CatName,
            Production.Products.productname AS ProdName,
            Production.Products.unitprice AS UnitPrice
    FROM    Production.Categories
            INNER JOIN Production.Products ON Production.Categories.categoryid=Prc
GO
IF OBJECT_ID('Sales.CategoryQtyYear','V') IS NOT NULL DROP VIEW Sales.CategoryQty\
GO
CREATE VIEW Sales.CategoryQtyYear
AS
SELECT  c.categoryname AS Category,
        SUM(od.qty) AS Qty,
        YEAR(o.orderdate) AS Orderyear
FROM    Production.Categories AS c
        INNER JOIN Production.Products AS p ON c.categoryid=p.categoryid
        INNER JOIN Sales.OrderDetails AS od ON p.productid=od.productid
        INNER JOIN Sales.Orders AS o ON od.orderid=o.orderid
GROUP BY c.categoryname, YEAR(o.orderdate);
GO

```

```
Commands completed successfully.
```

1.3 UTILIZAR OVER CON ORDERING:



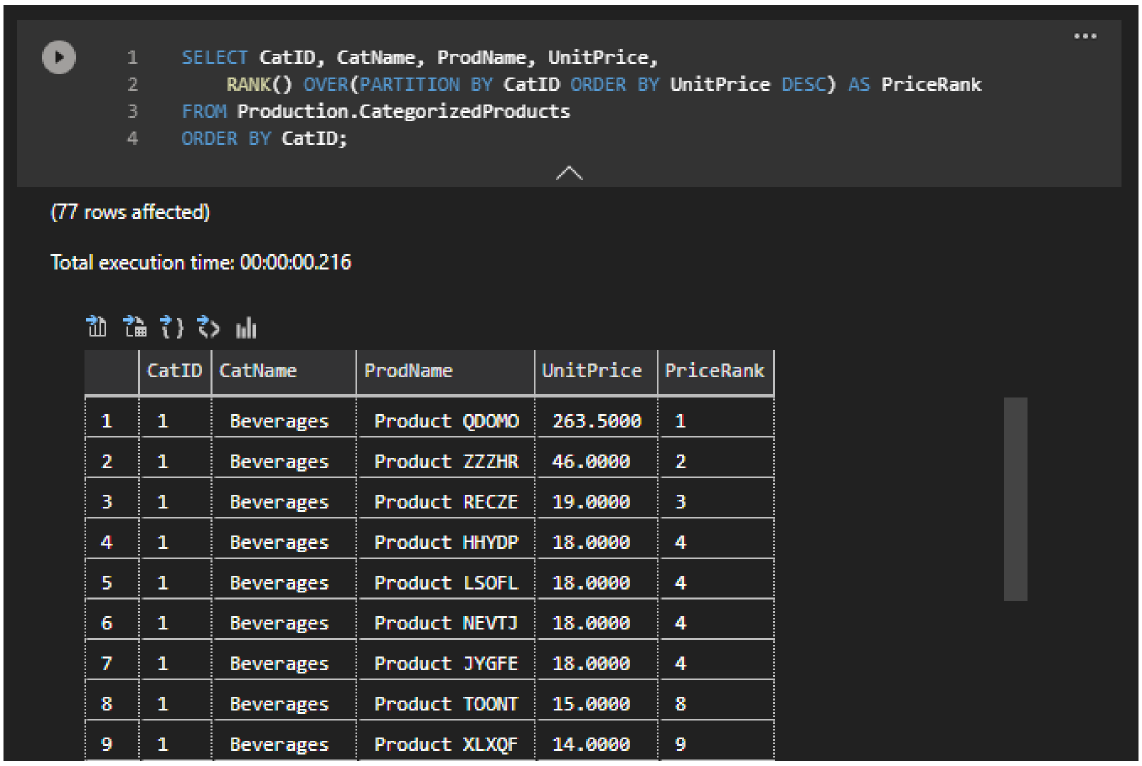
```
[4]  SELECT CatID, CatName, ProdName, UnitPrice,
         RANK() OVER(ORDER BY UnitPrice DESC) AS PriceRank
     FROM Production.CategorizedProducts
     ORDER BY PriceRank;
```

```
(77 rows affected)

Total execution time: 00:00:00.211
```

	CatID	CatName	ProdName	UnitPrice	PriceRank
3	6	Meat/Poultry	Product AOZBW	97.0000	3
4	3	Confections	Product QHFFP	81.0000	4
5	8	Seafood	Product CKEDC	62.5000	5
6	4	Dairy Products	Product UKXRI	55.0000	6
7	7	Produce	Product APITJ	53.0000	7
8	3	Confections	Product WUXYK	49.3000	8
9	1	Beverages	Product ZZZHR	46.0000	9
10	7	Produce	Product OFBNT	45.6000	10
11	3	Confections	Product SMIOH	43.9000	11
12	2	Condiments	Product ICKNK	43.9000	11
13	2	Condiments	Product WVJFP	40.0000	13
14	6	Meat/Poultry	Product BLCAX	39.0000	14
15	4	Dairy Products	Product OSFNS	38.0000	15

1.4 CREAR UN RANKING DE PRODUCTOS POR PRECIO EN ORDEN:



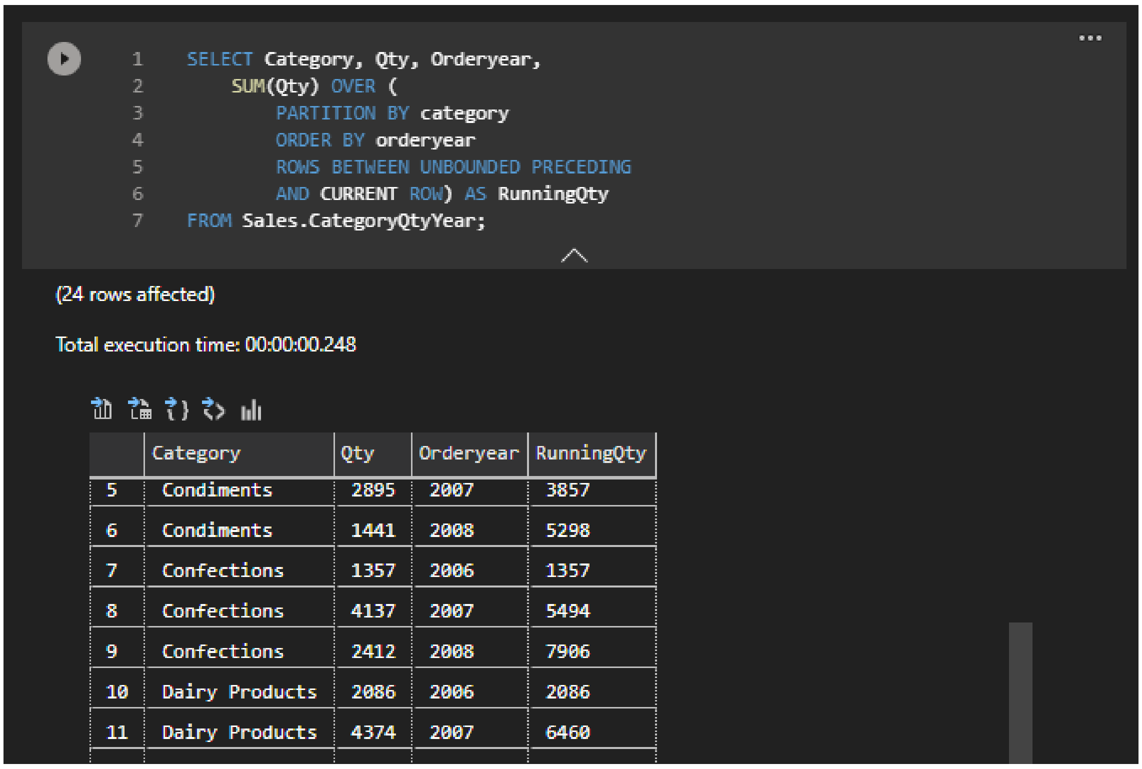
```
SELECT CatID, CatName, ProdName, UnitPrice,
    RANK() OVER(PARTITION BY CatID ORDER BY UnitPrice DESC) AS PriceRank
FROM Production.CategorizedProducts
ORDER BY CatID;
```

(77 rows affected)

Total execution time: 00:00:00.216

	CatID	CatName	ProdName	UnitPrice	PriceRank
1	1	Beverages	Product QDOMO	263.5000	1
2	1	Beverages	Product ZZZHR	46.0000	2
3	1	Beverages	Product RECZE	19.0000	3
4	1	Beverages	Product HHYDP	18.0000	4
5	1	Beverages	Product LSOFL	18.0000	4
6	1	Beverages	Product NEVTJ	18.0000	4
7	1	Beverages	Product JYGFE	18.0000	4
8	1	Beverages	Product TOONT	15.0000	8
9	1	Beverages	Product XLXQF	14.0000	9

1.5 USAR EL ENCUADRE PARA CREAR TOTAL ACUMULADO



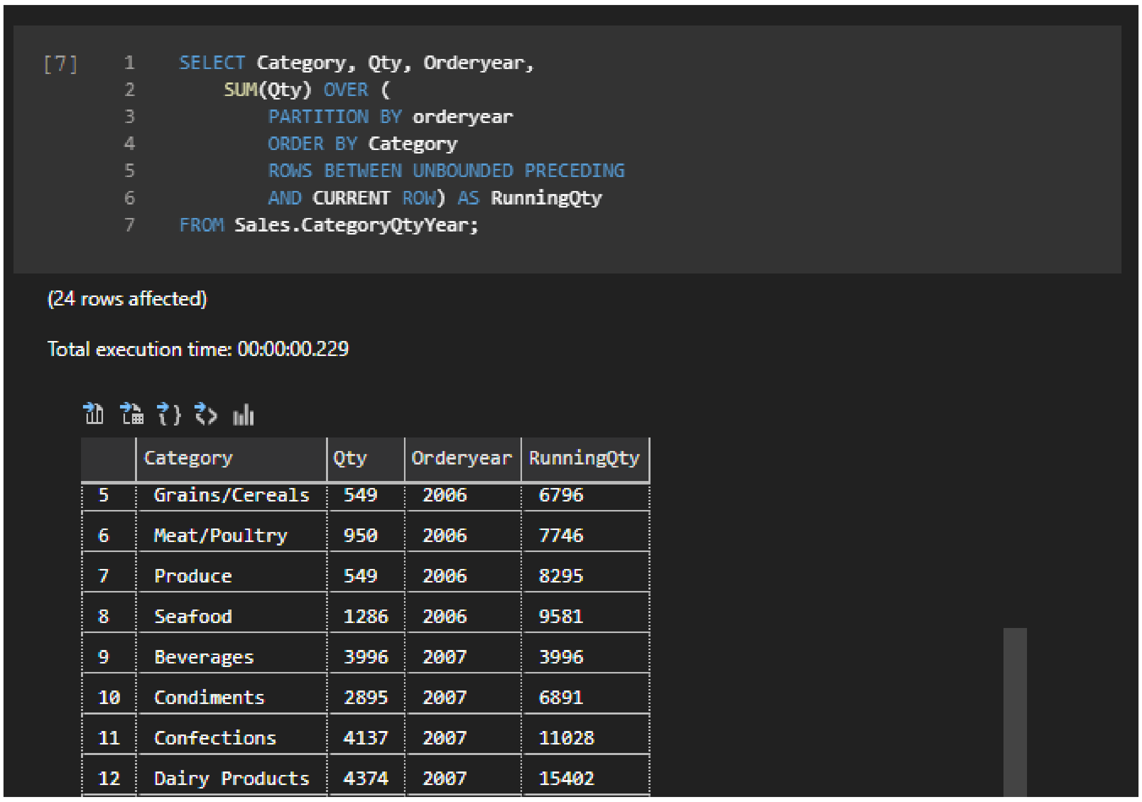
```
SELECT Category, Qty, Orderyear,
    SUM(Qty) OVER (
        PARTITION BY category
        ORDER BY orderyear
        ROWS BETWEEN UNBOUNDED PRECEDING
        AND CURRENT ROW) AS RunningQty
FROM Sales.CategoryQtyYear;
```

(24 rows affected)

Total execution time: 00:00:00.248

	Category	Qty	Orderyear	RunningQty
5	Condiments	2895	2007	3857
6	Condiments	1441	2008	5298
7	Confections	1357	2006	1357
8	Confections	4137	2007	5494
9	Confections	2412	2008	7906
10	Dairy Products	2086	2006	2086
11	Dairy Products	4374	2007	6460

1.6 MOSTRAR UN TOTAL ACUMULADO DE CANTIDAD POR AÑO



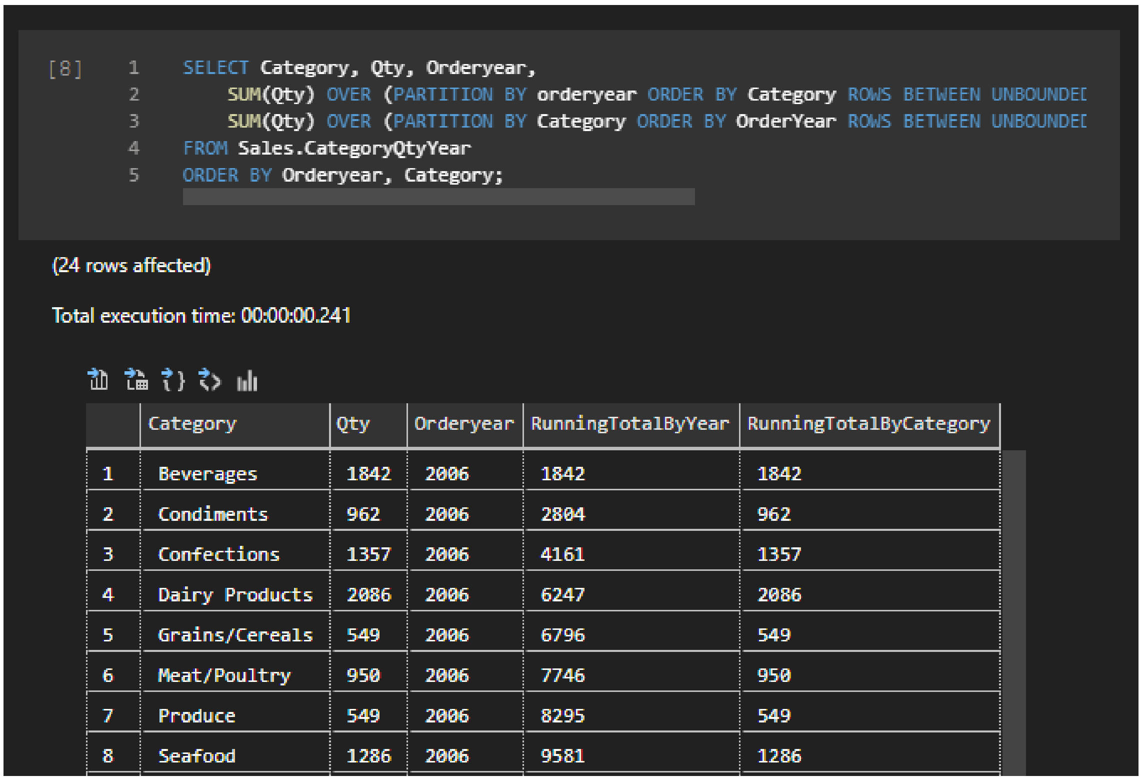
```
[7]  SELECT Category, Qty, Orderyear,
         SUM(Qty) OVER (
             PARTITION BY orderyear
             ORDER BY Category
             ROWS BETWEEN UNBOUNDED PRECEDING
             AND CURRENT ROW) AS RunningQty
     FROM Sales.CategoryQtyYear;
```

(24 rows affected)

Total execution time: 00:00:00.229

	Category	Qty	Orderyear	RunningQty
5	Grains/Cereals	549	2006	6796
6	Meat/Poultry	950	2006	7746
7	Produce	549	2006	8295
8	Seafood	1286	2006	9581
9	Beverages	3996	2007	3996
10	Condiments	2895	2007	6891
11	Confections	4137	2007	11028
12	Dairy Products	4374	2007	15402

1.7 MOSTRAR AMBOS LADO A LADO POR CATEGORÍA Y POR AÑO



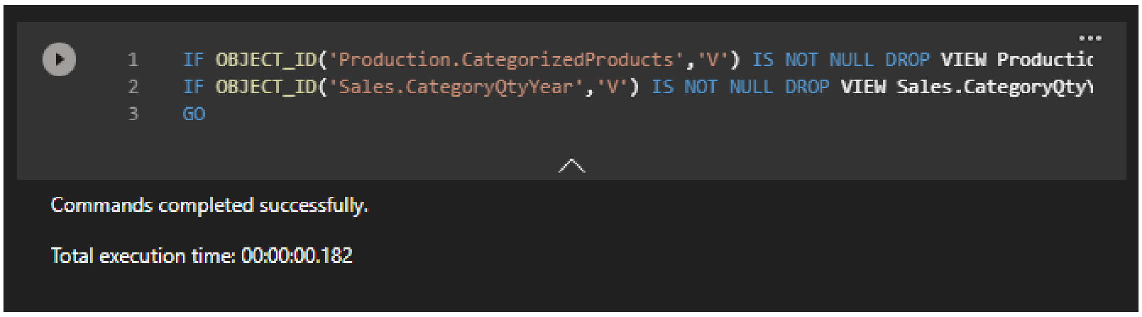
```
[8]  SELECT Category, Qty, Orderyear,
         SUM(Qty) OVER (PARTITION BY orderyear ORDER BY Category ROWS BETWEEN UNBOUNDEI
         SUM(Qty) OVER (PARTITION BY Category ORDER BY OrderYear ROWS BETWEEN UNBOUNDEI
     FROM Sales.CategoryQtyYear
     ORDER BY Orderyear, Category;
```

(24 rows affected)

Total execution time: 00:00:00.241

	Category	Qty	Orderyear	RunningTotalByYear	RunningTotalByCategory
1	Beverages	1842	2006	1842	1842
2	Condiments	962	2006	2804	962
3	Confections	1357	2006	4161	1357
4	Dairy Products	2086	2006	6247	2086
5	Grains/Cereals	549	2006	6796	549
6	Meat/Poultry	950	2006	7746	950
7	Produce	549	2006	8295	549
8	Seafood	1286	2006	9581	1286

1.8 LIMPIAR LOS CAMBIOS REALIZADOS

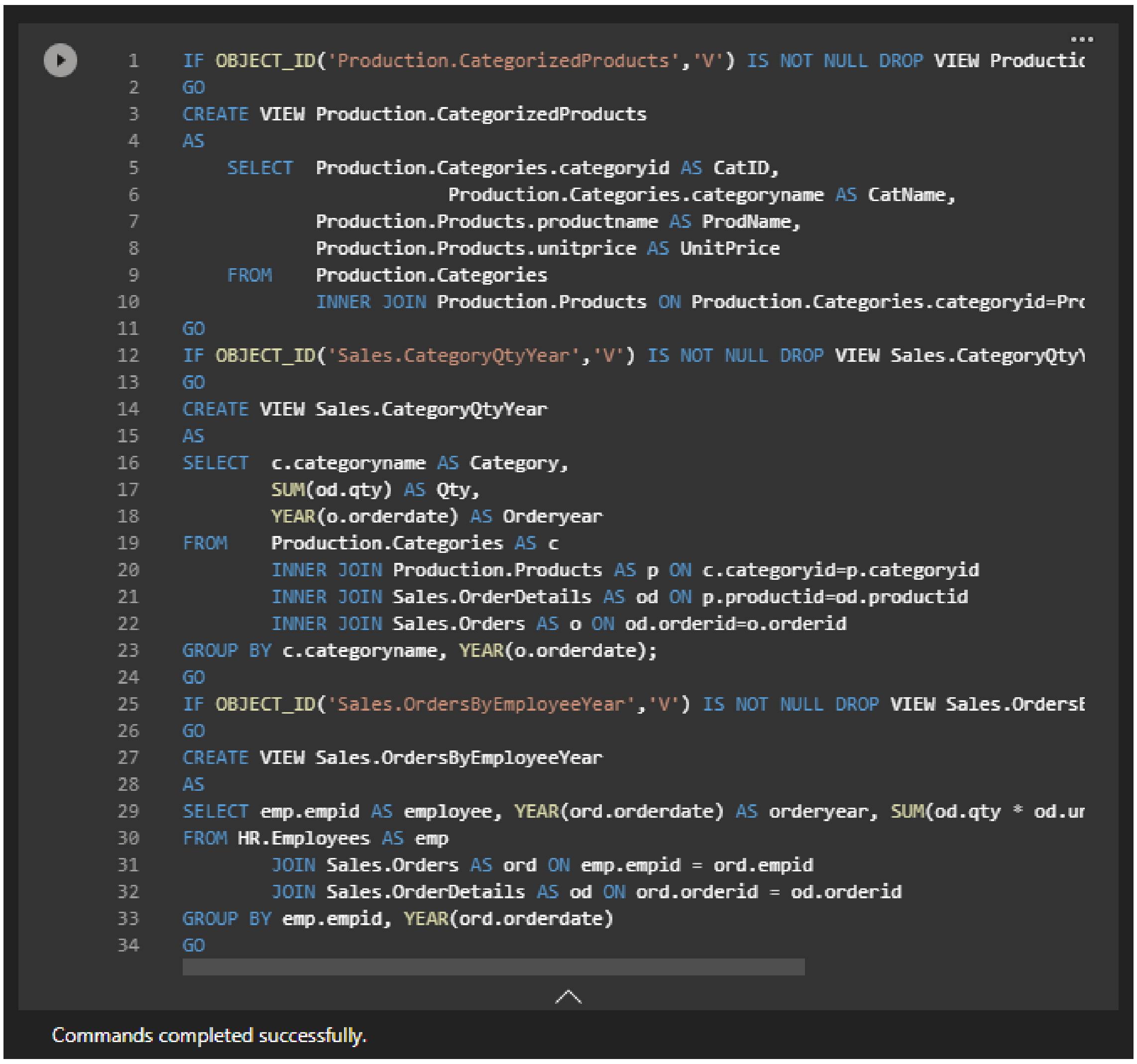


```
IF OBJECT_ID('Production.CategorizedProducts','V') IS NOT NULL DROP VIEW Productic
IF OBJECT_ID('Sales.CategoryQtyYear','V') IS NOT NULL DROP VIEW Sales.CategoryQty\
GO
```

```
Commands completed successfully.

Total execution time: 00:00:00.182
```

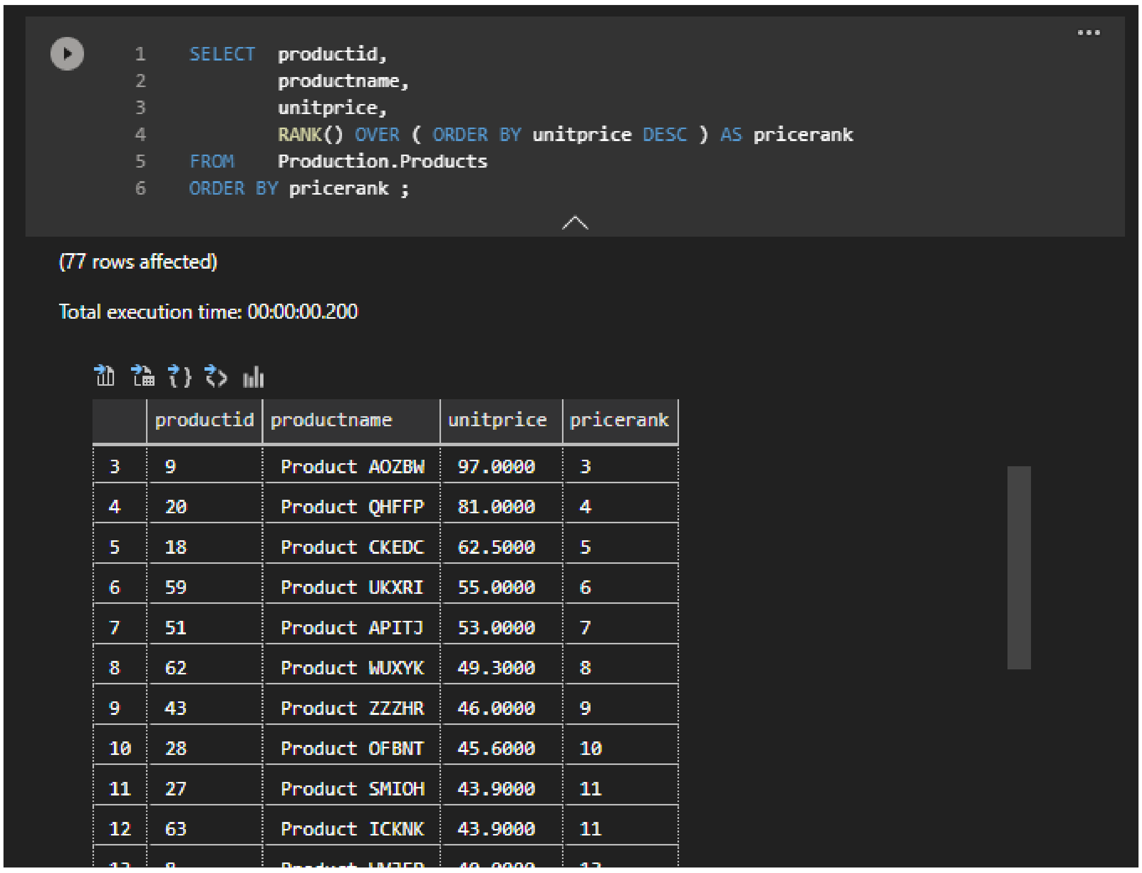
2.1 EJECUTAR LAS SIGUIENTES VISTAS



```
IF OBJECT_ID('Production.CategorizedProducts','V') IS NOT NULL DROP VIEW Productic
GO
CREATE VIEW Production.CategorizedProducts
AS
    SELECT  Production.Categories.categoryid AS CatID,
                  Production.Categories.categoryname AS CatName,
            Production.Products.productname AS ProdName,
            Production.Products.unitprice AS UnitPrice
    FROM    Production.Categories
            INNER JOIN Production.Products ON Production.Categories.categoryid=Prc
GO
IF OBJECT_ID('Sales.CategoryQtyYear','V') IS NOT NULL DROP VIEW Sales.CategoryQty\
GO
CREATE VIEW Sales.CategoryQtyYear
AS
SELECT  c.categoryname AS Category,
        SUM(od.qty) AS Qty,
        YEAR(o.orderdate) AS Orderyear
FROM    Production.Categories AS c
        INNER JOIN Production.Products AS p ON c.categoryid=p.categoryid
        INNER JOIN Sales.OrderDetails AS od ON p.productid=od.productid
        INNER JOIN Sales.Orders AS o ON od.orderid=o.orderid
GROUP BY c.categoryname, YEAR(o.orderdate);
GO
IF OBJECT_ID('Sales.OrdersByEmployeeYear','V') IS NOT NULL DROP VIEW Sales.OrdersE
GO
CREATE VIEW Sales.OrdersByEmployeeYear
AS
SELECT emp.empid AS employee, YEAR(ord.orderdate) AS orderyear, SUM(od.qty * od.ur
FROM HR.Employees AS emp
        JOIN Sales.Orders AS ord ON emp.empid = ord.empid
        JOIN Sales.OrderDetails AS od ON ord.orderid = od.orderid
GROUP BY emp.empid, YEAR(ord.orderdate)
GO
```

```
Commands completed successfully.
```

2.2 CREAR UN RANKING DE PRODUCTOS POR PRECIO UTILIZANDO RANK

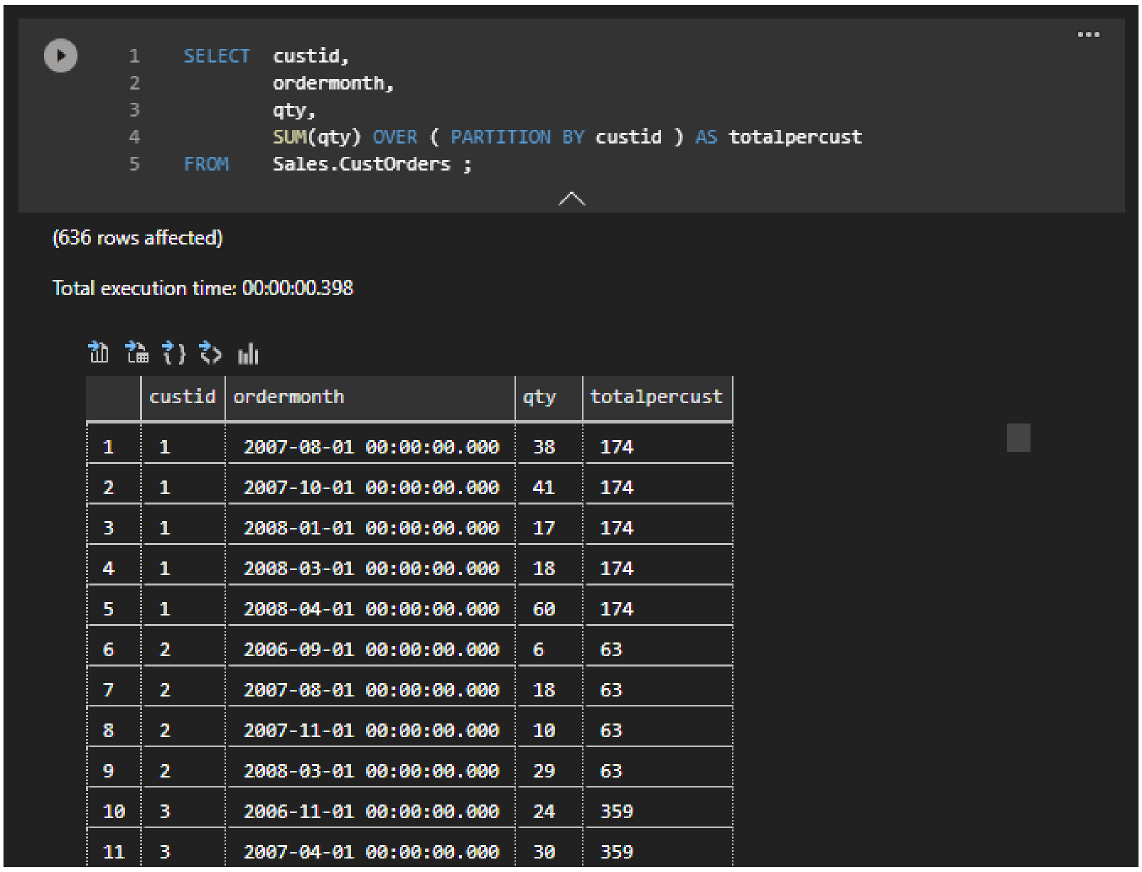

```
SELECT  productid,
        productname,
        unitprice,
        RANK() OVER ( ORDER BY unitprice DESC ) AS pricerank
FROM    Production.Products
ORDER BY pricerank ;
```

```
(77 rows affected)

Total execution time: 00:00:00.200
```

	productid	productname	unitprice	pricerank
3	9	Product AOZBW	97.0000	3
4	20	Product QHFFP	81.0000	4
5	18	Product CKEDC	62.5000	5
6	59	Product UKXRI	55.0000	6
7	51	Product APITJ	53.0000	7
8	62	Product WUXYK	49.3000	8
9	43	Product ZZZHR	46.0000	9
10	28	Product OFBNT	45.6000	10
11	27	Product SMIOH	43.9000	11
12	63	Product ICKNK	43.9000	11

2.3 CREAR UNA FUNCION DE VENTANA



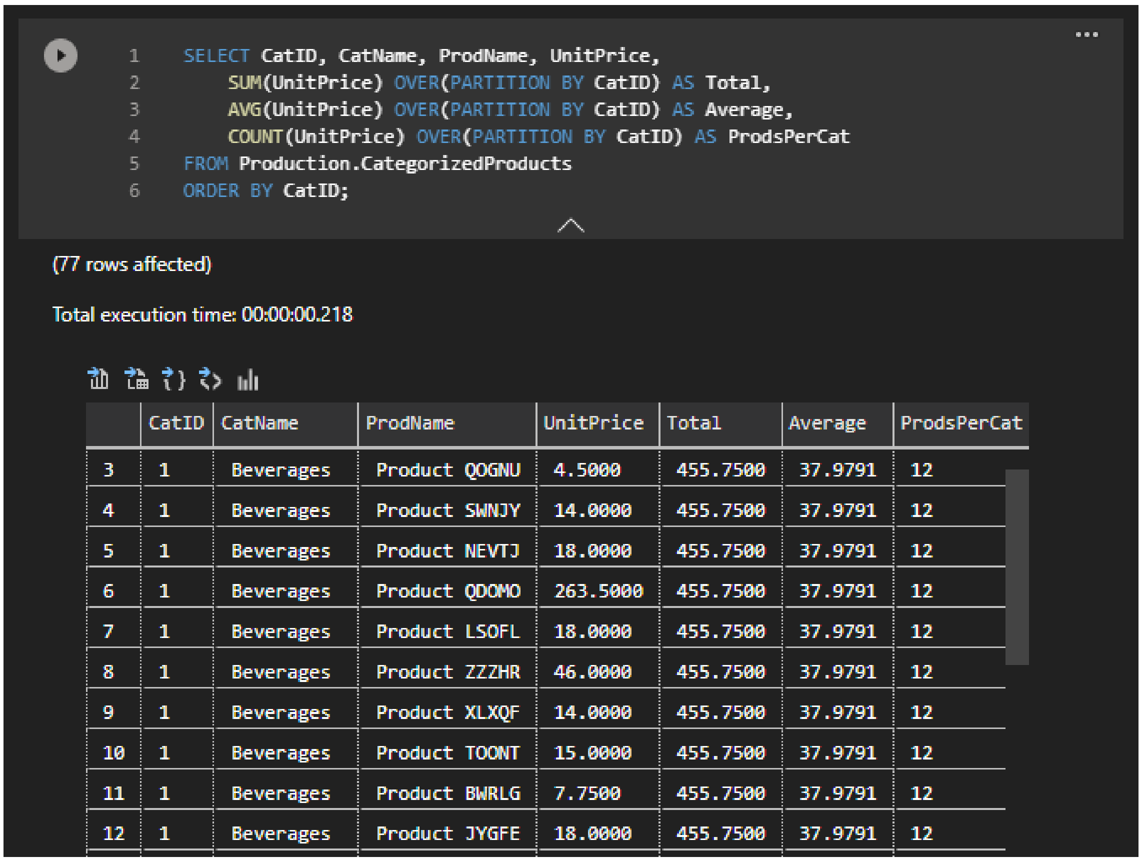
```
SELECT  custid,
        ordermonth,
        qty,
        SUM(qty) OVER ( PARTITION BY custid ) AS totalpercust
FROM    Sales.CustOrders ;
```

(636 rows affected)

Total execution time: 00:00:00.398

	custid	ordermonth	qty	totalpercust
1	1	2007-08-01 00:00:00.000	38	174
2	1	2007-10-01 00:00:00.000	41	174
3	1	2008-01-01 00:00:00.000	17	174
4	1	2008-03-01 00:00:00.000	18	174
5	1	2008-04-01 00:00:00.000	60	174
6	2	2006-09-01 00:00:00.000	6	63
7	2	2007-08-01 00:00:00.000	18	63
8	2	2007-11-01 00:00:00.000	10	63
9	2	2008-03-01 00:00:00.000	29	63
10	3	2006-11-01 00:00:00.000	24	359
11	3	2007-04-01 00:00:00.000	30	359

2.4 LADO POR LADO, USAR FUNCIONES DE AGREGACIONES



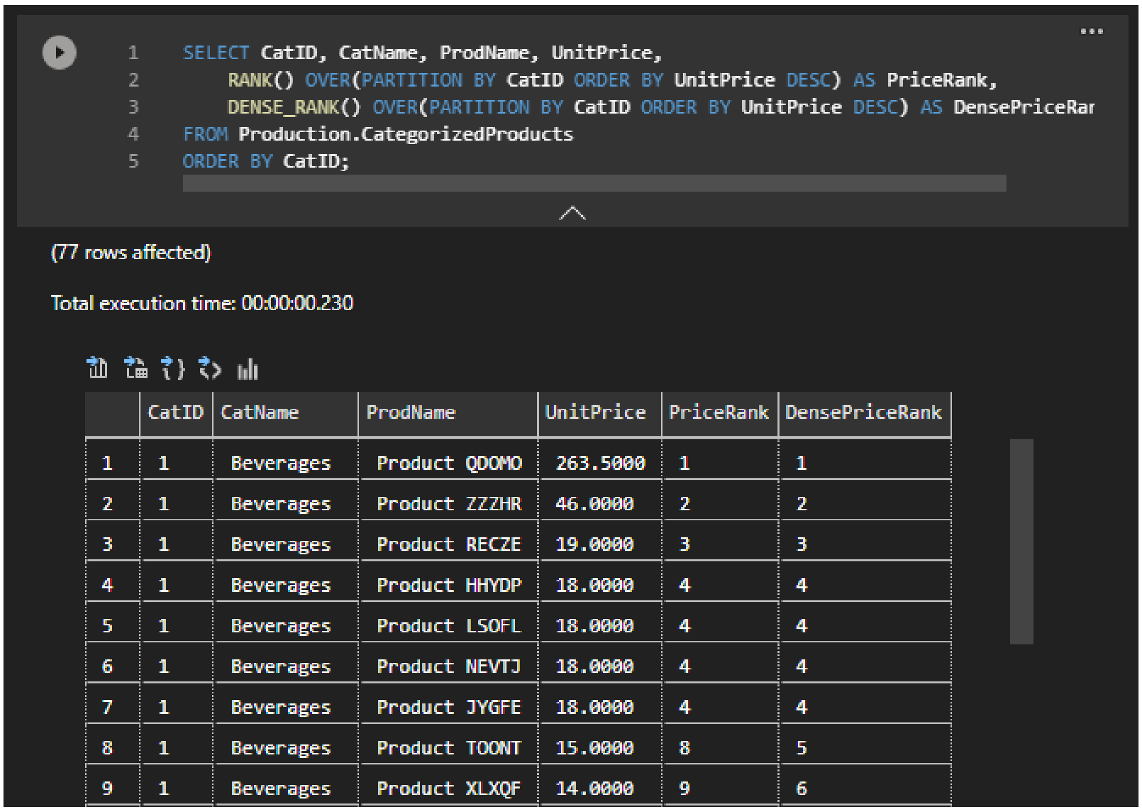
```
SELECT CatID, CatName, ProdName, UnitPrice,
    SUM(UnitPrice) OVER(PARTITION BY CatID) AS Total,
    AVG(UnitPrice) OVER(PARTITION BY CatID) AS Average,
    COUNT(UnitPrice) OVER(PARTITION BY CatID) AS ProdsPerCat
FROM Production.CategorizedProducts
ORDER BY CatID;
```

(77 rows affected)

Total execution time: 00:00:00.218

	CatID	CatName	ProdName	UnitPrice	Total	Average	ProdsPerCat
3	1	Beverages	Product QOGNU	4.5000	455.7500	37.9791	12
4	1	Beverages	Product SWNJY	14.0000	455.7500	37.9791	12
5	1	Beverages	Product NEVTJ	18.0000	455.7500	37.9791	12
6	1	Beverages	Product QDOMO	263.5000	455.7500	37.9791	12
7	1	Beverages	Product LSOFL	18.0000	455.7500	37.9791	12
8	1	Beverages	Product ZZZHR	46.0000	455.7500	37.9791	12
9	1	Beverages	Product XLXQF	14.0000	455.7500	37.9791	12
10	1	Beverages	Product TOONT	15.0000	455.7500	37.9791	12
11	1	Beverages	Product BWRLG	7.7500	455.7500	37.9791	12
12	1	Beverages	Product JYGFE	18.0000	455.7500	37.9791	12

2.5 COMPARAR RANK Y DENSE_RANK



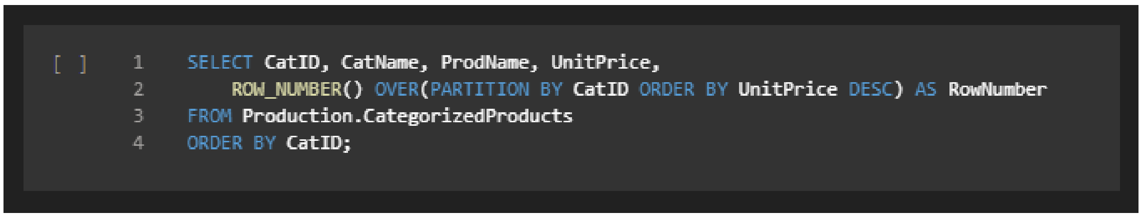
```sql
SELECT CatID, CatName, ProdName, UnitPrice,
    RANK() OVER(PARTITION BY CatID ORDER BY UnitPrice DESC) AS PriceRank,
    DENSE_RANK() OVER(PARTITION BY CatID ORDER BY UnitPrice DESC) AS DensePriceRar
FROM Production.CategorizedProducts
ORDER BY CatID;
```

(77 rows affected)

Total execution time: 00:00:00.230

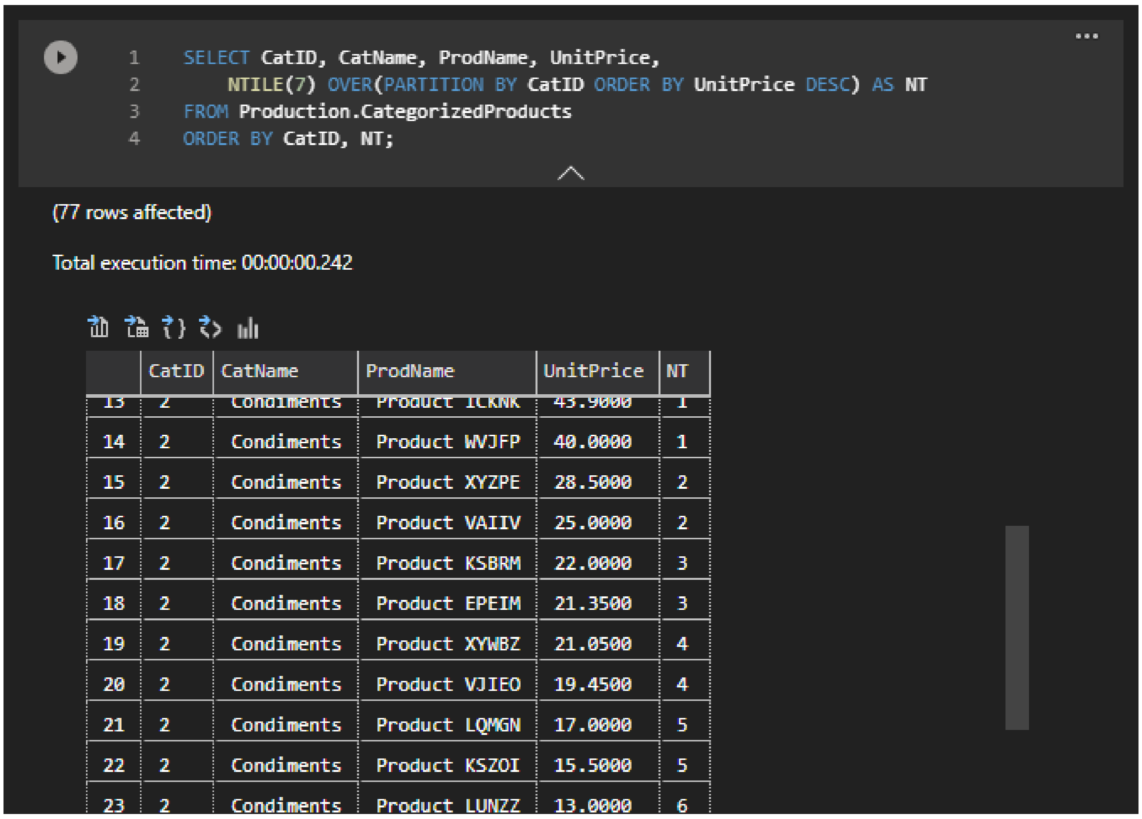
	CatID	CatName	ProdName	UnitPrice	PriceRank	DensePriceRank
1	1	Beverages	Product QDOMO	263.5000	1	1
2	1	Beverages	Product ZZZHR	46.0000	2	2
3	1	Beverages	Product RECZE	19.0000	3	3
4	1	Beverages	Product HHYDP	18.0000	4	4
5	1	Beverages	Product LSOFL	18.0000	4	4
6	1	Beverages	Product NEVTJ	18.0000	4	4
7	1	Beverages	Product JYGFE	18.0000	4	4
8	1	Beverages	Product TOONT	15.0000	8	5
9	1	Beverages	Product XLXQF	14.0000	9	6

2.6 AHORA UTILIZAR ROW_NUMERBR()



```sql
SELECT CatID, CatName, ProdName, UnitPrice,
    ROW_NUMBER() OVER(PARTITION BY CatID ORDER BY UnitPrice DESC) AS RowNumber
FROM Production.CategorizedProducts
ORDER BY CatID;
```

2.7 NTILE PARA CREAR 7 GRUPOS



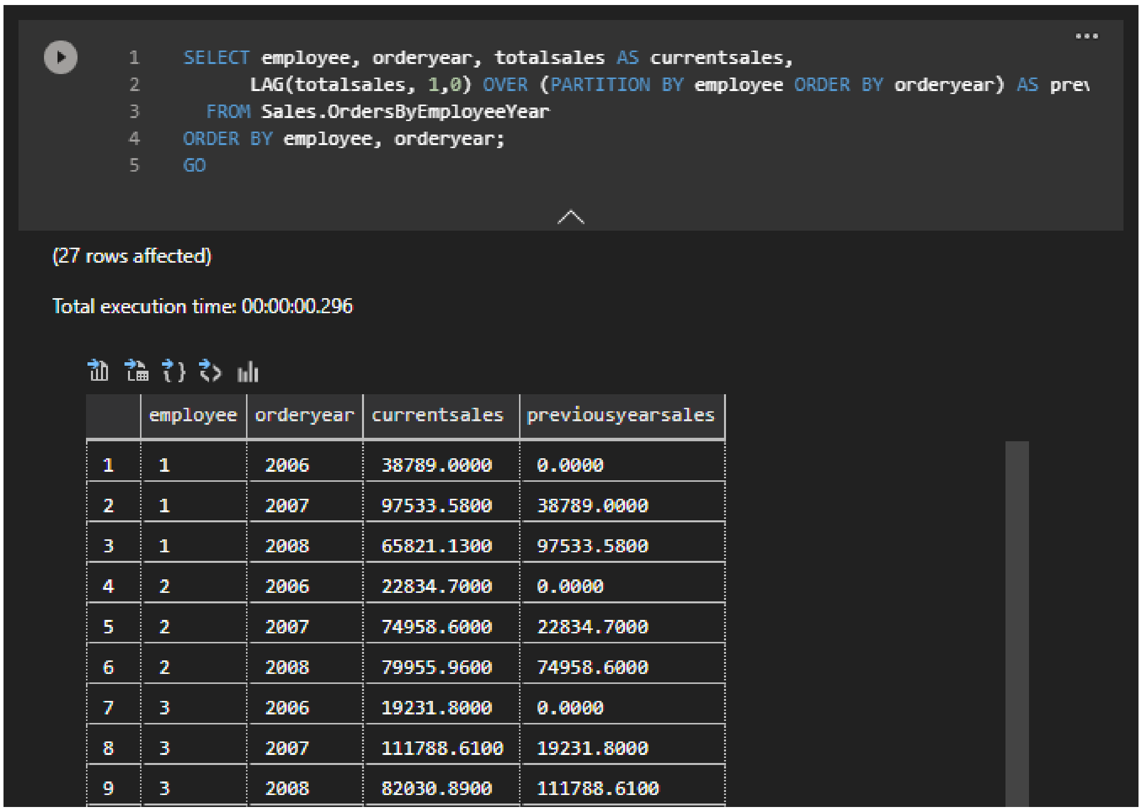
```sql
SELECT CatID, CatName, ProdName, UnitPrice,
    NTILE(7) OVER(PARTITION BY CatID ORDER BY UnitPrice DESC) AS NT
FROM Production.CategorizedProducts
ORDER BY CatID, NT;
```

(77 rows affected)

Total execution time: 00:00:00.242

	CatID	CatName	ProdName	UnitPrice	NT
13	2	Condiments	Product ICKNK	43.9000	1
14	2	Condiments	Product WVJFP	40.0000	1
15	2	Condiments	Product XYZPE	28.5000	2
16	2	Condiments	Product VAIIV	25.0000	2
17	2	Condiments	Product KSBRM	22.0000	3
18	2	Condiments	Product EPEIM	21.3500	3
19	2	Condiments	Product XYWBZ	21.0500	4
20	2	Condiments	Product VJIEO	19.4500	4
21	2	Condiments	Product LQMGN	17.0000	5
22	2	Condiments	Product KSZOI	15.5000	5
23	2	Condiments	Product LUNZZ	13.0000	6

2.8 FUNCIONES OFFSET



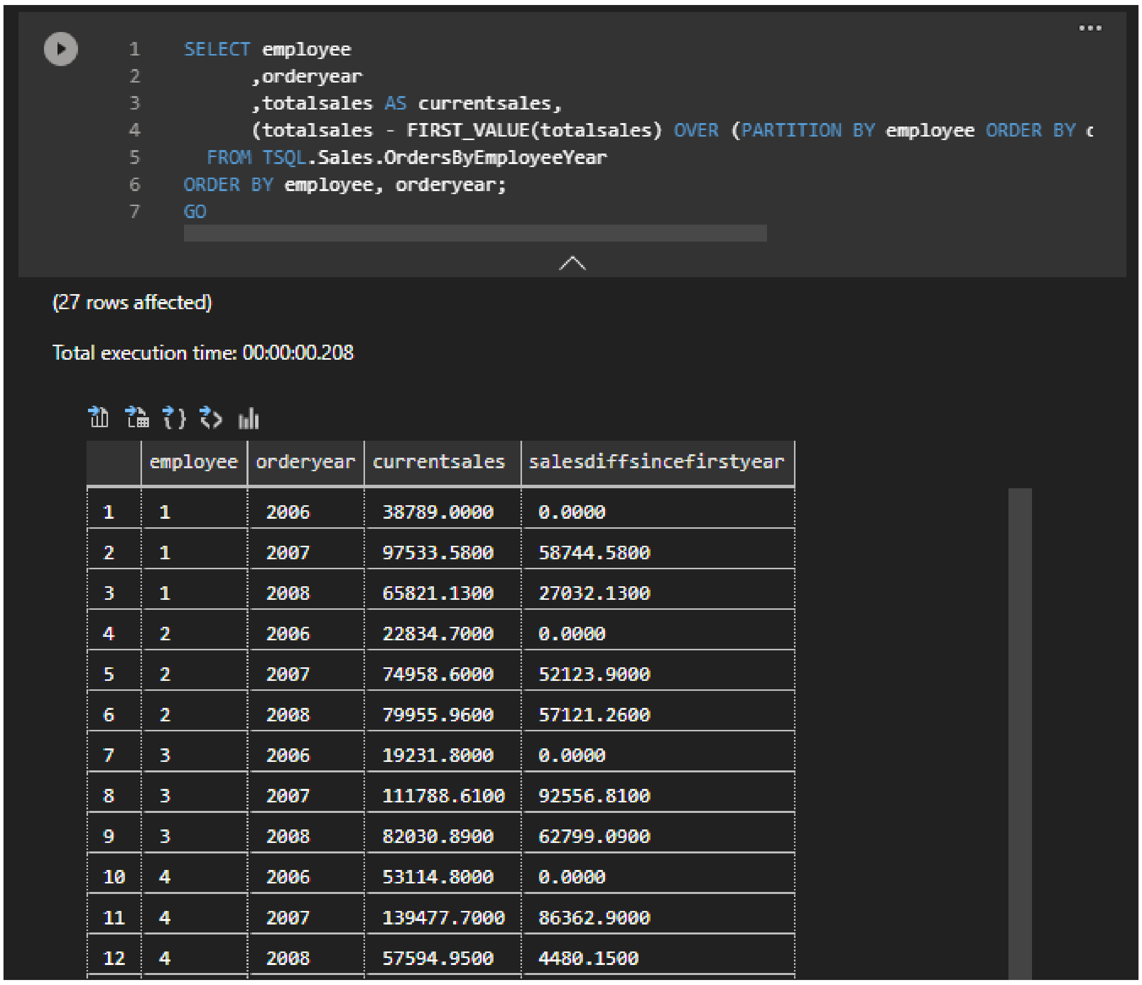
```sql
SELECT employee, orderyear, totalsales AS currentsales,
       LAG(totalsales, 1,0) OVER (PARTITION BY employee ORDER BY orderyear) AS prev
  FROM Sales.OrdersByEmployeeYear
ORDER BY employee, orderyear;
GO
```

(27 rows affected)

Total execution time: 00:00:00.296

	employee	orderyear	currentsales	previousyearsales
1	1	2006	38789.0000	0.0000
2	1	2007	97533.5800	38789.0000
3	1	2008	65821.1300	97533.5800
4	2	2006	22834.7000	0.0000
5	2	2007	74958.6000	22834.7000
6	2	2008	79955.9600	74958.6000
7	3	2006	19231.8000	0.0000
8	3	2007	111788.6100	19231.8000
9	3	2008	82030.8900	111788.6100

2.9 USAR FIRST_VALUE



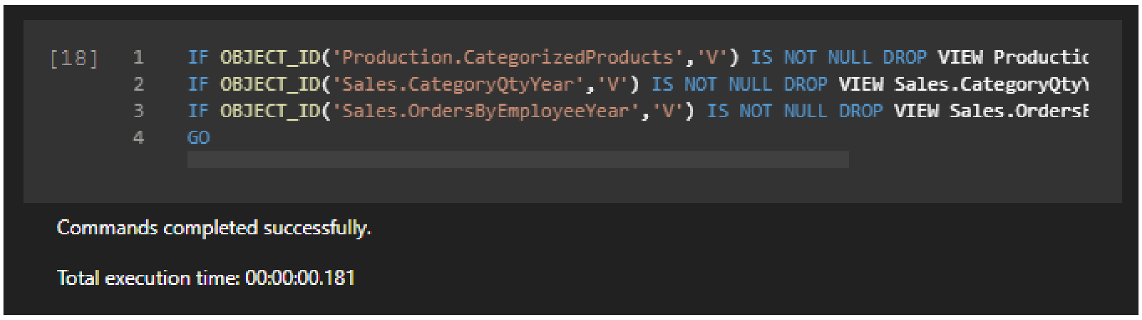
```
SELECT employee
      ,orderyear
      ,totalsales AS currentsales,
      (totalsales - FIRST_VALUE(totalsales) OVER (PARTITION BY employee ORDER BY c
  FROM TSQL.Sales.OrdersByEmployeeYear
ORDER BY employee, orderyear;
GO
```

```
(27 rows affected)

Total execution time: 00:00:00.208
```

	employee	orderyear	currentsales	salesdiffsincefirstyear
1	1	2006	38789.0000	0.0000
2	1	2007	97533.5800	58744.5800
3	1	2008	65821.1300	27032.1300
4	2	2006	22834.7000	0.0000
5	2	2007	74958.6000	52123.9000
6	2	2008	79955.9600	57121.2600
7	3	2006	19231.8000	0.0000
8	3	2007	111788.6100	92556.8100
9	3	2008	82030.8900	62799.0900
10	4	2006	53114.8000	0.0000
11	4	2007	139477.7000	86362.9000
12	4	2008	57594.9500	4480.1500

2.10 FINALMENTE LIMPIAR LOSCAMBIOS



```
[18]
IF OBJECT_ID('Production.CategorizedProducts','V') IS NOT NULL DROP VIEW Productic
IF OBJECT_ID('Sales.CategoryQtyYear','V') IS NOT NULL DROP VIEW Sales.CategoryQty\
IF OBJECT_ID('Sales.OrdersByEmployeeYear','V') IS NOT NULL DROP VIEW Sales.OrdersE
GO
```

```
Commands completed successfully.

Total execution time: 00:00:00.181
```