SQL LEARNING IMPLEMENTATION ON ADVENTURE WORKS DATASET

-BY SUNIDHI SHARMA

What is the AdventureWorks Database?

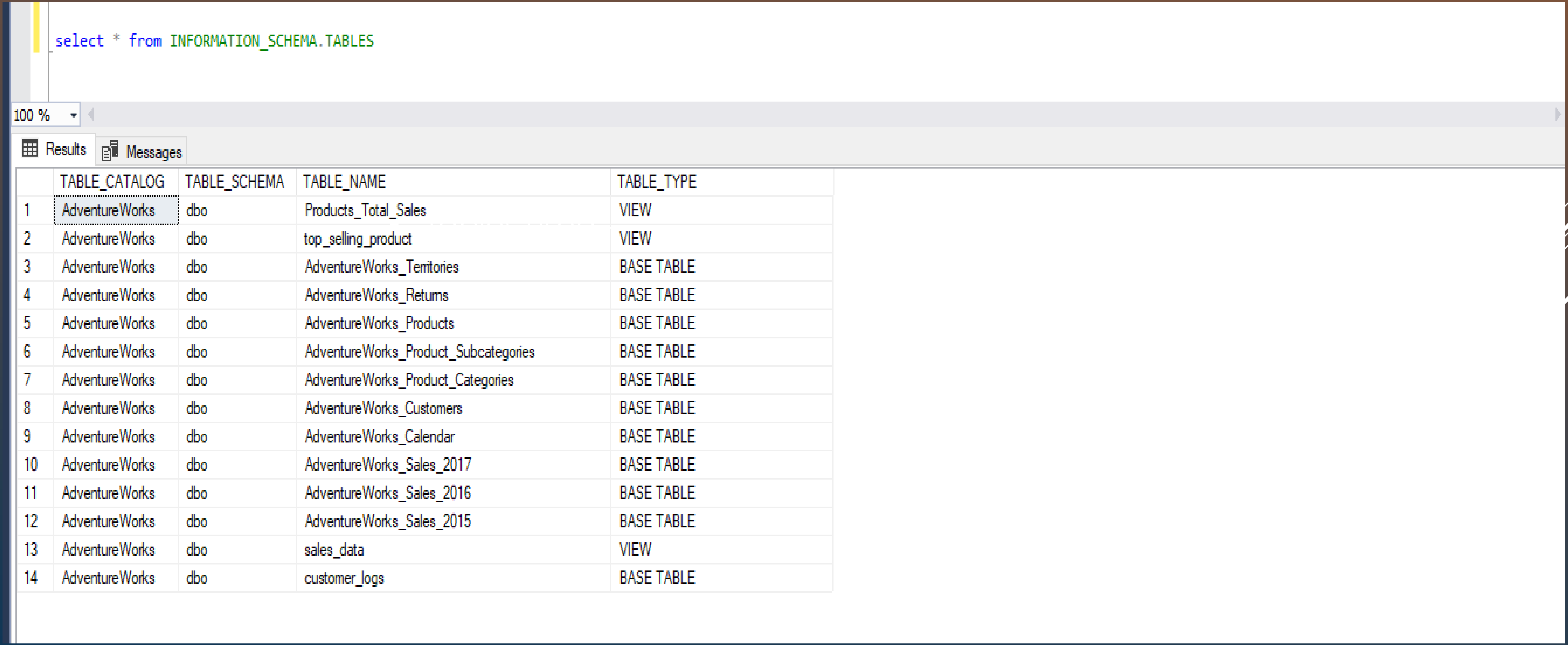
It’s a Microsoft product sample for an online transaction processing database of a fictitious multinational manufacturing company called Adventure Works Cycles, Which sells bicycles and their accessories.

CONCEPTS IMPLEMENTED :

- Aggregate Functions- Sum, Average, Count.
- String Functions – Like, Concat, Replace.
- Logical and Comparison Operators.
- Cast and Top Clause
- Union All
- Regex operation
- Date/Time Manipulation – datediff(), getdate(), month(), year().
- Case statement.
- Joins
- Identity function.
- Views.
- Transaction and Procedure.
- Trigger.
- Window's Function- rank, dense_rank, sum, average etc.

- Tables Used -

Base Table type are provided and View Table type is created.

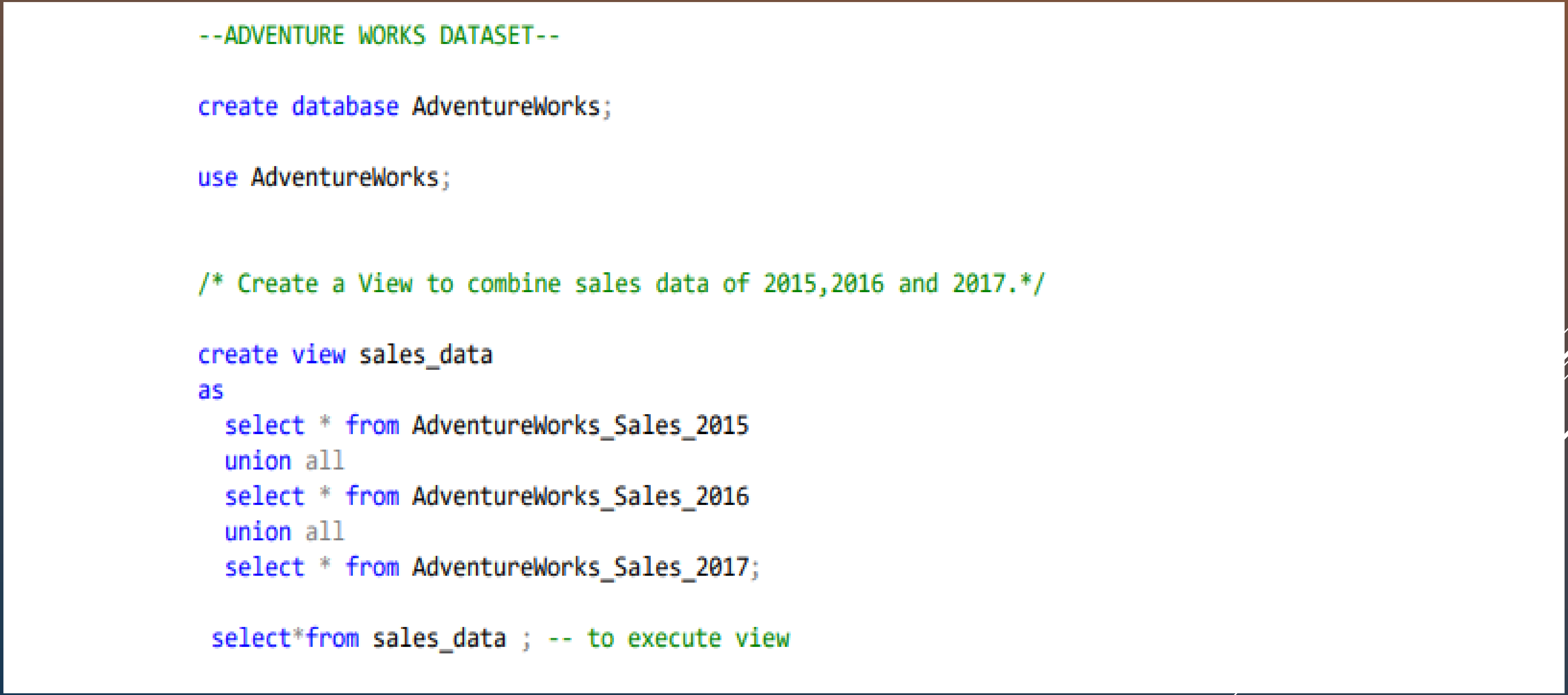


```sql
select * from INFORMATION_SCHEMA.TABLES
```

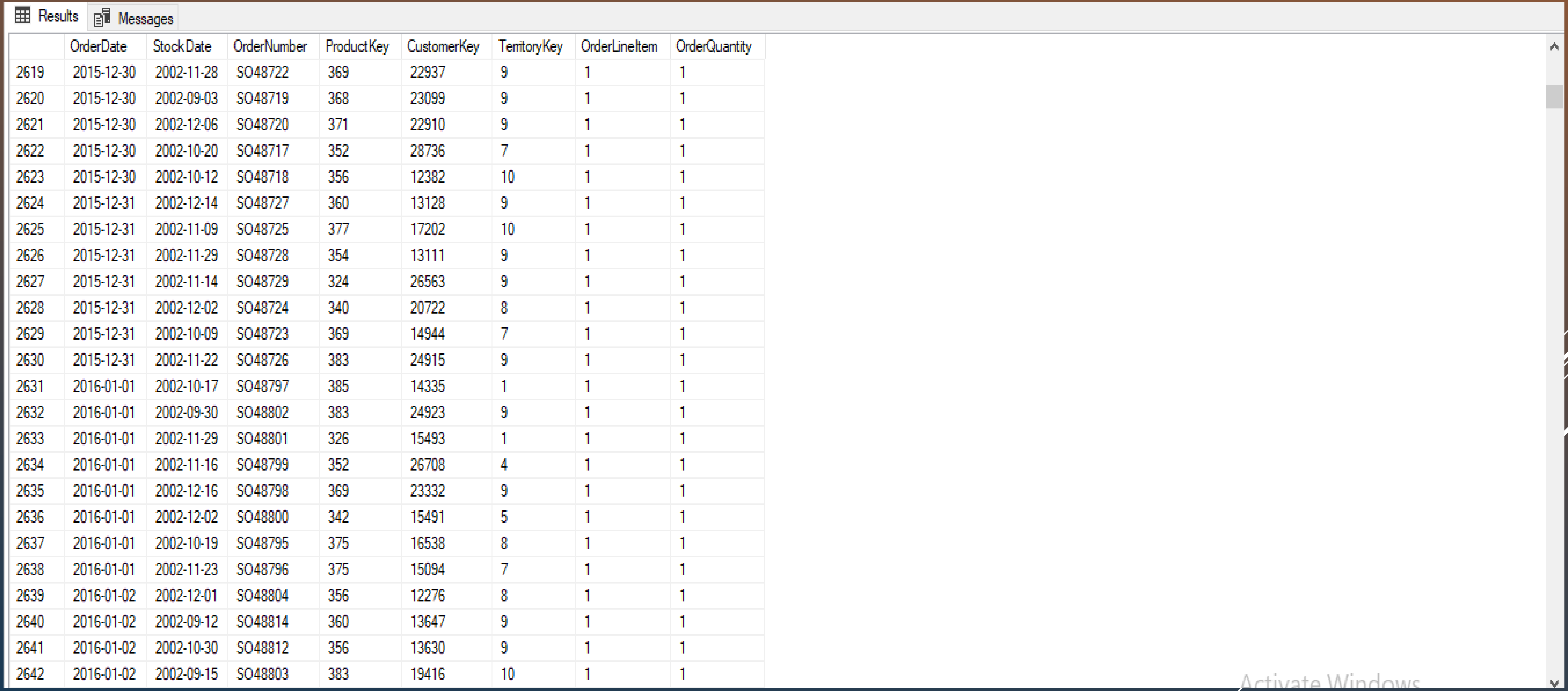
	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
1	AdventureWorks	dbo	Products_Total_Sales	VIEW
2	AdventureWorks	dbo	top_selling_product	VIEW
3	AdventureWorks	dbo	AdventureWorks_Territories	BASE TABLE
4	AdventureWorks	dbo	AdventureWorks_Returns	BASE TABLE
5	AdventureWorks	dbo	AdventureWorks_Products	BASE TABLE
6	AdventureWorks	dbo	AdventureWorks_Product_Subcategories	BASE TABLE
7	AdventureWorks	dbo	AdventureWorks_Product_Categories	BASE TABLE
8	AdventureWorks	dbo	AdventureWorks_Customers	BASE TABLE
9	AdventureWorks	dbo	AdventureWorks_Calendar	BASE TABLE
10	AdventureWorks	dbo	AdventureWorks_Sales_2017	BASE TABLE
11	AdventureWorks	dbo	AdventureWorks_Sales_2016	BASE TABLE
12	AdventureWorks	dbo	AdventureWorks_Sales_2015	BASE TABLE
13	AdventureWorks	dbo	sales_data	VIEW
14	AdventureWorks	dbo	customer_logs	BASE TABLE

So Let’s Begin Quarrying some Data.

1. Create a View to combine sales data of 2015,2016 and 2017.

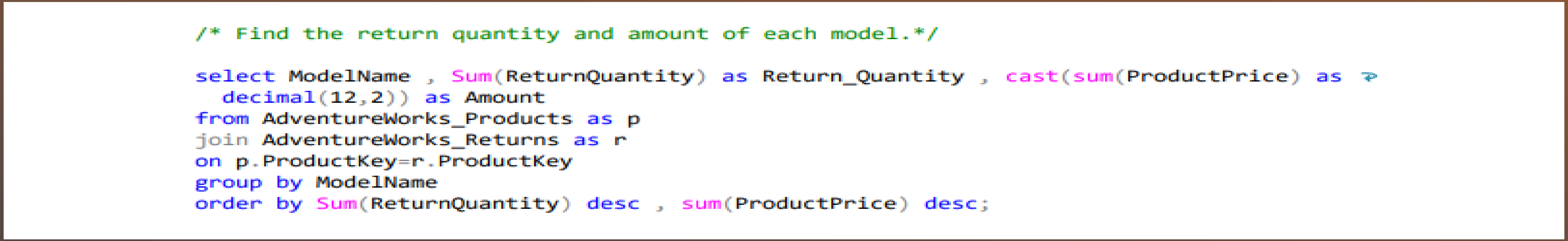


```
--ADVENTURE WORKS DATASET--

create database AdventureWorks;

use AdventureWorks;


/* Create a View to combine sales data of 2015,2016 and 2017.*/

create view sales_data
as
  select * from AdventureWorks_Sales_2015
  union all
  select * from AdventureWorks_Sales_2016
  union all
  select * from AdventureWorks_Sales_2017;

 select*from sales_data ; -- to execute view
```

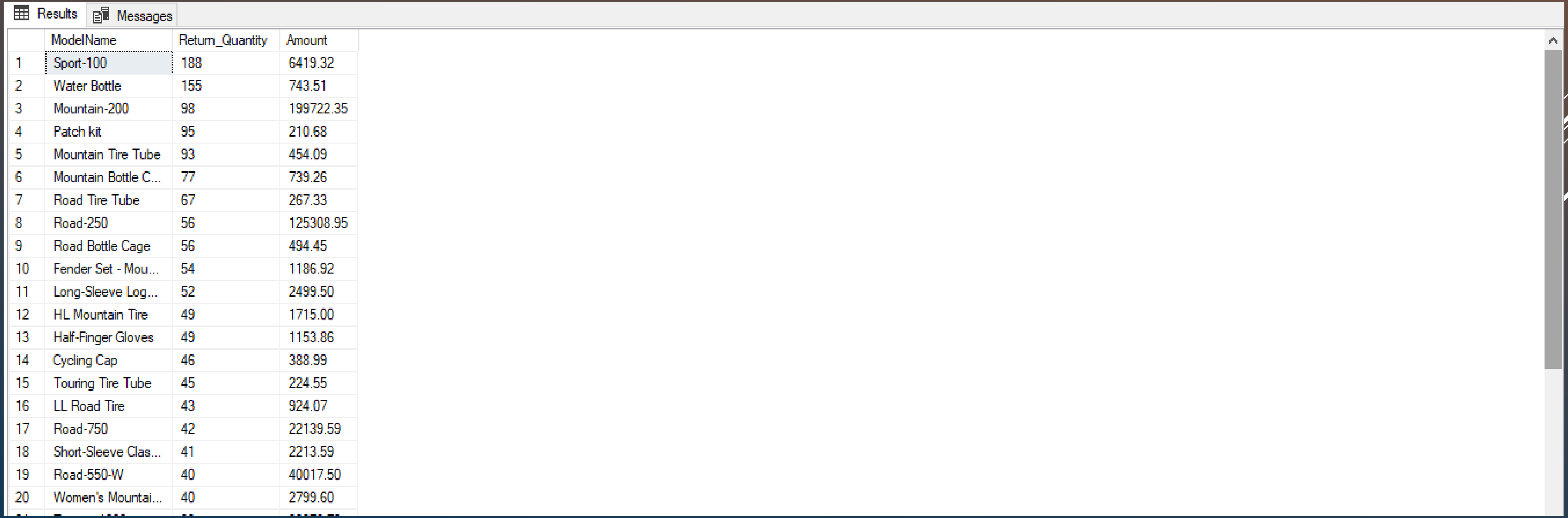

Results | Messages

	OrderDate	StockDate	OrderNumber	ProductKey	CustomerKey	TerritoryKey	OrderLineItem	OrderQuantity
2619	2015-12-30	2002-11-28	SO48722	369	22937	9	1	1
2620	2015-12-30	2002-09-03	SO48719	368	23099	9	1	1
2621	2015-12-30	2002-12-06	SO48720	371	22910	9	1	1
2622	2015-12-30	2002-10-20	SO48717	352	28736	7	1	1
2623	2015-12-30	2002-10-12	SO48718	356	12382	10	1	1
2624	2015-12-31	2002-12-14	SO48727	360	13128	9	1	1
2625	2015-12-31	2002-11-09	SO48725	377	17202	10	1	1
2626	2015-12-31	2002-11-29	SO48728	354	13111	9	1	1
2627	2015-12-31	2002-11-14	SO48729	324	26563	9	1	1
2628	2015-12-31	2002-12-02	SO48724	340	20722	8	1	1
2629	2015-12-31	2002-10-09	SO48723	369	14944	7	1	1
2630	2015-12-31	2002-11-22	SO48726	383	24915	9	1	1
2631	2016-01-01	2002-10-17	SO48797	385	14335	1	1	1
2632	2016-01-01	2002-09-30	SO48802	383	24923	9	1	1
2633	2016-01-01	2002-11-29	SO48801	326	15493	1	1	1
2634	2016-01-01	2002-11-16	SO48799	352	26708	4	1	1
2635	2016-01-01	2002-12-16	SO48798	369	23332	9	1	1
2636	2016-01-01	2002-12-02	SO48800	342	15491	5	1	1
2637	2016-01-01	2002-10-19	SO48795	375	16538	8	1	1
2638	2016-01-01	2002-11-23	SO48796	375	15094	7	1	1
2639	2016-01-02	2002-12-01	SO48804	356	12276	8	1	1
2640	2016-01-02	2002-09-12	SO48814	360	13647	9	1	1
2641	2016-01-02	2002-10-30	SO48812	356	13630	9	1	1
2642	2016-01-02	2002-09-15	SO48803	383	19416	10	1	1

2. Find the Return Quantity and amount of each model.



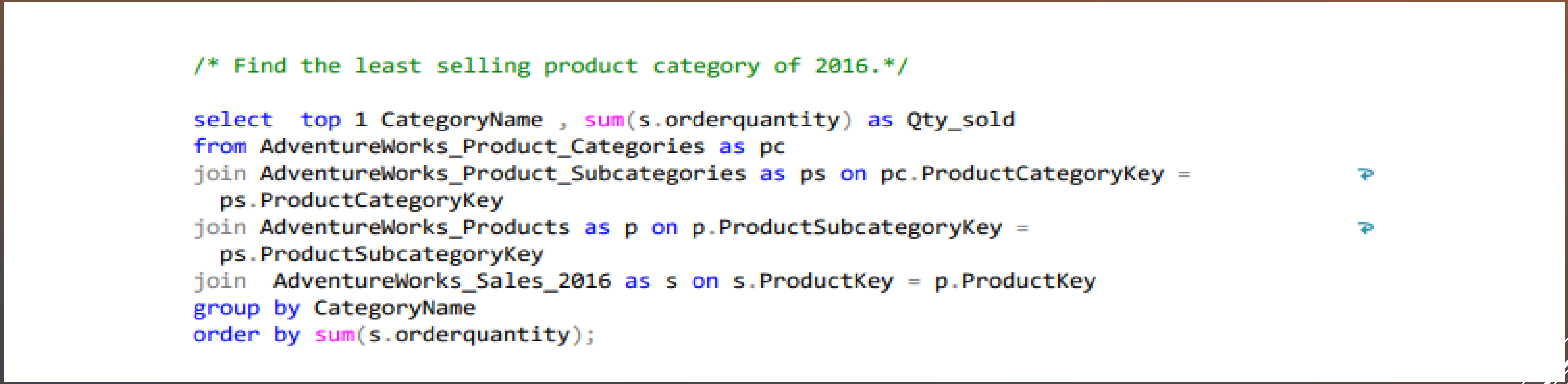
```sql
/* Find the return quantity and amount of each model.*/

select ModelName , Sum(ReturnQuantity) as Return_Quantity , cast(sum(ProductPrice) as
  decimal(12,2)) as Amount
from AdventureWorks_Products as p
join AdventureWorks_Returns as r
on p.ProductKey=r.ProductKey
group by ModelName
order by Sum(ReturnQuantity) desc , sum(ProductPrice) desc;
```



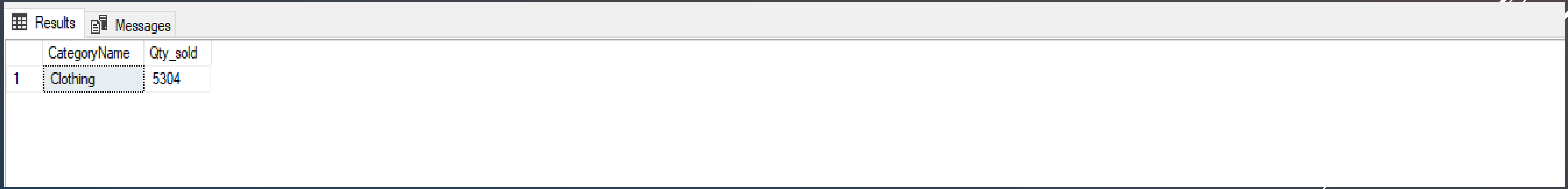
Results | Messages

	ModelName	Return_Quantity	Amount
1	Sport-100	188	6419.32
2	Water Bottle	155	743.51
3	Mountain-200	98	199722.35
4	Patch kit	95	210.68
5	Mountain Tire Tube	93	454.09
6	Mountain Bottle C...	77	739.26
7	Road Tire Tube	67	267.33
8	Road-250	56	125308.95
9	Road Bottle Cage	56	494.45
10	Fender Set - Mou...	54	1186.92
11	Long-Sleeve Log...	52	2499.50
12	HL Mountain Tire	49	1715.00
13	Half-Finger Gloves	49	1153.86
14	Cycling Cap	46	388.99
15	Touring Tire Tube	45	224.55
16	LL Road Tire	43	924.07
17	Road-750	42	22139.59
18	Short-Sleeve Clas...	41	2213.59
19	Road-550-W	40	40017.50
20	Women's Mountai...	40	2799.60

3. Find the least-selling Product Category of 2016.

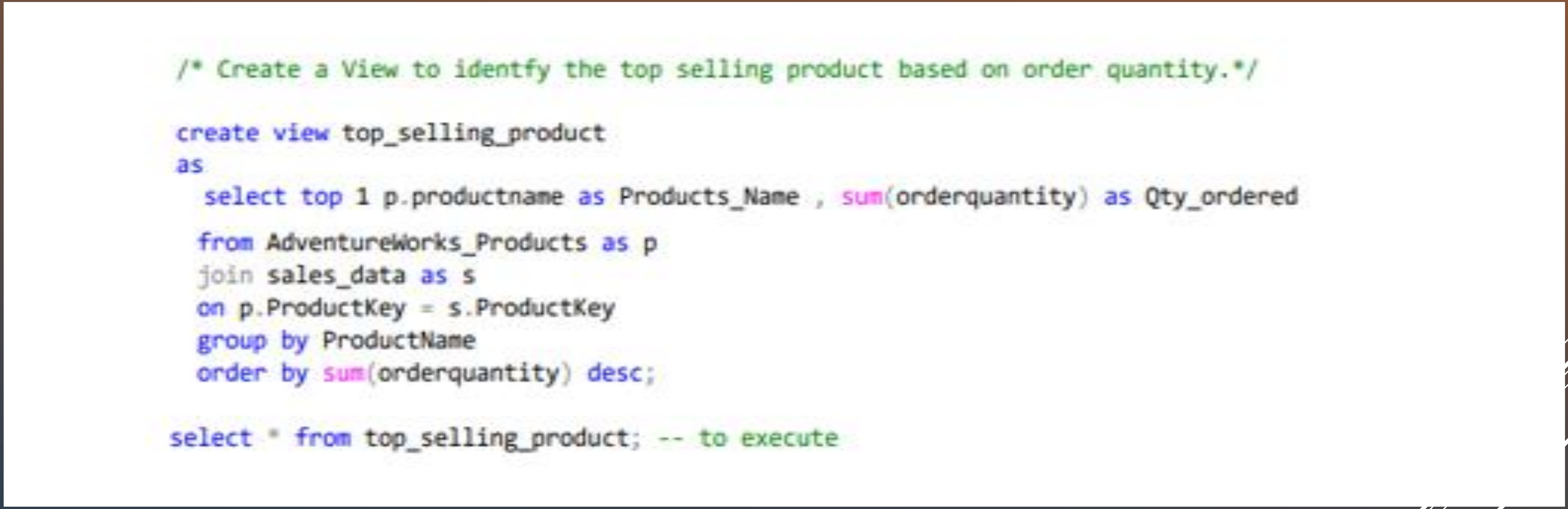


```
/* Find the least selling product category of 2016.*/

select  top 1 CategoryName , sum(s.orderquantity) as Qty_sold
from AdventureWorks_Product_Categories as pc
join AdventureWorks_Product_Subcategories as ps on pc.ProductCategoryKey =
  ps.ProductCategoryKey
join AdventureWorks_Products as p on p.ProductSubcategoryKey =
  ps.ProductSubcategoryKey
join  AdventureWorks_Sales_2016 as s on s.ProductKey = p.ProductKey
group by CategoryName
order by sum(s.orderquantity);
```

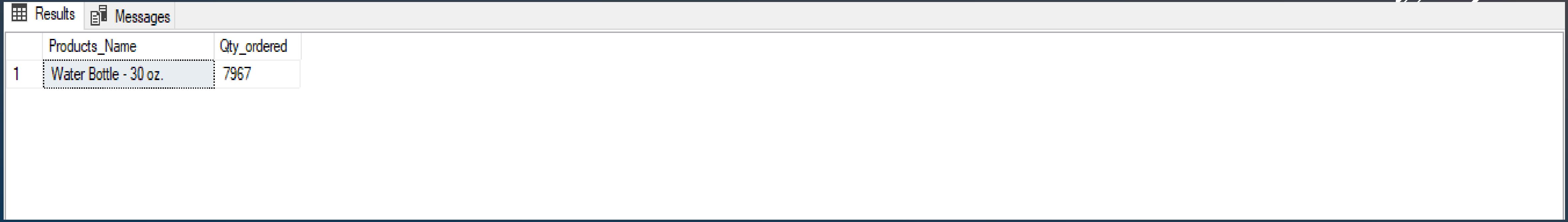


	CategoryName	Qty_sold
1	Clothing	5304

4. Create a View to identify the top-selling product based on order quantity.

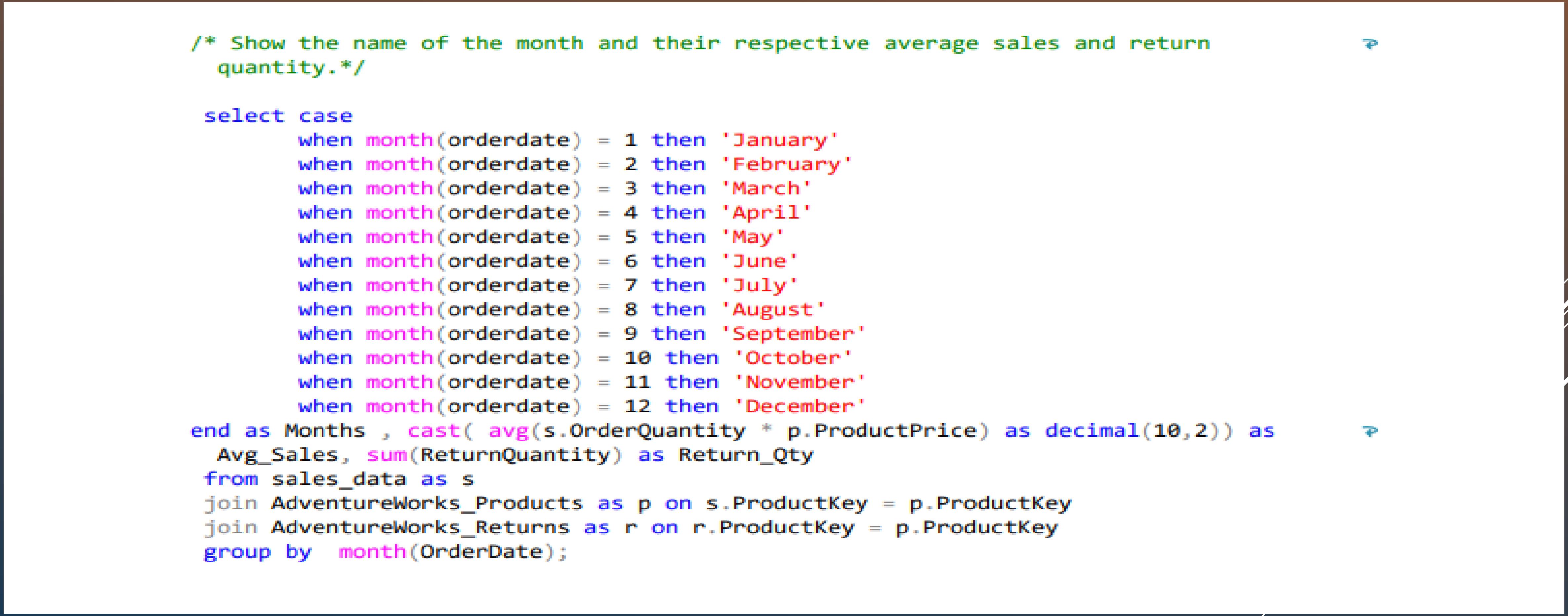


```
/* Create a View to identfy the top selling product based on order quantity.*/

create view top_selling_product
as
  select top 1 p.productname as Products_Name , sum(orderquantity) as Qty_ordered
  from AdventureWorks_Products as p
  join sales_data as s
  on p.ProductKey = s.ProductKey
  group by ProductName
  order by sum(orderquantity) desc;

select * from top_selling_product; -- to execute
```

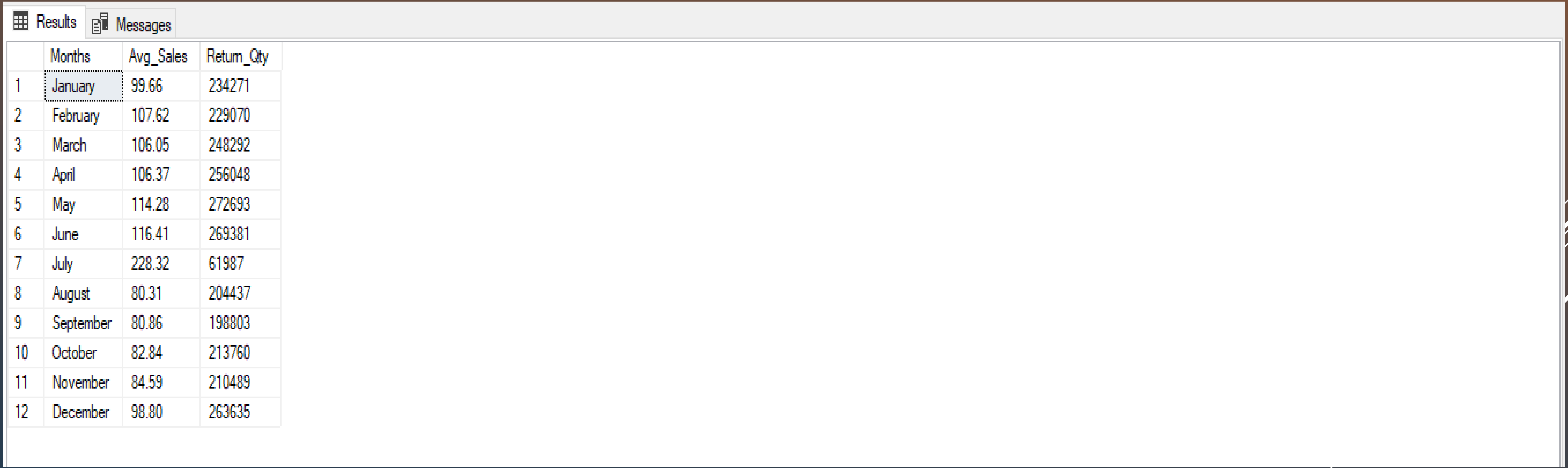


	Products_Name	Qty_ordered
1	Water Bottle - 30 oz.	7967

5. Show the name of the month and their respective average sales and return quantity.



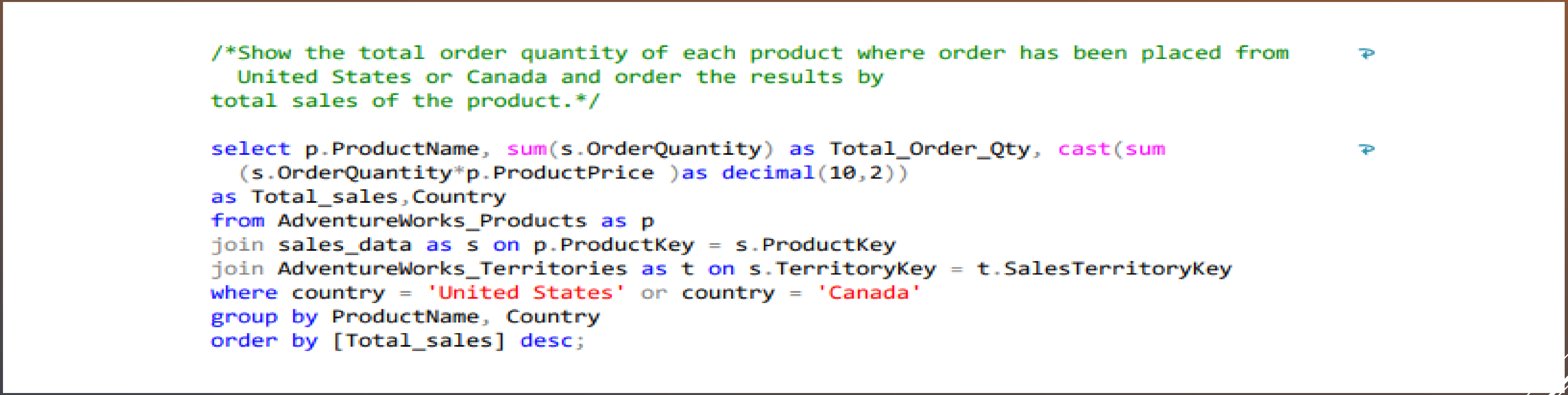
```
/* Show the name of the month and their respective average sales and return
 quantity.*/

 select case
        when month(orderdate) = 1 then 'January'
        when month(orderdate) = 2 then 'February'
        when month(orderdate) = 3 then 'March'
        when month(orderdate) = 4 then 'April'
        when month(orderdate) = 5 then 'May'
        when month(orderdate) = 6 then 'June'
        when month(orderdate) = 7 then 'July'
        when month(orderdate) = 8 then 'August'
        when month(orderdate) = 9 then 'September'
        when month(orderdate) = 10 then 'October'
        when month(orderdate) = 11 then 'November'
        when month(orderdate) = 12 then 'December'
end as Months , cast( avg(s.OrderQuantity * p.ProductPrice) as decimal(10,2)) as
  Avg_Sales, sum(ReturnQuantity) as Return_Qty
 from sales_data as s
 join AdventureWorks_Products as p on s.ProductKey = p.ProductKey
 join AdventureWorks_Returns as r on r.ProductKey = p.ProductKey
 group by  month(OrderDate);
```



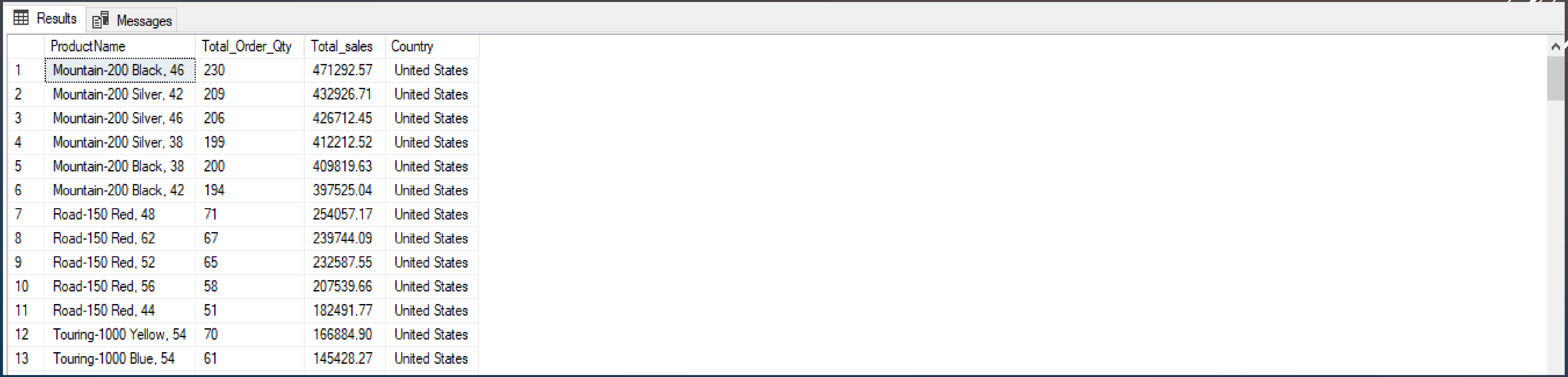
Results | Messages

	Months	Avg_Sales	Return_Qty
1	January	99.66	234271
2	February	107.62	229070
3	March	106.05	248292
4	April	106.37	256048
5	May	114.28	272693
6	June	116.41	269381
7	July	228.32	61987
8	August	80.31	204437
9	September	80.86	198803
10	October	82.84	213760
11	November	84.59	210489
12	December	98.80	263635

6. Show the total order quantity of each product where the order has been placed from the USA and Canada.



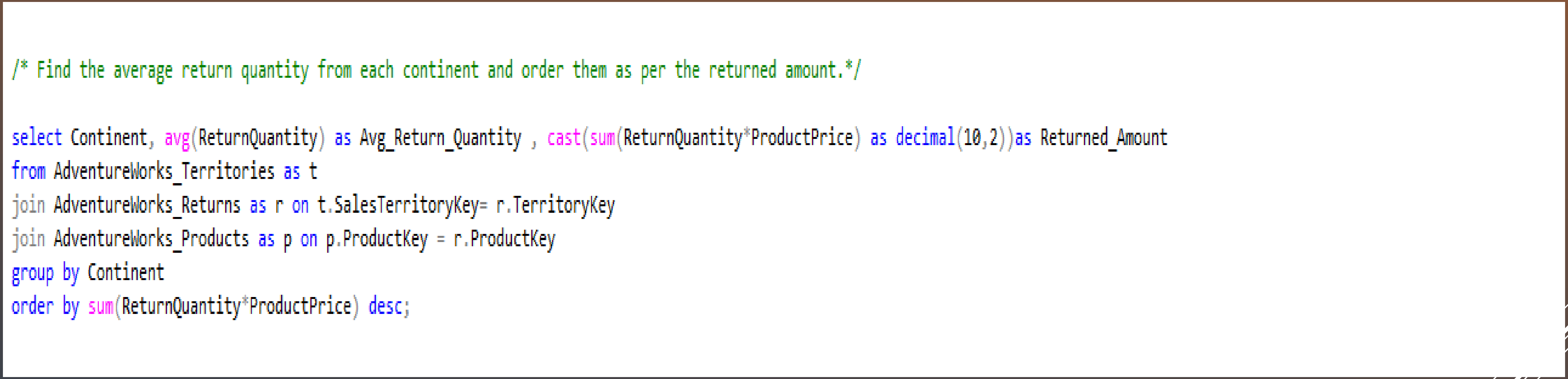
```
/*Show the total order quantity of each product where order has been placed from
  United States or Canada and order the results by
total sales of the product.*/

select p.ProductName, sum(s.OrderQuantity) as Total_Order_Qty, cast(sum
  (s.OrderQuantity*p.ProductPrice )as decimal(10,2))
as Total_sales,Country
from AdventureWorks_Products as p
join sales_data as s on p.ProductKey = s.ProductKey
join AdventureWorks_Territories as t on s.TerritoryKey = t.SalesTerritoryKey
where country = 'United States' or country = 'Canada'
group by ProductName, Country
order by [Total_sales] desc;
```



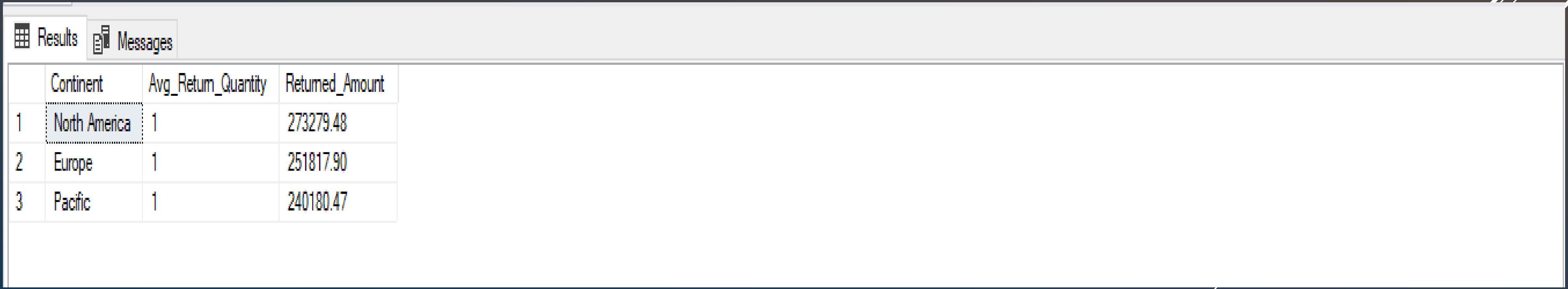
Results | Messages

	ProductName	Total_Order_Qty	Total_sales	Country
1	Mountain-200 Black, 46	230	471292.57	United States
2	Mountain-200 Silver, 42	209	432926.71	United States
3	Mountain-200 Silver, 46	206	426712.45	United States
4	Mountain-200 Silver, 38	199	412212.52	United States
5	Mountain-200 Black, 38	200	409819.63	United States
6	Mountain-200 Black, 42	194	397525.04	United States
7	Road-150 Red, 48	71	254057.17	United States
8	Road-150 Red, 62	67	239744.09	United States
9	Road-150 Red, 52	65	232587.55	United States
10	Road-150 Red, 56	58	207539.66	United States
11	Road-150 Red, 44	51	182491.77	United States
12	Touring-1000 Yellow, 54	70	166884.90	United States
13	Touring-1000 Blue, 54	61	145428.27	United States

7. Find the average return quantity from each continent and order them as per returned amount.

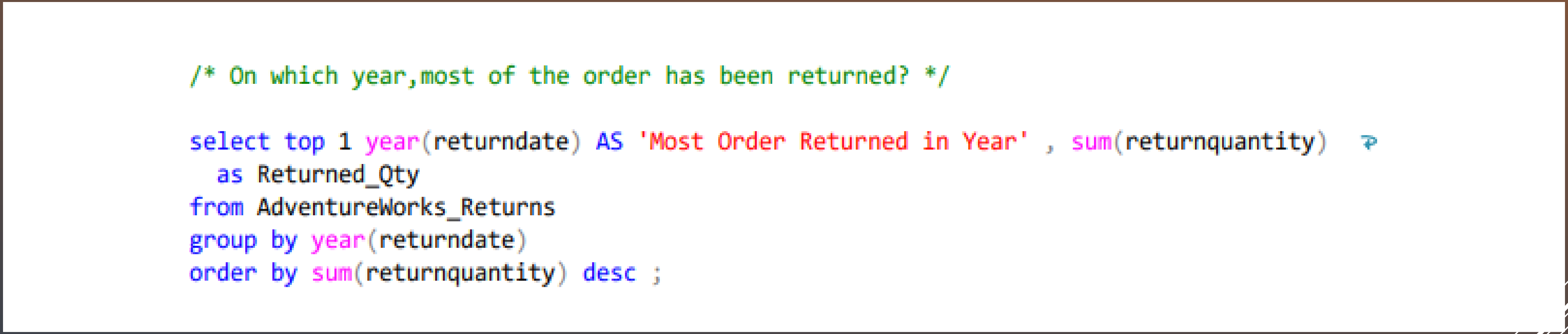


```sql
/* Find the average return quantity from each continent and order them as per the returned amount.*/

select Continent, avg(ReturnQuantity) as Avg_Return_Quantity , cast(sum(ReturnQuantity*ProductPrice) as decimal(10,2))as Returned_Amount
from AdventureWorks_Territories as t
join AdventureWorks_Returns as r on t.SalesTerritoryKey= r.TerritoryKey
join AdventureWorks_Products as p on p.ProductKey = r.ProductKey
group by Continent
order by sum(ReturnQuantity*ProductPrice) desc;
```

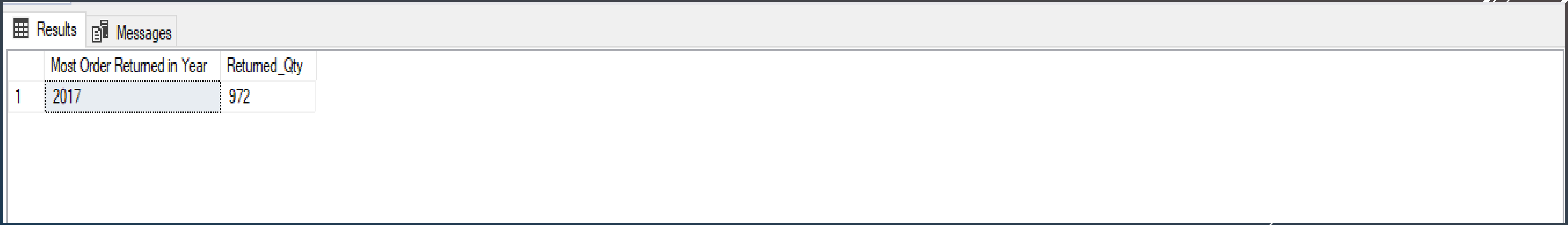


	Continent	Avg_Return_Quantity	Returned_Amount
1	North America	1	273279.48
2	Europe	1	251817.90
3	Pacific	1	240180.47

8. On which year most of the order has been returned?



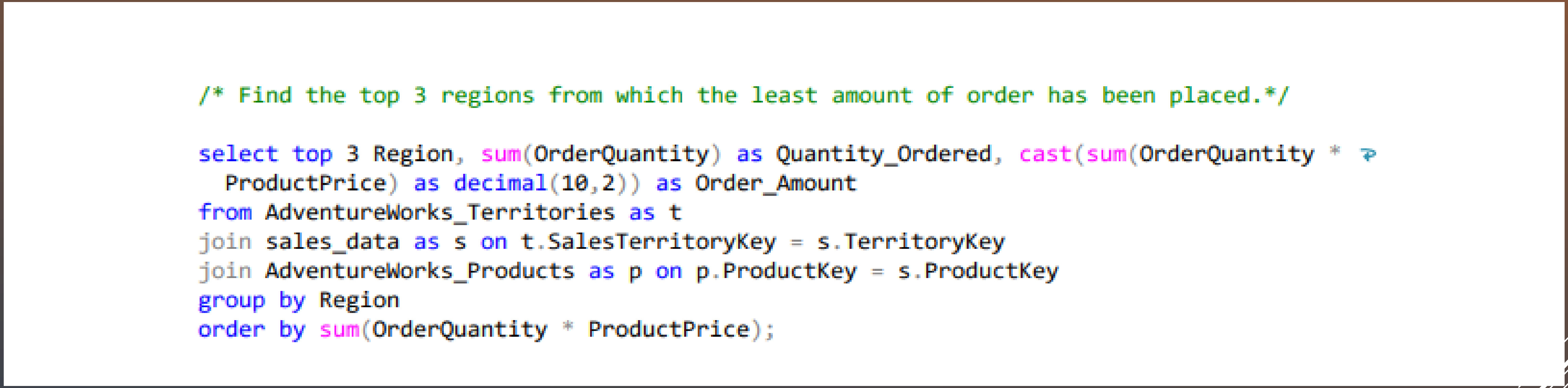
```
/* On which year,most of the order has been returned? */

select top 1 year(returndate) AS 'Most Order Returned in Year' , sum(returnquantity)
  as Returned_Qty
from AdventureWorks_Returns
group by year(returndate)
order by sum(returnquantity) desc ;
```



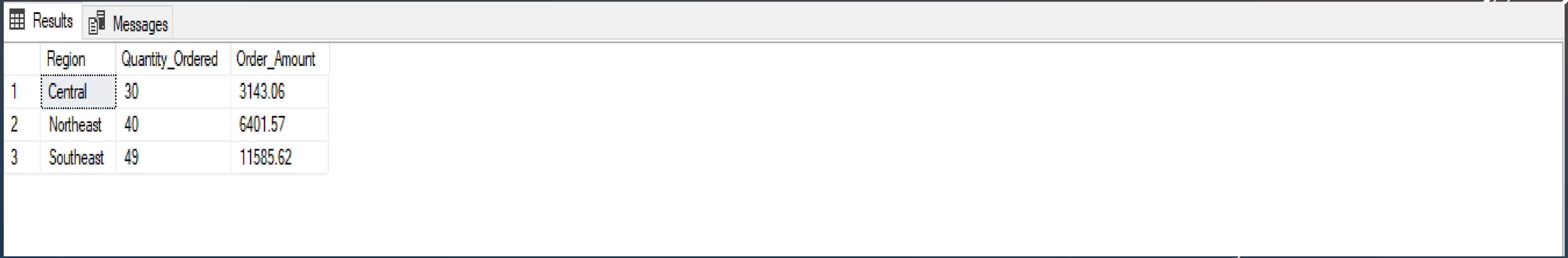
Results | Messages

	Most Order Returned in Year	Returned_Qty
1	2017	972

9. Find the top 3 regions from which the least amount of order has been placed.

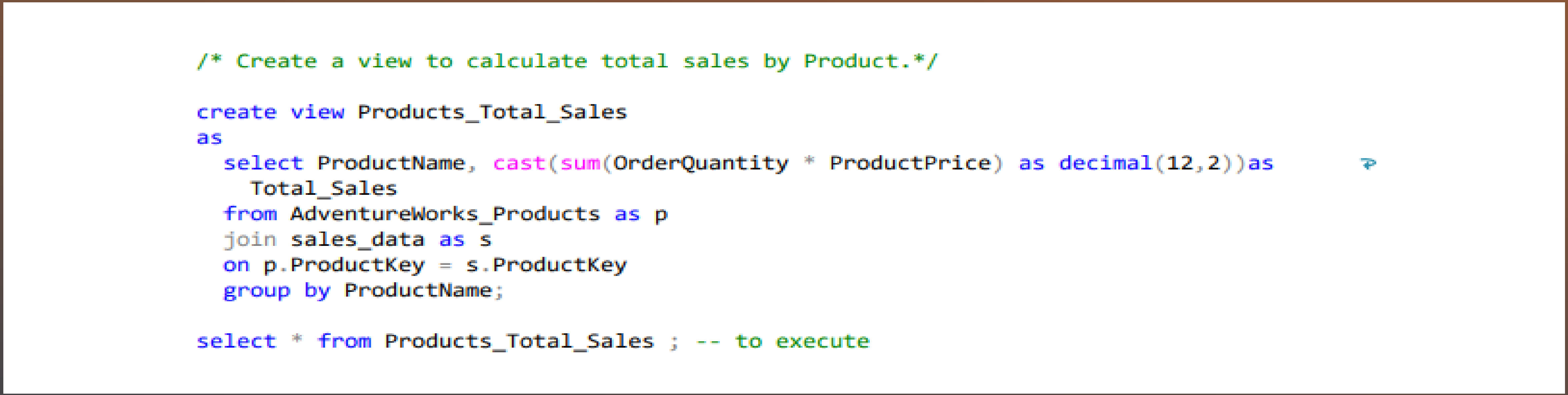


```
/* Find the top 3 regions from which the least amount of order has been placed.*/

select top 3 Region, sum(OrderQuantity) as Quantity_Ordered, cast(sum(OrderQuantity * 
  ProductPrice) as decimal(10,2)) as Order_Amount
from AdventureWorks_Territories as t
join sales_data as s on t.SalesTerritoryKey = s.TerritoryKey
join AdventureWorks_Products as p on p.ProductKey = s.ProductKey
group by Region
order by sum(OrderQuantity * ProductPrice);
```

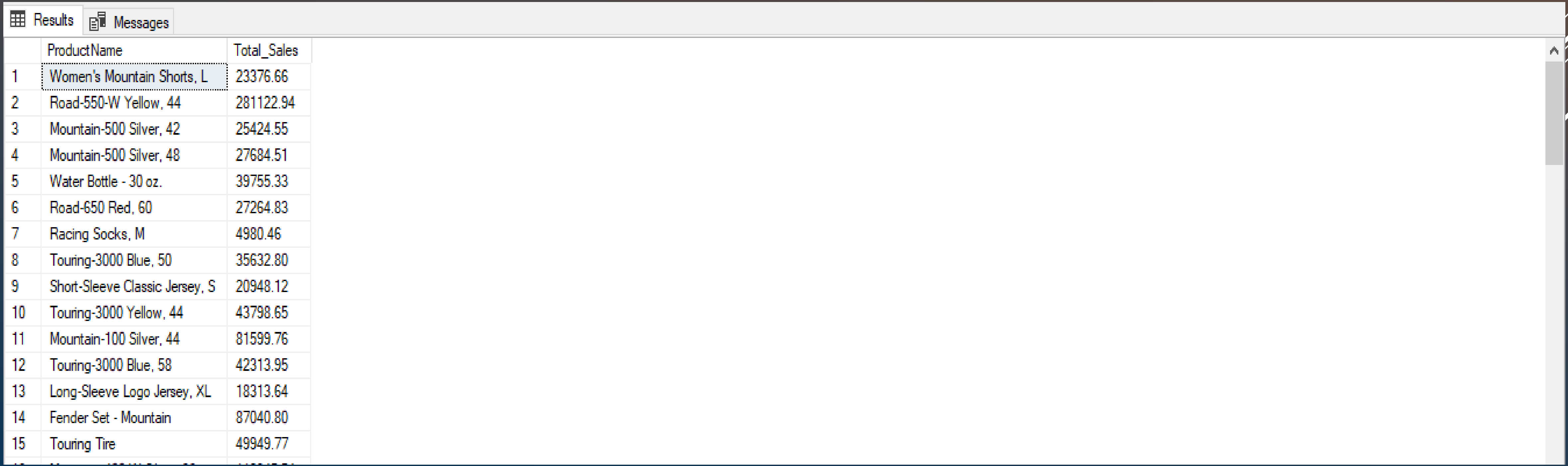


	Region	Quantity_Ordered	Order_Amount
1	Central	30	3143.06
2	Northeast	40	6401.57
3	Southeast	49	11585.62

10. Create a view to calculate total sales by product.

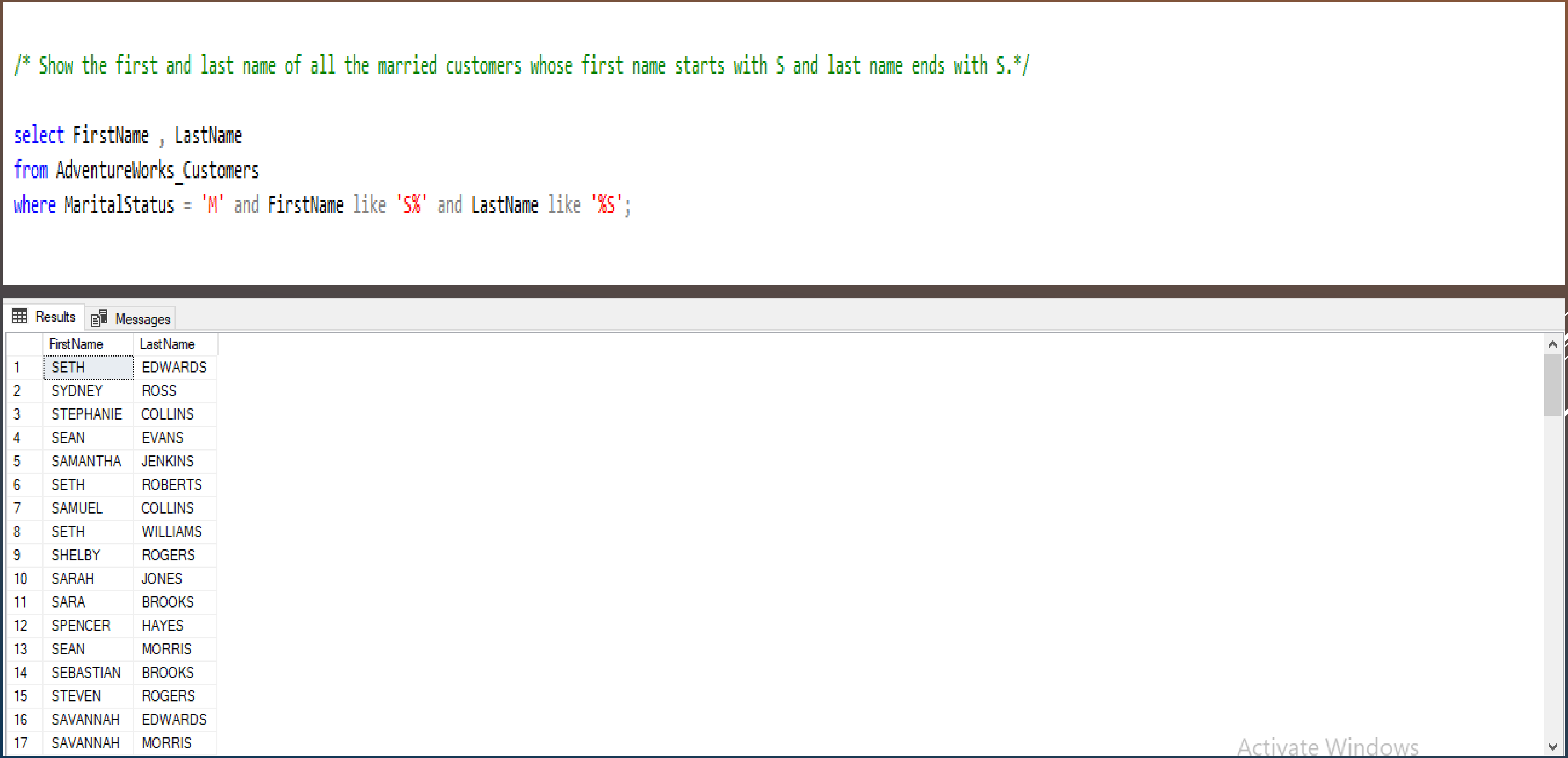


```sql
/* Create a view to calculate total sales by Product.*/

create view Products_Total_Sales
as
  select ProductName, cast(sum(OrderQuantity * ProductPrice) as decimal(12,2))as
    Total_Sales
  from AdventureWorks_Products as p
  join sales_data as s
  on p.ProductKey = s.ProductKey
  group by ProductName;

select * from Products_Total_Sales ; -- to execute
```



	ProductName	Total_Sales
1	Women's Mountain Shorts, L	23376.66
2	Road-550-W Yellow, 44	281122.94
3	Mountain-500 Silver, 42	25424.55
4	Mountain-500 Silver, 48	27684.51
5	Water Bottle - 30 oz.	39755.33
6	Road-650 Red, 60	27264.83
7	Racing Socks, M	4980.46
8	Touring-3000 Blue, 50	35632.80
9	Short-Sleeve Classic Jersey, S	20948.12
10	Touring-3000 Yellow, 44	43798.65
11	Mountain-100 Silver, 44	81599.76
12	Touring-3000 Blue, 58	42313.95
13	Long-Sleeve Logo Jersey, XL	18313.64
14	Fender Set - Mountain	87040.80
15	Touring Tire	49949.77

11. Show the first and last name of all the married customers whose name starts with S and last name ends with S.

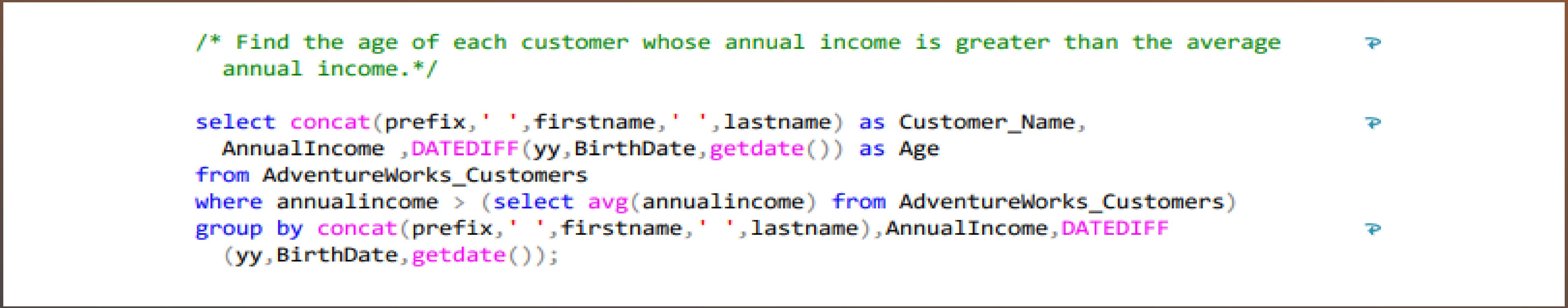


```
/* Show the first and last name of all the married customers whose first name starts with S and last name ends with S.*/

select FirstName , LastName
from AdventureWorks_Customers
where MaritalStatus = 'M' and FirstName like 'S%' and LastName like '%S';
```

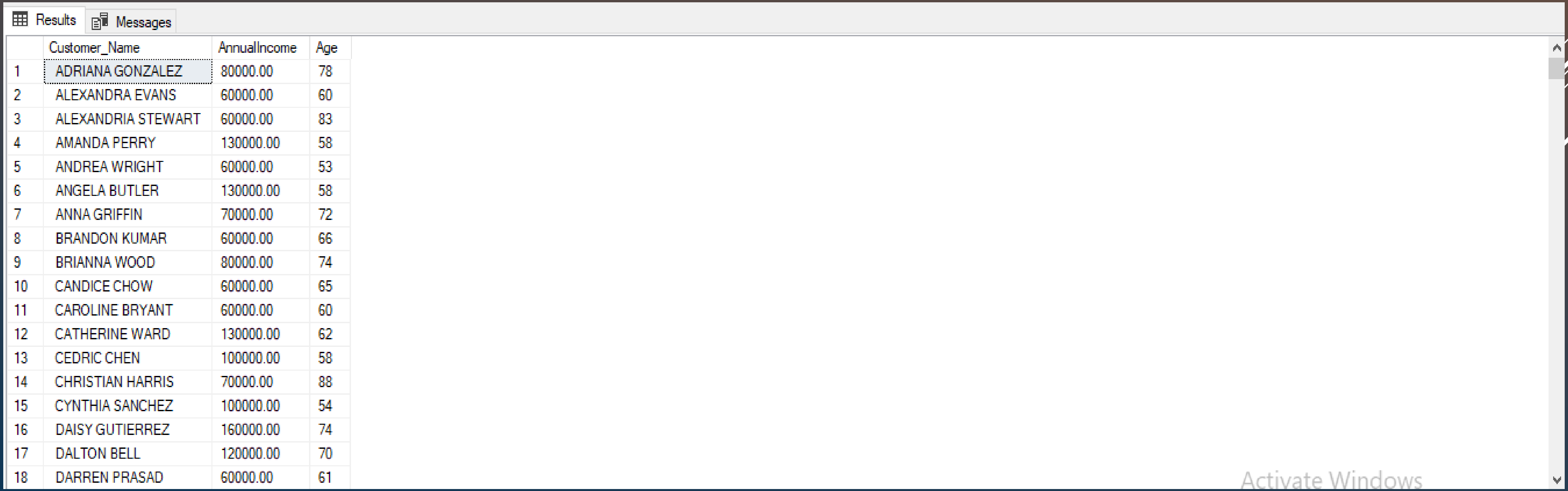
Results Messages

	FirstName	LastName
1	SETH	EDWARDS
2	SYDNEY	ROSS
3	STEPHANIE	COLLINS
4	SEAN	EVANS
5	SAMANTHA	JENKINS
6	SETH	ROBERTS
7	SAMUEL	COLLINS
8	SETH	WILLIAMS
9	SHELBY	ROGERS
10	SARAH	JONES
11	SARA	BROOKS
12	SPENCER	HAYES
13	SEAN	MORRIS
14	SEBASTIAN	BROOKS
15	STEVEN	ROGERS
16	SAVANNAH	EDWARDS
17	SAVANNAH	MORRIS

12. Find the age of each customer whose annual income is greater than the average annual income.



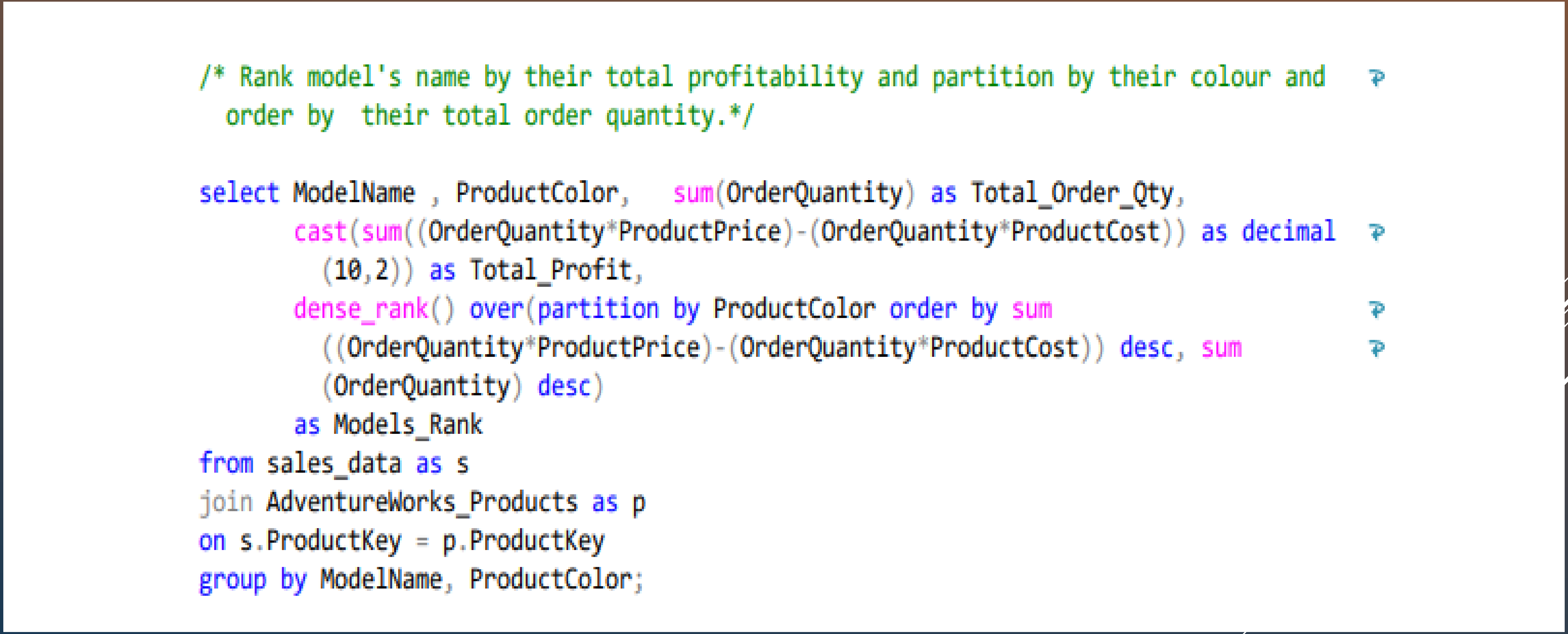
```
/* Find the age of each customer whose annual income is greater than the average
  annual income.*/

select concat(prefix,' ',firstname,' ',lastname) as Customer_Name,
  AnnualIncome ,DATEDIFF(yy,BirthDate,getdate()) as Age
from AdventureWorks_Customers
where annualincome > (select avg(annualincome) from AdventureWorks_Customers)
group by concat(prefix,' ',firstname,' ',lastname),AnnualIncome,DATEDIFF
  (yy,BirthDate,getdate());
```



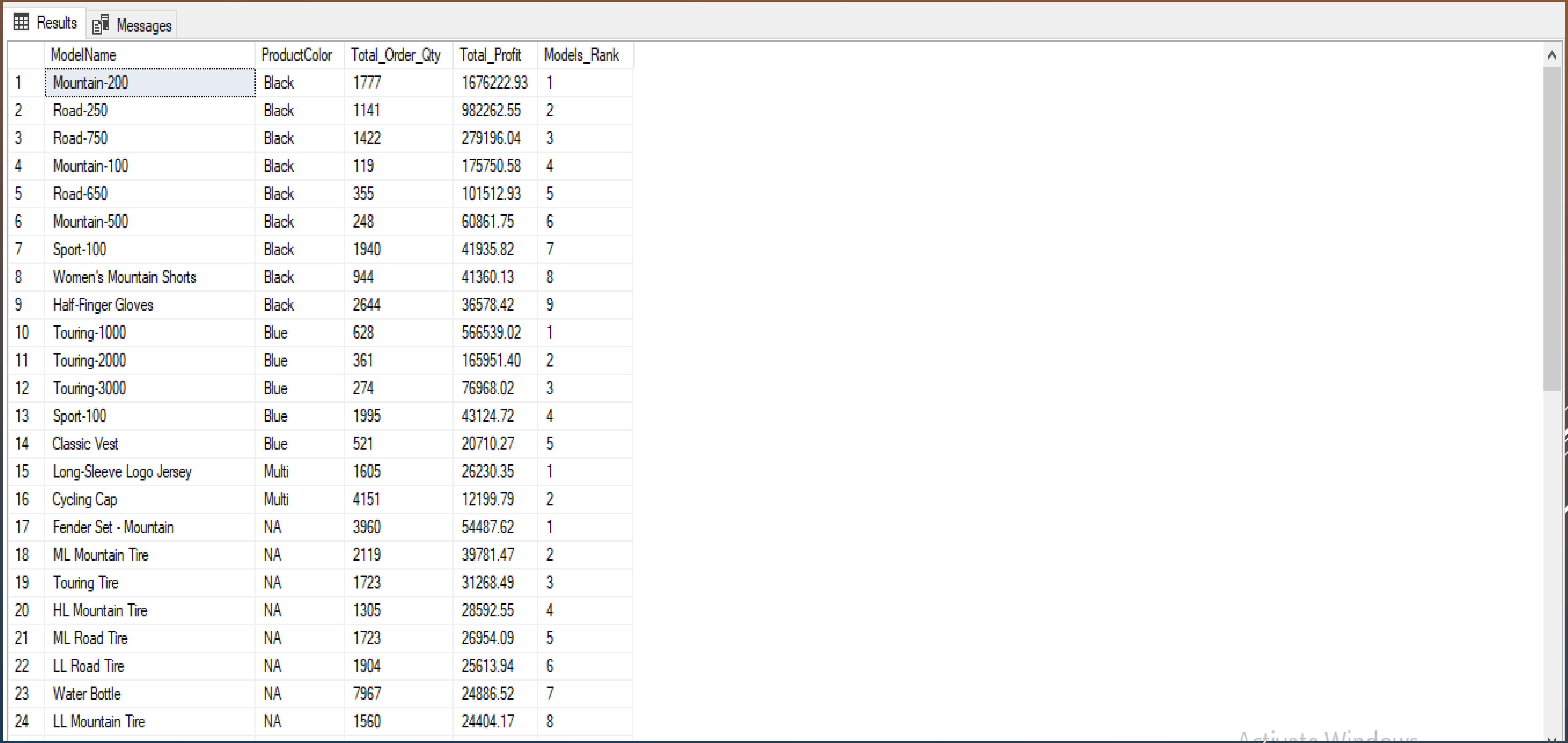
Results Messages

	Customer_Name	AnnualIncome	Age
1	ADRIANA GONZALEZ	80000.00	78
2	ALEXANDRA EVANS	60000.00	60
3	ALEXANDRIA STEWART	60000.00	83
4	AMANDA PERRY	130000.00	58
5	ANDREA WRIGHT	60000.00	53
6	ANGELA BUTLER	130000.00	58
7	ANNA GRIFFIN	70000.00	72
8	BRANDON KUMAR	60000.00	66
9	BRIANNA WOOD	80000.00	74
10	CANDICE CHOW	60000.00	65
11	CAROLINE BRYANT	60000.00	60
12	CATHERINE WARD	130000.00	62
13	CEDRIC CHEN	100000.00	58
14	CHRISTIAN HARRIS	70000.00	88
15	CYNTHIA SANCHEZ	100000.00	54
16	DAISY GUTIERREZ	160000.00	74
17	DALTON BELL	120000.00	70
18	DARREN PRASAD	60000.00	61

13. Rank the Model's name by their total profitability and partition by their color and order by their total order quantity.

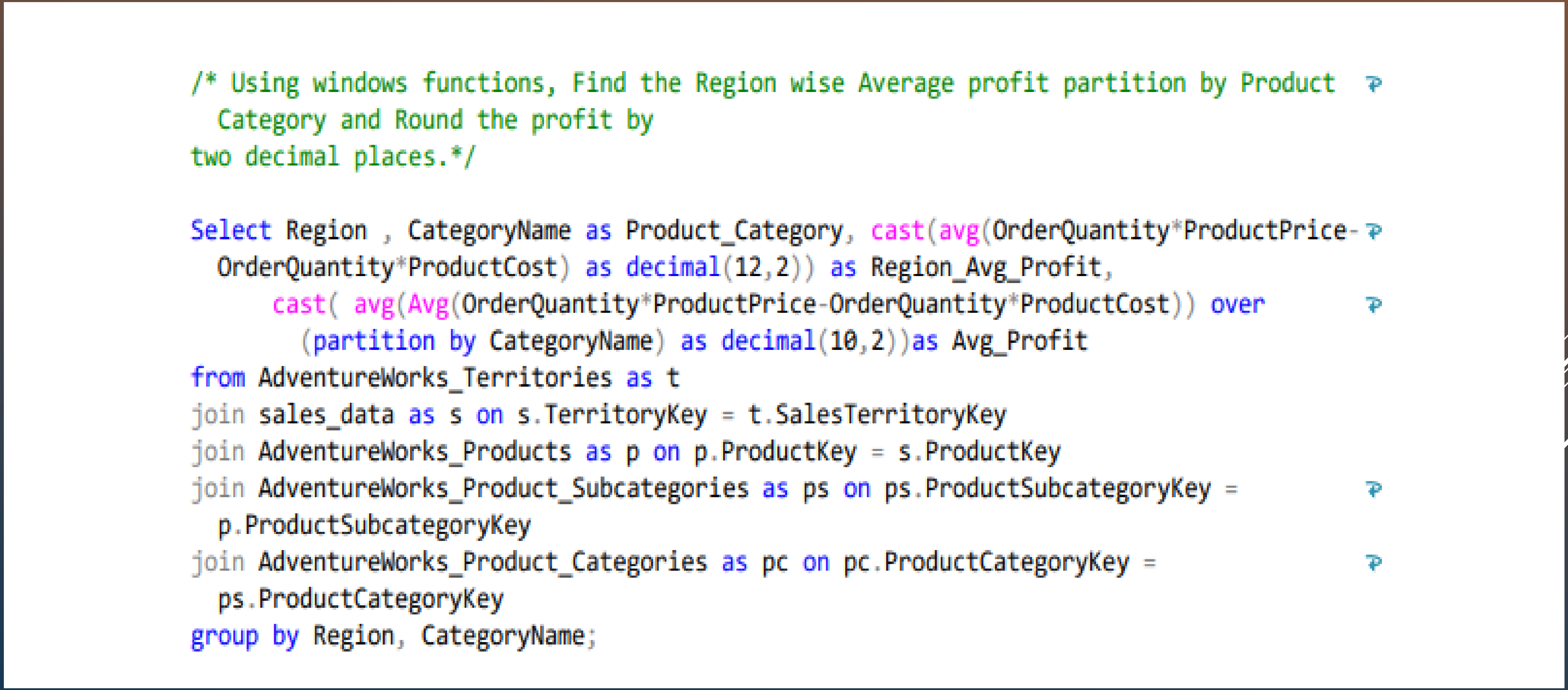


```
/* Rank model's name by their total profitability and partition by their colour and
  order by  their total order quantity.*/

select ModelName , ProductColor,   sum(OrderQuantity) as Total_Order_Qty,
       cast(sum((OrderQuantity*ProductPrice)-(OrderQuantity*ProductCost)) as decimal
         (10,2)) as Total_Profit,
       dense_rank() over(partition by ProductColor order by sum
         ((OrderQuantity*ProductPrice)-(OrderQuantity*ProductCost)) desc, sum
         (OrderQuantity) desc)
       as Models_Rank
from sales_data as s
join AdventureWorks_Products as p
on s.ProductKey = p.ProductKey
group by ModelName, ProductColor;
```

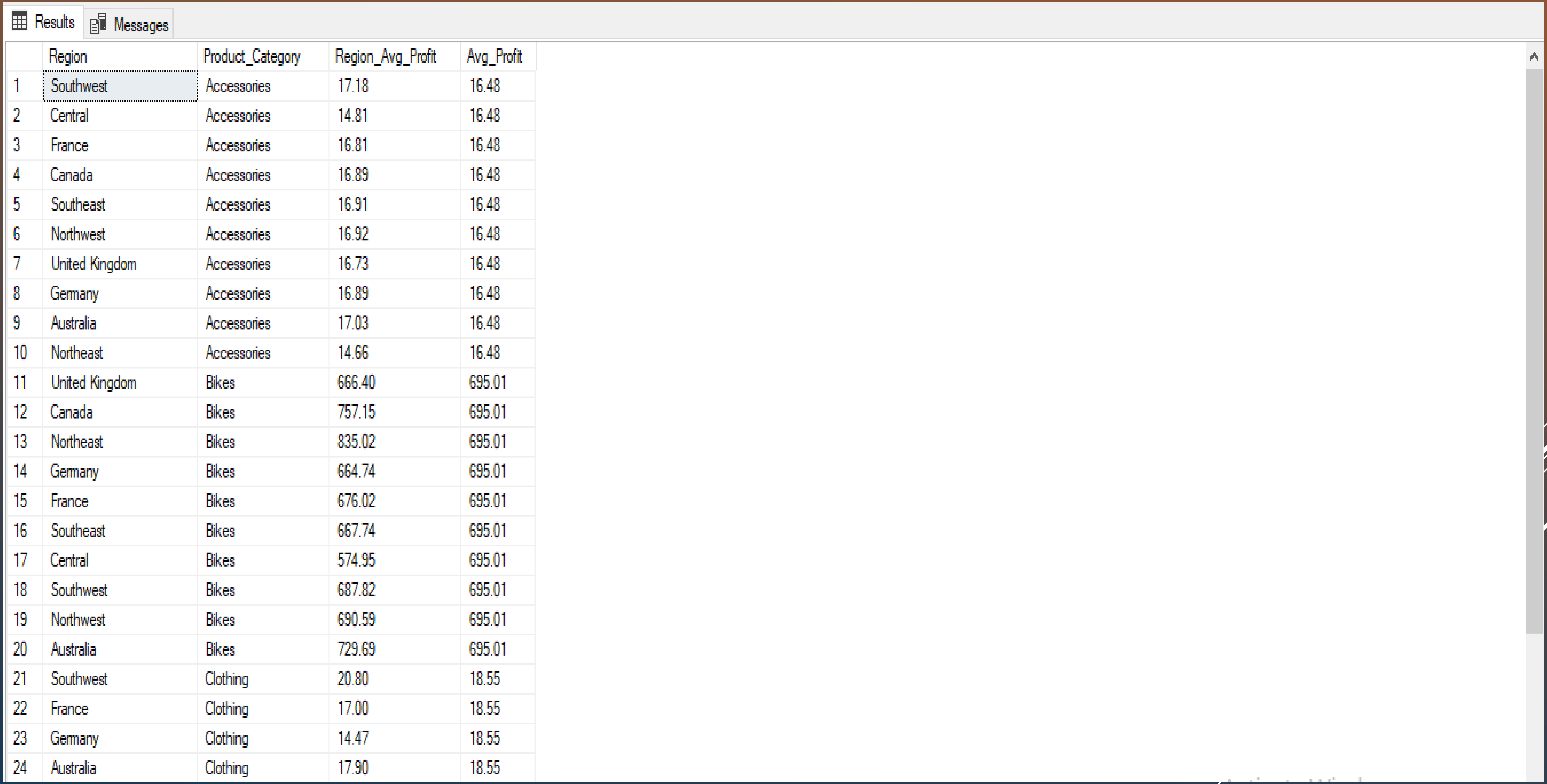


	ModelName	ProductColor	Total_Order_Qty	Total_Profit	Models_Rank
1	Mountain-200	Black	1777	1676222.93	1
2	Road-250	Black	1141	982262.55	2
3	Road-750	Black	1422	279196.04	3
4	Mountain-100	Black	119	175750.58	4
5	Road-650	Black	355	101512.93	5
6	Mountain-500	Black	248	60861.75	6
7	Sport-100	Black	1940	41935.82	7
8	Women's Mountain Shorts	Black	944	41360.13	8
9	Half-Finger Gloves	Black	2644	36578.42	9
10	Touring-1000	Blue	628	566539.02	1
11	Touring-2000	Blue	361	165951.40	2
12	Touring-3000	Blue	274	76968.02	3
13	Sport-100	Blue	1995	43124.72	4
14	Classic Vest	Blue	521	20710.27	5
15	Long-Sleeve Logo Jersey	Multi	1605	26230.35	1
16	Cycling Cap	Multi	4151	12199.79	2
17	Fender Set - Mountain	NA	3960	54487.62	1
18	ML Mountain Tire	NA	2119	39781.47	2
19	Touring Tire	NA	1723	31268.49	3
20	HL Mountain Tire	NA	1305	28592.55	4
21	ML Road Tire	NA	1723	26954.09	5
22	LL Road Tire	NA	1904	25613.94	6
23	Water Bottle	NA	7967	24886.52	7
24	LL Mountain Tire	NA	1560	24404.17	8

14. Using the Windows function, Find the region-wise average profit partition by Product Category and round the profit by two decimal places.

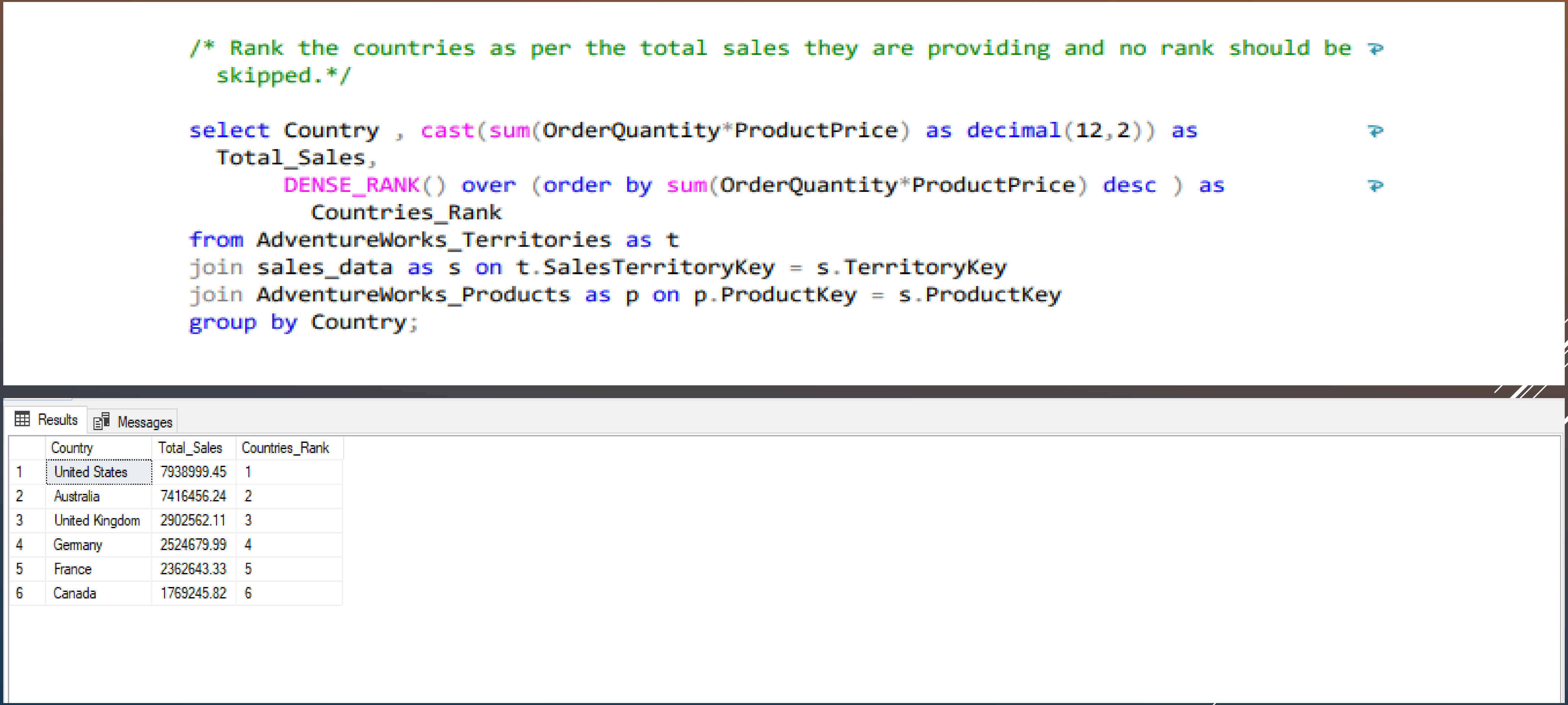


```
/* Using windows functions, Find the Region wise Average profit partition by Product
  Category and Round the profit by
two decimal places.*/

Select Region , CategoryName as Product_Category, cast(avg(OrderQuantity*ProductPrice-
  OrderQuantity*ProductCost) as decimal(12,2)) as Region_Avg_Profit,
      cast( avg(Avg(OrderQuantity*ProductPrice-OrderQuantity*ProductCost)) over
        (partition by CategoryName) as decimal(10,2))as Avg_Profit
from AdventureWorks_Territories as t
join sales_data as s on s.TerritoryKey = t.SalesTerritoryKey
join AdventureWorks_Products as p on p.ProductKey = s.ProductKey
join AdventureWorks_Product_Subcategories as ps on ps.ProductSubcategoryKey =
  p.ProductSubcategoryKey
join AdventureWorks_Product_Categories as pc on pc.ProductCategoryKey =
  ps.ProductCategoryKey
group by Region, CategoryName;
```



	Region	Product_Category	Region_Avg_Profit	Avg_Profit
1	Southwest	Accessories	17.18	16.48
2	Central	Accessories	14.81	16.48
3	France	Accessories	16.81	16.48
4	Canada	Accessories	16.89	16.48
5	Southeast	Accessories	16.91	16.48
6	Northwest	Accessories	16.92	16.48
7	United Kingdom	Accessories	16.73	16.48
8	Germany	Accessories	16.89	16.48
9	Australia	Accessories	17.03	16.48
10	Northeast	Accessories	14.66	16.48
11	United Kingdom	Bikes	666.40	695.01
12	Canada	Bikes	757.15	695.01
13	Northeast	Bikes	835.02	695.01
14	Germany	Bikes	664.74	695.01
15	France	Bikes	676.02	695.01
16	Southeast	Bikes	667.74	695.01
17	Central	Bikes	574.95	695.01
18	Southwest	Bikes	687.82	695.01
19	Northwest	Bikes	690.59	695.01
20	Australia	Bikes	729.69	695.01
21	Southwest	Clothing	20.80	18.55
22	France	Clothing	17.00	18.55
23	Germany	Clothing	14.47	18.55
24	Australia	Clothing	17.90	18.55

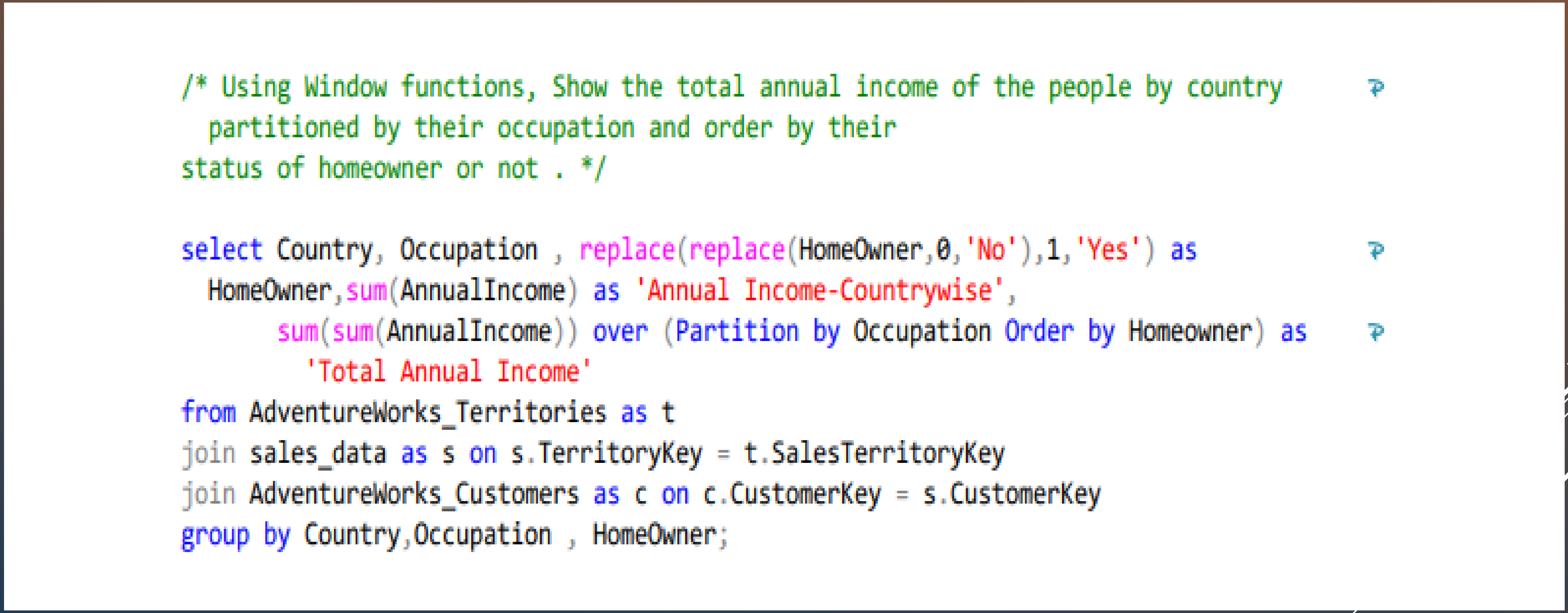
15. Rank the countries as per the total sales they are providing and no rank should be skipped.



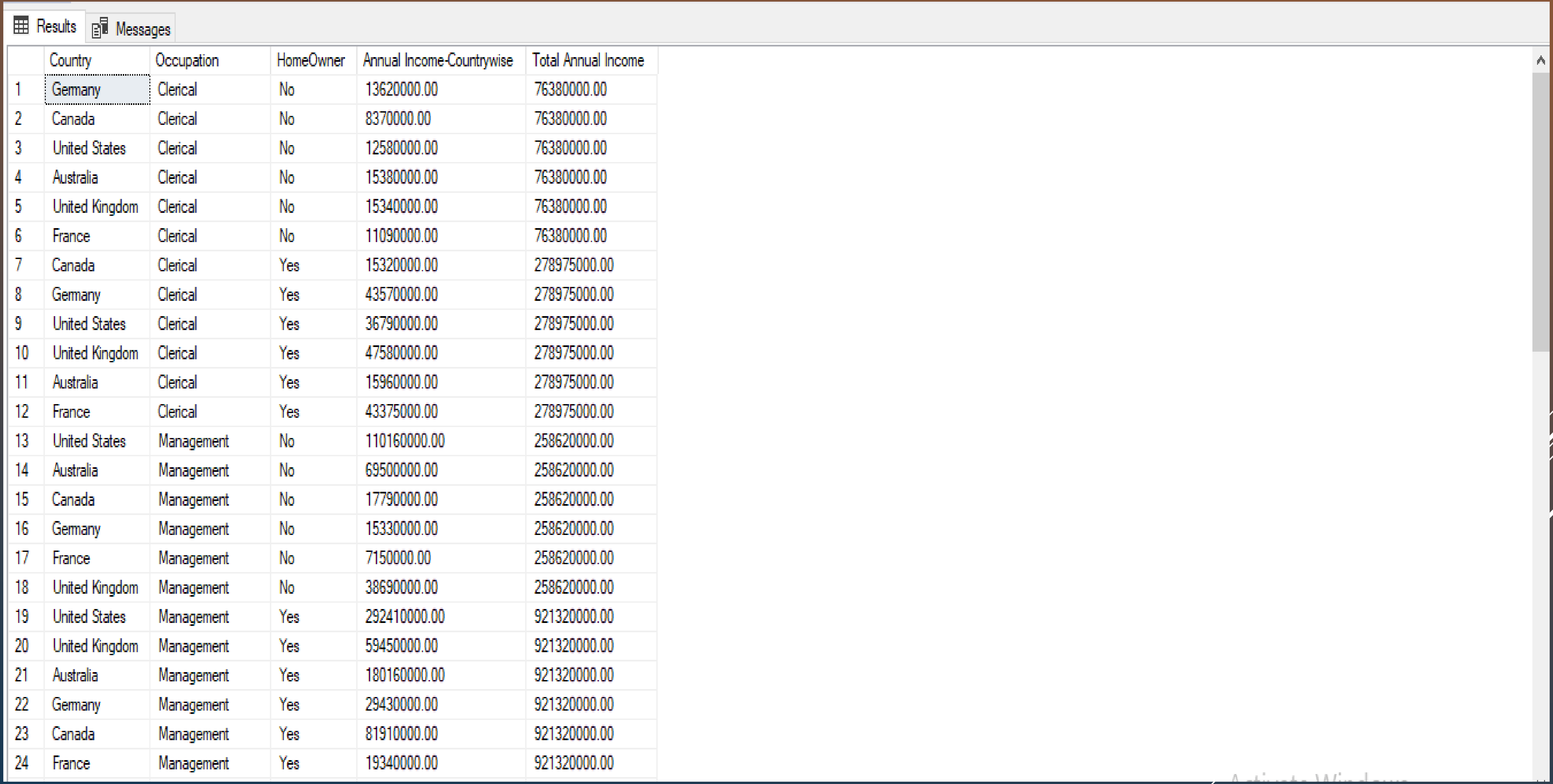
```
/* Rank the countries as per the total sales they are providing and no rank should be
  skipped.*/

select Country , cast(sum(OrderQuantity*ProductPrice) as decimal(12,2)) as
  Total_Sales,
        DENSE_RANK() over (order by sum(OrderQuantity*ProductPrice) desc ) as
          Countries_Rank
from AdventureWorks_Territories as t
join sales_data as s on t.SalesTerritoryKey = s.TerritoryKey
join AdventureWorks_Products as p on p.ProductKey = s.ProductKey
group by Country;
```

	Country	Total_Sales	Countries_Rank
1	United States	7938999.45	1
2	Australia	7416456.24	2
3	United Kingdom	2902562.11	3
4	Germany	2524679.99	4
5	France	2362643.33	5
6	Canada	1769245.82	6

16. Using Windows functions, Show the total annual income of the people by country partitioned by their occupation and order by their status of homeowner or not.

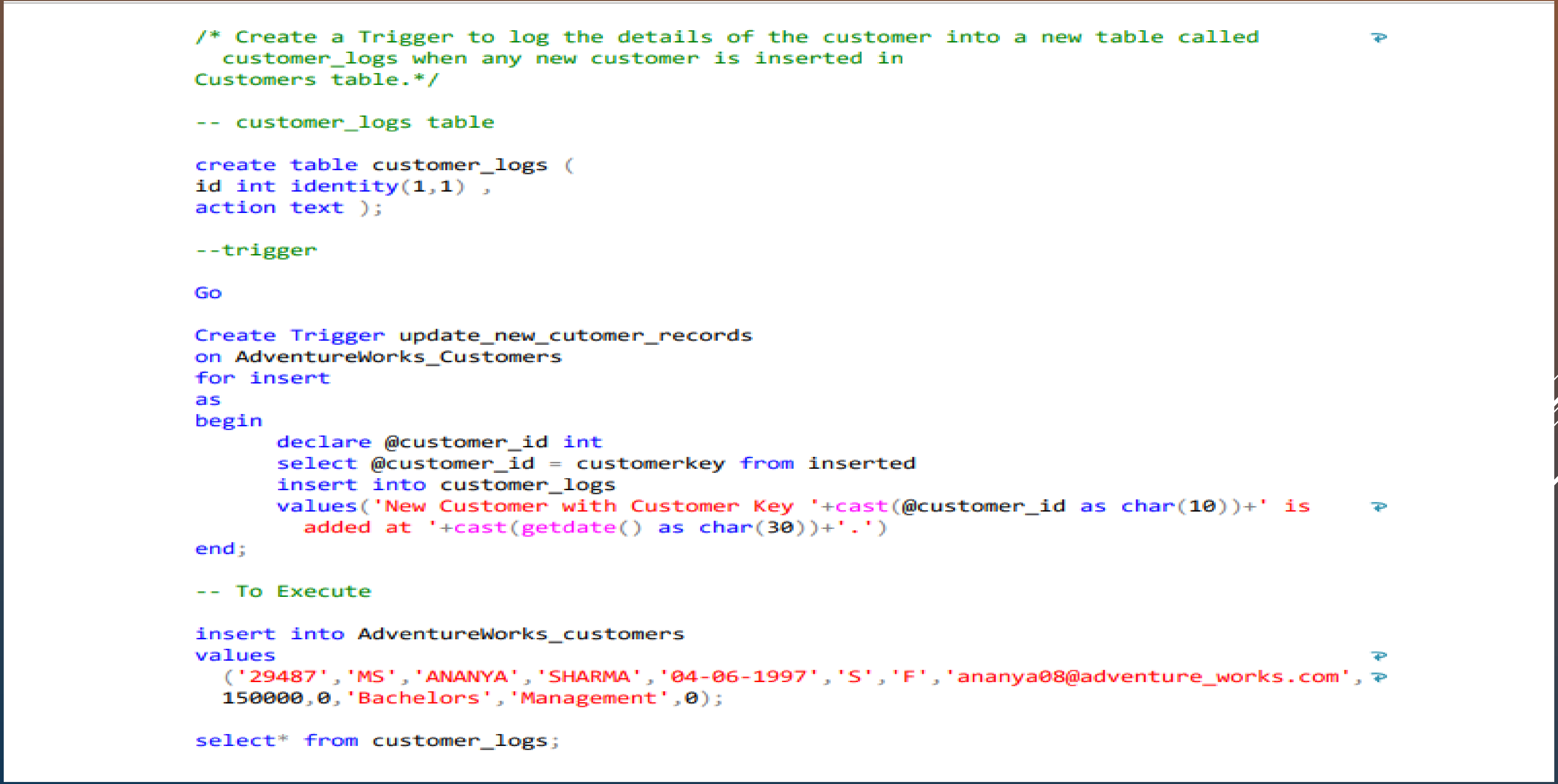


```
/* Using Window functions, Show the total annual income of the people by country
  partitioned by their occupation and order by their
status of homeowner or not . */

select Country, Occupation , replace(replace(HomeOwner,0,'No'),1,'Yes') as
  HomeOwner,sum(AnnualIncome) as 'Annual Income-Countrywise',
       sum(sum(AnnualIncome)) over (Partition by Occupation Order by Homeowner) as
         'Total Annual Income'
from AdventureWorks_Territories as t
join sales_data as s on s.TerritoryKey = t.SalesTerritoryKey
join AdventureWorks_Customers as c on c.CustomerKey = s.CustomerKey
group by Country,Occupation , HomeOwner;
```

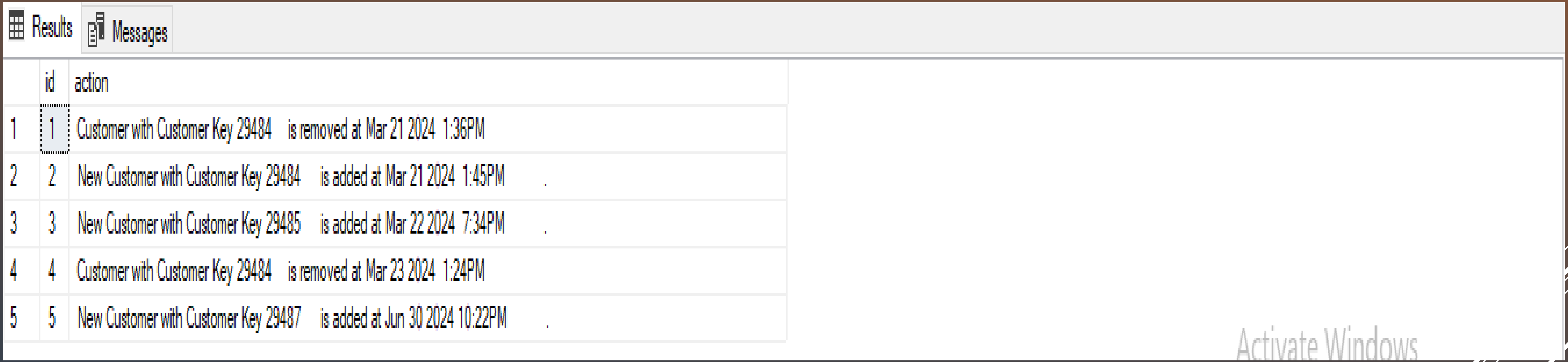

Results Messages

	Country	Occupation	HomeOwner	Annual Income-Countrywise	Total Annual Income
1	Germany	Clerical	No	13620000.00	76380000.00
2	Canada	Clerical	No	8370000.00	76380000.00
3	United States	Clerical	No	12580000.00	76380000.00
4	Australia	Clerical	No	15380000.00	76380000.00
5	United Kingdom	Clerical	No	15340000.00	76380000.00
6	France	Clerical	No	11090000.00	76380000.00
7	Canada	Clerical	Yes	15320000.00	278975000.00
8	Germany	Clerical	Yes	43570000.00	278975000.00
9	United States	Clerical	Yes	36790000.00	278975000.00
10	United Kingdom	Clerical	Yes	47580000.00	278975000.00
11	Australia	Clerical	Yes	15960000.00	278975000.00
12	France	Clerical	Yes	43375000.00	278975000.00
13	United States	Management	No	110160000.00	258620000.00
14	Australia	Management	No	69500000.00	258620000.00
15	Canada	Management	No	17790000.00	258620000.00
16	Germany	Management	No	15330000.00	258620000.00
17	France	Management	No	7150000.00	258620000.00
18	United Kingdom	Management	No	38690000.00	258620000.00
19	United States	Management	Yes	292410000.00	921320000.00
20	United Kingdom	Management	Yes	59450000.00	921320000.00
21	Australia	Management	Yes	180160000.00	921320000.00
22	Germany	Management	Yes	29430000.00	921320000.00
23	Canada	Management	Yes	81910000.00	921320000.00
24	France	Management	Yes	19340000.00	921320000.00

17. Create a trigger to log the details of the customer into a new table called customer_logs when any new customer is inserted in the Customer table.



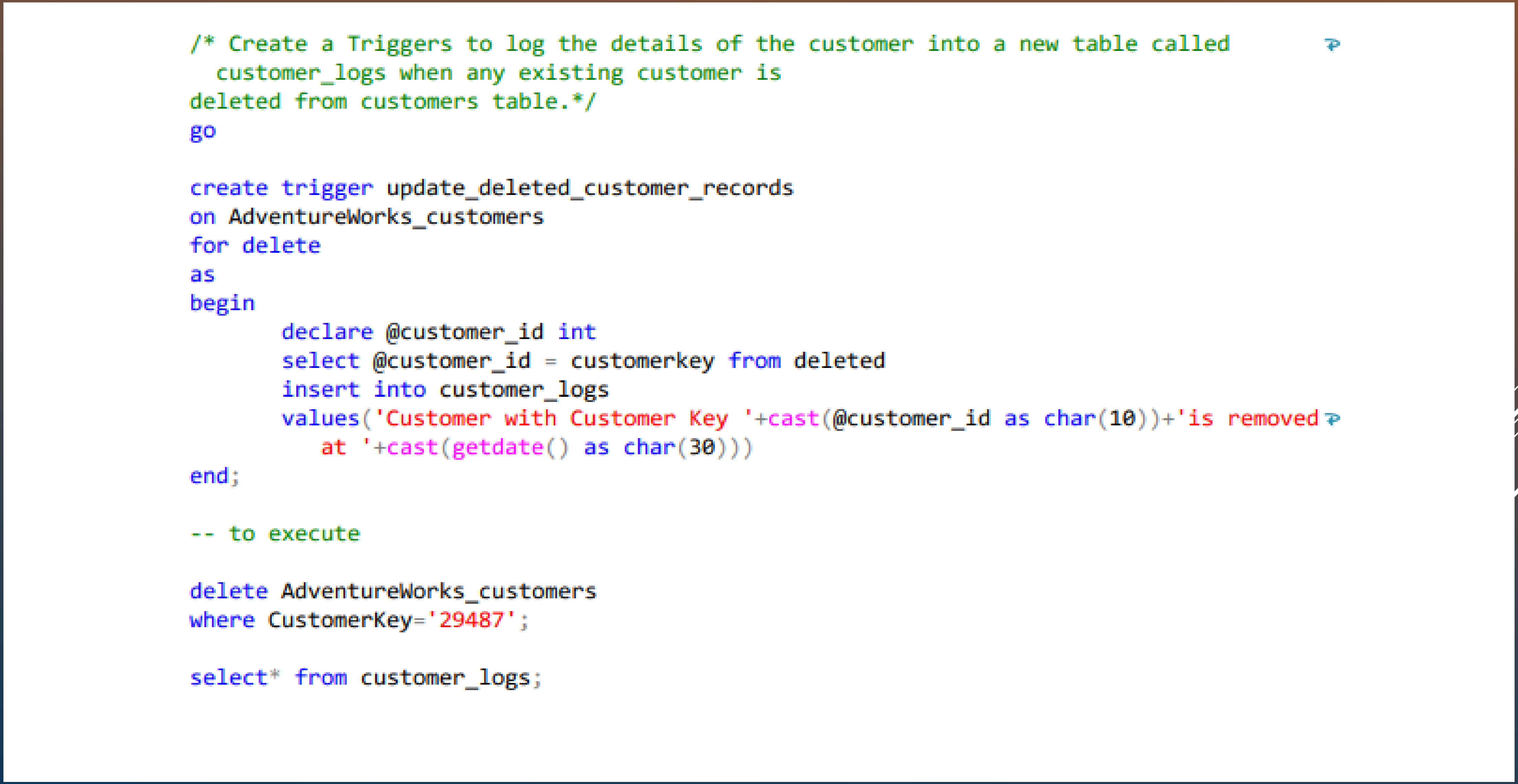
```
/* Create a Trigger to log the details of the customer into a new table called
   customer_logs when any new customer is inserted in
Customers table.*/

-- customer_logs table

create table customer_logs (
id int identity(1,1) ,
action text );

--trigger

Go

Create Trigger update_new_cutomer_records
on AdventureWorks_Customers
for insert
as
begin
      declare @customer_id int
      select @customer_id = customerkey from inserted
      insert into customer_logs
      values('New Customer with Customer Key '+cast(@customer_id as char(10))+' is
        added at '+cast(getdate() as char(30))+'.')
end;

-- To Execute

insert into AdventureWorks_customers
values
  ('29487','MS','ANANYA','SHARMA','04-06-1997','S','F','ananya08@adventure_works.com',
  150000,0,'Bachelors','Management',0);

select* from customer_logs;
```



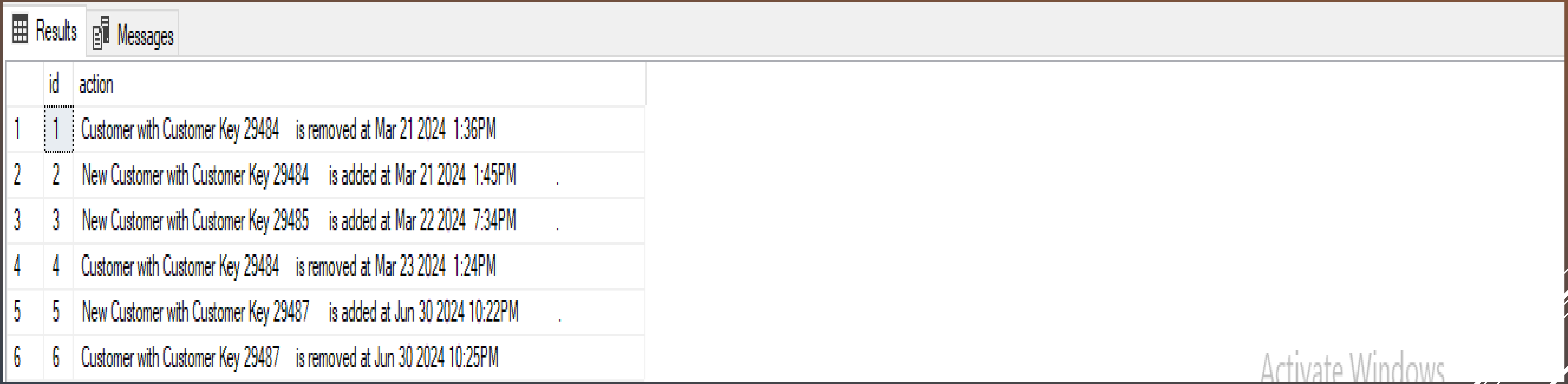
Results | Messages

	id	action
1	1	Customer with Customer Key 29484 is removed at Mar 21 2024 1:36PM
2	2	New Customer with Customer Key 29484 is added at Mar 21 2024 1:45PM .
3	3	New Customer with Customer Key 29485 is added at Mar 22 2024 7:34PM .
4	4	Customer with Customer Key 29484 is removed at Mar 23 2024 1:24PM
5	5	New Customer with Customer Key 29487 is added at Jun 30 2024 10:22PM .

18. Create a trigger to log the details of the customer into a new table called customer_logs when any existing customer is deleted from the Customer table.



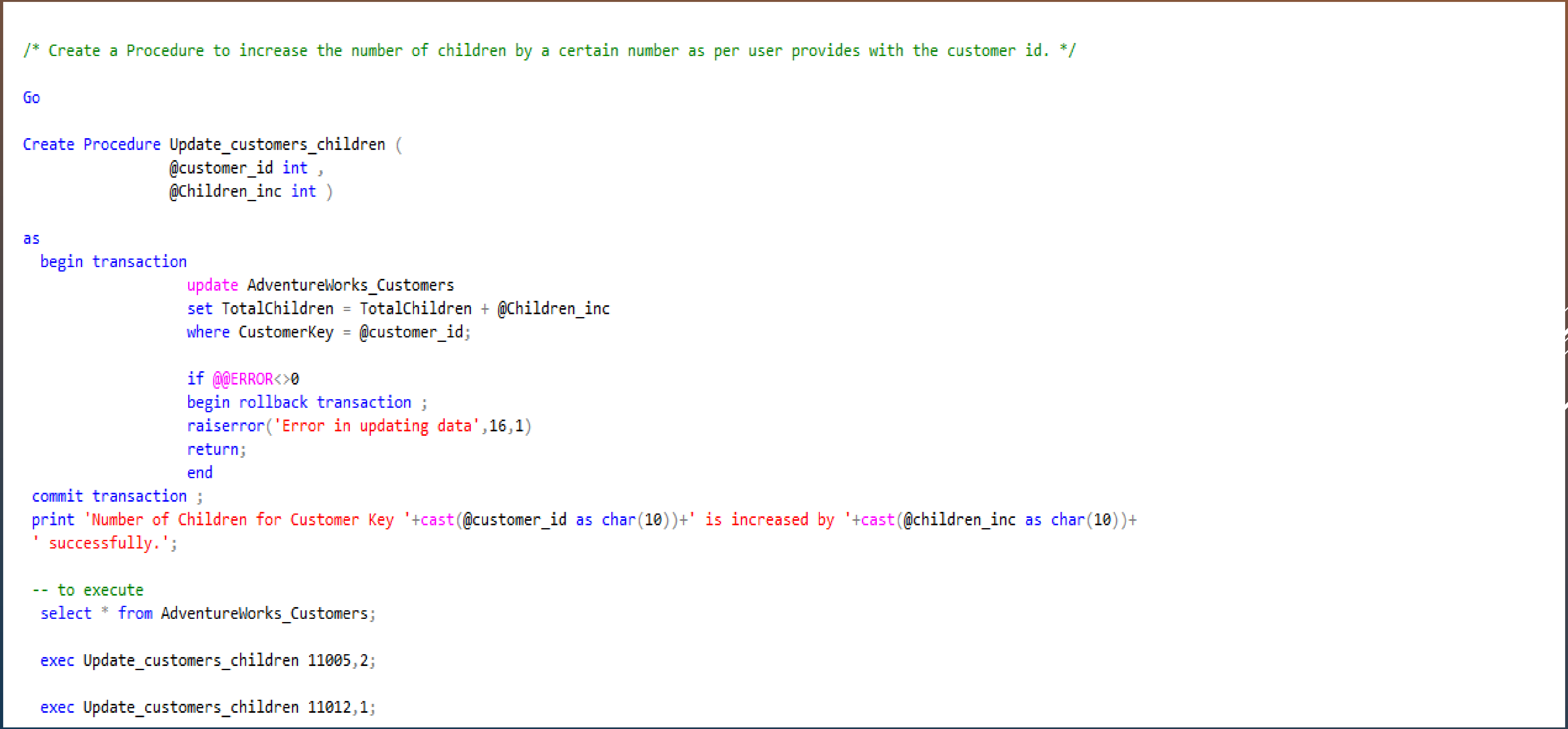
```
/* Create a Triggers to log the details of the customer into a new table called
  customer_logs when any existing customer is
deleted from customers table.*/
go

create trigger update_deleted_customer_records
on AdventureWorks_customers
for delete
as
begin
       declare @customer_id int
       select @customer_id = customerkey from deleted
       insert into customer_logs
       values('Customer with Customer Key '+cast(@customer_id as char(10))+'is removed
          at '+cast(getdate() as char(30)))
end;

-- to execute

delete AdventureWorks_customers
where CustomerKey='29487';

select* from customer_logs;
```



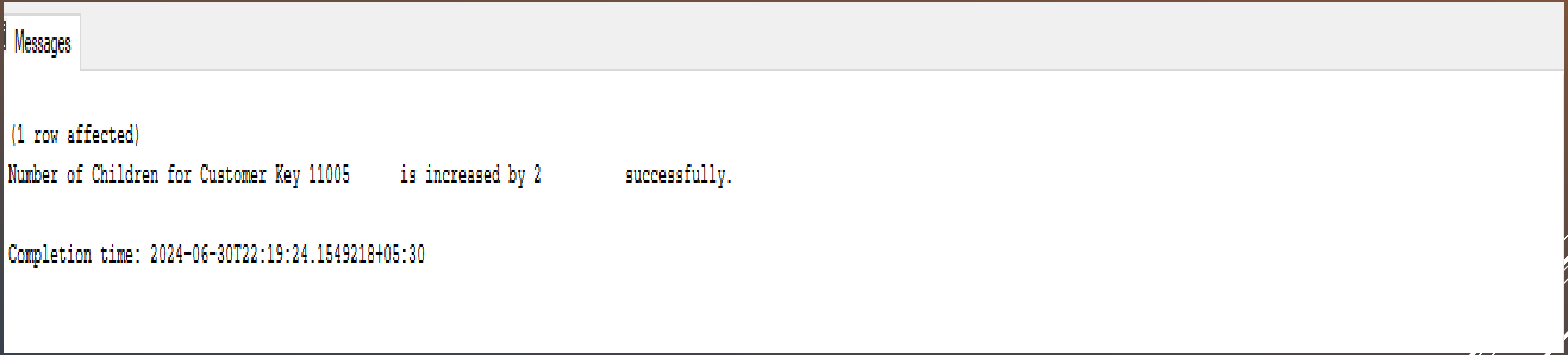
Results | Messages

	id	action
1	1	Customer with Customer Key 29484 is removed at Mar 21 2024 1:36PM
2	2	New Customer with Customer Key 29484 is added at Mar 21 2024 1:45PM .
3	3	New Customer with Customer Key 29485 is added at Mar 22 2024 7:34PM .
4	4	Customer with Customer Key 29484 is removed at Mar 23 2024 1:24PM
5	5	New Customer with Customer Key 29487 is added at Jun 30 2024 10:22PM .
6	6	Customer with Customer Key 29487 is removed at Jun 30 2024 10:25PM

19. Create a Procedure to increase the number of children by a certain number as per user provides with the customer ID.

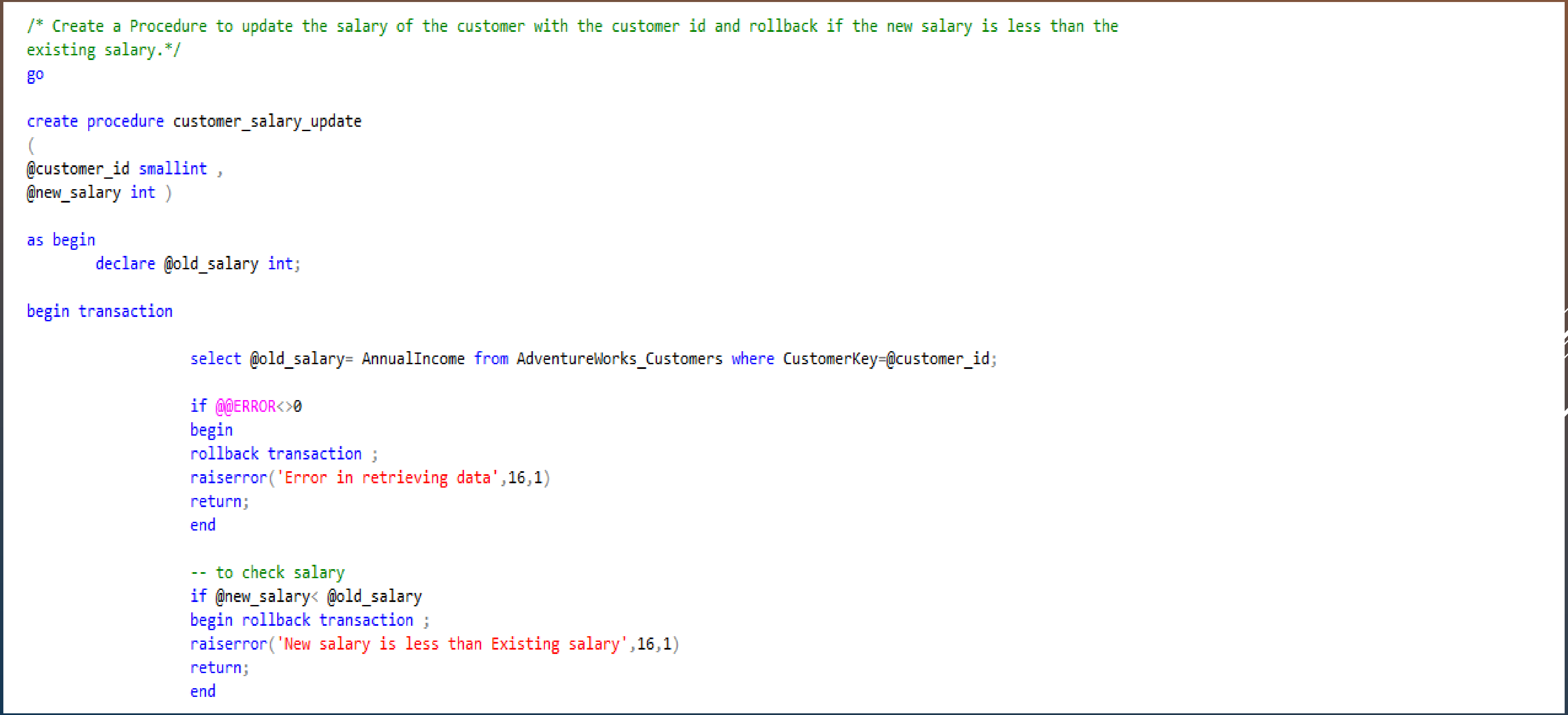


```sql
/* Create a Procedure to increase the number of children by a certain number as per user provides with the customer id. */

Go

Create Procedure Update_customers_children (
                 @customer_id int ,
                 @Children_inc int )

as
  begin transaction
                   update AdventureWorks_Customers
                   set TotalChildren = TotalChildren + @Children_inc
                   where CustomerKey = @customer_id;

                   if @@ERROR<>0
                   begin rollback transaction ;
                   raiserror('Error in updating data',16,1)
                   return;
                   end
 commit transaction ;
 print 'Number of Children for Customer Key '+cast(@customer_id as char(10))+' is increased by '+cast(@children_inc as char(10))+
 ' successfully.';

 -- to execute
  select * from AdventureWorks_Customers;

  exec Update_customers_children 11005,2;

  exec Update_customers_children 11012,1;
```

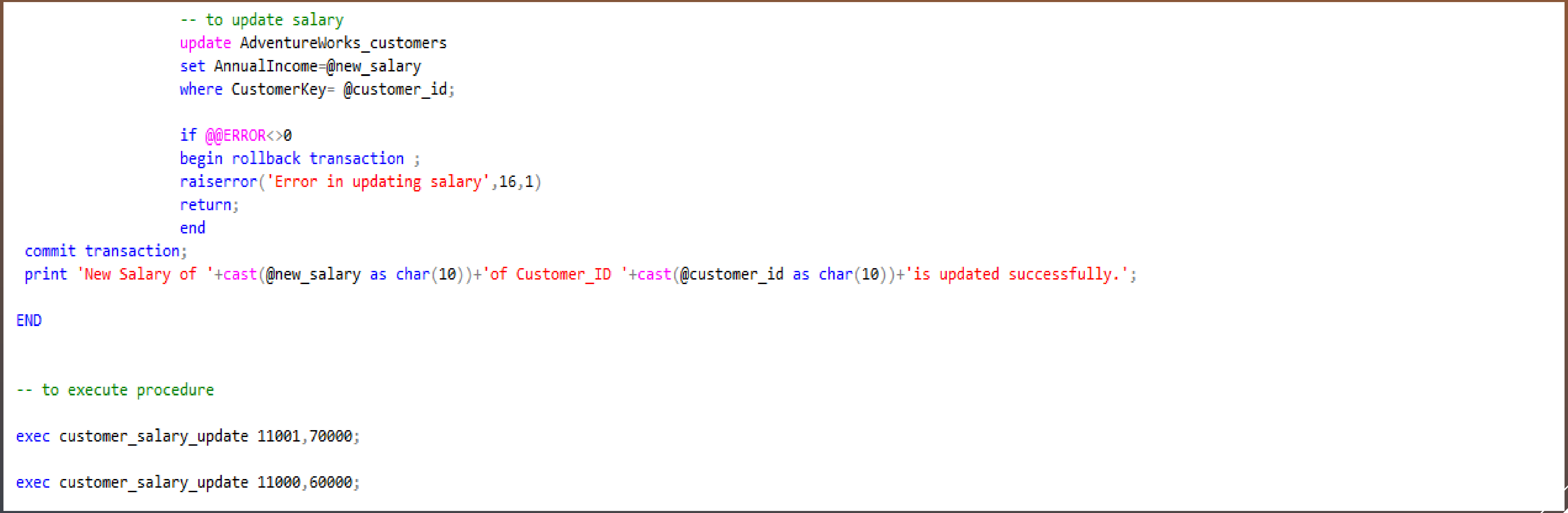


```
(1 row affected)
Number of Children for Customer Key 11005      is increased by 2          successfully.

Completion time: 2024-06-30T22:19:24.1549218+05:30
```

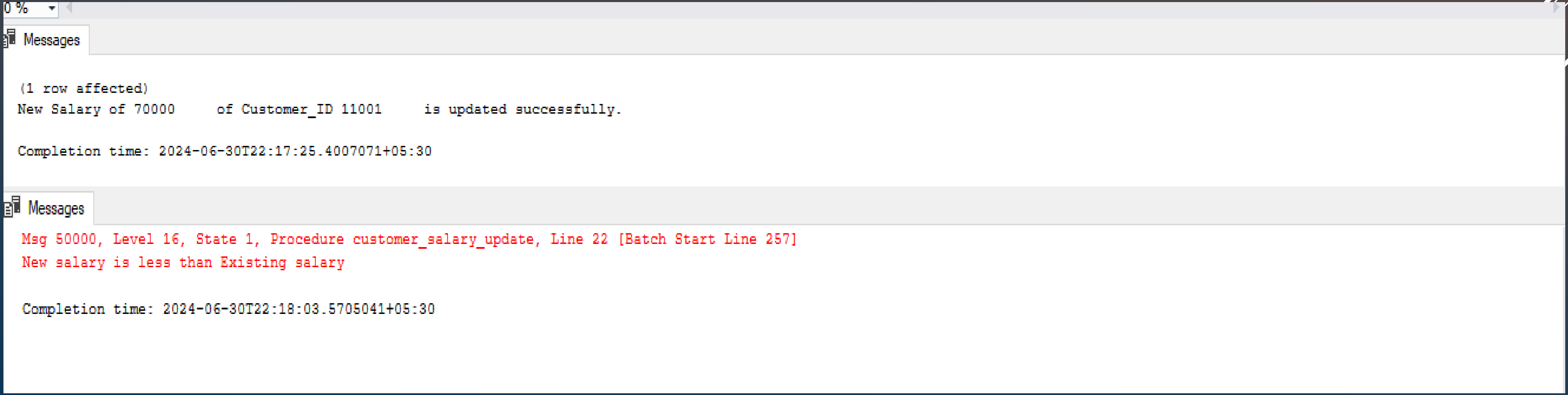
20. Create a Procedure to update the salary of the customer with the customer ID and rollback if the new salary is less than the existing salary.



```
/* Create a Procedure to update the salary of the customer with the customer id and rollback if the new salary is less than the
existing salary.*/
go

create procedure customer_salary_update
(
@customer_id smallint ,
@new_salary int )

as begin
        declare @old_salary int;

begin transaction

                    select @old_salary= AnnualIncome from AdventureWorks_Customers where CustomerKey=@customer_id;

                    if @@ERROR<>0
                    begin
                    rollback transaction ;
                    raiserror('Error in retrieving data',16,1)
                    return;
                    end

                    -- to check salary
                    if @new_salary< @old_salary
                    begin rollback transaction ;
                    raiserror('New salary is less than Existing salary',16,1)
                    return;
                    end
```



```
                -- to update salary
                update AdventureWorks_customers
                set AnnualIncome=@new_salary
                where CustomerKey= @customer_id;

                if @@ERROR<>0
                begin rollback transaction ;
                raiserror('Error in updating salary',16,1)
                return;
                end
 commit transaction;
 print 'New Salary of '+cast(@new_salary as char(10))+'of Customer_ID '+cast(@customer_id as char(10))+'is updated successfully.';

END


-- to execute procedure

exec customer_salary_update 11001,70000;

exec customer_salary_update 11000,60000;
```



Messages

```
(1 row affected)
New Salary of 70000     of Customer_ID 11001     is updated successfully.

Completion time: 2024-06-30T22:17:25.4007071+05:30
```

Messages

```
Msg 50000, Level 16, State 1, Procedure customer_salary_update, Line 22 [Batch Start Line 257]
New salary is less than Existing salary

Completion time: 2024-06-30T22:18:03.5705041+05:30
```

THANK-YOU