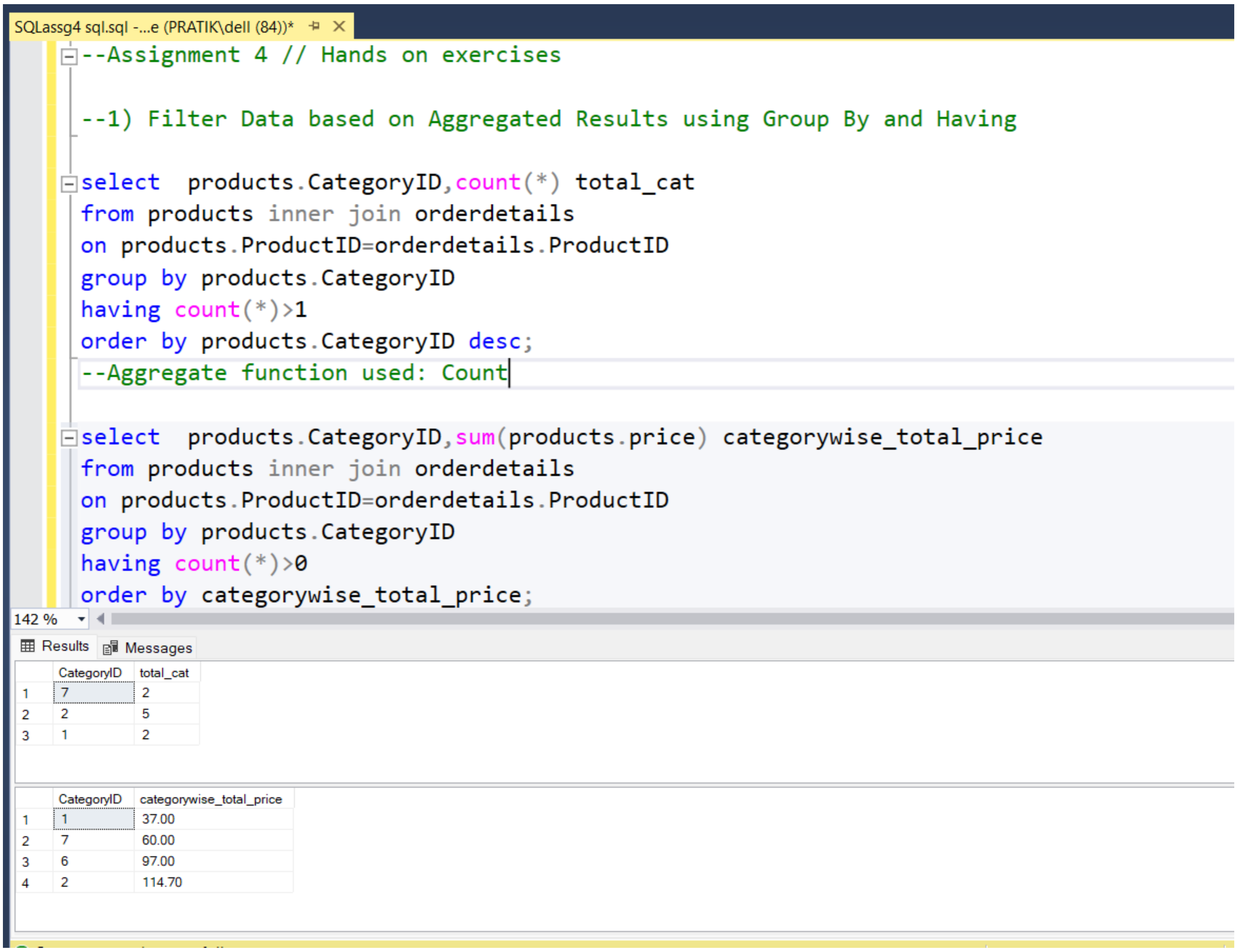
Assignment No: 4 (Hands on)

Name: Pratik Wani

- SQL Queries
  - Here I Used the Previously created database named as ‘Practice’ and table named as ‘Products’ and ‘OrderDetails’ to perform hands on exercises.

- <u>Filter Data based on Aggregated Results using Group By and Having:</u>



SQLassg4 sql.sql -...e (PRATIK\dell (84))*

```
--Assignment 4 // Hands on exercises

--1) Filter Data based on Aggregated Results using Group By and Having

select  products.CategoryID,count(*) total_cat
from products inner join orderdetails
on products.ProductID=orderdetails.ProductID
group by products.CategoryID
having count(*)>1
order by products.CategoryID desc;
--Aggregate function used: Count

select  products.CategoryID,sum(products.price) categorywise_total_price
from products inner join orderdetails
on products.ProductID=orderdetails.ProductID
group by products.CategoryID
having count(*)>0
order by categorywise_total_price;
```

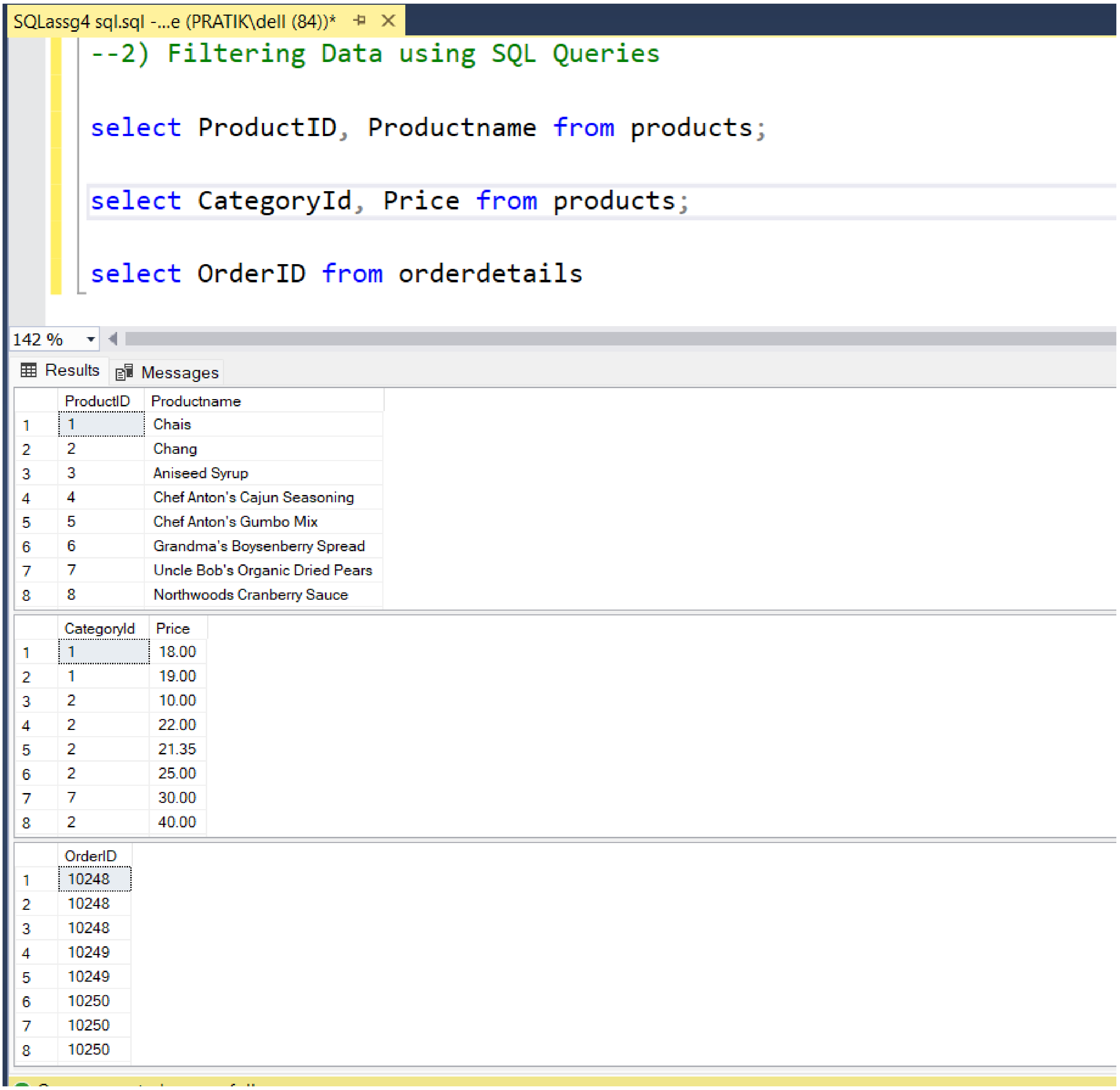
142 %

Results | Messages

	CategoryID	total_cat
1	7	2
2	2	5
3	1	2

	CategoryID	categorywise_total_price
1	1	37.00
2	7	60.00
3	6	97.00
4	2	114.70

- Filtering Data using SQL Queries:
    - We can filter the data using different column name


SQLassg4 sql.sql -...e (PRATIK\dell (84))*

```
--2) Filtering Data using SQL Queries

select ProductID, Productname from products;

select CategoryId, Price from products;

select OrderID from orderdetails
```

142 %

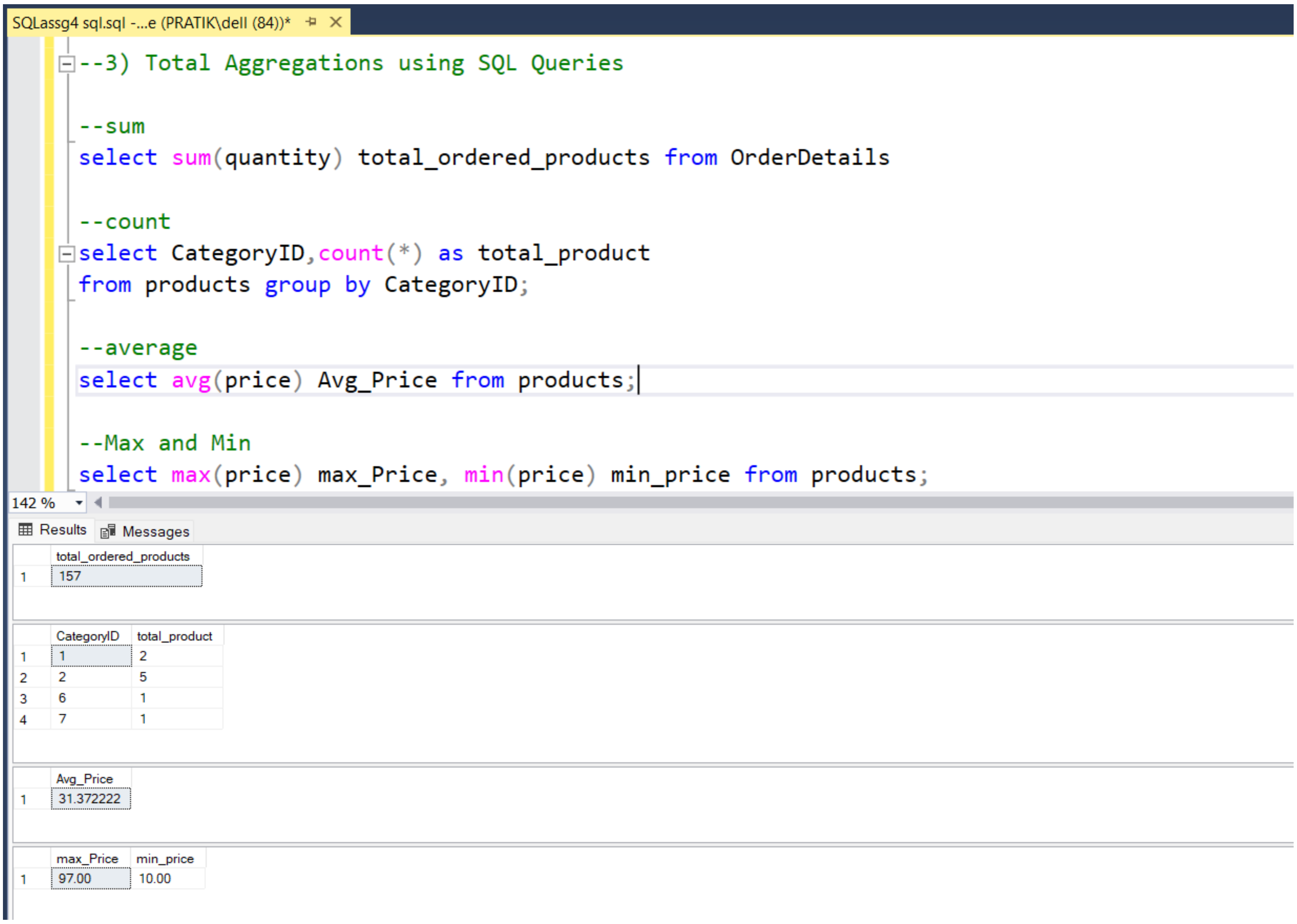
Results | Messages

	ProductID	Productname
1	1	Chais
2	2	Chang
3	3	Aniseed Syrup
4	4	Chef Anton's Cajun Seasoning
5	5	Chef Anton's Gumbo Mix
6	6	Grandma's Boysenberry Spread
7	7	Uncle Bob's Organic Dried Pears
8	8	Northwoods Cranberry Sauce

	CategoryId	Price
1	1	18.00
2	1	19.00
3	2	10.00
4	2	22.00
5	2	21.35
6	2	25.00
7	7	30.00
8	2	40.00

	OrderID
1	10248
2	10248
3	10248
4	10249
5	10249
6	10250
7	10250
8	10250

- <u>Total Aggregations using SQL Queries:</u>
    - Sum: Returns the Addition of the values
    - Count: Returns the Total count used with the group by statement
    - Avg: Returns the Average value of the values
    - Max: Returns the Max values
    - Min: Return the Min values



```sql
--3) Total Aggregations using SQL Queries

--sum
select sum(quantity) total_ordered_products from OrderDetails

--count
select CategoryID,count(*) as total_product
from products group by CategoryID;

--average
select avg(price) Avg_Price from products;

--Max and Min
select max(price) max_Price, min(price) min_price from products;
```

	total_ordered_products
1	157

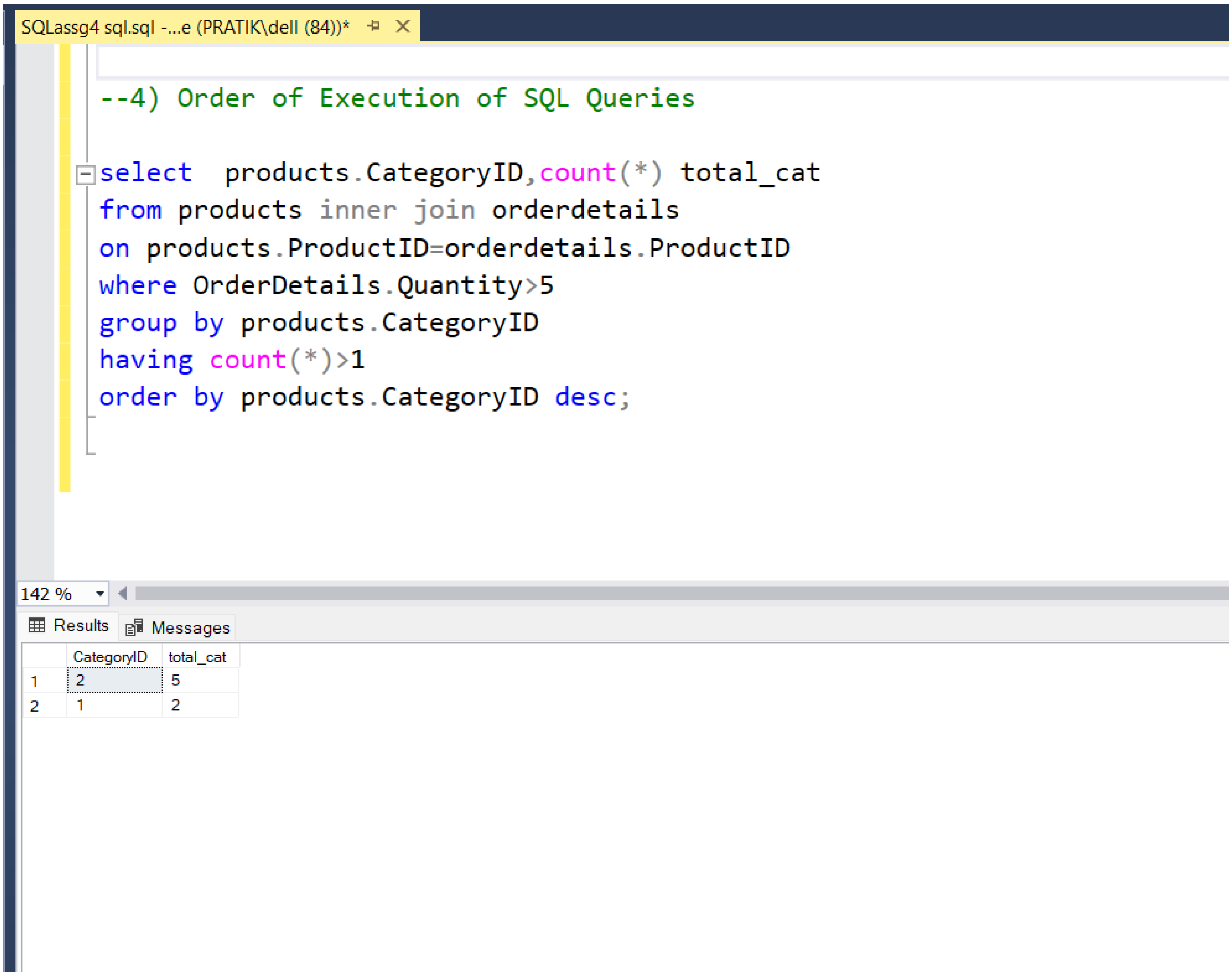
	CategoryID	total_product
1	1	2
2	2	5
3	6	1
4	7	1

	Avg_Price
1	31.372222

	max_Price	min_price
1	97.00	10.00

- <u>Order of Execution of SQL Queries:</u>

From → Where → Group By → Having → Select → Order By

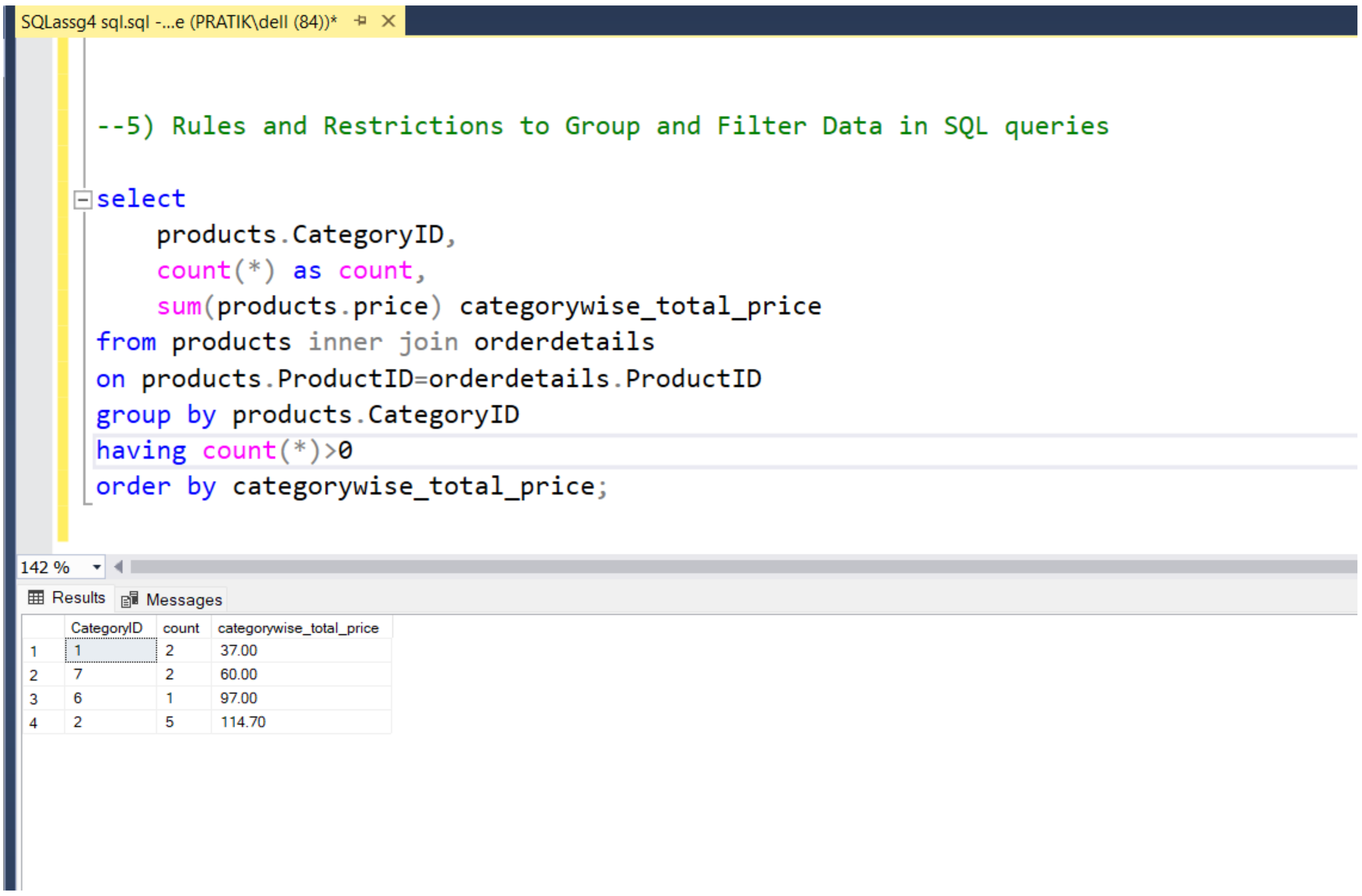


```sql
--4) Order of Execution of SQL Queries

select  products.CategoryID,count(*) total_cat
from products inner join orderdetails
on products.ProductID=orderdetails.ProductID
where OrderDetails.Quantity>5
group by products.CategoryID
having count(*)>1
order by products.CategoryID desc;
```

	CategoryID	total_cat
1	2	5
2	1	2

- Rules and Restrictions to Group and Filter Data in SQL queries:
  - Group By: Used of Group the data and enables us to use aggregate functions on groups of data returned from a query.
  - Filter: used to refine a query even more by running your aggregations against a limited set of the values in a column



```sql
--5) Rules and Restrictions to Group and Filter Data in SQL queries

select
    products.CategoryID,
    count(*) as count,
    sum(products.price) categorywise_total_price
from products inner join orderdetails
on products.ProductID=orderdetails.ProductID
group by products.CategoryID
having count(*)>0
order by categorywise_total_price;
```

	CategoryID	count	categorywise_total_price
1	1	2	37.00
2	7	2	60.00
3	6	1	97.00
4	2	5	114.70