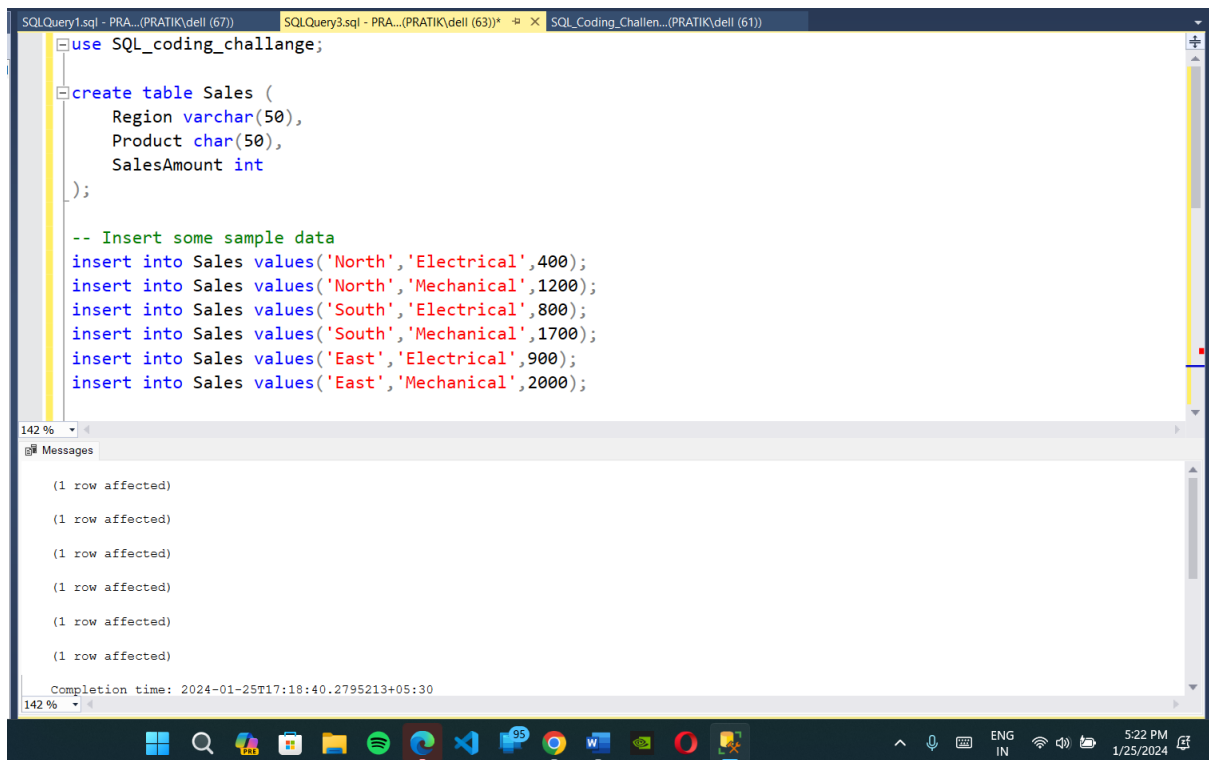


SQL Coding Challenge 1

- Database Information:
 - Created Database name as SQL_coding_challenge
 - Created one tables: Sales



The screenshot displays a SQL IDE window with three tabs: 'SQLQuery1.sql - PRA...(PRATIK\dell (67))', 'SQLQuery3.sql - PRA...(PRATIK\dell (63))', and 'SQL_Coding_Challen...(PRATIK\dell (61))'. The active tab shows the following SQL code:

```
use SQL_coding_challenge;

create table Sales (
    Region varchar(50),
    Product char(50),
    SalesAmount int
);

-- Insert some sample data
insert into Sales values('North', 'Electrical', 400);
insert into Sales values('North', 'Mechanical', 1200);
insert into Sales values('South', 'Electrical', 800);
insert into Sales values('South', 'Mechanical', 1700);
insert into Sales values('East', 'Electrical', 900);
insert into Sales values('East', 'Mechanical', 2000);
```

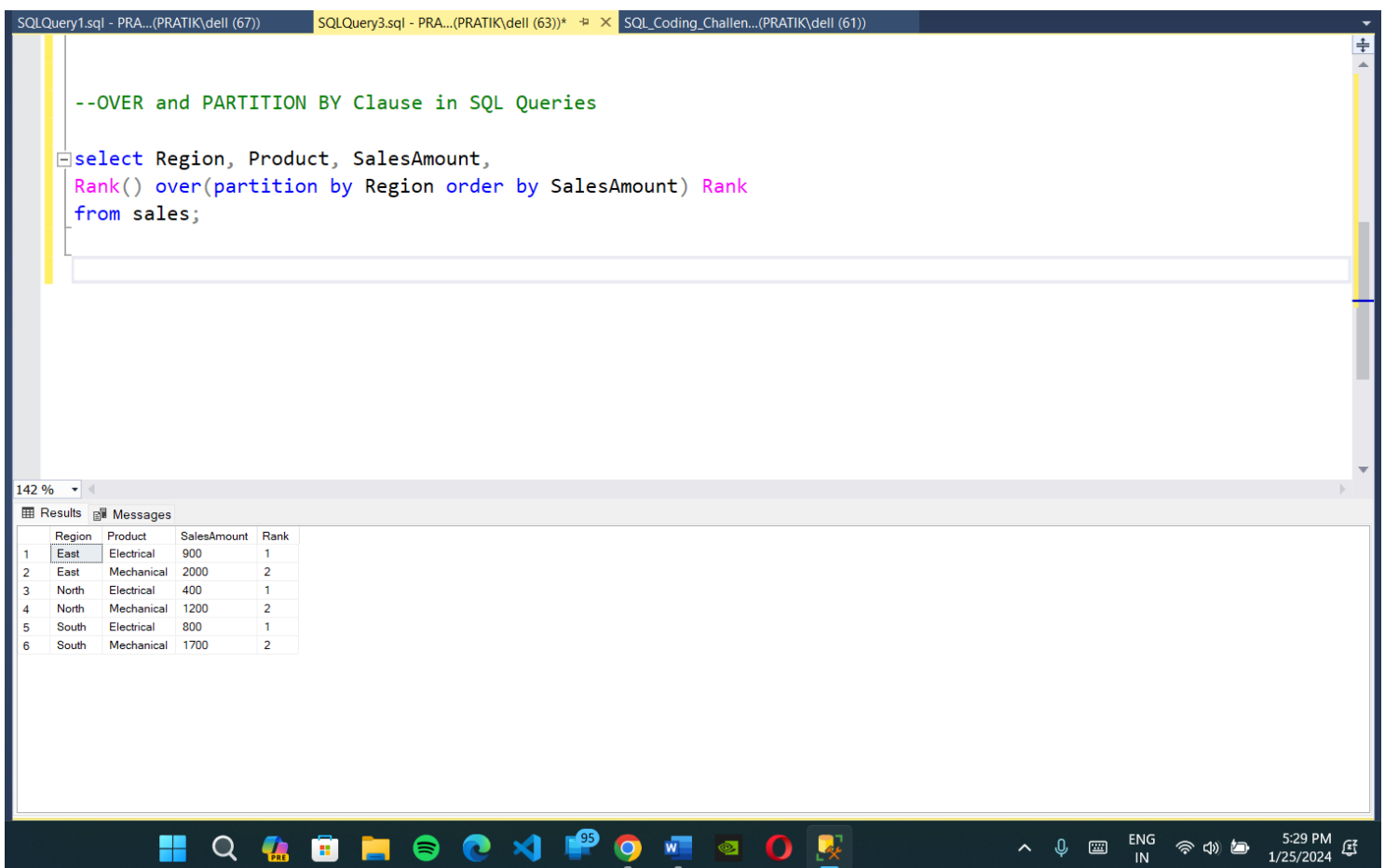
Below the code editor, the 'Messages' pane shows the execution results:

```
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
```

The completion time is noted as: 2024-01-25T17:18:40.2795213+05:30.

■ Over and Partition by:

- The OVER clause use to specifies the rows over which a window function.
- It can include both the PARTITION BY clause and the ORDER BY clause.
- PARTITION BY clause is used to divide the result set into groups based on some column
- As here we are dividing the result according to there region.
- Here also use RANK() function to give each row specific rank according to region and sales.
- Here It is partition by Region and order by sales amount
- So Rank 1 is for East region and Electrical product.



The screenshot shows a SQL IDE with three tabs: 'SQLQuery1.sql - PRA...(PRATIK\deli (67))', 'SQLQuery3.sql - PRA...(PRATIK\deli (63))*', and 'SQL_Coding_Challen...(PRATIK\deli (61))'. The active tab displays the following SQL query:

```
--OVER and PARTITION BY Clause in SQL Queries  
select Region, Product, SalesAmount,  
       Rank() over(partition by Region order by SalesAmount) Rank  
from sales;
```

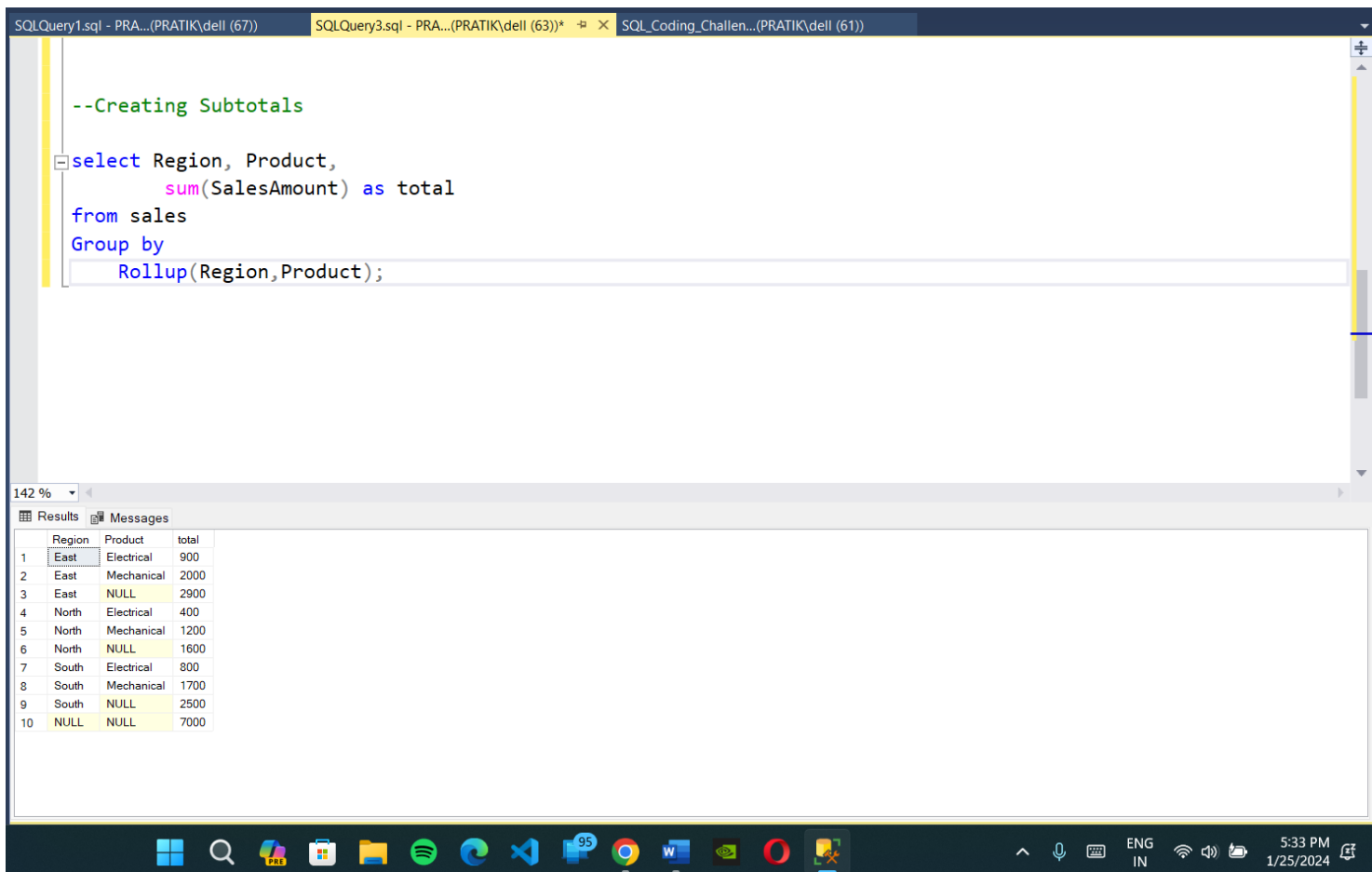
Below the query editor, the 'Results' pane shows a table with 6 rows and 4 columns: Region, Product, SalesAmount, and Rank. The data is as follows:

	Region	Product	SalesAmount	Rank
1	East	Electrical	900	1
2	East	Mechanical	2000	2
3	North	Electrical	400	1
4	North	Mechanical	1200	2
5	South	Electrical	800	1
6	South	Mechanical	1700	2

The Windows taskbar at the bottom shows the time as 5:29 PM on 1/25/2024, with system language set to ENG IN.

■ Creating Subtotals:

- To create the subtotals we need Group by Rollup.
- Subtotals is sum for Specific sections
- Here in these query Subtotal is created for Region wise and Product wise
- For East Region subtotal = 2900
- For North Region subtotal = 1600
- For South Region subtotal = 2500
- At the end we will also got the grand total as 7000.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query titled "--Creating Subtotals" in a green font. The query is as follows:

```
--Creating Subtotals
select Region, Product,
       sum(SalesAmount) as total
from sales
Group by
Rollup (Region, Product);
```

The bottom pane shows the results of the query in a table format. The table has three columns: Region, Product, and total. The results are as follows:

Region	Product	total
East	Electrical	900
East	Mechanical	2000
East	NULL	2900
North	Electrical	400
North	Mechanical	1200
North	NULL	1600
South	Electrical	800
South	Mechanical	1700
South	NULL	2500
NULL	NULL	7000

■ Total Aggregations using SQL Queries:

- When we use aggregations on the over() and order by clause we called it as total aggregation.
- We can also use aggregations by partitioning the result set
- If we didn't use partition by then aggregation is applied on whole set.
- Here In 1st example sum() is applied on whole set and we got the aggregation for whole result set which comes 7000
- In 2nd example sum() is applied by making partition according to the region so got sum for specific regions
- For East Region subtotal = 2900
- For North Region subtotal = 1600
- For South Region subtotal = 2500

SQLQuery1.sql - PRA...(PRATIK\ dell (67))* SQLQuery3.sql - PRA...(PRATIK\ dell (63))* SQL_Coding_Challen...(PRATIK\ dell (61))

```
--Total Aggregations using SQL Queries.  
--1)  
select  
    Region,  
    Product,  
    SalesAmount,  
    sum(salesamount) OVER () grand_sum  
from  
    Sales;
```

142 %

Results Messages

	Region	Product	SalesAmount	grand_sum
1	North	Electrical	400	7000
2	North	Mechanical	1200	7000
3	South	Electrical	800	7000
4	South	Mechanical	1700	7000
5	East	Electrical	900	7000
6	East	Mechanical	2000	7000

Opera Browser

Windows taskbar: 5:51 PM 1/25/2024

SQLQuery1.sql - PRA...(PRATIK\ dell (67))* SQLQuery3.sql - PRA...(PRATIK\ dell (63))* SQL_Coding_Challen...(PRATIK\ dell (61))

```
--2)  
select  
    Region,  
    Product,  
    SalesAmount,  
    sum(salesamount) OVER (partition by region) as sum_regionwise  
from  
    Sales;
```

142 %

Results Messages

	Region	Product	SalesAmount	sum_regionwise
1	East	Electrical	900	2900
2	East	Mechanical	2000	2900
3	North	Electrical	400	1600
4	North	Mechanical	1200	1600
5	South	Electrical	800	2500
6	South	Mechanical	1700	2500

Windows taskbar: 5:52 PM 1/25/2024