**A close-up of a logo

Description automatically generated**

**Course:** Relational Database

**Student Name:** Parimal Sawant

**Student ID:** A00325752

**Date:**  11 December 2024

**Analytic SQL Project**

**Introduction**

In today’s education environment, scholarship supports the aspiring students to help pursue their higher education. Scholarship provides financial assistant to students, allowing them to pursue their studies at higher colleges without the burden of financial part. The scholarships are majorly funded by businesses or organisations, which serves as sponsors to the scholarship. Sponsors are essential to promote education and to find out talent.

Applicants, typically apply for this scholarship during their secondary school to pursue higher education. The selection process may take some measures that are recommended by the sponsor. Most of the scholarships are awarded to deserving and talented students, there are instances where scholarships are unawarded as well. By implementation of database the stake holders will be able to track the scholarships efficiently and manage scholarship awards, to ensure the process remains streamlined.

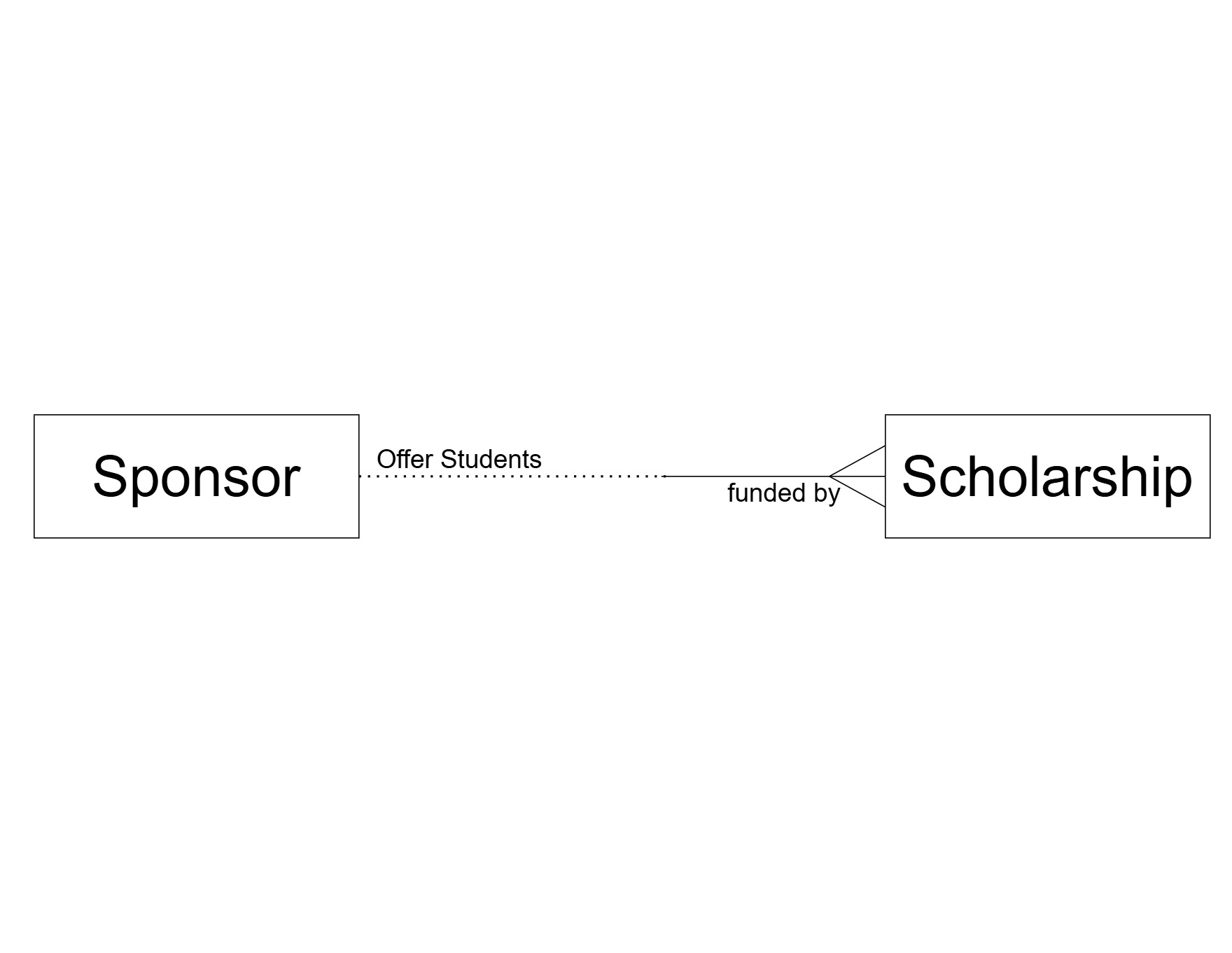
**Entities and Relationship**

The database consists of 2 key entities: Sponsor and Scholarship which are structured in a parent child relationship. The sponsor serves as a parent table here while the scholarship serves as a child table. The 2 parent and child tables are merged by a foreign key Sponsor\_Id which is primary key in the Sponsor table. This link between the two tables ensures that each scholarship is associated with one sponsor.

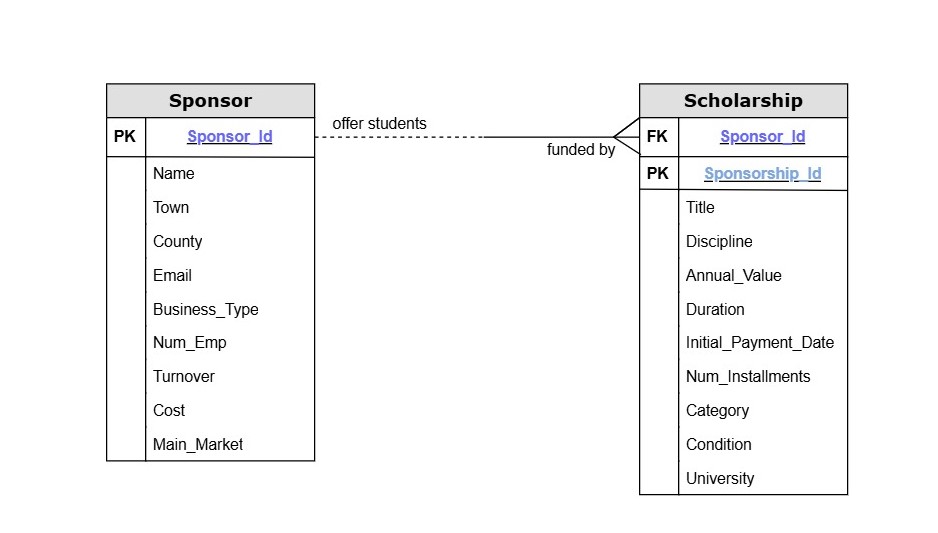
**Sponsor:** **Sponsor\_Id (PK)**, Name, Town, County, Email, Business\_Type, Num\_Emp, Turnover, Costs, Main\_Market

**Scholarship: Sponsor\_Id (FK)**, **Scholarship\_Id (PK)**, Title, Discipline, Annual\_Value, Duration, Initial\_Payment\_Date, Num\_Installments, Category, Condition, University

**ER Diagram**

****

**Fig 1.1 ER Diagram without attribute**

****

**Fig 1.2 ER Diagram with attribute**

**Create and Insert Query**

**Create Sponsor Table**

Create Table Sponsor(

Sponsor\_Id Number(2),

Name Varchar2(30) Constraint Sponsor\_Name\_Nn Not Null,

Town Varchar2(15),

County Varchar2(11),

Email Varchar2(35) Constraint Sponsor\_Email\_Uq Unique,

Business\_Type Varchar2(13),

Num\_Emp Number(5),

Turnover Number(10),

Costs Number(9),

Main\_Market Varchar2(7),

Constraint Sponsor\_Sponsor\_Id\_Pk Primary Key (Sponsor\_Id),

Constraint Sponsor\_Sponsor\_Id\_Ck Check (Sponsor\_Id > 0),

Constraint Sponsor\_Town\_Initcap\_Ck Check (Town = Initcap(Town)),

Constraint Sponsor\_County\_Initcap\_Ck Check (County = Initcap(County)),

Constraint Sponsor\_Business\_Initcap\_Ck Check (Business\_Type = Initcap(Business\_Type)),

Constraint Sponsor\_Business\_Type\_Ck Check (Business\_Type In ('Retail', 'Manufacturing', 'Service', 'Leisure')),

Constraint Sponsor\_Email\_Ck Check (Email Is Null Or Email LIKE '\_%@\_%.\_%'));

**Insert Data Sample for Sponsor Table**

Insert Into Sponsor Values(1, 'Eircom', 'Courtmacsherry', 'Dublin 2', '1stinitialsurname@eircom.ie', 'Service', 8000, 168000000, 23520000, 'Ireland');

**Create Scholarship Table**

Create Table Scholarship(

Sponsor\_Id Number(2),

Scholarship\_Id Number(3),

Title Varchar2(50) Constraint Scholarship\_Scholarship\_Id\_Nn Not Null,

Discipline Varchar2(11),

Annual\_Value Number(5),

Duration Number(1),

Initial\_Payment\_Date Date,

Num\_Instalments Number(2),

Category Varchar2(13),

Condition Varchar2(32),

University Varchar2(35),

Constraint Scholarship\_Sponsor\_Id\_Fk Foreign Key (Sponsor\_Id) References Sponsor (Sponsor\_Id),

Constraint Scholarship\_Scholarship\_Id\_Pk Primary Key (Scholarship\_Id),

Constraint Scholarship\_Scholarship\_Id\_Ck Check (Scholarship\_Id > 0),

Constraint Scholarship\_Discipline\_Ck Check (Discipline In ('Medical', 'Engineering', 'Psychology', 'Science')),

Constraint Scholarship\_Number\_Ck Check (Duration > 0 And Annual\_Value > 0 And Num\_Instalments > 0),

Constraint Scholarship\_Condition\_Ck Check (Condition In ('Must get average above 70%', 'Must pass exams', 'Reviewed at the end of each year')));

**Insert Data Sample Scholarship Table**

Insert Into Scholarship Values(21, 1, 'Diageo Ireland Student Scholarship', 'Medical', 16500, 3, '19-Apr-2024', 41, 'Undergraduate', 'Must get average above 70%', 'University of Westminster');

**SQL Analytic Query**

**Over Clause**

**Example 1.** List the Name, Business Type and Cost from Sponsor for each individual employee plus the overall total cost for all the Sponsor, only show results where Business Type is Retail and Service. Sort the results by Business Type and Name.

Cl scr

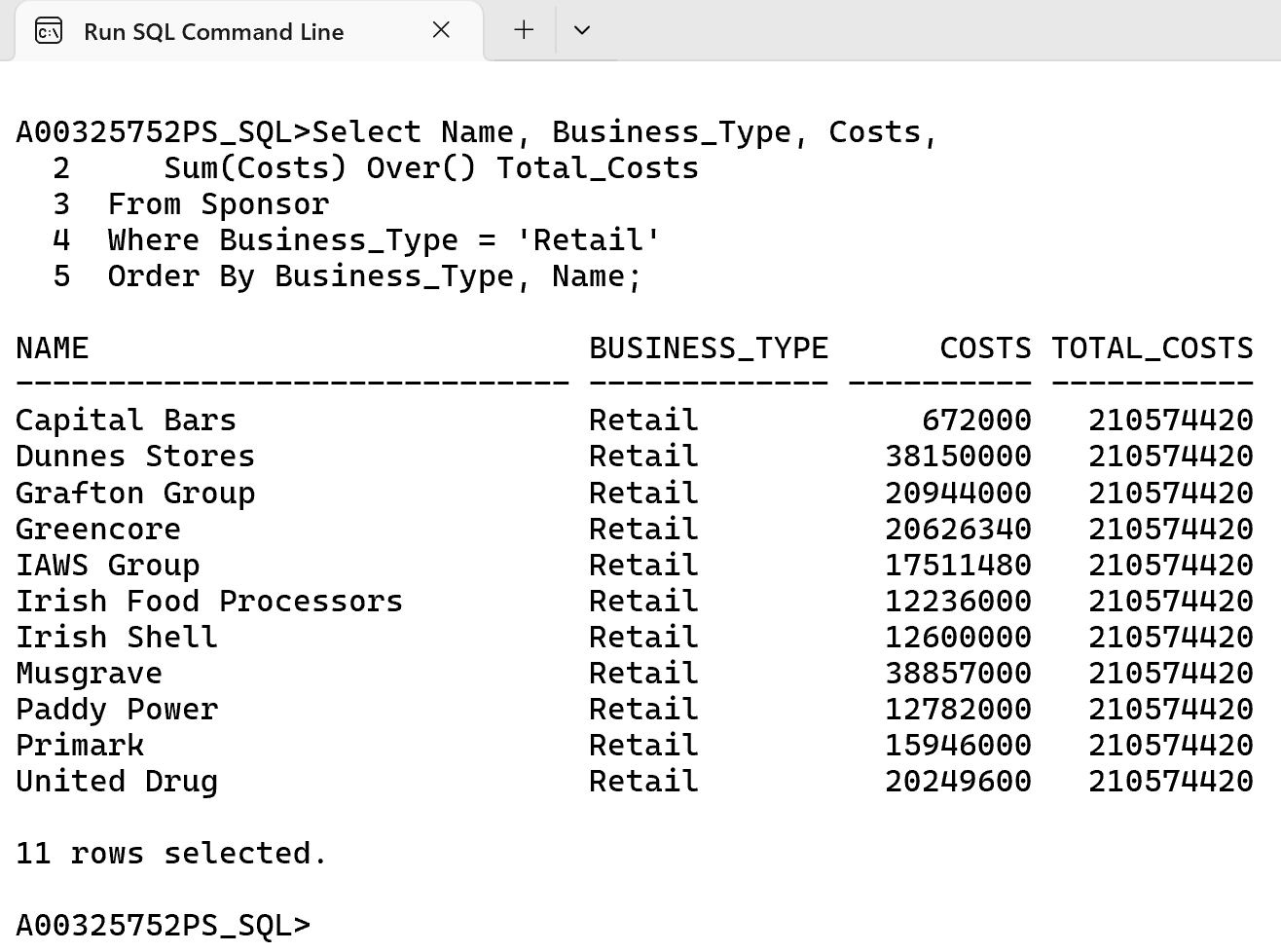
Select Name, Business\_Type, Costs,

Sum(Costs) Over() **Total\_Costs**

From Sponsor

Where Business\_Type = 'Retail'

Order By Business\_Type, Name;



This SQL query consist of Sum aggregate feature, but it also features OVER clause. Therefore, we know that it is an Analytic function instead of an aggregate function.

The OVER clause is the basic one as it will add up the total costs and display in the last column which is not feasible with aggregate functions.

**Over Clause**

**Example 2.** List the Name, Business\_Type, Costs for each sponsor plus the largest costs among the Sponsors. In addition, calculate and display the amount each individual cost falls short of the largest costs. Filter out results where Business Type in Service or Leisure. Sort the results by Name and then by Business Type.

Cl scr

Select Name, Business\_Type, Costs,

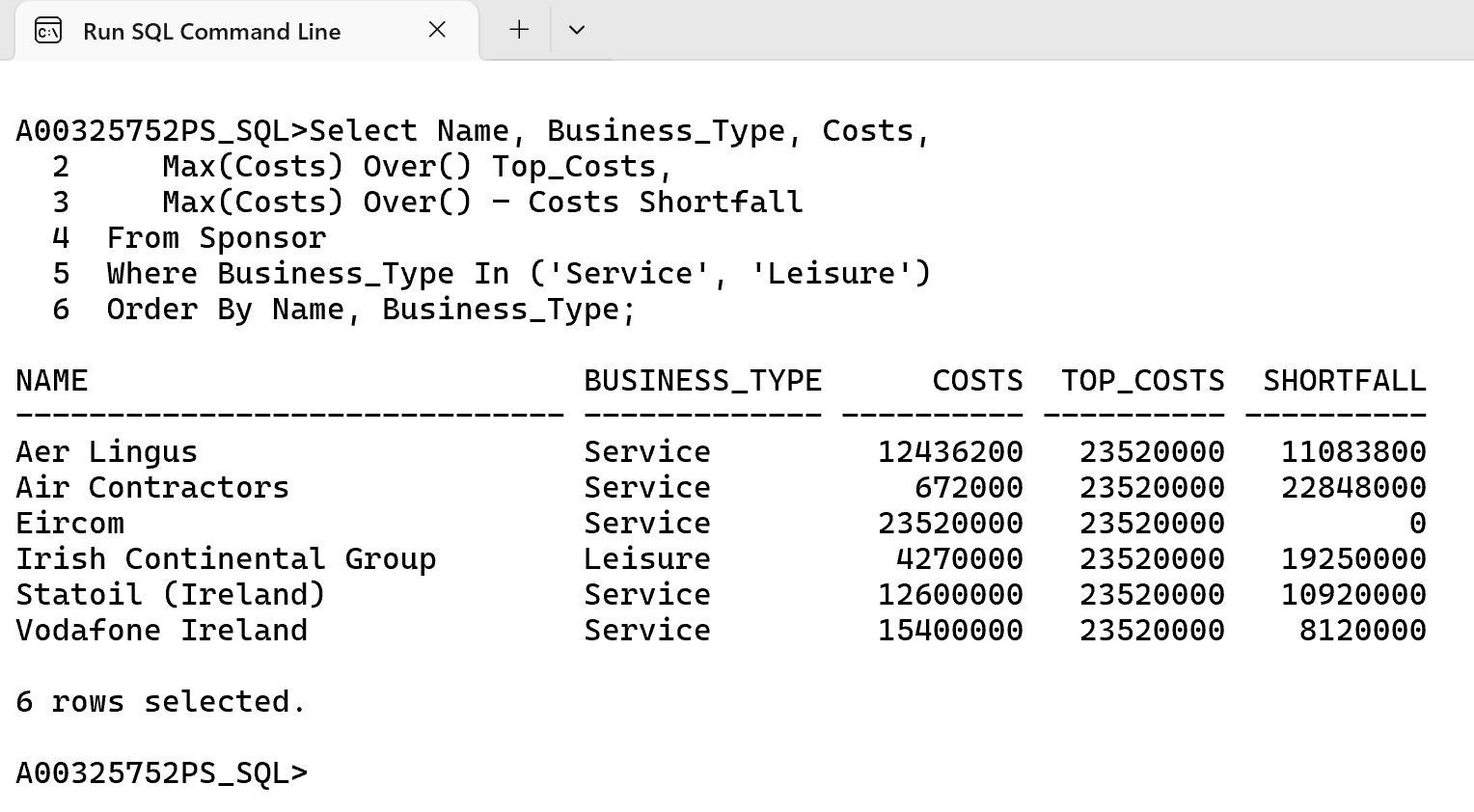
Max(Costs) Over() **Top\_Costs**,

Max(Costs) Over() – **Costs Shortfall**

From Sponsor

Where Business\_Type In ('Service', 'Leisure')

Order By Name, Business\_Type;



In this query we are using Over clause we have used in previous example but on the Shortfall column we are subtracting the actual cost with maximum cost.

**Over Clause**

**Example 3.** List the Name, Business\_Type, Costs for each sponsor plus the largest and smallest costs among the Sponsors. In addition, calculate and display the amount each individual cost falls short of the largest costs, and the amount each sponsor exceeds the smallest costs. Filter out results where Business Type in Service or Leisure. Sort the results by Name and then by Business Type.

Cl scr

Select Name, Business\_Type, Costs,

Max(Costs) Over() **Top\_Costs**,

Max(Costs) Over() – **Costs Shortfall**,

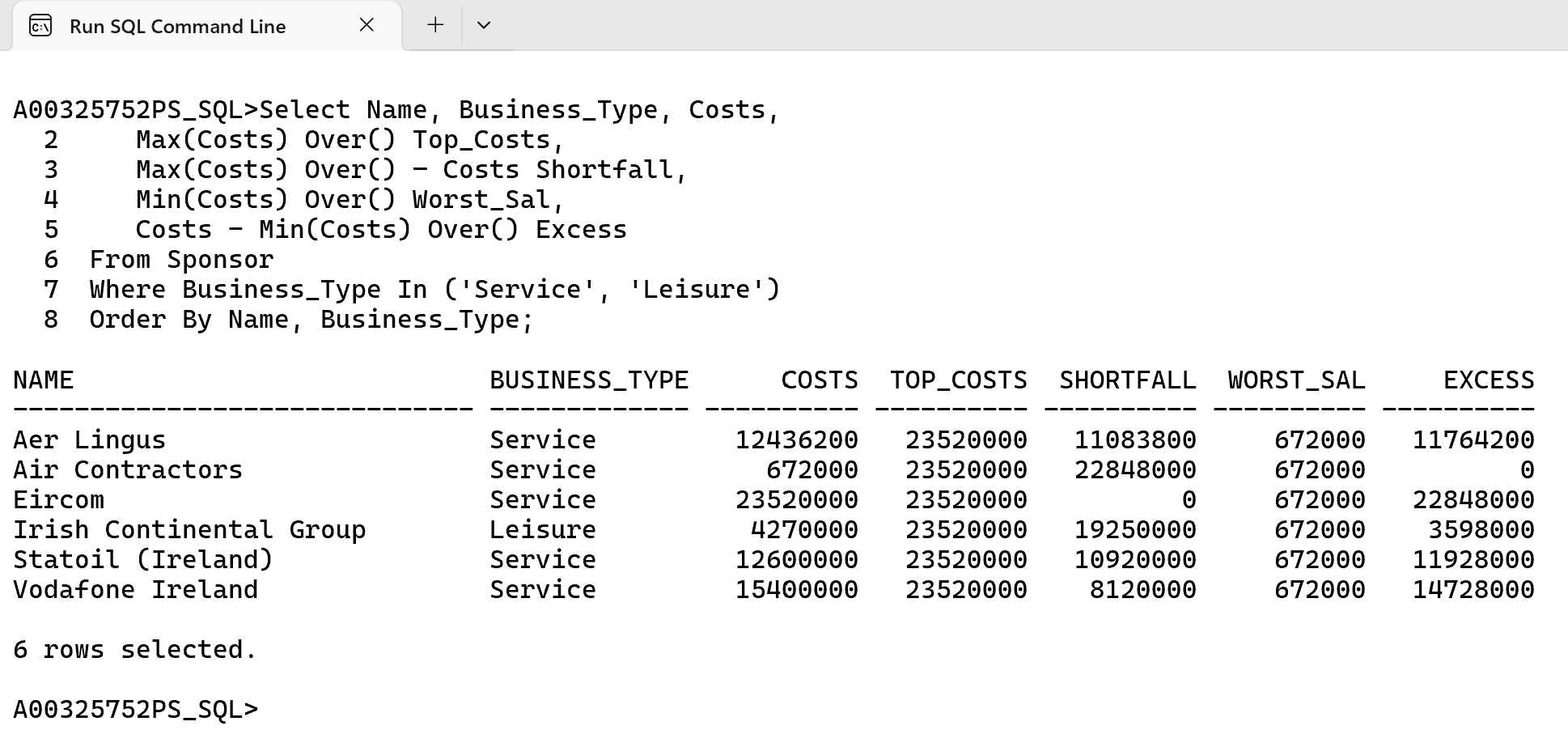
Min(Costs) Over() **Worst\_Sal**,

Costs - Min(Costs) Over() **Excess**

From Sponsor

Where Business\_Type In ('Service', 'Leisure')

Order By Name, Business\_Type;



This query is similar to the previous one but we are using Over clause here to show Minimum cost and the excess by subtracting cost with Minimum cost.

**Over Clause with all aggregate functions**

**Example 4.** List the Name, Business\_Type and Costs for each individual Sponsor plus the lowest, highest, average and overall total costs for every Business Type except Manufacturing. Sort the results by Name and then by Business Type.

Cl scr

Select Name, Business\_Type, Costs,

Max(Costs) Over() **Top\_Costs**,

Min(Costs) Over() **Bottom\_Costs**,

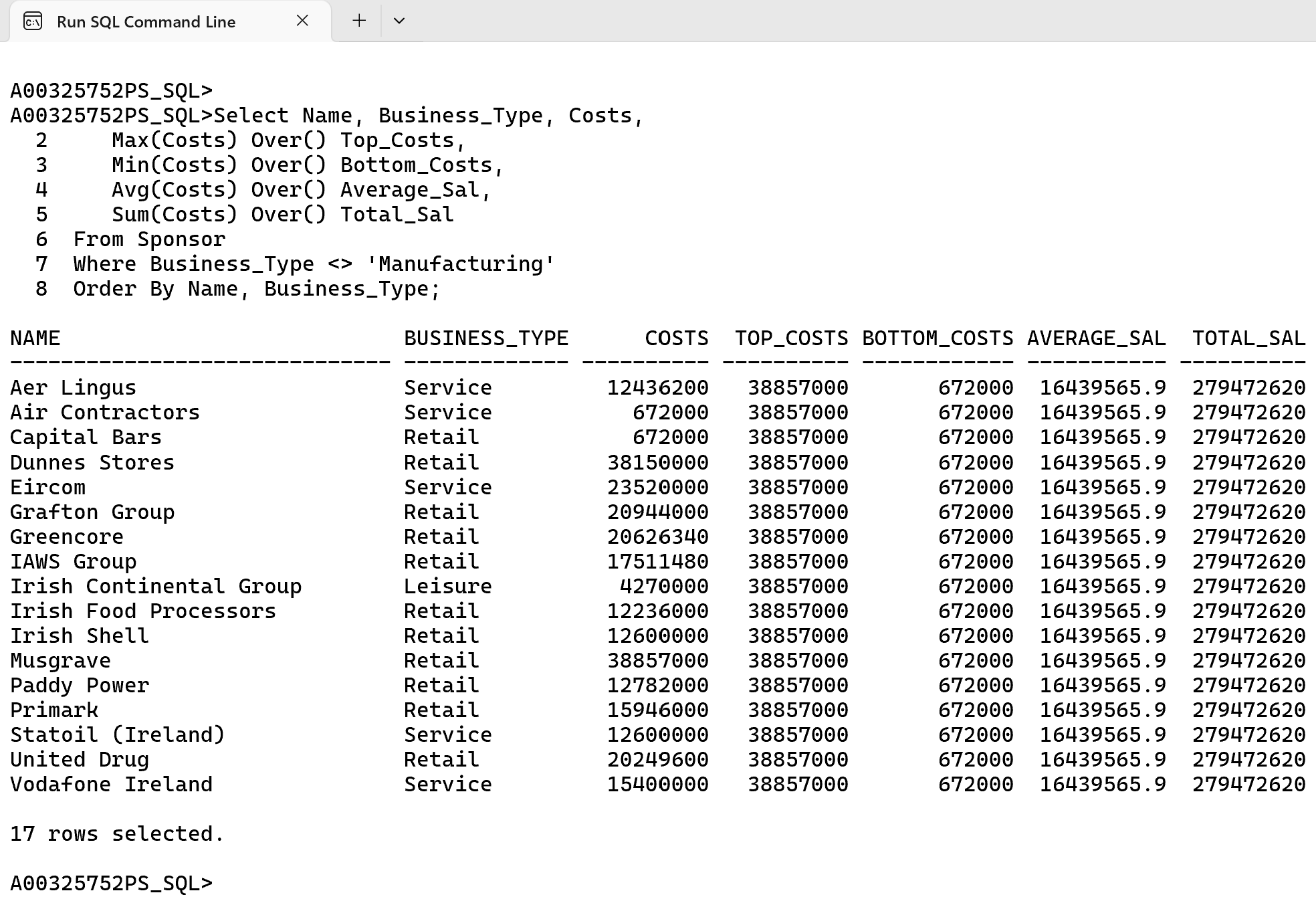
Avg(Costs) Over() **Average\_Sal**,

Sum(Costs) Over() **Total\_Sal**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Name, Business\_Type;



For this query we are using all the aggregate functions (Sum, Max, Min, Avg) along with the over clause.

**Over Clause with all aggregate functions**

**Example 5.** List the Name, Business\_Type and Costs for each individual Sponsor plus the lowest, highest, average and overall total costs along with the running total for every Business Type except Manufacturing. Sort the results by Business Type (ascending order) and then by Name.

Cl scr

Select Name, Business\_Type, Costs,

Max(Costs) Over(Order By Business\_Type) **Top\_Costs**,

Min(Costs) Over(Order By Business\_Type) **Bottom\_Costs**,

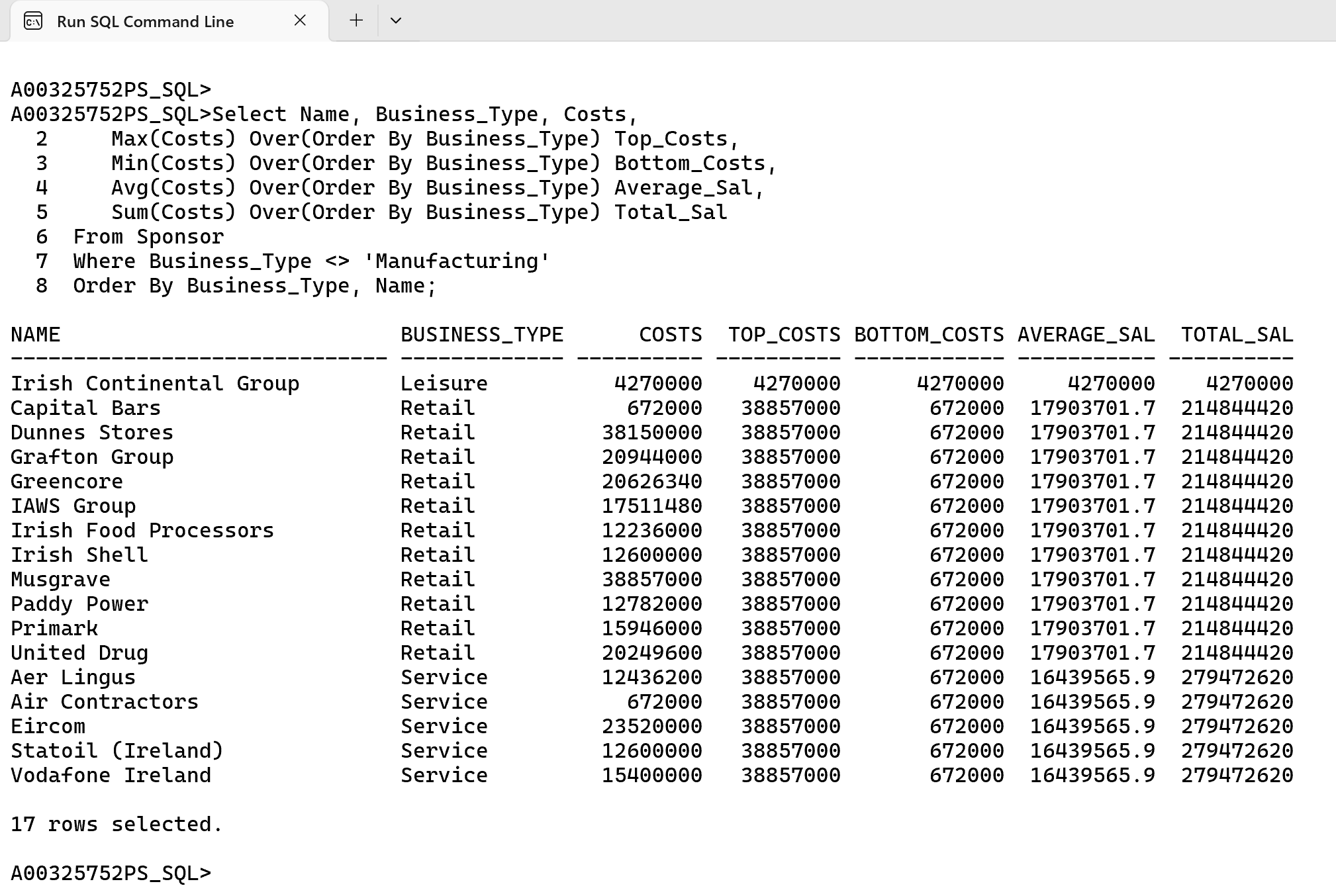
Avg(Costs) Over(Order By Business\_Type) **Average\_Sal**,

Sum(Costs) Over(Order By Business\_Type) **Total\_Sal**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type, Name;



In this query, the ORDER BY inside the OVER() clause organizes rows by Business\_Type for each window function, ensuring the aggregare calculations like MAX, MIN, AVG, and SUM are computed sequentially for all rows ordered by Business\_Type. This creates a running total and other aggregate values that adjust dynamically based on the sorted order.

**Over Clause with Order By**

**Example 6.** List the Name, Business\_Type and Costs for each individual Sponsor plus the lowest, highest, average and overall total costs for every Business Type except Manufacturing. Sort the results by Business Type **(descending order)** and then by Name.

Cl scr

Select Name, Business\_Type, Costs,

Max(Costs) Over(Order By Business\_Type Desc) **Top\_Costs**,

Min(Costs) Over(Order By Business\_Type Desc) **Bottom\_Costs**,

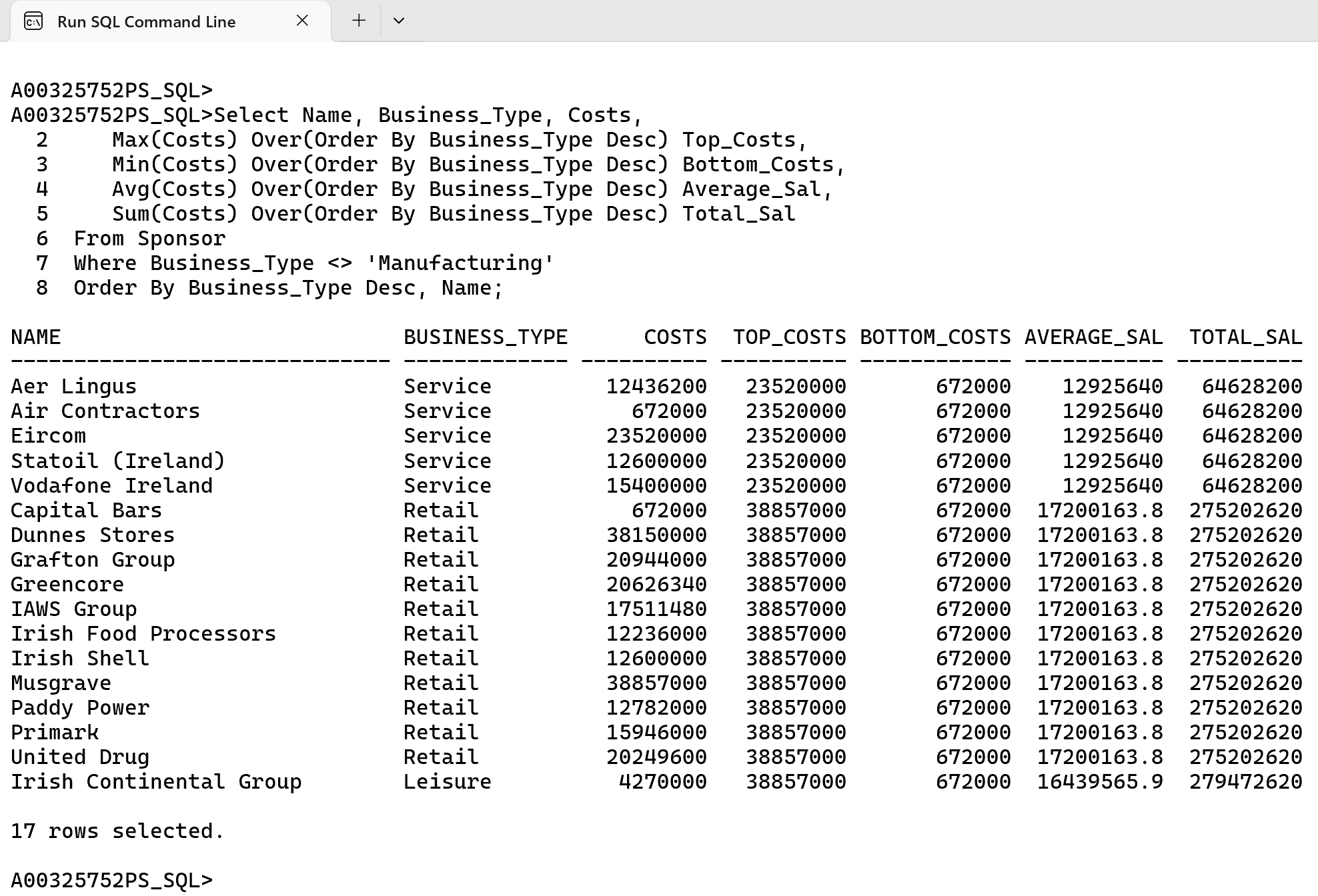
Avg(Costs) Over(Order By Business\_Type Desc) **Average\_Sal**,

Sum(Costs) Over(Order By Business\_Type Desc) **Total\_Sal**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type Desc, Name;



This query is similar to query from **Ex 5.** But since we are using Desc(Descending) the running totals are adjusted dynamically based on the descending order.

**Over Clause with Order By and all aggregate functions**

**Example 7.** List the Name, County, Business\_Type and Costs for each individual Sponsor plus the lowest, highest, average and overall total costs for every Business Type except Manufacturing. Order By County.

Cl scr

Select Name, County, Business\_Type, Costs,

Max(Costs) Over(Order By County) **Top\_Costs**,

Min(Costs) Over(Order By County) **Bottom\_Costs**,

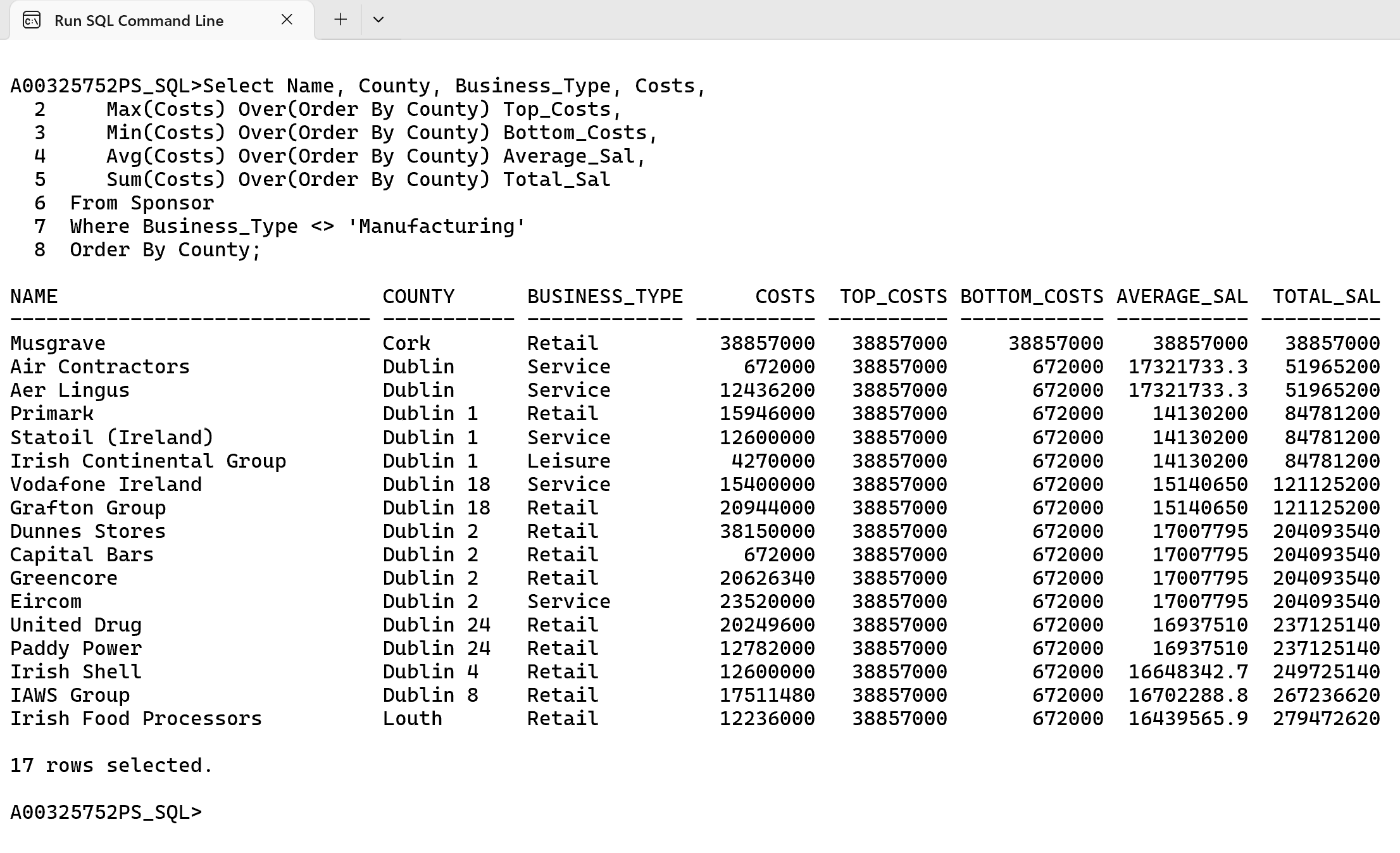
Avg(Costs) Over(Order By County) **Average\_Sal**,

Sum(Costs) Over(Order By County) **Total\_Sal**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By County;

****

In this query is the same as **Ex 6.** we are ordering it by county in order to get a good example of running totals.

**Over Clause with Order By with wrong Order By sequence**

**Example 8.** List the Name, County Business\_Type and Costs for each individual Sponsor plus the lowest, highest, average and overall total costs for every Business Type except Manufacturing. Order By **Name and Business\_Type**.

Cl scr

Select Name, County, Business\_Type, Costs,

Max(Costs) Over(Order By County) **Top\_Costs**,

Min(Costs) Over(Order By County) **Bottom\_Costs**,

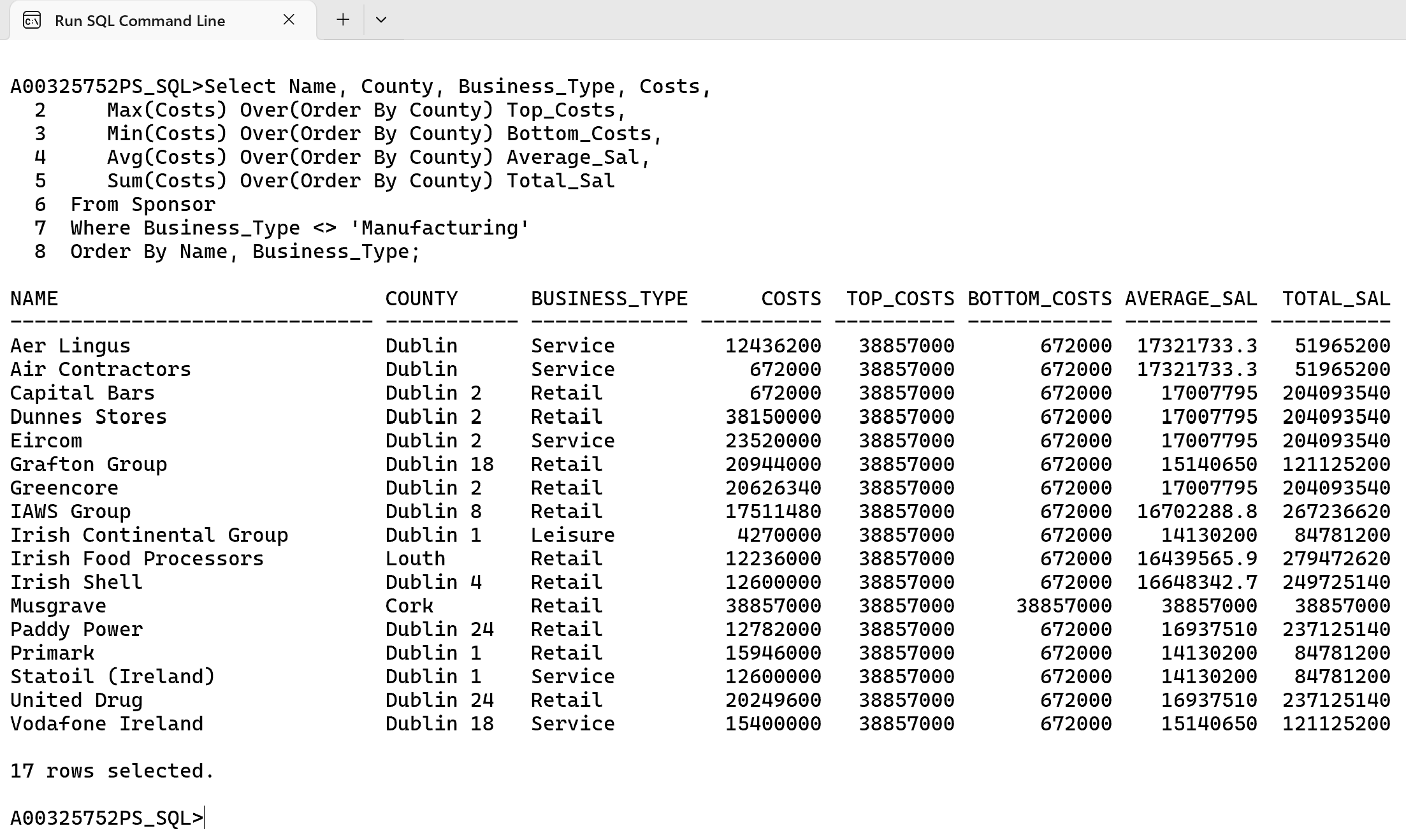
Avg(Costs) Over(Order By County) **Average\_Sal**,

Sum(Costs) Over(Order By County) **Total\_Sal**

From Sponsor

Where Business\_Type <> 'Manufacturing'

**Order By Name, Business\_Type;**

****

In the Over clause we are ordering it by County but in Order By Clause at the end of query we are ordering it by Name, Business Name. This generated query will require a lot of scrutiny before you will be make any sense out of them.

**Over Clause with Partition**

**Example 9.** List the Name, County Business Type and Costs for each individual Sponsor, as well as and a running costs total for business in county, plus the companies in each county by business type for every Business Type except Manufacturing.

Cl scr

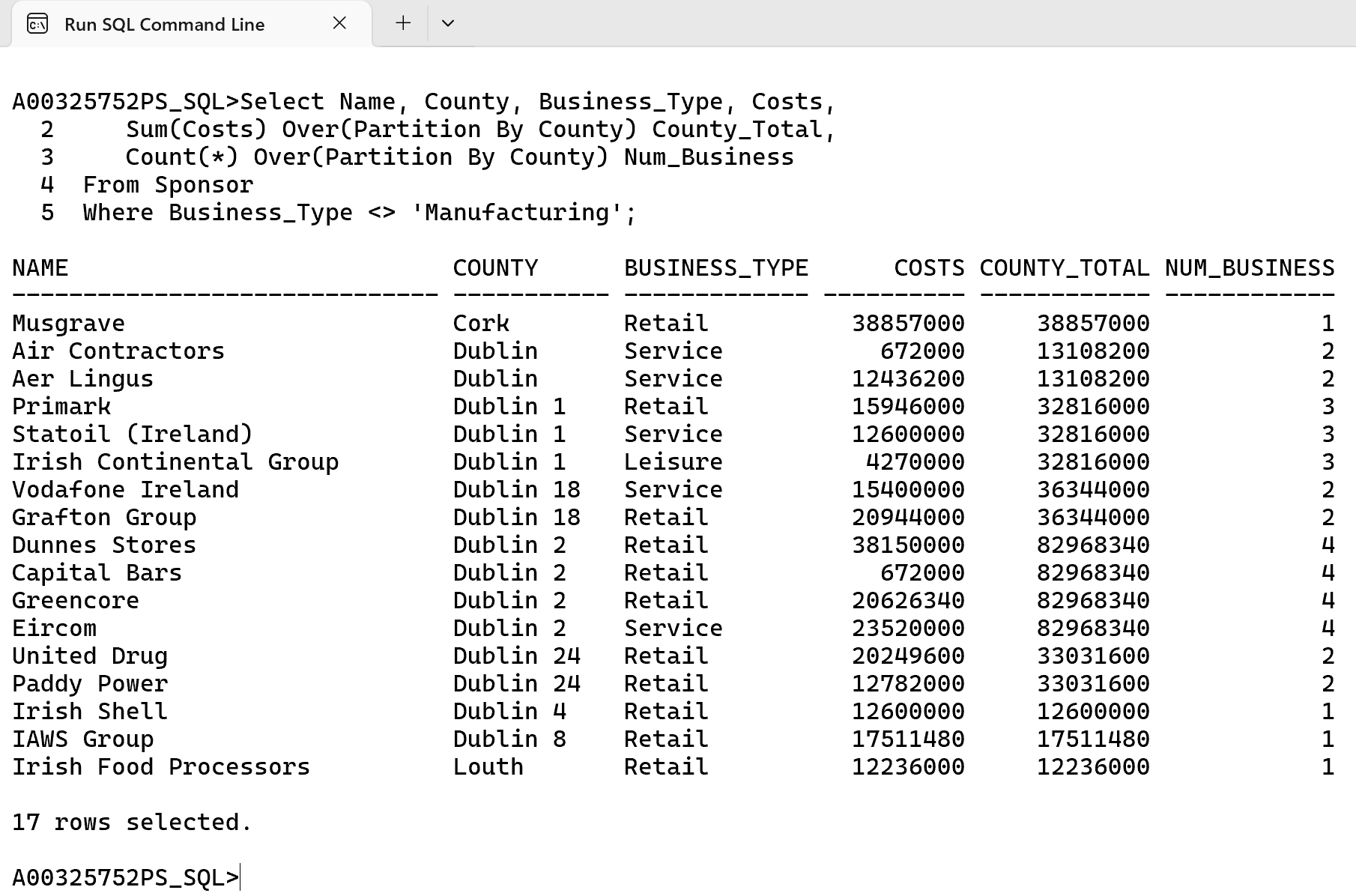
Select Name, County, Business\_Type, Costs,

Sum(Costs) Over(Partition By County) **County\_Total**,

Count(\*) Over(Partition By County) **Num\_Business**

From Sponsor

Where Business\_Type <> 'Manufacturing';



The PARTITION BY clause in the OVER() function here is similar to the Group By function in normal SQL query, this Partition By Clause groups rows by County. This allows aggregate functions like SUM and COUNT to compute totals and counts within each County while retaining row-level details.

**Over Clause with Partition and Order By**

**Example 10.** List the Name, County Business Type and Costs for each individual Sponsor, as well as and a running costs total for business in county, plus the companies in each county by business type for every Business Type except Manufacturing. Sort the results by County and then by Name

Cl scr

Select Name, County, Business\_Type, Costs,

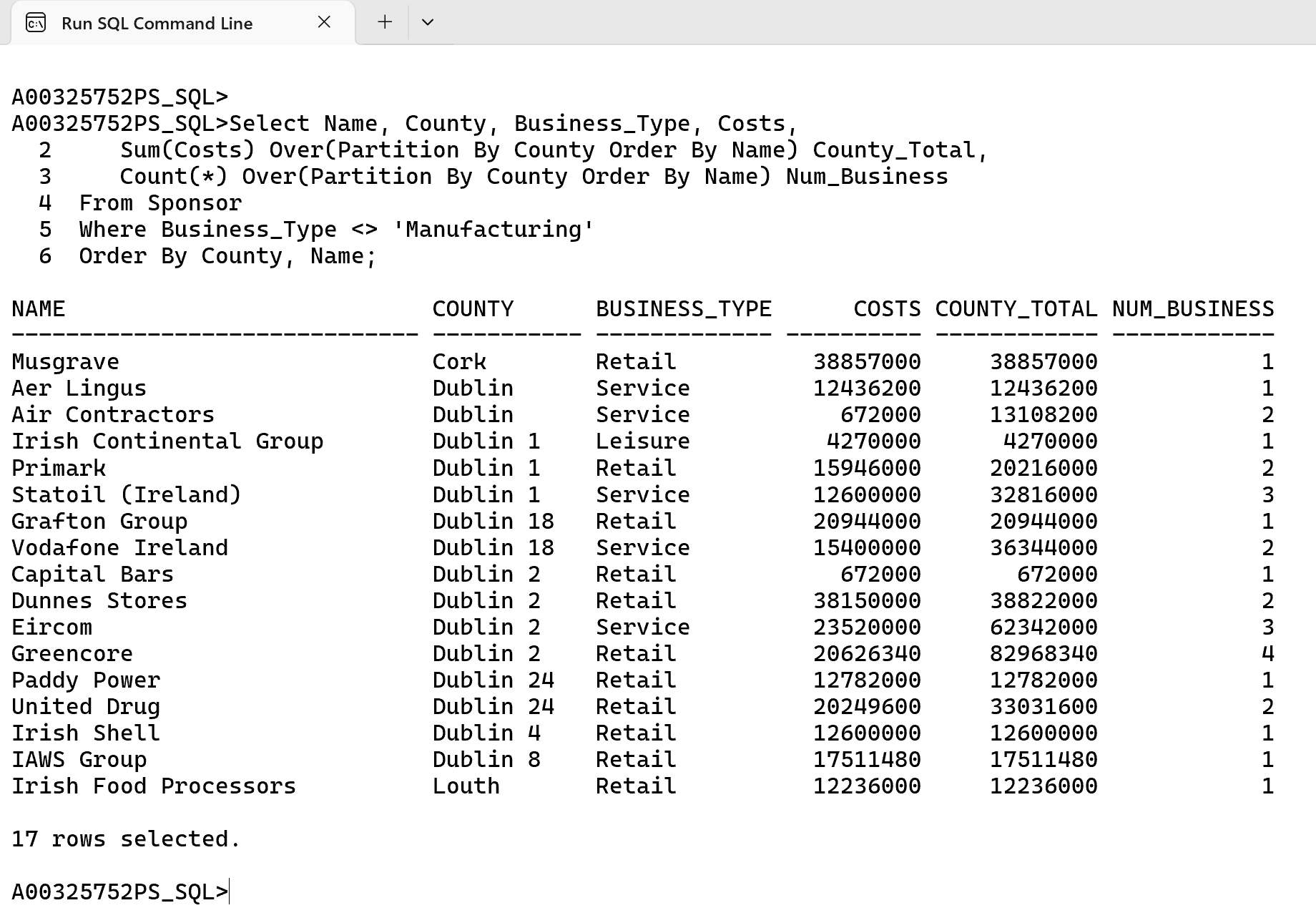
Sum(Costs) Over(Partition By County Order By Name) **County\_Total**,

Count(\*) Over(Partition By County Order By Name) **Num\_Business**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By County, Name;

****

In this query, the PARTITION BY County ORDER BY Name ensures that the running total (SUM) and count (COUNT) are calculated within each County, while also considering the ascending order of Name. This combination provides dynamic aggregates that respect both grouping by County and sorting by Name.

**Over Clause with Partition**

**Example 11.** List the Name, Business Type and Costs for each individual Sponsor, as well as and a running costs total, minimum costs and maximum costs for business type, for every Business Type except Manufacturing. Sort the results by Business Type and then by Name

Cl scr

Select Name, Business\_Type, Costs,

Sum(Costs) Over(Partition By Business\_Type) **Costs\_Total**,

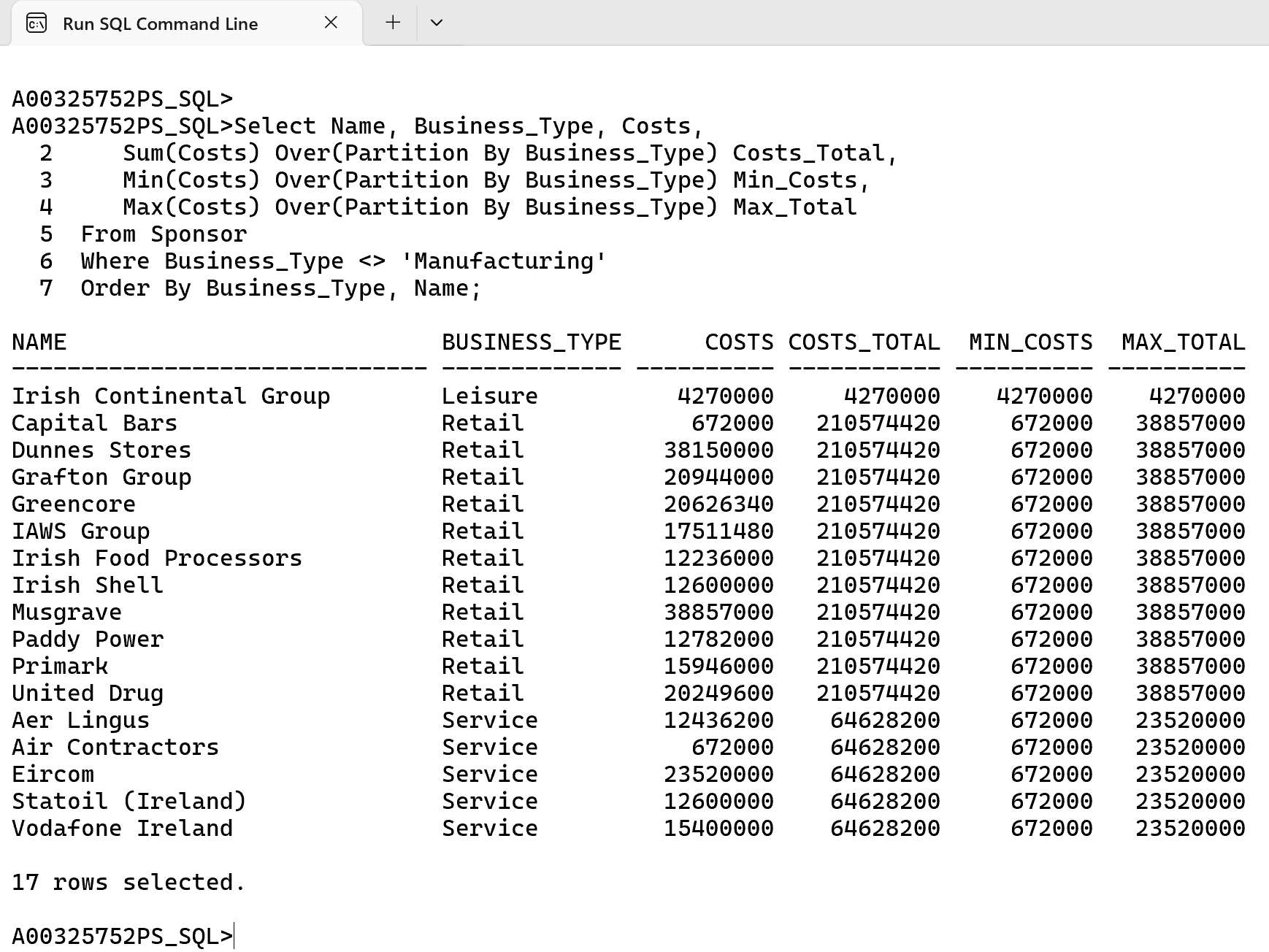
Min(Costs) Over(Partition By Business\_Type) **Min\_Costs**,

Max(Costs) Over(Partition By Business\_Type) **Max\_Total**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type, Name;



In this query, the PARTITION BY Business\_Type groups rows by Business\_Type and we are sorting by Business\_Type and Name ensures the results are organized for easier interpretation within each category.

**Over Clause with Partition and Order By**

**Example 12.** List the Name, Business Type and Costs for each individual Sponsor, as well as and a running costs total, minimum costs and maximum costs for business type, for every Business Type except Manufacturing. Sort the results by Business Type and then by Name

Cl scr

Break on Business\_Type Skip 1

Select Business\_Type, Name, Costs,

Sum(Costs) Over(Partition By Business\_Type Order By Name) **Costs\_Total**,

Min(Costs) Over(Partition By Business\_Type Order By Name) **Min\_Costs**,

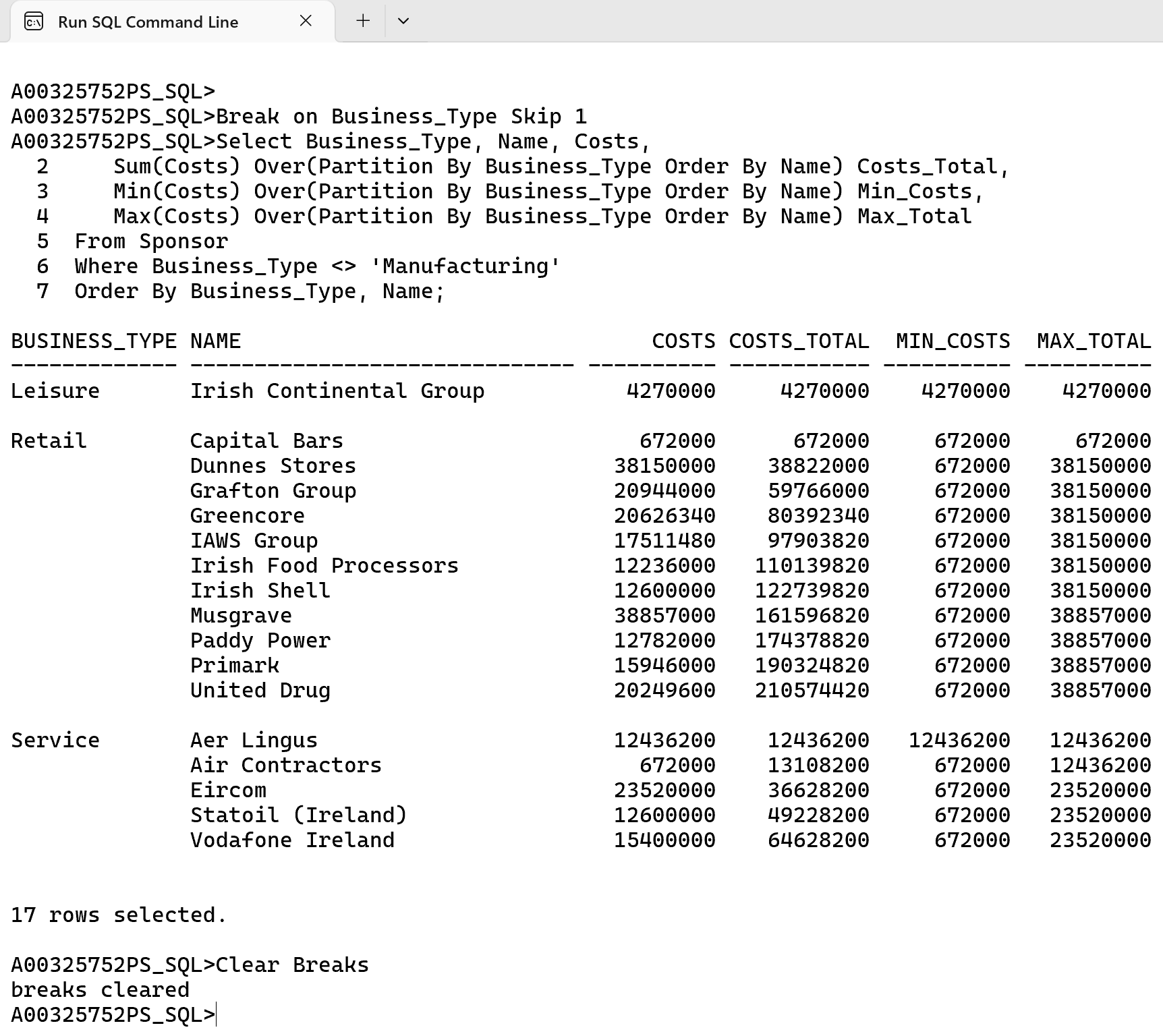
Max(Costs) Over(Partition By Business\_Type Order By Name) **Max\_Total**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type, Name;

Clear Breaks



This query is similar to the one we have done previously but by using Break on Business Type gives a better view for viewing the data.

**Over Clause with Partition, Order By and Windowing**

**Example 13.** List the Name, Business Type and Costs for each individual Sponsor, as well as and a running cost for current Name plus the two previous names processed at the same Business type, filter every Business Type except Manufacturing. Sort the results by Business Type and then by Name.

Cl scr

Break on Business\_Type Skip 1

Select Business\_Type, Name, Costs,

Sum(Costs) Over(Partition By Business\_Type Order By Name

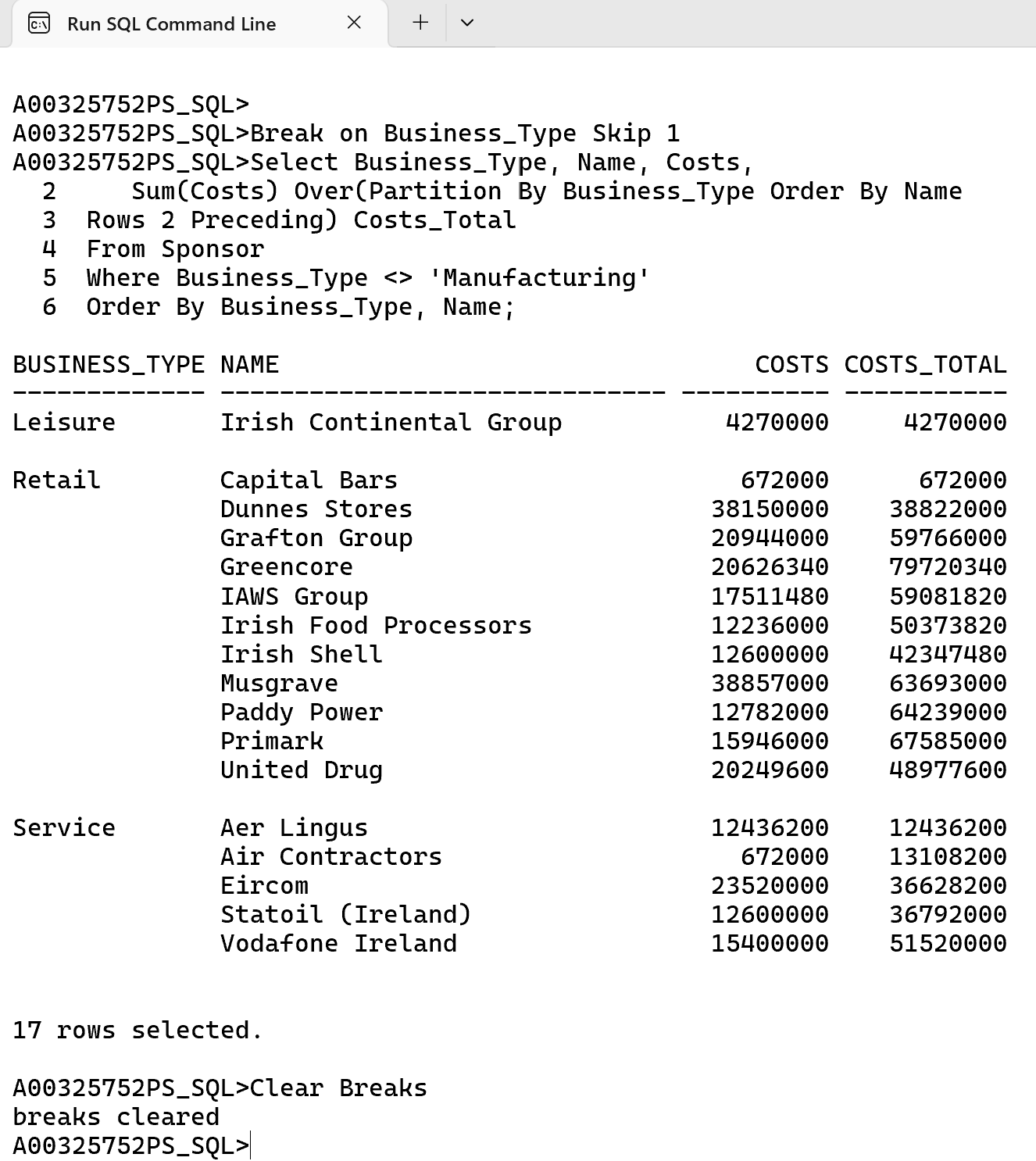
Rows 2 Preceding) **Costs\_Total**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type, Name;

Clear Breaks



This query we have used Rows 2 Preceding so that the running total will only add up to the values of it’s previous 2 rows Partition by Business\_Type.

**Over Clause with Partition, Order By and Windowing**

**Example 14.** List the Title, Discipline, Payment Date and Costs for each individual Scholarship, as well as and a running annual\_value for current scholarship from same discipline whose Payment date is 180 days preceding. Filter the results from 01- Jan 2024 to 31- Dec 2024. Sort the results by Business Type and then by Name.

Cl scr

Break on Discipline Skip 1

Select Discipline, Title, Initial\_Payment\_Date, Annual\_Value,

Sum(Annual\_Value) Over(Partition By Discipline Order By Initial\_Payment\_Date

Range 180 Preceding) **Annual\_Value\_Total**

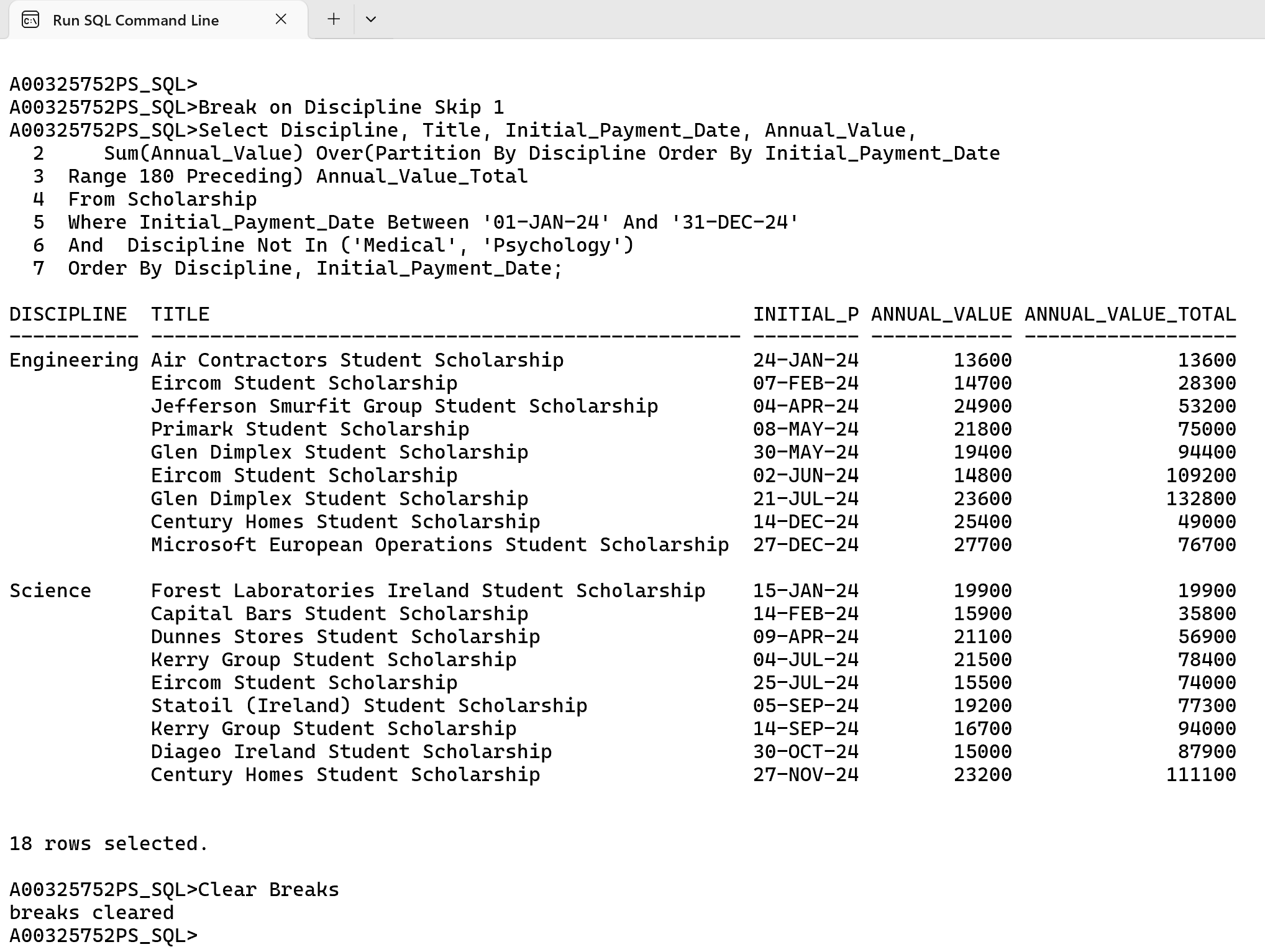
From Scholarship

Where Initial\_Payment\_Date Between '01-JAN-24' And '31-DEC-24'

And Discipline Not In ('Medical', 'Psychology')

Order By Discipline, Initial\_Payment\_Date;

Clear Breaks



This query we have used Range 180 Preceding so that the running total will only add up to the annual value only for those whose date is 180 days before the Initial\_Payment\_Date.

**Over Clause with Partition, Order By and Windowing**

**Example 15.** List the Title, Discipline, Payment Date and Costs for each individual Scholarship, as well as and a running annual\_value for current scholarship from same discipline whose Payment date is before and after the date in consideration (i.e +90 and -90 days). Filter the results from 01- Jan 2024 to 31- Dec 2024. Sort the results by Business Type and then by Name.

Cl scr

Break on Discipline Skip 1

Select Discipline, Title, Initial\_Payment\_Date, Annual\_Value,

Sum(Annual\_Value) Over(Partition By Discipline Order By Initial\_Payment\_Date

Range Between 90 Preceding And 90 Following) **Annual\_Value\_Total**

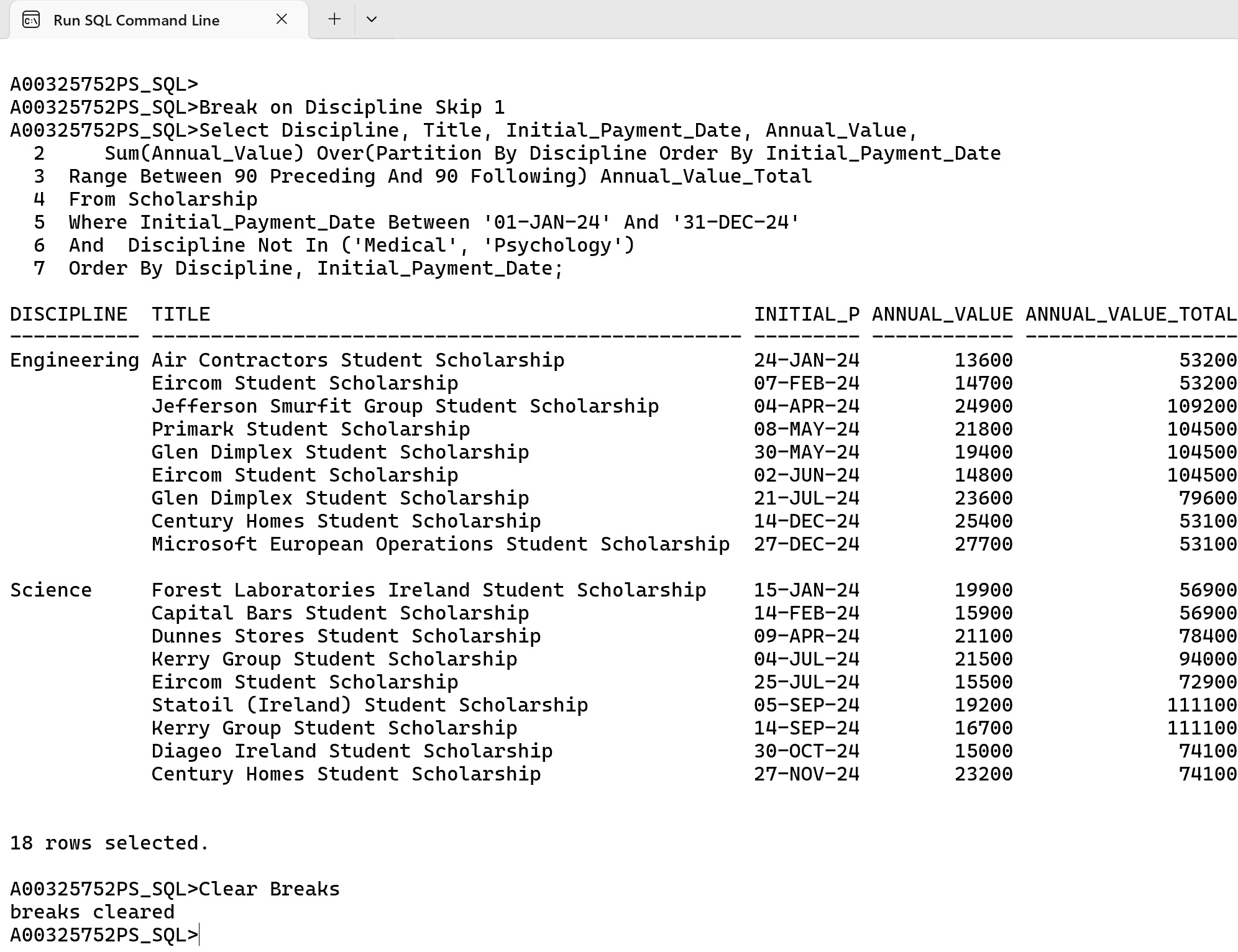
From Scholarship

Where Initial\_Payment\_Date Between '01-JAN-24' And '31-DEC-24'

And Discipline Not In ('Medical', 'Psychology')

Order By Discipline, Initial\_Payment\_Date;

Clear Breaks

****

This query is same as Example 14. But we are using initial\_payment\_date preceding and following of 90 days to add up into running total for annual payment.

**Over Clause with Partition, Order By and Windowing**

**Example 16.** Irrespective of the Department, **l**ist the Title, Payment Date and Day before 100 days of Payment Date. Examine the Payment Date for Title under consideration and display the Title and Payment Date of the first Title during the 100-day period that fell for current title (*there may be more than one title whose payment date is in the 100 days window but we are interested in the first of those*). Filter the results from 01- Jan 2024 to 31- Dec 2024 and discipline not equal to Medical or Psychology. Sort the results by Payment\_Date

Cl scr

Select Title, Initial\_Payment\_Date, Initial\_Payment\_Date - 100 Pay\_100, Annual\_Value,

First\_Value(Title) Over (Order By Initial\_Payment\_Date Asc

Range 100 Preceding) **Scholarship\_First**,

First\_Value(Initial\_Payment\_Date) Over (Order By Initial\_Payment\_Date Asc

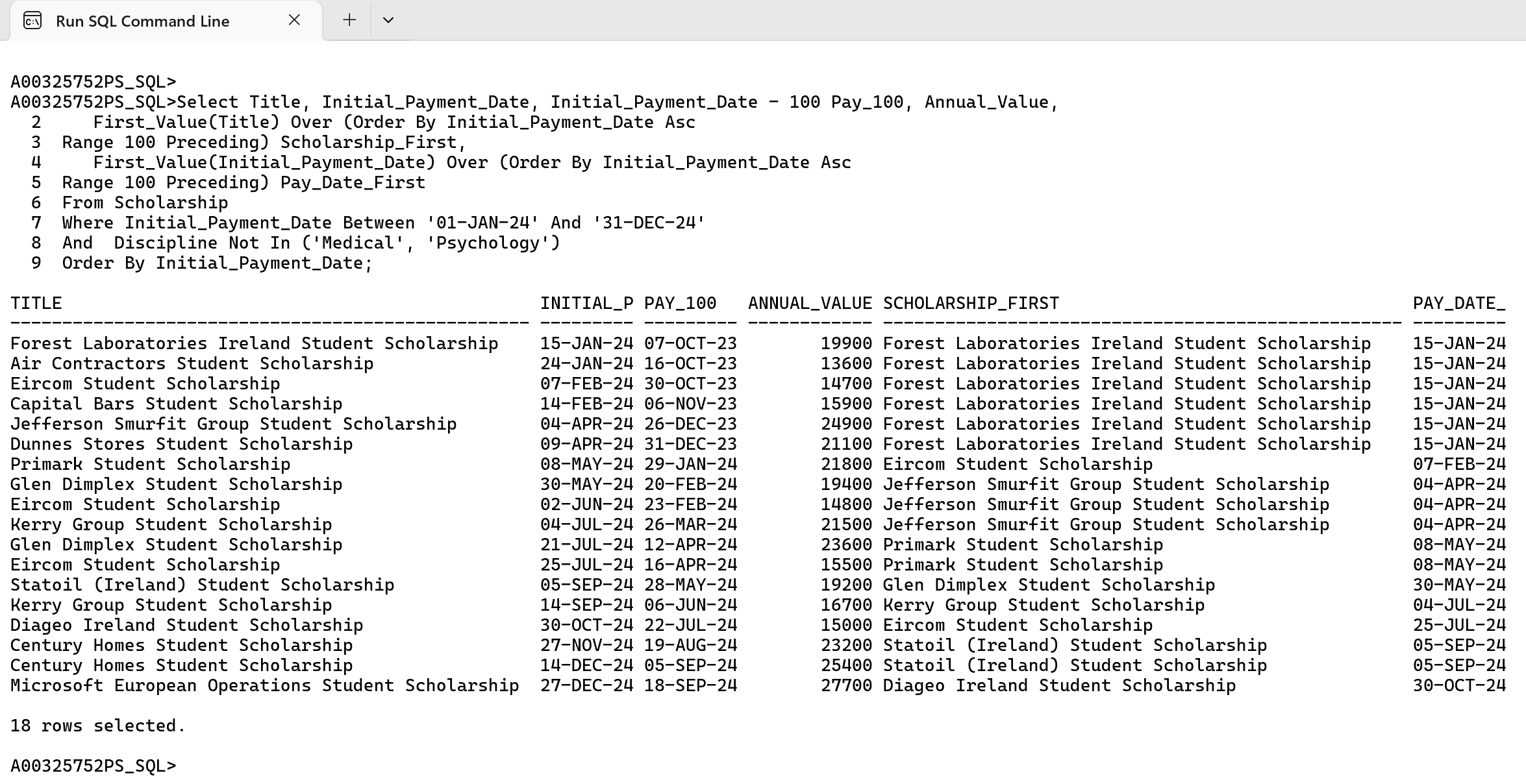
Range 100 Preceding) **Pay\_Date\_First**

From Scholarship

Where Initial\_Payment\_Date Between '01-JAN-24' And '31-DEC-24'

And Discipline Not In ('Medical', 'Psychology')

Order By Initial\_Payment\_Date;



This query identifies the first title and payment date within a 100-day window for each title, based on the Initial\_Payment\_Date. The FIRST\_VALUE() function, combined with the ORDER BY and RANGE 100 PRECEDING, retrieves the earliest title and payment date within the rolling 100-day window for the current row

**Over Clause with Partition, Order By and Windowing**

**Example 17.** Irrespective of the Department, **l**ist the Title, Payment Date and Day before 100 days of Payment Date. Examine the Payment Date for Title under consideration and display the average Annual Value from Title where fell on 30 days or before and likewise show average Annual value for Title which fell on 30 days after the Payment date was initiated. Sort the output by “before Initial Payment Date” or “after Initial Payment Date” . Filter the results from 01- Jan 2024 to 31- Dec 2024 and discipline not equal to Medical or Psychology. Sort the results by Payment\_Date

Cl scr

Select Title, Initial\_Payment\_Date,

**Initial\_Payment\_Date - 30** “Date – 30”,

**Initial\_Payment\_Date + 30** “Date + 30”, Annual\_Value,

Avg(Annual\_Value) Over (Order By Initial\_Payment\_Date Asc

Range 30 Preceding) **Avg\_Sal\_30Dy\_Before**,

Avg(Annual\_Value) Over (Order By Initial\_Payment\_Date Desc

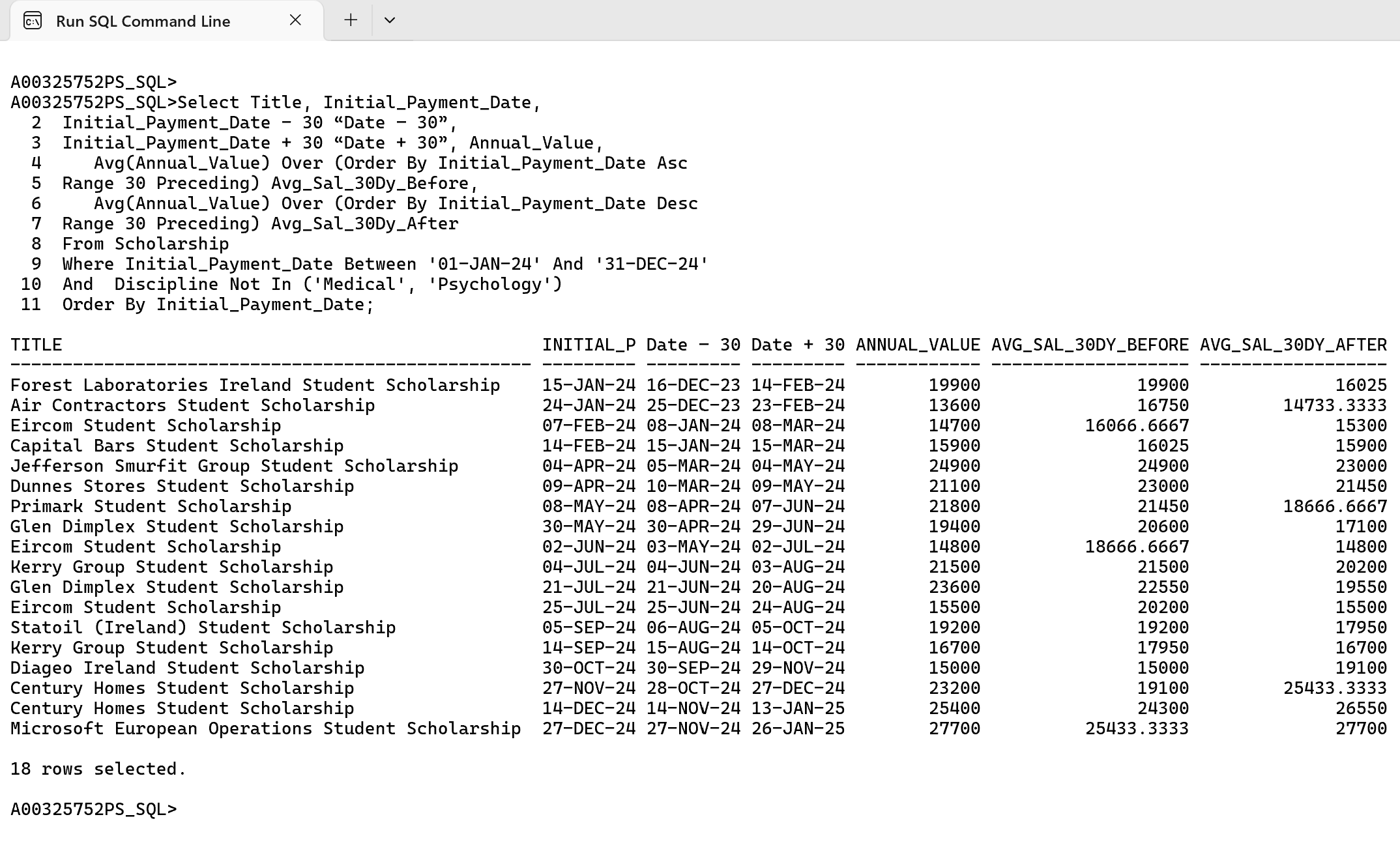
Range 30 Preceding) **Avg\_Sal\_30Dy\_After**

From Scholarship

Where Initial\_Payment\_Date Between '01-JAN-24' And '31-DEC-24'

And Discipline Not In ('Medical', 'Psychology')

Order By Initial\_Payment\_Date;



This query computes the average Annual\_Value for a rolling 30-day window before and after each Initial\_Payment\_Date, helping analyze trends around payment initiation. Filtering ensures only relevant disciplines and dates are included, and results are sorted chronologically by Initial\_Payment\_Date.

**Over Clause with Partition By**

**Example 18:** List the Name, Business Type, Main Market, Costs and Town for each individual Sponsor, as well as costs total, minimum costs broken by Business Type, Average cost broken by market and maximum costs for business Town, for every Business Type except Manufacturing. Sort the results by Business Type.

Cl scr

Select Name, Business\_Type, Main\_Market, Costs, Town,

Sum(Costs) Over(Partition By Business\_Type) **Sum\_Business\_Types**,

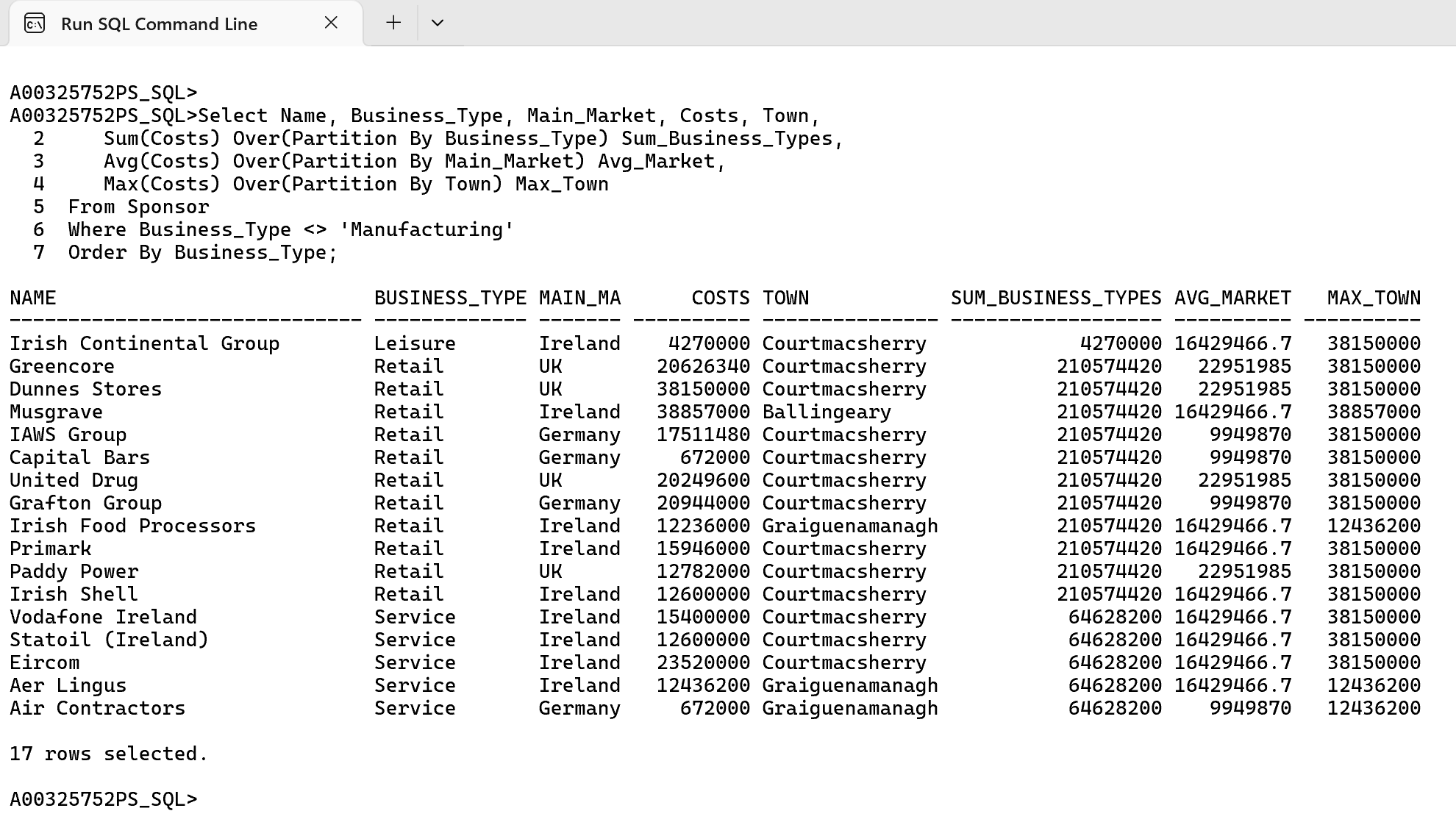
Avg(Costs) Over(Partition By Main\_Market) **Avg\_Market**,

Max(Costs) Over(Partition By Town) **Max\_Town**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Business\_Type;



This query calculates the total costs by Business\_Type, average costs by Main\_Market, and maximum costs by Town using the PARTITION BY clause. These aggregations retain row-level details while grouping values logically, excluding 'Manufacturing' from Business\_Type and sorting results by Business\_Type.

**Over Clause with Partition, Order By and Rank Function**

**Example 19.** List the Name from Sponsor and Annual\_Value, Discipline from Scholarship as well as the rank for Annual Value partitioned by Discipline. Limit the results to top 3 ranks only and sort the list by Discipline and Annual Value.

Cl scr

Break on Discipline Skip 1

Select \*

From (Select SP.Name, SC.Annual\_Value, SC.Discipline,

Dense\_Rank() Over (Partition By SC.Discipline Order By Annual\_Value Desc) As **Rank**

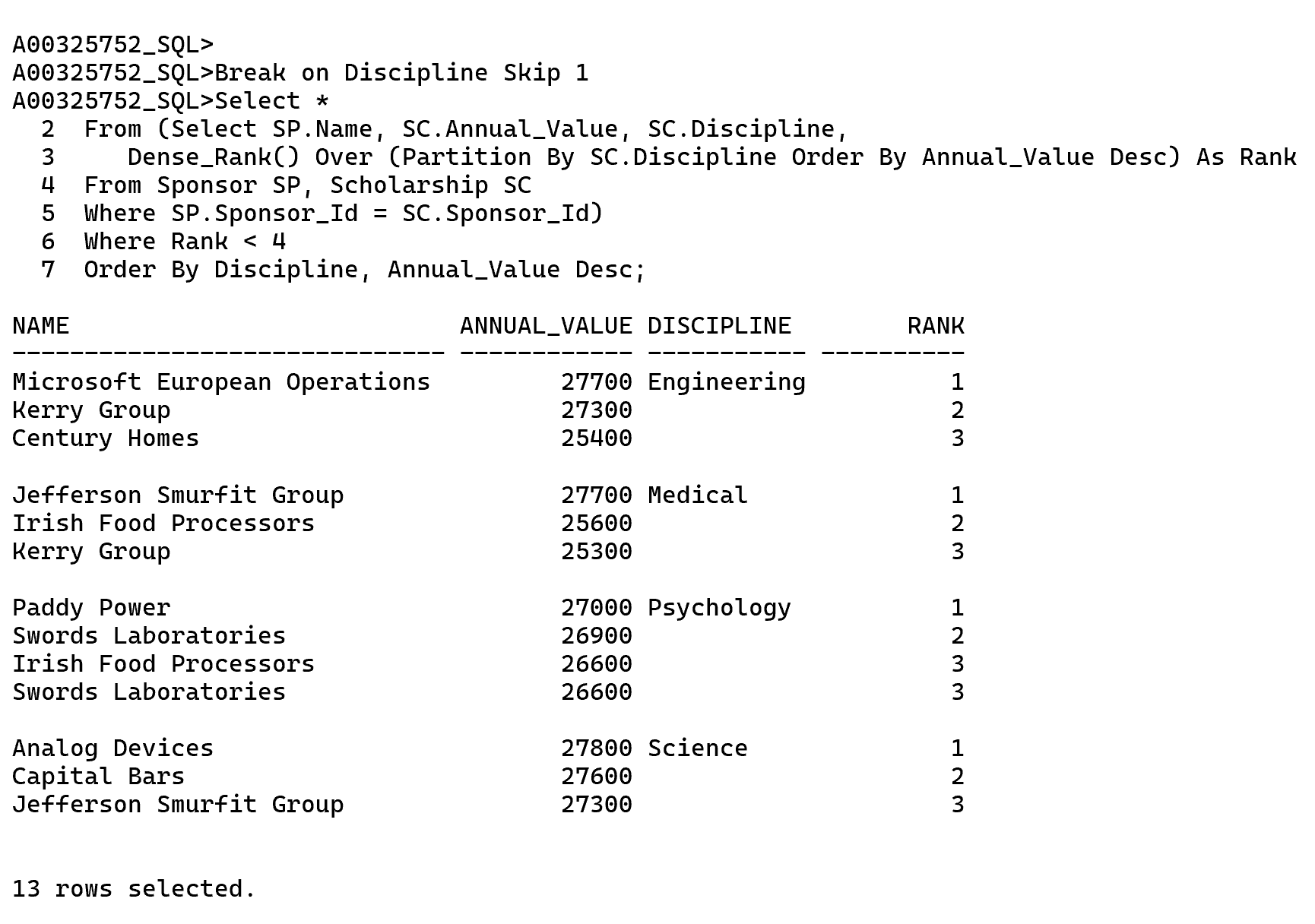
From Sponsor SP, Scholarship SC

Where **SP.Sponsor\_Id = SC.Sponsor\_Id**)

Where Rank < 4

Order By Discipline, Annual\_Value Desc;

Clear Breaks



In this query we are using Dense\_Rank to Rank the Annual Value based on Discipline. We have used Dense Rank instead of Rank to give a proper ranking without skipping any rank in between. We are limiting the results to top 3 ranks in this list to achieve this we have used in-line views so that we can add a condition on to the rank column.

**Example 19 Cont…**

Cl scr

Select SP.Name, SC.Annual\_Value, SC.Discipline,

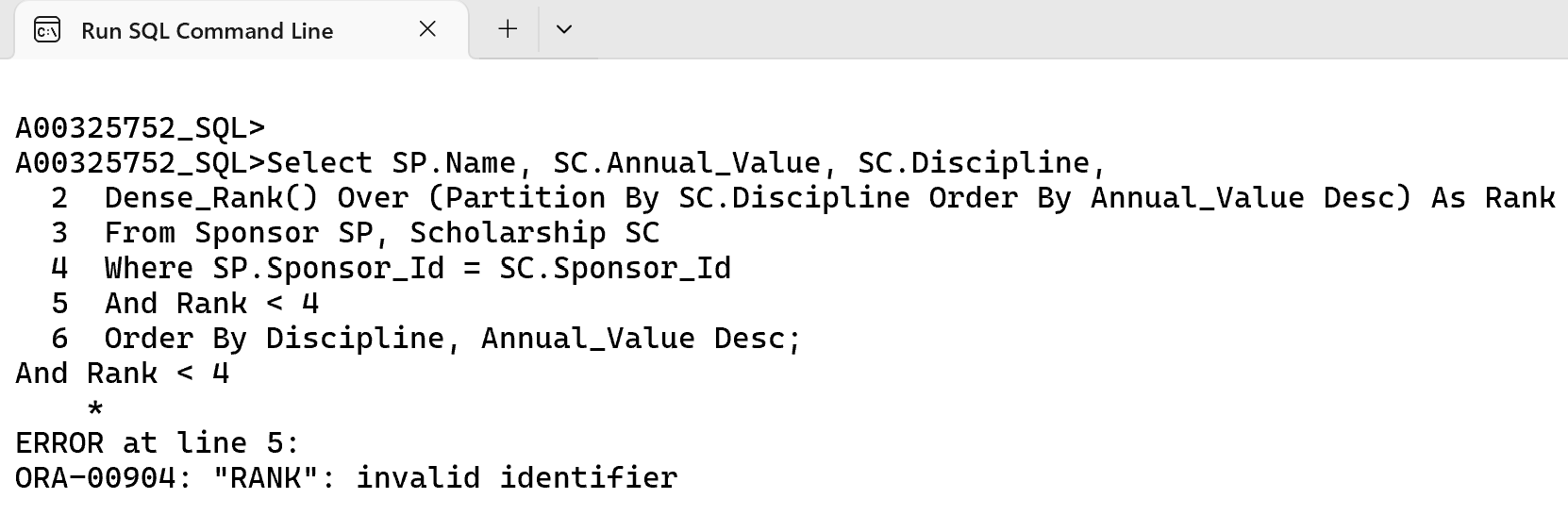
Dense\_Rank() Over (Partition By SC.Discipline Order By Annual\_Value Desc) As **Rank**

From Sponsor SP, Scholarship SC

Where SP.Sponsor\_Id = SC.Sponsor\_Id

**And Rank < 4**

Order By Discipline, Annual\_Value Desc;



The above query cannot be done without the inline view since the rank is getting processed inside the select column and Oracle doesn’t allow alias from the select column to be listed in the where clause.

**Over Clause with Partition, Order By and Rank Function**

**Example 20.** List the Name from Sponsor, Discipline from Scholarship as well as the rank for sum of Annual Value partitioned by Name and Discipline along with the Sum for Annual value partitioned by Name and Discipline. Limit the results to top 1 rank only.

Cl scr

Break on Discipline Skip 1

Select \*

From (Select Name, Discipline, Sum\_Annual\_Value,

Dense\_Rank() Over (Partition By Discipline Order By Sum\_Annual\_Value Desc) As **Rank**

From(Select SP.Name, SC.Annual\_Value, SC.Discipline,

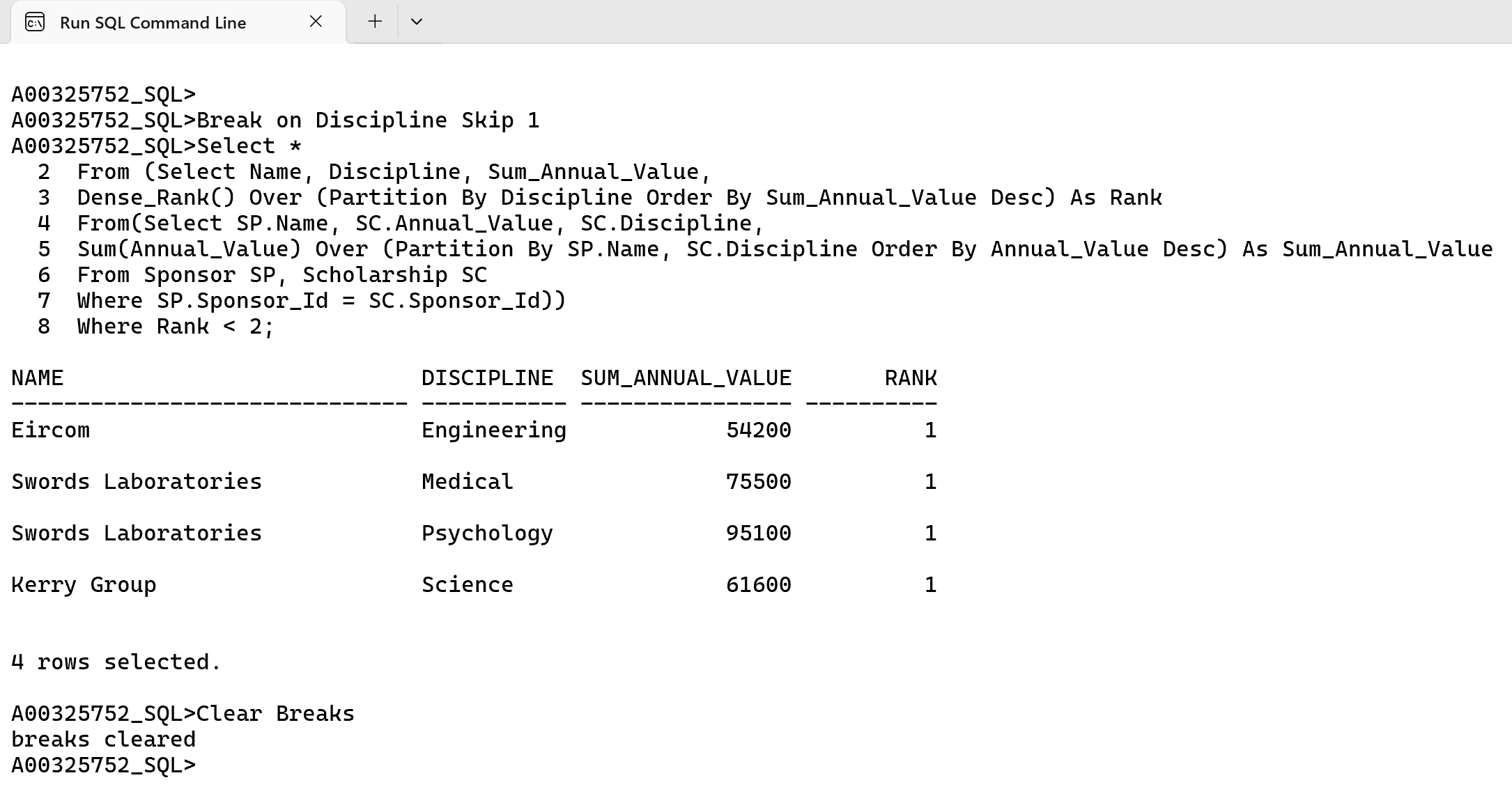
Sum(Annual\_Value) Over (Partition By SP.Name, SC.Discipline Order By Annual\_Value Desc) As **Sum\_Annual\_Value**

From Sponsor SP, Scholarship SC

Where **SP.Sponsor\_Id = SC.Sponsor\_Id)**)

Where Rank < 2;

Clear Breaks



In the above query I am using Rank to rank the top 1 sponsor among various discipline. This query runs on 2 inline views where in first I am showing the Rank and Sum Value and in second I have taken Sum\_Annual\_Value in second inline query so that I can process it in the first inline view with Order By Sum\_Annual Value as Oracle does not process the alias referred in same inline view.

**Keep, Over Clause with Partition By and Rank Function**

**Example 21.** From the Scholarship table select title, discipline and select minimum and maximum value for annual value that is partitioned by discipline. Filter the results where annual value between 16500 and 19000 and discipline must be in Psychology or Science.

Cl scr

Select Title, Discipline, Annual\_Value,

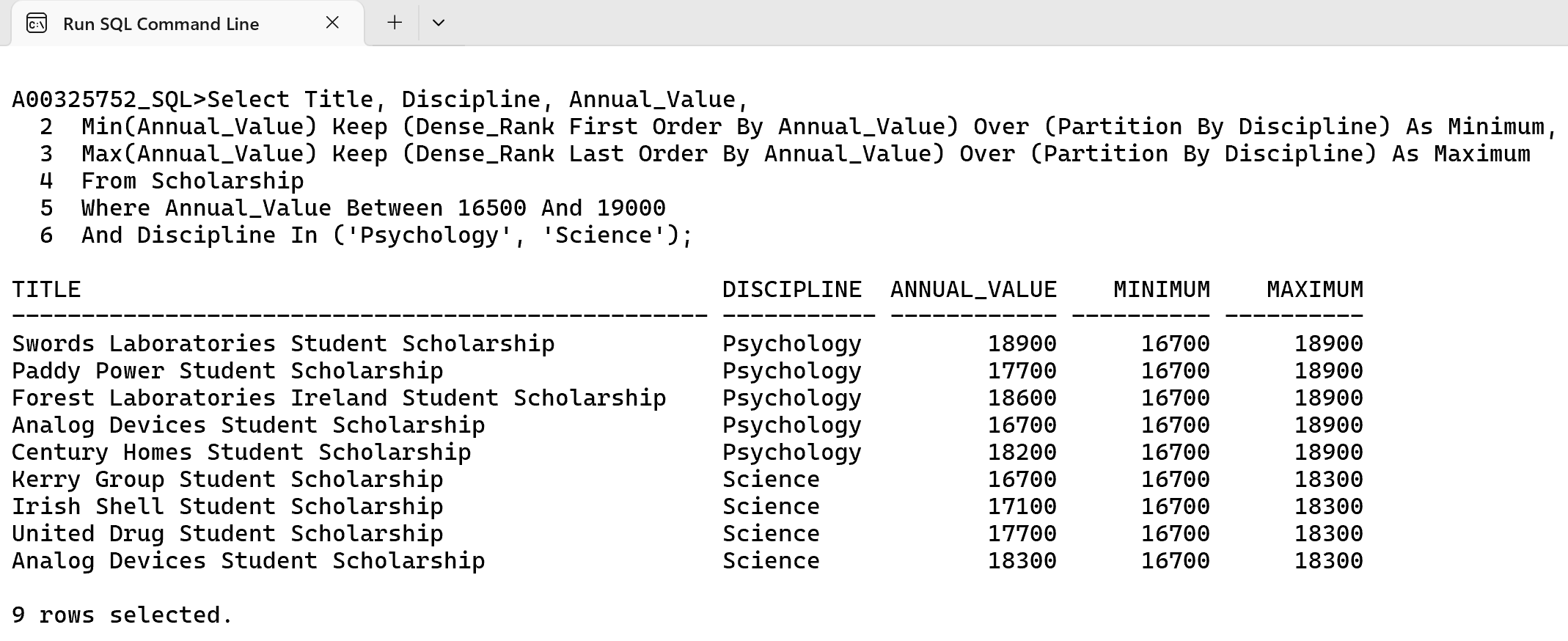
Min(Annual\_Value) Keep (Dense\_Rank First Order By Annual\_Value) Over (Partition By Discipline) As **Minimum**,

Max(Annual\_Value) Keep (Dense\_Rank Last Order By Annual\_Value) Over (Partition By Discipline) As **Maximum**

From Scholarship

Where Annual\_Value Between 16500 And 19000

And Discipline In ('Psychology', 'Science');



In this query the min and maximum summary functions are not used as it uses FIRST, LAST and KEEP that are picking the rows for which the values are being used. The Dense\_Rank here will take away confusion if there is a tie between two rows but this won’t affect the final results even if it is removed. This can be further simplified as below without the use of Dense\_Rank KEEP, FIRST and LAST.

**Example 21 Cont…**

Cl scr

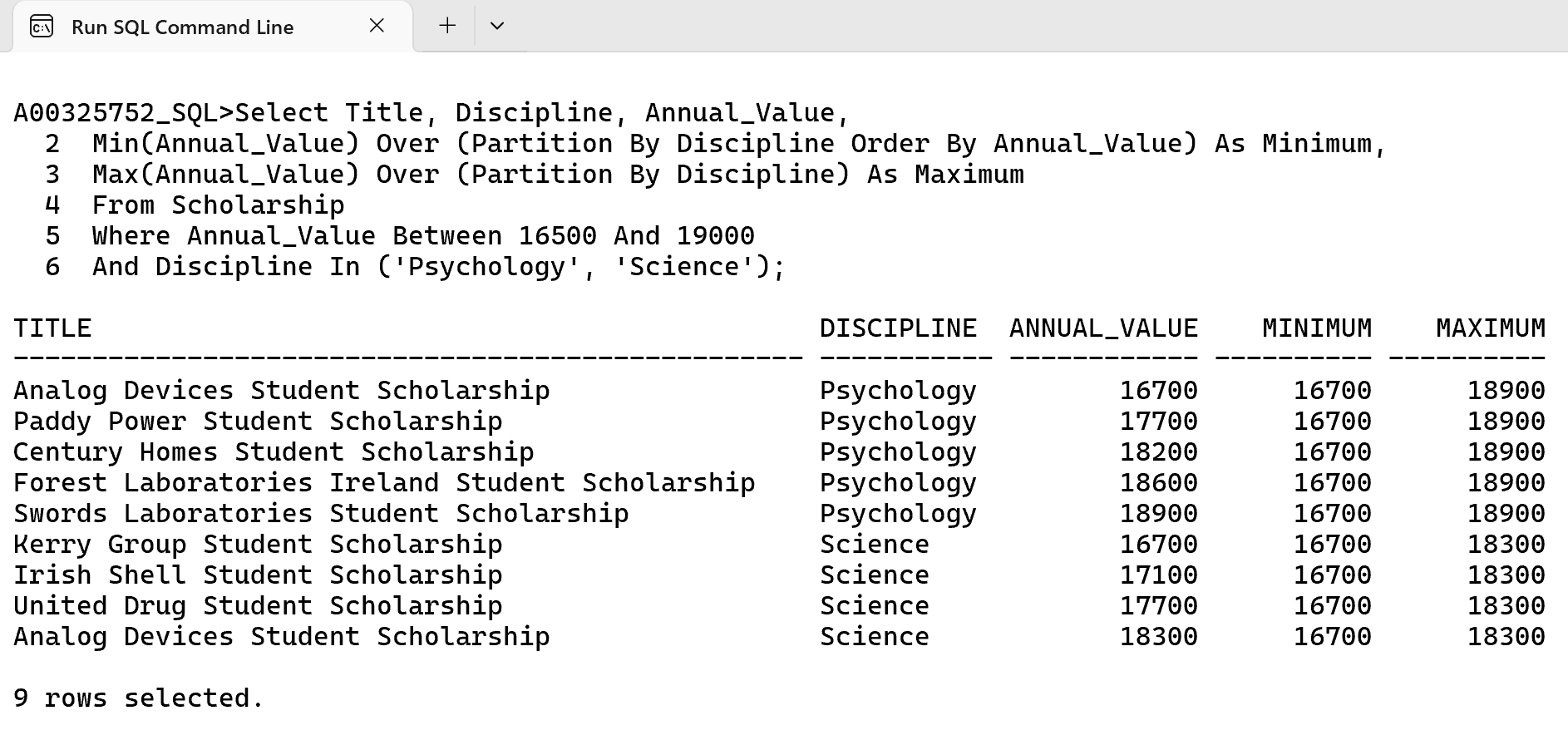
Select Title, Discipline, Annual\_Value,

Min(Annual\_Value) Over (Partition By Discipline Order By Annual\_Value) As **Minimum**,

Max(Annual\_Value) Over (Partition By Discipline) As **Maximum**

From Scholarship

Where Annual\_Value Between 16500 And 19000;



This will give the same output but this will query will be confused on which minimum and maximum value to take when there is a tie between two rows.

**Over Clause with Partition By, Ratio Function and Round Function**

**Example 22.** List the Name, Business Type, Main Market, Costs and Town for each individual Sponsor, as well as ratio of costs, maximum and minimum costs broken by Town, for every Business Type except Manufacturing. Sort the results by Business Type.

Cl scr

Break on Town Skip 1

Select Name, Business\_Type, Main\_Market, Costs, Town,

Round(Ratio\_To\_Report(Costs) Over(Partition By Town) \* 100, 2) As **Ratio\_Town** ,

Min(Costs) Over(Partition By Town) **Min\_Town**,

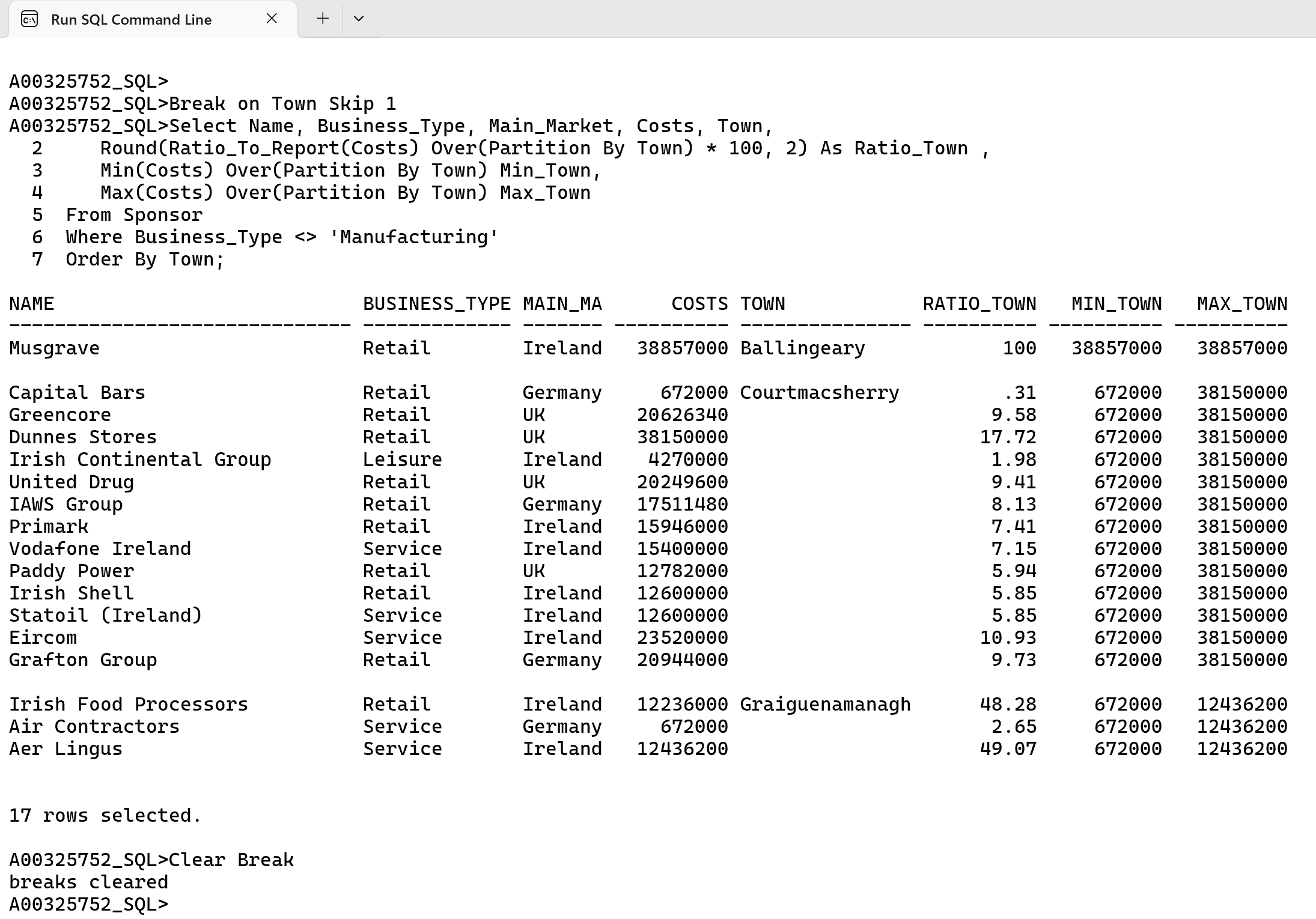
Max(Costs) Over(Partition By Town) **Max\_Town**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Town;

Clear Break



This query will give us total percent of ratio with the use of Ratio to Report function, minimum and maximum cost partitioned by Town. I have used the round function to round the ratio to report function into 2 decimal and multiplying it with 100 because it returns the value between 0 and 1.

**Over Clause with Partition By, Order By and Lag Function**

**Example 23.** List the Name, Town and Costs for each individual Sponsor, as well as and the cost for current Name, previous costs and costs difference between previous and current cost partition by Town and Group By Name, filter every Business\_Type except Manufacturing. Sort the results by Business Type and then by Name.

Cl scr

Break on Town Skip 1

Select Name, Town, Costs,

Lag(Costs,1,0) Over(Partition By Town Order By Name) **Cost\_Prev**,

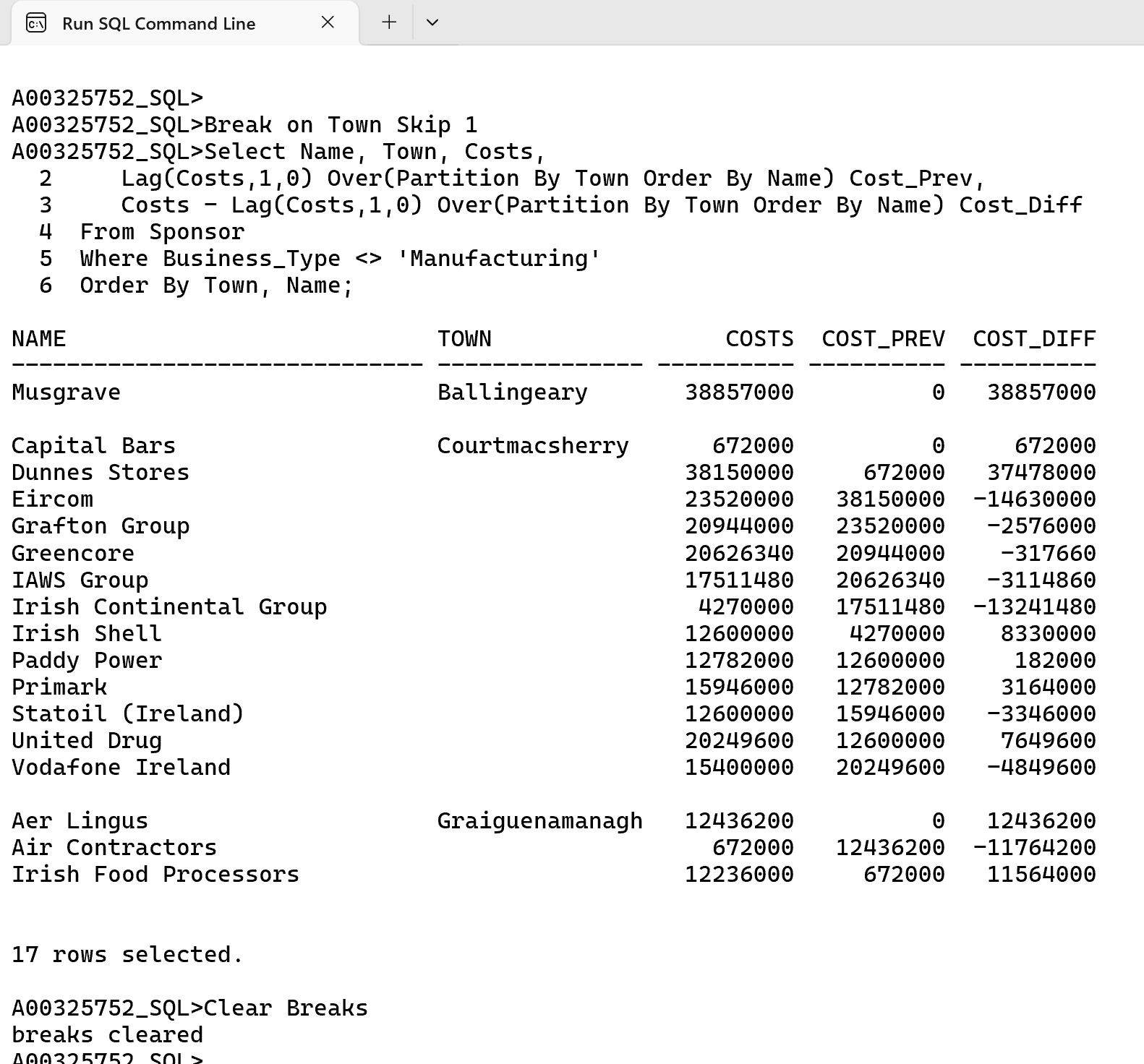
Costs - Lag(Costs,1,0) Over(Partition By Town Order By Name) **Cost\_Diff**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Town, Name;

Clear Breaks



In this query we are using Lag function to take the value from previous cost and we are subtracting it from the previous cost to get the difference. Lag(Costs,1,0) The costs here in syntax is column name, 1 is the row we need for lag the output and 0 is the default value if there are no to lag ahead it will take 0 as input.

**Over Clause with Partition By, Order By and Lead Function**

**Example 24.** List the Name, Town and Costs for each individual Sponsor, as well as and the cost for current Name, next costs and costs difference between next and current cost partition by Town and Group by Name, filter every Business\_Type except Manufacturing. Sort the results by Business Type and then by Name.

Cl scr

Break on Town Skip 1

Select Name, Town, Costs,

Lead(Costs,1,0) Over(Partition By Town Order By Name) **Cost\_Prev**,

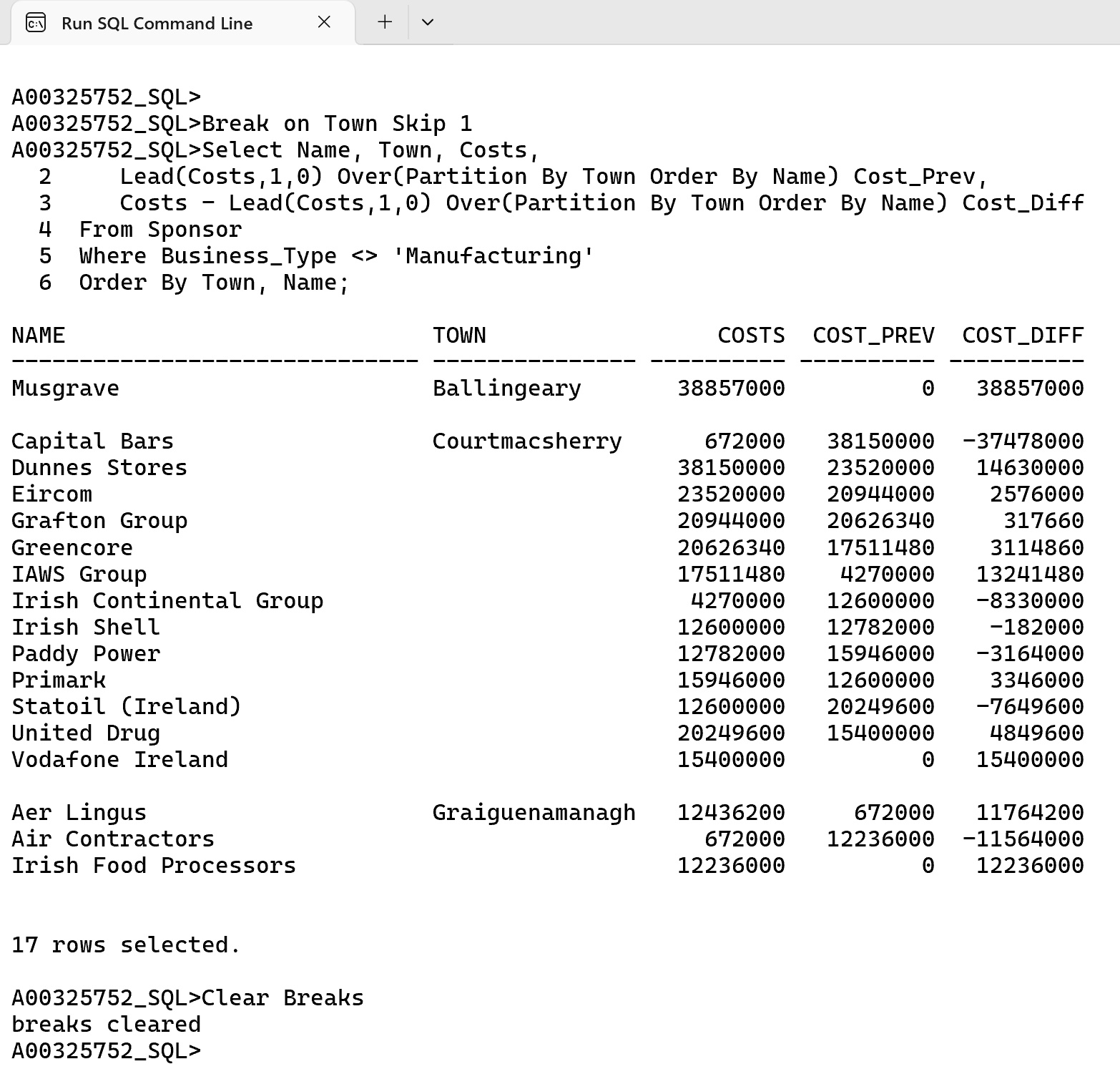
Costs - Lead(Costs,1,0) Over(Partition By Town Order By Name) **Cost\_Diff**

From Sponsor

Where Business\_Type <> 'Manufacturing'

Order By Town, Name;

Clear Breaks



This is the same as previous from Example 23 query but the only difference is we have a lead function here which takes the next value instead of the previous one.

**Conclusion**

Through Analytic queries in SQL. I have gained a strong foundation of SQL and analytic functions like Dense\_Rank(), Rank(), Lag(), Lead(), Min(), Max() and Sum() using Over Clause with Partition By, Order By and Windowing for advanced analysis by applying these skills to real-world scenarios like scholarship tracking.

**Youtube Video Link**

Link Video 1: <https://youtu.be/KhL8P3OeZNI>

Link Video 2: <https://youtu.be/SaYABtn7b0g>