

# SPRINT #2

## RELATIONAL DATABASES AND INTRODUCTION TO SQL

student:  
Leo Kalugin



Date: 25/09/2024

### SUMMARY

In this sprint, I will review the basic concepts of working with relational databases. I will begin hands-on experience with a database containing information from a company dedicated to selling products online. In this activity, I will focus on data related to completed transactions and the corporate information of the companies involved.

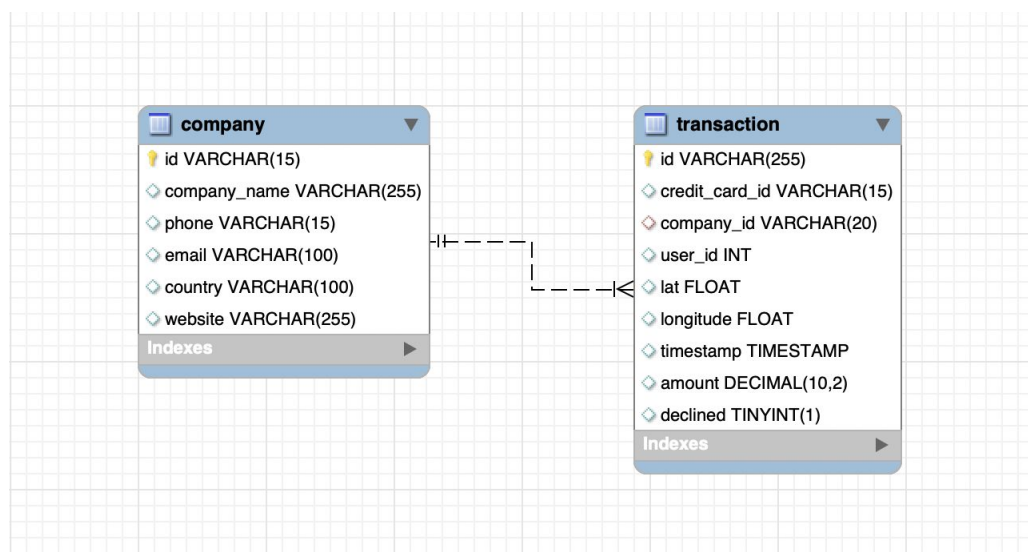
### RESULT

In this folder on the GitHub repository, you will find: the file **S2\_01.sql**, which contains all the scripts; **transactions\_schema.pdf**, which provides the database schema (also shown in this presentation); and this file, **Sprint\_2.pdf**, which includes screenshots of the work environment showing the queries I executed and the results obtained for each exercise:

[https://github.com/leocareer/DA\\_specialization/tree/main/Sprint\\_02](https://github.com/leocareer/DA_specialization/tree/main/Sprint_02)

### LEVEL 1 EXERCISE 1

From the attached documents ('data\_structure.sql' and 'db\_data.sql'), import the two tables. It shows the main features of the created schema and explains the different tables and variables that exist. Be sure to include a diagram that illustrates the relationship between the different tables and variables.



## LEVEL 1 EXERCISE 2

Using JOIN you will perform the following queries:

- List of countries that are shopping;
- From how many countries the purchases are made;
- Identify the company with the highest average sales.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings, along with a 'Limit to 5000 rows' dropdown. The SQL editor contains the following code:

```
4 -- List of countries that are shopping
5 • SELECT DISTINCT country 'Countries that are shopping'
6 FROM company AS t1
7 JOIN transaction AS t2
8 ON t1.id = t2.company_id
9 ORDER BY country;
10
```

Below the editor, the 'Result Grid' is displayed with the title 'Countries that are shopping'. It shows a list of 15 countries: Australia, Belgium, Canada, China, France, Germany, Ireland, Italy, Netherlands, New Zealand, Norway, Spain, Sweden, United Kingdom, and United States. The 'Action Output' pane at the bottom shows the execution details for 'Result 3':

	Time	Action	Response	Duration / Fetch Time
✓ 27...	11:39:42	SELECT DISTINCT country '...	15 row(s) returned	0.0015 sec / 0.00001...

The screenshot shows the same SQL IDE interface with the second query in the editor:

```
12 • SELECT count(DISTINCT country) 'How many countries are shopping'
13 FROM company AS t1
14 JOIN transaction AS t2
15 ON t1.id = t2.company_id;
16
```

The 'Result Grid' is titled 'How many countries are shopping' and displays a single result: 15. The 'Action Output' pane shows the execution details for 'Result 5':

	Time	Action	Response	Duration / Fetch Time
✓ 27...	11:43:44	SELECT count(DISTINCT co...	1 row(s) returned	0.0025 sec / 0.00000...

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 5000 rows' dropdown. The SQL editor contains the following query:

```
17 -- Identify the company with the highest average sales (solution with 'limit')
18 • SELECT company_name 'Company', ROUND(AVG(amount), 2) AS Average_sales
19 FROM transaction AS t1
20 JOIN company AS t2
21 ON t2.id = t1.company_id
22 GROUP BY t1.company_id
23 ORDER BY Average_sales DESC
24 LIMIT 1;
25
```

Below the editor is the 'Result Grid' section, which displays the query results in a table:

Company	Average_sales
Eget Ipsum Ltd	473.08

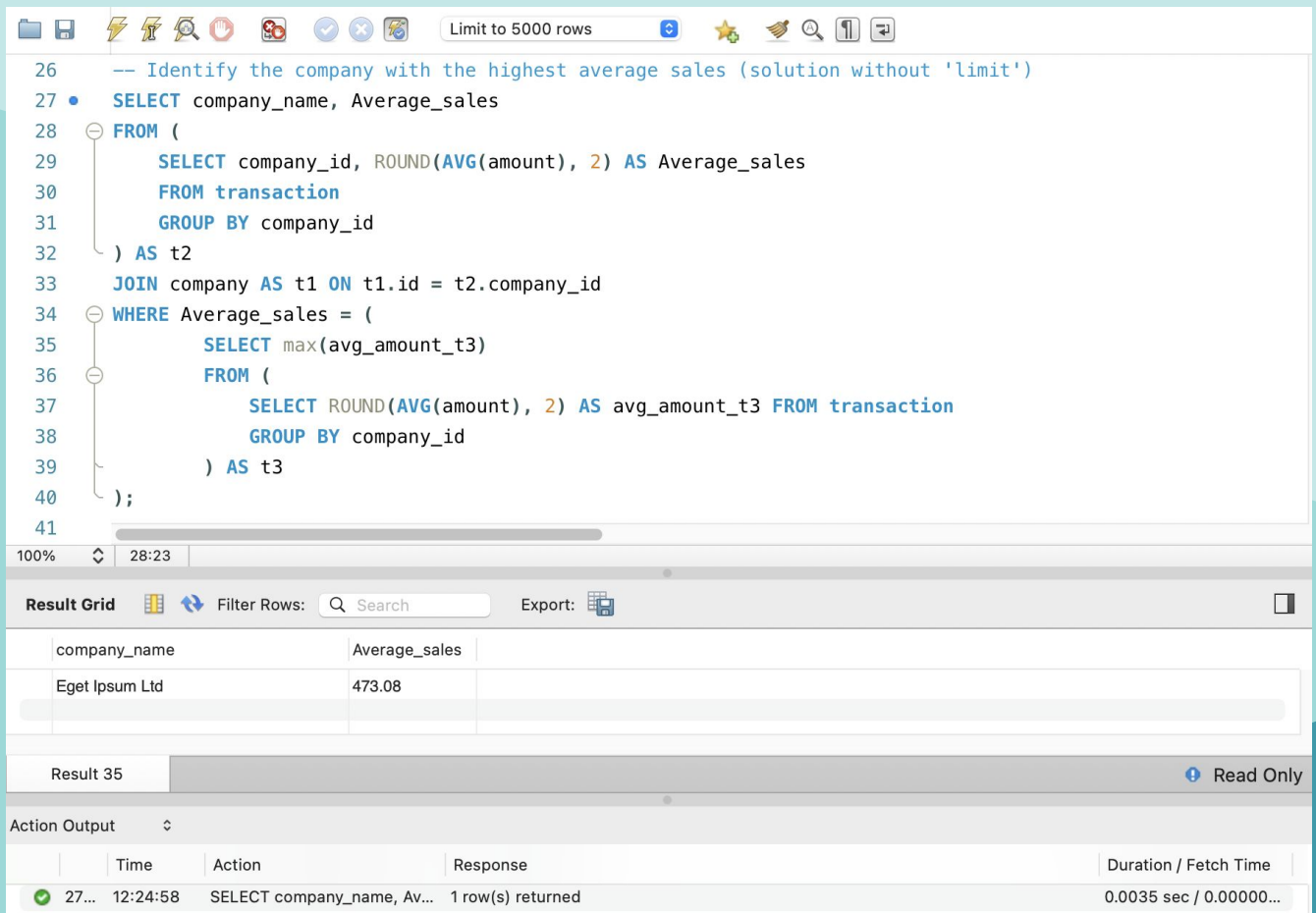
Below the result grid is the 'Action Output' section, which shows the execution details:

	Time	Action	Response	Duration / Fetch Time
✓ 27...	12:05:49	SELECT company_name 'Co...	1 row(s) returned	0.0060 sec / 0.0000...

Using LIMIT in SQL isn't always a good idea because:

1. Performance issues: With large datasets, the query often selects all rows, sorts them, and then trims the result to the specified limit. This can slow down execution for big tables.
2. Inaccurate results: LIMIT can hide important rows, especially without proper sorting, leading to random or non-representative samples.
3. Unpredictable behavior: Without an explicit ORDER BY clause, rows may be returned in a random order, producing unpredictable results.

While LIMIT is useful, it should be used carefully, considering these limitations. So I also implemented a solution without LIMIT:



The screenshot shows a SQL IDE interface with a query editor, a result grid, and an action output panel.

**Query Editor:** The query is as follows:

```
26 -- Identify the company with the highest average sales (solution without 'limit')
27 • SELECT company_name, Average_sales
28 FROM (
29     SELECT company_id, ROUND(AVG(amount), 2) AS Average_sales
30     FROM transaction
31     GROUP BY company_id
32 ) AS t2
33 JOIN company AS t1 ON t1.id = t2.company_id
34 WHERE Average_sales = (
35     SELECT max(avg_amount_t3)
36     FROM (
37         SELECT ROUND(AVG(amount), 2) AS avg_amount_t3 FROM transaction
38         GROUP BY company_id
39     ) AS t3
40 );
41
```

**Result Grid:** The result grid shows the following data:

company_name	Average_sales
Eget Ipsum Ltd	473.08

**Action Output:** The action output panel shows the following details:

Time	Action	Response	Duration / Fetch Time
12:24:58	SELECT company_name, Av...	1 row(s) returned	0.0035 sec / 0.00000...

## LEVEL 1 EXERCISE 3

Using only subqueries (without using JOIN):

- Show all transactions made by companies in Germany;
- List the companies that have made transactions for an amount higher than the average of all transactions;
- Companies that do not have registered transactions will be removed from the system, provide the list of these companies.

The screenshot shows a database IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 5000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area:

```
45 -- Show all transactions made by companies in Germany
46 • SELECT *
47 FROM transaction
48 WHERE company_id IN (
49     SELECT id FROM company
50     WHERE country = 'Germany'
51 );
```

Below the query editor, the 'Result Grid' is displayed, showing a table with 10 columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. The table contains 5 rows of data:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
108B1D1D-5B23-A76C-55...	CcU-2938	b-2222	275	83.7839	-178.86	2021-07-07 17:43:16	293.57	0
EA2C3281-C9C1-A387-44...	CcU-2938	b-2222	275	20.2004	-116.84	2021-05-09 10:25:08	119.36	1
0DD2E608-5C9E-D1B3-4...	CcU-2959	b-2234	275	9.68811	130.282	2021-04-17 05:30:17	252.47	1
AB069F53-965E-A2A8-CE...	CcU-2959	b-2234	275	1.64819	-158.007	2021-04-15 13:37:18	60.99	0
0466A42E-47CF-8D24-FD...	CcU-4219	b-2302	170	-43.9695	-117.525	2021-07-26 07:29:18	49.53	0

Below the result grid, there's a section for 'transaction 37' with an 'Apply' button. At the bottom, the 'Action Output' section shows a table with 5 columns: Time, Action, Response, and Duration / Fetch Time. It contains one row of data:

Time	Action	Response	Duration / Fetch Time
27... 12:27:51	SELECT * FROM transactio...	118 row(s) returned	0.0012 sec / 0.00005...

Limit to 5000 rows

```
53 -- List the companies that have made transactions for an amount higher than the average of all
54 -- transactions
55 • SELECT company_name 'Companies'
56 FROM company
57 WHERE id IN (
58     SELECT company_id
59     FROM transaction
60     WHERE amount > (
61         SELECT AVG(amount) AS comp_amount FROM transaction
62     )
63 ORDER BY company_name;
```

100% 53:60

**Result Grid** Filter Rows: Search Export:

Companies
A Institute
Ac Fermentum Incorporated
Ac Industries
Aliquam PC
Aliquam PC Limited

company 53 Read Only

**Action Output**

	Time	Action	Response	Duration / Fetch Time
✓ 27...	12:58:01	SELECT company_name 'Co...	70 row(s) returned	0.0027 sec / 0.00002...

Limit to 5000 rows

```
66 -- Companies that do not have registered transactions will be removed from the system, provide the
67 -- list of these companies
68 • SELECT company_name 'Companies'
69 FROM company
70 WHERE NOT EXISTS (
71     SELECT company_id FROM transaction
72 );
```

100% 30:79

**Result Grid** Filter Rows: Search Export:

Companies
-----------

company 62 Read Only

**Action Output**

	Time	Action	Response	Duration / Fetch Time
✓ 27...	13:07:13	SELECT company_name 'Co...	0 row(s) returned	0.00063 sec / 0.000...

## LEVEL 2 EXERCISE 1

Identify the five days that generated the largest amount of revenue for the company from sales. It shows the date of each transaction along with the sales total.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings, along with a 'Limit to 5000 rows' dropdown. The SQL editor contains the following query:

```
77 -- with 'limit'
78 • SELECT sum(amount) 'Sales total', DATE(timestamp) 'Date'
79 FROM transaction
80 WHERE declined = 0
81 GROUP BY DATE(timestamp)
82 ORDER BY sum(amount) DESC
83 LIMIT 5;
```

Below the editor is the 'Result Grid' section, which displays the query results in a table:

Sales total	Date
1532.36	2021-12-20
1397.96	2021-04-22
1344.37	2021-05-09
1337.62	2022-02-26
1325.12	2021-03-29

Below the result grid is the 'Action Output' section, which shows the execution details of the query:

Time	Action	Response	Duration / Fetch Time
27... 13:10:02	SELECT sum(amount) 'Sales...	5 row(s) returned	0.0026 sec / 0.00000...

The screenshot shows the DBeaver SQL editor interface. The top toolbar includes icons for file operations, navigation, and execution. A status bar at the top right indicates "Limit to 5000 rows".

```
-- the second implementation option with window function without 'limit'
86 • SELECT Sales_total, Sale_date
87 FROM (
88     SELECT sum(amount) AS Sales_total, DATE(timestamp) AS Sale_date,
89     ROW_NUMBER() OVER(ORDER BY sum(amount) DESC) AS ind_amount
90     FROM transaction
91     WHERE declined = 0
92     GROUP BY Sale_date
93 ) AS t
94 WHERE ind_amount <= 5
95 ORDER BY Sales_total;
```

The bottom panel displays the "Result Grid" with columns "Sales\_total" and "Sale\_date". It contains five rows of data:

Sales_total	Sale_date
1325.12	2021-03-29
1337.62	2022-02-26
1344.37	2021-05-09
1397.96	2021-04-22
1532.36	2021-12-20

Below the result grid, there is a section labeled "Result 70" with a "Read Only" indicator. At the very bottom, the "Action Output" panel shows the execution details for "SELECT Sales\_total, Sale\_d...":

	Time	Action	Response	Duration / Fetch Time
✓ 27 ...	13:18:30	SELECT Sales_total, Sale_d...	5 row(s) returned	0.0035 sec / 0.00000...



## LEVEL 2 EXERCISE 2

What is the average sales per country? It presents the results sorted from highest to lowest average.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings, along with a 'Limit to 5000 rows' dropdown. The SQL editor contains the following query:

```
98 -- What is the average sales per country? It presents the results sorted from highest to lowest average.
99 • SELECT country 'Countries', ROUND(AVG(amount), 2) AS Average_sales
100 FROM transactions.company AS t1
101 JOIN transactions.transaction AS t2
102 ON t1.id = t2.company_id
103 WHERE declined = 0
104 GROUP BY country
105 ORDER BY Average_sales DESC;
```

Below the editor is the 'Result Grid' section, which includes a 'Filter Rows' search bar and an 'Export' button. The results are displayed in a table:

Countries	Average_sales
United States	287.53
Ireland	285.83
Sweden	276.67
United Kingdom	271.77
Canada	261.94
Belgium	255.22

Below the table is the 'Action Output' section, which shows the execution details:

	Time	Action	Response	Duration / Fetch Time
28...	13:53:47	SELECT country 'Countries',...	15 row(s) returned	0.0045 sec / 0.00001...

## LEVEL 2 EXERCISE 3

In your company, a new project is being considered to launch some advertising campaigns to compete with the 'Non Institute' company. For this, they ask you for the list of all transactions carried out by companies that are located in the same country as this company.

- Display the list by applying JOIN and subqueries;
- Display the listing by applying only subqueries.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings, along with a 'Limit to 5000 rows' dropdown. The SQL editor contains the following query:

```
110 -- with join
111 • SELECT *
112 FROM transaction AS t1
113 JOIN company AS t2
114 ON t2.id = t1.company_id
115 WHERE country = (
116     SELECT country FROM company
117     WHERE company_name LIKE 'Non Institute'
118 )
119 AND company_name <> 'Non Institute'
120 ORDER BY company_name;
```

Below the editor is the 'Result Grid' section, which includes a search bar and an 'Export' button. It displays a table with 12 columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, declined, id, and company\_n. The table contains 7 rows of data.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined	id	company_n
2A5A3001-104F-1D1F-78...	CcU-3169	b-2354	272	-67.2525	-142.557	2022-02-27 18:35:15	30.76	0	b-2354	Ac Libero In
9679E769-32DC-2591-B8...	CcU-3169	b-2354	272	47.6643	130.202	2021-06-28 11:22:10	186.34	1	b-2354	Ac Libero In
6D69D98A-F18A-99BD-B...	CcU-3204	b-2374	272	-52.3724	70.1522	2021-05-26 02:33:06	144.33	1	b-2374	Amet Faucit
E5078B1B-9591-E204-CC...	CcU-3204	b-2374	272	69.5434	98.8783	2021-09-06 01:29:42	220.85	0	b-2374	Amet Faucit
1479B3D2-B7BA-C7BB-4...	CcU-2994	b-2326	133	66.2672	172.399	2021-08-09 00:58:07	309.45	0	b-2326	Enim Condi
152598C2-029D-D684-4B...	CcU-2994	b-2326	126	-67.0189	-141.672	2021-07-05 03:10:00	395.43	0	b-2326	Enim Condi

Below the result grid is the 'Action Output' section, which shows a table with 5 columns: Time, Action, Response, and Duration / Fetch Time. It contains one row of data:

Time	Action	Response	Duration / Fetch Time
27... 13:28:08	SELECT * FROM transaction...	70 row(s) returned	0.0021 sec / 0.00005...

The screenshot displays a SQL IDE interface. The top section contains a SQL query with line numbers 122 to 132. The query is a SELECT statement with a subquery in the WHERE clause. The subquery selects the id from the company table where the country is the same as the company in the main query, and the company name is not 'Non Institute'. The main query selects all columns from the transaction table where the company\_id is in the subquery's results.

```
122 -- with only subqueries
123 • SELECT *
124 FROM transaction
125 WHERE company_id IN (
126     SELECT id FROM company
127     WHERE country = (
128         SELECT country FROM company
129         WHERE company_name LIKE 'Non Institute'
130     )
131     AND company_name <> 'Non Institute'
132 );
```

Below the query editor is the 'Result Grid' section, which shows a table with 10 columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. The table contains three rows of data.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
2B928E1C-EC14-A760-0A...	CcU-2980	b-2246	275	-41.0496	161.685	2021-08-10 08:14:49	383.73	0
ACD2011A-A2B1-C365-41...	CcU-2980	b-2246	275	-54.4792	-82.7974	2022-03-05 20:41:20	60.07	1
4334349E-CEB0-3D68-A4...	CcU-3092	b-2310	275	-20.4859	150.87	2021-05-03 22:37:23	458.74	0

Below the result grid is the 'Action Output' section, which shows a table with 5 columns: Time, Action, Response, and Duration / Fetch Time. The table contains one row of data.

Time	Action	Response	Duration / Fetch Time
13:39:10	SELECT * FROM transaction...	70 row(s) returned	0.0014 sec / 0.00007...

## LEVEL 3 EXERCISE 1

It presents the name, telephone, country, date and amount of those companies that made transactions with a value between 100 and 200 euros and on any of these dates: April 29, 2021, July 20, 2021 and March 13, 2022. Sort the results from highest to lowest amount.

The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a 'Limit to 5000 rows' dropdown. Below the toolbar, a SQL query is entered in a text area. The query is as follows:

```
-- It presents the name, telephone, country, date and amount of those companies that made transactions
with a value between 100 and 200 euros and on any of these dates: April 29, 2021, July 20, 2021 and
March 13, 2022. Sort the results from highest to lowest amount.
SELECT company_name, phone, country, DATE(timestamp) 'date', amount
FROM company AS t1
JOIN transaction AS t2
ON t1.id = t2.company_id
WHERE DATE(timestamp) IN ('2021-04-29', '2021-07-20', '2022-03-13')
AND amount BETWEEN 100 AND 200
ORDER BY amount DESC;
```

Below the query editor, there's a 'Result Grid' section. It shows a table with 5 columns: company\_name, phone, country, date, and amount. The table contains 5 rows of data:

company_name	phone	country	date	amount
Interdum Feugiat Sed Associates	04 88 40 32 52	United Kingdom	2021-07-20	164.86
Nunc Interdum Incorporated	05 18 15 48 13	Germany	2022-03-13	164.32
Enim Condimentum Ltd	09 55 51 66 25	United Kingdom	2021-04-29	149.89
Lorem Eu Incorporated	01 83 66 62 07	Canada	2021-07-20	133.39
Nunc Interdum Incorporated	05 18 15 48 13	Germany	2021-04-29	111.51

Below the result grid, there's an 'Action Output' section. It shows a table with 4 columns: Time, Action, Response, and Duration / Fetch Time. The table contains 1 row of data:

Time	Action	Response	Duration / Fetch Time
13:43:01	SELECT company_name, ph...	5 row(s) returned	0.0019 sec / 0.00001...

## LEVEL 3 EXERCISE 2

We need to optimize the allocation of resources and it will depend on the operational capacity that is required, so they ask you for the information about the amount of transactions that the companies carry out, but the HR department is demanding and wants a list of the companies where you specify if they have more than 4 transactions or less.

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 5000 rows' dropdown. The SQL editor contains a query with a comment explaining the task: to optimize resource allocation by identifying companies with more than 4 transactions. The query uses a CASE statement to categorize companies based on their transaction count. The results are displayed in a 'Result Grid' with columns for company\_name and transaction\_count. The bottom section shows the 'Action Output' with a successful execution log.

```
145 -- We need to optimize the allocation of resources and it will depend on the operational capacity that
    is required, so they ask you for the information about the amount of transactions that the companies
    carry out, but the HR department is demanding and wants a list of the companies where you specify if
    they have more than 4 transactions or less.
146 • SELECT company_name,
147     CASE
148         WHEN count(t2.id) >= 4 THEN '>= 4 transactions'
149         ELSE '< 4 transactions'
150     END AS transaction_count
151 FROM transactions.company AS t1
152 JOIN transactions.transaction AS t2
153 ON t1.id = t2.company_id
154 GROUP BY t1.id
155 ORDER BY transaction_count DESC;
```

company_name	transaction_count
Non Institute	>= 4 transactions
Malesuada PC	>= 4 transactions
Lorem Eu Incorporated	>= 4 transactions
Ut Sempur Foundation	>= 4 transactions
Enim Condimentum Ltd	>= 4 transactions
Nunc Interdum Incorporated	>= 4 transactions
Arcu LLP	>= 4 transactions
Amet Nulla Donec Corporation	< 4 transactions
Necetur Biddulus Mus Inc	< 4 transactions

Result 91 Read Only

	Time	Action	Response	Duration / Fetch Time
28...	13:47:51	SELECT company_name, C...	100 row(s) returned	0.0028 sec / 0.00003...