# SPRINT #2

## RELATIONAL DATABASES AND INTRODUCTION TO SQL

student:
Leo Kalugin

Date: 03/10/2024

## SUMMARY

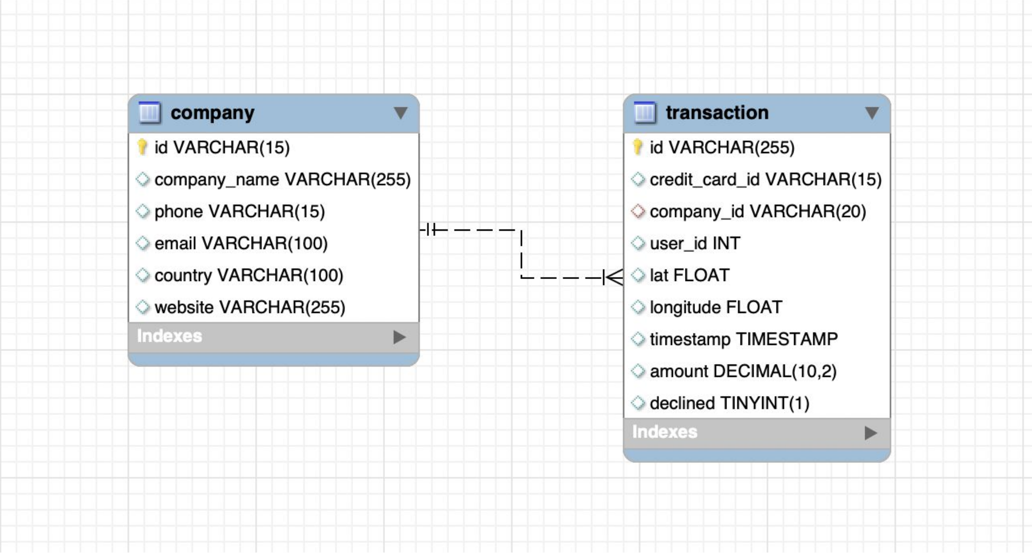
In this sprint, I will review the basic concepts of working with relational databases. I will begin hands-on experience with a database containing information from a company dedicated to selling products online. In this activity, I will focus on data related to completed transactions and the corporate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

## LEVEL 1 EXERCISE 1

From the attached documents ('data_structure.sql' and 'db_data.sql'), import the two tables. Show the main features of the diagram you have created and explain all the tables and variables that exist. Be sure to include a diagram that illustrates the relationships between the various tables and variables.



This database consists of two tables: ‘company’ and ‘transaction’.

**1.** ‘company’ contains information about companies involved in the transactions, each company is uniquely identified by an ‘id’:

• id (varchar(15)): primary key, uniquely identifies each company;
• company_name (varchar(255)): name of the company;
• phone (varchar(15)): contact phone number of the company;
• email (varchar(100)): email address of the company;
• country (varchar(100)): country where the company is based;
• website (varchar(255)): website URL of the company.

This table provides 6 variables, representing company-specific information. To store all variables, a variable-length string data type is used with length of 15, 100 and 255 characters for storage depending on the need. The ‘id’ field links the company table with the transaction table via a foreign key, establishing a one-to-many relationship (‘one’ for the ‘company’ table, ‘many’ for the ‘transaction’ table).

**2.** ‘transaction’ records individual transactions, with each transaction linked to a company:

• id (varchar(255)): primary key, uniquely identifies each transaction;
• credit_card (varchar(15)): the credit card number used for the transaction;
• company_id (varchar(20)): foreign key referencing the ‘id’ field in the ‘company’ table, linking the transaction to the relevant company;
• user_id (int): id of the user who made the transaction;
• lat (float): latitude where the transaction occurred;
• longitude (float): longitude where the transaction occurred;
• timestamp (timestamp): date and time when the transaction occurred;
• amount (decimal(10,2)): the amount of money involved in the transaction;
• decline (tinyint(1)): a flag indicating whether the transaction was declined (1 = declined, 0 = successful).

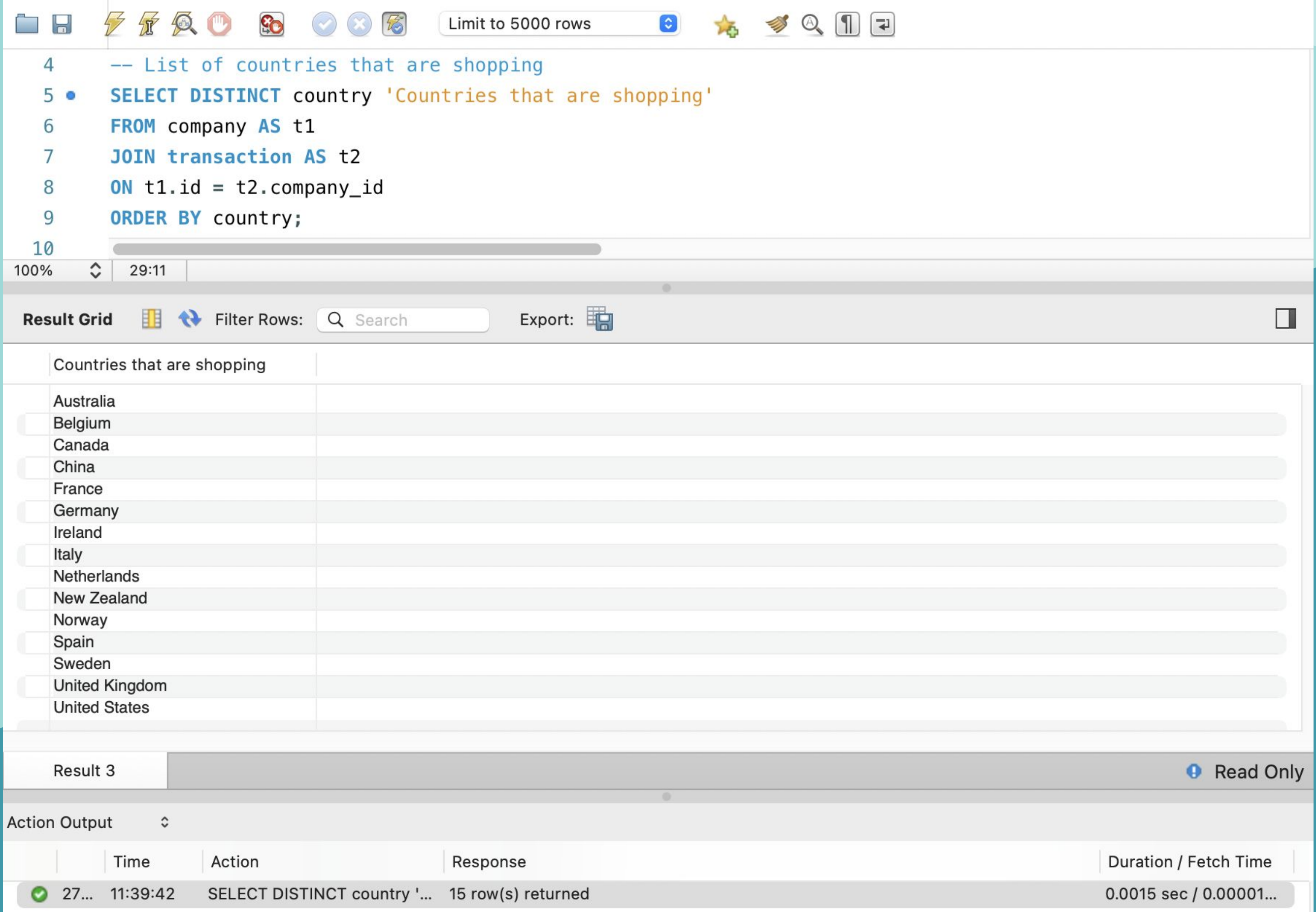
The data types used here also include variable-length strings, integer and float for storing numbers, timestamp for an exact timestamp, decimal(10,2) with up to 10 digits, including 2 decimal places, and tinyint is binary data type is efficient for storing true/false values.

The ‘company’ table is linked to the transaction table through the ‘company_id’ foreign key. This defines a one-to-many relationship: one company can have multiple transactions, but each transaction is associated with only one company. The primary key in both tables (‘id’ in ‘company’ and ‘id’ in ‘transaction’) ensures uniqueness for records in each respective table.

## LEVEL 1 EXERCISE 2

Using JOIN you will perform the following queries:

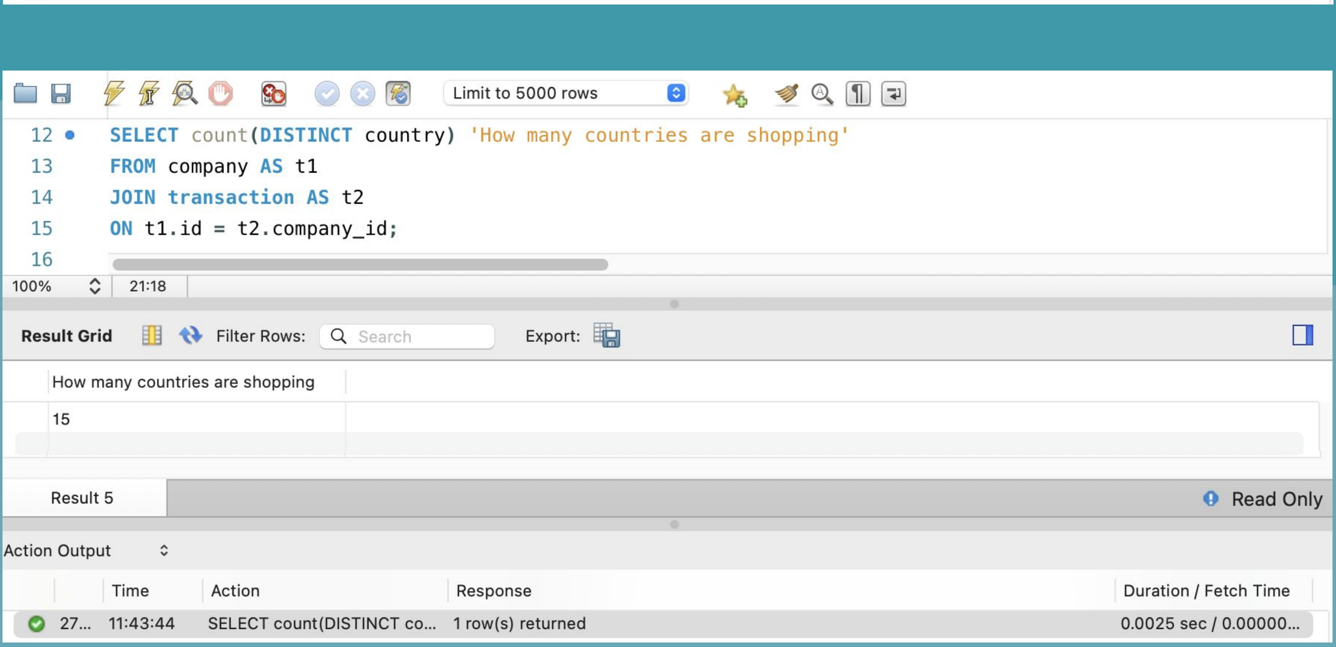
- List of countries that are shopping;
- From how many countries the purchases are made;
- Identify the company with the highest average sales.



```sql
-- List of countries that are shopping
SELECT DISTINCT country 'Countries that are shopping'
FROM company AS t1
JOIN transaction AS t2
ON t1.id = t2.company_id
ORDER BY country;
```

| Countries that are shopping |
|---|
| Australia |
| Belgium |
| Canada |
| China |
| France |
| Germany |
| Ireland |
| Italy |
| Netherlands |
| New Zealand |
| Norway |
| Spain |
| Sweden |
| United Kingdom |
| United States |

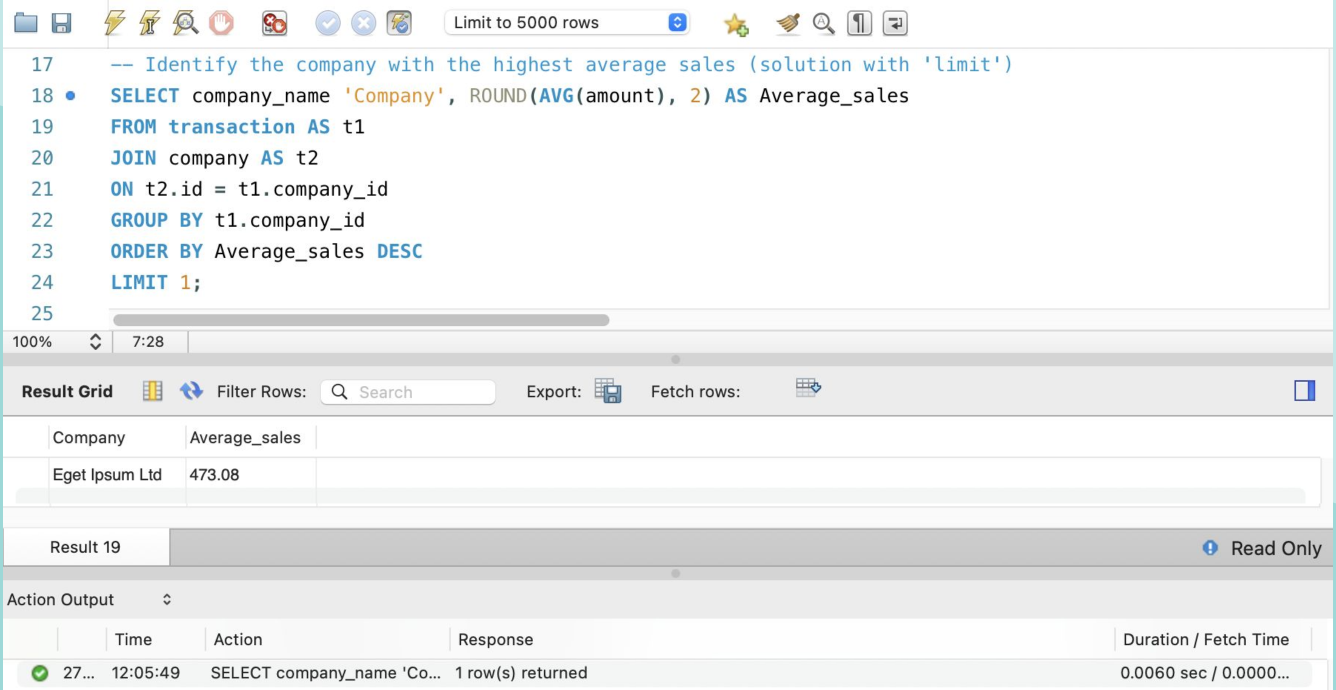
| | Time | Action | Response | Duration / Fetch Time |
|---|---|---|---|---|
| 27... | 11:39:42 | SELECT DISTINCT country '... | 15 row(s) returned | 0.0015 sec / 0.00001... |



```sql
SELECT count(DISTINCT country) 'How many countries are shopping'
FROM company AS t1
JOIN transaction AS t2
ON t1.id = t2.company_id;
```

| How many countries are shopping |
|---|
| 15 |

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



```sql
-- Identify the company with the highest average sales (solution with 'limit')
SELECT company_name 'Company', ROUND(AVG(amount), 2) AS Average_sales
FROM transaction AS t1
JOIN company AS t2
ON t2.id = t1.company_id
GROUP BY t1.company_id
ORDER BY Average_sales DESC
LIMIT 1;
```

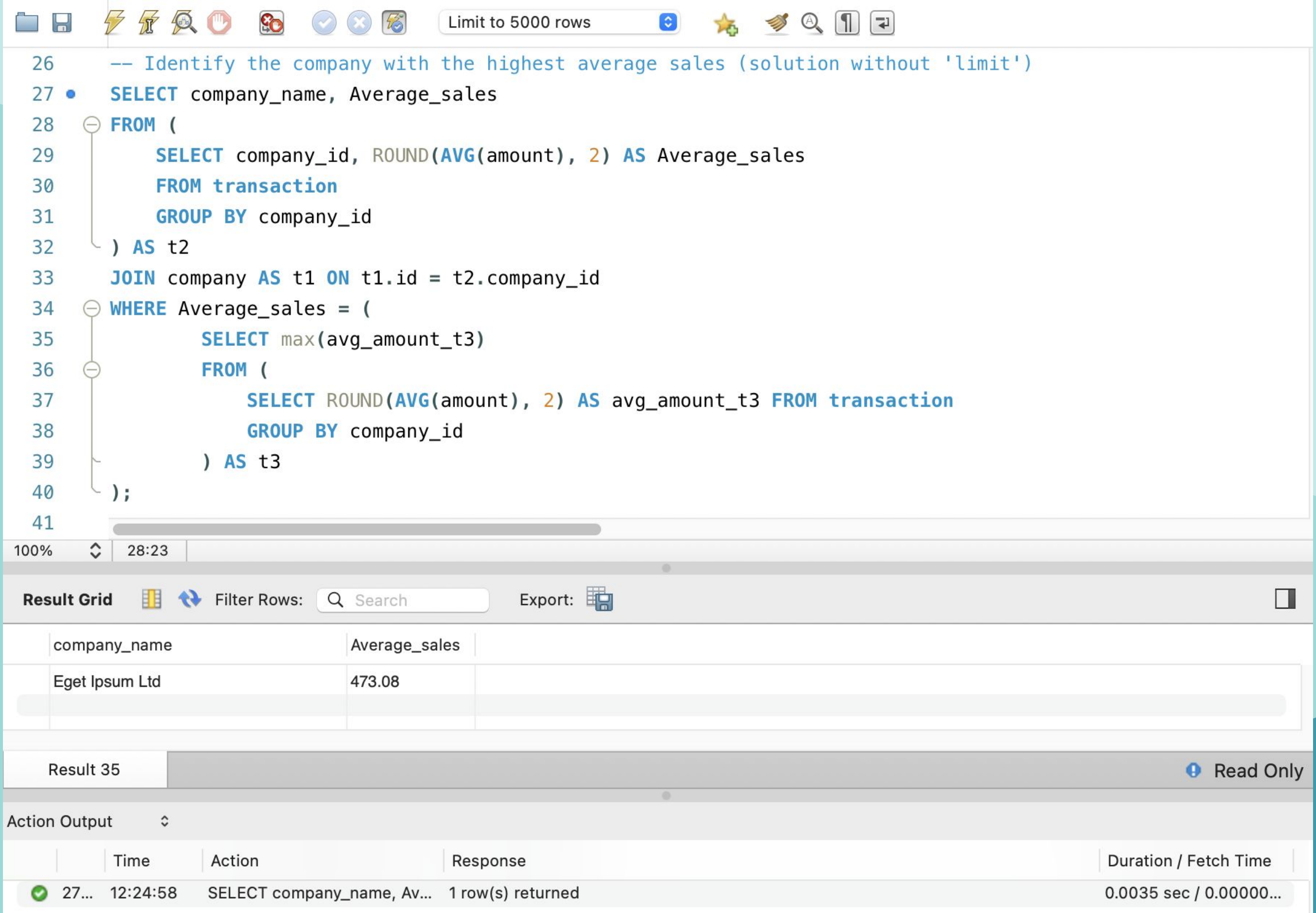
| Company | Average_sales |
|---|---|
| Eget Ipsum Ltd | 473.08 |

```
     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:



```
-- Identify the company with the highest average sales (solution without 'limit')
SELECT company_name, Average_sales
FROM (
    SELECT company_id, ROUND(AVG(amount), 2) AS Average_sales
    FROM transaction
    GROUP BY company_id
) AS t2
JOIN company AS t1 ON t1.id = t2.company_id
WHERE Average_sales = (
        SELECT max(avg_amount_t3)
        FROM (
            SELECT ROUND(AVG(amount), 2) AS avg_amount_t3 FROM transaction
            GROUP BY company_id
        ) AS t3
);
```

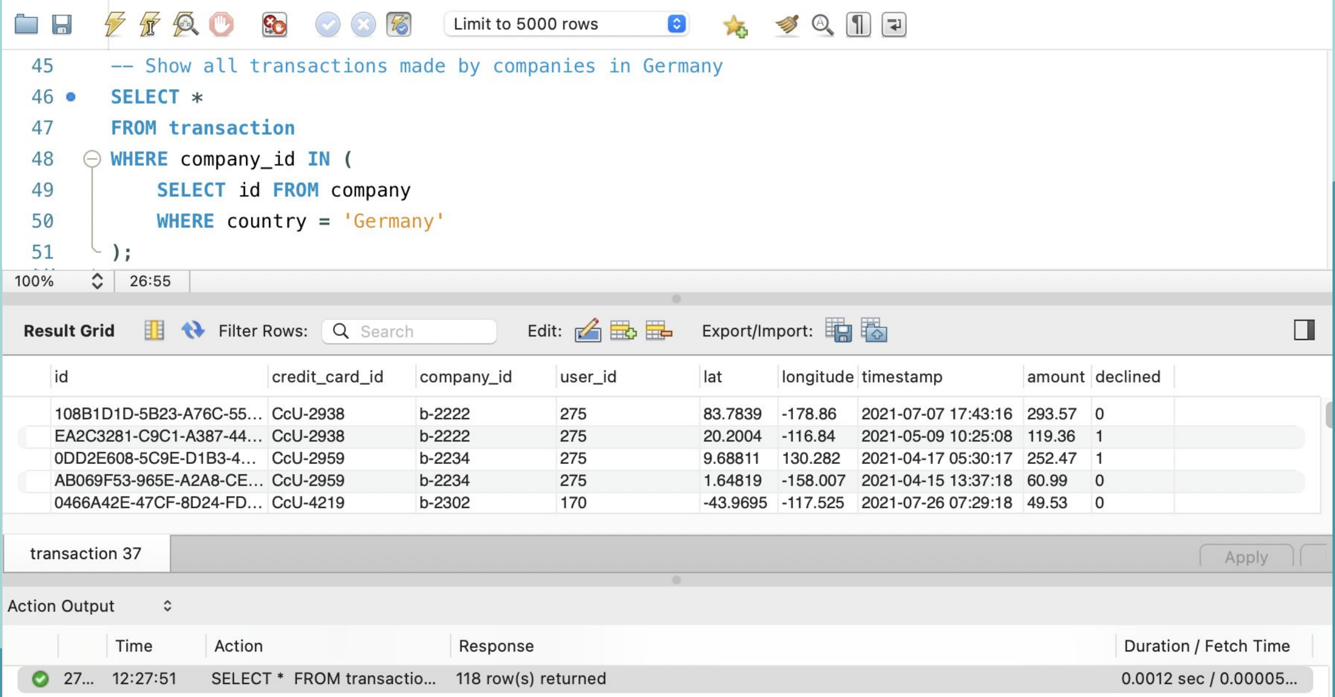
| company_name | Average_sales |
|---|---|
| Eget Ipsum Ltd | 473.08 |

```
    Time      Action                       Response            Duration / Fetch Time
27... 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):

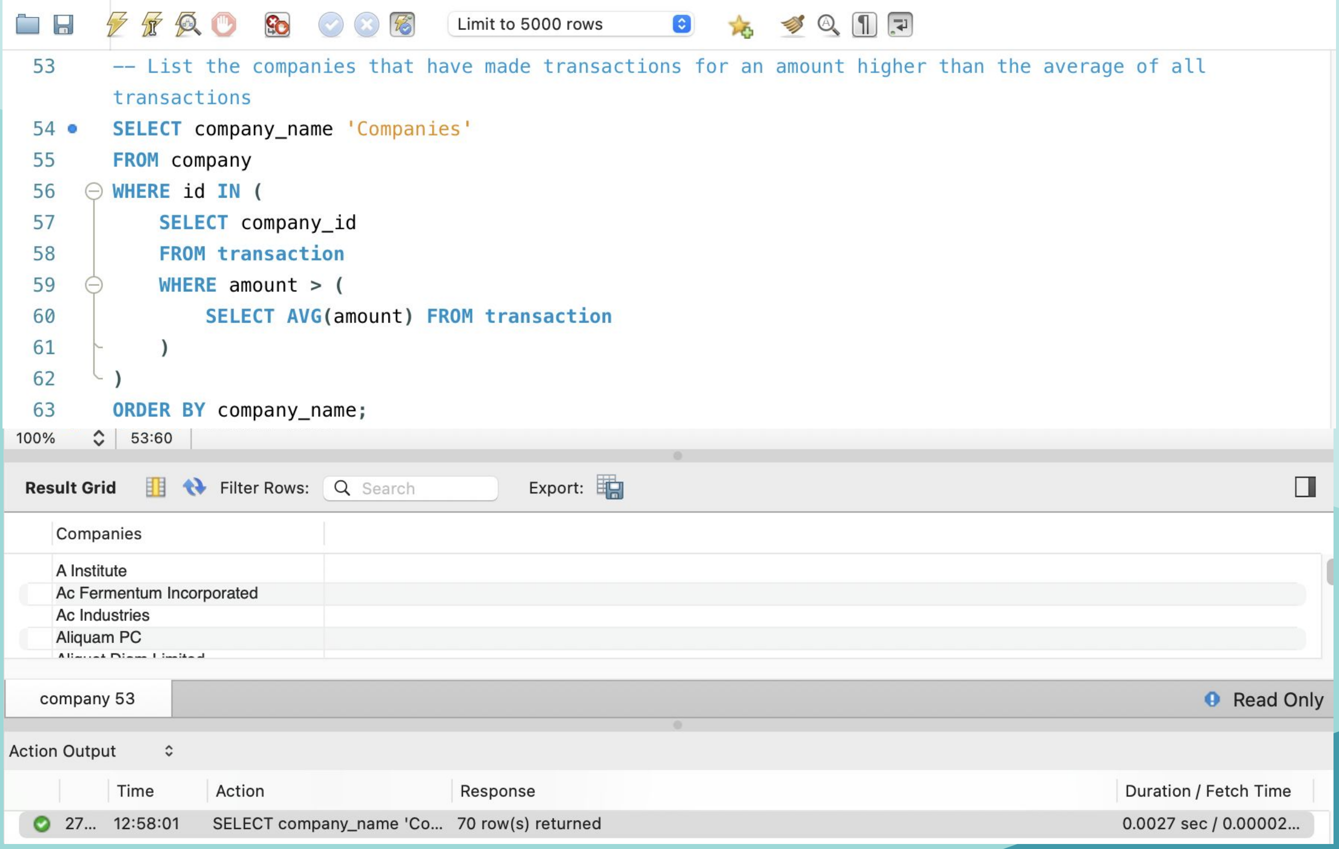
- ○ Show all transactions made by companies in Germany;
- ○ List the companies that have made transactions for an amount higher than the average of all transactions;
- ○ Companies that do not have registered transactions will be removed from the system, provide the list of these companies.



```sql
-- Show all transactions made by companies in Germany
SELECT *
FROM transaction
WHERE company_id IN (
    SELECT id FROM company
    WHERE country = 'Germany'
);
```

| 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 |

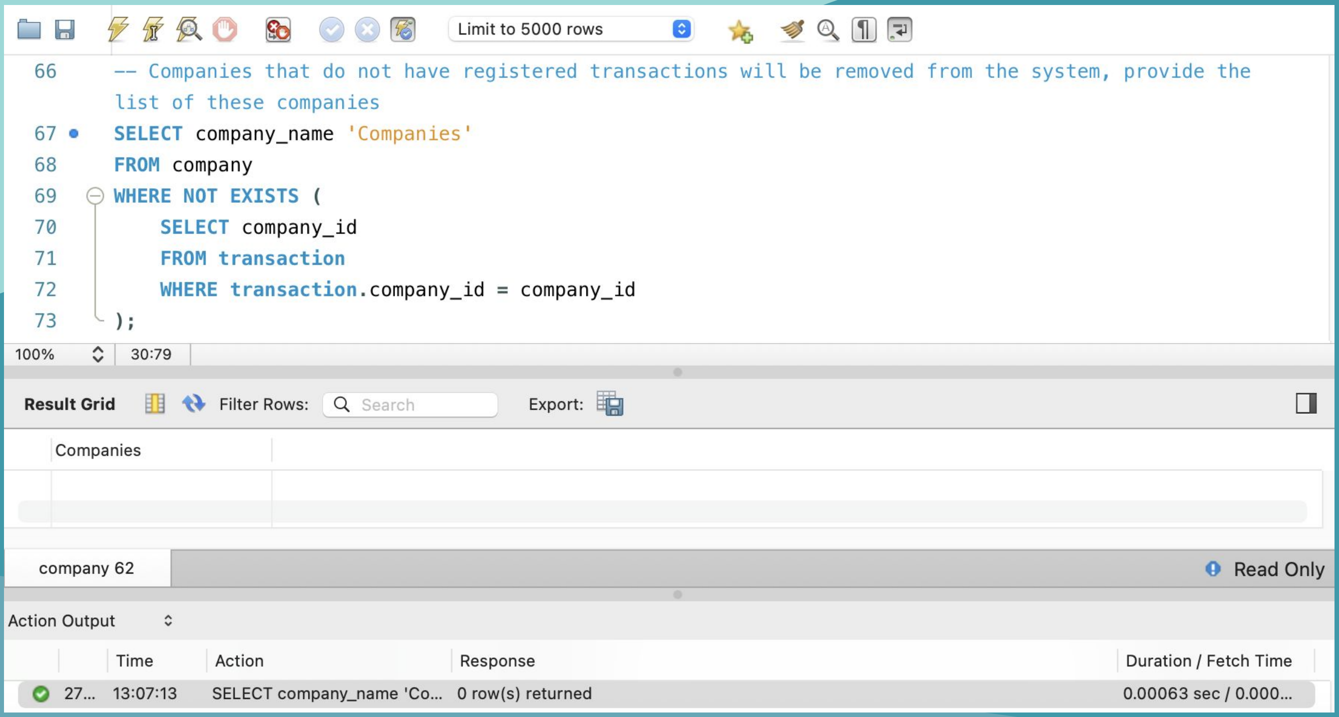
| | | Time | Action | Response | Duration / Fetch Time |
|---|---|---|---|---|---|
| ✓ | 27... | 12:27:51 | SELECT * FROM transactio... | 118 row(s) returned | 0.0012 sec / 0.00005... |



```sql
-- List the companies that have made transactions for an amount higher than the average of all transactions
SELECT company_name 'Companies'
FROM company
WHERE id IN (
    SELECT company_id
    FROM transaction
    WHERE amount > (
        SELECT AVG(amount) FROM transaction
    )
)
ORDER BY company_name;
```

| Companies |
|---|
| A Institute |
| Ac Fermentum Incorporated |
| Ac Industries |
| Aliquam PC |
| Aliquet Diam Limited |

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



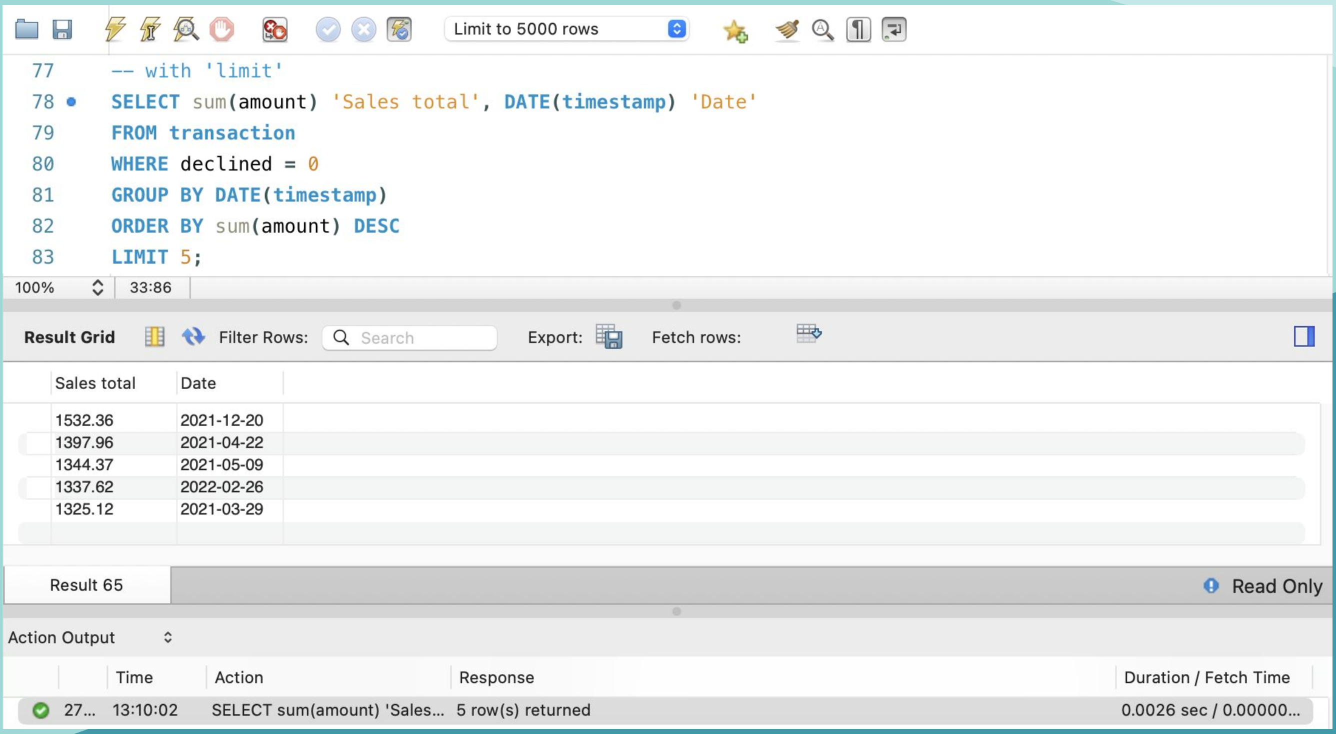
```sql
-- Companies that do not have registered transactions will be removed from the system, provide the list of these companies
SELECT company_name 'Companies'
FROM company
WHERE NOT EXISTS (
    SELECT company_id
    FROM transaction
    WHERE transaction.company_id = company_id
);
```

| Companies |
|---|

```
    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

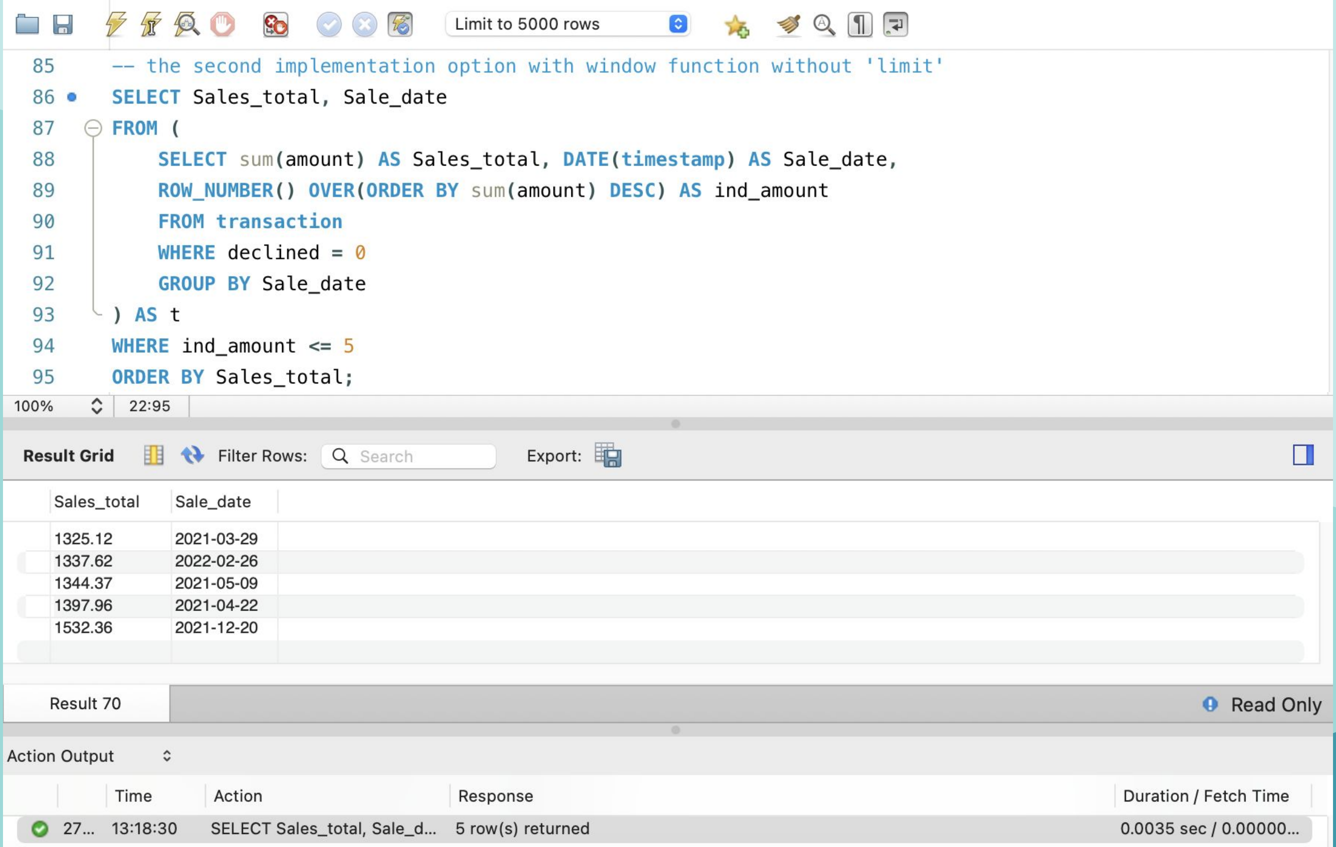
Identify the five days that generated the largest amount of revenue for the company from sales. It shows the date of each transaction along with the sales total.



```sql
-- with 'limit'
SELECT sum(amount) 'Sales total', DATE(timestamp) 'Date'
FROM transaction
WHERE declined = 0
GROUP BY DATE(timestamp)
ORDER BY sum(amount) DESC
LIMIT 5;
```

| 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 |

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



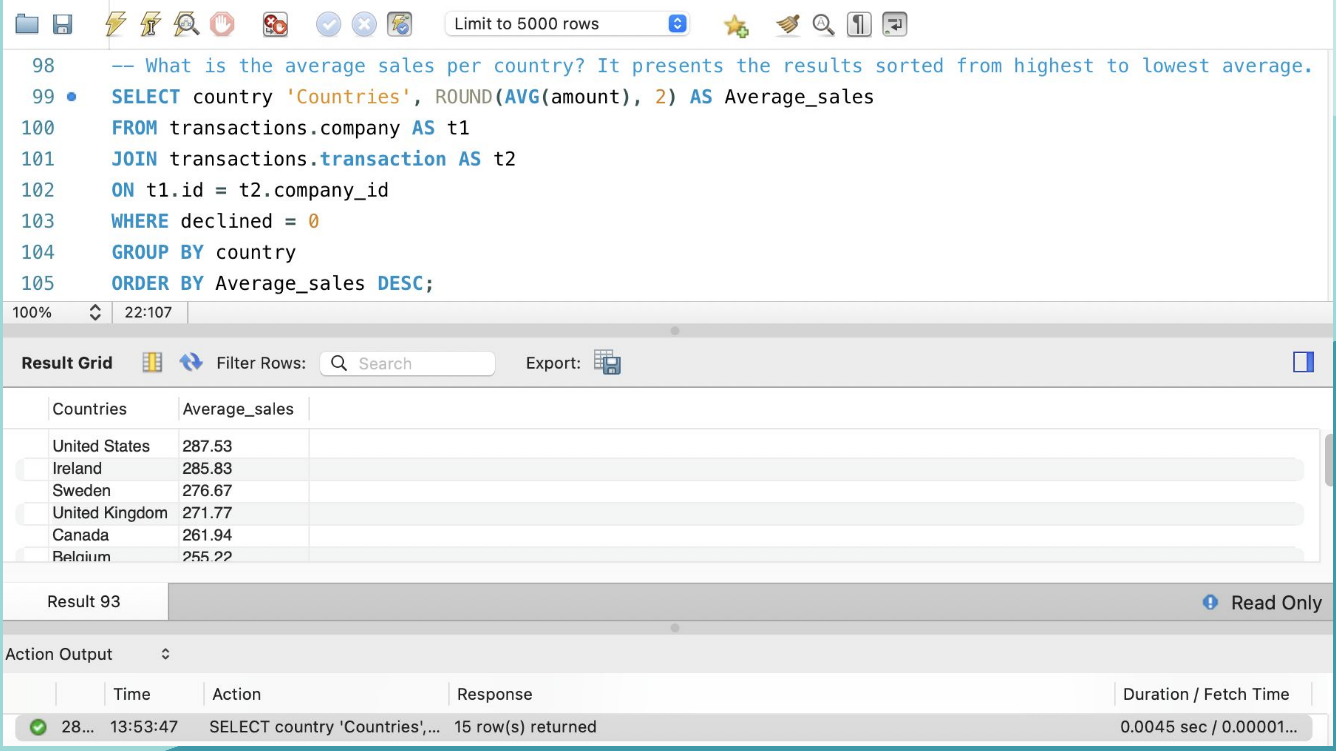
```
-- the second implementation option with window function without 'limit'
SELECT Sales_total, Sale_date
FROM (
    SELECT sum(amount) AS Sales_total, DATE(timestamp) AS Sale_date,
    ROW_NUMBER() OVER(ORDER BY sum(amount) DESC) AS ind_amount
    FROM transaction
    WHERE declined = 0
    GROUP BY Sale_date
) AS t
WHERE ind_amount <= 5
ORDER BY Sales_total;
```

| 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 |

| | 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.



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

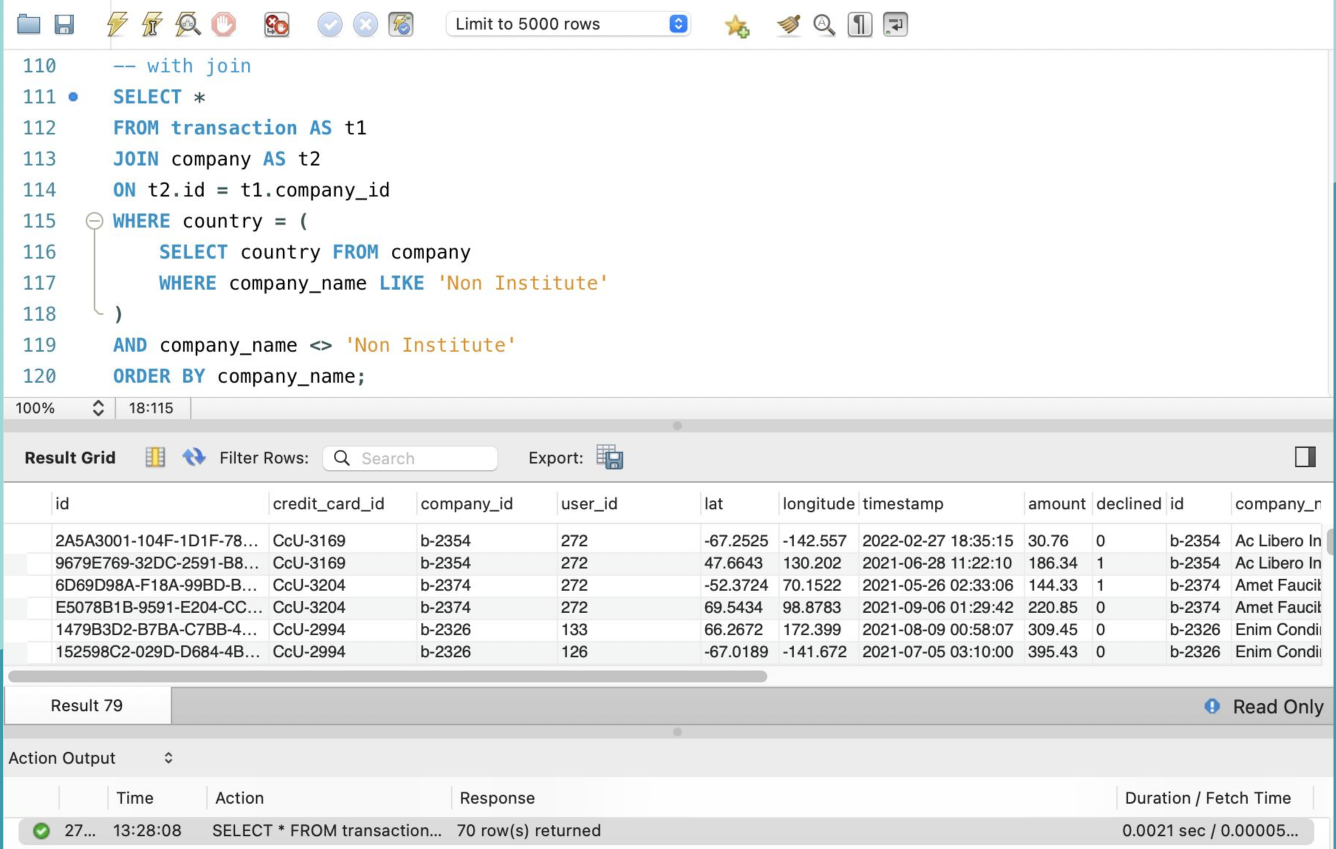
| Countries | Average_sales |
|---|---|
| United States | 287.53 |
| Ireland | 285.83 |
| Sweden | 276.67 |
| United Kingdom | 271.77 |
| Canada | 261.94 |
| Belgium | 255.22 |

| | 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.

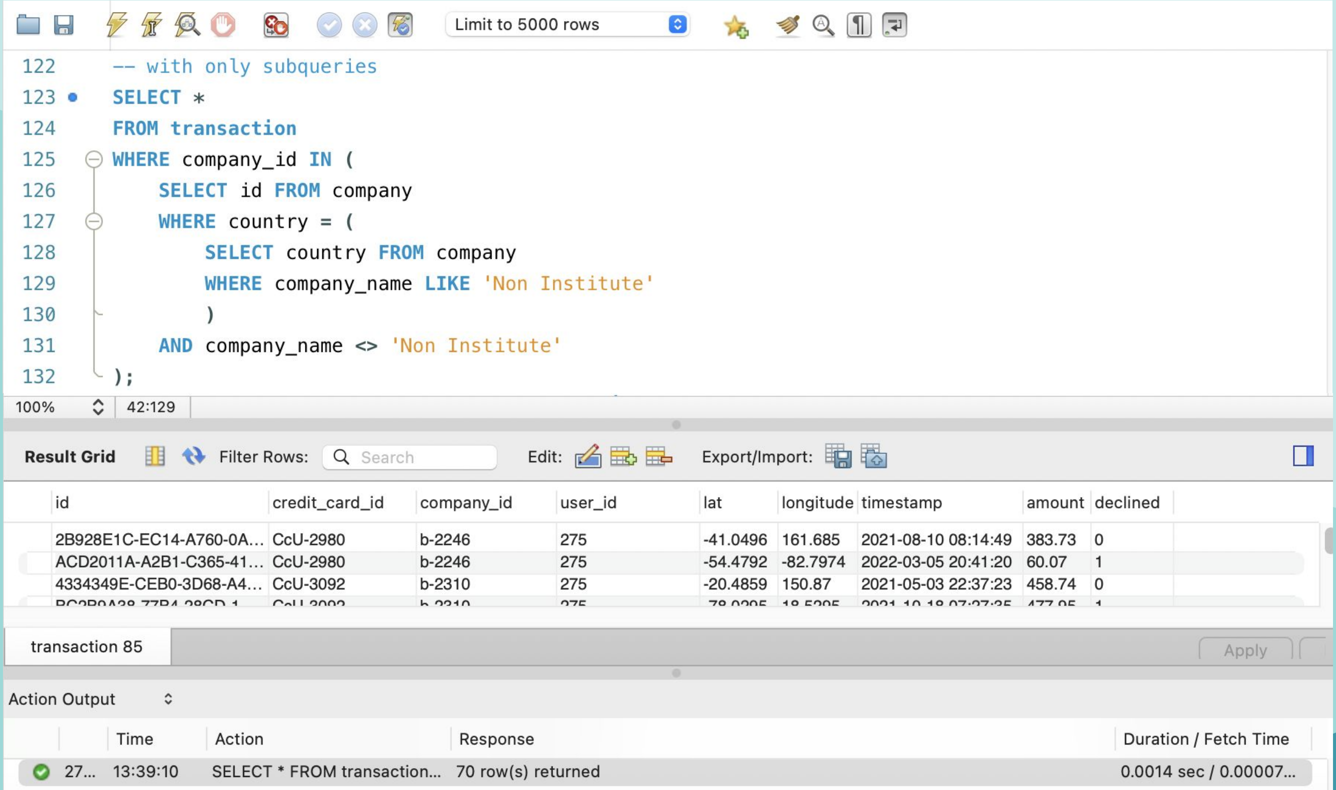
- Display the list by applying JOIN and subqueries;
- Display the listing by applying only subqueries.



```sql
-- with join
SELECT *
FROM transaction AS t1
JOIN company AS t2
ON t2.id = t1.company_id
WHERE country = (
    SELECT country FROM company
    WHERE company_name LIKE 'Non Institute'
)
AND company_name <> 'Non Institute'
ORDER BY company_name;
```

| id | credit_card_id | company_id | user_id | lat | longitude | timestamp | amount | declined | id | company_r |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

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



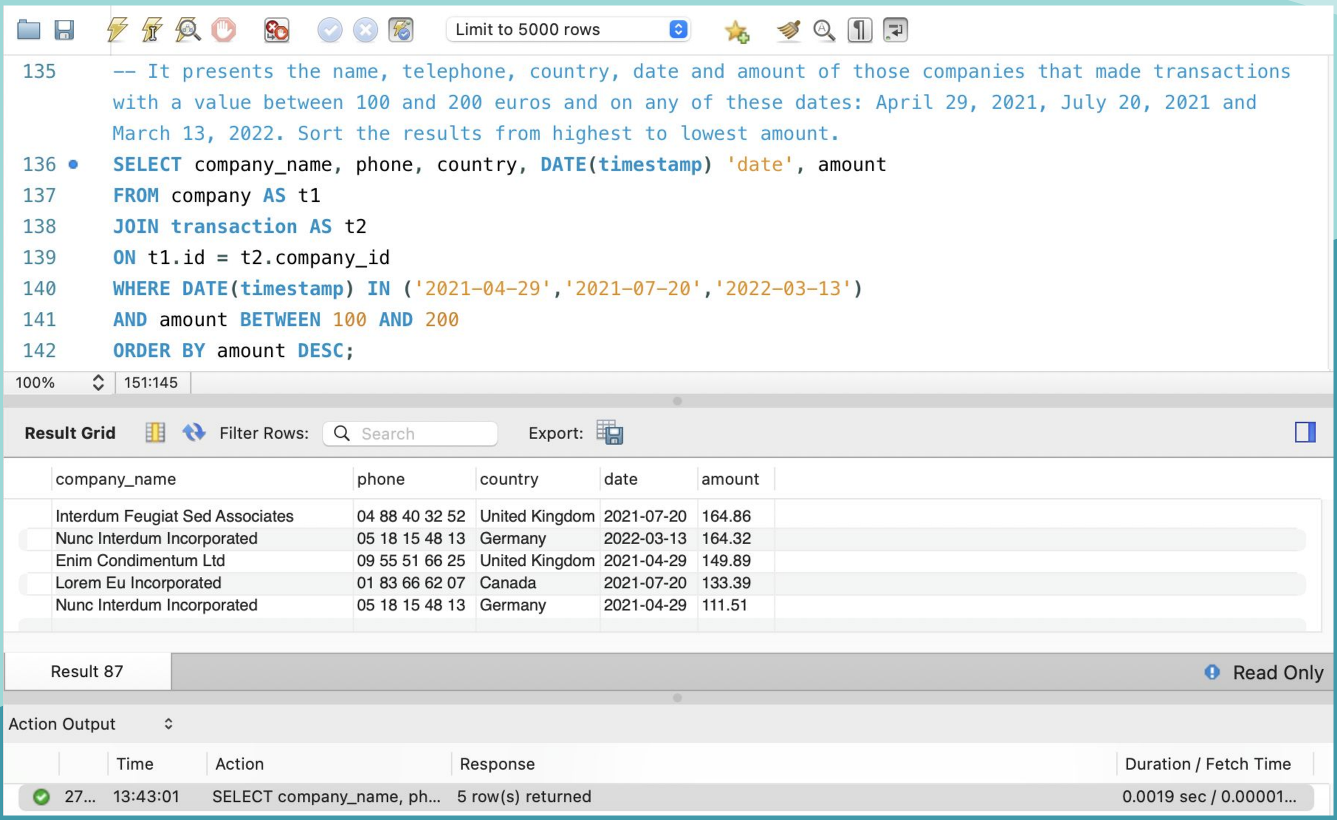
```sql
-- with only subqueries
SELECT *
FROM transaction
WHERE company_id IN (
    SELECT id FROM company
    WHERE country = (
        SELECT country FROM company
        WHERE company_name LIKE 'Non Institute'
        )
    AND company_name <> 'Non Institute'
);
```

| 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 |
| BC2B9A38-77B4-28CD-1 | CcU-3092 | b-2310 | 275 | -78.0295 | 18.5295 | 2021-10-18 07:27:35 | 477.05 | 1 |

| | Time | Action | Response | Duration / Fetch Time |
|---|---|---|---|---|
| 27... | 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.



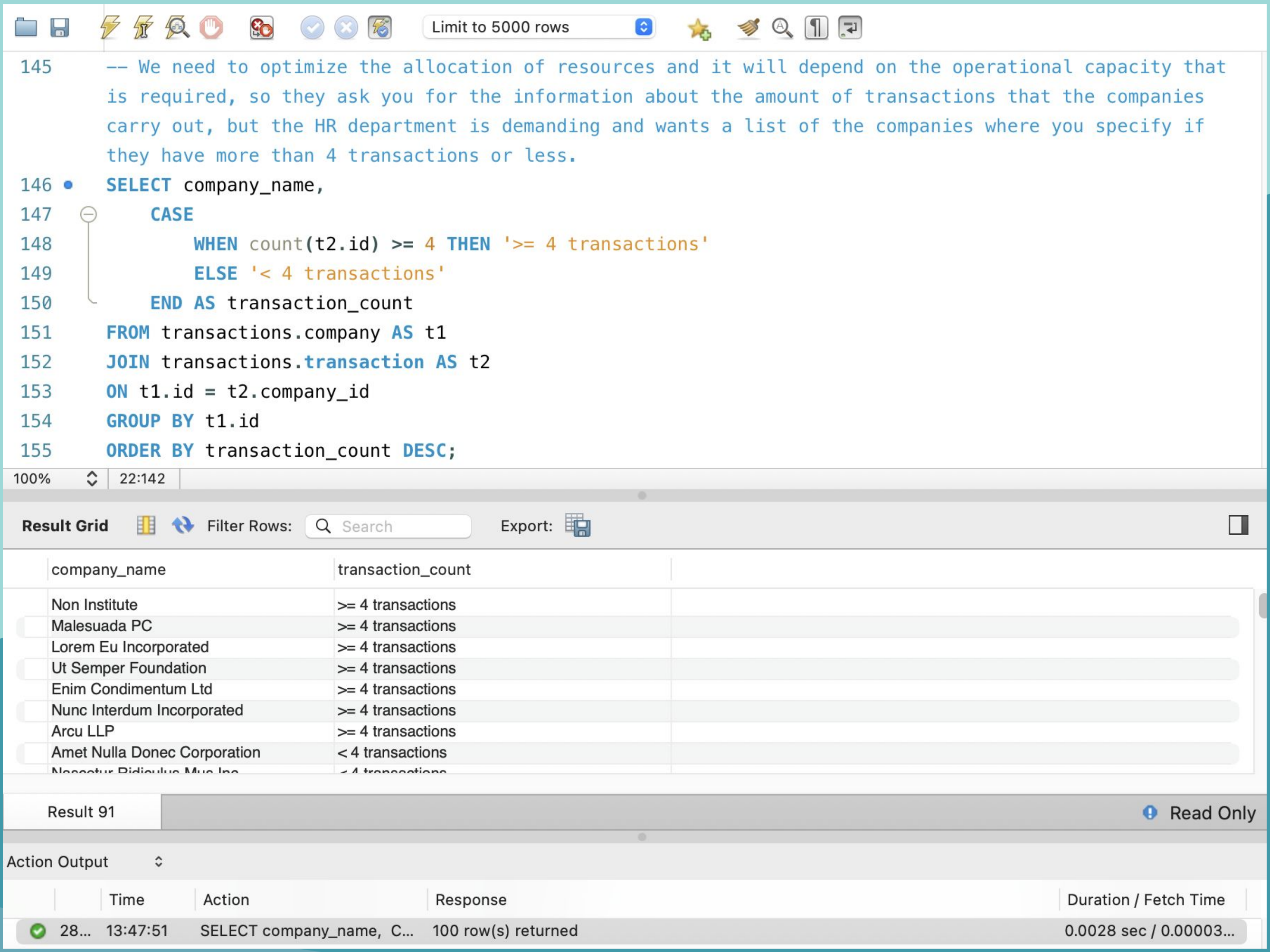
```sql
-- It presents the name, telephone, country, date and amount of those companies that made transactions
with a value between 100 and 200 euros and on any of these dates: April 29, 2021, July 20, 2021 and
March 13, 2022. Sort the results from highest to lowest amount.
SELECT company_name, phone,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

| 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 |

| | Time | Action | Response | Duration / Fetch Time |
|---|---|---|---|---|
| 27... | 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.



Limit to 5000 rows

```sql
-- 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.
SELECT company_name,
    CASE
        WHEN count(t2.id) >= 4 THEN '>= 4 transactions'
        ELSE '< 4 transactions'
    END AS transaction_count
FROM transactions.company AS t1
JOIN transactions.transaction AS t2
ON t1.id = t2.company_id
GROUP BY t1.id
ORDER BY transaction_count DESC;
```

| company_name | transaction_count |
|---|---|
| Non Institute | >= 4 transactions |
| Malesuada PC | >= 4 transactions |
| Lorem Eu Incorporated | >= 4 transactions |
| Ut Semper Foundation | >= 4 transactions |
| Enim Condimentum Ltd | >= 4 transactions |
| Nunc Interdum Incorporated | >= 4 transactions |
| Arcu LLP | >= 4 transactions |
| Amet Nulla Donec Corporation | < 4 transactions |
| Nascetur Ridiculus Mus Inc | < 4 transactions |

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