

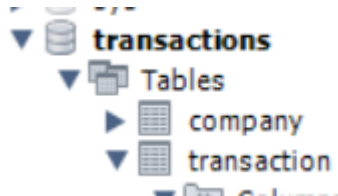
Autora: Giorgia Calvagna

Fecha de envío: 16/12/2024

Revisado por: Mailin Adriana Villán

## Tasca S2.01. Nocions bàsiques SQL

Para aprender más sobre cómo utilizar bases de datos relacionales, en este sprint 2 vamos a utilizar la base de datos *transaction* que está compuesta por las tablas *company* y *transaction*:



Para poder cargar la base de datos *transaction* fue necesario cargar los dos archivos sql *estructura\_dades* y *dades\_introduir*.

Aquí hay una visión general del archivo sql *estructura\_dades*:

```
1  -- Creamos la base de datos
2  CREATE DATABASE IF NOT EXISTS transactions;
3  USE transactions;
4
5  -- Creamos la tabla company
6  CREATE TABLE IF NOT EXISTS company (
7      id VARCHAR(15) PRIMARY KEY,
8      company_name VARCHAR(255),
9      phone VARCHAR(15),
10     email VARCHAR(100),
11     country VARCHAR(100),
12     website VARCHAR(255)
13 );
14
15
16 -- Creamos la tabla transaction
17 CREATE TABLE IF NOT EXISTS transaction (
18     id VARCHAR(255) PRIMARY KEY,
19     credit_card_id VARCHAR(15) REFERENCES credit_card(id),
20     company_id VARCHAR(15) REFERENCES company(id);
21 );
```

Output

#	Time	Action	Message	Duration / Fetch
1	12:45:54	select * from transaction LIMIT 0, 1000	587 row(s) returned	0.000 sec / 0.000 sec

Gracias a este archivo podemos ver cómo se crea una base de datos desde cero

```
CREATE DATABASE IF NOT EXISTS transactions;  
USE transactions;
```

y cómo se crean las tablas con sus respectivas características.

En detalle, vemos que la tabla *company* está formada por seis columnas y contiene los detalles de las empresas, como el id, el nombre, el número de teléfono, el correo electrónico, el país y el sitio web.

A cada registro de estas columnas se le ha asignado un valor varchar, que representa un tipo de datos de cadena alfanumérica de longitud variable. Entre las seis columnas, la de *id* representa la Primary Key:

```
-- Creamos la tabla company  
> CREATE TABLE IF NOT EXISTS company (  
  id VARCHAR(15) PRIMARY KEY,  
  company_name VARCHAR(255),  
  phone VARCHAR(15),  
  email VARCHAR(100),  
  country VARCHAR(100),  
  website VARCHAR(255)  
);
```

A continuación, vemos los datos relacionados con la construcción de la tabla *transaction*. Esta tabla contiene nueve columnas con informaciones sobre las transacciones realizadas, como: el identificador de la transacción, el identificador de la tarjeta de crédito utilizada, el identificador de la empresa, el usuario que realizó la transacción, la ubicación geográfica de la transacción, la marca de tiempo, el monto y un campo booleano para indicar si la transacción fue rechazada o no. En la tabla *transaction*, a diferencia de la tabla *company*, existen otras categorías de datos además de varchar, como boolean, float, decimal, int y timestamp. La columna *id* representa la Primary Key y *company\_id* corresponde a la Foreign Key gracias a la cual se puede dar una relación con la tabla *company*.

```
-- Creamos la tabla transaction  
CREATE TABLE IF NOT EXISTS transaction (  
  id VARCHAR(255) PRIMARY KEY,  
  credit_card_id VARCHAR(15) REFERENCES credit_card(id),  
  company_id VARCHAR(20),  
  user_id INT REFERENCES user(id),  
  lat FLOAT,  
  longitude FLOAT,  
  timestamp TIMESTAMP,  
  amount DECIMAL(10, 2),  
  declined BOOLEAN,  
  FOREIGN KEY (company_id) REFERENCES company(id)
```

Después de cargar la base de datos *transactions*, gracias al archivo *estructura\_dades*, cargamos todos los registros de las dos tablas a través del archivo *dades\_introduir*.

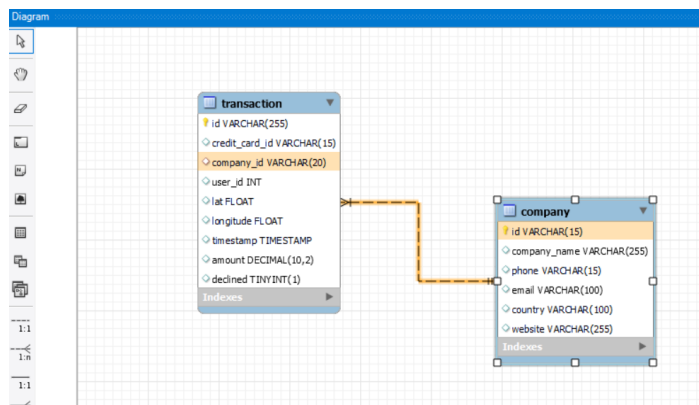
Datos de la tabla *company*:

```
1 • USE transactions;
2
3 -- Insertamos datos de company
4 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2222', 'Ac Fermentum Incorporated', '06 85 56 52 33', 'donec.porrtitor.tellus@yahoo
5 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2226', 'Magna A Neque Industries', '04 14 44 64 62', 'risus.donec.nibh@icloud.org'
6 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2230', 'Fusce Corp.', '08 14 97 58 85', 'risus@protonmail.edu', 'United States', '
7 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2234', 'Convallis In Incorporated', '06 66 57 29 50', 'mauris.ut@aol.couk', 'Germa
8 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2238', 'Ante Iaculis Nec Foundation', '08 23 04 99 53', 'sed.dictum.proin@outlook.
9 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2242', 'Donec Ltd', '01 25 51 37 37', 'at.iaculis@hotmail.couk', 'Norway', 'https:
10 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2246', 'Sed Nunc Ltd', '02 62 64 73 48', 'nibh@yahoo.org', 'United Kingdom', 'http:
11 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2250', 'Amet Nulla Donec Corporation', '07 15 25 14 74', 'mattis.integer.eu@protor
12 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2254', 'Nascetur Ridiculus Mus Inc.', '06 26 87 61 84', 'suspendisse.dui@icloud.ne
13 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2258', 'Vestibulum Lorem PC', '02 02 87 33 48', 'aenean.massa.integer@aol.net', 'E
14 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2262', 'Gravida Sagittis LLP', '03 81 28 33 97', 'turpis.vitae@google.ca', 'Sweder
15 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2266', 'Mus Aenean Eget Foundation', '06 25 15 52 43', 'mi.duis@hotmail.net', 'Swe
16 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES ( 'b-2270', 'Dis Parturient Institute', '05 36 29 78 74', 'purus@protonmail.org', 'Irel
```

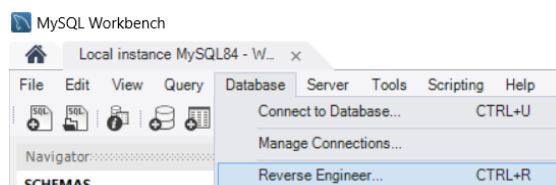
Datos de la tabla *transaction*:

```
105 -- Insertamos datos de transaction
106 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '10881D1D-5B23-A76C-55EF-C568E49A0500', 'Ccu-
107 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '7DC26247-20EC-53FE-E555-B6C2E55CA505', 'Ccu-
108 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '72997E96-DC2C-A4D7-7C24-66C302F8AE5A', 'Ccu-
109 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( 'A8069F53-965E-A2A8-CE06-CABC4FD92501', 'Ccu-
110 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '2F3B6AB6-147D-EB08-FEBD-9A4E2EA90B05', 'Ccu-
111 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '5B0EEF86-B8A1-EFAA-5EE1-27E7DC8F54A4', 'Ccu-
112 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '28928E3C-EC1A-A760-0A75-87147764906A', 'Ccu-
113 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '063F8AD9-99EC-60FB-20F7-25720D176A0A', 'Ccu-
114 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( 'D33B21E8-A61E-B0BC-5DE7-1DAAC210AE0', 'Ccu-
115 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '841AC1A0-9CAG-2AF7-EB2C-BC9C77C1EBB8', 'Ccu-
116 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( 'D3470F3E-96B3-799A-40F1-E42C143BAC5A', 'Ccu-
117 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( 'D6F37D7A-A07E-01AA-DE27-476CB9D26ABE', 'Ccu-
118 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( 'DE684792-SDEE-5E12-D0B5-C5661D48F42A', 'Ccu-
119 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) VALUES ( '4832160E-E11C-BBC3-41D8-B870468687A3', 'Ccu-
```

Una vez cargadas las dos tablas, podemos analizar el diagrama que ilustra la relación entre ambas:



Obtuvimos este diagrama utilizando la opción Reverse Engineer de MySQL Workbench:



A través del diagrama podemos identificar el tipo de modelo de nuestra base de datos, la tabla de hechos y la tabla de dimensiones, así como las respectivas claves primarias y foráneas.

La base de datos *transactions* es un modelo estrella (o dimensional) cuya tabla de hechos es la tabla *transaction* y la tabla de dimensiones es la tabla *company*. La relación entre las dos tablas es de muchos a uno.

A través de la relación entre las dos tablas podemos explorar en detalle el tipo de transacciones realizadas por las empresas.

## Nivel 1, Ejercicio 2

Para relacionar las dos tablas usaremos JOIN y queremos analizar los siguientes datos.

- Listado de los países que están haciendo compras:

```
1 • select distinct country
2   from transactions.company
3   join transactions.transaction
4   on company.id = transaction.company_id
5  where amount > 0;
```

The screenshot shows a SQL query execution interface. The query is: `select distinct country from transactions.company join transactions.transaction on company.id = transaction.company_id where amount > 0;`. The results are displayed in a table with one column, 'country', listing: Canada, Germany, Italy, United Kingdom, Sweden, United States, New Zealand, and Norway. Below the table, the 'Output' section shows a log entry: '1 13:08:26 select distinct country from transactions.company join transactions.transaction on company.id = transaction.co... 15 row(s) returned'.

A través de un *select distinct* obtenemos un listado sin duplicados de los países que están generando transacciones. Al escribir solo JOIN, por defecto se realizará un INNER JOIN entre la tabla *company* y la tabla de *transaction*. Insertando en el filtro *where* para obtener solo los importes mayores a 0, podemos recuperar la información relativa a las empresas que están generando compras.

- Desde cuántos países se realizan las compras:

```

12 • select count(distinct company.country) as country
13 from transactions.company
14 join transactions.transaction
15 on company.id = transaction.company_id
16 where amount > 0;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

country
15

Output

Action Output

#	Time	Action	Message	Duration /
1	13:35:12	select count(distinct company.country) as country from transactions.company join transactions.transaction on ...	1 row(s) returned	0.016 sec

Utilizando la función *count* podemos obtener información sobre el número de empresas que realizan transacciones. Para obtener el número único de países sin duplicados añadimos la clave *distinct* y el resultado es de 15 países.

- Identificar la compañía con la media más grande de ventas:

```

21 • select company.company_name, round(avg(transaction.amount), 2) as avg_sales
22 from transactions.company
23 join transactions.transaction
24 on company.id = transaction.company_id
25 group by company.company_name
26 order by avg_sales DESC
27 limit 1;
28

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows: |

company_name	avg_sales
Eget Ipsum Ltd	473.08

Result 20 x

Output

Action Output

#	Time	Action	Message
1	13:10:55	select company.company_name, round(avg(transaction.amount), 2) as avg_sales from transactions.company j...	1 row(s) returned

Para identificar la empresa con mayor promedio de ventas, buscamos el promedio en el *select* y, a través de *round*, lo redondeamos a dos decimales para facilitar la presentación. Luego conectamos las dos tablas para recuperar la información relativa a los montos y conectarla con el nombre de la empresa.

Las dos tablas se conectarán mediante la clave relativa al ID de la empresa (primary key de la tabla *company* y foreign key de la tabla *transaction*).

El *group by* agrupa los resultados por el nombre de la empresa (*company\_name*).

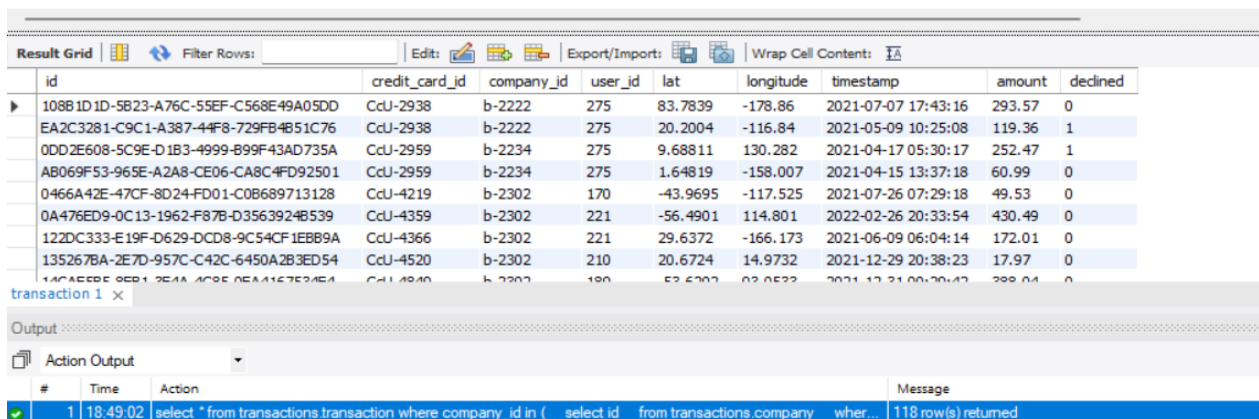
Los valores obtenidos se ordenan de manera descendente y la visualización se limita al primer resultado que contiene el valor más alto respecto a las ventas promedio de las empresas. Antes de poner el límite a 1, podría ser una buena práctica averiguar que no haya más de una empresa con el mismo promedio de ventas. De esta forma, no corremos el riesgo de perder informaciones relevantes.

### Nivel 1, Ejercicio 3

Para relacionar las dos tablas usaremos subqueries, en cambio de las JOINS, y queremos analizar los siguientes datos.

- Muestra todas las transacciones realizadas por empresas de Alemania:

```
1 • select *
2   from transactions.transaction
3  where company_id in (
4      select id
5      from transactions.company
6      where country = 'Germany'
7  );
```



id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
108B1D1D-5B23-A76C-55EF-C568E49A05DD	CcU-2938	b-2222	275	83.7839	-178.86	2021-07-07 17:43:16	293.57	0
EA2C3281-C9C1-A387-44F8-729FB4B51C76	CcU-2938	b-2222	275	20.2004	-116.84	2021-05-09 10:25:08	119.36	1
0DD2E608-5C9E-D1B3-4999-B99F43AD735A	CcU-2959	b-2234	275	9.68811	130.282	2021-04-17 05:30:17	252.47	1
AB069F53-965E-A2A8-CE06-CA8C4FD92501	CcU-2959	b-2234	275	1.64819	-158.007	2021-04-15 13:37:18	60.99	0
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	170	-43.9695	-117.525	2021-07-26 07:29:18	49.53	0
0A476ED9-0C13-1962-F87B-D3563924B539	CcU-4359	b-2302	221	-56.4901	114.801	2022-02-26 20:33:54	430.49	0
122DC333-E19F-D629-DCD8-9C54CF1EBB9A	CcU-4366	b-2302	221	29.6372	-166.173	2021-06-09 06:04:14	172.01	0
135267BA-2E7D-957C-C42C-6450A2B3ED54	CcU-4520	b-2302	210	20.6724	14.9732	2021-12-29 20:38:23	17.97	0

transaction 1 x

Output

Action Output

#	Time	Action	Message
1	18:49:02	select * from transactions.transaction where company_id in ( select id from transactions.company where country = 'Germany' );	118 row(s) returned

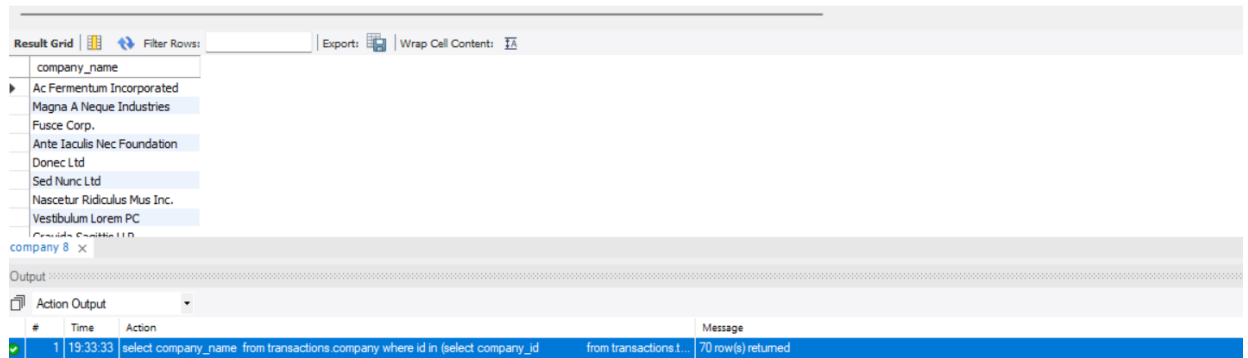
Esta consulta selecciona todas las transacciones realizadas por empresas ubicadas en Alemania. La subquery busca ID de empresas alemanas en la tabla *company*. La consulta externa devuelve todos los registros de la tabla *transaction* que contengan los *company\_id* filtrados por la subquery. Así podemos conocer cuantas empresas alemanas hicieron transacciones. También en las subconsultas, al igual que con las JOIN, las dos tablas se relacionan a través de sus respectivas primary key y foreign key.

- Lista las empresas que han realizado transacciones por un *amount* superior a la media de todas las transacciones:

```

10 • select company_name
11   from transactions.company
12  where id in (select company_id
13               from transactions.transaction
14               where amount > (select avg(amount) from transactions.transaction)
15             );

```



The screenshot shows a database interface with two main sections: 'Result Grid' and 'Output'.

**Result Grid:** This section displays the results of the SQL query. It shows a table with one column, 'company\_name', and a list of company names. The companies listed are: Ac Fermentum Incorporated, Magna A Neque Industries, Fusce Corp., Ante Taculis Nec Foundation, Donec Ltd, Sed Nunc Ltd, Nascetur Ridiculus Mus Inc., and Vestibulum Lorem PC. The table is scrollable, and the first few rows are visible.

**Output:** This section shows the execution details of the query. It includes a table with columns for '#', 'Time', 'Action', and 'Message'. The first row shows the query execution time as 19:33:33 and the message 'select company\_name from transactions.company where id in (select company\_id from transactions.transaction where amount > (select avg(amount) from transactions.transaction))'. The message also indicates that 70 row(s) were returned.

La consulta devuelve los nombres de las empresas que tienen transacciones por encima del monto promedio. Lo hace combinando subconsultas para filtrar los datos paso a paso y asegurarse de obtener exactamente la información necesaria.

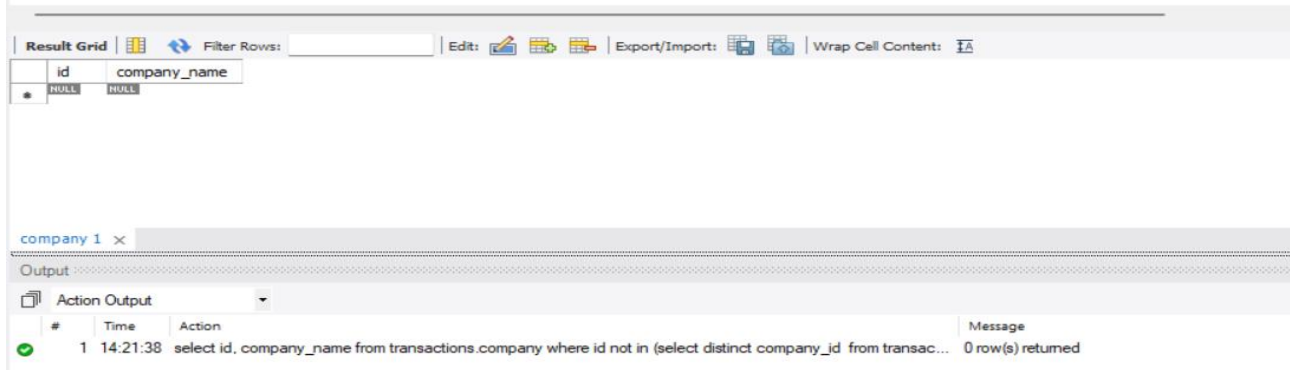
La subconsulta más interna calcula el promedio de los montos de todas las transacciones usando la función `AVG(amount)`. Esto nos devuelve un valor único, que es el monto promedio general.

La subconsulta intermedia selecciona los *company\_id* de aquellas empresas que tienen al menos una transacción con un monto mayor al promedio que obtuvimos antes. Básicamente, aquí identificamos qué empresas superan ese umbral.

La consulta principal busca en la tabla *company* los nombre (*company\_name*) de las empresas cuyo id coincide con los *company\_id* obtenidos en la subconsulta anterior. Aquí relacionamos las empresas con transacciones que superan el promedio.

- Eliminarán del sistema las empresas que no tienen transacciones registradas, entrega el listado de estas empresas:

```
1 • select id, company_name
2   from transactions.company
3  where id not in (select distinct company_id
4                  from transactions.transaction);
```



The screenshot shows a database management interface. At the top, a SQL query is entered in a text area. Below the query, there is a 'Result Grid' section. The grid has two columns: 'id' and 'company\_name'. The first row of data shows 'NULL' for both columns. Below the grid, there is an 'Output' section. The output shows a message: '0 row(s) returned'.

id	company_name
NULL	NULL

company 1 x

Output

Action Output

#	Time	Action	Message
1	14:21:38	select id, company_name from transactions.company where id not in (select distinct company_id from transac...	0 row(s) returned

El hecho de que el output muestre "0 row returned" indica que no hay empresas en la tabla *company* que no tengan transacciones asociadas. Esto significa que todas las empresas presentes en la base de datos tienen al menos una transacción registrada.

Para obtener este resultado, se utilizó una subconsulta que se basó en el NOT IN en la cláusula *where* para extraer los datos de las empresas que no estaban presentes en la tabla *transaction*, respondiendo así a la solicitud de entregar la lista de las empresas que no tienen transacciones registradas y eliminar esa lista del sistema, si fuera necesario.



## Nivel 2, Ejercicio 1

- Identifica los cinco días que se generó la cantidad más grande de ingresos a la empresa por ventas. Muestra la fecha de cada transacción junto con el total de las ventas:

```
4 • SELECT date(timestamp) AS transaction_date, SUM(amount) AS total_sales
5 FROM transactions.transaction
6 GROUP BY transaction_date
7 ORDER BY total_sales DESC
8 LIMIT 5;
```

transaction_date	total_sales
2021-03-29	1564.87
2021-12-20	1532.36
2021-06-15	1469.90
2021-05-09	1463.73
2021-06-21	1443.11

Result 4 x

Output

Action Output

#	Time	Action	Message
1	15:56:12	SELECT date(timestamp) AS transaction_date, SUM(amount) AS total_sales FROM transaction GROUP BY tr...	5 row(s) returned

La idea es identificar los 5 días en los que se generó la mayor cantidad de ingresos por ventas para la empresa. Para lograr esto, vamos a utilizar únicamente la tabla de transacciones.

Seleccionamos dos columnas de la tabla de transacciones: *timestamp* y *amount*.

Para extraer solo la fecha, sin la hora, se utiliza la función DATE.

Para obtener la suma total del monto de las transacciones de cada día se utiliza la función SUM sobre los registros de la columna *amount*.

Luego, agrupamos estos resultados por la fecha de transacción, usando GROUP BY.

Después, ordenamos los resultados de forma descendente según el total de ventas de cada día (*total\_sales*), de manera que los días con mayor facturación queden primero.

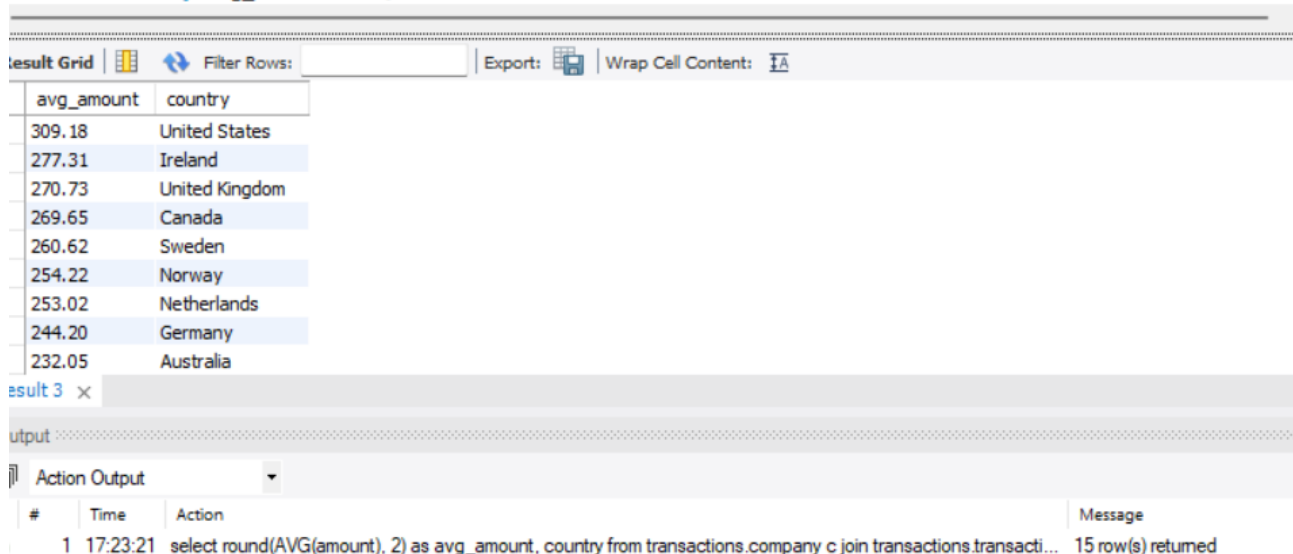
Por último, limitamos los resultados a solo los primeros 5 registros, usando LIMIT 5.

De esta manera, obtenemos la lista de los 5 días con el mayor volumen de ventas, mostrando la fecha y el total de ventas de cada uno de esos días

## Nivel 2, Ejercicio 2

- ¿Cuál es la media de ventas por país? Presenta los resultados ordenados de mayor a menor medio:

```
4 • select round(AVG(amount), 2) as avg_amount, country
5   from transactions.company c
6  join transactions.transaction t
7   on c.id = t.company_id
8  group by c.country
9  order by avg_amount desc;
```



result Grid | Filter Rows: | Export: | Wrap Cell Content: |

avg_amount	country
309.18	United States
277.31	Ireland
270.73	United Kingdom
269.65	Canada
260.62	Sweden
254.22	Norway
253.02	Netherlands
244.20	Germany
232.05	Australia

result 3 x

Output

Action Output

#	Time	Action	Message
1	17:23:21	select round(AVG(amount), 2) as avg_amount, country from transactions.company c join transactions.transacti...	15 row(s) returned

Para solucionar esta petición, calculamos el promedio de los montos (*amount*) de las transacciones por país y lo redondeamos a dos decimales para facilitar la presentación. Relacionamos las transacciones con la empresa correspondiente a través de sus identificadores (*company\_id* e *id*). Esto nos asegura que cada transacción está vinculada al país donde se encuentra la empresa. Agrupamos los datos por país, para que el promedio calculado sea específico de cada región. Ordenamos los resultados de forma descendente, mostrando primero los países con el promedio de ventas más alto.

## Nivel 2, Ejercicio 3

- En tu empresa, se plantea un nuevo proyecto para lanzar algunas campañas publicitarias para hacer competencia a la compañía "Non Institute". Para lo cual, te piden la lista de todas las transacciones realizadas por empresas que están situadas en el mismo país que esta compañía.

Primero mostramos el listado aplicando JOIN y subconsultas:

```
1 • select t.id as id_transaction, t.company_id, c.company_name, c.country
2   from transactions.transaction t
3  join transactions.company c
4    on t.company_id = c.id
5  where c.country = (
6    select country
7    from transactions.company
8   where company_name = 'Non Institute'
9  )
```

Result Grid

id_transaction	company_id	company_name	country
2B928E1C-EC14-A760-0A75-871477649D6A	b-2246	Sed Nunc Ltd	United Kingdom
ACD2011A-A2B1-C365-41E1-2AB00C65147A	b-2246	Sed Nunc Ltd	United Kingdom
4334349E-CEB0-3D68-A4D4-FEB7718A1ACE	b-2310	Non Magna LLC	United Kingdom
BC2B9A38-77B4-28CD-1FE8-14DED863E773	b-2310	Non Magna LLC	United Kingdom
1479B3D2-B7BA-C7BB-4CE3-8D7C2DE85ABB	b-2326	Enim Condimentum Ltd	United Kingdom
152598C2-029D-D684-4B66-91EDF393EBFF	b-2326	Enim Condimentum Ltd	United Kingdom
1B636B58-A2E8-7C69-D9C9-C54535DAFD3B	b-2326	Enim Condimentum Ltd	United Kingdom
20418DE5-B804-BE9B-BD7A-A95C1BFD8F5C	b-2326	Enim Condimentum Ltd	United Kingdom
239B8576-6C0E-137A-C2F6-3180A188A2D3	b-2326	Enim Condimentum Ltd	United Kingdom
267C4A86-7BA7-1C5E-0718-2824983C87DD	b-2326	Enim Condimentum Ltd	United Kingdom

Result 10 x

Output

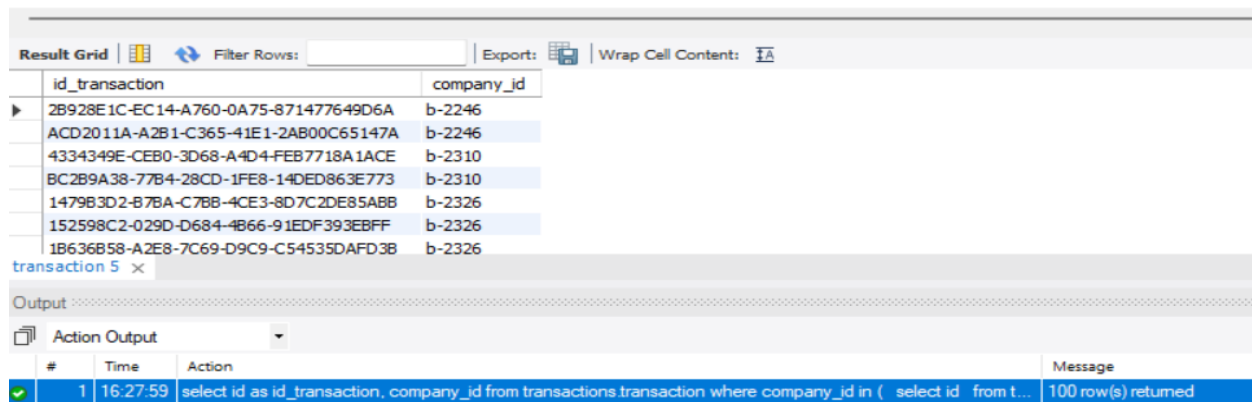
Action Output

#	Time	Action	Message
1	16:34:50	select t.id as id_transaction, t.company_id, c.company_name, c.country from transactions.transaction t join tr...	100 row(s) returned

Esta primera solución utiliza una JOIN para conectar las tablas *transaction* y *company*, combinado con una subconsulta para identificar el país de la compañía "Non Institute".

Ahora presentamos el mismo listado aplicando solo subconsultas:

```
1 • select id as id_transaction, company_id
2   from transactions.transaction
3  where company_id in (
4      select id
5      from transactions.company
6     where country = (
7         select country
8         from transactions.company
9        where company_name = 'Non Institute'
10     )
11 );
```



The screenshot shows a database interface with a 'Result Grid' and an 'Action Output' section. The 'Result Grid' displays a table with two columns: 'id\_transaction' and 'company\_id'. It lists 10 rows of transaction IDs and their corresponding company IDs. The 'Action Output' section shows a message indicating that the query returned 100 rows.

id_transaction	company_id
2B928E1C-EC14-A760-0A75-871477649D6A	b-2246
ACD2011A-A2B1-C365-41E1-2AB00C65147A	b-2246
4334349E-CEB0-3D68-A4D4-FEB7718A1ACE	b-2310
BC2B9A38-77B4-28CD-1FE8-14DED863E773	b-2310
1479B3D2-B7BA-C7BB-4CE3-8D7C2DE85ABB	b-2326
152598C2-029D-D684-4B66-91EDF393EBFF	b-2326
1B636B58-A2E8-7C69-D9C9-C54535DAFD3B	b-2326

transaction 5 x

Output:

Action Output

#	Time	Action	Message
1	16:27:59	select id as id_transaction, company_id from transactions.transaction where company_id in ( select id from t...	100 row(s) returned

Esta segunda solución utiliza dos subconsultas: una para encontrar el país y otra para identificar las empresas en ese país, filtrando directamente las transacciones y el id de las empresas.

Las dos consultas devuelven el mismo número de resultados (100 rows), pero los campos son distintos.

Si quisiéramos obtener la lista con la misma información además de tener el mismo resultado numérico, en la consulta con las subqueries habríamos tenido que incluir dos subconsultas en el SELECT, además de en el WHERE. De esta manera, podríamos extraer la información correspondiente a los campos *company\_name* y *country*. Cabe decir que esta versión con subconsultas sería menos eficiente en comparación con una solución basada en JOIN, porque cada subconsulta se ejecuta para cada fila resultante.

### Nivel 3, Ejercicio 1

- Presenta el nombre, teléfono, país, fecha y amount, de aquellas empresas que realizaron transacciones con un valor comprendido entre 100 y 200 euros y en alguna de estas fechas: 29 de abril del 2021, 20 de julio del 2021 y 13 de marzo del 2022. Ordena los resultados de mayor a menor cantidad.

```
1 • select c.company_name, c.phone, c.country, DATE(t.timestamp) AS transaction_date, t.amount
2   from transactions.company c
3  join transactions.transaction t
4   on c.id = t.company_id
5  where DATE(timestamp) IN ('2021-04-29', '2021-07-20', '2022-03-13') and amount BETWEEN 100 AND 200
6  ORDER BY t.amount desc;
```

	company_name	phone	country	transaction_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

Result 5 ×			
Output			
Action Output			
#	Time	Action	Message
✓ 1	12:24:43	select c.company_name, c.phone, c.country, DATE(t.timestamp) AS transaction_date, t.amount from transacti...	5 row(s) returned

Lo que hace esta consulta es mostrar información de las empresas que realizaron transacciones específicas. En concreto, seleccionamos el nombre de la empresa, su número de teléfono, el país, la fecha de la transacción (convertida a solo fecha usando DATE) y el monto de la transacción.

Primero, conectamos la tabla *company* (que contiene los datos de las empresas) con la tabla *transaction* (donde están las transacciones) utilizando una JOIN, vinculando el campo *id* de la tabla *company* con el campo *company\_id* de la tabla *transaction*. Esto permite relacionar cada transacción con su empresa correspondiente.

Luego, filtramos las transacciones con dos condiciones en la cláusula WHERE:

1. La fecha de la transacción debe ser una de estas tres: '2021-04-29', '2021-07-20' o '2022-03-13'. Para eso, usamos *DATE(timestamp)* para obtener solo la fecha de la columna *timestamp*.
2. El monto (*amount*) debe estar entre 100 y 200 euros.

Finalmente, ordenamos los resultados por el monto de las transacciones de mayor a menor usando ORDER BY t.amount DESC.

En resumen, esta consulta devuelve una lista ordenada de transacciones realizadas en fechas específicas, con montos en un rango concreto, mostrando también información de la empresa asociada.

### Nivel 3, Ejercicio 2

- Necesitamos optimizar la asignación de los recursos y dependerá de la capacidad operativa que se requiera, por lo cual te piden la información sobre la cantidad de transacciones que realicen las empresas, pero el departamento de recursos humanos es exigente y quiere un listado de las empresas donde especifiques si tienen más de 4 transacciones o menos:

```
1 • SELECT
2     c.company_name,
3     t.company_id,
4     COUNT(t.id) AS num_transactions,
5     'More than 4 transactions' AS transaction_status
6 FROM transactions.transaction t
7 JOIN transactions.company c
8 ON t.company_id = c.id
9 GROUP BY c.company_name, t.company_id
10 HAVING COUNT(t.id) > 4
11
12 UNION ALL
13
14 SELECT
15     c.company_name,
16     t.company_id,
17     COUNT(t.id) AS num_transactions,
18     'Less than 4 transactions' AS transaction_status
19 FROM transactions.transaction t
20 JOIN transactions.company c
21 ON t.company_id = c.id
22 GROUP BY c.company_name, t.company_id
```

```

12 UNION ALL
13
14 SELECT
15     c.company_name,
16     t.company_id,
17     COUNT(t.id) AS num_transactions,
18     'Less than 4 transactions' AS transaction_status
19 FROM transactions.transaction t
20 JOIN transactions.company c
21 ON t.company_id = c.id
22 GROUP BY c.company_name, t.company_id
23 HAVING COUNT(t.id) <= 4
24
25 ORDER BY num_transactions DESC;
26

```

company_name	company_id	num_transactions	transaction_status
Nunc Interdum Incorporated	b-2302	105	More than 4 transactions
Ut Semper Foundation	b-2346	59	More than 4 transactions
Enim Condimentum Ltd	b-2326	57	More than 4 transactions
Arcu LLP	b-2278	56	More than 4 transactions
Lorem Eu Incorporated	b-2362	54	More than 4 transactions
Malesuada PC	b-2494	52	More than 4 transactions
Non Institute	b-2618	30	More than 4 transactions
Ac Fermentum Incorporated	b-2222	2	Less than 4 transactions
Magna A Neque Industries	b-2226	2	Less than 4 transactions
Fusce Corp.	b-2230	2	Less than 4 transactions
Convallis In Incorporated	b-2234	2	Less than 4 transactions
Ante Iaculis Nec Foundation	b-2238	2	Less than 4 transactions

Result 2 x

Output

Action Output

#	Time	Action	Message
1	16:56:37	SELECT c.company_name, t.company_id, COUNT(t.id) AS num_transactions, 'More than 4 trans...	100 row(s) returned

Esta consulta está diseñada para clasificar a las empresas según la cantidad de transacciones que han realizado, dividiéndolas en dos categorías: aquellas que tienen más de cuatro transacciones y las que tienen cuatro o menos. Para lograr esto, primero seleccionamos las empresas que tienen más de cuatro transacciones, contando el número total de estas con la función COUNT. Además, mostramos el nombre y el ID de la empresa, junto con una etiqueta que indica "More than 4 transactions". El nombre de la empresa se ha incluido para que el resultado final sea más claro y útil. Esto lo logramos haciendo un JOIN entre las tablas *transaction* y *company*, aprovechando la relación entre *transaction.company\_id* y *company.id*.

Este resultado se filtra con la cláusula HAVING para asegurarnos de que solo se incluyan empresas con más de cuatro transacciones.

Luego, hacemos lo mismo para las empresas que tienen cuatro o menos transacciones. Usamos la misma lógica, pero aquí la etiqueta es "Less than 4 transactions", y el filtro en la cláusula HAVING asegura que solo se incluyan empresas con cuatro o menos.

Ambas partes se combinan utilizando UNION ALL, lo que nos permite juntar las dos categorías en una única tabla final. Finalmente, ordenamos todos los resultados de mayor a menor cantidad de transacciones con ORDER BY. Esto nos da un listado claro y organizado, que refleja cuántas transacciones tiene cada empresa y en qué categoría se encuentra.

Es cierto que esta consulta podría haberse resuelto con la función CASE, que permite crear etiquetas condicionales directamente dentro de una sola consulta. Sin embargo, preferí esta alternativa con UNION ALL porque me parece más intuitiva y fácil de leer. Al dividir la lógica en dos partes separadas, es más claro entender cómo se clasifica a las empresas en cada categoría, y el código se vuelve más sencillo de modificar si es necesario.