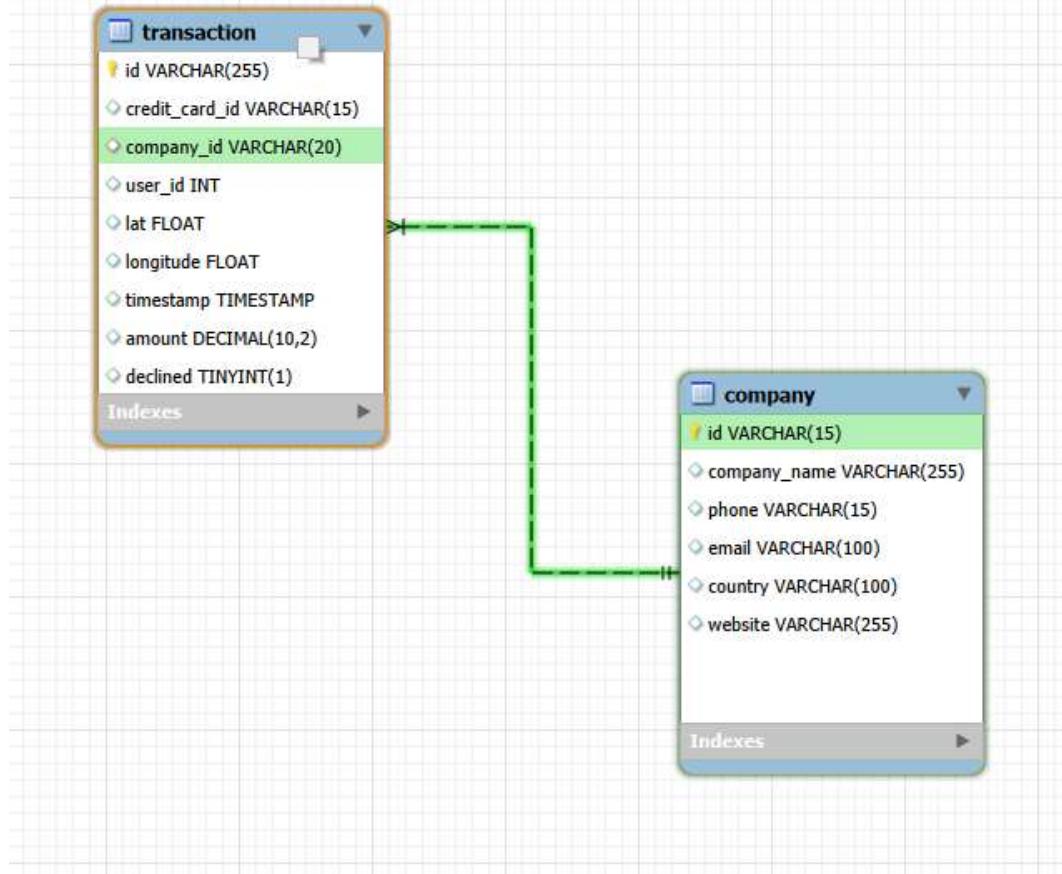


NIVEL 1

Ejercicio 1



La BBDD está compuesta por dos tablas principales: **company** y **transaction**, relacionadas entre sí mediante una clave foránea. El tipo de relación es de 1:N (uno a muchos). Se incluyen distintos tipos de datos: VARCHAR, INT, FLOAT, TIMESTAMP, DECIMAL, TINYINT.

La tabla **company** tiene como PRIMARY KEY el identificador **id VARCHAR(15)** que es único para cada empresa.

Por otro lado, la tabla **transaction** tiene como PRIMARY KEY **id VARCHAR(255)** que corresponde al identificador único de la transacción y como FOREIGN KEY **company_id VARCHAR(20)** que hace referencia a **company.id**.

Ejercicio 2 – Utilizando sólo JOIN

Para comprobar que la base de datos funciona correctamente, se realizan las siguientes acciones:

Comprobaciones previas

- Comprobación de valores NULL de la tabla company;

estructura dades Sprint 2*

```

1 • USE transactions;
2
3 # Comprobaciones previas
4 • SELECT COUNT(*)
  FROM transactions.company
 WHERE id IS NULL;

```

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

	COUNT(*)
▶	0

Output

Action Output

#	Time	Action	Message
1	13:34:50	SELECT COUNT(*) FROM transactions.company WHERE id IS NULL	1 row(s) returned

- Comprobación de valores NULL de la tabla transaction

estructura dades Sprint 2*

```

7
8 • SELECT COUNT(*)
  FROM transactions.transaction
 WHERE id IS NULL;
11

```

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

	COUNT(*)
▶	0

Output

Action Output

#	Time	Action	Message
1	13:34:50	SELECT COUNT(*) FROM transactions.company WHERE id IS NULL	1 row(s) returned
2	13:39:38	SELECT COUNT(*) FROM transactions.transaction WHERE id IS NULL	1 row(s) returned

- Comprobación de valores diferentes, misma cantidad de empresas en ambas tablas

estructura dades Sprint 2*

```

11
12 • SELECT COUNT(DISTINCT company_id)
  FROM transaction;
14
15

```

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

	COUNT(DISTINCT company_id)
▶	100

Output

Action Output

#	Time	Action	Message
1	13:42:51	USE transactions	0 row(s) affected
2	13:43:40	SELECT COUNT(DISTINCT company_id) FROM transaction	1 row(s) returned

estructura dades Sprint 2*

```

14
15
16 •   SELECT COUNT(DISTINCT id)
17     FROM company;
18

```

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

COUNT(DISTINCT id)
100

Output

Action Output

#	Time	Action	Message
1	13:42:51	USE transactions	0 row(s) affected
2	13:43:40	SELECT COUNT(DISTINCT company_id) FROM transaction	1 row(s) returned
3	13:48:20	SELECT COUNT(DISTINCT id) FROM company	1 row(s) returned

Como se puede observar, hay la misma cantidad de empresas (valores diferentes) tanto en la tabla company como en la tabla transaction.

Ejercicio 2.1.

- Listado de los países que están generando ventas:

Sprint 2

```

20
21     # EJERCICIO 2 UTILIZANDO JOIN
22     # Listado de países que están generando ventas
23 •   SELECT DISTINCT c.country
24     FROM company c
25     JOIN transaction t
26     ON c.id = t.company_id;
27

```

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

country
Germany
Australia
United States
New Zealand
Norway
United Kingdom
Italy
Belgium
Sweden
Ireland
China
Canada
France
Netherlands
Spain

Result 1

Action Output

#	Time	Action	Message
1	01:06:37	USE transactions	0 row(s) affected
2	01:16:39	SELECT DISTINCT c.country FROM company c JOIN transaction t ON c.id = t.compa...	15 row(s) returned

Aquí se usa el SELECT DISTINCT para obtener los países únicos que tienen al menos una transacción registrada. Para cruzar los datos y obtener sólo los que tienen coincidencias en ambas tablas usamos JOIN (INNER JOIN).

Ejercicio 2.2.

- Desde cuántos países se generan las ventas

The screenshot shows the MySQL Workbench interface with a query editor titled "Sprint 2". The query is:

```
27
28  # Desde cuántos países se generan ventas
29 •  SELECT COUNT(DISTINCT c.country) AS total_paises
30  FROM company c
31  JOIN transaction t
32  ON c.id=t.company_id;
```

The result grid shows a single row with the column "total_paises" containing the value "15". The output pane shows the following log entries:

#	Time	Action	Message
1	01:06:37	USE transactions	0 row(s) affected
2	01:16:39	SELECT DISTINCT c.country FROM company c JOIN transaction t ON c.id = t.company_id	15 row(s) returned
3	01:22:21	SELECT COUNT(DISTINCT c.country) AS total_paises FROM company c JOIN transaction t ON c.id = t.company_id	1 row(s) returned
4	01:22:56	SELECT COUNT(DISTINCT c.country) AS total_paises FROM company c JOIN transaction t ON c.id = t.company_id	1 row(s) returned

En esta consulta se pide un número entero por este motivo usamos COUNT DISTINCT de la columna **country** de la tabla company y se le otorga un alias con AS total_paises.

Ejercicio 2.3.

- Identifica la empresa con la media más grande de ventas

The screenshot shows the MySQL Workbench interface with a query editor titled "Sprint 2". The query is:

```
33
34  # Identifica la empresa con la media más grande de ventas
35 •  SELECT c.company_name, ROUND(AVG(t.amount),2) AS media_ventas
36  FROM company c
37  JOIN transaction t
38  ON c.id = t.company_id
39  WHERE t.declined = 0
40  GROUP BY c.company_name
41  ORDER BY media_ventas DESC
42  LIMIT 1;
```

The result grid shows a single row with columns "company_name" and "media_ventas", both containing the value "Ac Fermentum Incorporated". The output pane shows the following log entries:

#	Time	Action	Message
1	01:06:37	USE transactions	0 row(s) affected
2	01:16:39	SELECT DISTINCT c.country FROM company c JOIN transaction t ON c.id = t.company_id	15 row(s) returned
3	01:22:21	SELECT COUNT(DISTINCT c.country) AS total_paises FROM company c JOIN transaction t ON c.id = t.company_id	1 row(s) returned
4	01:22:56	SELECT COUNT(DISTINCT c.country) AS total_paises FROM company c JOIN transaction t ON c.id = t.company_id	1 row(s) returned
5	01:43:41	SELECT c.company_name, ROUND(AVG(t.amount),2) AS media_ventas FROM company c JOIN transaction t ON c.id = t.company_id WHERE t.declined = 0 GROUP BY c.company_name ORDER BY media_ventas DESC LIMIT 1;	1 row(s) returned

The screenshot shows the MySQL Workbench interface with a query editor titled "Result 4". The query is identical to the one in the previous screenshot:

```
33
34  # Identifica la empresa con la media más grande de ventas
35 •  SELECT c.company_name, ROUND(AVG(t.amount),2) AS media_ventas
36  FROM company c
37  JOIN transaction t
38  ON c.id = t.company_id
39  WHERE t.declined = 0
40  GROUP BY c.company_name
41  ORDER BY media_ventas DESC
42  LIMIT 1;
```

The output pane shows the same log entries as the previous screenshot, indicating the query was executed successfully.

En esta consulta se introduce la función AVG para obtener la media o promedio de la columna **amount**, se redondea a dos decimales con **ROUND**, se agrupa por nombre de la empresa, se ordena de forma descendente y se limita al primer registro mediante **LIMIT 1**. Además, se usa la expresión **declined = 0** (0 representa FALSE mientras que 1 representa TRUE) para eliminar las transacciones que hayan sido rechazadas.

Ejercicio 3 – Utilizando sólo subconsultas

Ejercicio 3.1.

- Muestra todas las transacciones realizadas por empresas de Alemania

```

estructura dades Sprint 2* 
46 •  SELECT * 
47   FROM transaction 
48   WHERE declined = 0 
49   AND company_id IN ( 
50     SELECT id 
51       FROM company 
52      WHERE country = 'Germany' 
53 );

```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount
00138D3B-206D-4C03-94B7-63A2676EB9B4	CcS-4899	b-2222	318	41.3781	12.447	2020-03-25 10:43:43	426.36
0013C1B6-3B84-4D6C-8154-E2B3FEBCA8E9	CcS-5070	b-2222	489	41.3814	2.18176	2020-12-17 18:15:37	316.90
00201A11-2E62-44C4-941D-198FC8DB77F0	CcU-3512	b-2222	193	55.5704	-3.65129	2021-01-22 23:44:27	453.04
00235618-045C-4D49-9DCB-B3A9405D8923	CcS-8137	b-2222	3556	59.8421	18.729	2020-09-09 15:43:19	263.14
005A5A7B-1F1A-4B6C-9B15-1625A78C9C38	CcS-8998	b-2222	4417	41.1591	-8.63905	2024-05-15 09:10:11	442.01

transaction 1 ×

Output

#	Time	Action	Message	Duration / Fetch
1	10:33:24	USE transactions	0 row(s) affected	0.000 sec
2	11:03:30	SELECT * FROM transaction WHERE declined = 0 AND co...	13269 row(s) returned	0.000 sec / 0.078 sec

En esta consulta primero realizamos una subconsulta para delimitar el país ‘Germany’. Una vez acotado el país, la introducimos en la OUTER QUERY mediante la FOREIGN KEY **company_id** en la tabla **transaction** que la vincula a la PRIMARY KEY **id** en la tabla **company**.

Ejercicio 3.2.

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

estructura dades Sprint 2*

```

55  # Lista de las empresas que han realizado transacciones por un "amount" superior a la
56  # media de todas las transacciones
57 •  SELECT company_name
58   FROM company
59   WHERE EXISTS (
60     SELECT company_id
61     FROM transaction
62     WHERE amount > (
63       SELECT AVG(amount)
64       FROM transaction
65       WHERE declined = 0
66     )
67     AND declined = 0
68   );
69

```

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

company_name
Ac Fermentum Incorporated
Magna A Neque Industries
Fusce Corp.
Convallis In Incorporated
Ante Iaculis Nec Foundation
Donec Ltd
Sed Nunc Ltd
Amet Nulla Donec Corporation

company 7 Read Only

Output

Action Output
Time Action Message Duration / Fetch
11 12:26:54 SELECT company_name FROM company WHERE id IN (...) 100 row(s) returned 0.125 sec / 0.000 sec
12 12:27:17 SELECT company_name FROM company WHERE EXISTS (...) 100 row(s) returned 0.047 sec / 0.000 sec

En este ejercicio tenemos 3 capas o niveles de consultas. QUERY-SUBQUERY NIVEL 2- SUBQUERY 1, es decir, hay una SUBQUERY dentro de otra SUBQUERY que a su vez está dentro de una MAIN QUERY (OUTER QUERY).

En la primera SUBQUERY se busca el promedio (**AVG**) del amount de todas las transacciones excluyendo las transacciones rechazadas (**declined = 0**).

A continuación, introducimos esta información a la SUBQUERY superior mediante la cláusula **WHERE** para encontrar las empresas que cumplen dicha condición y también excluimos las transacciones rechazadas (**declined = 0**), este paso se repite porque ahora sí que sirve para descartar las transacciones rechazadas mientras que en la SUBQUERY que busca el **AVG** se usa para no desvirtuar el resultado final del promedio.

Por último, buscamos las empresas que cumplen las condiciones establecidas en las SUBQUERIES.

Ejercicio 3.3.

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

Opción A – Subconsulta correlacionada

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Shows the query:

```
69
70      # Eliminarán del sistema las empresas que no tienen transacciones registradas, entrega el listado
71      # de estas empresas
72 •  SELECT *
73      FROM company c
74      WHERE NOT EXISTS (
75          SELECT 1
76          FROM transaction t
77          WHERE t.company_id = c.id
78      );
```
- Result Grid:** Displays a table with columns: id, company_name, phone, email, country, website. The data row is: * NULL NULL NULL NULL NULL NULL.
- Action Output:** Shows the log entry:

#	Time	Action	Message	Duration / Fetch
1	11:06:07	SELECT * FROM company c WHERE NOT EXISTS (SEL...	0 row(s) returned	0.000 sec / 0.000 sec

En este ejercicio buscaremos, mediante una SUBQUERY CORRELACIONADA, las empresas de la tabla **company** que no figuren (NOT EXISTS) en la tabla **transaction**. Con SELECT usamos 1 porque no nos interesa recuperar todas las columnas y filas de la tabla especificada, sólo saber si existen registros devolviendo 1 para cada fila que coincida.

En esta última consulta no incluimos declined = 0 porque el objetivo es encontrar empresas que no tiene ningún registro en la tabla **transaction**, independientemente de si fueron rechazadas o no.

Opción B – Usando NOT IN

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Shows the query:

```
80
81      # OPCIÓN B - Usando NOT IN
82 •  SELECT *
83      FROM company
84      WHERE id NOT IN (
85          SELECT company_id
86          FROM transaction
87      );
```
- Result Grid:** Displays a table with columns: id, company_name, phone, email, country, website. The data row is: * NULL NULL NULL NULL NULL NULL.
- Action Output:** Shows the log entry:

#	Time	Action	Message	Duration / Fetch
1	11:07:34	SELECT * FROM company WHERE id NOT IN (SELECT c...	0 row(s) returned	0.063 sec / 0.000 sec

NIVEL 2

Ejercicio 1

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

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```
94 •  SELECT DATE(timestamp) AS fecha, SUM(amount) AS total_ingeresos
95   FROM transaction
96  WHERE declined = 0
97  GROUP BY fecha
98 ORDER BY total_ingeresos DESC
99 LIMIT 5;
```

The results grid displays the following data:

fecha	total_ingeresos
2022-12-13	14337.44
2019-11-18	13591.32
2023-02-20	13332.59
2017-12-20	13318.43
2019-03-18	12680.95

The output pane shows the execution details:

Action Output	#	Time	Action	Message	Duration / Fetch
1	11:53:12	SELECT DATE(timestamp) AS fecha, SUM(amount) AS tota...	5 row(s) returned		0.125 sec / 0.000 sec

Para esta consulta, debemos extraer la fecha de la columna **timestamp**, sumar el **amount** de todas las transacciones que no fueron rechazadas (**declined = 0**) y agrupar por día.

Se usa **DATE()** para ignorar la hora del **timestamp**. Filtramos con **WHERE declined = 0** para asegurar que sólo contamos ventas exitosas. Finalmente, ordenamos de mayor a menor y limitamos a los 5 primeros resultados.

Ejercicio 2

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

The screenshot shows the MySQL Workbench interface with a query editor and results grid. The query is:

```
104 •  SELECT c.country AS pais, AVG(t.amount) AS media_ventas
105   FROM transaction t
106  JOIN company c
107  ON t.company_id = c.id
108 WHERE t.declined = 0
109 GROUP BY c.country
110 ORDER BY media_ventas DESC;
```

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

pais	media_ventas
Australia	265.535393
United States	264.419466
Belgium	260.971218
Germany	260.829097
Ireland	260.388751
Spain	260.276923
France	259.905738
New Zealand	259.585048
Norway	259.141783
Netherlands	258.336546
Italy	258.123060
Canada	257.413501

Result 9 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:16:17	SELECT c.country AS pais, AVG(t.amount) AS media_ventas...	15 row(s) returned	0.312 sec / 0.000 sec

Para responder a esta consulta, es necesario realizar un **JOIN** entre ambas tablas, ya que la información del país está en la tabla **company** y el monto de la venta en **transaction**.

Se unen las tablas a través de **company_id** de la tabla **transaction** e id de la tabla **company**. La función **AVG()** calcula el promedio de los montos y el **ORDER BY ... DESC** asegura que los países con mayor media aparezcan primero.

Ejercicio 3

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

- Muestra el listado aplicando JOIN y subconsultas

estructura dades Sprint 2* ×

```

115 •  SELECT t.*
116   FROM transaction t
117   JOIN company c
118   ON t.company_id = c.id
119   WHERE t.declined = 0
120   AND c.country = (
121     SELECT country
122     FROM company
123     WHERE company_name = 'Non Institute')
124   AND company_name != 'Non Institute';
  
```

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

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount
008629B4-C9A9-406C-A3D2-71FDA47BC546	CcS-7063	b-2246	2482	45.7666	4.83048	2015-07-30 12:12:42	486.4
00B72BA4-54A3-4B8E-B13F-2D57535AA17A	CcS-8475	b-2246	3894	55.6212	-3.7546	2017-10-26 22:08:26	414.0
01F075B1-D7AE-4D02-AAD9-5FFD72A43F3C	CcS-8700	b-2246	4119	55.856	-3.15783	2018-01-27 13:44:36	103.7
023FFCE8-E618-4938-BF56-C8DF80540ADD	CcS-7816	b-2246	3235	46.3568	1.82755	2016-12-19 11:53:45	219.2
02683BE8-EF91-4564-957B-D6F1662AB7C5	CcS-9471	b-2246	4890	42.1332	12.396	2017-01-10 21:09:29	326.8
02C2F29E-CEF2-4C1E-A594-F476E8F279C0	CcS-9082	b-2246	4501	39.4662	-0.373246	2020-05-24 01:17:29	155.7
02F468DC-426C-47C2-880A-D8825B7A81AF	CcS-6913	b-2246	2332	52.175	19.3508	2023-03-17 16:36:27	305.3

Result 11 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	12:57:36	SELECT t.* FROM transaction t JOIN company c ON t.com...	12213 row(s) returned	0.000 sec / 0.062 sec

En este caso, la subconsulta obtiene el país de “Non Institute”, y el **JOIN** permite filtrar tanto por el país de la empresa como por el estado de la transacción. Como interesa analizar las otras empresas del país, es decir, la competencia local, se añade la condición **c.company_name != ‘Non Institute’**.

- Muestra el listado aplicando sólo subconsultas

The screenshot shows the MySQL Workbench interface. The query editor window displays the following SQL code:

```

127 •  SELECT *
128   FROM transaction
129   WHERE declined = 0
130   AND company_id IN (
131     SELECT id
132       FROM company
133     WHERE country = (
134       SELECT country
135         FROM company
136       WHERE company_name = 'Non Institute'
137       AND company_name != 'Non Institute'
138     );
  
```

The results grid below shows the output of the query. The columns are: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, and amount. The data includes:

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount
▶	008629B4-C9A9-406C-A3D2-71FDA47BC546	CcS-7063	b-2246	2482	45.7666	4.83048	2015-07-30 12:12:42	486.44
	00B72BA4-54A3-4B8E-B13F-2D57535AA17A	CcS-8475	b-2246	3894	55.6212	-3.7546	2017-10-26 22:08:26	414.06
	01F075B1-D7AE-4D02-AAD9-5FFD72A43F3C	CcS-8700	b-2246	4119	55.856	-3.15783	2018-01-27 13:44:36	103.73
	023FFCE8-E618-4938-BF56-C8DF80540ADD	CcS-7816	b-2246	3235	46.3568	1.82755	2016-12-19 11:53:45	219.28

The transaction details window shows the following output:

#	Time	Action	Message	Duration / Fetch
1	13:28:12	SELECT * FROM transaction WHERE declined = 0 AND c...	12213 row(s) returned	0.000 sec / 0.047 sec

Aquí se filtra directamente en la tabla **transaction**. La subconsulta anidada identifica primero el país objetivo y luego recupera todos los IDs de empresas de dicho país para filtrar la tabla principal

NIVEL 3

Ejercicio 1

Presenta el nombre, teléfono, país, fecha y amount, de aquellas empresas que realizaron transacciones con un valor comprendido entre 350 y 400 euros y en alguna de estas fechas: 29 de abril del 2015, 20 de julio del 2018 y 13 de marzo del 2024. Ordena los resultados de mayor a menor cantidad.

Se ha utilizado la función **DATE(timestamp)** para asegurar que la comparación se haga sólo por el día, ignorando la hora, y el operador **IN** para agrupar las tres fechas específicas.

El rango de importes se ha acotado usando el operador **BETWEEN 350 AND 400** incluyendo ambos límites. Como se ha visto anteriormente, se añade la condición **declined = 0** para mostrar sólo las transacciones exitosas.

Por último, se utiliza **ORDER BY t.amount DESC** para cumplir con el requisito de mayor a menor cantidad.

estructura dades Sprint 2*

```

145 •  SELECT c.company_name AS nombre, c.phone AS telefono, c.country AS pais,
146      DATE(t.timestamp) AS fecha, t.amount AS importe
147  FROM company c
148 JOIN transaction t
149 ON c.id = t.company_id
150 WHERE t.declined = 0
151 AND t.amount BETWEEN 350 AND 400
152 AND DATE(t.timestamp) IN ('2015-04-29', '2018-07-20', '2024-03-13')
153 ORDER BY t.amount DESC;

```

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

nombre	telefono	pais	fecha	importe
Aliquam PC	01 45 73 52 16	Germany	2024-03-13	399.84
Auctor Mauris Vel LLP	08 09 28 74 14	United States	2018-07-20	399.51
At Pede Corp.	06 14 48 33 15	Italy	2015-04-29	390.69
Aliquam PC	01 45 73 52 16	Germany	2024-03-13	388.29
Orci Adipiscing Limited	03 18 00 77 81	United Kingdom	2018-07-20	373.71
Fringilla LLC	08 29 15 93 57	New Zealand	2015-04-29	367.62
Pede Cum Ltd	07 62 26 48 38	Norway	2018-07-20	356.87
Auctor Mauris Vel LLP	08 09 28 74 14	United States	2024-03-13	353.75

Result 16 × Read Only

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:48:28	SELECTc.company_name AS nombre, c.phone AS telefon...	8 row(s) returned	0.078 sec / 0.000 sec

Ejercicio 2

Necesitamos optimizar la asignación de los recursos y dependerá de la capacidad operativa que se requiera, por lo cual te piden información sobre la cantidad de transacciones que realizan las empresas, pero el departamento de recursos humanos es exigente y quiere un listado de las empresas donde especifiques si tienen más de 400 transacciones o menos.

Sprint 2*

```

160 •  SELECT c.company_name, COUNT(t.id) AS total_transacciones,
161   CASE
162     WHEN COUNT(t.id) > 400 THEN 'Más de 400'
163     ELSE '400 o menos'
164   END AS categoria
165  FROM company c
166 LEFT JOIN transaction t
167    ON t.company_id = c.id
168    AND t.declined = 0
169 GROUP BY c.company_name;

```

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

company_name	total_transacciones	categoria
Ac Fermentum Incorporated	2400	Más de 400
Magna A Neque Industries	406	Más de 400
Fusce Corp.	445	Más de 400
Convallis In Incorporated	1512	Más de 400
Ante Iaculis Nec Foundation	470	Más de 400
Donec Ltd	447	Más de 400
Sed Nunc Ltd	1538	Más de 400
Amet Nulla Donec Corporation	1508	Más de 400

Result 1 × Read Only

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:08:06	USE transactions	0 row(s) affected	0.000 sec
2	17:08:26	SELECTc.company_name, COUNTt.id) AS total_transacciones, CASE WHEN ...	100 row(s) returned	0.750 sec / 0.000 sec

Para esta consulta se usa **LEFT JOIN** porque queremos que aparezcan todas las empresas, incluso las que tienen 0 transacciones. Si se usara **INNER JOIN**, las empresas sin transacciones desaparecerían.

Con **COUNT(t.id)** cuenta las transacciones por empresa. A continuación, se aplica **CASE** que clasifica según la cantidad. Se asemeja a una función de Python. Dentro del **LEFT JOIN** se incluye la condición **declined = 0** para filtrar las transacciones rechazadas.

Por último, se agrupa por **c.company_name**.