

SPRINT 2: Bases de datos relacionales e introducción a SQL

Elaborado por:

Vanesa Pérez Ramírez

Fecha: 18/09/2025

SPRINT 2: Bases de datos relacionales e introducción a SQL.....	1
Nivel 1	3
Ejercicio 1	3
a) A partir de los documentos adjuntos, importa las dos tablas.	3
b) Muestra las características principales del esquema creado y explica las diferentes tablas y variables que existen. Asegúrate de incluir un diagrama que ilustre las diferentes tablas y variables.....	3
Ejercicio 2	6
a) Lista de países que están generando ventas.	6
b) Desde cuantos países se generan las ventas.	7
c) Identifica la compañía con la media más grande de ventas.....	8
Ejercicio 3	9
a) Muestras todas las transacciones realizadas por las empresas de Alemania	9
b) Lista las empresas que han realizado transacciones por una amount superior a la media de todas las transacciones.	10
c) Eliminar del sistema las empresas que no tienen transacciones registradas, entrega el listado de estas empresas.....	11
Nivel 2	12
Ejercicio 1	12
Ejercicio 2	13
Ejercicio 3	14
a) Muestra el listado aplicando JOIN y subconsultas.	14
b) Muestra el listado sólo aplicando subconsultas.	15
Nivel 3	16
Ejercicio 1	16
Ejercicio 2	17

Nivel 1

Ejercicio 1

a) A partir de los documentos adjuntos, importa las dos tablas.

El primer archivo que se carga y se ejecuta es el de “estructura de dades” con él se han creado la base de datos llamada **transactions** con sus tablas: **company** y **transaction**.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'transactions' expanded, showing 'company' and 'transaction' tables. The center pane shows the SQL script:

```

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

```

The right pane shows the 'Output' window with the following data:

#	Time	Action	Message	Duration / Fetch
100099	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.046 sec
100100	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.047 sec
100101	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.032 sec
100102	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.078 sec
100103	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.093 sec
100104	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.094 sec
100105	12.21.03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.110 sec

A continuación, se ha cargado y ejecutado el archivo “dades_introducir_sprint2”, con esta acción se han cargado los registros de las tablas.

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with 'transactions' expanded, showing 'company' and 'transaction' tables. The center pane shows the SQL script:

```

1  USE transactions;
2
3  -- Insertamos datos de company
4  INSERT INTO company (id, company_name, phone, email, country, website) VALUES (
5  'b-2222',
6  'b-2226',
7  'b-2230',
8  'b-2234',
9  'b-2238',
10 'b-2242',
11 'b-2246',
12 'b-2250',
13 'b-2254',
14 'b-2258',
15 'b-2262',
16 );

```

The right pane shows the 'Output' window with the following data:

#	Time	Action	Message	Duration / Fetch
99610	12.20.39	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.032 sec
99611	12.20.39	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.031 sec
99612	12.20.39	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.062 sec
99613	12.20.40	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.032 sec
99614	12.20.40	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.031 sec
99615	12.20.40	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.031 sec
99616	12.20.40	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longi...)	1 row(s) affected	0.031 sec

b) Muestra las características principales del esquema creado y explica las diferentes tablas y variables que existen. Asegúrate de incluir un diagrama que ilustre las diferentes tablas y variables.

La base de datos creada es **transactions**. Es una base de datos formada por dos tablas relacionadas:

- Company
- Transaction

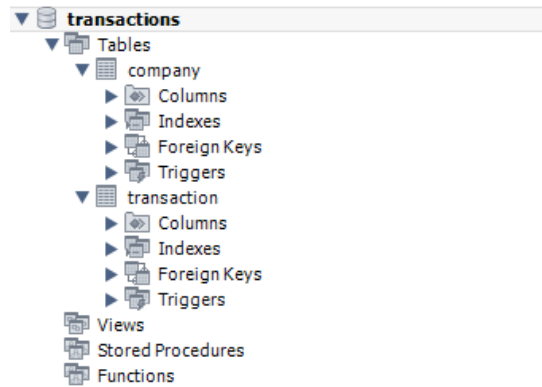


TABLA COMPANY

La tabla company nos da información de las compañías. Dispone de 6 variables y 100 registros. El nombre de sus variables y las características de los datos que contienen son las que aparecen en la siguiente tabla:

Table: company

Columns:

id	varchar(15) PK
company_name	varchar(255)
phone	varchar(15)
email	varchar(100)
country	varchar(100)
website	varchar(255)

La variable id es la Primary Key.

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

TABLA TRANSACTION

La tabla transacción nos da la información sobre las transacciones de los cobros con tarjeta de las diferentes compañías clientes. La tabla transaction dispone de 9 variables y 100000 registros.

Los nombres y las características de las variables son las que aparecen en la siguiente tabla:

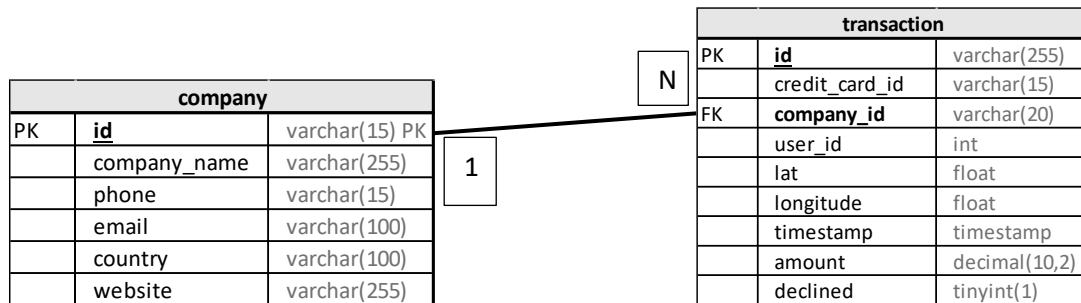
Table: transaction

Columns:

id	varchar(255) PK
credit_card_id	varchar(15)
company_id	varchar(20)
user_id	int
lat	float
longitude	float
timestamp	timestamp
amount	decimal(10,2)
declined	tinyint(1)

Automatic context help disabled. Use the toolbar manually get help for the current caret position or toggle automatic help.

Donde la variable id es la Primary Key de la tabla y la variable company_id es la Foreign Key que relaciona la tabla transaction con la variable id de la tabla company. La relación existente es 1 a N y se muestra en la siguiente imagen:



Ejercicio 2

Utilizando JOIN realizarás las siguientes consultas:

a) Lista de países que están generando ventas.

En este caso, para obtener el nombre de los países que generan ventas he realizado una join entre las tablas company y transaction de la bbdd y he seleccionado la variable country con un DISTINCT para que no aparezcan las duplicidades de los países que salen de las transacciones los filtros que aplicado a través del WHERE han sido las transacciones con un valor superior a 0 para no tener en cuenta si hay alguna mal realizada y las declined = 0 para no tener en cuenta las transacciones que hayan sido rechazadas.

```

4 #Lista de países que están generando ventas
5 SELECT DISTINCT company.country
6 FROM company
7 JOIN transaction
8 ON transaction.company_id = company.id
9 WHERE transaction.amount > 0 AND transaction.declined = 0
10 ORDER BY country;

```

Result Grid

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

Result 3

Output

Action Output

#	Time	Action	Message
1	10:21:56	SELECT * FROM transaction WHERE EXISTS (SELECT id FROM company WHERE country = "Germany" AND company.id = transaction.id)	13291 row(s) returned
2	10:24:54	SELECT DISTINCT company.country FROM company WHERE EXISTS (SELECT transaction.company_id FROM transaction WHERE transaction.amount > 0 AND transaction.declined = 0)	100 row(s) returned
3	10:26:57	SELECT DISTINCT company.country FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction.amount > 0 AND transaction.declined = 0	15 row(s) returned

b) Desde cuantos países se generan las ventas.

En este caso, para obtener el nombre de los países que generan ventas he realizado una join entre las tablas company y transaction de la bbdd y he aplicado la función de agregación COUNT aplicado a la variable country con un DISTINCT para que no aparezcan las duplicidades de los países que salen de las transacciones los filtros que aplicado a través del WHERE han sido las transacciones con un valor superior a 0 para no tener en cuenta si hay alguna mal realizada y las declined = 0 para no tener en cuenta las transacciones que hayan sido reclinadas.

The screenshot shows a SQL IDE interface with a query editor and a results panel. The query is as follows:

```

13
14 # Desde cuantos países se generan las ventas.
15 • SELECT count(distinct company.country) as num_paises_con_ventas
16 FROM company
17 JOIN transaction
18 ON transaction.company_id = company.id
19 WHERE transaction.amount > 0 AND transaction.declined = 0;

```

The results panel shows a single row with the value 15 for the column num_paises_con_ventas.

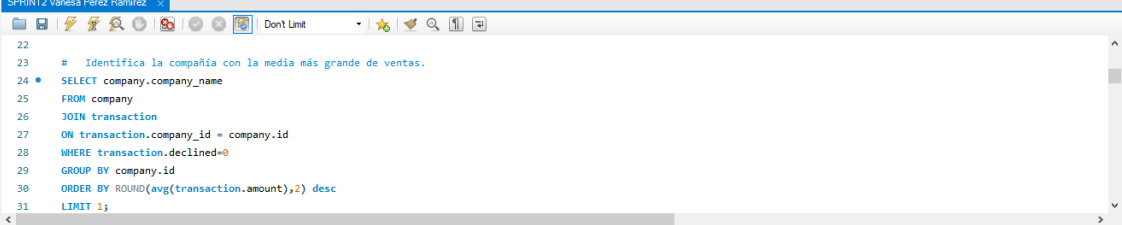
num_paises_con_ventas
15

The output panel shows the execution log with the following messages:

#	Time	Action	Message
1	10:21:56	SELECT * FROM transaction WHERE EXISTS(SELECT id FROM company WHERE country = "Germany" AND company.id = tran...	13291 row(s) returned
2	10:24:54	SELECT DISTINCT company.company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transaction...	100 row(s) returned
3	10:26:57	SELECT DISTINCT company.country FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction...	15 row(s) returned
4	10:28:17	SELECT count(distinct company.country) as num_paises_con_ventas FROM company JOIN transaction ON transaction.company_id = c...	1 row(s) returned

c) *Identifica la compañía con la media más grande de ventas.*

He seleccionado el nombre de la compañía, realizando una JOIN entre las tablas de la base de datos y aplicando un filtro para eliminar las ventas declinadas, y he realizado una agrupación por el id de la compañía en el que he aplicado una ordenación de mayor a menor de los registros en función de la media de ventas de cada grupo y limitando el resultado que dá por pantalla a 1 para que dé el primero de la lista que será la compañía que tiene la media de ventas mayor.



```

22
23 # Identifica la compañía con la media más grande de ventas.
24 • SELECT company.company_name
25 FROM company
26 JOIN transaction
27 ON transaction.company_id = company.id
28 WHERE transaction.declined=0
29 GROUP BY company.id
30 ORDER BY ROUND(avg(transaction.amount),2) desc
31 LIMIT 1;

```

Result Grid

company_name
Ac Fermentum Incorporated

Result 6 x

Output

#	Time	Action	Message
1	10:21:56	SELECT * FROM transaction WHERE EXISTS(SELECT id FROM company WHERE country = "Germany" AND company.id = tran...	13291 row(s) returned
2	10:24:54	SELECT DISTINCT company.company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transaction...	100 row(s) returned
3	10:26:57	SELECT DISTINCT company.country FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction...	15 row(s) returned
4	10:28:17	SELECT count(distinct company.country) as num_paises_con_ventas FROM company JOIN transaction ON transaction.company_id = c...	1 row(s) returned
5	10:29:11	SELECT company.company_name FROM company JOIN transaction ON transaction.company_id = company.id GROUP BY company.id...	1 row(s) returned
6	10:30:28	SELECT company.company_name FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction de...	1 row(s) returned

Ejercicio 3

Utilizando sólo subconsultas (sin utilizar JOIN):

a) Muestras todas las transacciones realizadas por las empresas de Alemania

Para la obtención de los resultados que nos solicitan he seleccionado todas las variables de la tabla transaction en la que he aplicado un filtro WHERE EXISTS, para que seleccione todos los elementos de la tabla transaction que estén incluido en la subconsulta, que lo que busca es el id de las compañías que pertenecen a Alemania y cuya id está en la tabla transaction.

The screenshot shows a SQL query editor with the following query:

```

34 # Muestras todas las transacciones realizadas por las empresas de Alemania
35 SELECT *
36 FROM transaction
37 WHERE EXISTS(
38     SELECT company.id
39     FROM company
40     WHERE company.country = "Germany" AND company.id = transaction.company_id)
41 ORDER BY transaction.company_id ;
42

```

The query is executed, and the results are displayed in a table with the following columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined. The table contains 13291 rows of transaction data.

Below the table, the 'Output' section shows the execution results:

```

12 11:37:01 SELECT company.company_name, count(transaction.id), CASE WHEN count(transaction.id) <= 400 THEN "<= 400" ELSE "> 400" ... 100 row(s) returned
13 11:43:38 SELECT * FROM transaction WHERE EXISTS( SELECT company.id FROM company WHERE company.country = "Germany" A... 13291 row(s) returned

```

- b) Lista las empresas que han realizado transacciones por una amount superior a la media de todas las transacciones.

En la presente consulta hemos seleccionado los nombres de las compañías de la tabla company que estén en el listado que resultante de la subconsulta en la que estamos aplicando a la tabla transacion el filtro de que la cantidad de venta de la transacción sea inferior a la media total que en este caso lo que realizado a través de otra subconsulta, he añadido los filtros para sacar las declinadas.

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query is as follows:

```

43 #Lista las empresas que han realizado transacciones por una amount superior a la media de todas las transacciones.
44
45 SELECT DISTINCT company.company_name
46 FROM company
47 WHERE EXISTS (
48     SELECT transaction.company_id
49     FROM transaction
50     WHERE transaction.amount > (SELECT avg(transaction.amount) FROM transaction) AND transaction.declined = 0 AND transaction.company_id = company.id
51 )
52 ORDER BY company.company_name;
  
```

The results pane displays a list of company names, including:

- A Institute
- Ac Fermentum Incorporated
- Ac Industries
- Ac Libero Inc.
- Alquam Erat Volutpat LLP
- Alquam Iaculis Lacus Corp.
- Alquam PC
- Alquet Diam Limited
- Alquet Sem Limited
- Alquet Vel Volutate Incorporated
- Amet Faucibus UI Foundation
- Amet Institute
- Amet Lorem LLP
- Amet Luctus Volutate Foundation
- Amet Nulla Donec Corporation
- Ante Iaculis Nec Foundation
- Arcu LLP
- At Associates
- At Pede Corp.
- Auctor Mauris Corp.
- Auctor Mauris Vel LLP
- Augue Foundation
- Convallis In Incorporated
- Cras Consulting
- Cras Vehicula Alquet Industries
- Dictum Eu Corp.
- Die Parturient Institute
- Dolor Vitae Limited
- Donec Fringilla PC
- Donec Ltd
- Dui Cras Associates
- Dui Quis Institute
- Egestas Nunc Sed Limited
- Eget Ipsum Ltd

The output pane at the bottom shows the execution of the query, indicating that 100 rows were returned.

#	Time	Action	Message	
9	10:34:45	SELECT DISTINCT company.company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transaction...	FROM transact... 100 row(s) returned	0
10	10:35:35	SELECT DISTINCT company.company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transact...	FROM transact... 100 row(s) returned	0

- c) *Eliminar del sistema las empresas que no tienen transacciones registradas, entrega el listado de estas empresas.*

No hay empresas sin transacciones: selecciono el nombre de la compañía que no exista el resultado obtenido en la subconsulta que selecciona todas las compañías que han realizado transacciones.

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query in Spanish, which translates to: "Eliminate from the system the companies that do not have registered transactions, deliver the list of these companies." The query is as follows:

```

54
55 # Eliminarán del sistema las empresas que no tienen transacciones registradas, entrega el listado de estas empresas.
56 SELECT company_name
57 FROM company
58 WHERE NOT EXISTS (
59     SELECT transaction.company_id
60     FROM transaction
61     WHERE company_id = transaction.company_id
62 )
63 ORDER BY company_name;
64

```

The bottom pane shows the execution results. The first table, 'company_name', contains one row: 'company 12'. Below this, the 'Output' pane shows the execution plan for the query, listing 13 steps with their respective durations and messages. The final step (13) shows that 0 rows were returned, indicating that all companies in the 'company' table have at least one transaction recorded in the 'transaction' table.

#	Time	Action	Message	Duration / Fetch
1	10:21:56	SELECT * FROM transaction WHERE EXISTS (SELECT id FROM company WHERE country = "Germany" AND company.id = tran...	13291 row(s) returned	0.094 sec / 0.046 sec
2	10:24:54	SELECT DISTINCT company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transac...	100 row(s) returned	0.078 sec / 0.000 sec
3	10:26:57	SELECT DISTINCT company.country FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction...	15 row(s) returned	0.421 sec / 0.000 sec
4	10:28:17	SELECT count(distinct company.country) as num_paises_con_ventas FROM company JOIN transaction ON transaction.company_id = c...	1 row(s) returned	0.437 sec / 0.000 sec
5	10:29:11	SELECT company_name FROM company JOIN transaction ON transaction.company_id = company.id GROUP BY company_id	1 row(s) returned	0.547 sec / 0.000 sec
6	10:30:28	SELECT company_name FROM company JOIN transaction ON transaction.company_id = company.id WHERE transaction.de...	1 row(s) returned	0.500 sec / 0.000 sec
7	10:31:57	SELECT * FROM transaction WHERE EXISTS (SELECT company_id FROM company WHERE company.country = "Germany" AN...	13291 row(s) returned	0.125 sec / 0.031 sec
8	10:33:42	SELECT DISTINCT company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transac...	100 row(s) returned	0.062 sec / 0.000 sec
9	10:34:45	SELECT DISTINCT company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transac...	100 row(s) returned	0.094 sec / 0.000 sec
10	10:35:35	SELECT DISTINCT company_name FROM company WHERE EXISTS (SELECT transaction.company_id FROM transac...	100 row(s) returned	0.172 sec / 0.000 sec
11	10:41:30	SELECT company_name FROM company WHERE id NOT IN (SELECT company_id FROM transaction) ORDER BY compa...	0 row(s) returned	0.109 sec / 0.000 sec
12	10:45:39	SELECT company_name FROM company WHERE NOT EXISTS (SELECT transaction.company_id FROM transaction ...	0 row(s) returned	0.000 sec / 0.000 sec
13	10:47:11	SELECT company_name FROM company WHERE NOT EXISTS (SELECT transaction.company_id FROM transaction ...	0 row(s) returned	0.000 sec / 0.000 sec

Nivel 2

Ejercicio 1

Identifica los cinco días en los que se generó la cantidad más grande de ingresos en la empresa por ventas. Muestra la fecha de cada transacción juntamente con el total de ventas.

He seleccionado la variable timestamp a la que he cambiado el formato con la función DATE_FORMAT que es la que modifica la fecha al formato francés que es con el que yo quiero trabajar para filtrar por fechas sin tener en cuenta la hora, por otro lado estoy aplicando la función de agregación SUM a la variable amount para saber el total de ventas por fecha realizado por GROUP BY. Al ordenar el total de ventas de mayor a menor y limitando la muestra de resultados a 5 registros obtenemos los 5 días con mayores ventas.

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query is as follows:

```

66 #NIVEL 2
67
68 #Ejercicio 1
69 #Identifica los cinco días en los que se generó la cantidad más grande de ingresos en la empresa por ventas. Muestra la fecha de cada transacción juntamente con el total de ve
70 • SELECT DATE_FORMAT(timestamp, "%d/%m/%Y") AS fecha, SUM(amount) AS total_ventas
71 FROM transaction
72 GROUP BY fecha
73 ORDER BY total_ventas DESC
74 LIMIT 5;
75

```

The results pane displays a table with two columns: fecha and total_ventas. The data is as follows:

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

The bottom pane shows the Action Output, which lists the execution of the query and the number of rows returned for each step.

Ejercicio 2

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

Para el cálculo de la media de ventas por país se ha seleccionado el nombre del país y aplicado la función de agregación para el cálculo de la media del importe de las transacciones redondeadas a 2 dígitos de la tabla resultante de la JOIN entre las dos tablas y se ha aplicado el filtro de declined=0 para que los cálculos se realicen únicamente con las transacciones ejecutadas y cobradas realmente.

Se ha realizado la agrupación de los países a través de GROUP BY y ordenado los datos a través de la media de ventas en orden descendente, de mayor a menor.

The screenshot shows a SQL IDE window titled "SPRINT2 Vanessa Perez Ramire...". The query editor contains the following SQL code:

```

77
78 # Quina és la mitjana de vendes per país? Presenta els resultats ordenats de major a menor mitjà.
79
80 SELECT company.country, Round(avg(transaction.amount),2) as media_ventas
81 FROM company
82 JOIN transaction
83 ON transaction.company_id = company.id
84 WHERE transaction.declined = 0
85 GROUP BY company.country
86 ORDER BY media_ventas DESC;
87

```

The "Result Grid" shows the following data:

country	media_ventas
Australia	265.54
United States	264.42
Belgium	260.97
Germany	260.83
Ireland	260.39
Spain	260.28
France	259.91
New Zealand	259.59
Norway	259.14
Netherlands	258.34
Italy	258.12
Canada	257.41
Sweden	257.39
United Kingdom	256.68
China	252.60

The "Output" pane shows the execution log:

#	Time	Action	Message
1	10:58:21	SELECT company.country, Round(avg(transaction.amount),2) as media_ventas FROM company JOIN transaction ON transaction.comp...	15 row(s) returned
2	10:58:33	SELECT company.country, Round(avg(transaction.amount),2) as media_ventas FROM company JOIN transaction ON transaction.comp...	15 row(s) returned

Ejercicio 3

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

a) Muestra el listado aplicando JOIN y subconsultas.

Del enunciado entiendo que he de dar las transacciones de las compañías que están en el mismo país que Non Institute, con lo que no han de aparecer las transacciones de la company Non Institute. Y solicitan todas las transacciones con lo que las declinadas también están incluidas en el listado.

He decidido seleccionar todos los datos de la tabla resultante de la join entre la tabla transaction y la tabla resultante de la subconsulta que proporciona los id de las company que son del mismo país que la empresa Non Institute y que además no son Non Institute.

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query is as follows:

```

89 # Ejercicio 3
90 #En tú empresa, se plantea un nuevo proyecto para lanzar algunas campañas publicitarias para hacer competencia a la compañía "Non Institute".
91 # Para esto, te solicitan la lista de todas las transacciones realizadas por empresas que están situadas en el mismo País que esta compañía.
92 #a) Muestra el listado aplicando JOIN y subconsultas.
93 SELECT *
94 FROM transaction
95 JOIN (
96     SELECT company_id
97     FROM company
98     WHERE company.country = (
99         SELECT company.country
100     FROM company
101     WHERE company.company_name = "Non Institute")
102     AND company.company_name <> "Non Institute"
103 )AS company_id_pais_non_institute
104 ON transaction.company_id = company_id_pais_non_institute.id;
  
```

The results grid displays a table with the following columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined, and id. The table contains 30 rows of transaction data.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined	id
00862984-CPA...	CS-7063	b-2246	2482	45.7666	4.33048	2015-07-30 12:12:42	486.44	0	b-2246
00872BA4-54A3...	CS-8475	b-2246	3894	55.6212	-3.7546	2017-10-26 22:08:26	414.06	0	b-2246
01F07981-074E...	CS-8700	b-2246	4119	55.856	-3.15783	2018-12-27 13:44:36	103.73	0	b-2246
023FCEB8-EF18...	CS-7816	b-2246	3235	46.3568	1.82755	2016-12-19 11:53:45	219.28	0	b-2246
026838EB-EF91...	CS-9471	b-2246	4890	42.1332	12.396	2017-01-10 21:09:29	326.87	0	b-2246
02CF29E-CEF2...	CS-9082	b-2246	4501	39.4662	-0.373246	2020-05-24 01:17:29	155.72	0	b-2246
02F468DC-426C...	CS-6913	b-2246	2332	52.175	19.3508	2023-03-17 16:36:27	305.35	0	b-2246
03068E3B-817B...	CS-5302	b-2246	721	51.9233	18.926	2021-12-02 23:06:02	339.58	0	b-2246
03478FE6-8EB5...	CS-7674	b-2246	3093	45.768	4.84271	2021-12-30 08:40:24	172.93	0	b-2246
03AEBD0E-DC97...	CS-6121	b-2246	1540	50.8113	10.3145	2018-11-11 11:28:49	114.77	0	b-2246
03CA36D3-88FF...	CS-8036	b-2246	3455	52.5178	13.4131	2017-02-25 15:38:21	440.27	0	b-2246
04494182-96D0...	CS-6791	b-2246	2210	41.9542	12.4607	2018-05-17 17:53:53	241.59	0	b-2246
045AACF6-FF85...	CS-5363	b-2246	782	39.2464	-7.90454	2018-08-09 22:12:54	188.58	0	b-2246
0489F0AC-95A4...	CS-7296	b-2246	2715	51.1647	10.7348	2021-03-30 04:29:22	205.45	0	b-2246
0490C36A-802...	CS-7539	b-2246	2958	46.4281	1.64603	2023-05-20 19:28:00	424.64	0	b-2246
04A0ABDE-FB25...	CS-4871	b-2246	290	52.0589	5.55327	2022-11-04 05:35:28	461.34	0	b-2246
0512193E-8639...	CS-6551	b-2246	2070	39.684	-8.57102	2024-07-08 02:51:38	82.15	0	b-2246
05531112-8559...	CS-8392	b-2246	3811	46.2212	1.99333	2023-02-02 11:21:26	132.86	0	b-2246
056A48CF-E178...	CS-4885	b-2246	304	51.8547	19.1249	2015-01-18 08:53:57	51.94	0	b-2246
0570EE9F-EADS...	CS-4895	b-2246	314	52.0796	4.30003	2023-04-01 16:24:43	209.79	0	b-2246
0584A0D8-8268...	CS-7082	b-2246	2501	46.138	2.37594	2020-01-15 17:35:10	122.73	0	b-2246
058BE441-6FEC...	CS-7991	b-2246	3410	40.3368	-3.26072	2023-08-05 18:57:23	199.04	0	b-2246
06002FEA-C7B0...	CS-6071	b-2246	1490	52.3959	19.5064	2020-12-03 20:04:59	114.09	0	b-2246
06249924-FD42...	CS-6676	b-2246	2095	55.0364	-3.92437	2015-03-17 23:10:56	535.77	0	b-2246
06708ECB-E017...	CS-7216	b-2246	2635	38.959	-8.06706	2023-02-26 04:35:06	137.81	0	b-2246
06D9EC8E-5AB7...	CS-4951	b-2246	370	39.2632	-8.63151	2015-04-13 02:41:15	9.24	0	b-2246
070JD520-8F27...	CS-8237	b-2246	3656	41.7315	12.1951	2016-05-15 11:45:18	327.91	0	b-2246
081F8CD2-F002...	CSU-3470	b-2246	199	55.4537	-3.47315	2023-02-03 16:52:03	327.43	0	b-2246

The output section shows the execution of the query, indicating that 12233 row(s) were returned.

b) Muestra el listado sólo aplicando subconsultas.

Del enunciado entiendo que he de dar las transacciones de las compañías que están en el mismo país que Non Institute, con lo que no han de aparecer las transacciones de la company Non Institute. Y solicitan todas las transacciones con lo que las declinadas también están incluidas en el listado.

Para obtener los resultados sin utilizar JOIN, he seleccionado todos los campos de la tabla transaction que estén incluidos en la tabla resultante de la subconsulta que filtra los registros de la tabla company para que el resultado que dé sean las compañías que estén en el mismo país que la empresa Non Institute y además tiene el filtro de en ese listado no aparezca la compañía Non institutes y que los id que identifican a la compañía estén en las dos tablas.

106 #b) Muestra el listado sólo aplicando subconsultas.

107 • SELECT *

108 FROM transaction

109 WHERE EXISTS (SELECT *

110 FROM company

111 WHERE company.country = (

112 SELECT company.country

113 FROM company

114 WHERE company.company_name = "Non Institute")

115 AND company.company_name <> "Non Institute" AND company.id = transaction.company_id)

116 ;

117

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
008629B4-C9A9...	Cc5-7063	b-2246	2482	45.7666	4.83048	2015-07-30 12:12:42	486.44	0
00872BA4-54A3...	Cc5-8475	b-2246	3894	55.6212	-3.7546	2017-10-26 22:08:26	414.06	0
01F075B1-07AE...	Cc5-8700	b-2246	4119	55.856	-3.15783	2018-01-27 13:44:36	103.73	0
023FFCE8-E618...	Cc5-7816	b-2246	3235	46.3568	1.82755	2016-12-19 11:53:45	219.28	0
026838EB-EF91...	Cc5-9471	b-2246	4890	42.1332	12.396	2017-01-10 21:09:29	326.87	0
02C2F29E-CEP2...	Cc5-9082	b-2246	4501	39.4662	-0.373246	2020-05-24 01:17:29	155.72	0
02F468DC-426C...	Cc5-6913	b-2246	2332	52.175	19.3508	2023-03-17 16:36:27	305.35	0
0306BE3B-8176...	Cc5-5302	b-2246	721	51.9233	18.926	2021-12-02 23:06:02	339.58	0
0347BF6E-4EB5...	Cc5-7674	b-2246	3093	45.768	4.84271	2021-12-30 08:40:24	172.93	0
03AEBD0E-DC97...	Cc5-6121	b-2246	1540	50.8113	10.3145	2018-11-11 11:28:49	114.77	0
03CA36D3-88FF...	Cc5-8036	b-2246	3455	52.5178	13.4131	2017-02-25 15:38:21	440.27	0
04494182-96D0...	Cc5-6791	b-2246	2210	41.9542	12.4607	2018-05-17 17:53:53	241.59	0
045AACF6-FF85...	Cc5-5363	b-2246	782	39.2464	-7.90454	2018-08-09 22:12:54	188.58	0
0489FDAC-86A4...	Cc5-7296	b-2246	2715	51.1647	10.7348	2021-03-30 04:29:22	205.45	0
0490C36A-4B02...	Cc5-7539	b-2246	2958	46.4281	1.64603	2023-05-20 19:28:00	424.64	0
04A0ABDE-FB25...	Cc5-4871	b-2246	290	52.0589	5.55327	2022-11-04 05:35:28	461.34	0
0512193E-8639...	Cc5-6651	b-2246	2070	39.684	-8.57102	2024-07-08 02:51:38	82.15	0
05523112-8559...	Cc5-8392	b-2246	3811	46.2212	1.99333	2023-02-02 11:21:26	132.86	0
056A48CF-E178...	Cc5-4885	b-2246	304	51.8547	19.1249	2015-01-18 08:53:57	51.94	0
0570EE9F-EAD5...	Cc5-4895	b-2246	314	52.0796	4.30003	2023-04-01 16:24:43	209.79	0

transaction 5 x

Output

Action Output

#	Time	Action	Message
1	11:07:26	SELECT * FROM transaction JOIN (SELECT company.id FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
2	11:13:23	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company....	100000 row(s) returned
3	11:17:08	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	Error Code: 1064. You have an error in your SQL syntax; check the manual that correspond
4	11:17:23	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	1543 row(s) returned
5	11:17:31	SELECT * FROM transaction WHERE transaction.company_id IN (SELECT id FROM company WHERE country = (SELECT country FR...	12233 row(s) returned
6	11:17:57	SELECT * FROM transaction JOIN (SELECT company.id FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
7	11:18:02	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	1543 row(s) returned
8	11:32:55	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
9	11:34:37	SELECT company.company_name, company.phone, company.country, DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") AS tech...	8 row(s) returned
10	11:34:43	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned

Nivel 3

Ejercicio 1

Presenta el nombre, teléfono, país, fecha y amount, de las empresas que realizaron transacciones con un valor comprendido entre 350 y 400 euros y en alguna de las siguientes fechas: 29/04/2015, 20/07/2018 y 13/03/2024. Ordena los resultados de mayor a menor cantidad.

En este ejercicio que solicitado la creación de una tabla a través del SELECT que contenga las variables que solicitan, para ello he realizado una JOIN ya que los datos solicitados son de las dos tablas de la base de datos. Por otro lado, he transformado el formato de la variable timestamp al formato de fecha que necesito para hacer los filtros por fechas que se nos solicitan.

En los filtros he utilizado los operadores de lógica BETWEEN e IN. He incluido el filtro declined =0 para eliminar de los cálculos los registros decinados.

NIVEL 3

EJERCICIO 1

Presenta el nombre, teléfono, país, fecha y amount, de las empresas que realizaron transacciones con un valor comprendido entre 350 y 400 euros y en alguna de las siguientes fechas: 29/04/2015, 20/07/2018 y 13/03/2024. Ordena los resultados de mayor a menor cantidad.

SELECT

company_name, company.phone, company.country, DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") AS fecha, transaction.amount

FROM company

JOIN transaction

ON company.id = transaction.company_id

WHERE transaction.declined = 0 AND DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") IN ("29/04/2015", "20/07/2018", "13/03/2024") AND transaction.amount BETWEEN 350 AND 400

ORDER BY transaction.amount DESC;

Result Grid

company_name	phone	country	fecha	amount
Aliquam PC	01 45 73 52 16	Germany	13/03/2024	399.84
Auctor Mauris Vel LLP	08 09 28 74 14	United States	20/07/2018	399.51
At Pede Corp.	06 14 48 33 15	Italy	29/04/2015	390.69
Aliquam PC	01 45 73 52 16	Germany	13/03/2024	386.29
Oris Adipiscing Limited	03 18 00 77 81	United Kingdom	20/07/2018	373.71
Fringilla LLC	08 29 15 93 57	New Zealand	29/04/2015	367.62
Pede Cum Ltd	07 62 26 48 38	Norway	20/07/2018	356.87
Auctor Mauris Vel LLP	08 09 28 74 14	United States	13/03/2024	353.75

Result 6

Output

Action Output

#	Time	Action	Message
1	11:07:26	SELECT * FROM transaction JOIN (SELECT company_id FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
2	11:13:23	SELECT * FROM transaction WHERE EXISTS (SELECT company_id FROM company WHERE company.country = (SELECT company...	100000 row(s) returned
3	11:17:08	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version fo...
4	11:17:23	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	1543 row(s) returned
5	11:17:31	SELECT * FROM transaction WHERE transaction.company_id IN (SELECT id FROM company WHERE country = (SELECT country FR...	12233 row(s) returned
6	11:17:57	SELECT * FROM transaction JOIN (SELECT company_id FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
7	11:18:02	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	1543 row(s) returned
8	11:32:55	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
9	11:34:37	SELECT company_name, company.phone, company.country, DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") AS fech...	8 row(s) returned
10	11:34:43	SELECT * FROM transaction WHERE EXISTS (SELECT * FROM company WHERE company.country = (SELECT company.country FR...	12233 row(s) returned
11	11:35:39	SELECT company_name, company.phone, company.country, DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") AS fech...	8 row(s) returned

Ejercicio 2

Necesitamos la optimización y la asignación de los recursos y dependerá de la capacidad operativa que se requiera, para esto te solicitan la información sobre la cantidad de transacciones que realizan las empresas, pero el departamento de rrhh es exigente y quiere un listado de las empresas donde se especifiquen si tienen más de 400 transacciones o menos

En este caso, he incluido las transacciones anuladas a que las considero como una acción realizada aunque no se haya aprobado, con lo que no aplico filtro de declined = 0.

Para poder obtener la información que nos han solicitado he realizado la consulta que solicita la a través de SELECT el nombre de la compañía y el número de las transacciones a través de la función de agregación COUNT, y he creado una columna en la que se indica si el número de transacciones está por encima de 400 o no, lo he realizado con CASE.

Se han agrupado los datos por el nombre de la compañía y ordenado los mismo por el número de transacciones en orden descendente de mayor a menor.

The screenshot shows a SQL query in a text editor and its results in a table. The query is as follows:

```

131 #Necesitamos la optimización y la asignación de los recursos y dependerá de la capacidad operativa que se requiera, para esto te solicitan la información sobre
132 #la cantidad de transacciones que realizan las empresas, pero el departamento de rrhh es exigente y quiere un listado de las empresas donde se especifiquen si tienen
133 #más de 400 transacciones o menos.
134
135 • SELECT company.company_name, count(transaction.id),
136     CASE
137     WHEN count(transaction.id) <= 400 THEN "<= 400"
138     ELSE "> 400"
139     END AS grupo_transacciones
140 FROM transaction
141 JOIN company
142 ON company.id = transaction.company_id
143 GROUP BY company.company_name
144 ORDER BY count(transaction.id) DESC;
145
146
147

```

The results are shown in a table with the following columns: company_name, count(transaction.id), and grupo_transacciones. The table contains 30 rows of data, sorted by the count of transactions in descending order.

company_name	count(transaction.id)	grupo_transacciones
Corvallis In Incorporated	1514	> 400
Ac Libero Inc.	1513	> 400
Amet Nulla Donec Corporation	1511	> 400
Dui Cras Associates	1508	> 400
Egestas Nunc Sed Limited	1504	> 400
Amet Fausibus UI Foundation	1494	> 400
Non Magna LLC	1489	> 400
Maecenas Malesuada Fringilla Inc.	1486	> 400
Quam A Fella Industries	1481	> 400
Turpis Consequy	1481	> 400
Non Justo Corp.	1479	> 400
Placerat LLP	1479	> 400
Gravida Sagittis LLP	1475	> 400
Lorem Ipsum Dolor Corp.	1465	> 400
Ut Senper Foundation	1460	> 400
Tempor Diam Institute	1454	> 400
Nulla Integer Vulputate Corp.	1449	> 400
Aliquet Sem Limited	1444	> 400
Nunc Ac PC	1440	> 400
Quisque Libero LLC	485	> 400
Cras Consulting	474	> 400
Magna Incorporated	474	> 400
Ante Inaculis Nec Foundation	472	> 400
Dictum Eu Corp.	472	> 400
Tortor Nunc Commodo Company	470	> 400
Tincidunt Associates	466	> 400

The bottom of the screenshot shows the 'Output' tab with the following message:

```

11 11:35:39 SELECT company.company_name, company.phone, company.country, DATE_FORMAT(transaction.timestamp, "%d/%m/%Y") AS fec... 8 row(s) returned
12 11:37:01 SELECT company.company_name, count(transaction.id), CASE WHEN count(transaction.id) <= 400 THEN "<= 400" ELSE "> 400" ... 100 row(s) returned

```