

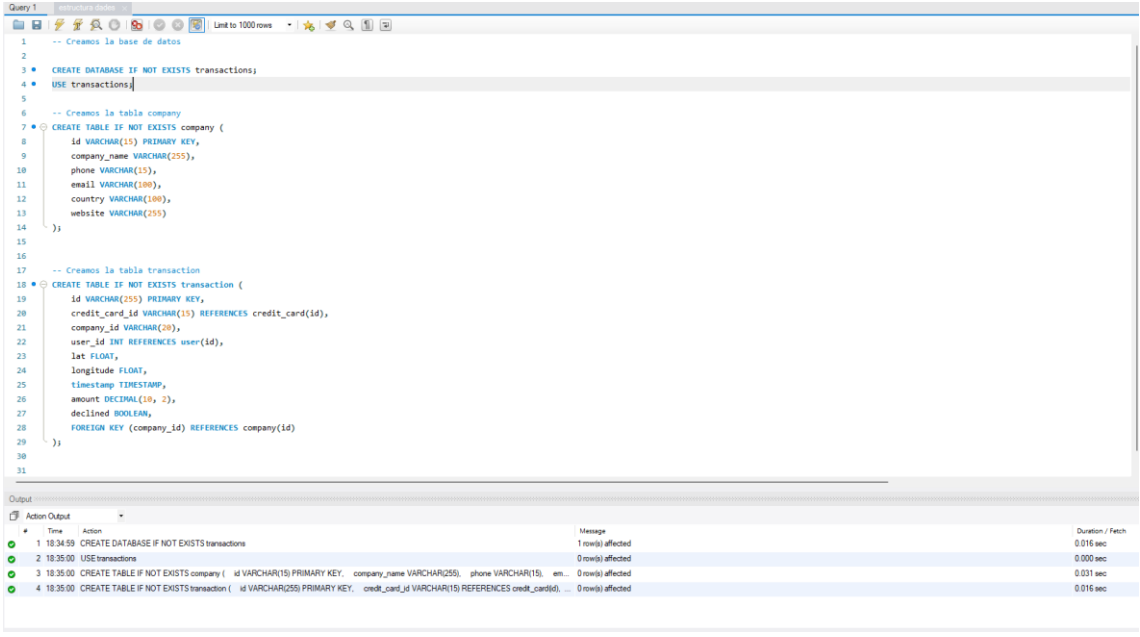
TAREA S21: NOCIONES BÁSICAS DE SQL

Nivel 1

Ejercicio 1

A partir de los documentos adjuntos (estructura_dades y dades_introduir), importa las dos tablas. Muestra las características principales del esquema creado y explica las diferentes tablas y variables que existen. Asegúrate de incluir un diagrama que ilustre la relación entre las diferentes tablas y variables.

Primero creamos las tablas según se ve en el snippet siguiente



```
1 -- Creamos la base de datos
2
3 CREATE DATABASE IF NOT EXISTS transactions;
4 USE transactions;
5
6 -- Creamos la tabla company
7 CREATE TABLE IF NOT EXISTS company (
8   id VARCHAR(15) PRIMARY KEY,
9   company_name VARCHAR(255),
10  phone VARCHAR(15),
11  email VARCHAR(180),
12  country VARCHAR(180),
13  website VARCHAR(255)
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(20),
21   user_id INT REFERENCES user(id),
22   lat FLOAT,
23   longitude FLOAT,
24   timeStamp TIMESTAMP,
25   amount DECIMAL(10, 2),
26   declined BOOLEAN,
27   FOREIGN KEY (company_id) REFERENCES company(id)
28 );
29
30
31
```

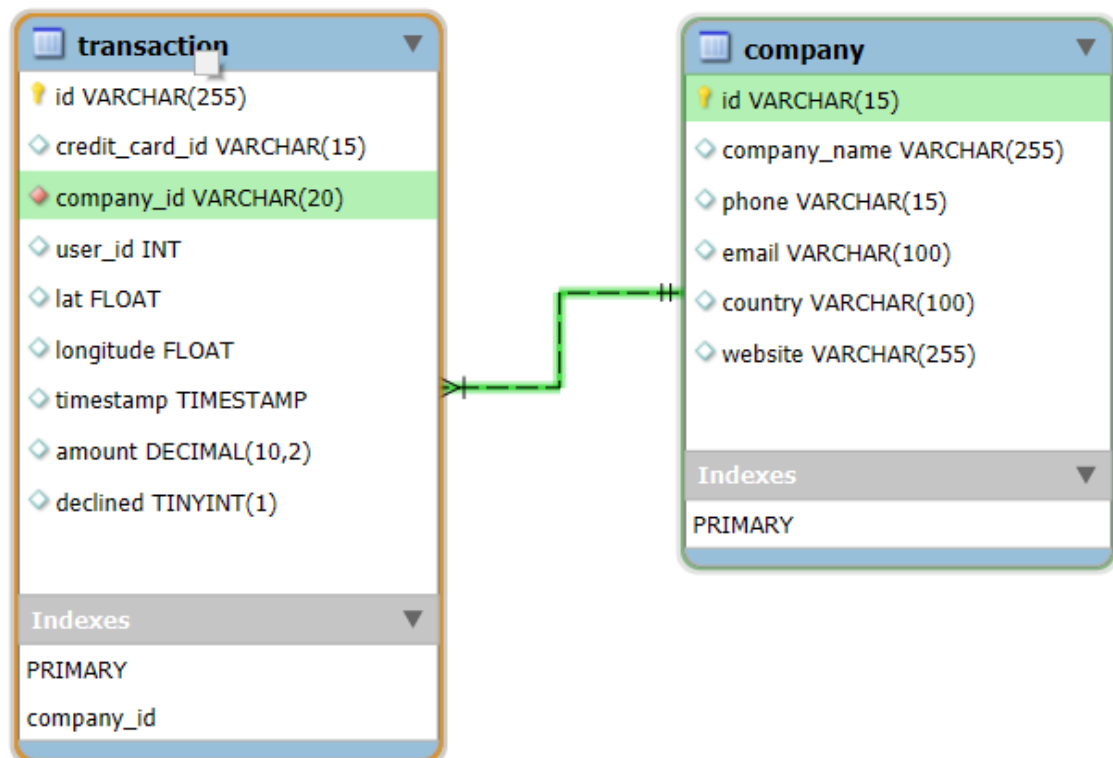
#	Time	Action	Message	Duration / Fetch
1	10:34:59	CREATE DATABASE IF NOT EXISTS transactions	1 row(s) affected	0.016 sec
2	10:35:00	USE transactions	0 row(s) affected	0.000 sec
3	10:35:00	CREATE TABLE IF NOT EXISTS company (id VARCHAR(15) PRIMARY KEY, company_name VARCHAR(255), phone VARCHAR(15), em...	0 row(s) affected	0.031 sec
4	10:35:00	CREATE TABLE IF NOT EXISTS transaction (id VARCHAR(255) PRIMARY KEY, credit_card_id VARCHAR(15) REFERENCES credit_card(id), ...	0 row(s) affected	0.016 sec

Cargamos los datos en las tablas anteriormente creadas y según el siguiente snippet.

The screenshot displays a SQL database environment. On the left, a sidebar shows the 'transacciones' schema with a table 'transaction'. The main area shows the table structure with columns: id (VARCHAR(255)), credit_card_id (VARCHAR(15)), company_id (VARCHAR(20)), user_id (INT), lat (FLOAT), longitude (FLOAT), timestamp (TIMESTAMP), amount (DECIMAL(10,2)), and declined (TINYINT(1)). Below the schema, a list of transactions is shown, each with a timestamp and a message. The output window at the bottom shows the execution of an INSERT statement, indicating that 1 row was affected.

➤ ESQUEMA DE LA BASE DE DATOS:

El esquema de la BBDD “*dades_introduir*” es el siguiente:



➤ **COMPOSICIÓN DE LA BASE DE DATOS**

La BBDD llamada “*dades_introduir*” consta de **2 tablas** llamadas:

1. “*transaction*”:

Esta tabla facilita información de las transacciones producidas en la empresa.

El número total de transacciones es de 587.

Cada transacción se corresponde con un registro de la tabla “*transaction*” de “*id*” diferente, siendo este id la primary Key de la tabla “*transaction*” y una variable de tipo VARCHAR.

Cada registro contiene además otros datos de la transacción y que son:

- **Número de tarjeta de crédito** con la que se realizó la transacción que corresponde a la variable denominada “*credit_card_id*”, de tipo Varchar y longitud 15 caracteres.
- **Número de cliente** de la empresa que hizo la transacción, que corresponde a la variable “*company_id*”, de tipo Varchar y longitud 20 caracteres.
- **Número de usuario de la persona** que realizó la transacción, que corresponde a la variable “*user_id*”, de tipo entero.
- **Latitud**: Ubicación de la empresa que realizó la transacción, que corresponde a la variable “*lat*”, de tipo flotante.
- **Longitud**: Ubicación de la empresa que realizó la transacción, que corresponde a la variable “*longitude*”, de tipo flotante.
- **Momento** en que se produjo la transacción: Día, fecha y hora, que corresponde a la variable “*timestamp*”, de tipo timestamp.
- **Cantidad dineraria de la transacción**, sin moneda especificada, que corresponde a la variable “*amount*”, de tipo decimal 10,2 (con 2 cifras decimales).”.
- **Estado final de la transacción**, que corresponde a la variable “*declined*”, el cual informa de si la transacción resulto aceptada o no. Teniendo esta variable 2 únicos valores 0 y 1. Podríamos intuir que el valor 0 es = a False y por tanto no declinada (=aceptada) y el valor 1 seria igual a True y por tanto a transacción si declinada. Se trata de una variable de tipo tinyint (1)

2. “*company*”:

Facilita información sobre la empresa que realizó la transacción (o compra).

El número total de empresas es de 100.

Cada empresa se corresponde con un registro de la tabla “*company*” de “*id*” diferente, siendo este “*id*” el primary Key de la tabla “*company*”. Esta

id de la tabla “*company*” a su vez actúa de clave foránea (Foreign Key) de la (“*company_id*”) en la tabla *transaction*.

- **Nombre de la compañía** que realiza la transacción, que corresponde a la variable “*company_name*”.
- **Número de teléfono de la compañía** que realiza la transacción, que corresponde a la variable “*phone*”.
- **Email de la compañía** que realiza la transacción, que corresponde a la variable “*email*”.
- **País de la compañía** que realiza la transacción, que corresponde a la variable “*country*”.
- **Página web de la compañía** que realiza la transacción, que corresponde a la variable “*website*”.

➤ **UNION DE TABLAS EN LA BBDD**

Ambas tablas están unidas mediante relación **n a 1** través de las variables “*company_id*” (de la tabla “*transaction*”) e “*id*” (de la tabla “*company*”) como vemos en el esquema a continuación.

➤ **OBSERVACIONES:**

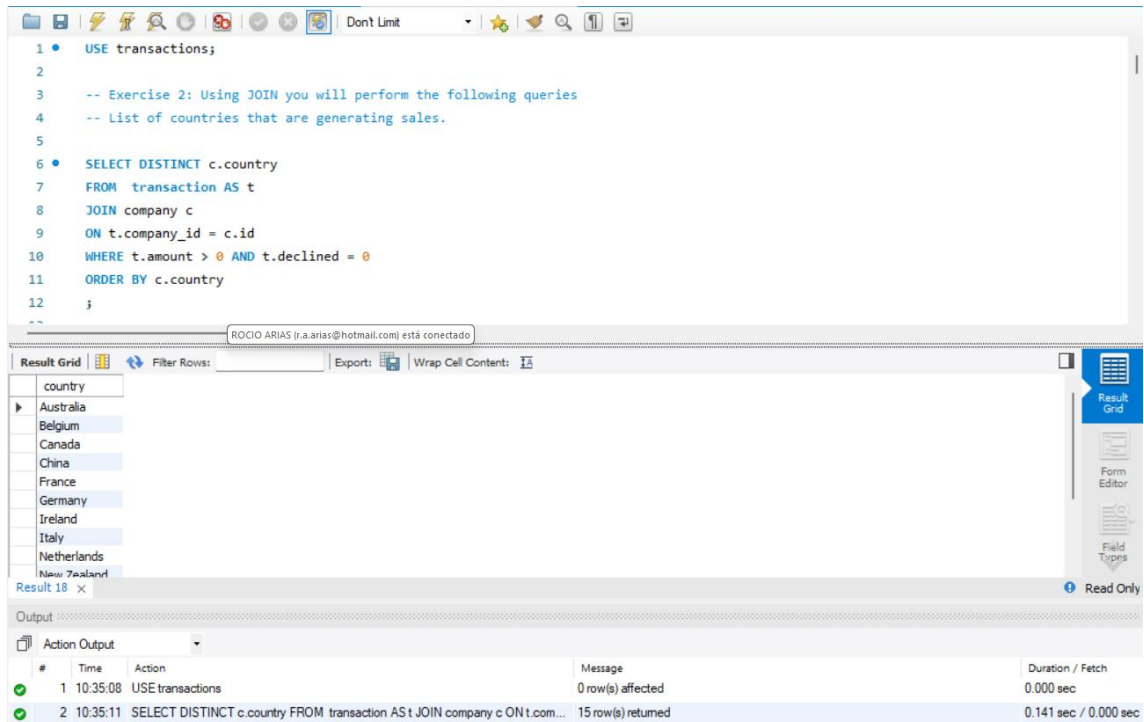
- La variable “*id*” de la tabla “*company*” es de longitud 15 mientras que la variable “*company_id*” de la tabla “*transaction*” es de longitud 255.

Ejercicio 2

Utilizando JOIN realizarás las siguientes consultas:

- Listado de los países que están generando ventas.
 - Desde cuántos países se generan las ventas.
 - Identifica la compañía con la mayor media de ventas
-

- Caso 1: Listado de los países que están generando ventas.



Criterio considerado:

“países que generan ventas” son aquellos en los que se cumplen las 2 siguientes premisas:

1. La cantidad facturada es positiva (amount > 0)
2. La transacción ha tenido éxito (declined = 0)

Explicación de la estrategia de código:

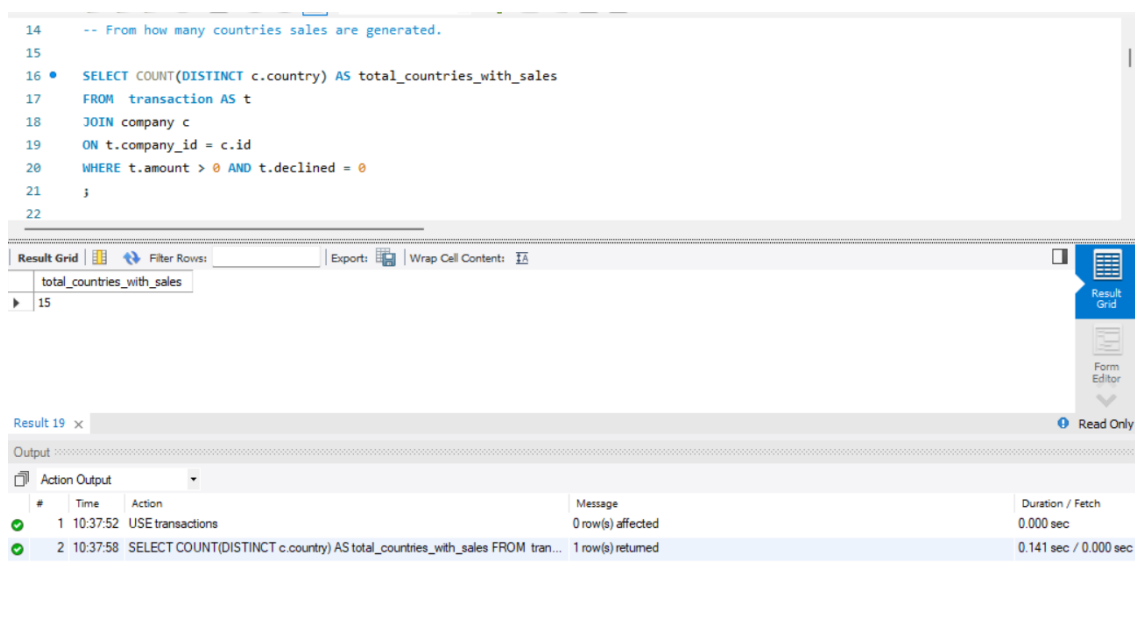
1. FROM transaction AS t JOIN company c ON t.company_id = c.id: El punto de partida es unir (JOIN) la tabla de transacciones (transaction) con la tabla de compañías (company). La unión se realiza comparando company_id de la tabla de transacciones con el id de la tabla de compañías. Este paso es esencial para poder asociar los datos geográficos de una compañía (el país) con sus registros de transacciones.

2. WHERE t.amount > 0 AND t.declined = 0: Se aplica un filtro para seleccionar únicamente las transacciones que representan ventas válidas y completadas. La condición t.amount > 0 excluye cualquier transacción sin valor monetario o devoluciones, mientras que t.declined = 0 asegura que solo se incluyan las transacciones que fueron aprobadas y procesadas con éxito.

3. **SELECT DISTINCT c.country:** Se selecciona la columna country de la tabla de compañías. El uso de DISTINCT es crucial aquí, ya que elimina los nombres de países duplicados del resultado. Esto garantiza que cada país que ha tenido al menos una venta válida aparezca solo una vez en la lista final.

4. **ORDER BY c.country:** Se ordenan los resultados alfabéticamente por el nombre del país. Esto no afecta a qué países se muestran, pero presenta la lista final de una manera organizada y fácil de leer.

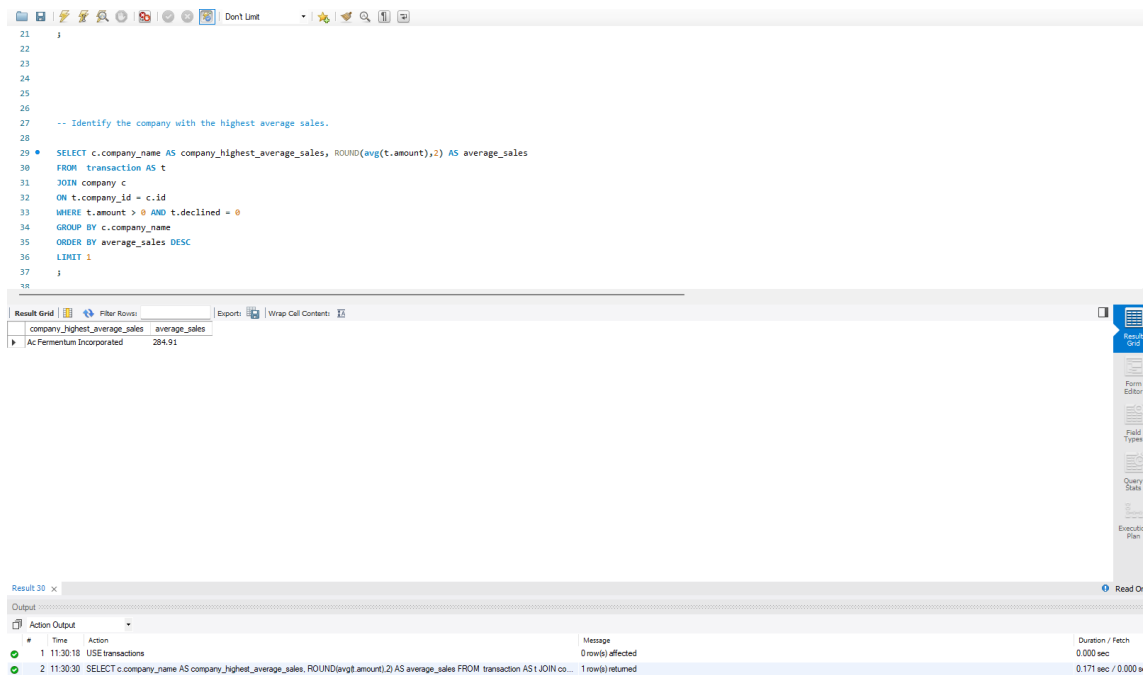
- Caso 2: Desde cuántos países se generan las ventas.



Explicacion de codigo

1. **FROM transaction AS t JOIN company c ON t.company_id = c.id:** Al igual que en el caso anterior, se une la tabla de transacciones con la de compañías para poder acceder a la información del país de cada transacción.
2. **WHERE t.amount > 0 AND t.declined = 0:** Se utiliza el mismo filtro para asegurar que el análisis se basa únicamente en ventas reales y exitosas, excluyendo transacciones fallidas o de valor cero/negativo.
3. **SELECT COUNT(DISTINCT c.country):** Aquí reside la lógica principal. La función COUNT() se aplica sobre una lista de países únicos (DISTINCT c.country). Primero, el motor de la base de datos identifica todos los países únicos que cumplen con la condición WHERE, y luego COUNT() cuenta cuántos elementos hay en esa lista. Esto proporciona el número exacto de países distintos que generan ventas.

○ Caso 3: Identifica la compañía con la mayor media de ventas.



```
21 ;
22
23
24
25
26
27 -- Identify the company with the highest average sales.
28
29 • SELECT c.company_name AS company_highest_average_sales, ROUND(avg(t.amount),2) AS average_sales
30 FROM transaction AS t
31 JOIN company c
32 ON t.company_id = c.id
33 WHERE t.amount > 0 AND t.declined = 0
34 GROUP BY c.company_name
35 ORDER BY average_sales DESC
36 LIMIT 1
37 ;
```

company_highest_average_sales	average_sales
Ac Fermentum Incorporated	294.91

#	Time	Action	Message	Duration / Fetch
1	11:30:18	USE transaction	0 row(s) affected	0.000 sec
2	11:30:30	SELECT c.company_name AS company_highest_average_sales, ROUND(avg(t.amount),2) AS average_sales FROM transaction AS t JOIN co...	1 row(s) returned	0.171 sec / 0.000

Justificación del código

1. FROM transaction AS t JOIN company c ON t.company_id = c.id: Se realiza la unión de las tablas transaction y company para poder vincular las ventas con los nombres de las compañías.
2. WHERE t.amount > 0 AND t.declined = 0: Se filtran las transacciones para incluir solo ventas válidas, que serán la base para calcular el promedio.
3. GROUP BY c.company_name: Este es un paso clave. Agrupa todas las filas de transacciones por el nombre de la compañía. Esto permite que la función de agregación AVG() que se usa a continuación se calcule de forma independiente para cada compañía.
4. SELECT c.company_name, ROUND(avg(t.amount),2) AS average_sales: Para cada grupo de compañía creado en el paso anterior, se calcula el promedio (AVG) del monto (t.amount) de sus transacciones. La función ROUND() se utiliza para redondear el resultado a dos decimales, mejorando la presentación. Se seleccionan tanto el nombre de la compañía como su promedio de ventas calculado.
5. ORDER BY average_sales DESC: Una vez que se ha calculado el promedio de ventas para todas las compañías, los resultados se ordenan de forma descendente (DESC) según este valor. Esto sitúa a la compañía con el promedio más alto en la primera posición.

6. LIMIT 1: Finalmente, esta cláusula restringe la salida del resultado a una sola fila: la primera. Debido al ordenamiento del paso anterior, esta fila corresponde precisamente a la compañía con el promedio de ventas más alto.

Ejercicio 3

Utilizando sólo subconsultas (sin utilizar JOIN):

- Muestra todas las transacciones realizadas por empresas de Alemania.
- Lista las empresas que han realizado transacciones por un amount superior a la media de todas las transacciones.
- Eliminarán del sistema las empresas que no tienen transacciones registradas, entrega el listado de estas empresas.

- Caso 1: Muestra todas las transacciones realizadas por empresas de Alemania

The screenshot shows a SQL IDE interface. The top pane contains a SQL query for Exercise 3, which uses a subquery to filter transactions by company country. The bottom pane displays the results of the query in a table format.

```

37 -- Exercise 3: Using only subqueries (without using JOIN)
38 -- Show all transactions made by companies from Germany.
39
40 • SELECT *
41 FROM transaction AS t
42 WHERE t.company_id IN (SELECT id
43                        FROM company
44                        WHERE company.country = 'Germany')
45 ;
46

```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
00138D38-206D-4C03-94B7-63A2676EB9B4	CcS-4899	b-2222	318	41.3781	12.447	2020-03-25 10:43:43	426.36	0
0013C1B6-3B84-4D6C-8154-E2B3FEBCA8E9	CcS-5070	b-2222	489	41.3814	2.18176	2020-12-17 18:15:37	316.90	0
00201A11-2E62-44C4-941D-198FC8D877F0	CcU-3512	b-2222	193	55.5704	-3.65129	2021-01-22 23:44:27	453.04	0
00235618-0A5C-4D49-9DCB-83A9405D8923	CcS-8137	b-2222	3556	59.8421	18.729	2020-09-09 15:43:19	263.14	0
005A5A7B-1F1A-4B6C-9B15-1625A78C9C38	CcS-8998	b-2222	4417	41.1591	-8.63905	2024-05-15 09:10:11	442.01	0
00687139-48B2-4FFA-8E73-820376F04AB4	CcS-4870	b-2222	289	51.1966	10.4669	2019-03-09 19:37:49	524.84	0
0074F4D0-32F1-4827-8758-55896314623A	CcS-8081	b-2222	3500	39.7016	-8.50325	2016-12-26 23:06:57	491.90	0
00AAB9CD-39D6-4DCB-8A1D-13BE73DC90A9	CcS-6797	b-2222	2216	55.7652	-3.76245	2021-04-25 03:06:59	167.15	0
00BE09D4-6920-47D8-ABE8-325E2269829D	CcS-4983	b-2222	402	38.708	-9.12993	2019-02-27 15:25:16	141.66	0
00DA0383-E048-4577-8ED1-3C56C258FF2F	CcS-9223	b-2222	4642	51.1742	10.2027	2019-03-21 11:47:34	325.62	0
00DD11DE-ED01-4BBD-93A0-174D183A59DF	CcS-7681	b-2222	3100	45.7565	4.83109	2024-01-28 18:20:49	242.53	0
01449CE0-98E9-4DE5-9810-728C68A00E6F	CcS-5424	b-2222	843	47.0163	2.26064	2024-02-17 19:37:14	451.71	0

transaction 21 x

Output

#	Time	Action	Message	Duration / Fetch
1	10:44:46	USE transactions	0 row(s) affected	0.000 sec
2	10:45:06	SELECT * FROM transaction AS t WHERE t.company_id IN (SELECT id FROM c...	13291 row(s) returned	0.000 sec / 0.047 sec

Justificación código

1. Subconsulta Interna: (SELECT id FROM company WHERE company.country = 'Germany')

- Propósito: El primer paso que ejecuta el sistema es resolver esta subconsulta. Su objetivo es actuar como un filtro dinámico.
 - Lógica Técnica: La subconsulta escanea la tabla company y crea una lista temporal en memoria que contiene únicamente los id de aquellas compañías cuyo country es 'Germany'. El resultado es una lista de identificadores (ej: [12, 45, 83, ...]).
2. Consulta Externa: `SELECT * FROM transaction AS t WHERE t.company_id IN (...)`
- Propósito: La consulta principal se encarga de seleccionar los datos finales de la tabla transaction.
 - Lógica Técnica: Recorre cada fila de la tabla transaction. Para cada transacción, la cláusula `WHERE t.company_id IN (...)` comprueba si el `company_id` de esa transacción existe dentro de la lista de IDs generada por la subconsulta (la lista de IDs de empresas alemanas). Si hay una coincidencia, la transacción completa (`SELECT *`) se incluye en el resultado final. Este método es una alternativa al JOIN para filtrar registros de una tabla basándose en condiciones de otra.
 - Caso 2: Lista las empresas que han realizado transacciones por un amount superior a la media de todas las transacciones.

The screenshot shows a SQL IDE interface. The main window displays a SQL query with line numbers 49 to 65. The query is as follows:

```

49 SELECT DISTINCT
50   c.company_name
51 FROM
52   company AS c
53 WHERE
54   c.id IN (
55     SELECT
56       t.company_id
57     FROM
58       transaction AS t
59     WHERE
60       t.amount > (SELECT
61                     AVG(t.amount)
62                   FROM
63                     transaction AS t
64                 )
65   );

```

Below the query editor, the 'Result Grid' is visible, showing a table with the following data:

company_name
Ac Fermentum Incorporated
Magna A Neque Industries
Fusce Corp.
Convallis In Incorporated
Ante Taculis Nec Foundation

At the bottom, the 'Output' window shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	10:46:13	USE transactions	0 row(s) affected	0.000 sec
2	10:46:54	SELECT DISTINCT c.company_name FROM company AS c WHERE c.id...	100 row(s) returned	0.047 sec / 0.000 sec

Justificación del Código:

Subconsulta más Interna: (SELECT AVG(t.amount) FROM transaction AS t)

- Propósito: Calcular un único valor de referencia.
- Lógica Técnica: Esta subconsulta se ejecuta primero. Calcula el valor promedio (AVG) de la columna amount sobre *todas* las transacciones de la tabla transaction. El resultado es un único valor numérico (un escalar), que representa el promedio general de ventas.

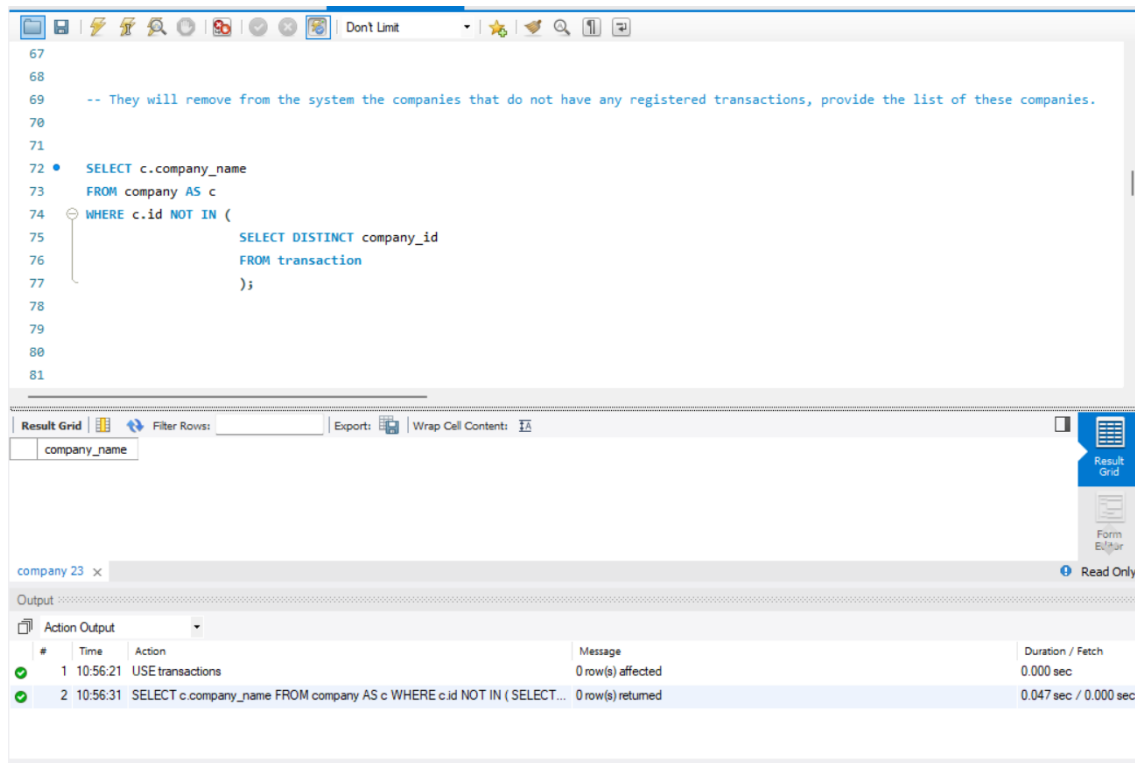
2. Subconsulta Intermedia: (SELECT t.company_id FROM transaction AS t WHERE t.amount > ...)

- Propósito: Identificar las compañías que cumplen el criterio.
- Lógica Técnica: Esta consulta utiliza el valor promedio calculado en el paso anterior. Escanea la tabla transaction y selecciona el company_id de cada transacción cuyo amount es estrictamente mayor que el promedio general. El resultado es una lista de company_ids, que puede contener duplicados si una misma empresa tuvo varias transacciones por encima del promedio.

3. Consulta Externa: SELECT DISTINCT c.company_name FROM company AS c WHERE c.id IN (...)

- Propósito: Obtener los nombres únicos de las empresas identificadas.
- Lógica Técnica: La consulta principal filtra la tabla company. La cláusula WHERE c.id IN (...) selecciona solo aquellas compañías cuyo id está presente en la lista generada por la subconsulta intermedia. Finalmente, SELECT DISTINCT c.company_name se utiliza para asegurar que cada nombre de empresa aparezca solo una vez en el resultado final, eliminando los duplicados que podrían surgir si una empresa tuvo múltiples transacciones que cumplieran la condición.

- Caso 3: Lista las empresas que han realizado transacciones por un amount superior a la media de todas las transacciones.



Justificación del Código:

Subconsulta Interna: (SELECT DISTINCT company_id FROM transaction)

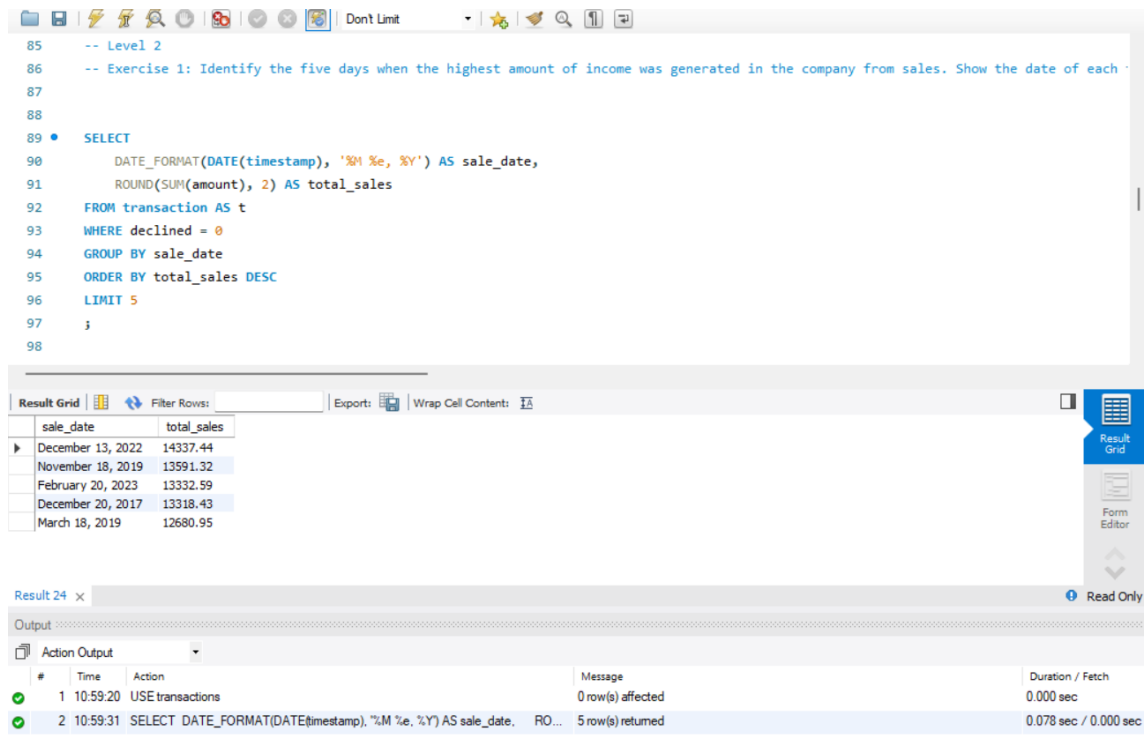
- Propósito: Crear una lista de todas las empresas que sí tienen actividad.
 - Lógica Técnica: La subconsulta escanea la tabla transaction y genera una lista con todos los company_id únicos que aparecen en ella. DISTINCT es usado para optimizar, creando una lista sin duplicados de todos los IDs de empresas que han realizado al menos una transacción.
2. Consulta Externa: SELECT c.company_name FROM company AS c WHERE c.id NOT IN (...)
- Propósito: Encontrar las empresas que no están en la lista de empresas activas.
 - Lógica Técnica: La consulta principal recorre la tabla company. La cláusula WHERE c.id NOT IN (...) es la clave: para cada empresa, comprueba si su id NO se encuentra en la lista generada por la subconsulta. Si el id no está en la lista (lo que significa que esa

empresa no tiene ninguna transacción registrada), se selecciona su company_name para el resultado final. Este enfoque de "exclusión" es muy eficiente para encontrar registros en una tabla que no tienen correspondencia en otra.

Nivel 2

Ejercicio 1

Identifica los cinco días que se generó la mayor cantidad de ingresos en la empresa por ventas. Muestra la fecha de cada transacción junto con el total de las ventas



The screenshot shows a SQL IDE interface. The top pane contains a SQL query. The bottom pane shows the results of the query in a table format. The query is as follows:

```
-- Level 2
-- Exercise 1: Identify the five days when the highest amount of income was generated in the company from sales. Show the date of each
SELECT
    DATE_FORMAT(DATE(timestamp), '%M %e, %Y') AS sale_date,
    ROUND(SUM(amount), 2) AS total_sales
FROM transaction AS t
WHERE declined = 0
GROUP BY sale_date
ORDER BY total_sales DESC
LIMIT 5
;
```

The results table shows the following data:

sale_date	total_sales
December 13, 2022	14337.44
November 18, 2019	13591.32
February 20, 2023	13332.59
December 20, 2017	13318.43
March 18, 2019	12680.95

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
1	10:59:20	USE transactions	0 row(s) affected	0.000 sec
2	10:59:31	SELECT DATE_FORMAT(DATE(timestamp), '%M %e, %Y') AS sale_date, ROUND(SUM(amount), 2) AS total_sales FROM transaction AS t WHERE declined = 0 GROUP BY sale_date ORDER BY total_sales DESC LIMIT 5	5 row(s) returned	0.078 sec / 0.000 sec

Justificación del código:

1. WHERE declined = 0: Se aplica un filtro inicial para asegurar que solo se consideren las transacciones que fueron completadas exitosamente. Este paso es fundamental para garantizar que el cálculo de ingresos sea preciso y no se vea contaminado por intentos de venta fallidos.
2. DATE_FORMAT(DATE(timestamp), '%M %e, %Y') AS sale_date: Esta expresión es clave para la agrupación.

- Primero, DATE(timestamp) extrae únicamente la porción de fecha (ignorando la hora, minuto y segundo) de cada registro de timestamp. Esto permite tratar todas las transacciones ocurridas en un mismo día natural como parte del mismo conjunto.
 - Luego, DATE_FORMAT(...) da un formato específico y legible a esa fecha (ej. "September 21, 2025"). Esto es principalmente para la presentación del resultado final.
3. GROUP BY sale_date: Agrupa todas las filas (transacciones) que comparten la misma fecha (sale_date) en un único registro resumen. Este es el paso que permite realizar cálculos agregados, como una suma, para cada día por separado.
 4. ROUND(SUM(amount), 2) AS total_sales: Para cada grupo diario creado en el paso anterior, la función SUM(amount) calcula la suma total de los montos de todas sus transacciones. Posteriormente, ROUND(..., 2) redondea esta suma a dos decimales, una práctica estándar para representar valores monetarios.
 5. ORDER BY total_sales DESC: Ordena los registros diarios agregados de forma descendente, basándose en el total de ventas (total_sales) calculado para cada uno. Esto coloca los días de mayores ingresos en las primeras posiciones.
 6. LIMIT 5: Finalmente, restringe la salida a las primeras 5 filas del resultado ya ordenado. De esta manera, se asegura de mostrar únicamente los cinco días con el mayor volumen de ventas, cumpliendo con el requisito del ejercicio.
-

Ejercicio 2

¿Cuál es el promedio de ventas por país? Presenta los resultados ordenados de mayor a menor medio

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

```

97 ;
98
99
100 -- Exercise 2: What is the average sales by country? Present the results sorted from highest to lowest average.
101
102 • SELECT c.country, ROUND(AVG(t.amount),2) AS sales_average
103 FROM transaction AS t
104 JOIN company AS c ON c.id = t.company_id
105 WHERE declined = 0
106 GROUP BY c.country
107 ORDER BY sales_average DESC
108 ;

```

The results grid shows the following data:

country	sales_average
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

The output log shows the following actions:

#	Time	Action	Message	Duration / Fetch
1	11:00:47	USE transactions	0 row(s) affected	0.032 sec
2	11:01:47	SELECT c.country, ROUND(AVG(t.amount),2) AS sales_average FROM transacti...	15 row(s) returned	0.172 sec / 0.000 sec

Justificación del código:

1. FROM transaction AS t JOIN company AS c ON c.id = t.company_id: Se realiza una unión (JOIN) entre la tabla de transacciones y la de compañías. Esta operación es indispensable para vincular cada transacción con los datos de su compañía correspondiente, específicamente para acceder al country de la compañía, que es el criterio de agrupación principal.
2. WHERE declined = 0: Al igual que en el caso anterior, se filtran las transacciones para incluir solo las ventas exitosas en el cálculo del promedio, evitando que los datos sean sesgados por operaciones fallidas.
3. GROUP BY c.country: Agrupa todas las filas de transacciones según el país de la compañía. Se crea un "cubo" o grupo por cada país (ej. uno para todas las transacciones de Alemania, otro para las de Francia, etc.).
4. ROUND(AVG(t.amount), 2) AS sales_average: Dentro de cada grupo de país, la función de agregación AVG(t.amount) calcula el valor promedio de las ventas. ROUND se usa para formatear este promedio a dos decimales.
5. ORDER BY sales_average DESC: Ordena los resultados finales (una fila por país) de mayor a menor según el valor del promedio de ventas, presentando los países más rentables en primer lugar.

Ejercicio 3

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

- Muestra el listado aplicando JOIN y subconsultas.
- Muestra el listado aplicando solamente subconsultas.

Versión 1: Usando JOIN y Subconsultas

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

```
-- Exercise 3: In your company, a new project is proposed to launch some advertising campaigns to compete with the company "Non Institute". For this, you are asked for the list of all transactions made by companies that
-- Applying Join and subqueries

SELECT *
FROM transaction as t
JOIN company c on c.id = t.company_id
WHERE c.country = (
    SELECT country
    FROM company
    WHERE company_name = "Non Institute"
)
```

The results grid displays a table with columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined, id, company_name, phone, email, country, website. The table contains 1377 rows of transaction data.

Below the results grid, the 'Action Output' tab shows the execution details:

#	Time	Action	Message	Duration / Fech
1	11:45:27	USE transactions	0 row(s) affected	0.000 sec
2	11:45:58	SELECT * FROM transaction as t JOIN company c on c.id = t.company_id WHERE c.country = (SELECT country FROM company WHERE company_name = "Non Institute")	1377 row(s) returned	0.016 sec / 0.047 sec

Justificación código:

1. Subconsulta: (SELECT country FROM company WHERE company_name = "Non Institute"): Esta subconsulta se ejecuta primero y de forma independiente. Su único propósito es encontrar y devolver un único valor (un escalar): el nombre del país donde se encuentra la compañía "Non Institute".

2. Consulta Principal:

- JOIN company c on c.id = t.company_id: Une las tablas de transacciones y compañías para poder filtrar por un atributo de la compañía (el país).
- WHERE c.country = ...: Compara el país de cada compañía en la tabla unida con el valor único devuelto por la subconsulta. De esta forma, filtra la tabla unida para quedarse solo con las filas correspondientes a transacciones de compañías que están en el mismo país que "Non Institute".

Versión 2: Muestra el listado aplicando solamente subconsultas.

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

```

133
134 -- Applying only subqueries
135
136 • SELECT *
137 FROM transaction as t
138
139 WHERE t.company_id IN (
140     SELECT id
141     FROM company
142     WHERE country = (
143         SELECT country
144         FROM company
145         WHERE company_name = "Non Institute"
146     )
147 )
148 ;

```

The results pane shows a table with the following data:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
008629B4-C9A9-406C-A3D2-71FDA47BC546	CcS-7063	b-2246	2482	45.7666	4.83048	2015-07-30 12:12:42	486.44	0
00872BA4-54A3-4B8E-B13F-2D57535AA17A	CcS-8475	b-2246	3894	55.6212	-3.7546	2017-10-26 22:08:26	414.06	0
01F075B1-D7AE-4D02-AAD9-5FFD72A43F3C	CcS-8700	b-2246	4119	55.856	-3.15783	2018-01-27 13:44:36	103.73	0
023FFCE8-E618-4938-BF56-C8DF80540ADD	CcS-7816	b-2246	3235	46.3568	1.82755	2016-12-19 11:53:45	219.28	0
026838EB-EF91-4564-957B-D6F1662AB7C5	CcS-9471	b-2246	4890	42.1332	12.396	2017-01-10 21:09:29	326.87	0

The output pane shows the following actions:

#	Time	Action	Message	Duration / Fetch
1	11:04:47	USE transactions	0 row(s) affected	0.000 sec
2	11:06:58	SELECT * FROM transaction as t WHERE t.company_id IN (SELECT id	13776 row(s) returned	0.000 sec / 0.015 sec

Este enfoque descompone el problema de manera jerárquica, de adentro hacia afuera:

1. Subconsulta Interna: (SELECT country FROM company WHERE company_name = "Non Institute"): Al igual que en la versión anterior, esta consulta se resuelve primero, obteniendo el nombre del país objetivo (ej. 'USA').

2. Subconsulta Intermedia: (SELECT id FROM company WHERE country = ...): El resultado de la consulta interna ('USA') se usa aquí. Esta consulta genera una lista de todos los id de las compañías que se encuentran en ese país.
3. Consulta Externa: SELECT * FROM transaction as t WHERE t.company_id IN (...): La consulta principal selecciona todas las columnas de la tabla transaction. La cláusula WHERE t.company_id IN actúa como filtro final, seleccionando únicamente aquellas transacciones cuyo company_id está presente en la lista de IDs generada por la subconsulta intermedia.

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 de 2015, 20 de julio de 2018 y 13 de marzo de 2024. Ordena los resultados de mayor a menor cantidad.

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

```

145 WHERE company_name = "Non Institute"
146 )
147 )
148 }
149
150 -- Level 3
151 -- Exercise 1: Present the name, phone number, country, date, and amount of those companies that made transactions with a value between 350 and 400 euros on any of these dates: April 29, 2015, July 20, 2018, and March 1
152
153 SELECT
154     c.company_name,
155     c.phone,
156     c.country,
157     DATE_FORMAT(t.timestamp, '%M %e, %Y') AS transaction_date,
158     t.amount
159 FROM transaction AS t
160 JOIN company AS c ON t.company_id = c.id
161 WHERE
162     t.declined = 0
163     AND t.amount BETWEEN 350 AND 400
164     AND DATE(t.timestamp) IN (
165         STR_TO_DATE('2015-04-29', '%Y-%m-%d'),
166         STR_TO_DATE('2018-07-20', '%Y-%m-%d'),
167         STR_TO_DATE('2024-03-13', '%Y-%m-%d')
168     )
169 ORDER BY t.amount DESC
170 }
  
```

The results grid shows the following data:

company_name	phone	country	transaction_date	amount
Alquam PC	01 45 73 52 36	Germany	March 13, 2024	399.84
Auctor Mauris Vel LLP	08 09 28 74 14	United States	July 20, 2018	399.51
At Pede Corp.	06 14 48 33 15	Italy	April 29, 2015	390.69
Alquam PC	01 45 73 52 36	Germany	March 13, 2024	388.29
Ono Adipiscing Limited	03 18 00 77 81	United Kingdom	July 20, 2018	372.71
Fringilla LLC	08 29 15 93 57	New Zealand	April 29, 2015	367.62
Pede Cum Ltd	07 62 26 48 38	Norway	July 20, 2018	356.87
Auctor Mauris Vel LLP	08 09 28 74 14	United States	March 13, 2024	353.75

The output pane shows the execution of the query, indicating that 0 rows were affected and 0.031 seconds were taken to return the results.

Justificación del código:

1. FROM transaction AS t JOIN company AS c ON t.company_id = c.id: Se realiza una unión de tablas para poder acceder y mostrar columnas de ambas (company_name, phone de la tabla company, y amount, timestamp de la tabla transaction) en un único resultado.
 2. WHERE t.declined = 0: Se aplica un filtro base para excluir transacciones fallidas. Es un paso de integridad de datos para asegurar que el análisis se realiza únicamente sobre ventas efectivas.
 3. AND t.amount BETWEEN 350 AND 400: Se utiliza el operador BETWEEN para filtrar las transacciones cuyo monto se encuentra dentro de un rango numérico inclusivo. Es una forma técnicamente eficiente y legible de expresar una condición de $t.amount \geq 350$ AND $t.amount \leq 400$.
 4. AND DATE(t.timestamp) IN (...): Esta es la cláusula de filtrado de fecha, que tiene dos componentes técnicos importantes:
 - DATE(t.timestamp): Esta función es crucial porque extrae solo la parte de la fecha del campo timestamp, descartando la hora. Esto permite comparar el día de la transacción con las fechas objetivo, sin importar a qué hora del día se realizó la venta.
 - IN (STR_TO_DATE(...), ...): El operador IN permite comprobar si la fecha de la transacción coincide con cualquiera de las fechas en una lista definida. STR_TO_DATE se utiliza para convertir las cadenas de texto (ej: '2015-04-29') a un tipo de dato de fecha real, asegurando una comparación correcta y robusta entre los valores.
 5. DATE_FORMAT(t.timestamp, '%M %e, %Y'): A diferencia de la función DATE() usada en el WHERE para la lógica de filtrado, DATE_FORMAT se usa en el SELECT con un propósito de presentación. Transforma la fecha de la transacción a un formato de texto específico y legible para el usuario final (ej: "March 13, 2024").
 6. ORDER BY t.amount DESC: Una vez que se han filtrado todas las transacciones que cumplen con los criterios de monto y fecha, esta cláusula ordena el conjunto de resultados final de mayor a menor según el valor de amount, para presentar las transacciones más valiosas primero.
-

Ejercicio 2

Necesitamos optimizar la asignación de los recursos y dependerá de la capacidad operativa que se requiera, por lo que te piden la 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.

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

```

173 -- Exercise 2: We need to optimize the allocation of resources, which will depend on the operational capacity required. Therefore, they ask you for information about the number of transactions that companies carry out,
174
175 SELECT
176     c.id,
177     c.company_name,
178     COUNT(t.id) AS 'number_of_transactions',
179     CASE
180         WHEN COUNT(t.id) > 400 THEN 'Higher than 400'
181         WHEN COUNT(t.id) < 400 THEN 'Smaller 400'
182         ELSE 'Exactly 400'
183     END AS 'transaction_state'
184 FROM transaction AS t
185 JOIN company AS c ON t.company_id = c.id
186 GROUP BY c.id, c.company_name
187 ORDER BY COUNT(t.id) DESC
188
189
190
191
192
193
194

```

The results grid shows the following data:

id	company_name	number_of_transactions	transaction_state
b-2222	Ac Fermentum Incorporated	2401	Higher than 400
b-2302	Nunc Interdum Incorporated	1599	Higher than 400
b-2330	Donec Fringilla PC	1593	Higher than 400
b-2366	Mauris Institute	1586	Higher than 400
b-2614	Rutrum Non Inc.	1585	Higher than 400
b-2350	Aliquet Vel Vulputate Incorporated	1583	Higher than 400
b-2566	Aliquam PC	1576	Higher than 400
b-2574	Orn Idipiscing Limited	1567	Higher than 400
b-2478	Elam Suspendisse Fermentum Industries	1566	Higher than 400
b-2550	Auctor Mauris Corp.	1564	Higher than 400
b-2266	Mus Aenean Eget Foundation	1563	Higher than 400
b-2326	Brim Condimentum Ltd	1550	Higher than 400
b-2464	Iusto Eui Arcu Ltd	1548	Higher than 400
b-2402	At Pede Corp.	1544	Higher than 400

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

Nota: He considerado también interesante indicar aquellos valores con transacción igual a 400, no solamente las mayores y menores.

Justificación del código:

1. FROM transaction AS t JOIN company AS c ON t.company_id = c.id: Se unen las tablas para poder asociar cada transacción con el nombre de su compañía y agrupar los resultados por compañía en un paso posterior.
2. GROUP BY c.id, c.company_name: Este es el paso fundamental de la consulta. Agrupa todas las filas de la tabla unida que pertenecen a la misma compañía (identificada por su id y company_name) en un solo registro resumen. Esto permite que las funciones de agregación (como COUNT) operen sobre el conjunto de transacciones de cada compañía de forma individual.
3. COUNT(t.id) AS 'number_of_transactions': Para cada grupo de compañía definido en el GROUP BY, esta función de agregación cuenta el número de

transacciones asociadas. Se cuenta t.id (el ID de la transacción) para obtener un total preciso de operaciones por compañía.

4. CASE ... END AS 'transaction_state': Esta estructura implementa lógica condicional que se ejecuta *después* de la agregación.
 - Propósito Técnico: Permite crear una nueva columna derivada (transaction_state) cuyo valor depende del resultado del cálculo de COUNT(t.id) para cada compañía.
 - Funcionamiento: Para cada fila del resultado agrupado, el CASE evalúa el valor de COUNT(t.id). Si es mayor a 400, asigna la etiqueta 'Higher than 400'. Si no, comprueba si es menor a 400 y asigna 'Smaller 400'. Si ninguna de las dos condiciones anteriores es cierta, el ELSE captura el caso restante (el conteo es exactamente 400).
5. ORDER BY COUNT(t.id) DESC: Finalmente, ordena el listado de compañías de forma descendente basándose en el número de transacciones que ha realizado cada una. Esto se hace para que las compañías con mayor volumen de operaciones aparezcan primero, proporcionando un ranking de actividad.