

# DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

## SPRINT 02

Bases de datos relacionales e introducción a SQL



**Letiane Benincá**  
[benincalf@gmail.com](mailto:benincalf@gmail.com)

**IT ACADEMY**



## PRINCIPALES COMANDOS

**SELECT:** extraer datos específicos de una o más tablas.

**WHERE:** filtrar los resultados según condiciones específicas (valores de rango, fechas o otros).

**JOIN:** combinar datos de dos tablas relacionadas utilizando claves comunes.

**GROUP BY:** agrupar los resultados en base a una columna específica para realizar cálculos como conteo o promedio.

**ORDER BY:** ordena los resultados de manera ascendente (ASC) o descendente (DESC) según una columna.

**CASE:** crear columnas personalizadas basadas en condiciones específicas.

**BETWEEN:** filtra valores que estén dentro de un rango determinado.

**IN:** seleccionar valores específicos.

**COUNT:** contar el número de registros en una columna específica.

**AVG:** promedio de valores en una columna.

**WHERE NOT IN:** filtra registros que no coinciden con una lista de valores específicos.

## EJERCICIO 01

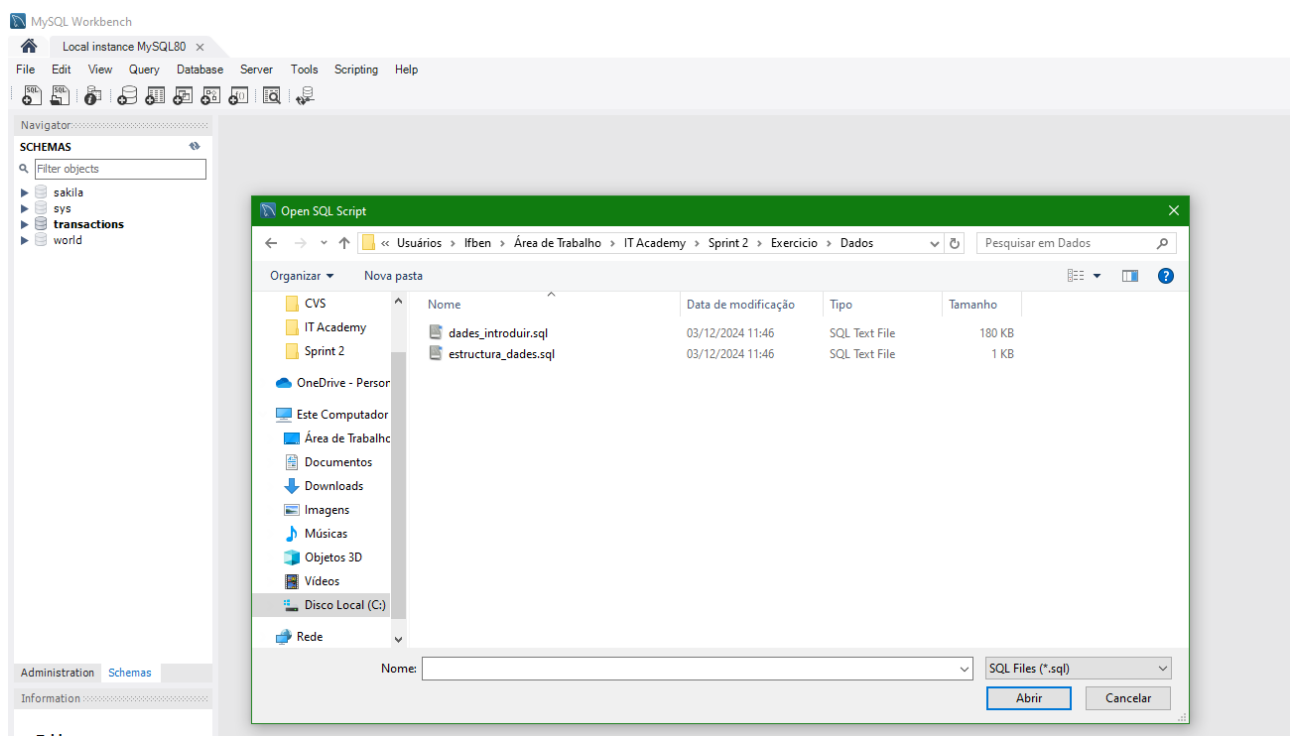
A partir de los documentos adjuntos (*estructura\_dades* y *dades\_introduir*), importa las dos tablas. Muestra las características principales del esquema creado y explica las diferentes tablas y variables que existen. Asegúrate de incluir un diagrama que ilustre la relación entre las diferentes tablas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

Datos a introducir:

*Dades\_introduir.sql*

*Estructura\_dades.sql*

IMPORTAR TABLAS:



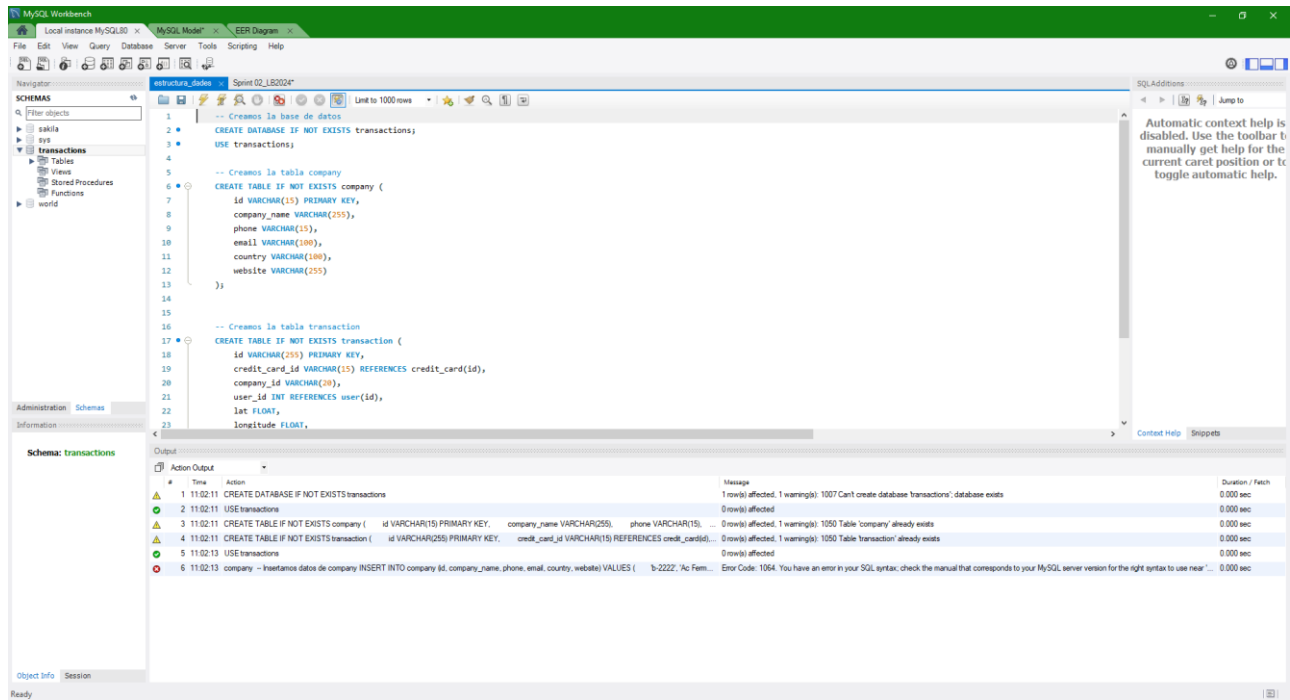
## MIRAR ESQUEMAS:

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'schemas' tab with a list of databases: sakila, sys, transactions, and world. The 'estructura\_datos' tab is active, displaying a list of INSERT statements for the 'company' table. The statements are numbered 1 through 31, each inserting a new company record with fields: id, company\_name, phone, email, country, and website. The data is split across two columns in the editor.

## EXECUTAR TABLAS (o ctrl + shift + enter):

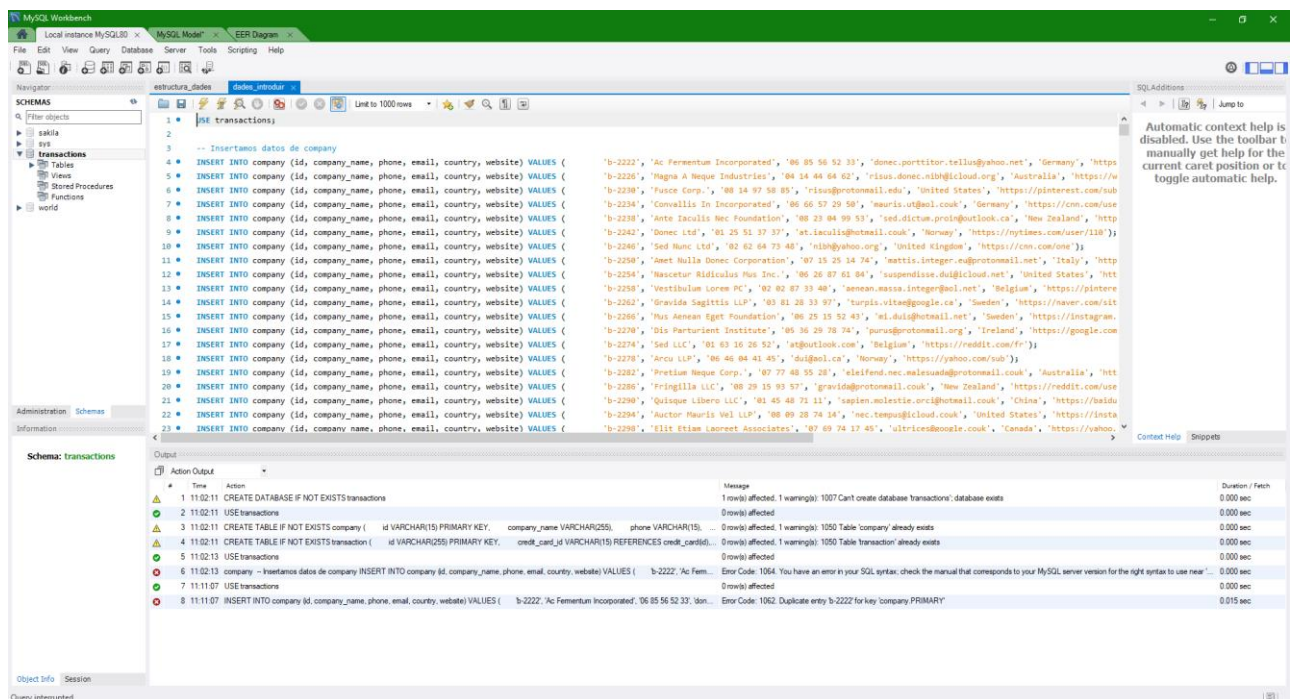
The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'schemas' tab with a list of databases: sakila, sys, transactions, and world. The 'estructura\_datos' tab is active, displaying a list of INSERT statements for the 'company' table. The statements are numbered 1 through 31, each inserting a new company record with fields: id, company\_name, phone, email, country, and website. The data is split across two columns in the editor. The 'Table: company' structure is visible in the bottom left pane, showing columns: id (varchar(15)), company\_name (varchar(255)), and phone (varchar(15)).

## ESTRUCTURA DATOS:



El archivo describe las informaciones iniciales para la tabla de estructuración de los datos, como *id* (llave primaria), nombre de la empresa, teléfono, email, país e website. Desde esta forma estructurada, podremos linkar los datos con la información de los datos a seguir:

## DATOS A INTRODUCIR:

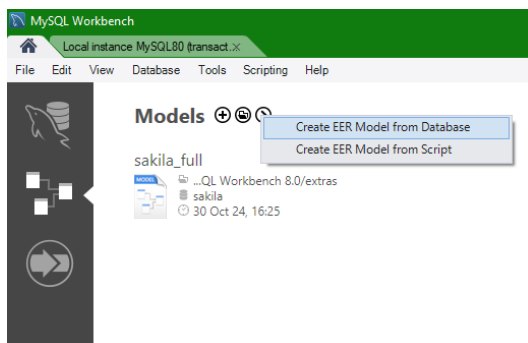




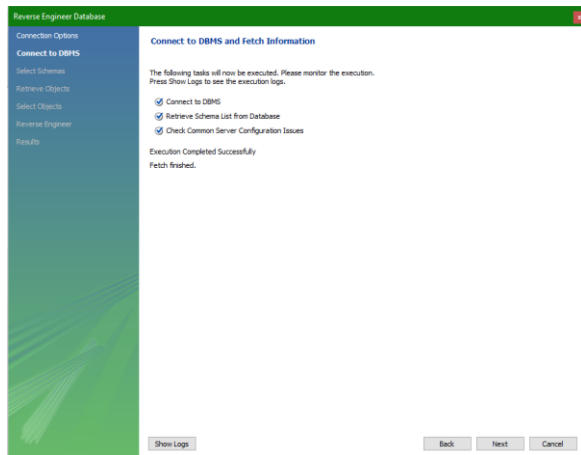
En este, están los datos de los clientes, estructurados de acuerdo con la estructura inicial. Con *id* (llave primaria) y los datos rellenos de los clientes.

## DIAGRAMA RELACIONAL (ER):

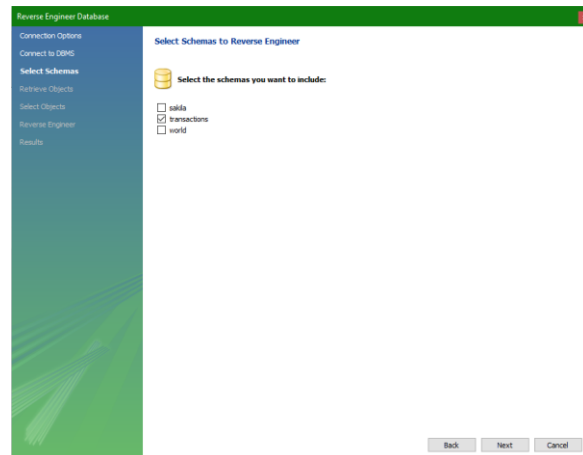
Como llegar hasta él:



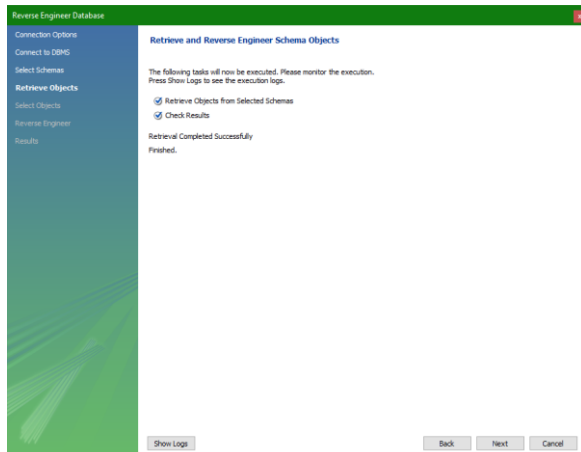
1



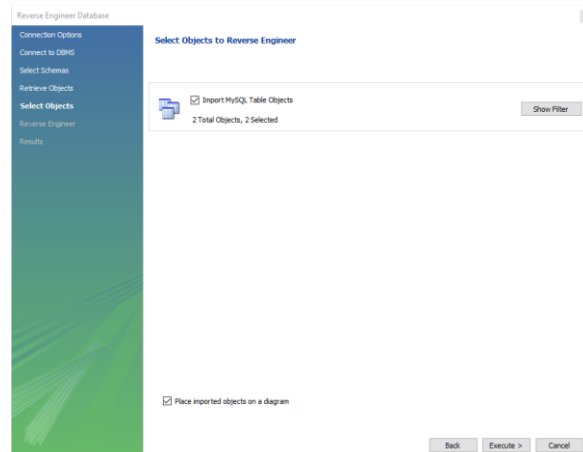
2



3

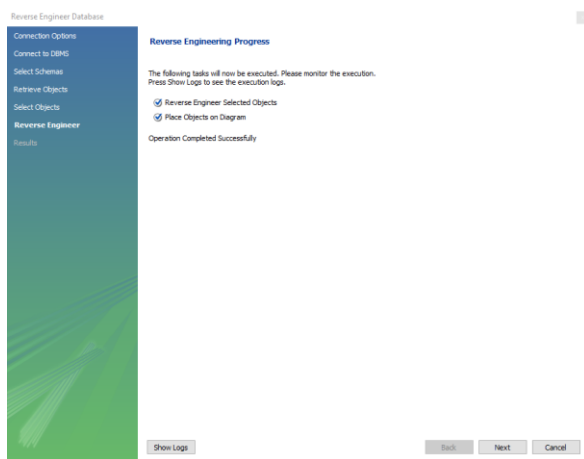


4

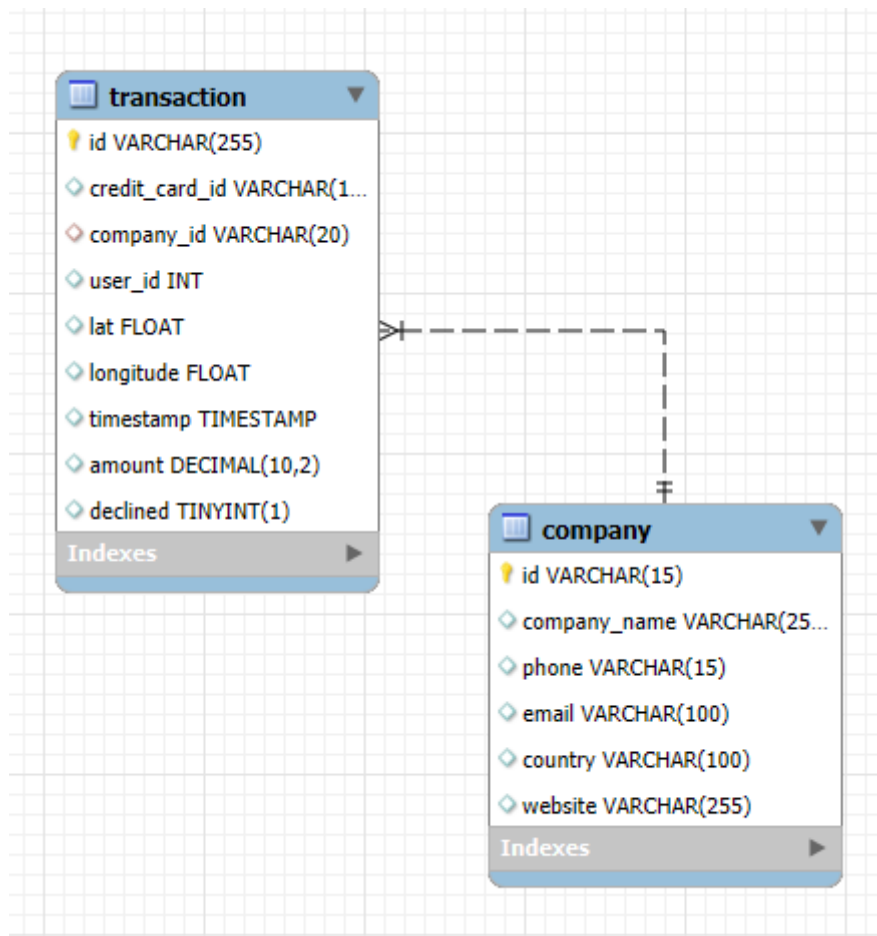


5

6

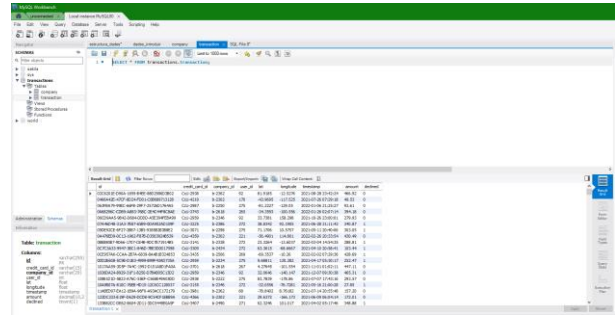
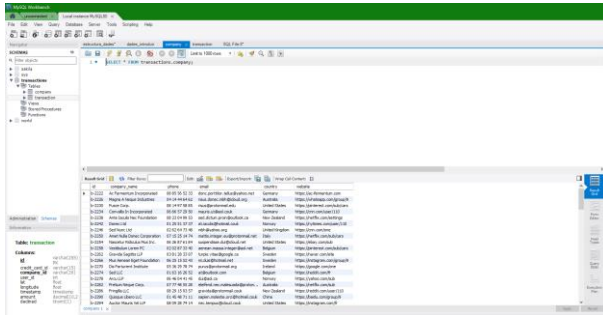


7



Muestra una relación 1:N entre las tablas *company* (estructura\_dades) y *transaction* (dades\_introducir). Una compañía puede estar vinculada a múltiples transacciones. La conexión se establece mediante por el *company\_id* en la tabla *transaction*, que referencia la columna *id* de la tabla *company*.

## ABRIR LAS TABLAS:



## SELECT → FROM company

SQL Developer - Script 02\_S

```

1 #Nivel 01: Ejercicio 01
2 select *
3 from company;
4
5 select *
6 from transaction;
7
8 #Nivel 01: Ejercicio 02
9 #Listado de los países que están realizando compras
10 select distinct country
11 from company;
12
13 #Desde cuántos países se realizan las compras
14 select count(distinct company.country)
15 from transaction
  
```

Result Grid

id	company_name	phone	email	country	website
b-2222	Ac Fermentum Incorporated	08 85 56 52 33	dorc.pottitor.telus@yahoo.net	Germany	https://ac-fermentum.com
b-2226	Magna A Heque Industries	04 14 44 64 62	dorc.dorec.nbi@cloud.org	Australia	https://whatsapp.com/group/9
b-2230	Fuuse Corp.	08 14 97 58 85	rusus@protonmail.edu	United States	https://pinterest.com/sub/cars
b-2234	Convallis Inc Incorporated	06 66 57 29 50	maurus.ut@aol.co.uk	Germany	https://tumblr.com/user/110
b-2238	Anite Jaculis Nec Foundation	08 23 04 99 53	sed.dctum.pron@outlook.ca	New Zealand	https://netflix.com/settings
b-2242	Donec Ltd	01 25 51 37 37	at.iculis@btmail.co.uk	Norway	https://mytimes.com/user/110
b-2246	Sed Nunc Ltd	02 62 64 73 48	nbi@yahoo.org	United Kingdom	https://tumblr.com/one
b-2250	Amet Nulla Donec Corporation	07 16 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://netflix.com/sub/cars
b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@cloud.net	United States	https://ebay.com/sub
b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.massa.integer@aol.net	Belgium	https://pinterest.com/sub/cars
b-2262	Gravida Sagittis LLP	03 81 28 33 97	turpis.vitae@google.ca	Sweden	https://naver.com/site
b-2266	Mus Aenean Eget Foundation	06 25 15 52 43	mi.duis@hotmail.net	Sweden	https://instagram.com/group/9
b-2270	Dis Parturient Institute	05 36 29 78 74	purus@protonmail.org	Ireland	https://google.com/one
b-2274	Sed LLC	01 63 16 26 52	at@outlook.com	Belgium	https://reddit.com/fr
b-2278	Arcu LLP	06 46 04 41 45	dui@aol.ca	Norway	https://yahoo.com/sub
b-2282	Prehium Heque Corp.	07 77 48 55 28	eleifend.nec.malesuada@proton...	Australia	https://netflix.com/sub
b-2286	Fringilla LLC	08 29 15 93 57	gravida@protonmail.co.uk	New Zealand	https://reddit.com/user/110

company 8 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5	10:59:37	SELECT * FROM transaction join company on transaction.company_id = company.id where company.country = (select country from company where...	100 row(s) returned	0.000 sec / 0.000 sec
6	10:59:53	SELECT * FROM transaction where company_id in (select id from company where country = (select country from company where comp...	100 row(s) returned	0.016 sec / 0.000 sec
7	11:00:36	select company_name, company.phone, company.country, date(transaction.timestamp) AS FECHA, transaction amount from transaction join ...	4 row(s) returned	0.015 sec / 0.000 sec
8	11:41:23	select * from company LIMIT 0, 1000	100 row(s) returned	0.000 sec / 0.000 sec

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

## Retorna 587 líneas



## SELECT → FROM transaction

The screenshot shows the SQL Data Analysts interface. The query editor contains the following SQL code:

```

1 #Nivel 01: Ejercicio 01
2 select *
3 from company;
4
5 select *
6 from transaction;
7
8 #Nivel 01: Ejercicio 02
9 #Listado de los países que están realizando compras
10 select distinct country
11 from company;
12
13 #Desde cuántos países se realizan las compras
14 select count(distinct company.country)
15 from transaction

```

The results grid displays a table with the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. The table contains 100 rows of data.

The output pane shows the execution of the query, with the following messages:

```

6 10:53:53 SELECT * FROM transaction where company_id in (select id from company where country = (select country from company where comp... 100 row(s) returned
7 11:00:36 select * from company company_name, company phone, company country, date(transaction.timestamp) AS FECHA, transaction amount from transaction join ... 4 row(s) returned
8 11:41:23 select * from company LIMIT 0, 1000 100 row(s) returned
9 11:45:46 select * from transaction LIMIT 0, 1000 587 row(s) returned

```

Retorna 587 líneas. Son las dos tablas con los valores existentes. Para poder “verlas” hay que escribir **SELECT \*** (*asterisco = para traer todos los valores introducidos*). Después **FROM** + el nombre de la tabla que se quiere sacar los datos.

## EJERCICIO 02

Utilizando JOIN realizarás las siguientes consultas:

- Listado de los países que están realizando compras.
- Desde cuántos países se realizan las compras.
- Identifica la compañía con el promedio más alto de ventas.

Listado de los países que están realizando compras.

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```

7
8 #Nivel 01: Ejercicio 02
9 #Listado de los países que están realizando compras
10 • select distinct country
11   from company;
12
13 #Desde cuántos países se realizan las compras
14 • select count(distinct company.country)
15   from transaction
16  join company on transaction.company_id = company.id;
17
18 #Identifica la compañía con el promedio más alto de ventas
19 • select
20   transaction.company_id,
21   count(transaction.id) AS VENTAS

```

The results pane shows the output of the first query, which is a list of countries:

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

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

#	Time	Action	Message	Duration / Fetch
7	11:00:36	select company.company_name, company.phone, company.country, date(transaction.timestamp) AS FECHA, transaction.amount from transaction join ...	4 row(s) returned	0.015 sec / 0.000 sec
8	11:41:23	select * from company LIMIT 0, 1000	100 row(s) returned	0.000 sec / 0.000 sec
9	11:45:46	select * from transaction LIMIT 0, 1000	587 row(s) returned	0.000 sec / 0.015 sec
10	11:47:09	select distinct country from company LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec

Usando el comando seleccionar (*select*) con *distinct* para evitar repeticiones, nos salen 15 países que están realizando compras.

Desde cuántos países se realizan las compras:

The screenshot shows the SQL Data Analytics Studio interface. The main window displays a SQL query in the editor:

```

7
8 #Nivel 01: Ejercicio 02
9 #Estado de los países que están realizando compras
10 • select distinct country
11   from company;
12
13 #Desde cuántos países se realizan las compras
14 • select count(distinct company.country)
15   from transaction
16  join company on transaction.company_id = company.id;
17
18 #Identifica la compañía con el promedio más alto de ventas
19 • select
20   transaction.company_id,
21   count(transaction.id) AS VENTAS

```

The Results pane shows the output of the query:

count(distinct company.country)
15

The Output pane shows the execution log:

Time	Action	Message	Duration / Fetch
8 11:41:23	select * from company LIMIT 0, 1000	100 row(s) returned	0.000 sec / 0.000 sec
9 11:45:46	select * from transaction LIMIT 0, 1000	587 row(s) returned	0.000 sec / 0.015 sec
10 11:47:09	select distinct country from company LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
11 11:48:13	select count(distinct company.country) from transaction join company on transaction.company_id = company.id LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Con *count* contamos el número de países y el comando *distinct* para evitar repetirlos. Los campos *company\_id* de la tabla **transaction** y el campo *country* de la tabla **company**. Con *JOIN* para combinar las tablas (conectándolas a través de la relación entre *transaction.company\_id* y *company.id*). Nos resultan 15 países.

Se identifica la compañía con el promedio más alto de ventas:

The screenshot displays the DBeaver SQL editor interface. The main editor contains two SQL queries. The first query, starting at line 14, counts the number of distinct countries from the `transaction` table. The second query, starting at line 19, joins the `transaction` and `company` tables, groups the results by `transaction.company_id`, orders them by the count of transactions (labeled as `ventas`) in descending order, and limits the output to 1 row. Below the editor, the 'Result Grid' shows the results of the second query, with a single row for `company_id` `b-2302` having `10500` `VENTAS`. The 'Output' pane at the bottom shows the execution log, indicating that the queries were executed successfully and returned the expected number of rows.

```

13
14 • #Desde cuántos países se realizan las compras
15   select count(distinct company.country)
16   from transaction
17   join company on transaction.company_id = company.id;
18
19 • #Identifica la compañía con el promedio más alto de ventas
20   select
21     transaction.company_id,
22     count(transaction.id) AS VENTAS
23   from transaction
24   join company on transaction.company_id = company_id
25   group by transaction.company_id
26   order by ventas desc
27   limit 1;

```

company_id	VENTAS
b-2302	10500

#	Time	Action	Message	Duration / Fetch
9	11:45:46	select * from transaction LIMIT 0, 1000	567 row(s) returned	0.000 sec / 0.015 sec
10	11:47:09	select distinct country from company LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
11	11:48:13	select count(distinct company.country) from transaction join company on transaction.company_id = company.id LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
12	11:48:52	select transaction.company_id, count(transaction.id) AS VENTAS from transaction join company on transaction.company_id = company_id group by transaction.company_id order by ventas desc limit 1	1 row(s) returned	0.063 sec / 0.000 sec

Como resultado, la compañía `b_2302`, fue la que realizó más compras con un total de 10500. *SELECT* para seleccionar el identificador de la empresa (*transaction.company\_id*) y contar el número de transacciones asociadas con *COUNT(transaction.id)*, asignando el nombre *ventas* al resultado. Luego, se conectaron las tablas **transaction** y **company** mediante *JOIN*. Con *GROUP BY transaction.company\_id*, se agrupan las transacciones por empresa, y con *ORDER BY ventas DESC* para ordenar los resultados de mayor a menor número de transacciones. Por fin, usado el comando *limit 1* para delimitar solamente una respuesta.

## EJERCICIO 03

Utilizando solo subconsultas (sin utilizar JOIN):

- Muestra todas las transacciones realizadas por empresas de Alemania.
- Lista las empresas que han realizado transacciones por un monto superior al promedio de todas las transacciones.
- Se eliminarán del sistema las empresas que no tienen transacciones registradas; entrega el listado de estas empresas.

Transacciones realizadas por empresas de Alemania (Germany):

The screenshot displays a SQL IDE interface. The main editor shows a query with a subquery to filter transactions for Germany. The results table shows columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. Below the table is an output log showing the execution of the query and the number of rows returned.

```

25 order by ventas desc
26 limit 1;
27
28 #Nivel 01: Ejercicio 03
29 #Transacciones realizadas por empresas de Alemania
30 • select *
31   from transaction
32  where company_id in (
33      select id
34      from company
35     where country = "Germany"
36  )
37
38 #Empresas que han realizado por un monto superior al promedio de todas las ventas
39 • select

```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
1088 ID ID-5823-A76C-55EF-C568E49A0SD0	CdJ-2938	b-2222	275	83.7839	-178.86	2021-07-07 17:43:16	293.57	0
EA2C3281-C9C1-A387-44F8-729F84B51C76	CdJ-2938	b-2222	275	20.2004	-116.84	2021-05-09 10:25:08	119.36	1
00D2E608-5C9E-D1B5-4999-899F-43AD735A	CdJ-2959	b-2234	275	9.68811	130.282	2021-04-17 05:30:17	252.47	1
A008F53-965E-A3A8-C85F-C48C-FD02501	CdJ-2959	b-2234	275	1.64819	-158.007	2021-04-15 13:37:18	60.99	0
0466A2E-47CF-4D24-FD01-C0B88713128	CdJ-4219	b-2302	170	-43.9695	-117.525	2021-07-26 07:29:18	49.53	0
0A476ED9-0C13-1962-F87B-D3563924B539	CdJ-4359	b-2302	221	-56.4901	114.801	2022-02-26 20:33:54	430.49	0
122DC333-E19F-0629-DCD8-9C54CF-1E8B9A	CdJ-4366	b-2302	221	29.6372	-166.173	2021-06-09 06:04:14	172.01	0
1352678A-2E7D-957C-C43C-6450A2B3ED54	CdJ-4520	b-2302	210	20.6724	14.9732	2021-12-29 20:38:23	17.97	0
14C4E585-8F81-2E4A-4C25-0E4A167534F4	CdJ-4849	b-2302	189	-53.6202	93.0533	2021-12-31 00:29:42	388.04	0
158A3ACB-5A1C-08CC-658D-6373CC678F1C	CdJ-4849	b-2302	183	42.5424	-170.347	2022-03-08 05:02:19	240.29	0
162C7E78-ADAD-7082-A748-9112B8920CCC	CdJ-4527	b-2302	210	-69.1381	58.0017	2021-04-11 05:59:18	231.26	0
1717FD68-ADAD-7082-A748-9112B8920CCC	CdJ-4219	b-2302	172	69.4892	-138.411	2021-12-29 16:18:54	249.91	0
1753A288-9FC1-52E8-5C39-AJFFB97B003A	CdJ-4345	b-2302	222	57.9422	-114.729	2021-08-17 05:32:08	497.84	0
186F530C-DC27-8F6E-682F-18B61CB37726	CdJ-4310	b-2302	225	-72.7448	36.6211	2021-12-30 13:13:45	238.16	0
18C4E20C-4E4C-F35E-2198-C6680B081DC25	CdJ-4849	b-2302	177	68.0133	91.4839	2021-09-24 18:55:25	237.04	0
18CCBA7C-ABC1-813D-FAF3-46A897429368	CdJ-4219	b-2302	173	51.3881	-156.371	2021-06-21 03:21:34	58.16	0
19E1EC3E-2119-1EFD-8AAE-5930D4A4E53F	CdJ-4219	b-2302	154	-56.0839	116.987	2021-07-06 21:40:15	29.63	0

transaction 13 x

Output

#	Time	Action	Message	Duration / Fetch
10	11:47:09	select distinct country from company LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
11	11:48:13	select count(distinct company.country) from transaction join company on transaction.company_id = company.id LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
12	11:48:52	select transaction.company_id, count(transaction.id) AS VENTAS from transaction join company on transaction.company_id = company.id group by tra...	1 row(s) returned	0.063 sec / 0.000 sec
13	11:49:54	select * from transaction where company_id in (select id from company where country = "Germany") LIMIT 0, 1000	118 row(s) returned	0.000 sec / 0.000 sec

Se utilizó una subconsulta con *WHERE*, primero, seleccioné el *company\_id* de la tabla **transaction** y lo comparé con los *id* de empresas cuya columna *country* sea **Germany** en la tabla **company**.

Empresas que han realizado transacciones por un monto superior al promedio de todas las transacciones:

The screenshot shows a SQL IDE interface. The top pane contains a SQL query:
 

```

  36  )
  37  )
  38  #Empresas que han realizado por un monto superior al promedio de todas las ventas
  39  select
  40  id, company_name, phone, email, country, website
  41  from company
  42  where id in (
  43  select company_id
  44  from transaction
  45  where amount >= (
  46  select avg(transaction.amount)
  47  from transaction
  48  )
  49  )
  50  )
  
```

 The bottom pane displays the 'Result Grid' with 15 rows of data. The columns are: id, company\_name, phone, email, country, and website. The data includes companies like Ac Fermentum Incorporated, Magna A Neque Industries, Fuceo Corp., Ante Jacula Nec Foundation, Donec Ltd., Sed Nunc Ltd., Nascetur Ridiculus Mus Inc., Vestibulum Lorem PC, Gravida Sagittis LLP, Mus Aenean Eget Foundation, De Parturient Institute, Sed LLC, Arcu LLP, Fringilla LLC, Ekt Elam Laoreet Associates, Nunc Interdum Incorporated, and Augue Foundation.
 The 'Output' pane at the bottom shows the execution log with the following messages:
 

```

  13 11:49:54 select * from transaction where company_id in (select id from company where country = "Germany") LIMIT 0, 1000 118 row(s) returned 0.000 sec / 0.000 sec
  14 11:51:46 select company_name, company.phone, company.country, from company where id in (select company_id from transaction where amount >= (select avg(transaction.amount) from transaction)) 70 row(s) returned 0.000 sec
  15 11:52:23 select company_name from company where id in (select company_id from transaction where amount >= (select avg(transaction.amount) from transaction)) 70 row(s) returned 0.000 sec / 0.000 sec
  16 11:54:47 select id, company_name, phone, email, country, website from company where id in (select company_id from transaction where amount >= (select avg(transaction.amount) from transaction)) 70 row(s) returned 0.000 sec / 0.000 sec
  
```

La subconsulta calculó el promedio general de los montos (*SELECT AVG(amount) FROM transaction*). Luego, se seleccionaron los *company\_id* de las empresas cuyas transacciones tienen un monto mayor a este promedio.

Empresas que no tienen transacciones registradas:



The screenshot shows a SQL IDE interface with a query editor, a toolbar, and an output window. The query editor contains the following SQL code:

```

48 }
49 );
50
51 #Empresas que no tienen transacciones registradas
52 • select
53   company_name
54   from
55   company
56   where id not in (
57     select distinct company_id
58     from transaction
59   )
60
61 #Nivel 02: Ejercicio 01
62 #Los cinco días que se generó la mayor cantidad de ingresos en la empresa

```

The output window shows the execution results of the query. The first row is highlighted in blue, indicating an error:

#	Time	Action	Message	Duration / Fetch
14	11:51:46	select company_name, company.phone, company.country, from company where id in (select company_id from transaction where amount > (select avg(amount) from transaction ) ...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '...' at line 1	0.000 sec
15	11:52:23	select company_name from company where id in (select company_id from transaction where amount > (select avg(amount) from transaction ) ...	70 row(s) returned	0.000 sec / 0.000 sec
16	11:54:47	select id, company_name, phone, email, country, website from company where id in (select company_id from transaction where amount > (select avg(amount) from transaction ) ...	70 row(s) returned	0.000 sec / 0.000 sec
17	11:55:53	select company_name from company where id not in (select distinct company_id from transaction ) LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec

No ha retornado ninguna empresa. La subconsulta dentro de *WHERE NOT IN*, selecciona los *company\_id* desde la tabla **transaction**. Con *NOT IN*, filtrado de las empresas de la tabla **company** cuyo *id* no aparece en los resultados de la subconsulta.

# NIVEL 02

## EJERCICIO 01

Identifique los cinco días en los que se generó el mayor ingreso para la empresa por ventas. Muestre la fecha de cada transacción junto con el total de las ventas.

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```

57 select distinct company_id
58   from transaction
59  );
60
61 #Nivel 02: Ejercicio 01
62 #Los cinco días que se generó la mayor cantidad de ingresos en la empresa
63 • select date(timestamp) as date,
64    sum(amount) as total
65   from transaction
66  where declined = 0
67  group by date(timestamp)
68  order by total desc
69  limit 5;
70
71 #Nivel 02: Ejercicio 02
  
```

The results pane displays a table with two columns: **date** and **total**. The data is as follows:

date	total
2021-12-20	1532.36
2021-04-22	1397.96
2021-05-09	1344.37
2022-02-26	1337.62
2021-03-29	1325.12

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

#	Time	Action	Message	Duration / Fetch
15	11:52:23	select company_name from company where id in (select company_id from transaction where amount > (select avg(amount) from transaction ...	70 row(s) returned	0.000 sec / 0.000 sec
16	11:54:47	select id, company_name, phone, email, country, website from company where id in (select company_id from transaction where amount >= (select ...	70 row(s) returned	0.000 sec / 0.000 sec
17	11:55:53	select company_name from company where id not in (select distinct company_id from transaction ) LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
18	11:56:32	select date(timestamp) as date, sum(amount) as total from transaction where declined = 0 group by date(timestamp) order by total desc limit 5	5 row(s) returned	0.000 sec / 0.000 sec

Aquí se presentan los 5 días con mayor ingreso, utilizando el comando de *LIMIT 5*. En la lateral de la fecha si describinan el valor total de las ventas *SUM(AMOUNT)*.

## EJERCICIO 02

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

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

```

69 limit 5;
70
71 #Nivel 02: Ejercicio 02
72 #¿Cuál es el promedio de ventas por país?
73 • select company.country,
74    avg(transaction.amount) as VENTAS
75 from transaction
76 join company on transaction.company_id = company.id
77 where declined = 0
78 group by company.country
79 order by ventas desc
80
81 #Nivel 02: Ejercicio 03
82 #Empresas ubicadas en el mismo país que Non Institute
83 • SELECT *
  
```

The results pane displays a table with two columns: **country** and **VENTAS**. The data is ordered by average sales in descending order.

country	VENTAS
United States	287.531111
Ireland	285.825357
Sweden	276.668362
United Kingdom	271.767527
Canada	261.941930
Belgium	255.217500
Norway	251.114918
Italy	243.942222
Germany	242.239185
Netherlands	240.940000
China	222.240000
Australia	177.331667
France	169.410000
New Zealand	167.061667
Spain	26.220000

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

- 16 11:54:47 select id, company\_name, phone, email, country, website from company where id in ( select company\_id from transaction where amount >= ( select ... 70 row(s) returned
- 17 11:55:53 select company\_name from company where id not in ( select distinct company\_id from transaction ) LIMIT 0, 1000 0 row(s) returned
- 18 11:56:32 select date(timestamp) as date, sum(amount) as total from transaction where declined = 0 group by date(timestamp) order by total desc limit 5 5 row(s) returned
- 19 11:57:13 select company.country, avg(transaction.amount) as VENTAS from transaction join company on transaction.company\_id = company.id where declined = 0 15 row(s) returned

Calculamos el promedio de ventas por país y mostramos los resultados ordenados del mayor a menor. Lo que hice fue tomar las tablas **transaction** (con las transacciones), y **company** (con la información de las empresas). Cálculo del promedio de ventas con **AVG(transaction.amount)** (tabla transactions +. columna amount) renomando como VENTAS con los datos de la tabla **transaction**. Usé **JOIN** para conectar las tablas, relacionando el **company\_id** de las tabla transacciones con el **id** de las empresas de la tabla company. Luego, filtré las transacciones que no fueron rechazadas con un **WHERE declined = 0**. Agrupé los datos por país usando **GROUP BY company.country** (tabla company +. columna country). Para ordenarlo utilicé **ORDER BY** para que los países con mayores promedios de ventas aparezcan primero, con el **DESC** (descendiente).

## EJERCICIO 03

Su empresa está considerando lanzar campañas publicitarias para **competir con la compañía "Non Institute"**. Para ello, solicitan la lista de todas las transacciones realizadas por **empresas ubicadas en el mismo país que dicha compañía**.

Muestre la lista aplicando JOIN y subconsultas.

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

```

78 group by company.country
79 order by ventas desc;
80
81 #Nivel 02: Ejercicio 03
82 #Empresas ubicadas en el mismo país que Non Institute
83 •
84 SELECT *
85 FROM transaction
86 join company on transaction.company_id = company.id
87 where company.country = (
88     select country
89     from company
90     where company_name = 'Non Institute'
91 );
92 #Subconsultas
  
```

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, id, company\_name, phone, email, country, and website. The results show a list of transactions from companies located in the United Kingdom, which is the same country as 'Non Institute'.

The output table has the following structure:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined	id	company_name	phone	email	country	website
28938E1C-EC14-A760-0A75-871477E48D6A	CcJ-2980	b-2346	275	-41.0486	151.685	2021-08-10 08:14:49	383.73	0	b-2346	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	htq
ACD2011A-A2B1-1C3E5-41E1-2A00C65147A	CcJ-2980	b-2346	275	-54.4792	-82.7974	2022-03-05 20:41:20	60.07	1	b-2346	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	htq
4334349E-CEB0-3D68-A4D4-FEB7718A1ACE	CcJ-3092	b-2310	275	-20.4859	150.87	2021-05-03 22:37:23	458.74	0	b-2310	Non Magna LLC	06 71 73 13 17	nisl.quisque.fringilla@hotmail.ca	United Kingdom	htq
8C289A38-77B4-28CD-1FE8-14DED863E773	CcJ-3092	b-2310	275	-78.0295	18.5295	2021-10-18 07:27:35	477.95	1	b-2310	Non Magna LLC	06 71 73 13 17	nisl.quisque.fringilla@hotmail.ca	United Kingdom	htq
147963D2-878A-C78B-4CE3-8D7CDE8548B	CcJ-2994	b-2326	133	66.2672	172.399	2021-08-09 00:58:07	309.45	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
152599C2-02D0-D054-4861-9AEDP3928FF	CcJ-2994	b-2326	126	-47.0189	-141.672	2021-07-05 03:10:00	395.43	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
18636858-A2E9-7C5D-09C9-C5453DAFD03B	CcJ-2994	b-2326	131	70.2543	-13.1336	2021-07-06 08:48:46	195.06	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
204180E5-8804-8E98-8D7A-A95C18F0BF5C	CcJ-2994	b-2326	126	-79.1145	1.51481	2022-01-03 15:59:29	479.52	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
23988576-6C0E-137A-C2F6-3180A188A2D3	CcJ-2994	b-2326	126	23.6174	137.222	2021-08-26 06:04:05	43.90	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
267C4A86-78A7-1C5E-0718-2824863C87D0	CcJ-2994	b-2326	126	-17.5259	104.915	2021-10-01 21:08:53	122.63	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
314C33E2-83D7-49E4-4E2D-28CA534B19D0	CcJ-2994	b-2326	126	-67.8476	-119.978	2021-04-06 17:24:44	91.59	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
3578688E-7B1D-8887-3BC7-2088673AA31E	CcJ-2994	b-2326	126	87.0665	-22.7339	2021-07-26 22:59:24	303.60	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
360C7814F7AF-643A-0946-AD3802683C86	CcJ-2994	b-2326	116	-7.93005	-79.0733	2021-08-21 10:19:58	494.82	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
391E1CFD-D653-E458-A729-F2B93247858	CcJ-2994	b-2326	118	60.5512	103.904	2021-10-09 00:50:38	271.27	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
3C4D7C2A-A402-8941-625A-D64CA5326E8	CcJ-2994	b-2326	117	23.1627	-10.381	2021-04-25 19:11:52	441.27	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq
3ED634C2-01CA-D6E9-D2BD-24B53A7912F7	CcJ-4219	b-2326	137	69.0549	96.5033	2021-03-29 11:38:38	478.54	0	b-2326	Enim Conditum Ltd	09 55 51 66 25	imperdiet.non.vestibulum@yahoo.net	United Kingdom	htq

The output table shows 20 rows of data, each representing a transaction from a company located in the United Kingdom. The query is executed successfully, 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, id, company\_name, phone, email, country, and website.

Aquí debemos encontrar todas las transacciones realizadas por empresas que están en el mismo país que **Non Institute**. Para hacerlo, primero conecté dos tablas: **transaction** (que contiene los datos de las transacciones) y **company** (que tiene información sobre las empresas), usando **JOIN**, relacionando la columna **company\_id** de la tabla **transaction** con la columna **id** de la tabla **company**. Usando **WHERE** para buscar solo las empresas ubicadas en el mismo país que **Non Institute**. Para esto, utilicé una subconsulta que selecciona el país (**country** – column de la tabla **company**) de la empresa **Non Institute**.

Muestre la lista aplicando solo subconsultas.

The screenshot shows a SQL IDE interface with a query editor on the left, a result grid in the center, and an output pane at the bottom. The query in the editor is as follows:

```

90  })
91
92  #Subconsultas
93  * SELECT *
94  FROM transaction
95  where company_id in (
96      select id
97      from company
98      where country = (
99          select country
100         from company
101         where company_name = 'Non Institute'
102     )
103  )
104  })
  
```

The result grid displays the following columns: `id`, `credit_card_id`, `company_id`, `user_id`, `lat`, `longitude`, `timestamp`, `amount`, and `declined`. It contains 20 rows of transaction data.

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

- 20 11:59:24 SELECT \* FROM transaction join company on transaction.company\_id = company.id where company.country = ( select country from company where comp... 100 row(s) returned 0.000 sec / 0.000 sec
- 21 12:00:42 SELECT \* FROM transaction where company\_id in ( select id from company where country = ( select country from company where comp... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use ne... 0.000 sec
- 22 12:00:51 Subconsultas SELECT \* FROM transaction where company\_id in ( select id from company where country = ( select country from company ... Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use ne... 0.000 sec
- 23 12:00:59 SELECT \* FROM transaction where company\_id in ( select id from company where country = ( select country from company where comp... 100 row(s) returned 0.000 sec / 0.000 sec

Para llegar a este resultado utilizando solo subconsultas, el objetivo fue listar todas las transacciones realizadas por empresas ubicadas en el mismo país que **Non Institute**. Primero, seleccioné toda la información de la tabla **transaction**, si aplicó **WHERE company\_id IN** para limitar las transacciones a aquellas realizadas por empresas específicas. Para obtener esas empresas, utilicé una subconsulta que buscó los **id** de las empresas que están en el mismo país que **Non Institute**.

Para determinar el país de **Non Institute**, usé una subconsulta adicional dentro de la anterior, que selecciona el país desde la tabla **company** donde el nombre de la empresa coincide con la empresa e cuestión. Con este dato, la subconsulta principal seleccionó todas las empresas en ese país, y con **WHERE** relacionó las transacciones con esas empresas. El resultado final muestra únicamente las transacciones de empresas que comparten país con **Non Institute**.

## EJERCICIO 01

Presente el nombre, teléfono, país, fecha y valor (amount) de las empresas que realizaron transacciones con valores entre 100 y 200 euros en alguna de estas fechas: 29 de abril de 2021, 20 de julio de 2021 y 13 de marzo de 2022. Ordene los resultados de mayor a menor valor.

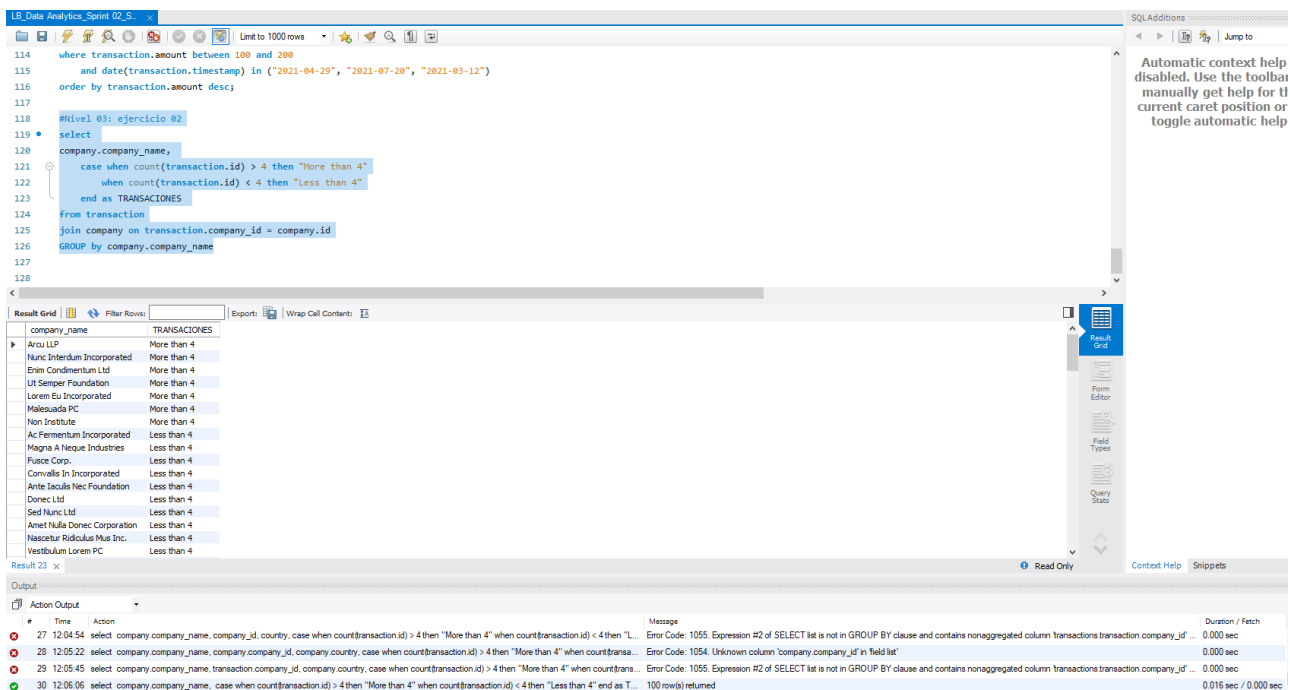
[illegible]

Otra vez, conecté las tablas **transaction** y **company** usando un *JOIN*, relacionando la columna *company\_id* en la tabla **transaction** con la columna *id* en la tabla **company**. Si aplica *WHERE* con *t.amount BETWEEN 100 AND 200* (las transacciones cuyo valor estaba dentro del rango indicado). Si añade la condición con *AND DATE(t.timestamp) IN* para incluir solo las transacciones que ocurrieron en las fechas indicadas para el ejercicio. Por último, para ordenar los resultados de mayor a menor valor, use *ORDER BY* (ordenar por) *transactio.amount* (tabla transaction + .columna amount) *DESC* (los valores se mostrarán comenzando por el más alto).



## EJERCICIO 02

Para optimizar la asignación de recursos operativos, es necesario conocer la cantidad de transacciones realizadas por empresa. El departamento de recursos humanos requiere una lista de las empresas especificando si tienen más o menos de 4 transacciones.



The screenshot displays a database IDE with a SQL query and its execution results. The query is as follows:

```

114 where transaction.amount between 100 and 200
115 and date(transaction.timestamp) in ("2021-04-29", "2021-07-20", "2021-03-12")
116 order by transaction.amount desc;
117
118 #Level 03: ejercicio 02
119 •
120 select
121   company.company_name,
122   case when count(transaction.id) > 4 then "More than 4"
123        when count(transaction.id) < 4 then "Less than 4"
124        end as TRANSACCIONES
125 from transaction
126 join company on transaction.company_id = company.id
127 group by company.company_name
128

```

The results grid shows the following data:

company_name	TRANSACCIONES
Arcu LLP	More than 4
Nunc Interdum Incorporated	More than 4
Enim Conditum Ltd	More than 4
Ut Semper Foundation	More than 4
Lorem Eu Incorporated	More than 4
Malesuada PC	More than 4
Non Institute	More than 4
Ac Fermentum Incorporated	Less than 4
Magna A Neque Industries	Less than 4
Fusce Corp.	Less than 4
Convallis In Incorporated	Less than 4
Ante Saeles Nec Foundation	Less than 4
Donec Ltd	Less than 4
Sed Nunc Ltd	Less than 4
Amet Nulla Donec Corporation	Less than 4
Nascetur Ridiculus Mus Inc.	Less than 4
Vestibulum Lorem PC	Less than 4

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

```

27 12:04:54 select company.company_name, company_id, country, case when count(transaction.id) > 4 then "More than 4" when count(transaction.id) < 4 then "Less than 4" end as TRANSACCIONES from transaction join company on transaction.company_id = company.id group by company.company_name; Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'transactions.transaction.company_id' in field list 0.000 sec
28 12:05:22 select company.company_name, company_id, country, case when count(transaction.id) > 4 then "More than 4" when count(transaction.id) < 4 then "Less than 4" end as TRANSACCIONES from transaction join company on transaction.company_id = company.id group by company.company_name; Error Code: 1054. Unknown column 'company.company_id' in field list 0.000 sec
29 12:05:45 select company.company_name, transaction.company_id, company.country, case when count(transaction.id) > 4 then "More than 4" when count(transaction.id) < 4 then "Less than 4" end as TRANSACCIONES from transaction join company on transaction.company_id = company.id group by company.company_name; Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'transactions.transaction.company_id' in field list 0.000 sec
30 12:06:06 select company.company_name, case when count(transaction.id) > 4 then "More than 4" when count(transaction.id) < 4 then "Less than 4" end as TRANSACCIONES from transaction join company on transaction.company_id = company.id group by company.company_name; 100 row(s) returned 0.016 sec / 0.000 sec

```

Objetivo era listar las empresas y clasificarlas según la cantidad de transacciones que han realizado, indicando si tienen "Más de 4" o "Menos de 4" transacciones. Con la función *CASE WHEN* para crear una columna personalizada llamada TRANSACCIONES. Aquí se configura una condición: si el número de transacciones (*COUNT(transaction.id)*) es mayor a 4, el resultado será "More than 4". Si es menor a 4, mostrará "Less than 4". Aplicando el *CASE WHEN* dentro del *SELECT*, lo que permite clasificar los datos directamente en la consulta y al final añadir *END* para finalizar el comando. Conectar las tablas **transaction** y **company** utilizando *JOIN*, relacionando el *company\_id* de la tabla **transaction** con el *id* de la tabla **company**. Por ultimo, *GROUP BY company.company\_name* para agrupar los resultados por el nombre de la empresa y contar sus transacciones. El resultado muestra el nombre de cada empresa y una clasificación indicando si tienen más o menos de 4 transacciones.