

# DATA ANALYTICS – SQL – SPRINT 3

Ana Claudia da Costa

# Data Analytics – SQL – Sprint 3

## Índice

Sprint 3.....	2
Nivel 1 .....	2
Ejercicio 1 .....	2
Ejercicio 2 .....	5
Ejercicio 3 .....	6
Ejercicio 4 .....	9
Nivel 2 .....	10
Ejercicio 1 .....	10
Ejercicio 2 .....	11
Ejercicio 3 .....	12
Nivel 3 .....	13
Ejercicio 1 .....	13
Ejercicio 2 .....	20
Revisión .....	21

## Sprint 3

### Nivel 1

#### Ejercicio 1

Tu tarea es diseñar y crear una tabla llamada "credit\_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después de crear la tabla será necesario que ingreses la información del documento denominado "datos\_introducir\_credit". Recuerda mostrar el diagrama y realizar una breve descripción del mismo.

```
4  -- Nivel 1
5  -- Tu tarea es diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito.
6  -- La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas
7  -- ("transaction" y "company").
8  -- Después de crear la tabla será necesario que ingreses la información del documento denominado "datos_introducir_credit".
9  -- Recuerda mostrar el diagrama y realizar una breve descripción del mismo.
10
11  -- creamos la tabla credit card
12  • CREATE TABLE credit_card (
13    id VARCHAR(20) PRIMARY KEY,
14    iban VARCHAR(50),
15    pan INT,
16    pin VARCHAR(4),
17    cvv INT,
18    expiring_date VARCHAR(20)
19  );
20
21  -- visualizamos las características de la tabla 'credit_card'
22  • DESC credit_card;
```

Output

#	Time	Action	Message	Duration / Fetch
1	10:53:34	CREATE TABLE credit_card (id VARCHAR(20) PRIMARY KEY, iban VARCHAR(50), pa...	0 row(s) affected	0.031 sec
2	10:53:40	DESC credit_card	6 row(s) returned	0.000 sec / 0.000 sec

```
21  -- visualizamos las características de la tabla 'credit_card'
22  • DESC credit_card;
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI		
iban	varchar(50)	YES			
pan	int	YES			
pin	varchar(4)	YES			
cvv	int	YES			
expiring_date	varchar(20)	YES			

Result 3 x

Output

#	Time	Action	Message	Duration / Fetch
4	11:08:07	DESC credit_card	6 row(s) returned	0.000 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

```

27 -- Insertamos datos de credit_card
28 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984
29 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2945', 'D026854763748537475216568689', '5142423821948828', '9080', '8
30 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2952', 'B6451VQL52710525608255', '4556 453 55 5287', '4598', '438', '
31 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2959', 'CR7242477244335841535', '372461377349375', '3583', '667', '02
32 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2966', 'BG72LKTQ15627628377363', '448566 886747 7265', '4900', '130',
33 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2973', 'PT87806228135092429456346', '544 58654 54343 384', '8760', '8
34 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2980', 'DE39241881883086277136', '402400 7145845969', '5075', '596',
35 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2987', 'GE89681434837748781813', '3763 747687 76666', '2298', '797',
36 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2994', 'BH62714428368066765294', '344283273252593', '7545', '595', '0
37 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3001', 'CY49087426654774581266832116', '511722 924833 2244', '9562',

```

```

5029
5030 -- relacionamos la tabla 'credit_card' con la tabla 'transaction' haciendo llave foranea el campo 'credit_card_id' de la tabla 'transaction'
5031
5032 • ALTER TABLE transaction
5033 ADD CONSTRAINT transaction_ibfk_2
5034 FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)
5035 ON DELETE SET NULL;
5036
5037 • DESC transaction;

```

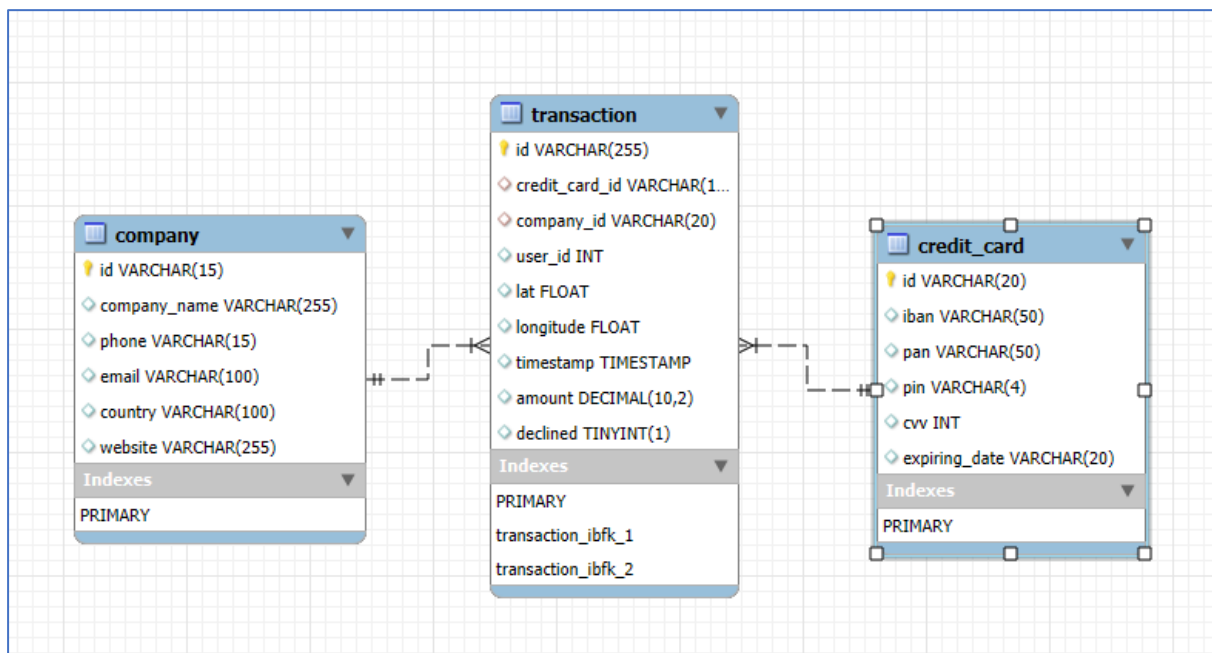
Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	

Result 12 x

Output

#	Time	Action	Message	Duration / Fetch
5034	12:57:01	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9580', 'X...	1 row(s) affected	0.000 sec
5035	12:57:01	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9581', 'X...	1 row(s) affected	0.000 sec
5036	12:57:27	ALTER TABLE transaction ADD CONSTRAINT transaction_ibfk_2 FOREIGN KEY (credi...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0	2.421 sec

## Diagrama ER



En el diagrama de la base de datos Transactions ahora hay 3 tablas.

Author: Ana Cláudia da Costa

## Data Analytics – SQL – Sprint 3

Hemos agregado la tabla 'credit\_card' con los campos id varchar(20), iban varchar(50), pan varchar(50), pin varchar(4), cvv int y expiring\_date varchar(20)

Luego hemos definido la relación de la tabla 'credit\_card' con la tabla 'transaction' a través de la PK 'id' que es FK en la tabla 'transaction'.

La relación es de 1 (credit\_card) a N (transaction).

# Data Analytics – SQL – Sprint 3

## Nivel 1

### Ejercicio 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta asociado a su tarjeta de crédito con ID CcU-2938. La información que debe mostrarse para este registro es: TR323456312213576817699999. Recuerda mostrar que el cambio se realizó.

The screenshot displays a SQL IDE interface with two panels showing the execution of SQL queries to update a credit card record.

**Top Panel (Query 1):**

```
5038 -- Ejercicio 2
5039 -- actualizar informacion en un registro determinado
5040
5041 -- consultamos el registro del id CcU-2938 en la tabla 'credit_card'
5042 • SELECT *
5043 FROM credit_card
5044 WHERE id = 'CcU-2938';
```

The **Result Grid** shows the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22

The **Output** panel shows the following action output:

#	Time	Action	Message	Duration / Fetch
5037	12:58:12	DESC transaction	9 row(s) returned	0.000 sec / 0.000 sec
5038	13:48:33	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

**Bottom Panel (Query 2):**

```
5046 -- actualizamos el campo 'iban' con el dato que se pide: TR323456312213576817699999
5047 • UPDATE credit_card
5048 SET iban = 'TR323456312213576817699999'
5049 WHERE id = 'CcU-2938'
5050 ;
5051
5052 -- consultamos el registro del id CcU-2938 en la tabla 'credit_card' para verificar la actualización
5053 • SELECT *
5054 FROM credit_card
5055 WHERE id = 'CcU-2938';
5056
```

The **Result Grid** shows the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR323456312213576817699999	5424465566813633	3257	984	10/30/22

The **Output** panel shows the following action output:

#	Time	Action	Message	Duration / Fetch
5039	13:50:02	UPDATE credit_card SET iban = 'TR323456312213576817699999' WHERE id = 'CcU-2...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
5040	13:50:02	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

## Nivel 1

### Ejercicio 3

En la tabla "transaction" ingresa un nuevo usuario con la siguiente información:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lato	829.999
longitud	-117.999
amunt	111.11
declined	0

The screenshot shows a SQL IDE interface. The top pane contains a SQL script with the following content:

```
5060 -- no se podría ingresar los datos pues credit_card_id y company_id son llaves foraneas y no existen en sus respectivas tablas.
5061 -- así que primero creamos los datos en sus respectivas tablas.
5062
5063 -- insertamos el id 'b-9999' en la tabla company
5064 • INSERT INTO company
5065   (id, company_name, phone, email, country, website)
5066   VALUES
5067   ('b-9999', 'TestCompany', '06 85 56 52 33', 'test.company@test.com', 'TestCountry', 'https://test.com/site');
5068
```

The bottom pane displays the results of the execution. The 'Result Grid' shows a single row of data:

id	company_name	phone	email	country	website
b-9999	TestCompany	06 85 56 52 33	test.company@test.com	TestCountry	https://test.com/site

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
5056	11:54:53	INSERT INTO company (id, company_name, phone, email, country, website) VALUES (...)	1 row(s) affected	0.016 sec
5057	11:55:00	SELECT * FROM company WHERE id = 'b-9999'	1 row(s) returned	0.000 sec / 0.000 sec

El único dato que tenemos de la tabla 'company' es el id entonces he elegido rellenar las demás columnas como si fuera un usuario de test con datos no reales.

Data Analytics – SQL – Sprint 3

```

5071 -- consultamos a ver que se ha ingresado correctamente
5072 • SELECT *
5073 FROM company
5074 WHERE id = 'b-9999';
5075

```

id	company_name	phone	email	country	website
b-9999	TestCompany	06 85 56 52 33	test.company@test.com	TestCountry	https://test.com/site

company 29 x

Output

#	Time	Action	Message	Duration / Fetch
94	13:36:08	SELECT * FROM company WHERE id = b-9999'	1 row(s) returned	0.141 sec / 0.000 sec

[illegible]

```
5085 -- finalmente insertamos los datos en la tabla transaction
5086 • INSERT INTO transaction
5087 (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
5088 VALUES
5089 ('10881D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
5090
5091 -- consultamos la inserción en la tabla transaction
5092 • SELECT *
5093 FROM transaction
5094 WHERE credit_card_id = 'CcU-9999';
```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
10881D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	111.11	0	

transaction 25

#	Time	Action	Message	Duration / Fetch
5061	11:59:07	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)	1 row(s) affected	0.015 sec
5062	11:59:12	SELECT * FROM transaction WHERE credit_card_id = 'CcU-9999'	1 row(s) returned	0.000 sec / 0.000 sec

Veo que el campo 'timestamp' tiene valor NULL. Procedo a actualizarlo para que pase a considerar el timestamp de la hora del ingreso de la transacion.

Author: Ana Cláudia da Costa



# Data Analytics – SQL – Sprint 3

```
5106 -- actualizamos la información de timestamp en el registro
5107 • UPDATE transaction SET timestamp = CURRENT_TIMESTAMP WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';
5108
5109 -- consultamos la actualización
5110 • SELECT *
5111 FROM transaction
5112 WHERE credit_card_id = 'CcU-9999';
5113
```

Result Grid

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
10881D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	2025-09-30 13:12:59	111.11	0

transaction 52 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
133	13:12:59	UPDATE transaction SET timestamp = CURRENT_TIMESTAMP WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
134	13:14:07	SELECT * FROM transaction WHERE credit_card_id = 'CcU-9999';	1 row(s) returned	0.000 sec / 0.000 sec

```
5111 -- actualizamos la columna 'timestamp' para que no reciba NULL pero sí la fecha y hora de ingreso de la transacción en la base de datos.
5112 -- el cambio pasa a ser valido a partir de esta actualización.
5113 • ALTER TABLE transaction
5114 MODIFY COLUMN timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP;
5115 -- consulto la actualización
5116 • DESC transaction;
5117 --
```

Result Grid

Field	Type	Null	Key	Default	Extra
user_id	int	YES	MUL	NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	NO		CURRENT_TIMESTAMP	DEFAULT GENERATED
amount	decimal(10,2)	YES		NULL	

Result 53 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
135	13:21:02	ALTER TABLE transaction MODIFY COLUMN timestamp TIMESTAMP NOT NULL DEF...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	5.719 sec
136	13:21:34	DESC transaction	9 row(s) returned	0.000 sec / 0.000 sec

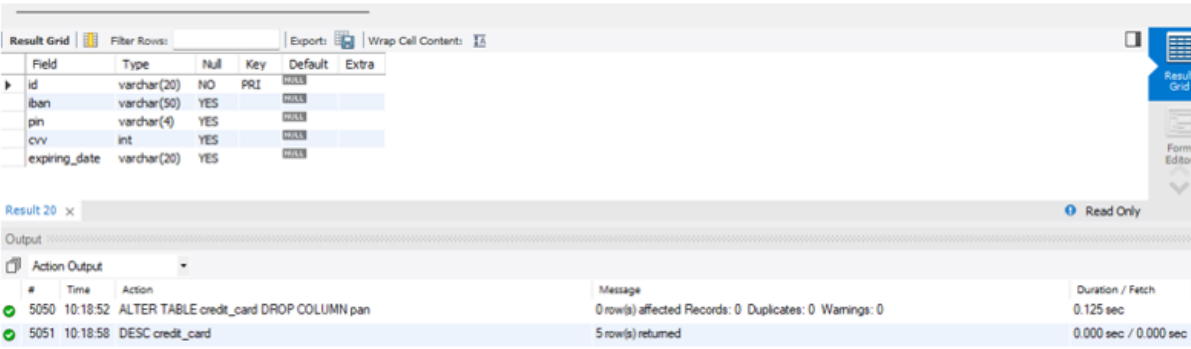
# Data Analytics – SQL – Sprint 3

## Nivel 1

### Ejercicio 4

Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit\_card. Recuerda mostrar el cambio realizado.

```
5100 -- Ejercicio 4
5101 -- eliminar la columna "pan" de la tabla 'credit_card'.
5102
5103 -- consulto las columnas de la tabla 'credit_card'
5104 • DESC credit_card;
5105
5106 -- elimino de la tabla la columna 'pan'
5107 • ALTER TABLE credit_card
5108   DROP COLUMN pan;
5109
5110 -- consulto las columnas de la tabla 'credit_card' para certificame del cambio.
5111 • DESC credit_card;
5112
```



The screenshot displays a SQL IDE interface. At the top, a code editor shows a series of SQL commands. Below the editor, a 'Result Grid' shows the structure of the 'credit\_card' table after the 'pan' column has been dropped. The table has five columns: 'id', 'iban', 'pin', 'cvv', and 'expiring\_date'. The 'id' column is the primary key. Below the result grid, an 'Output' pane shows the execution log, including the 'ALTER TABLE' command and the 'DESC' command, both of which executed successfully.

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI		
iban	varchar(50)	YES			
pin	varchar(4)	YES			
cvv	int	YES			
expiring_date	varchar(20)	YES			

#	Time	Action	Message	Duration / Fetch
5050	10:18:52	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.125 sec
5051	10:18:58	DESC credit_card	5 row(s) returned	0.000 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

## Nivel 2

### Ejercicio 1

Elimina de la tabla transacción el registro con ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de datos.

**Transaction 21:**

```
5090 -- verificación del id
5091 • SELECT *
5092 FROM transaction
5093 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
5094
```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	Ccs-5019	b-2370	438	41.5972	12.2218	2016-12-21 20:07:18	155.63	0

**Transaction 22:**

```
5095 -- eliminación de la línea
5096 • DELETE FROM transaction
5097 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
5098
5099 -- verificación de la eliminación
5100 • SELECT *
5101 FROM transaction
```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined

## Nivel 2

### Ejercicio 2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesaria que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Media de compra realizado por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

```
5156 -- creamos la vista
5157 • CREATE OR REPLACE VIEW VistaMarketing AS
5158 SELECT c.company_name, c.phone, c.country, ROUND (AVG(t.amount), 2) AS purchase_average
5159 FROM company AS c
5160 LEFT JOIN transaction AS t
5161 ON c.id = t.company_id
5162 GROUP BY c.company_name, c.phone, c.country
5163 ORDER BY purchase_average DESC;
5164
5165 -- consulto la vista creada
5166 • SELECT *
5167 FROM vistamarketing;
```

company_name	phone	country	purchase_average
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Pretium Neque Corp.	07 77 48 55 28	Australia	276.16
Urna Convallis Associates	06 01 24 77 04	United States	274.24
At Associates	09 56 61 10 65	New Zealand	272.21
Metus Vitae Associates	08 25 44 40 66	Australia	270.08

vistamarketing 54 x

Output

#	Time	Action	Message	Duration / Fetch
140	13:46:57	CREATE OR REPLACE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c...	0 row(s) affected	0.000 sec
141	13:47:04	SELECT * FROM vistamarketing	101 row(s) returned	0.500 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

## Nivel 2

### Ejercicio 3

Filtra la vista VistaMarketing para mostrar sólo las compañías que tienen su país de residencia en "Germany".

The screenshot shows a SQL IDE interface. At the top, a query is entered in a text area:

```
5141 -- filtro pays 'Germany' en la VistaMarketing
5142
5143 • SELECT *
5144 FROM vistamarketing
5145 WHERE country = 'Germany';
5146
```

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

company_name	phone	country	media_ventas
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Nunc Interdum Incorporated	05 18 15 48 13	Germany	259.32
Convalis In Incorporated	06 66 57 29 50	Germany	257.75
Ac Industries	09 34 65 40 60	Germany	255.15
Rutrum Non Inc.	02 66 31 61 09	Germany	255.14
Auctor Mauris Corp.	05 62 87 14 41	Germany	254.77
Auque Foundation	06 88 43 15 63	Germany	253.51

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

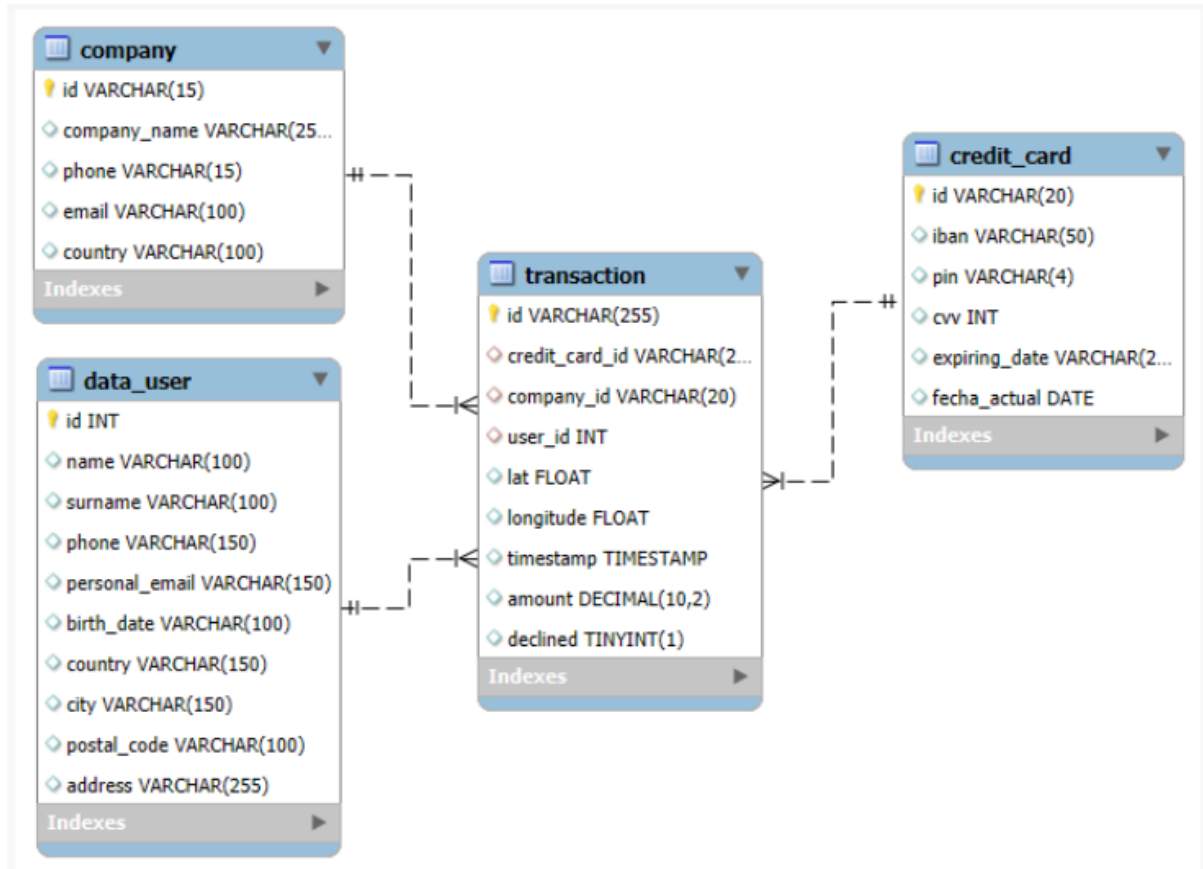
#	Time	Action	Message	Duration / Fetch
5064	12:52:20	CREATE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c.country, ROU...	0 row(s) affected	0.032 sec
5065	13:02:37	SELECT * FROM vistamarketing WHERE country = 'Germany'	8 row(s) returned	0.047 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

## Nivel 3

### Ejercicio 1

La próxima semana tendrás una nueva reunión con los gerentes de marketing. Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Te pide que le ayudes a dejar los comandos ejecutados para obtener el siguiente diagrama:



```
5149 -- crear la tabla 'user'
5150 CREATE TABLE IF NOT EXISTS user (
5151     id CHAR(10) PRIMARY KEY,
5152     name VARCHAR(100),
5153     surname VARCHAR(100),
5154     phone VARCHAR(150),
5155     email VARCHAR(150),
5156     birth_date VARCHAR(100),
5157     country VARCHAR(150),
5158     city VARCHAR(150),
5159     postal_code VARCHAR(100),
5160     address VARCHAR(255)
5161 );
5162
```

Output

#	Time	Action	Message	Duration / Fetch
5065	13:02:37	SELECT * FROM vistamarketing WHERE country = 'Germany'	8 row(s) returned	0.047 sec / 0.000 sec
5066	13:28:02	CREATE TABLE IF NOT EXISTS user (id CHAR(10) PRIMARY KEY, name VARCHAR(100), surname VARCHAR(100), phone VARCHAR(150), email VARCHAR(150), birth_date VARCHAR(100), country VARCHAR(150), city VARCHAR(150), postal_code VARCHAR(100), address VARCHAR(255))	0 row(s) affected	0.078 sec

# Data Analytics – SQL – Sprint 3

```
5163 -- insertamos los datos en la tabla 'user'
5164 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5165 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5166 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5167 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5168 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5169 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5170 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5171 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5172 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5173 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5174 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5175 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5176 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5177 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5178 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5179 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5180 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5181 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
```

Output

#	Time	Action	Message	Duration / Fetch
10065	13:30:27	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address)	1 row(s) affected	0.000 sec
10066	13:30:27	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address)	1 row(s) affected	0.000 sec

```
10185 -- actualizamos el nombre de la tabla 'user' a 'data_user'
10186 • RENAME TABLE user TO data_user;
10187
```

Output

#	Time	Action	Message	Duration / Fetch
10072	13:44:07	ALTER TABLE user TO data_user	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 'user TO data_user' at line 1	0.032 sec
10073	13:45:24	RENAME TABLE user TO data_user	0 row(s) affected	0.031 sec

```
10167 -- alteramos el tipo del campo 'id' de la tabla 'data_user' para que coincida con el tipo del campo 'user_id' de la tabla 'transaction'
10168 • ALTER TABLE data_user
10169 MODIFY COLUMN id INT;
10170
```

Output

#	Time	Action	Message	Duration / Fetch
42	09:47:02	ALTER TABLE data_user MODIFY COLUMN id INT	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0	0.266 sec

Ha fallado el intento de crear la FK en la tabla de hechos así que paso a revisar si coinciden la cantidad de usuarios en las tablas 'data\_user' y 'transaction'.

```
10174 -- consultamos cuántos 'id' hay en la tabla 'data_user'
10175 • SELECT COUNT(id) AS cantidad_usuarios
10176 FROM data_user
10177 ;
10178
```

Result Grid

cantidad_usuarios
5000

Output

#	Time	Action	Message	Duration / Fetch
25	09:23:57	SELECT COUNT(id) AS cantidad_usuarios FROM data_user	1 row(s) returned	0.047 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

```
l0179 -- consultamos cuantos 'user_id' hay en la tabla 'transaction'
l0180 • SELECT COUNT(DISTINCT t.user_id) AS cantidad_usuarios_transaction
l0181 FROM transaction AS t;
l0182
```

Result Grid

cantidad_usuarios_transaction
5001

Result 18 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
26	09:24:30	SELECT COUNT(id) AS cantidad_usuarios FROM data_user	1 row(s) returned	0.016 sec / 0.000 sec
27	09:25:25	SELECT COUNT(DISTINCT t.user_id) AS cantidad_usuarios_transaction FROM transaci...	1 row(s) returned	0.094 sec / 0.000 sec

Veo que en la tabla 'transaction' hay un registro a un usuario no coincidente en la tabla 'data\_user'. Es la transacción que se insertó anteriormente y que tiene como user\_id el usuario 9999 que no existe en la tabla 'data\_user'.

```
l0183 -- consultamos el 'id' que solo aparece en la tabla 'transaction'
l0184 • SELECT t.user_id
l0185 FROM transaction AS t
l0186 WHERE t.user_id NOT IN
l0187 (
l0188     SELECT id
l0189     FROM data_user AS d
l0190 )
l0191 ;
l0192
```

Result Grid

user_id
9999

transaction 19 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
27	09:25:25	SELECT COUNT(DISTINCT t.user_id) AS cantidad_usuarios_transaction FROM transaci...	1 row(s) returned	0.094 sec / 0.000 sec
28	09:26:31	SELECT t.user_id FROM transaction AS t WHERE t.user_id NOT IN ( SELECT id FROM...	1 row(s) returned	0.047 sec / 0.000 sec

Así que pasamos a crear los datos en la tabla 'data\_user' con las mismas características de las tablas 'company' y 'credit\_card': usuario de test, datos no reales.

```
l0193 -- creamos el user de test '9999' ya que se ha insertado una transaccion para este usuario que no existe en la tabla 'user'
l0194 • INSERT INTO data_user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "9999", "TesteTest", "Test
l0195 |
l0196
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
28	09:26:31	SELECT t.user_id FROM transaction AS t WHERE t.user_id NOT IN ( SELECT id FROM...	1 row(s) returned	0.047 sec / 0.000 sec
29	09:27:21	INSERT INTO data_user (id, name, surname, phone, email, birth_date, country, city, post...	1 row(s) affected	0.016 sec



# Data Analytics – SQL – Sprint 3

```
l0226 -- consultamos el usuario creado
l0227 • SELECT *
l0228 FROM data_user
l0229 WHERE id = '9999';
l0230
l0231
l0232 -- definimos la relación entre la tabla 'user' y la tabla 'transaction creando la constraint 'fk_transaction_to_datauser'
```

id	name	surname	phone	personal_email	birth_date	country	city	postal_code	address
9999	TesteTest	Test	+99-999-9999	test.testesw@test.com	Jan 01, 1999	Test States	Test	99999	999 Rmuqvxgw St

data\_user 56 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
142	22:50:34	SELECT * FROM data_user WHERE id = '9999'	1 row(s) returned	0.235 sec / 0.000 sec

```
l0210 -- definimos la relación entre la tabla 'user' y la tabla 'transaction creando la constraint 'fk_transaction_to_datauser'
l0211 -- el campo 'id' de la tabla 'user' será FK en la tabla 'transaction'
l0212 • ALTER TABLE transaction
l0213 ADD CONSTRAINT fk_transaction_to_datauser
l0214 FOREIGN KEY (user_id) REFERENCES data_user(id)
l0215 ON DELETE SET NULL;
l0216
l0217
l0218 -- consultamos los datos de la tabla 'transaction' para conferir la FK creada
l0219 • DESC transaction;
```

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES	MUL	NULL	
lat	float	YES		NULL	

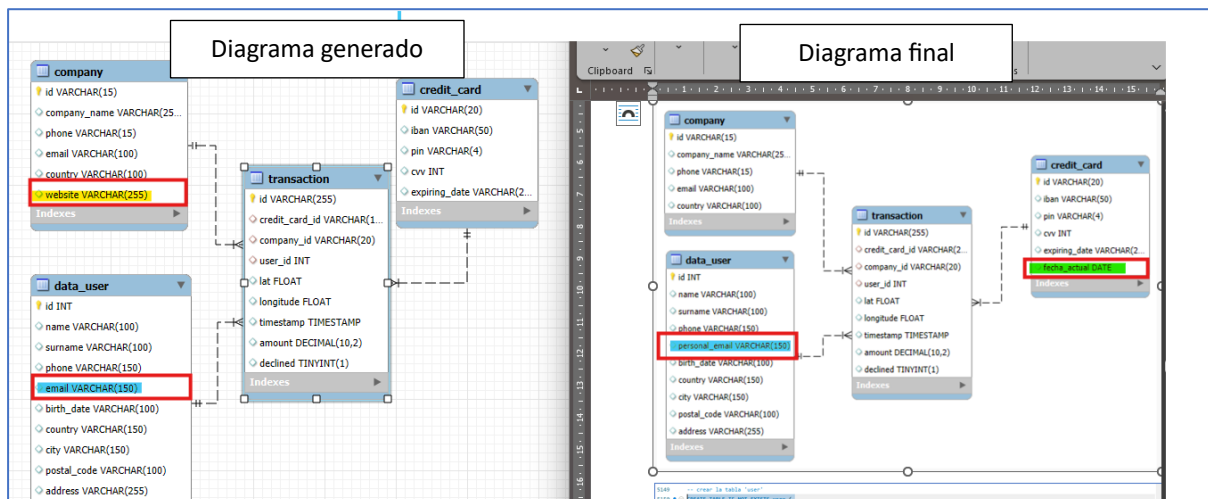
Result 36 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
107	10:36:50	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_to_datauser FOREIGN K...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0	4.407 sec
108	10:37:00	DESC transaction	9 row(s) returned	0.000 sec / 0.000 sec

Genero el diagrama y lo comparo con el modelo que se pide en el ejercicio.



Paso por detallar lo que se ha identificado:

-Tabla 'company' – la columna 'website' no existe en el diagrama final y se deberá borrar;

Author: Ana Cláudia da Costa

# Data Analytics – SQL – Sprint 3

- Tabla 'credit\_card' – hay que crear la columna 'fecha\_actual' de tipo 'DATE' para que coincida con el diagrama final;
- Tabla 'data\_user' – hay que actualizar el nombre de la columna 'email' para 'personal\_email'.

Ejecuto lo que se ha detectado.

```
l0221 -- consultamos las columnas de la tabla 'company'
l0222 • DESC company;
l0223
```

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
company_name	varchar(255)	YES		NULL	
phone	varchar(15)	YES		NULL	
email	varchar(100)	YES		NULL	
country	varchar(100)	YES		NULL	
website	varchar(255)	YES		NULL	

Result 38 x

#	Time	Action	Message	Duration / Fetch
109	11:04:24	DESC company	6 row(s) returned	0.015 sec / 0.000 sec

```
l0224 -- borramos de la tabla 'company' la columna 'website'
l0225 • ALTER TABLE company
l0226 DROP COLUMN website;
l0227
l0228 -- consultamos para comprobar las columnas de la tabla company
l0229 • DESC company;
l0230
```

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
company_name	varchar(255)	YES		NULL	
phone	varchar(15)	YES		NULL	
email	varchar(100)	YES		NULL	
country	varchar(100)	YES		NULL	

Result 39 x

#	Time	Action	Message	Duration / Fetch
111	11:06:38	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.109 sec
112	11:07:25	DESC company	5 row(s) returned	0.000 sec / 0.000 sec

```
l0231 -- consultamos las columnas de la tabla 'credit_card'
l0232 • DESC credit_card;
l0233
l0234
```

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	YES		NULL	
pin	varchar(4)	YES		NULL	
cvv	int	YES		NULL	
expiring_date	varchar(20)	YES		NULL	

Result 41 x

#	Time	Action	Message	Duration / Fetch
113	11:09:24	DESC credit_card	5 row(s) returned	0.000 sec / 0.000 sec

# Data Analytics – SQL – Sprint 3

```
l0234 -- creamos la columna 'fecha_actual' de tipo 'DATE' en la tabla 'credit_card'
l0235 • ALTER TABLE credit_card
l0236 ADD fecha_actual DATE;
l0237
l0238 -- consultamos las columnas de la tabla 'credit_card' para comprobar que se creo la columna correctamente
l0239 • DESC credit_card;
l0240
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	YES		NULL	
pin	varchar(4)	YES		NULL	
cvv	int	YES		NULL	
expiring_date	varchar(20)	YES		NULL	
fecha_actual	date	YES		NULL	

Result 42

Output

Action Output

#	Time	Action	Message	Duration / Fetch
115	11:19:27	ALTER TABLE credit_card ADD fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.109 sec
116	11:20:02	DESC credit_card	6 row(s) returned	0.000 sec / 0.000 sec

```
l0241 -- consultamos las columnas de la tabla 'data_user'
l0242 • DESC data_user;
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 46

Output

Action Output

#	Time	Action	Message	Duration / Fetch
121	11:25:03	DESC data_user	10 row(s) returned	0.000 sec / 0.000 sec

```
l0244 -- cambiamos el nombre del campo 'email' a 'personal_email'
l0245 • ALTER TABLE data_user
l0246 RENAME COLUMN email TO personal_email;
l0247
l0248 -- consultamos para comprobar el cambio
l0249 • DESC data_user;
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
personal_email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 47

Output

Action Output

#	Time	Action	Message	Duration / Fetch
123	11:26:00	ALTER TABLE data_user RENAME COLUMN email TO personal_email	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
124	11:26:11	DESC data_user	10 row(s) returned	0.000 sec / 0.000 sec

## Data Analytics – SQL – Sprint 3

Vuelvo a generar el diagrama para comparación con el diagrama final.

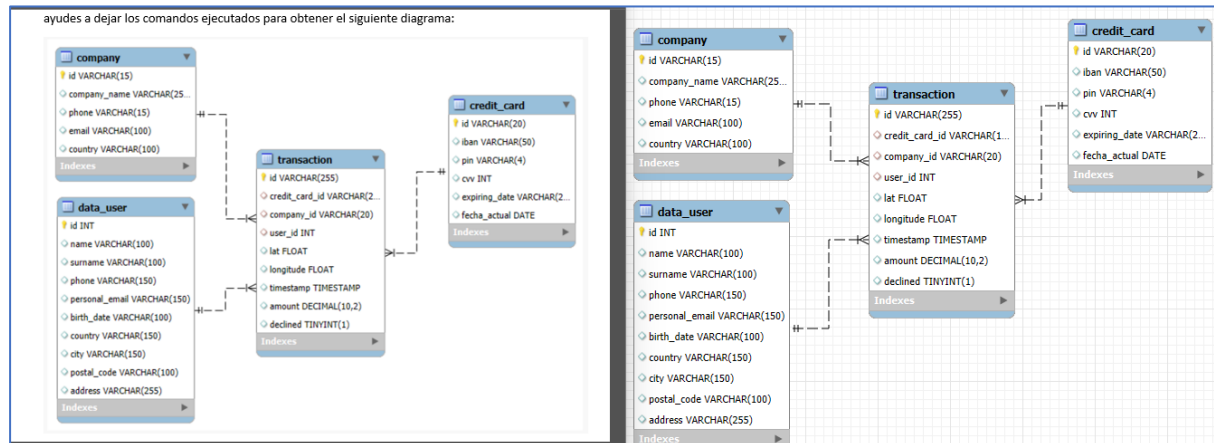
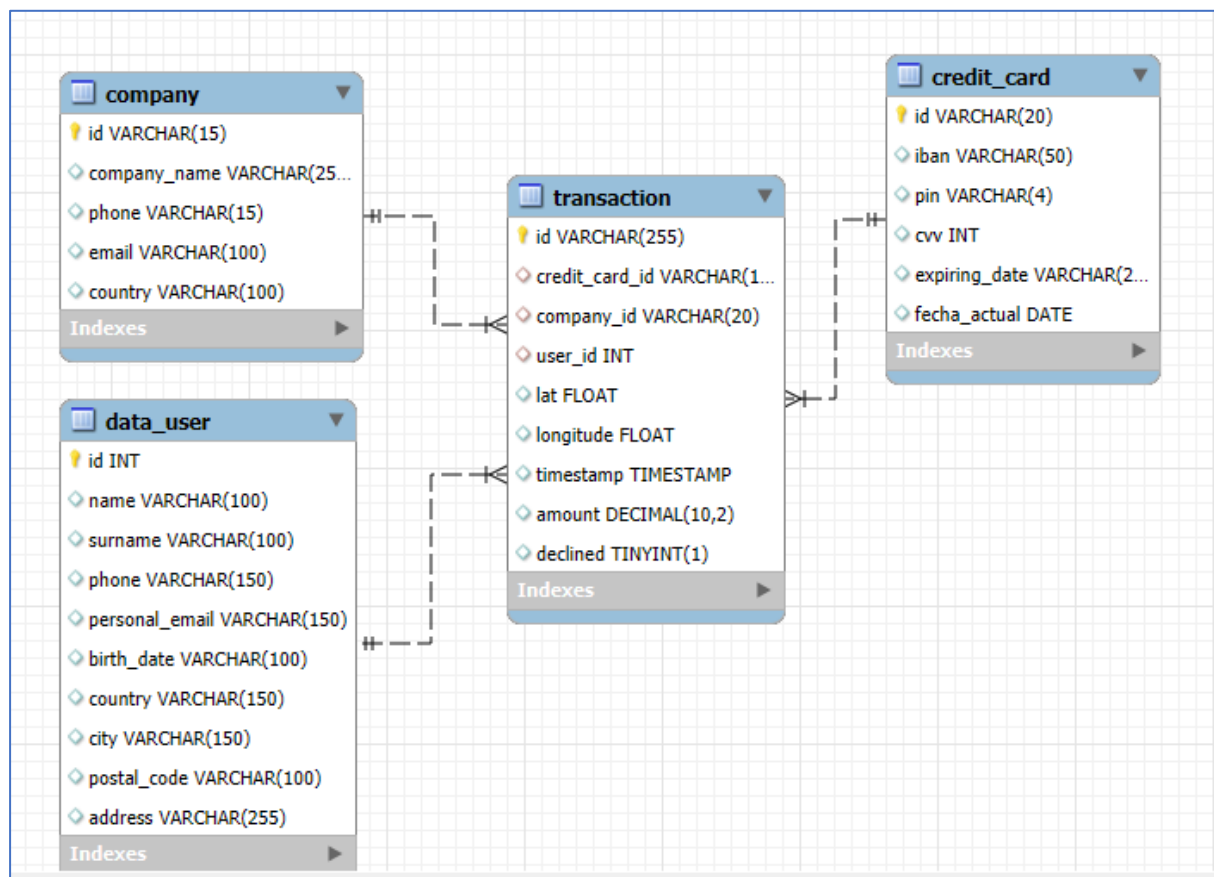


Diagrama final.



En el diagrama vemos la tabla de hechos 'transaction' que tiene 'id' as PK y 'credit\_card\_id', 'company\_id' y 'user\_id' como FK.

La relación es de 1 (tablas company, data\_user y credit\_card) a N (tabla transaction).

# Data Analytics – SQL – Sprint 3

## Nivel 3

### Ejercicio 2

La empresa también le pide crear una vista llamada "InformeTecnico" que contenga la siguiente información:

- ID de la transacción
- Nombre del usuario/a
- Apellido del usuario/a
- IBAN de la tarjeta de crédito usada.
- Nombre de la compañía de la transacción realizada.
- Asegúrese de incluir información relevante de las tablas que conocerá y utilice alias para cambiar de nombre columnas según sea necesario.

Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción.

En mi punto de visto considero importante que un informe técnico también incluya la información de país de usuario, país de la compañía y el importe de la compra.

Hago la ordenación en el SELECT y no en la creación de la vista pues en mi interpretación así lo piden en el ejercicio.

```
10261 • CREATE VIEW Informe_Tecnico AS
10262     SELECT t.id AS transaction_id, d.name AS user_name, d.surname AS user_surname, d.country AS user_country,
10263            cc.iban, c.company_name, c.country AS company_country, t.amount AS purchase_amount
10264     FROM transaction AS t
10265     LEFT JOIN data_user AS d
10266         ON t.user_id = d.id
10267     LEFT JOIN credit_card AS cc
10268         ON t.credit_card_id = cc.id
10269     LEFT JOIN company AS c
10270         ON t.company_id = c.id
10271     ;
```

transaction_id	user_name	user_surname	user_country	iban	company_name	company_country	purchase_amount
FFFD31D6-9495-47CE-B54A-7D88E1CC274B	Bmrgli	Tprvvmrc	United Kingdom	XX794814451211289182490922	Turpis Company	Netherlands	74.54
FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfried	Vlqqdl	Netherlands	XX636251701647892036676034	Amet Nulla Donec Corporation	Italy	148.91
FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	Sweden	XX162677143304223631437567	Nunc Interdum Incorporated	Germany	234.22
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzipa	Uirzjuh	Portugal	XX395114267082019952567052	Viverra Donec Foundation	United Kingdom	349.13

informe\_tecnico 48 x

Output

#	Time	Action	Message	Duration / Fetch
126	11:53:13	CREATE VIEW Informe_Tecnico AS SELECT t.id AS transaction_id, d.name AS user_na...	0 row(s) affected	0.031 sec
127	11:53:24	SELECT * FROM informe_tecnico ORDER BY transaction_id DESC	100000 row(s) returned	0.031 sec / 1.157 sec

# Data Analytics – SQL – Sprint 3

```
10274 • SELECT *
10275 FROM informe_tecnico
10276 ORDER BY transaction_id DESC;
10277
```

transaction_id	user_name	user_surname	user_country	iban	company_name	company_country	purchase_amount
FFFD31D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgli	Tprvrmrc	United Kingdom	XX794814451211289182490922	Turpis Company	Netherlands	74.54
FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfrldi	Vlqcdil	Netherlands	XX636251701647892036676034	Amet Nulla Donec Corporation	Italy	148.91
FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	Sweden	XX162677143304223631437567	Nunc Interdum Incorporated	Germany	234.22
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzipa	Uirzjuh	Portugal	XX395114267082019952567052	Viverra Donec Foundation	United Kingdom	349.13

informe\_tecnico 48 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
126	11:53:13	CREATE VIEW Informe_Tecnico AS SELECT t.id AS transaction_id, d.name AS user_na...	0 row(s) affected	0.031 sec
127	11:53:24	SELECT * FROM informe_tecnico ORDER BY transaction_id DESC	100000 row(s) returned	0.031 sec / 1.157 sec

## Revisión

Sugerencia: Cambiar el nombre de la Constraint a un nombre que sea acorde con las tablas para que a simples vista se pueda saber cuales son las tablas que corresponden. (Ramiro Aguilar)

N1E2 – hay que describir que se hace en cada una de las consultas.

N1E3 – describir los pasos y el antes y el después de cada consulta (insertar en los comentarios.)

N1E4 – describir el antes de la consulta.

N2E1 – recortar del pantallazo la consulta que no corresponde al ejercicio. El ejercicio habla de ‘compras’ pero en la consulta está ‘ventas’. Fijarse en lo que se pide.

N2E2 – incluir el SELECT de la vista creada y el idioma en las tablas debería de coincidir.

N3E1 – Fijarse en el modelo, hay campos en las tablas que ya estaban creadas que no coinciden con el modelo final que se pide en el ejercicio.

Describir por que verificas la cantidad de usuarios en las tablas ‘data\_user’ y ‘transaction’.

Falta pantallazo de la creación de la FK.

N3E2 – escribir por que la ordenación de la consulta de la vista está en el SELECT y no en la creación de la vista.

El ejercicio pide evaluar si se incluye más columnas en la vista. Revisar y explicar si se incluye o no.

Revisado por: Vanessa Plaza