

Sprint 3

Manipulacion de Tablas

**Nivell 1****- Exercici 1**

La teva tasca és dissenyar i crear una taula anomenada "credit_card" que emmagatzemi detalls crucials sobre les targetes de crèdit. La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules ("transaction" i "company"). Després de crear la taula serà necessari que ingressis la informació del document denominat "dades_introduir_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

Table: credit_card

Columns:

id	varchar(15)
iban	varchar(34)
pan	varchar(19)
pin	char(4)
cvv	char(3)
expiring_date	varchar(10)

CREATE TABLE credit_card

```
(id VARCHAR (15) PRIMARY KEY,
iban VARCHAR (34),
pan VARCHAR (19),
pin CHAR (4),
cvv CHAR (3),
expiring_date VARCHAR (10),
CONSTRAINT chk_pin_format CHECK (pin REGEXP "[0-9]{4}$"),
CONSTRAINT chk_cvv_format CHECK (cvv REGEXP "[0-9]{3}$")
);
```

Output

#	Time	Action	Message
1	11:24:35	CREATE TABLE credit_card (id VARCHAR (15) PRIMARY KEY, iban VARCHAR (...)	0 row(s) affected

Se crea una tabla llamada credit_card en el esquema transactions.

Se le agregan los campos requeridos (id, iban, pan, pin, cvv, expiring_date) como columnas y se contrastan sus características en el archivo "dades_introduir_credit", para ver que de que se componen. (Se puede comenzar con VARCHAR 255 y luego adaptar a los datos, sin embargo, he optado por primero observar los datos y sus características y luego crear la tabla acorde a sus valores)

Para poder ligar la nueva tabla con las otras existentes a través de su PrimaryKey (credit_card.id), se explora que columna de las otras tablas hace referencia a esta. La tabla transaction se relaciona con su ForeignKey (credit_card_id) con la nueva tabla. Para que se pueda relacionar esta nueva tabla de credit_card con la existente a través de su PrimaryKey, debe tener las mismas características que la ya existente ForeignKey

de transaction. (VARCHAR(15)).

Se tiene en cuenta las características específicas de cada columna, que valores pueden tener, cuantos dígitos serán permitidos, por si serán exclusivamente números o también pueden contener letras, por si son obligatorios u opcionales.

En los campos pin y cvv se le ha aplicado un **check constraint** para garantizar de que solo contengan números del 0-9, además de que cumplan con la longitud obligatoria de 4 y 3.

Se especifica la PrimaryKey, el id de esta tabla, esto garantiza la unicidad de este valor además de que implica la función NOT NULL.

Posteriormente se le introducen todos los datos contenidos en el archivo “dades_introducir_credit”

The screenshot displays a database management interface with the following components:

- Navigator:** Shows the 'credit_card' table structure with columns: id (char(8) PK), iban (varchar(34)), pan (char(19)), pin (char(4)), cvv (char(3)), and expiring_date (varchar(10)).
- Query 1:** A SQL script titled 'datos_introducir_credit' containing 15 INSERT statements for the 'credit_card' table. Each statement provides values for id, iban, pan, pin, cvv, and expiring_date.
- Output:** A table showing the execution results of the queries.

#	Time	Action	Message
551	10:44:10	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4...	1 row(s) affected
552	10:44:10	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-4...	1 row(s) affected

Luego verifico si se han introducido los datos en la tabla de credit_card

23
24
25 • **SELECT ***
26 **FROM credit_card;**
27
28
29
30

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22
	CcU-2945	DO26854763748537475216568689	5142423821948828	9080	887	08/24/23
	CcU-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	06/29/21
	CcU-2959	CR7242477244335841535	372461377349375	3583	667	02/24/23
	CcU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	10/29/24
	CcU-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	01/30/25

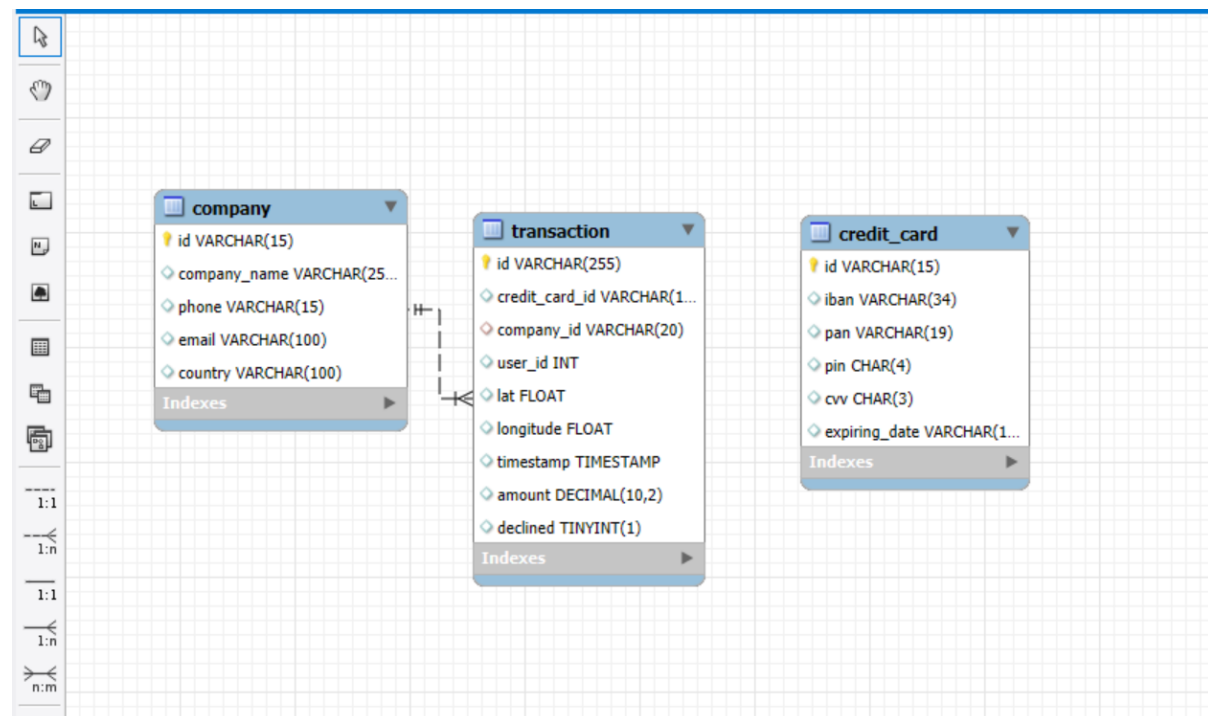
credit_card 2 x Apply

Output

Action Output

#	Time	Action	Message
8	10:27:46	CREATE TABLE transactions.credit_card (id VARCHAR (15) PRIMARY KEY, iban ...	0 row(s) affected
9	10:45:37	SELECT * FROM credit_card LIMIT 0, 50000	275 row(s) returned

El diagrama resultante es:



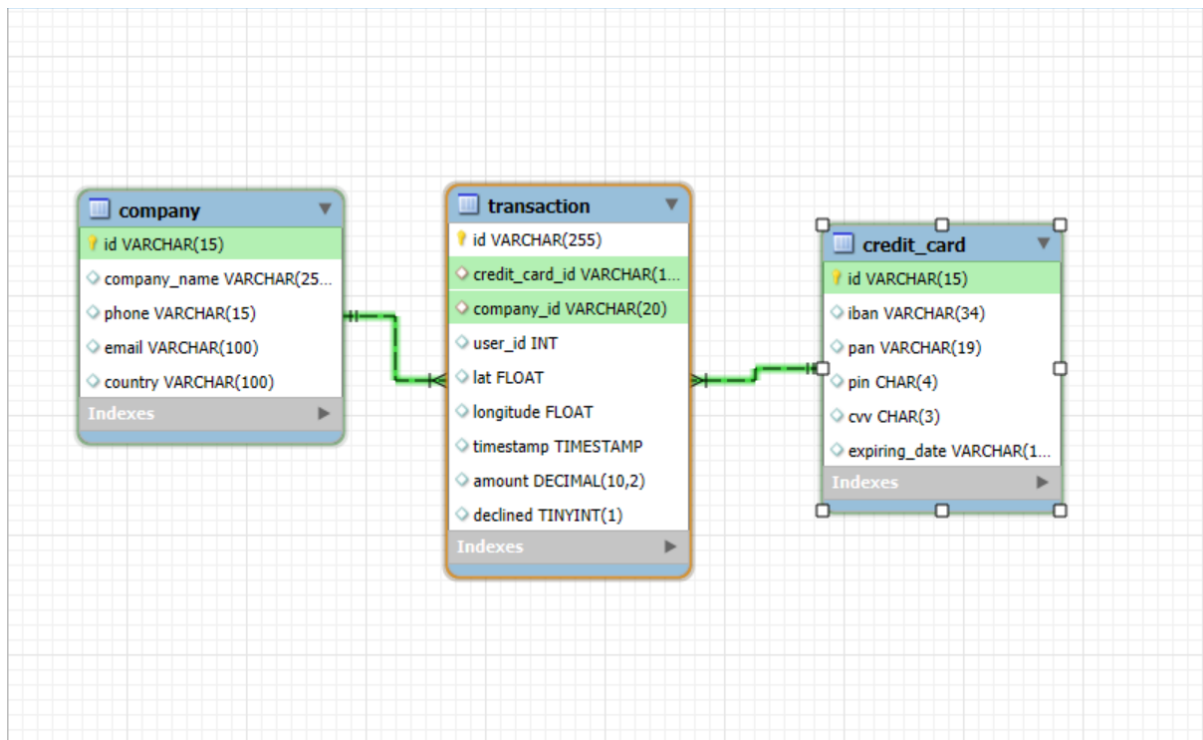
En este diagrama se ve que se ha agregado la tabla credit_card. Ahora se debe entablar la relación correspondiente con la tabla de transacciones.

Se le establece la relación con transaction de la siguiente manera:

```
25
26
27 • ALTER TABLE transaction
28   ADD CONSTRAINT `foreignkeycredit`
29     FOREIGN KEY (`credit_card_id`)
30     REFERENCES `credit_card` (`id`);
31
32
33
34
35
36
37
38
```

Output			
Action Output			
#	Time	Action	Message
✓ 1	13:24:10	ALTER TABLE transaction ADD CONSTRAINT `foreignkeycredit` FOREIGN KE...	586 row(s) affected Records: 586 Duplicates: 0 Wan

Y se llega a este diagrama:



- Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte de l'usuari amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: R323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

Se cambia el registro del iban del usuario con el id CcU-2938, al que inicialmente se le habían asignado estos valores:

31
32 • `SELECT *`
33 `FROM credit_card`
34 `WHERE id = "CcU-2938";`
35
36
37
38

Result Grid

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

credit_card 3 x Apply

Output

Action Output

#	Time	Action	Message
✓ 9	10:45:37	<code>SELECT * FROM credit_card LIMIT 0, 50000</code>	275 row(s) returned
✓ 10	10:50:23	<code>SELECT * FROM credit_card WHERE id = "CcU-2938" LIMIT 0, 50000</code>	1 row(s) returned

33
34
35
36 • `UPDATE credit_card SET iban = "R323456312213576817699999" WHERE id = "CcU-2938";`
37 • `SELECT *`
38 `FROM credit_card`
39 `WHERE id = "CcU-2938";`
40

Result Grid

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	R323456312213576817699999	5424465566813633	3257	984	10/30/22
*	NULL	NULL	NULL	NULL	NULL	NULL

credit_card 4 x Apply

Output

Action Output

#	Time	Action	Message
✓ 11	10:52:43	<code>UPDATE credit_card SET iban = "R323456312213576817699999" WHERE id = "...</code>	1 row(s) affected Rows matched: 1 Changed: 1 Wa
✓ 12	10:52:59	<code>SELECT * FROM credit_card WHERE id = "CcU-2938" LIMIT 0, 50000</code>	1 row(s) returned

Con el comando **UPDATE**, se cambia el número de cuenta de este usuario.

Primero se elige la tabla en la que se quiere realizar el cambio, luego el campo que se quiere "colocar" (**SET**) que en este caso es el iban y se inserta el valor que debe tener, después del =. Además, se tiene que especificar para quien se realiza este cambio, el usuario con el id CcU-2938. Siempre es recomendable usar la Primary-Key, para filtrar el registro en el que se debe efectuar el cambio, para que no se cambien más valores si estos están repetidos, a no ser que sea el propósito.

- Exercici 3

En la taula "transaction" ingressa un nou usuari amb la següent informació:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

```
55 • INSERT INTO company (id)
56   VALUES ("b-9999");
57
58 • INSERT INTO transaction (Id, credit_card_id, company_id, user_id, lat, longitude, amount, decli
59   VALUES ("108B1D1D-5B23-A76C-55EF-C568E49A99DD", "CcU-9999", "b-9999", "9999", "829.999", "-117.
60   111.11, 0);
61 • SELECT *
62   FROM transaction
63   WHERE company_id = "b-9999";
64
```

Result Grid

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount
▶	108B1D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	NULL	111.11
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 15 x Apply

Output

Action Output

#	Time	Action	Message
✓ 6	14:47:11	INSERT INTO transaction (Id, credit_card_id, company_id, user_id, lat, longitud...	1 row(s) affected
✓ 7	14:48:10	SELECT * FROM transaction WHERE company_id = "b-9999" LIMIT 0, 50000	1 row(s) returned

Para poder efectuar esta agregación a la tabla de transacciones, primeramente, hay que agregarle (**INSERT INTO**) a la tabla de company el company_id, ya que esta tabla está conectada con su Primary Key (los ids de las companys) a la tabla de transacciones.

Posteriormente se agregan los otros datos en la tabla de transacciones.

Podemos verificar que en la tabla de transacciones se han agregado los campos requeridos. El timestamp se ha quedado nulo ya que el dato no existe.

- Exercici 4

Des de recursos humans et solliciten eliminar la columna "pan" de la taula credit_card. Recordra mostrar el canvi realitzat.

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane shows a tree view with 'transactions' > 'Tables' > 'credit_card'. The 'Columns' list for 'credit_card' includes 'id', 'iban', 'pin', 'cvv', and 'expiring_date'. The 'Table: credit_card' details show the column types: 'id' (varchar(15) PK), 'iban' (varchar(34)), 'pin' (char(4)), 'cvv' (char(3)), and 'expiring_date' (varchar(10)).

The main editor shows the following SQL commands:

```
62
63
64 • Alter table credit_card drop pan;
65 • SELECT *
66 FROM credit_card;
67
68
69
```

The 'Result Grid' shows the results of the SELECT statement:

id	iban	pin	cvv	expiring_date
CcU-2938	R323456312213576817699999	3257	984	10/30/22
CcU-2945	DO26854763748537475216568689	9080	887	08/24/23
CcU-2952	BG45IVQL52710525608255	4598	438	06/29/21
CcU-2959	CR7242477244335841535	3583	667	02/24/23
CcU-2966	BG72LKTQ15627628377363	4900	130	10/29/24

The 'Output' pane shows the execution log:

#	Time	Action	Message
20	11:03:57	CREATE TABLE transactions.credit_card (id VARCHAR (15) PRIMARY KEY, iban ...	Error Code: 1050. Table 'credit_card' already exist
21	11:04:00	Alter table credit_card drop pan	0 row(s) affected Records: 0 Duplicates: 0 Warn
22	11:04:08	SELECT * FROM credit_card LIMIT 0, 50000	275 row(s) returned



Nivell 2

Exercici 1

Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de dades.

```
72
73 • delete from transaction where id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
74 • SELECT *
75 FROM transaction where id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
76
77
78
79
```

Result Grid

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 7 x Apply

Output

Action Output

#	Time	Action	Message
✓ 22	11:04:08	SELECT * FROM credit_card LIMIT 0, 50000	275 row(s) returned
✓ 23	11:19:01	delete from transaction where id = "02C6201E-D90A-1859-B4EE-88D2986D3B02"	0 row(s) affected
✓ 24	11:19:05	SELECT * FROM transaction where id = "02C6201E-D90A-1859-B4EE-88D2986D3B02"	0 row(s) returned

Aquí eliminamos el registro y luego verificamos que este eliminado.

Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar anàlisi i estratègies efectives. S'ha sol·licitat crear una vista que proporcioni detalls clau sobre les companyies i les seves transaccions. Serà necessària que creïs una vista anomenada VistaMarketing que contingui la següent informació: Nom de la companyia. Telèfon de contacte. País de residència. Mitjana de compra realitzat per cada companyia. Presenta la vista creada, ordenant les dades de major a menor mitjana de compra.

The screenshot displays a database management interface. On the left, a 'SCHEMAS' pane shows a tree view with databases like 'biblioteca', 'db_companies', and 'transactions'. The 'transactions' database is selected, showing its tables and views. The 'vistamarketing' view is highlighted. The main pane shows the SQL script for creating and querying the view. The script includes a 'CREATE VIEW' statement and a 'SELECT' statement. Below the script, a 'Result Grid' shows the output of the query, displaying columns: 'nombrecompañia', 'telefono', 'pais', and 'compramedia'. The output lists five companies with their respective phone numbers, countries, and average purchase amounts. At the bottom, an 'Output' pane shows the execution log, indicating that the view was created successfully and the query returned 101 rows.

```
90 • CREATE VIEW `vistamarketing` AS
91 SELECT company_name as nombrecompañia, phone as telefono, country as pais,
92 round(avg(amount),2) as compramedia
93 FROM company
94 JOIN transaction ON company_id = company.id
95 WHERE declined = 0
96 GROUP BY nombrecompañia, telefono, pais
97 ORDER BY compramedia DESC;
98
99 • SELECT *
100 FROM vistamarketing;
```

nombrecompañia	telefono	pais	compramedia
Eget Ipsum Ltd	03 67 44 56 72	United States	481.86
Sed Id Limited	07 28 18 18 13	United States	477.51
Neque Tellus Incorporated	04 43 18 34 19	Ireland	477.10
Nunc Sit Incorporated	07 28 42 63 63	Norway	461.83
Non Magna LLC	06 71 73 13 17	United Kingdom	458.74

Output:

#	Time	Action	Message
1	15:00:10	CREATE VIEW `vistamarketing` AS SELECT company_name as nombrecompañia...	0 row(s) affected
2	15:00:12	SELECT * FROM vistamarketing LIMIT 0, 50000	101 row(s) returned

Primero se crea la tabla con los resultados que se quieren ver y luego se crea una Vista a la que podemos acceder con SELECT con **CREATE VIEW**.

Exercici 3

Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany"

```
111
112
113
114 • SELECT *
115 FROM Vistamarketing
116 WHERE pais = "Germany";
117
118
119
120
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
nombrecompanyia	telefono	pais	compramedia
Ac Industries	09 34 65 40 60	Germany	396.15
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.99
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.57
Aliquam PC	01 45 73 52 16	Germany	280.34
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90

Vistamarketing 16 x

Output

Action Output

#	Time	Action	Message
7	14:48:10	SELECT * FROM transaction WHERE company_id = "b-9999" LIMIT 0, 50000	1 row(s) returned
8	14:51:40	SELECT * FROM Vistamarketing WHERE pais = "Germany" LIMIT 0, 50000	8 row(s) returned

Desde la tabla creada anteriormente en vistamarketing, se filtran los resultados a través de la condición "Germany".



Nivell 3

Exercici 1

La setmana vinent tindràs una nova reunió amb els gerents de màrqueting. Un company del teu equip va realitzar modificacions en la base de dades, però no recorda com les va realitzar. Et demana que l'ajudis a deixar els comandos executats per a obtenir el següent diagrama:

Primero agrego la tabla user con los registros proporcionados.

SCHEMAS

Filter objects

user

- Columns
 - id
 - name
 - surname
 - phone
 - email
 - birth_date
 - country
 - city
 - postal_code
 - address

Administration Schemas

Information

Table: user

Columns:

- id int PK
- 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)

Limit to 50000 rows

```
109
110
111 • SELECT *
112 FROM user;
113
114
115
116
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: [F1](#)

	id	name	surname	phone	email	birth_date	country	city	postal_o
1	Zeus	Gamble		1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544
2	Garrett	Mcconnell	(718) 257-2412		integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464
3	Claran	Harrison	(522) 598-1365		interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbus	56518
4	Howard	Stafford	1-411-740-3269		ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	77417

user 11 x Apply

Output

Action Output

#	Time	Action	Message
306	11:29:53	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal...	1 row(s) affected
307	11:29:53	SET foreign_key_checks = 1	0 row(s) affected
308	11:30:47	SELECT * FROM user LIMIT 0, 50000	275 row(s) returned

Realizo cambios en la tabla de user:

```
134
135
136
137
138 • ALTER TABLE `user`
139     RENAME TO `data_user`;
140 • ALTER TABLE `data_user`
141     CHANGE COLUMN `email` `personal_email` VARCHAR (100);
142
143 • ALTER TABLE data_user
144     DROP FOREIGN KEY data_user_ibfk_1;
145
146
147
148
149
150
151
152
```

Output

Primero cambio el nombre con el comando **RENAME TO**.

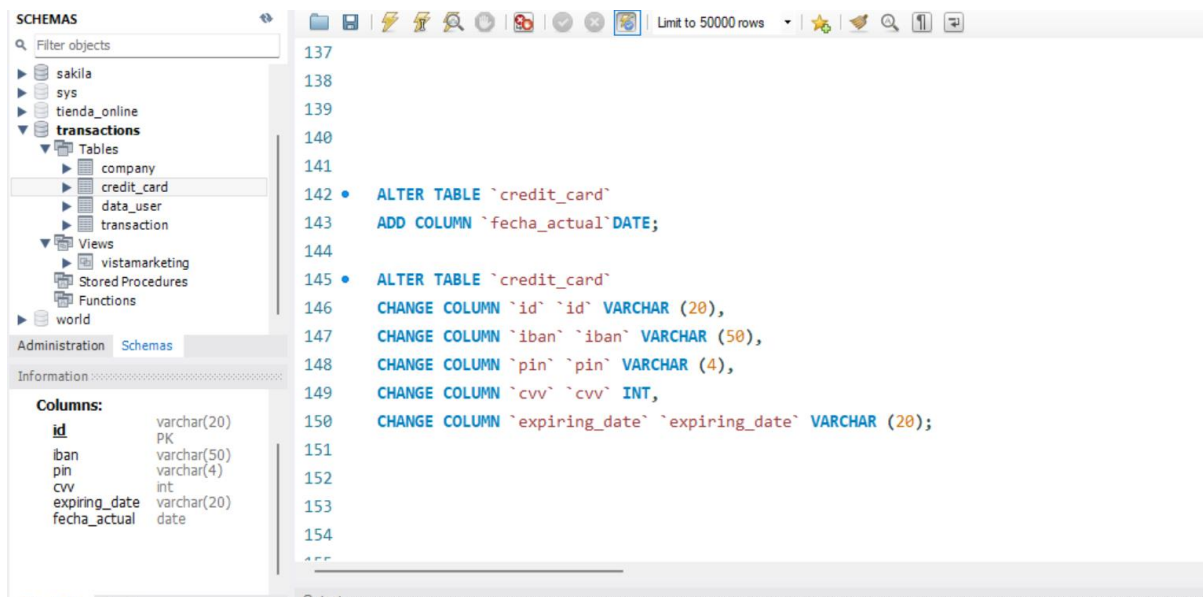
Luego cambio el nombre de email con el comando **CHANGE COLUMN** estableciendo también los nuevos parámetros (VARCHAR(100)).

Luego elimino una Foreign Key que se ha establecido automáticamente entre data_user y transaction (después de haber visto esta relación en Reverse Engineer) con **DROP FOREIGN KEY +** el nombre que se le ha asignado a esta foreign key automáticamente.

Acomodo los datos en la tabla Company:

```
119
120
121
122
123
124
125
126
127
128 • ALTER TABLE `company`
129     DROP COLUMN `website`;
130
131
132
133
134
135
136
137
```

Es borrar la columna website, se hace con **DROP COLUMN**.

Realizo cambios en la tabla de credit_card:

The screenshot shows a database management interface. On the left, a 'SCHEMAS' panel displays a tree view of databases including 'sakila', 'sys', 'tienda_online', 'transactions', and 'world'. The 'transactions' database is expanded, showing tables like 'company', 'credit_card', 'data_user', and 'transaction'. The 'credit_card' table is selected, and its schema is shown in the 'Information' panel below. The schema lists columns: 'id' (varchar(20), PK), 'iban' (varchar(50)), 'pin' (varchar(4)), 'cvv' (int), 'expiring_date' (varchar(20)), and 'fecha_actual' (date). The main area displays SQL queries for altering the 'credit_card' table. The queries are as follows:

```
137
138
139
140
141
142 • ALTER TABLE `credit_card`
143     ADD COLUMN `fecha_actual` DATE;
144
145 • ALTER TABLE `credit_card`
146     CHANGE COLUMN `id` `id` VARCHAR (20),
147     CHANGE COLUMN `iban` `iban` VARCHAR (50),
148     CHANGE COLUMN `pin` `pin` VARCHAR (4),
149     CHANGE COLUMN `cvv` `cvv` INT,
150     CHANGE COLUMN `expiring_date` `expiring_date` VARCHAR (20);
151
152
153
154
155
```

Primero se le agrega una columna con el comando **ADD COLUMN + el nombre de la nueva columna (fecha_actual)** y se establece el formato DATE.

Luego se alteran los data types de la tabla hacia los que están requeridos en este ejercicio con el comando **CHANGE COLUMN + el nombre de la columna + el nuevo nombre de la columna (en este caso es el mismo) + el nuevo datatype de la columna**.

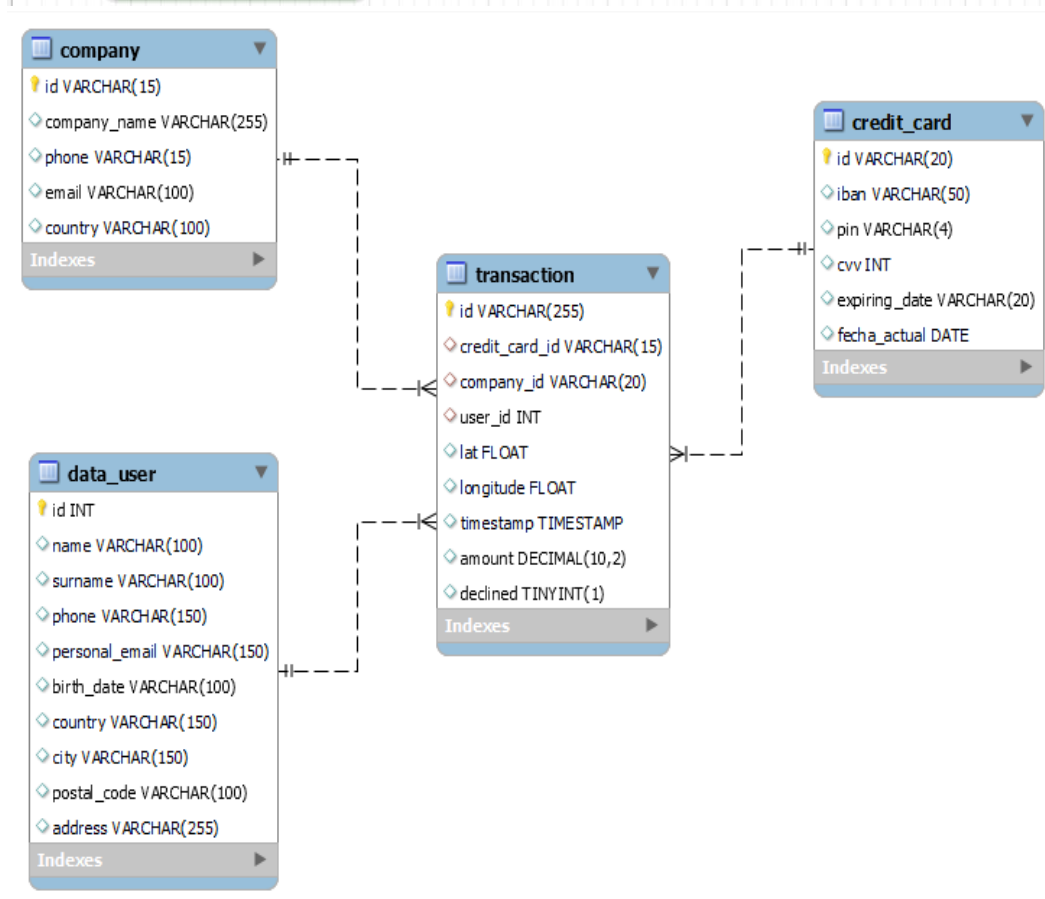
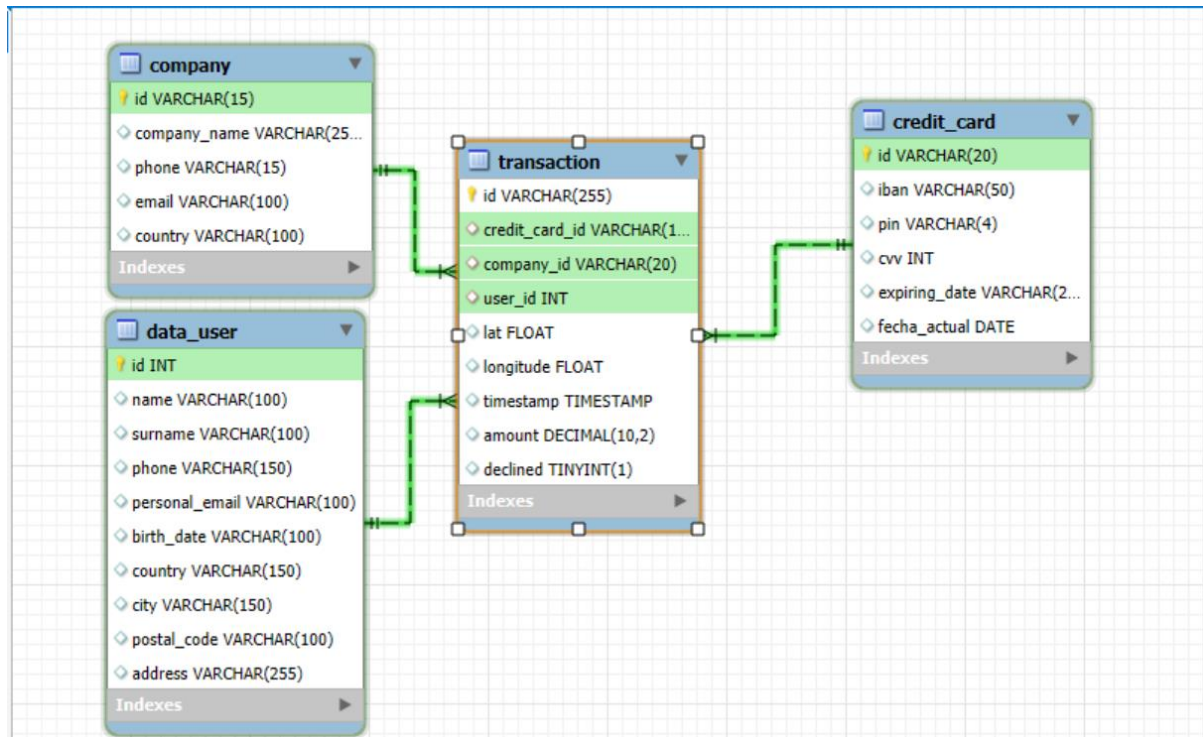
Realizo cambios en la tabla de transaction:

```
165
166
167
168
169
170
171
172 • ALTER TABLE `transaction`
173     ADD CONSTRAINT `foreignkeyuser`
174     FOREIGN KEY (`user_id`)
175     REFERENCES `data_user`(`id`);
176
177
178
179
180
181
182
183
```

Aqui agrego una FOREIGN KEY para que se relacione con la tabla de data_user.

Se hace con el comando **ADD CONSTRAINT + el nombre de la foreign key que eligo + FOREIGN KEY (la columna de la tabla que es la foreign key) + REFERENCES la tabla con la que se quiere entablar la relación (la columna a la que hace referencia).**

Llegamos a este diagrama:



(La referencia)

Se obvia la impresión de los outputs ya que en la tabla se puede ver que han sido exitosos.

Exercici 2

L'empresa també et sol·licita crear una vista anomenada "InformeTecnico" que contingui la següent informació:

- ID de la transacció
- Nom de l'usuari/ària
- Cognom de l'usuari/ària
- IBAN de la targeta de crèdit usada.
- Nom de la companyia de la transacció realitzada.
- Assegura't d'incloure informació rellevant de totes dues taules i utilitza àlies per a canviar de nom columnes segons sigui necessari.

Mostra els resultats de la vista, ordena els resultats de manera descendent en funció de la variable ID de transaction.

The screenshot shows a database management interface with a 'SCHEMAS' panel on the left and a main editor area. The 'SCHEMAS' panel shows a tree structure with 'transactions' expanded, containing 'Views' and 'Stored Procedures'. The main editor area shows the SQL code for creating the view 'informetecnico' and the results of its execution.

SQL Code:

```

160
161 • CREATE VIEW `informetecnico` AS
162 SELECT transaction.id as Idtransaccion, name as NombreUsuario, surname as Apellido,
163 iban, company_name AS nombrecompañia
164 FROM transaction
165 JOIN company ON company_id = company.id
166 JOIN data_user ON user_id = data_user.id
167 JOIN credit_card ON credit_card_id = credit_card.id
168 ORDER BY Idtransaccion DESC;
169 • SELECT*
170 FROM informetecnico;
  
```

Result Grid:

Idtransaccion	NombreUsuario	Apellido	iban	nombrecompañia
FE96CE47-8D59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Ind.
FE809ED4-2DB6-55AC-C915-929516E46468	Molly	Gilliam	SE2813123487163628531121	Nunc Interdum Incon
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incon
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC

Action Output:

#	Time	Action	Message
1	15:01:56	CREATE VIEW `informetecnico` AS SELECT transaction.id as Idtransaccion, na...	0 row(s) affected
2	15:01:59	SELECT* FROM informetecnico LIMIT 0, 50000	586 row(s) returned

Aquí se visualizan todos los datos que deben incluirse en el InformeTecnico, además de que se le asignan nombres más explicativos. Luego se crea una nueva Vista con **CREATE VIEW**.