

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.

```
1 • USE transactions;
2 • CREATE TABLE IF NOT EXISTS credit_card (
3     id VARCHAR(10) PRIMARY KEY,
4     iban VARCHAR(35),
5     pan VARCHAR (35),
6     pin CHAR (4),
7     cvv CHAR(3),
8     expiring_date VARCHAR (8));
9

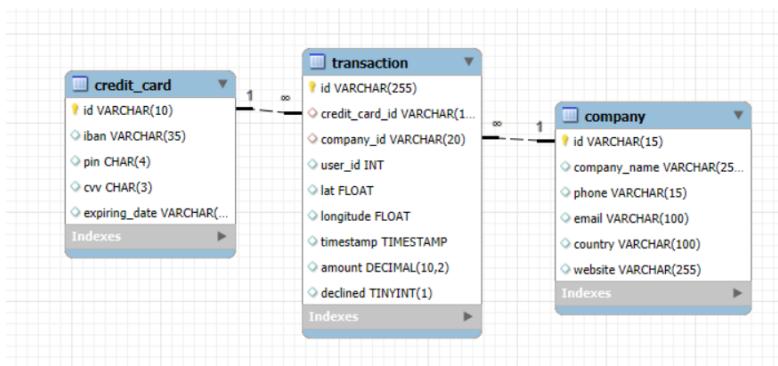
Output
Action Output
# Time Action Message
1 14:08:40 CREATE TABLE IF NOT EXISTS credit_card ( id VARCHAR(10) PRIMARY KEY, iban VARCHAR(35), pa... 0 row(s) affected

10 • ALTER TABLE transaction
11     ADD CONSTRAINT fk_transaction_credit_card_id
12         FOREIGN KEY (credit_card_id)
13             REFERENCES credit_card(id);
14
15

Output
Action Output
# Time Action Message
1 14:14:15 ALTER TABLE transaction ADD CONSTRAINT fk_transaction_credit_card_id FOREIGN KEY (credit_card_id)... 100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
```

Nota 1: Se creó la tabla 'credit_card' con las columnas:

id como Primary Key, identifica de forma única cada tarjeta, iban, pan, pin, cvv, expiring_date. Se añadió una Foreign Key en la tabla 'transaction' relacionando credit_card_id con id de la nueva tabla, lo que asegura la relación entre las transacciones y las tarjetas de crédito.



Nota 2: El diagrama muestra un modelo estrella donde la tabla 'transaction' funciona como tabla de hechos, ya que guarda las transacciones, las tablas 'credit_card' y 'company' son tablas de dimensiones, porque aportan detalles adicionales a cada transacción. Así, cada transacción se relaciona con una tarjeta y con una empresa.

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 shows the MySQL Workbench interface. A script window contains the following code:

```
16  # Ejercicio 2
17  UPDATE credit_card SET iban = 'TR323456312213576817699999'
18  WHERE id = 'CcU-2938';
19
20 •   SELECT iban
21   FROM credit_card
22  WHERE id = 'CcU-2938';
23
```

Below the script is a Result Grid showing the output of the SELECT query:

iban
TR323456312213576817699999

At the bottom, the Output pane displays the log of actions:

#	Time	Action	Message
1	19:17:07	UPDATE credit_card SET iban = 'TR323456312213576817699999' WHERE id = 'CcU-2938'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
2	19:17:12	SELECT iban FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned

Ejercicio 3

En la tabla "transaction" ingresa una nueva transacción con la siguiente información:

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

The screenshot shows the MySQL Workbench interface. A script window contains the following code:

```
20  # Ejercicio 3
21 •  INSERT INTO company (id) VALUES ('b-9999');
22 •  INSERT INTO credit_card (id) VALUES ('CcU-9999');
23 •  INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
24    VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', 9999, 829.999, -117.999, 111.11 , 0);
25
```

Below the script is a Result Grid showing the output of the transaction insertion:

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

At the bottom, the Output pane displays the log of actions:

#	Time	Action	Message
1	14:29:51	INSERT INTO company (id) VALUES (b-9999)	1 row(s) affected
2	14:29:55	INSERT INTO credit_card (id) VALUES (CcU-9999)	1 row(s) affected
3	14:30:00	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', 9999, 829.999, -117.999, 111.11 , 0)	1 row(s) affected

Nota: Antes de ingresar la nueva transacción se creó el 'company_id' en la tabla 'company' y el 'credit_card_id' en la tabla 'credit_card' ya que no existían y la transacción depende de estas Foreign Key. Como la tabla 'transaction' contiene datos tanto de 'company' como de credit_card, si estos registros no estaban en las tablas correspondientes no se podía añadir la transacción. Finalmente, se inserta la transacción con todos sus datos (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined).

Ejercicio 4

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

```
27      # Ejercicio 4
28 •  ALTER TABLE credit_card DROP COLUMN pan;
29 •  SHOW COLUMNS from credit_card;
```

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window with the above SQL commands. Below it is a 'Result Grid' table showing the columns of the 'credit_card' table. The table has six columns: id, iban, pin, cvv, expiring_date, and pan (which is listed as YES in the 'Key' column). Below the table is a 'Result 2' section containing the output of the executed statements. The output shows three rows: 1) SHOW COLUMNS from credit_card (6 rows returned), 2) ALTER TABLE credit_card DROP COLUMN pan (0 rows affected, 0 records, 0 duplicates, 0 warnings), and 3) SHOW COLUMNS from credit_card again (5 rows returned).

Field	Type	Null	Key	Default	Extra
id	varchar(10)	NO	PRI	NULL	
iban	varchar(35)	YES		NULL	
pin	char(4)	YES		NULL	
cvv	char(3)	YES		NULL	
expiring_date	varchar(8)	YES		NULL	
pan		YES			

Result 2 ×

Output

#	Time	Action	Message
1	14:19:36	SHOW COLUMNS from credit_card	6 row(s) returned
2	14:19:44	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	14:19:57	SHOW COLUMNS from credit_card	5 row(s) returned

Nivel 2

Ejercicio 1

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

```
30      # NIVEL 2
31      # Ejercicio 1
32 •  DELETE FROM transaction
33      WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
```

The screenshot shows the MySQL Workbench interface. It displays the output of the DELETE statement. The output table shows one row: 1 14:31:47 DELETE FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'. The message column indicates that 1 row(s) were affected.

Output

#	Time	Action	Message
1	14:31:47	DELETE FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) affected

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.

```
35  # Ejercicio 2
36  • CREATE VIEW `VistaMarketing`AS
37  SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount),2) media_compras
38  FROM transaction t
39  JOIN company c ON t.company_id = c.id
40  GROUP BY c.id, c.company_name
41  ORDER BY media_compras DESC;
42
43  • SELECT * FROM transactions.vistamarketing;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:    

company_name	phone	country	media_compras
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
Aliquet Diam Limited	02 76 61 47 46	United States	269.60
Nec Luctus LLC	02 14 71 75 73	Norway	268.60
Neque Tellus Incorporated	04 43 18 34 19	Ireland	267.85
Tortor Nunc Commodo Company	05 35 92 77 16	United States	267.84
Cras Consulting	07 50 10 85 63	Belgium	267.44

vistamarketing 7 ×

Output

Action Output

#	Time	Action	Message
1	15:25:57	CREATE VIEW `VistaMarketing`AS SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount),2) ...	0 row(s) affected
2	15:26:00	SELECT * FROM transactions.vistamarketing	101 row(s) returned

Ejercicio 3

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

```
45  #Ejercicio 3
46  • SELECT * FROM transactions.vistamarketing
47  WHERE country = 'Germany';
```

Result Grid | Filter Rows: Export: Wrap Cell Content:    

company_name	phone	country	media_compras
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Nunc Interdum Incorporated	05 18 15 48 13	Germany	259.32
Convallis 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
Augue Foundation	06 88 43 15 63	Germany	253.51
Aliquam PC	01 45 73 52 16	Germany	253.14

vistamarketing 9 ×

Output

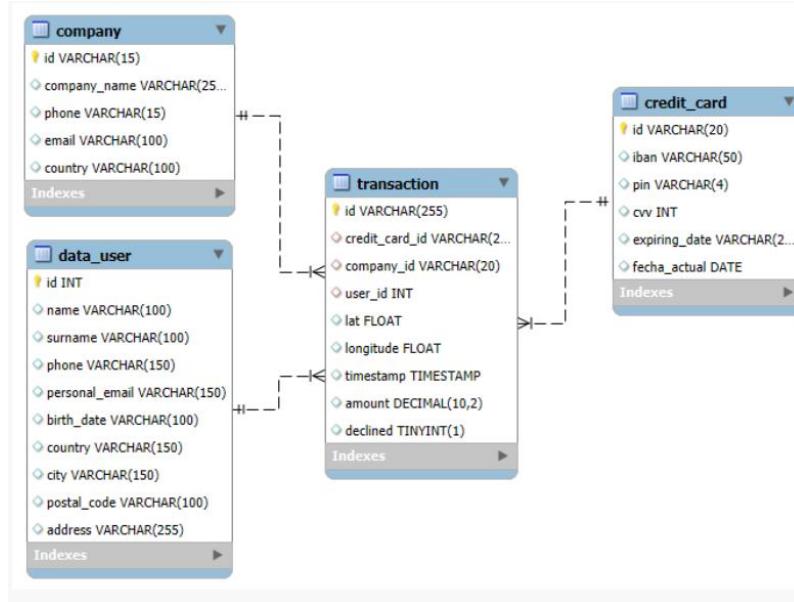
Action Output

#	Time	Action	Message
1	15:30:19	SELECT * FROM transactions.vistamarketing WHERE country = 'Germany'	8 row(s) returned

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:



1. Se crea la tabla ‘user’ con sus columnas, se establece id como la PRIMARY KEY.

```
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)
);
```

2. se añade toda la información de la tabla ‘user’ mediante INSERT INTO.
3. Antes de crear la Foreign Key entre user_id de la tabla ‘transaction’ y id de la tabla ‘user’, se eliminaron registros en ‘transaction’ que contenían valores de user_id que no existían en la tabla ‘user’. En este caso, se identificaron por el company_id.

```
DELETE FROM transaction
WHERE company_id = 'b-9999';
```

4. Se modificó el tipo de dato de id de la tabla ‘user’ para que coincidiera con user_id de la tabla ‘transaction’ (de CHAR(10) a INT), ya que los tipos deben coincidir para establecer una relación de Foreign Key.

```
ALTER TABLE user
MODIFY id INT;
```

5. Se crea la FOREIGN KEY

```
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_user
FOREIGN KEY (user_id)
REFERENCES user(id);
```

The screenshot shows the MySQL Workbench interface with the following details:

- SQL Editor:** Contains the following SQL code:


```

56 • CREATE TABLE IF NOT EXISTS user (
57     id CHAR(10) PRIMARY KEY,
58     name VARCHAR(100),
59     surname VARCHAR(100),
60     phone VARCHAR(150),
61     email VARCHAR(150),
62     birth_date VARCHAR(100),
63     country VARCHAR(150),
64     city VARCHAR(150),
65     postal_code VARCHAR(100),
66     address VARCHAR(255)
67 );
68
69 • DELETE FROM transaction
70 WHERE company_id = 'b-9999';
71
72 • ALTER TABLE user
73     MODIFY id INT;
74
75 • ALTER TABLE transaction
76     ADD CONSTRAINT fk_transaction_user
77     FOREIGN KEY (user_id)
78     REFERENCES user(id);
      
```
- Output Panel:** Shows the execution log with two entries:

#	Time	Action	Message
15	13:24:24	DELETE FROM transaction WHERE company_id = 'b-9999'	0 row(s) affected
16	13:25:05	ALTER TABLE user MODIFY id INT	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

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.

```
70  # Ejercicio 2
71 • CREATE VIEW `InformeTecnico` AS
72   SELECT t.id AS Transaction_ID, u.name AS User_Name, u.surname AS User_Surname,
73          cc.iban AS Credit_card_ibан, c.company_name AS Company_Name
74   FROM transaction t
75   JOIN user u ON t.user_id = u.id
76   JOIN credit_card cc ON t.credit_card_id = cc.id
77   JOIN company c ON t.company_id = c.id
78   ORDER BY Transaction_ID DESC;
79
80 • SELECT * FROM transactions.informetecnico;
```

The screenshot shows a database interface with the following details:

Result Grid:

Transaction_ID	User_Name	User_Surname	Credit_card_iban	Company_Name
FFF031D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgt	Tprvymrc	XX794814451211289182490922	Turpe Company
FFFCF76D-ECP0-4985-A2D0-B2A7B75998FC	Dflrd	Vlqjgdl	XX636251701647892036676034	Amet Nulla Donec Corporation
FFFCF98D-2C7C-4ADE-98F2-7533EF4DF126	Securp	FaoFvfly	XX162677143304223631437567	Nunc Interdum Incorporated
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzipa	Urzjulh	XX395114267082019952567052	Viverra Donec Foundation
FFF9E3CE-234E-408C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537570367941	Convallis In Incorporated
FFF9E178-6CD2-4DF9-9980-49AE06880981	Jevpx	Xwcvzwmn	XX321405515711654384711481	Mus Aenean Eget Foundation
FFFB867C9-17B5-4B1F-ADF9-F8023AA449E	Fdgnd	Lvhfqyxi	XX278446342932680979729426	Cras Vehicula Aliquet Industries
FFF71427D-18FA-4NNN-A73C-499A44C8F76	Nirvaa	Fneonxii	YY415nnn775775nnnnnnnnnnnnnn	Placerat I I P

informetecnico 27 x

Action Output:

#	Time	Action	Message
7	18:20:28	CREATE VIEW `InformeTecnico` AS SELECT t.id AS Transaction_ID, u.name AS User_Name, u.surname AS ...	0 row(s) affected
8	18:20:35	SELECT * FROM transactions.informetecnico	99999 row(s) returned