

## Tarea S3.01. Manipulación de tablas-Radostin Pavlov

### Nivel 1

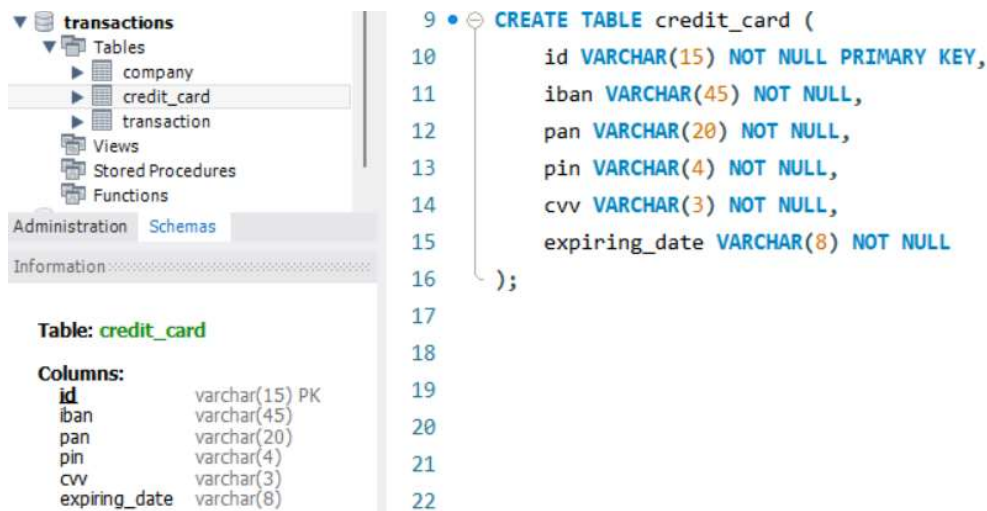
Ejercicio1: Tu tarea es diseñar y crear una tabla llamada "credit\_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después 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.

### Solución:

Debido a que los datos se van a introducir a través de una serie de queries preestablecidas contenidas en el archivo datos\_introducir\_credit.sql, los campos de la tabla nueva han de tener un formato adecuado para alojar los datos de dichas queries. Si abrimos este archivo observamos que las queries tienen esta forma:

```
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984', '10/30/22');
```

Al inspeccionar los datos, se observa que todos los campos tendrían que estar en formato texto (varchar) con una longitud adecuada. Justificación para ello: el id y el iban contienen letras, el pan, pin y cvv pueden tener un cero al principio y las fechas no están en el formato estándar DATE. El formato de las fechas se puede cambiar una vez introducidas como varchar. El campo id tiene que ser un PRIMARY KEY haciendo referencia a transaction.credit\_card\_id. Por ello, la longitud y formato del campo id se configuran para ser las mismas que las de transaction.credit\_card\_id, es decir varchar(15). Las longitudes de los campos iban y pan se han escogido a base de prueba y error ya que hay campos con longitudes anómalas. Por lo tanto, para crear la tabla nueva se ha usado la siguiente query:



The screenshot displays a database management interface. On the left, a tree view shows the 'transactions' schema containing tables 'company', 'credit\_card', and 'transaction'. Below this, the 'Information' tab for the 'credit\_card' table is active, showing its columns: 'id' (varchar(15) PK), 'iban' (varchar(45)), 'pan' (varchar(20)), 'pin' (varchar(4)), 'cvv' (varchar(3)), and 'expiring\_date' (varchar(8)). On the right, the SQL query to create the table is shown:

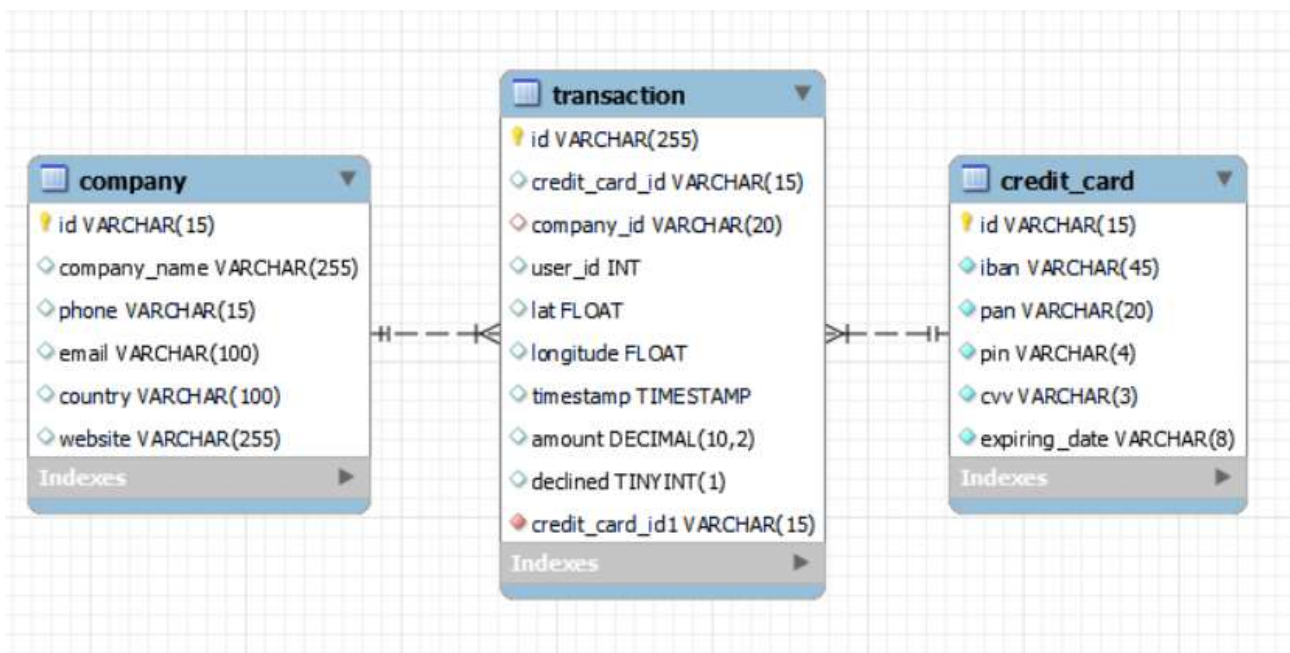
```
9 • CREATE TABLE credit_card (  
10     id VARCHAR(15) NOT NULL PRIMARY KEY,  
11     iban VARCHAR(45) NOT NULL,  
12     pan VARCHAR(20) NOT NULL,  
13     pin VARCHAR(4) NOT NULL,  
14     cvv VARCHAR(3) NOT NULL,  
15     expiring_date VARCHAR(8) NOT NULL  
16 );  
17  
18  
19  
20  
21  
22
```

Después se ha ejecutado el archivo datos\_introducir\_credit.sql para introducir los datos de las tarjetas. La tabla ya poblada se muestra a continuación:

19 • **SELECT \* FROM transactions.credit\_card;**

Result Grid							Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	id	iban	pan	pin	cvv	expiring_date				
▶	CdJ-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22				
	CdJ-2945	DO26854763748537475216568689	5142423821948828	9080	887	08/24/23				
	CdJ-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	06/29/21				
	CdJ-2959	CR7242477244335841535	372461377349375	3583	667	02/24/23				
	CdJ-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	10/29/24				
	CdJ-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	01/30/25				
	CdJ-2980	DE3924188 1883086277136	402400 7145845969	5075	596	07/24/22				
	CdJ-2987	GE89681434837748781813	3763 747687 76666	2298	797	10/31/23				
	CdJ-2994	BH62714428368066765294	344283273252593	7545	595	02/28/22				
	CdJ-3001	CY49087426654774581266832110	511722 924833 2244	9562	867	09/16/22				
	CdJ-3008	LU507216693616119230	4485744464433884	1856	740	04/05/25				
	CdJ-3015	PS119398216295715968342456821	3784 662233 17389	3246	822	01/31/22				
	CdJ-3022	GT91695162850556977423121857	5164 1379 4842 3951	5610	342	04/25/25				
	CdJ-3029	AZ62317413982441418123739746	3429 279566 77631	9708	505	09/02/23				
	CdJ-3036	AZ39336002925842865843941994	3768 451556 48766	2232	565	10/27/25				
	CdJ-3043	TN6488143310514852179535	455676 6437463635	5969	196	06/07/25				
	CdJ-3050	FR5167744369175836831854477	4024007123722	4834	126	10/09/23				
	CdJ-3057	LU931822574697545215	3484 621767 21237	6805	848	09/14/25				
	CdJ-3064	PS146965545449253377627273133	3467 732741 26810	3865	498	06/03/25				
	CdJ-3071	NO8923814763512	3464 789562 23352	6625	661	12/20/23				
	CdJ-3078	ISO25127145884623279548733	4539 322 74 2377	9405	720	03/08/23				

A continuación se actualiza el modelo de la base de datos (con Reverse Engineer). Las tablas transaction y credit\_card guardan una relación n:1 ya que una tarjeta puede estar asociada a muchas compras (pero cada compra a solo una tarjeta).



```

22 -- Ejercicio 2: El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con: Id CcU-2938.
23 -- Se requiere actualizar la información que identifica una cuenta bancaria a nivel internacional (identificado como "IBAN"): TR3234563122135768
24 -- Recuerda mostrar que el cambio se realizó.
25
26 • UPDATE credit_card
27 SET iban = 'TR323456312213576817699999' -- era 'TR301950312213576817638661'
28 WHERE id = 'CcU-2938';
29
30 • SELECT * FROM transactions.credit_card;
31

```

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR323456312213576817699999	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
CcU-2980	DE3924188 1883086277136	402400 7145845969	5075	596	07/24/22
CcU-2987	GE89681434837748781813	3763 747687 76666	2298	797	10/31/23
CcU-2994	BH62714428368066765294	344283273252593	7545	595	02/28/22
CcU-3001	CY49087426654774581266832110	511722 924833 2244	9562	867	09/16/22

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; lat: 829.999; longitud: -117.999; amount: 111.11; declined: 0

Primero se comprueba que la compañía no está registrada en la tabla company:

```

35 • select *
36 from company
37 where id = 'b-9999';
38

```

id	company_name	phone	email	country	website
NULL	NULL	NULL	NULL	NULL	NULL

Se debe crear primero el registro de la compañía en la tabla company que es la madre de transaction. Se comprueba que todos los campos de esta (menos el id que es el PK) admiten valores nulos:

Column Name	Datatype	PK	NN
id	VARCHAR(15)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
company_name	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>
phone	VARCHAR(15)	<input type="checkbox"/>	<input type="checkbox"/>
email	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>
country	VARCHAR(100)	<input type="checkbox"/>	<input type="checkbox"/>
website	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>

Esto significa que se pueden crear registros solo con el `company_id`, que es lo único que tenemos de momento. El resto de campos se pueden rellenar a posteriori. Se ejecuta la consulta de inserción y después se comprueba que la compañía se ha registrado:

```









39 • INSERT INTO company (id)
40 VALUES ('b-9999');
41
42 • select *
43 from company
44 where id = 'b-9999';
45

```

id	company_name	phone	email	country	website
b-9999	NULL	NULL	NULL	NULL	NULL

Ahora ya se pueden insertar los datos de la transacción:

```
46 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
47   VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
48
49 • select *
50   from transaction
51  where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
52
```

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

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

No obstante, se observa que el valor del timestamp introducido es 'null'. Para solucionarlo hay que cambiar el modo del timestamp a CURRENT\_TIMESTAMP, borrar el registro y volver a insertarlo:

```
53 • ALTER TABLE transaction
54   MODIFY COLUMN timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
55
56 • DELETE FROM transaction
57  where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
58
59 • select *
60   from transaction
61  where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
62
63 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
64   VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', '111.11', '0');
65
66 • select *
67   from transaction
68  where id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
69
```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
▶	108B1D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	2024-02-28 19:55:48	111.11	0

```
71 -- Ejercicio 4: Desde recursos humanos te solicitan eliminar la columna
72 -- "pan" de la tabla credit_card. Recuerda mostrar el cambio realizado.
73 • ALTER TABLE credit_card
74   DROP COLUMN pan;
75
76 • select * from credit_card;
77
```

Result Grid					
Filter Rows:					
Edit:   Export/Import:   Wrap Cell Content: T A					
	id	iban	pin	cvv	expiring_date
▶	CcU-2938	TR323456312213576817699999	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
	CcU-2973	PT87806228135092429456346	8760	887	01/30/25
	CcU-2980	DE39241881883086277136	5075	596	07/24/22
	CcU-2987	GE89681434837748781813	2298	797	10/31/23

Se comprueba que el pan ya no está.



```

79 # Nivel 2
80 -- Ejercicio 1: Elimina el registro con IBAN 02C6201E-D90A-1859-B4EE-*88D2986D3B02 de la base de datos.
81
82 • SELECT * FROM credit_card
83 WHERE iban = '02C6201E-D90A-1859-B4EE-*88D2986D3B02';

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
id	iban	pin	cvv	expiring_date
NULL	NULL	NULL	NULL	NULL

El registro no existe. El código iban más parecido es: 02C6201E-D90A-1859-B4EE-88D2986D3B02

```

82 • SELECT * FROM credit_card
83 WHERE iban = '02C6201E-D90A-1859-B4EE-*88D2986D3B02';
84
85 • DELETE FROM credit_card
86 WHERE iban = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
87
88

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
id	iban	pin	cvv	expiring_date
NULL	NULL	NULL	NULL	NULL

```

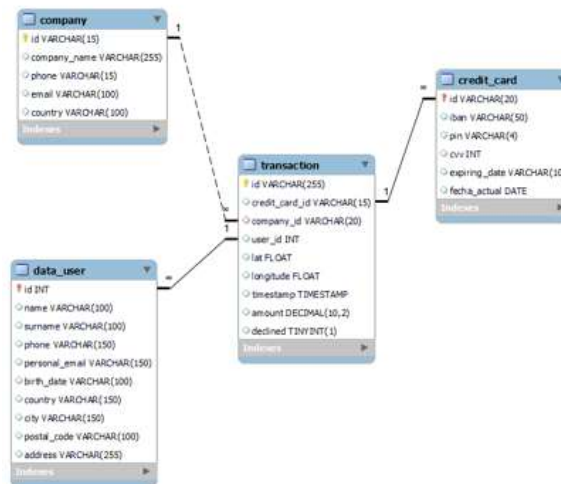
90 -- Ejercicio 2: Crear una vista llamada VistaMarketing que contenga la siguiente
91 -- información: Nombre de la compañía. Teléfono de contacto. País de residencia.
92 -- Media de compra realizada por cada compañía. Presenta la vista creada, ordenando
93 -- los datos de mayor a menor promedio de compra
94
95 • CREATE VIEW VistaMarketing AS
96     SELECT
97         company_name AS Compañía,
98         phone AS Teléfono,
99         country AS País,
100         AVG(amount) AS `Promedio compra`
101     FROM
102         company
103     JOIN
104         transaction ON company.id = transaction.company_id
105     GROUP BY company_name
106     ORDER BY `Promedio compra` DESC;
107
108 • SELECT * FROM transactions.vistamarketing;

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Compañía	Teléfono	País	Promedio compra
Eget Ipsum Ltd	03 67 44 56 72	United States	473.075000
Non Magna LLC	06 71 73 13 17	United Kingdom	468.345000
Sed Id Limited	07 28 18 18 13	United States	461.210000
Justo Eu Arcu Ltd	08 42 56 71 52	Italy	443.635000
Eget Tincidunt Dui Institute	05 35 93 32 44	Netherlands	442.520000

### NIVEL 3

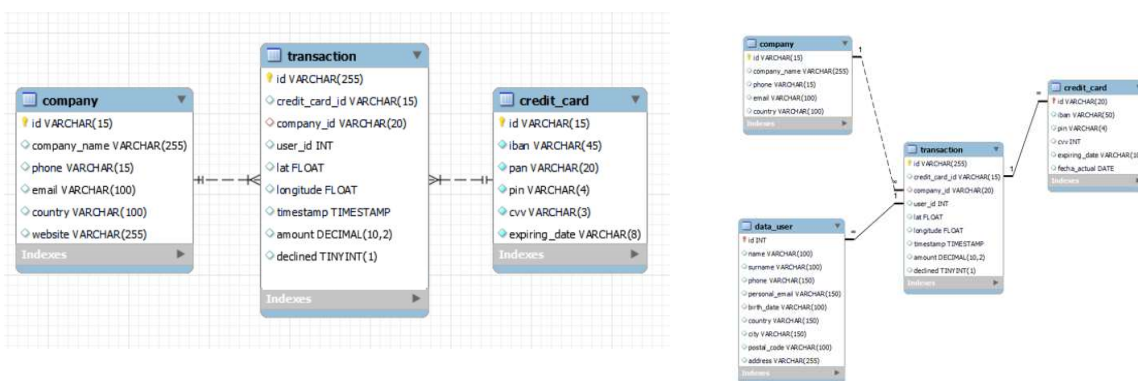
Ejercicio1: La próxima semana tendrás una nueva reunión con los gerentes de marketing. Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Te pide que le ayudes a dejar los comandos ejecutados para obtener las siguientes modificaciones (se espera que realicen 6 cambios):



En esta actividad, es necesario que describas el "paso a paso" de las tareas realizadas. Es importante realizar descripciones sencillas, simples y fáciles de comprender. Para realizar esta actividad deberás trabajar con los archivos denominados "estructura\_datos\_user" y "datos\_introducir\_user".

### Solución

Para poder resolver el problema, habría que comparar el modelo con una instantánea previa a los cambios realizados por el compañero. A continuación se compara el modelo nuevo con la versión anterior:



Se observan las siguientes modificaciones:

- Aparece una tabla adicional llamada data\_user con una relación de 1:n respecto a la tabla transaction. Se entiende que se trata de usuarios independientes de las compañías de la tabla company, que simplemente consultarían datos.

- Ha desaparecido el campo website de la tabla company.
- En la tabla credit\_card ha aparecido un campo llamado fecha\_actual de tipo DATE.

Observación: La relación 1:n entre transaction y credit\_card parece estar invertida respecto a como tendría que ser.

Se inspecciona el contenido del archivo estructura\_datos\_user.sql.

```

estructura_datos_user.sql
Archivo  Editar  Ver

-- Creamos la tabla user

CREATE INDEX idx_user_id ON transaction(user_id);

CREATE TABLE IF NOT EXISTS user (
  id INT 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),
  FOREIGN KEY(id) REFERENCES transaction(user_id)
);

```

Se observa que la primera query crea un índice idx\_user\_id en la tabla transaction sobre la clave user\_id. La segunda query crea la tabla llamada user con campos idénticos a los de la tabla nueva del modelo. Hay que cambiar el nombre user, debido a que la tabla del modelo se llama data\_user. Esto se podría hacer directamente editando el archivo sql, pero habría que editar también el archivo que introduce los datos en la tabla. Se opta por renombrar la tabla después de crearla y poblarla. Finalmente se observa que el código crea una clave foránea que hace referencia a la clave user\_id de la tabla transaction, con lo cual se establece la relación entre user y transaction.

Se abre el archivo datos\_introducir\_user.sql para su inspección.

```

estructura_datos_user.sql  datos_introducir_user.sql
Archivo  Editar  Ver

SET foreign_key_checks = 0;

-- Insertamos datos de user
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "1", "Zeus", "Gamble
"1-282-581-0551", "interdum.enim@protonmail.edu", "Nov 17, 1985", "United States", "Lowell", "73544", "348-7818 Sagittis St.");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "2", "Garrett",
"Mcconnell", "(718) 257-2412", "integer.vitae.nibh@protonmail.org", "Aug 23, 1992", "United States", "Des Moines", "59464", "903
Sit Ave");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "3", "Ciaran",
"Harrison", "(522) 598-1365", "interdum.feugiat@aol.org", "Apr 29, 1998", "United States", "Columbus", "56518", "736-2063 Tellus
St.");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "4", "Howard",
"Stafford", "1-411-740-3269", "ornare.egestas@icloud.edu", "Feb 18, 1989", "United States", "Kailua", "77417", "Ap #545-2244 Era
Rd.");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "5", "Hayfa",
"Pierce", "1-554-541-2077", "et.malesuada.fames@hotmail.org", "Sep 26, 1998", "United States", "Sandy", "31564", "341-2821
Ultrices Av.");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "6", "Joel", "Tyson"
"(718) 288-8020", "gravida.nunc.sed@yahoo.ca", "Oct 15, 1989", "United States", "Nashville", "96838", "888-2799 Amet Street");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "7", "Rafael",
"Jimenez", "(817) 689-0478", "eget@outlook.ca", "Dec 4, 1981", "United States", "Hillsboro", "29874", "8627 Malesuada Rd.");
INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (      "8", "Nissim",

```

Se observa que una primera consulta deshabilita temporalmente la comprobación de claves foráneas (SET foreign\_key\_checks = 0). A continuación una serie de consultas insertan los datos en la tabla

user. Finalmente, se habilita de nuevo la comprobación de claves foráneas (SET foreign\_key\_checks = 1).

Paso 1 y 2. Se ejecutan en MySQL Workbench los dos archivos (primero "estructura\_datos\_user" y luego "datos\_introducir\_user") para crear y poblar la tabla user:

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' pane shows the 'user' table structure with 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)), and address (varchar(255)).

The 'Result Grid' shows the following data:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	903 Sit Ave
3	Claran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbus	56518	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	77417	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Sandy	31564	341-2821 Ultrices Av.
6	Joel	Tyson	(718) 288-8020	gravidam.nunc.sed@yahoo.ca	Oct 15, 1989	United States	Nashville	96838	888-2799 Amet Street
7	Rafael	Jimenez	(817) 689-0478	egestas@outlook.ca	Dec 4, 1981	United States	Hillsboro	29874	8627 Malesuada Rd.
8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	Jackson	61750	Ap #251-7144 Integer St.
9	Mannix	Mcclain	(590) 883-2184	aliquam.nisi@outlook.com	Jan 24, 1987	United States	Richmond	35987	647-3080 Lacus. St.
10	Robert	McCarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	Eugene	85526	P.O. Box 773, 3594 Ornare St.
11	Joan	Baird	(981) 429-8106	et@outlook.net	Feb 25, 1990	United States	Lincoln	35211	P.O. Box 687, 8917 Ligula St.
12	Benedict	Wheeler	1-515-824-2855	tinidunt.donec.vitae@hotmail.co.uk	Aug 6, 1999	United States	Lewiston	92393	748-8694 Porttitor Avenue
13	Allegra	Stanton	1-927-753-6488	proin.egat@protonmail.ca	May 19, 1990	United States	Kearney	14947	4457 Ante. Av.
14	Sara	Flynn	1-311-646-9333	integer@outlook.net	Dec 27, 1988	United States	Warren	20288	P.O. Box 865, 4397 Ante St.
15	Noelani	Patrick	1-723-488-5894	sem.magna@google.com	Sep 17, 1993	United States	Orlando	47987	596-5044 Sapien. Street
16	Eric	Roth	1-218-549-8253	lorem.sit@yahoo.net	Sep 7, 1988	United States	Reading	96697	P.O. Box 541, 5137 Non Road
17	Bruce	Gill	(744) 732-8628	metus@aol.net	Mar 4, 1990	United States	Davenport	43415	Ap #836-9508 Vitae. Ave
18	Russell	Jimenez	(657) 779-2438	oro@outlook.edu	Aug 26, 1993	United States	Hattiesburg	75647	4095 Quam Rd.
19	Nicholas	Travis	1-330-223-9652	libero.dui@hotmail.com	Jul 15, 1981	United States	Jacksonville	71727	Ap #459-539 Lectus Avenue
20	Kelsey	Bates	(653) 724-4754	ullamcorper.nisi@aol.com	May 6, 1981	United States	Gulftport	50423	824-3624 Lacinia St.
21	Hall	Reeves	(241) 759-9235	erat.egat@hotmail.edu	Jul 22, 1987	United States	Warren	85521	Ap #745-5948 Sollicitudin St.
22	Allistair	Holmes	1-265-323-0812	donec.tempor.est@protonmail.com	Nov 5, 1990	United States	Montpelier	85914	Ap #794-4229 Ante Rd.
23	Kelsie	Bass	1-837-832-5631	consequat@google.ca	Apr 2, 1990	United States	Jefferson ...	97237	407-7562 A. Road

Paso 3. Se cambia el nombre de la tabla user a data\_user:

The screenshot shows the 'Schemas' pane with the 'data\_user' table structure. The columns are: 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)), and address (varchar(255)).

The SQL editor shows the following commands:

```

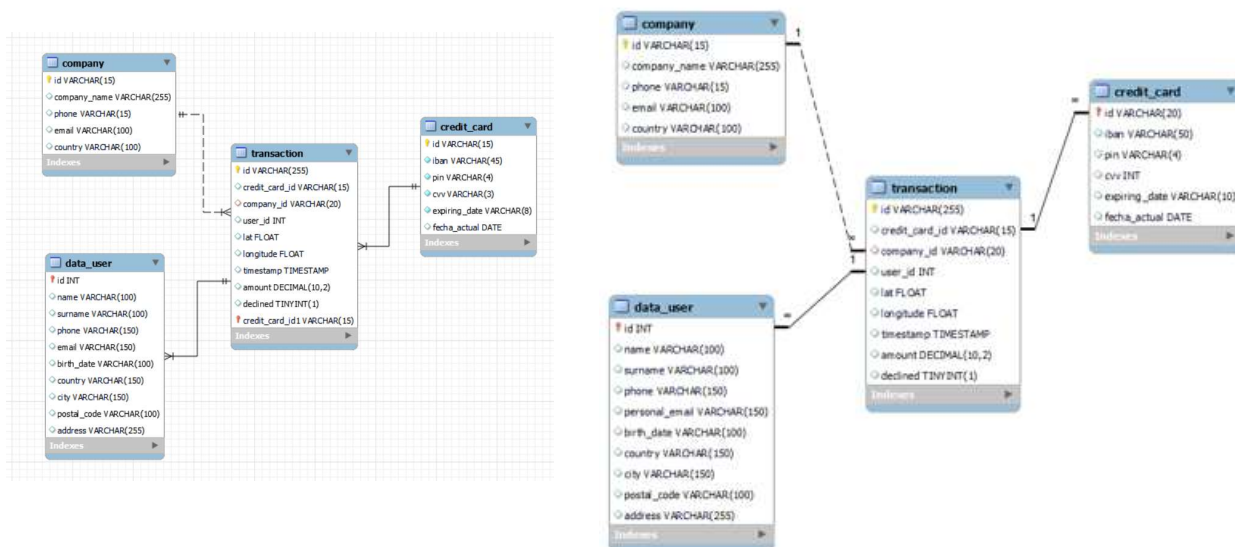
125
126 • select * from user;
127
128 • alter table user rename to data_user;

```

Paso 4. Se elimina el campo website de la tabla company  
ALTER TABLE company DROP COLUMN website;

Paso 5. Se añade el campo fecha\_actual en la tabla credit\_card  
ALTER TABLE credit\_card ADD fecha\_actual DATE;

Paso 6. Mediante la función Reverse Engineer de MySQL Workbench se genera el modelo y se compara con el modelo anterior:





Filter objects

- ▶ sakila
- ▶ sys
- ▼ transactions
  - ▶ Tables
  - ▼ Views
    - ▶ infometecnico
    - ▶ vistamarketing
  - ▶ Stored Procedures
  - ▶ Functions
- ▶ ventas
- ▶ world

```
144
145 -- Ejercicio 2: La empresa también te solicita crear una vista llamada "InformeTecnico" que contenga la siguiente información:
146 -- 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.
147 -- Asegúrate de incluir información relevante de ambas tablas y utiliza alias para cambiar de nombre columnas según sea necesario.
148 -- Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción.
149
150 ● CREATE VIEW InformeTecnico AS
151     SELECT
152         name AS Nombre,
153         surname AS Apellido,
154         transaction.id AS "ID de la transacción",
155         IBAN,
156         company_name AS "Compañía",
157         DATE(timestamp) AS Fecha,
158         amount AS Cantidad
159     FROM
160         data_user
161     JOIN
162         transaction ON transaction.user_id = data_user.id
163     JOIN
164         credit_card ON credit_card.id = transaction.credit_card_id
165     JOIN
166         company ON company.id = transaction.company_id;
167
168 ● SELECT * FROM transactions.InformeTecnico ORDER BY "ID de la transacción" DESC;
```

Administration Schemas

Information

No object selected

Result Grid		Filter Rows:	Export:	Wrap Cell Content:			
	Nombre	Apellido	ID de la transacción	IBAN	Compañía	Fecha	Cantidad
▶	Kenyon	Hartman	108B1D1D-5B23-A76C-55EF-C568E49A05DD	TR323456312213576817699999	Ac Fermentum Incorporated	2021-07-07	293.57
	Kenyon	Hartman	EA2C3281-C9C1-A387-44F8-729FB4B51C76	TR323456312213576817699999	Ac Fermentum Incorporated	2021-05-09	119.36
	Kenyon	Hartman	7DC26247-20EC-53FE-E555-B6C2E55CA5D5	DO26854763748537475216568689	Magna A Neque Industries	2022-02-04	312.50
	Kenyon	Hartman	FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	DO26854763748537475216568689	Magna A Neque Industries	2021-06-15	480.13
	Kenyon	Hartman	72997E96-DC2C-A4D7-7C24-66C302F8AE5A	BG45IVQL52710525608255	Fusce Corp.	2022-01-30	239.87
	Kenyon	Hartman	8768FDE2-A231-B916-8644-F7DCD13CAFC2	BG45IVQL52710525608255	Fusce Corp.	2021-05-06	460.38
	Kenyon	Hartman	0DD2E608-5C9E-D1B3-4999-B99F43AD735A	CR7242477244335841535	Convallis In Incorporated	2021-04-17	252.47