

Sprint 3: Manipulación de tablas en SQL

Curso de especialización Data Analytics - IT Academy - Barcelona Activa

Nicolás Aros Marzá

23/12/2025

Nivel 1

Ejercicio 1

Su tarea es diseñar y crear una tabla llamada “credit_card” que almacena detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de manera única cada tarjeta y establecer una relación adecuada con las otras dos tablas (transaction y company).

Después de crear la tabla, deberá ingresar la información en el documento llamado `dades_introducir_credit`. Recuerde mostrar el diagrama y hacer una breve descripción de él.

La primera acción para responder a esta solicitud es crear la tabla `credit_card` mediante la instrucción `CREATE TABLE`. En los parámetros se indica el nombre y las características de las columnas que se necesitan para ingresar de manera adecuada los datos que se encuentran en archivo `datos_introducir_sprint3_credit.sql`.

Las columnas creadas corresponden a `id`, `iban`, `pan`, `pin`, `cvv`, y `expiring_date`. Todas las variables son de tipo carácter y tienen un límite de extensión acorde a la información que se le agregará más adelante. La única variable que es distinta es `id`, puesto que es de tipo `PRIMARY KEY`, que permitirá relacionar esta tabla con las otras del esquema.

Creación de tabla `credit_card` y definición de las columnas.

```
1 -- Nivel 1
2 -- Ejercicio 1
3 USE transactions;
4
5 CREATE TABLE IF NOT EXISTS credit_card
6   id VARCHAR(15) PRIMARY KEY,
7   iban VARCHAR(100),
8   pan VARCHAR(100),
9   pin VARCHAR(150),
10  cvv VARCHAR(150),
11  expiring_date VARCHAR(100));
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

Output

#	Time	Action	Message
1	14:30:51	USE transactions	0 row(s) affected
2	14:30:51	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, iban VARCHAR(100), pan VARCHAR(100), pin VARCHAR(150), cvv VARCHAR(150), expiring_date VARCHAR(100));	0 row(s) affected

Después de esta tarea se ejecuta el archivo `datos_introducir_sprint3_credit.sql` para ingresar información a la tabla `credit_card`.

El siguiente paso es establecer la relación de la tabla `credit_card` y la tabla `transaction`. Para esto se utilizó la instrucción `ALTER TABLE`, y las cláusulas `ADD CONSTRAINT` y `FOREIGN KEY`. Con esto, se establece que las columnas `credit_card_id` de la tabla `transaction` e `id` de la tabla `credit_card` contienen la información para poner en relación estas tablas. Se utilizó `CONSTRAINT` para especificar el nombre de la relación y facilitar el trabajo a futuro en el caso que se necesite realizar un cambio.

La relación entre las tablas es del tipo 1:N, puesto que una fila de la tabla `credit_card` puede estar relacionado con muchas filas de la tabla `transaction`, pero cada transacción solo se relaciona con una fila de la tabla de tarjetas.

Cambios en la tabla transaction para relacionar con credit_card.

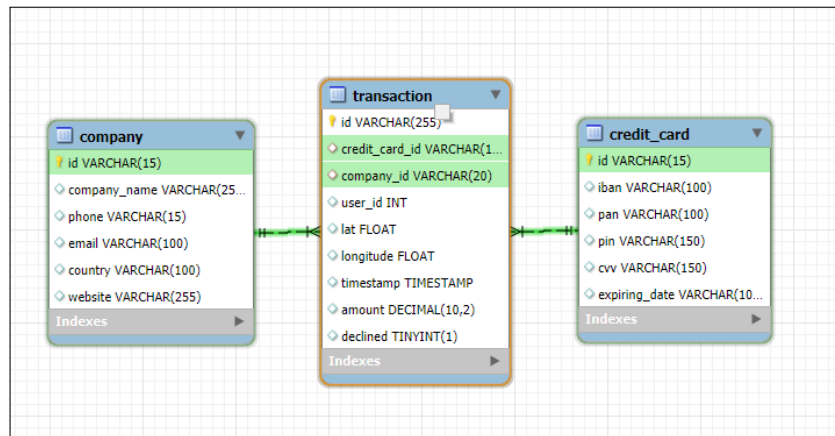
```
16 ALTER TABLE transaction
17 ADD CONSTRAINT fk_credit_card
18 FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47 -- Ejercicio 2
```

Output

#	Time	Action	Message
4998	14:39:00	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9579', 'XX296393091587170202131236', '9690060468...	1 row(s) affected
4999	14:39:00	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9580', 'XX781258889851950806677358', '5541182364...	1 row(s) affected
5000	14:39:00	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9581', 'XX915670516405388124398147', '2624305470...	1 row(s) affected
5001	14:40:43	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

El resultado de todas las operaciones realizadas anteriormente se encuentran en el siguiente esquema:

Esquema tras la incoportación de la tabla credit_card.



Ejercicio 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta asociado a la tarjeta de crédito conID CcU-2938. La información que se mostrará para este registro es: TR32345631213571769999. Recuerde demostrar que el cambio se hizo.

En primer lugar, para modificar la información de una celda en una tabla se utiliza la instrucción **UPDATE**. en la que se define la tabla a modificar. En la cláusula **SET** se indica la columna que se actualizará y con **WHERE** se filtra la observación que se quiere cambiar.

Actualización de datos en la tabla `credit_card`.

The screenshot shows a SQL IDE interface. The top pane contains the following SQL code:

```
21 • UPDATE credit_card
22   SET iban = 'TR323456312213576817699999'
23   WHERE id = 'CcU-2938';
24
25 • SELECT *
26   FROM credit_card
27   WHERE id = 'CcU-2938';
```

The bottom pane shows the 'Result Grid' with the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR302950312213576817638661	5424465566813633	3257	984	10/30/22

Below the result grid, the 'Action Output' pane shows the execution log:

#	Time	Action	Message
4999	14:39:00	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (CcU-9580, 'XX781258889851950006677358', '5541182...	1 row(s) affected
5000	14:39:00	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (CcU-9581, 'XX915670516405388124398147', '2624305...	1 row(s) affected
5001	14:40:43	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
5002	15:06:03	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)	Error Code: 1826. Duplicate foreign key constraint name fk_credit_ca
5003	15:06:14	SELECT * FROM credit_card WHERE id = CcU-2938	1 row(s) returned

Como se observa en la imagen, el cambio se realizó con éxito.

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
lat: 829.999
longitude: -117.999
amount: 111.11
declined: 0
```

Para ingresar una nueva observación en la tabla `transaction`, primero se incorporan los registros necesarios en las tablas relacionadas (`credit_card` y `company`). Todas estas operaciones se hacen mediante la instrucción `INSERT INTO`. En particular, se añade un nuevo valor a la columna `id` en la tabla `credit_card` (CcU-9999) y otro a la tabla `company` (b-9999). Dado que no se proporciona información para el resto de las columnas, estas quedan con valores nulos.

A continuación, en la tabla `transaction` se agregan una nueva fila con valores válidos en las variables `id`, `credit_card_id`, `company_id`, `user_id`, `lat`, `longitude`, `amount` y `declined`. Dado a que no hay información sobre el momento de la transacción, la columna `timestamp` queda con valor `NULL`.

El detalle de las operaciones realizadas y el resultado en la tabla se encuentra en la siguiente imagen:

Agregar nueva observación a la tabla transaction.

```

30 • INSERT INTO company (id)
31   VALUES ('b-9999');
32
33 • INSERT INTO credit_card (id)
34   VALUES ('Ccu-9999');
35
36 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
37   VALUES ('10881D1D-5B23-A76C-55EF-C568E49A990D', 'Ccu-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);
38
39 • SELECT *
40   FROM transaction
41  WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A990D'
42
43
44
45
46

```

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
10881D1D-5B23-A76C-55EF-C568E49A990D	Ccu-9999	b-9999	9999	829.999	-117.999	10881D1D-5B23-A76C-55EF-C568E49A990D	111.11	0

transaction 3 x Apply

Output

#	Time	Action	Message
5004	15:06:41	SELECT * FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A990D'	0 row(s) returned
5005	15:09:07	INSERT INTO company (id) VALUES ('b-9999')	1 row(s) affected
5006	15:09:10	INSERT INTO credit_card (id) VALUES ('Ccu-9999')	1 row(s) affected
5007	15:09:13	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('10881D1D-5B23-A76C-55EF-C568E49A990D', 'Ccu-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);	1 row(s) affected
5008	15:09:16	SELECT * FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A990D'	1 row(s) returned

La incorporación del dato referido a la tabla `user_id` se realiza en el ejercicio 1 del nivel 3.

Ejercicio 4

De recursos humanos le piden que elimine la columna `pan` de la tabla `credit_card`. Recuerde mostrar el cambio realizado.

Para eliminar una columna se utiliza la instrucción `ALTER TABLE` en la cual se declara la tabla que será modificada y en la cláusula `DROP COLUMN` se indica que la columna eliminada será `pan`.

En la siguiente imagen se muestra el código implementado y se encuentra la consulta para corroborar que la columna fue eliminada con éxito.

Eliminación columna `pan` de la tabla `credit_card`.

The screenshot displays a database management interface. The top section shows the following SQL code:

```
44 ALTER TABLE credit_card
45 DROP COLUMN pan;
46
47 SELECT *
48 FROM credit_card;
```

Below the code editor, the 'Result Grid' shows the columns: `id`, `iban`, `pin`, `cvv`, and `expiring_date`. The grid contains 10 rows of data, each starting with a card ID (e.g., Ccd-4857) followed by its corresponding values.

The bottom section, 'Output', shows the execution log with the following entries:

- 5006 15:09:10 INSERT INTO credit_card (id) VALUES (CcdU-9999) 1 row(s) affected
- 5007 15:09:13 INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('10881D1D-5B23-A76...' 1 row(s) affected
- 5008 15:09:16 SELECT * FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A990D' 1 row(s) returned
- 5009 15:15:09 ALTER TABLE credit_card DROP COLUMN pan 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
- 5010 15:15:13 SELECT * FROM credit_card 5001 row(s) returned

Nivel 2

Ejercicio 1

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

Para eliminar una fila se utiliza la instrucción `DELETE FROM` en la cual se declara la tabla que será modificada (`transaction`) y con la cláusula `WHERE` se indica el criterio para seleccionar la observación que se eliminará. En este caso se filtra según un valor específico en la variable `id`.

En la siguiente imagen se muestra el código implementado y se encuentra la consulta para corroborar que la fila con `id = 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD` fue eliminada con éxito.

Eliminación de casos de la tabla transaction.

The screenshot displays a database management interface with a SQL editor at the top and a results/output pane at the bottom. The SQL editor contains the following queries:

```
54 DELETE FROM transaction
55 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
56
57 SELECT *
58 FROM transaction
59 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
60
61
62
63
64
65
66
67
68
69
70
--
```

The results pane shows the execution of these queries:

#	Time	Action	Message
5011	15:16:41	USE transactions	0 row(s) affected
5012	15:16:41	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, iban VARCHAR(100), pan VARCHAR(100), pin VA...	0 row(s) affected, 1 warning(s): 1050 Table 'credit_card' already exists
5013	15:16:41	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)	Error Code: 1026. Duplicate foreign key constraint name 'fk_credit_card'
5014	15:16:44	DELETE FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) affected
5015	15:16:52	SELECT * FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) returned

Ejercicio 2

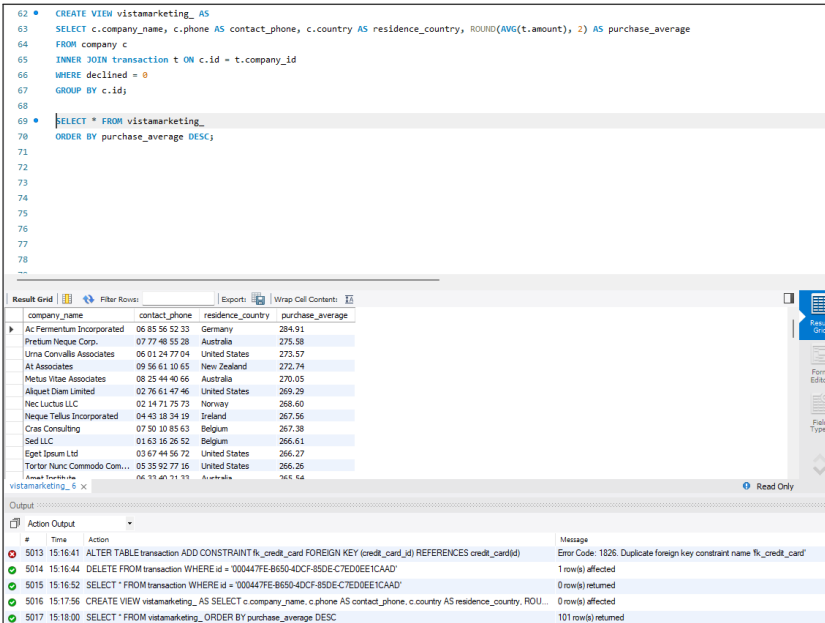
La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado la creación de una vista que proporcione detalles clave sobre las empresas y sus transacciones. Deberá crear una vista llamada **VistaMarketing** que contenga la siguiente información: Nombre de la empresa. Teléfono de contacto. País de residencia. Compra promedio realizada por cada empresa. Presenta la vista creada, ordenando los datos de la compra promedio más alta a la más baja.

La vista solicitada por el equipo de marketing se genera mediante la instrucción **CREATE VIEW**. Dentro de la cláusula **SELECT** se especifican las variables requeridas, aplicando alias a las columnas para que los nombres coincidan exactamente con los términos utilizados en la solicitud.

Se incluye el criterio de ordenación basado en el promedio de compras de cada empresa directamente al consultar la vista, utilizando la cláusula **ORDER BY** sobre la variable **purchase_average**. Aunque este criterio podría haberse integrado en la definición original de la vista, el resultado final es idéntico en términos de visualización de datos.

A continuación, se presenta el código utilizado para la creación de la vista junto con los resultados obtenidos tras su ejecución.

Creación y consulta de la vista "VistaMarketing"



```
62 CREATE VIEW vistaMarketing AS
63 SELECT c.company_name, c.phone AS contact_phone, c.country AS residence_country, ROUND(AVG(t.amount), 2) AS purchase_average
64 FROM company c
65 INNER JOIN transaction t ON c.id = t.company_id
66 WHERE declined = 0
67 GROUP BY c.id;
68
69 SELECT * FROM vistaMarketing
70 ORDER BY purchase_average DESC;
```

company_name	contact_phone	residence_country	purchase_average
Ac Fennentum Incorporated	06 85 56 52 33	Germany	284.91
Pretium Neque Corp.	07 77 46 55 28	Australia	275.58
Uma Conville Associates	06 01 24 77 04	United States	273.57
At Associates	09 56 61 10 65	New Zealand	272.74
Nexus Vitae Associates	08 25 44 40 66	Australia	270.05
Aliquet Diam Limited	02 76 61 47 46	United States	269.29
Nec Lucius LLC	02 14 71 75 73	Norway	268.60
Neque Tellus Incorporated	04 43 18 34 19	Ireland	267.56
Cras Consulting	07 50 10 85 63	Belgium	267.38
Sed LLC	01 63 16 26 52	Belgium	266.61
Eget Ipsum Ltd	03 67 44 56 72	United States	266.27
Tortor Nunc Commodo Com...	05 35 92 77 16	United States	266.26
Amet Turpis Inc	06 19 40 71 13	Australia	266.64

Output

#	Time	Action	Message
5013	15:16:41	ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id)	Error Code: 1826. Duplicate foreign key constraint name 'fk_credit_card'
5014	15:16:44	DELETE FROM transaction WHERE id = '000447FE-8650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) affected
5015	15:16:52	SELECT * FROM transaction WHERE id = '000447FE-8650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) returned
5016	15:17:56	CREATE VIEW vistaMarketing AS SELECT c.company_name, c.phone AS contact_phone, c.country AS residence_country, ROU...	0 row(s) affected
5017	15:18:00	SELECT * FROM vistaMarketing ORDER BY purchase_average DESC	101 row(s) returned

Ejercicio 3

Filtrar la vista VistaMarketing para mostrar solo a las empresas que tienen su país de residencia en Germany.

Esta consulta mantiene la estructura de la anterior, incorporando un filtro específico mediante la cláusula **WHERE** para mostrar únicamente las empresas cuyo país de residencia sea Alemania (Germany). Aunque el enunciado del ejercicio no solicita explícitamente una ordenación, se ha mantenido el criterio según el promedio de ventas (**purchase_average**) para conservar la coherencia visual con los resultados obtenidos previamente.

Consulta a la vista VistaMarketing solo empresas alemanas.

The screenshot shows a database management tool interface. The top pane displays a SQL query:

```
73 SELECT * FROM vistamarketing_  
74 WHERE residence_country = 'Germany'  
75 ORDER BY purchase_average DESC;  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89
```

The bottom pane shows the 'Result Grid' with the following data:

company_name	contact_phone	residence_country	purchase_average
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.91
Nunc Interdum Incorporated	05 18 15 48 13	Germany	299.32
Conville In Incorporated	06 66 97 29 50	Germany	257.69
Ac Industries	09 34 65 40 60	Germany	255.17
Rubrum Non Inc.	02 66 31 61 09	Germany	255.14
Auctor Mauris Corp.	05 62 87 14 41	Germany	254.68
Augue Foundation	06 88 43 15 63	Germany	253.56
Aliquam PC	01 45 73 52 16	Germany	252.96

The bottom pane also shows the 'Output' section with the following actions:

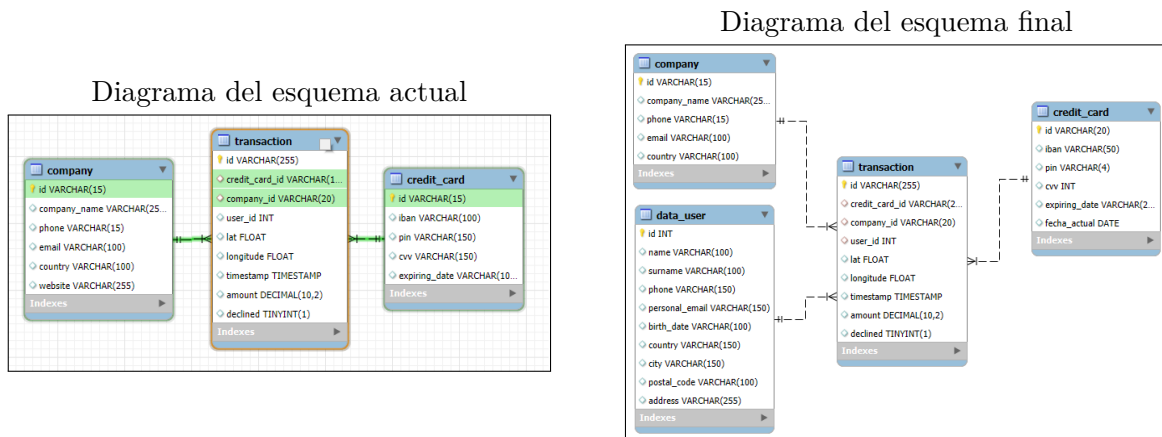
#	Time	Action	Message
5014	15:16:44	DELETE FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	1 row(s) affected
5015	15:16:52	SELECT * FROM transaction WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'	0 row(s) returned
5016	15:17:56	CREATE VIEW vistamarketing_ AS SELECT c.company_name, c.phone AS contact_phone, c.country AS residence_country, ROU...	0 row(s) affected
5017	15:18:00	SELECT * FROM vistamarketing_ ORDER BY purchase_average DESC	101 row(s) returned
5018	15:18:57	SELECT * FROM vistamarketing_ WHERE residence_country = 'Germany' ORDER BY purchase_average DESC	8 row(s) returned

Nivel 3

Ejercicio 1

La próxima semana tendrás una nueva reunión con los gerentes de marketing. Un colega de su equipo hizo modificaciones a la base de datos, pero no recuerda cómo las hizo. Te pide ayuda para dejar los códigos para obtener el siguiente diagrama:

El diagrama de la izquierda muestra las características actuales del esquema de datos `transactions` y en la de la derecha se muestra el esquema al cual se debe llegar.



Las diferencias se encuentran en la cantidad de tablas dentro del esquema, en el nombre y el tipo de variables, y en la presencia de algunas columnas. Para llegar al esquema que había modificado nuestro compañero de trabajo, se realizaron distintos cambios en cada una de las tablas, los cuales se presentan a continuación.

Tabla `user` / `data_user`

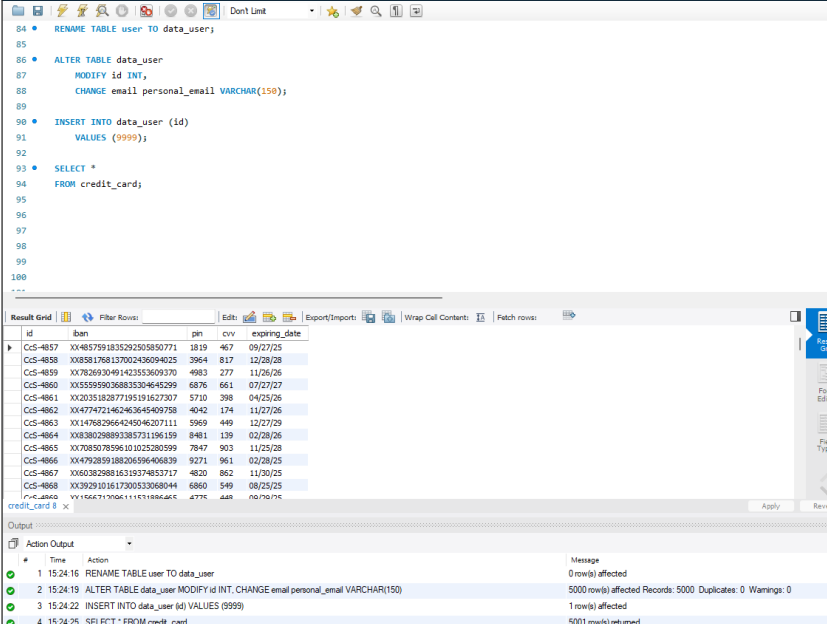
El primer paso consiste en la creación de la tabla `user` y la población de sus registros mediante la ejecución de los scripts "`estructura datos user.sql`" y "`datos introducir sprint3 user.sql`", disponibles en el directorio del proyecto.

Posteriormente, se procedió a renombrar la tabla `user` como `data_user` utilizando la instrucción `RENAME TABLE`. Sobre esta nueva estructura, se aplicó una sentencia `ALTER TABLE` para modificar el tipo de dato de la columna `id` a `INT`. Asimismo, se renombró la columna `email` por `personal_email`, ajustando su extensión máxima y restricciones según los requerimientos.

Además, se insertó un nuevo registro con el `user_id = 'b-9999'`, para cumplir con los requerimientos del ejercicio 3 del nivel 1.

El código utilizado y el resultado de los cambios en la tabla se muestran en la siguiente imagen:

Código de cambios en tabla data_user.



The screenshot displays a database management interface. The top section shows a list of SQL statements being executed:

```
84 • RENAME TABLE user TO data_user;
85
86 • ALTER TABLE data_user
87   MODIFY id INT,
88   CHANGE email personal_email VARCHAR(150);
89
90 • INSERT INTO data_user (id)
91   VALUES (9999);
92
93 • SELECT *
94   FROM credit_card;
```

Below the code, the 'Result Grid' shows the data returned by the SELECT statement. It contains 10 rows with columns: id, iban, pin, cvv, and expiring_date.

	id	iban	pin	cvv	expiring_date
CS-4857	XX4857591835292505850771	1819	467	09/27/25	
CS-4858	X08581768137002436094025	3964	817	12/28/28	
CS-4859	XX7826930491423553609370	4983	277	11/26/26	
CS-4860	XX5559590368835304645299	6876	661	07/27/27	
CS-4861	XX2035362077195191627307	5710	398	04/25/26	
CS-4862	XX4747271462463645409758	4042	174	11/27/26	
CS-4863	XX147682966424046207111	5969	449	12/27/29	
CS-4864	XX8380298893385731196159	8481	139	02/28/26	
CS-4865	XX7085076596101025300599	7847	903	11/25/28	
CS-4866	XX4792859188206596406839	9271	961	02/28/25	
CS-4867	XX6038298816319374853717	4820	862	11/30/25	
CS-4868	XX3929101617300533068044	6860	549	08/25/25	

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

#	Time	Action	Message
1	15:24:16	RENAME TABLE user TO data_user	0 row(s) affected
2	15:24:19	ALTER TABLE data_user MODIFY id INT, CHANGE email personal_email VARCHAR(150)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
3	15:24:22	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected
4	15:24:25	SELECT * FROM credit_card	5001 row(s) returned

Tras estas modificaciones, se estableció la relación entre las tablas mediante una restricción de clave foránea (FOREIGN KEY). Este código vincula la columna `user_id` de la tabla `transaction` con el identificador único (`id`) de la tabla `data_user`, garantizando que cada transacción esté asociada a un usuario existente.

Relación de la tabla transaction y data_user.

The screenshot shows a SQL IDE window with a script editor and an output pane. The script editor contains the following SQL code:

```
ALTER TABLE transaction
ADD CONSTRAINT fk_data_user
FOREIGN KEY (user_id) REFERENCES data_user(id);
```

The output pane shows the execution results of the script:

#	Time	Action	Message
1	15:24:16	RENAME TABLE user TO data_user	0 row(s) affected
2	15:24:19	ALTER TABLE data_user MODIFY id INT, CHANGE email personal_email VARCHAR(150)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
3	15:24:22	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected
4	15:24:25	SELECT * FROM credit_card	5001 row(s) returned
5	15:25:24	ALTER TABLE transaction ADD CONSTRAINT fk_data_user FOREIGN KEY (user_id) REFERENCES data_user(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Tabla company

continuación, se optimizó la estructura de la tabla **company** eliminando información no requerida mediante la sentencia **DROP COLUMN** sobre la columna **website**. Finalmente, se realizó una consulta general (**SELECT ***) para verificar que la estructura se actualizó correctamente.

Código de cambios en tabla company.

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

- SQL Editor:** Contains the following queries:


```

      102 ALTER TABLE company;
      103 DROP COLUMN website;
      104
      105 SELECT *
      106 FROM company;
      107
      108
      109
      110
      111
      112
      113
      114
      115
      116
      117
      118
      
```
- Result Grid:** Displays the results of the SQL queries. The first query shows the structure of the 'company' table, and the second query shows the data rows.

id	company_name	phone	email	country
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.por@titor.tellus@yahoo.net	Germany
b-2226	Magna A Neque Industries	04 14 44 64 62	rius.donec.nbh@icloud.org	Australia
b-2230	Fusce Corp.	08 14 97 58 85	rius@protonmail.edu	United States
b-2234	Conville In Incorporated	06 66 57 29 50	mauris.ut@ad.co.uk	Germany
b-2238	Ante Laculis Nec Foundation	08 23 04 99 53	sed.dictum.pron@outlook.ca	New Zealand
b-2242	Donec Ltd	01 25 51 37 37	at.laculis@hotmail.co.uk	Norway
b-2246	Sed Nunc Ltd	02 62 64 73 48	nbh@yahoo.org	United Kingdom
b-2250	Amet Nula Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy
b-2254	Nascentur Biddulus Mus Inc.	06 26 87 61 84	suspendisse.du@icloud.net	United States
b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.mass@integer@aol.net	Belgium
b-2262	Gravida Sagitta LLP	03 81 28 33 97	turpis.vitae@google.ca	Sweden
b-2266	Mus Aenean Eget Foundation	06 25 15 52 43	mi.duis@hotmail.net	Sweden
b-2270	The Quam element Nuncius Ltd	05 35 76 76 74	mi.duis@protonmail.com	Ireland
- Output:** Shows the execution log of the queries.

#	Time	Action	Message
3	15:24:22	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected
4	15:24:25	SELECT * FROM credit_card	5001 row(s) returned
5	15:25:24	ALTER TABLE transaction ADD CONSTRAINT fk_data_user FOREIGN KEY (user_id) REFERENCES data_user(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
6	15:26:33	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
7	15:26:42	SELECT * FROM company	101 row(s) returned

Tabla credit_card

En la tabla **credit_card**, se realizaron ajustes mediante la instrucción **MODIFY** para adaptar la longitud y el tipo de dato de las columnas **id**, **iban**, **pin** y **cvv** a los establecidos en el esquema requerido. Asimismo, se incorporó una nueva columna denominada **fecha_actual** de tipo **DATE**, configurando un valor por defecto (**DEFAULT**) que registra automáticamente la fecha del sistema al insertar nuevos registros.

El código utilizado y el resultado de los cambios en la tabla se muestran en la siguiente imagen:

Código de cambios en tabla credit_card.

ALTER TABLE credit_card
 MODIFY id VARCHAR (20) NOT NULL,
 MODIFY iban VARCHAR (50),
 MODIFY pin VARCHAR (4),
 MODIFY cvv INT,
 ADD fecha_actual DATE DEFAULT (CURRENT_DATE);
 SELECT *
 FROM credit_card;

id	iban	pin	cvv	expiring_date	fecha_actual
CG-4857	XX4857591835292505850771	1819	467	09/27/25	2025-12-16
CG-4858	XX8581768137002436094025	3964	817	12/28/28	2025-12-16
CG-4859	XX7826930491423353609370	4983	277	11/26/26	2025-12-16
CG-4860	XX555959636835304645299	6876	661	07/27/27	2025-12-16
CG-4861	XX2035182877195191627307	5710	398	04/25/26	2025-12-16
CG-4862	XX4774721462463645409758	4042	174	11/27/26	2025-12-16
CG-4863	XX1476829664245046207111	5969	449	12/27/29	2025-12-16
CG-4864	XX826029693385731196159	8481	139	02/26/26	2025-12-16
CG-4865	XX7085078596101025280599	7847	903	11/25/28	2025-12-16
CG-4866	XX4792859188206596406839	9271	961	02/28/25	2025-12-16
CG-4867	XX6038298816319374653717	4820	862	11/30/25	2025-12-16
CG-4868	XX3929101617300533086244	6860	549	08/25/25	2025-12-16
CG-4869	XX1666711066111671886246	4976	448	06/20/26	2025-12-16

credit_card 10 x

Output

#	Time	Action	Message
5	15:25:24	ALTER TABLE transaction ADD CONSTRAINT fk_data_user FOREIGN KEY (user_id) REFERENCES data_user(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
6	15:26:33	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
7	15:26:42	SELECT * FROM company	101 row(s) returned
8	15:27:44	ALTER TABLE credit_card MODIFY id VARCHAR (20) NOT NULL MODIFY iban VARCHAR (50), MODIFY pin VARCHAR (4), MO...	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
9	15:27:47	SELECT * FROM credit_card	5001 row(s) returned

Ejercicio 2

La compañía también le pide que cree una vista llamada InformeTecnico que contiene la siguiente información:

ID de transacción
Nombre de usuario
Apellido de usuario
IBAN de la tarjeta de crédito usada
Nombre de la empresa de la transacción realizada

Asegúrese de incluir información relevante de las tablas que conocerá y utilice alias para cambiar el nombre de las columnas según sea necesario.

Muestra los resultados de la vista, ordena los resultados hacia abajo en función de la variable de ID de transacción.

Al igual que en el ejercicio 2 del nivel 2, se crea la vista mediante la instrucción `CREATE VIEW` y en la cláusula `SELECT` se indican las variables solicitadas. Además, Se utiliza la variable `transaction_id` para cumplir con el requisito de ordenación `ORDER BY` descendente (`DESC`).

Se han incluido las variables `timestamp`, `personal_email`, `country` y `city` para facilitar la gestión de las transacciones. `timestamp` permite identificar el momento exacto de la operación y el email funciona como medio de contacto directo con el cliente en caso de errores. Además, la información de ciudad y país permite conocer la zona horaria del usuario, algo muy útil para asegurar que cualquier comunicación se realice en un horario adecuado.

Creación y consulta de la vista InformeTecnico

The screenshot displays a database management interface with a SQL editor at the top and a results grid at the bottom.

SQL Editor:

```
121 CREATE VIEW InformeTecnico AS
122 SELECT t.id AS transaction_id, timestamp, c.company_name, u.name AS user_name, u.surname AS user_surname, u.personal_email, cc.iban, u.country, u.city
123 FROM transaction AS t
124 INNER JOIN data_user AS u
125     ON t.user_id = u.id
126 INNER JOIN credit_card AS cc
127     ON t.credit_card_id = cc.id
128 INNER JOIN company AS c
129     ON t.company_id = c.id
130 ORDER BY t.id DESC;
131
132 SELECT * FROM InformeTecnico;
133
134
135
136
137
```

Results Grid:

id	company_name	phone	email	country
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany
b-2226	Magna A Neque Industries	04 14 44 64 62	maus.donec.rnib@icloud.org	Australia
b-2230	Puente Corp	08 14 97 88 85	maus@protonmail.edu	United States
b-2234	Convalis In Incorporated	06 66 57 29 50	mauris.ut@aol.co.uk	Germany
b-2238	Ante Lacus Nec Foundation	08 23 04 99 53	sed.dictum.prim@outlook.ca	New Zealand
b-2242	Donec Ltd	01 25 51 37 37	at.lacus@protonmail.co.uk	Norway
b-2246	Sed Nunc Ltd	02 62 64 73 48	rnib@yahoo.org	United Kingdom
b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy
b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@icloud.net	United States
b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.massa.integer@aol.net	Belgium
b-2262	Gravida Sagittis LLP	03 81 28 33 97	curpis_visee@google.ca	Sweden
b-2266	Mus Aenean Eget Foundation	06 25 15 52 43	mi.dui@protonmail.net	Sweden
b-2270	Pro Darkusmet Turbida	04 16 16 78 74	mauris@protonmail.com	Ireland

Action Output:

#	Time	Action	Message
1	15:29:20	CREATE VIEW InformeTecnico AS SELECT t.id AS transaction_id, timestamp, c.company_name, u.name AS user_name, u.surname AS user_surname, u.personal_email, cc.iban, u.country, u.city FROM transaction AS t INNER JOIN data_user AS u ON t.user_id = u.id INNER JOIN credit_card AS cc ON t.credit_card_id = cc.id INNER JOIN company AS c ON t.company_id = c.id ORDER BY t.id DESC;	0 row(s) affected
2	15:29:24	SELECT * FROM InformeTecnico;	101 row(s) returned