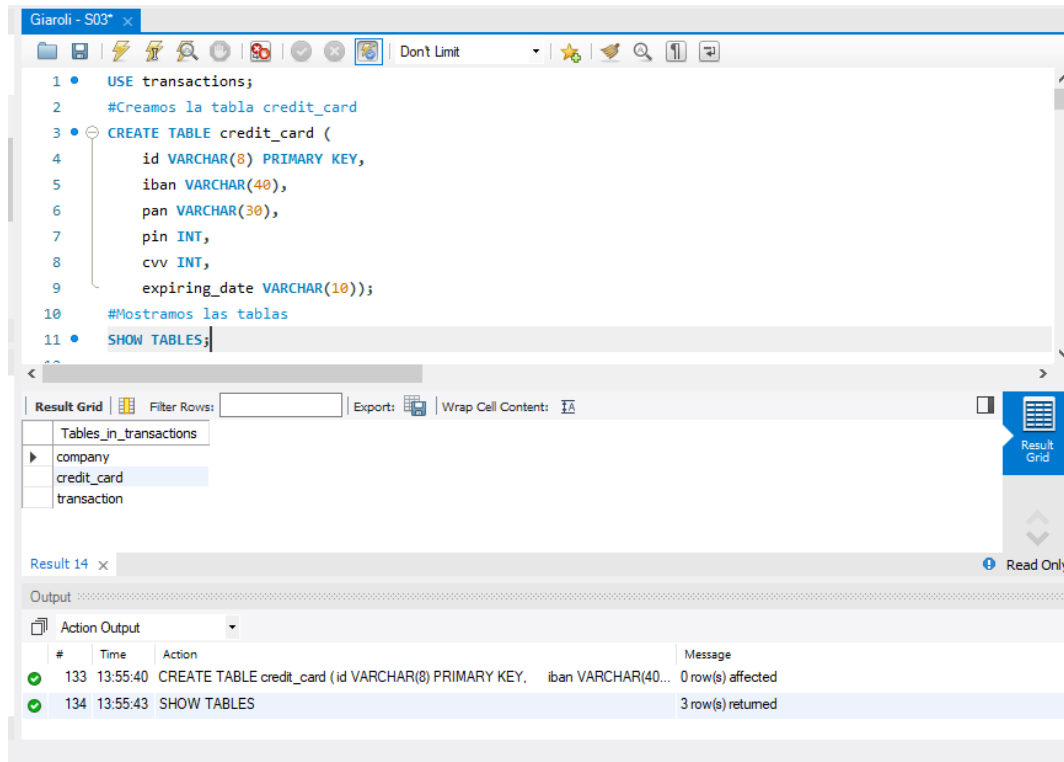


Nivel 1

Ejercicio 1

Creamos la tabla “credit_card” con el siguiente comando:



```
1 • USE transactions;
2 • #Creamos la tabla credit_card
3 • CREATE TABLE credit_card (
4     id VARCHAR(8) PRIMARY KEY,
5     iban VARCHAR(40),
6     pan VARCHAR(30),
7     pin INT,
8     cvv INT,
9     expiring_date VARCHAR(10));
10 • #Mostramos las tablas
11 • SHOW TABLES;
```

The screenshot shows a SQL IDE window titled 'Giaroli - S03*' with a toolbar and a 'Don't Limit' dropdown. The SQL editor contains the code above. Below the editor, the 'Result Grid' tab is active, showing a list of tables: 'Tables_in_transactions', 'company', 'credit_card', and 'transaction'. The 'credit_card' table is selected. Below the table list, the 'Output' tab is active, showing the results of the SQL execution. The output is as follows:

#	Time	Action	Message
133	13:55:40	CREATE TABLE credit_card (id VARCHAR(8) PRIMARY KEY, iban VARCHAR(40...	0 row(s) affected
134	13:55:43	SHOW TABLES	3 row(s) returned

Decidimos que los campos id, iban y pan sean de tipo VARCHAR porque además de números pueden incluir espacios en blanco. Pin y cvv son de tipo INTEGER y expiring_date también será VARCHAR, pero luego de insertar los datos cambiaremos el formato para que puedan corresponderse a los de tipo DATE.

Definimos **id como PK**. Y en otra consulta, definimos también a id como la FK de la tabla transaction donde transaction.credit_card_id=credit_card.id. Ambas tablas tienen una **relación de N a 1 (transaction->credit_card)**, porque puede haber varias comprar realizadas con una misma tarjeta.

Corrección: eliminamos FK anterior y añadimos FK que hace referencia a la tabla credit_card en lugar de a transactions:

```

16 #Creamos la FK de transactions
17 • SHOW CREATE TABLE credit_card;
18 • ALTER TABLE credit_card
19   DROP FOREIGN KEY `fk_transaction_credit_card_id`;
20 • DROP INDEX idx_transaction_credit_card_id ON transaction;
21
22 • ALTER TABLE transaction
23   ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);
24
25 • DESCRIBE transaction;
  
```

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

Result 3 x

#	Time	Action	Message
✓ 18	10:24:29	ALTER TABLE transaction ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);	586 row(s) affected Records: 586 Duplicates: 0 Warnings: 0
✓ 19	10:26:12	DESCRIBE transaction;	9 row(s) returned

Una vez creada la tabla e ingresados los datos, podemos modificar los datos de expiring_date para lograr que sean de tipo DATE con los siguientes comandos.

```

302 #Modificamos formato del campo expiring_date para que tenga formato de tipo DATE en una nueva columna y luego
303 ## pero antes desactivamos el modo seguro de actualización y al final de la consulta lo volvemos a activar para
304 • SET SQL_SAFE_UPDATES = 0;
305 • ALTER TABLE credit_card ADD expiring_date_ok DATE;
306 • UPDATE credit_card SET expiring_date_ok = str_to_date(expiring_date, '%m/%d/%y');
307 • ALTER TABLE credit_card DROP COLUMN expiring_date;
308 • ALTER TABLE credit_card CHANGE expiring_date_ok expiring_date DATE;
309 • SET SQL_SAFE_UPDATES = 1;
310 • SELECT * FROM credit_card;
  
```

id	iban	pan	pin	cvv	expiring_date
CdU-2938	TR301950312213576817638661	5424465566813633	3257	984	2022-10-30
CdU-2945	DO26854763748537475216568689	5142423821948828	9080	887	2023-08-24
CdU-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	2021-06-29
CdU-2959	CR7242477244335841535	372461377349375	3583	667	2023-02-24
CdU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	2024-10-29

credit_card 3 x

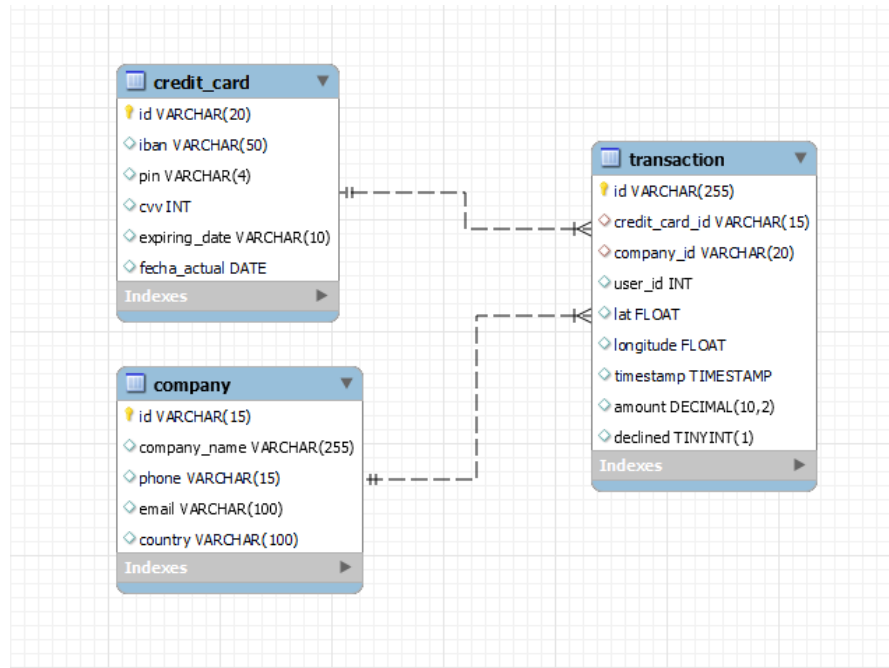
#	Time	Action	Message
✗ 12	15:59:01	ALTER TABLE credit_card CHANGE expiring_date_ok expiring_date	Error Code: 1064. You have an error in your SQL syntax; check the n
✓ 13	15:59:08	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✗ 14	15:59:13	ALTER TABLE credit_card CHANGE expiring_date_ok expiring_date	Error Code: 1064. You have an error in your SQL syntax; check the n
✓ 15	15:59:28	ALTER TABLE credit_card CHANGE expiring_date_ok expiring_date DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 16	15:59:37	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
✓ 17	15:59:49	SELECT * FROM credit_card	275 row(s) returned

Para evitar futuros errores podríamos pedir al cliente que los nuevos datos de fechas de caducidad se registren con formato YYYY-MM-DD.

En el caso de que esto no sea posible, tal vez sería conveniente continuar con el formato VARCHAR o buscar alguna solución alternativa.

La **base de datos “Transactions”** queda entonces compuesta por **tres tablas: “Company”, “Credit_Card” y “Transaction”**.

DB TRANSACTIONS



Company y Transaction ya han sido comentadas en el Sprint1. La nueva tabla incorporada al modelo es Credit_Card, compuesta por 6 campos que permiten identificar cada tarjeta de crédito por su id. Este campo actúa como PK de la tabla y a la vez como FK con la tabla “Transaction”. Además, la tabla contiene información sobre el iban, pan, pin, cvv y fecha de caducidad de cada tarjeta.

La base de datos queda conformada como un **modelo estrella**, con la tabla **Transaction como tabla de hechos** y las otras dos como tablas de dimensiones.

Ejercicio 2

Para actualizar el IBAN del usuario con ID CcU-2938 realizamos los siguientes comandos:

The screenshot shows a SQL IDE window titled "Giaroli - S03" with a file named "SQL File 3". The script contains the following SQL commands:

```
318
319 #corregimos el dato
320 • UPDATE credit_card
321 SET iban = 'R323456312213576817699999'
322 WHERE id = 'CcU-2938';
323
324 #verificamos corrección
325 • SELECT id, iban
326 FROM credit_Card
327 WHERE id = 'CcU-2938';
328
```

Below the script, the "Result Grid" shows the results of the queries:

id	iban
CcU-2938	R323456312213576817699999
NULL	NULL

The "Output" pane shows the "Action Output" for the executed queries:

#	Time	Action	Message
1	10:10:20	SELECT id, iban FROM credit_Card WHERE id = 'CcU-2938'	1 row(s) returned
2	10:12:45	UPDATE credit_card SET iban = 'R323456312213576817699999' WHERE id = 'CcU...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
3	10:13:14	SELECT id, iban FROM credit_Card WHERE id = 'CcU-2938'	1 row(s) returned

Ejercicio 3

Para poder ingresar un nuevo registro en la tabla Transaction, activamos y desactivamos el modo seguro de actualización con SET FOREIGN KEY CHECKS.

Luego actualizamos la tabla con el nuevo registro con el comando INSERT INTO. Al verificar los datos introducidos vemos que no se ha impreso timestamp así que lo hacemos en otra query (UPDATE).

Finalmente, revisamos que los datos introducidos sean correctos.

The screenshot shows a SQL IDE window titled "Giaroli - S03" with a file named "dades_introduir". The script contains the following SQL commands:

```

330 • SET FOREIGN_KEY_CHECKS = 0;
331 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined)
332 • SET FOREIGN_KEY_CHECKS = 1;
333
334 • UPDATE transaction
335   SET timestamp = now()
336   WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';
337
338 • SELECT *
339   FROM transaction
340   WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';

```

Below the script, the "Result Grid" shows the results of the SELECT query:

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
▶	10881D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	2024-03-08 10:41:45	111.11	0
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The "Output" pane shows the execution log:

#	Time	Action	Message
21	10:41:45	UPDATE transaction SET timestamp = now() WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
22	10:41:56	SELECT * FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';	1 row(s) returned

Ejercicio 4

Para eliminar la columna pan de la tabla credit_card realizamos los siguientes comandos:

The screenshot shows a SQL IDE window titled "Giaroli - S03" with a file named "SQL File 3" and a tab "dades_introduir". The SQL editor contains the following commands:

```
341
342
343 #N1.Exercici 4: eliminar la columna "pan" de la taula credit_card.
344 • ALTER TABLE credit_card DROP COLUMN pan;
345 • SHOW COLUMNS FROM credit_card;
346
347
348
349
```

Below the editor, the "Result Grid" shows the structure of the "credit_card" table:

Field	Type	Null	Key	Default	Extra
id	varchar(8)	NO	PRI	NULL	
iban	varchar(40)	YES		NULL	
pin	int	YES		NULL	
cvv	int	YES		NULL	
expiring_date	date	YES		NULL	

The "Output" pane shows the execution log:

#	Time	Action	Message
23	10:52:10	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
24	10:52:46	SELECT COLUMNS FROM credit_card	Error Code: 1054. Unknown column 'COLUMNS' in 'field list'
25	10:53:00	SHOW COLUMNS FROM credit_card	5 row(s) returned

Nivel 2

Ejercicio 1

Para eliminar de la tabla Transaction el registro con ID 02C6201E-D90A-1859-B4EE-88D2986D3B02, realizamos lo siguiente:

The screenshot shows a database management interface with a SQL editor and a results pane. The SQL editor contains the following code:

```
343
344
345  ### Nivell 2
346  #N2.Exercici 1: Elimina de la taula transaction el registre amb ID 02C6201E-D90A-1859-B4EE-88D2986D3B02
347  • DELETE FROM transaction
348    WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
349
350  #corroboramos borrado de datos
351  • SELECT *
352    FROM transaction
353    WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
```

Below the editor is a toolbar with options like "Result Grid", "Filter Rows", "Edit", "Export/Import", and "Wrap Cell Content". A table structure is visible with columns: id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, and declined. All cells are currently empty.

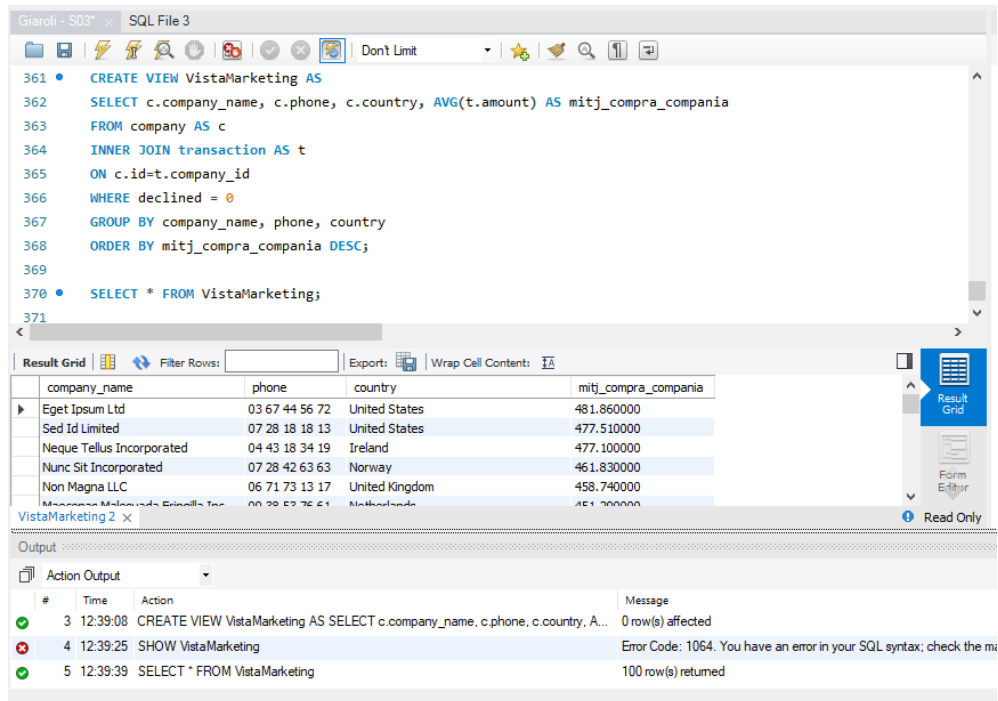
The bottom pane shows the "Output" section with "Action Output" selected. It displays two rows of execution results:

#	Time	Action	Message
✓ 26	10:38:18	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02'	0 row(s) affected
✓ 27	10:38:29	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02'	0 row(s) returned

Ejercicio 2

Para crear la vista VistaMarketing tenemos que crear una vista compleja, ya que utilizaremos datos de dos tablas, con funciones de agregación y agrupación.

En este caso tendremos en cuenta al booleano Declined ya que el enunciado pide las compras realizadas.



```
361 • CREATE VIEW VistaMarketing AS
362 SELECT c.company_name, c.phone, c.country, AVG(t.amount) AS mitj_compra_compania
363 FROM company AS c
364 INNER JOIN transaction AS t
365 ON c.id=t.company_id
366 WHERE declined = 0
367 GROUP BY company_name, phone, country
368 ORDER BY mitj_compra_compania DESC;
369
370 • SELECT * FROM VistaMarketing;
371
```

company_name	phone	country	mitj_compra_compania
Eget Ipsum Ltd	03 67 44 56 72	United States	481.860000
Sed Id Limited	07 28 18 18 13	United States	477.510000
Neque Tellus Incorporated	04 43 18 34 19	Ireland	477.100000
Nunc Sit Incorporated	07 28 42 63 63	Norway	461.830000
Non Magna LLC	06 71 73 13 17	United Kingdom	458.740000
Magnis Mauris Egesta Inc	00 38 53 76 61	Netherlands	451.700000

Output

#	Time	Action	Message
3	12:39:08	CREATE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c.country, A...	0 row(s) affected
4	12:39:25	SHOW VistaMarketing	Error Code: 1064. You have an error in your SQL syntax; check the mi
5	12:39:39	SELECT * FROM VistaMarketing	100 row(s) returned

Ejercicio 3

Para filtrar de la vista VistaMarketing sólo a las compañías con sede en Alemania realizamos la siguiente consulta:

The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```
369
370 • SELECT * FROM VistaMarketing;
371
372 #N2.Exercici 3: Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de res
373 • SELECT *
374 FROM VistaMarketing
375 WHERE country = 'Germany';
376
377
378
```

The results pane displays a table with the following data:

company_name	phone	country	mitj_compra_compania
Ac Industries	09 34 65 40 60	Germany	396.150000
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.990000
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.570000
Aliquam PC	01 45 73 52 16	Germany	280.340000
Rutrum Non Inc.	02 66 31 61 09	Germany	266.900000
Aliquam Teneant Incorporated	05 12 15 48 12	Germany	347.647600

The Output pane shows the execution log:

#	Time	Action	Message
5	12:39:39	SELECT * FROM VistaMarketing	100 row(s) returned
6	12:45:14	SELECT company_name, avg(amount) AS gasto_medio FROM company INNER JOI...	1 row(s) returned
7	12:46:49	SELECT * FROM VistaMarketing WHERE country = 'Germany'	8 row(s) returned

Nivel 3

Ejercicio 1

Para obtener el modelo requerido haremos los siguientes cambios:

- 1) eliminamos el campo website de la tabla company

SQL File 3: Giaroli S1.01

```

377  ### Nivell 3
378  #N3.Exercici 1:
379  #Para obtener el modelo requerido haremos los siguientes cambios:
380
381  #1)eliminamos el campo website de la tabla company
382  • ALTER TABLE company DROP COLUMN website;
383  • SELECT * FROM company;
  
```

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	risus.donec.nibh@idoud.org	Australia
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany
b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.proin@outlook.ca	New Zealand
b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.couk	Norway
b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@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@idoud.net	United States
b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.massaa.integer@aol.net	Belgium

Output: Action Output

#	Time	Action	Message
7	12:46:49	SELECT * FROM VistaMarketing WHERE country = 'Germany'	8 row(s) returned
8	13:08:38	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
9	13:08:48	SELECT * FROM company	100 row(s) returned

- 2) en la tabla credit_card cambiamos el formato de 4 campos y añadimos uno nuevo.

SQL File 4: Giaroli S04

```

386  • SET SQL_SAFE_UPDATES = 0;
387  • ALTER TABLE credit_card
388      ADD fecha_actual DATE DEFAULT (CURRENT_DATE),
389      MODIFY COLUMN id VARCHAR(20),
390      MODIFY COLUMN iban VARCHAR(50),
391      MODIFY COLUMN pin VARCHAR(4),
392      MODIFY COLUMN expiring_date VARCHAR(10);
393  • SET SQL_SAFE_UPDATES = 1;
394  ##corroboramos los cambios realizados:
395  • DESCRIBE credit_card;
396  • SELECT * FROM credit_card;
  
```

id	iban	pin	cvv	expiring_date	fecha_actual
CcU-2938	R323456312213576817699999	3257	984	2022-10-30	2024-03-14
CcU-2945	DO26854763748537475216568689	9080	887	2023-08-24	2024-03-14
CcU-2952	BG45IVQL52710525608255	4598	438	2021-06-29	2024-03-14
CcU-2959	CR7242477244335841535	3583	667	2023-02-24	2024-03-14
CcU-2966	BG72LKTQ15627628377363	4900	130	2024-10-29	2024-03-14

Output: Action Output

#	Time	Action	Message
81	13:39:20	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
82	13:39:23	DESCRIBE credit_card	6 row(s) returned
83	13:39:29	SELECT * FROM credit_card	275 row(s) returned

3) Creamos la tabla User

The screenshot shows a database IDE with the following SQL code in the editor:

```

403     phone VARCHAR(150),
404     email VARCHAR(150),
405     birth_date VARCHAR(100),
406     country VARCHAR(150),
407     city VARCHAR(150),
408     postal_code VARCHAR(100),
409     address VARCHAR(255),
410     FOREIGN KEY(id) REFERENCES transaction(user_id)
411 );
412 ##corroboramos creación de tabla user
413 • DESCRIBE user;
414

```

Below the code, the 'Result Grid' shows the structure of the 'user' table:

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	HULL	
name	varchar(100)	YES		HULL	
surname	varchar(100)	YES		HULL	
phone	varchar(150)	YES		HULL	
email	varchar(150)	YES		HULL	
birth_date	varchar(100)	YES		HULL	

The 'Output' pane shows the execution results:

#	Time	Action	Message
20	13:41:48	CREATE TABLE IF NOT EXISTS user (id INT PRIMARY KEY, name VAR...	0 row(s) affected
21	13:41:51	DESCRIBE user	10 row(s) returned

Corregimos la FK de user

The screenshot shows a database IDE with the following SQL code in the editor:

```

410 ##Corregimos FK de tabla user
411 • SHOW CREATE TABLE user;
412 • ALTER TABLE user DROP FOREIGN KEY `user_ibfk_1`;
413
414 #tenemos problemas para crearla por la transaccion agregada en el ejercicio 3, la eliminamos
415 • SELECT * FROM transaction WHERE user_id NOT IN (SELECT id FROM user);
416 • DELETE FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';
417
418 • ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES user(id);
419

```

Below the code, the 'Result Grid' shows the structure of the 'transaction' table:

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

The 'Output' pane shows the execution results:

#	Time	Action	Message
36	10:50:26	DELETE FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99...	1 row(s) affected
37	10:50:28	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES user(id)	586 row(s) affected Records: 586 Duplicates: 0 Warnings: 0
38	10:50:49	DESCRIBE user	10 row(s) returned
39	10:51:14	DESCRIBE transaction	9 row(s) returned

a. Insertamos los datos

Girol - S03* x

```

689 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
690 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
691 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
692 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
693 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
694
695 • SET foreign_key_checks = 1;
696
697 ###corroboramos inserción de datos
698 • SELECT * FROM user;

```

Result Grid

	id	name	surname	phone	email	birth_date	country	city	postal_code
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544	
2	Garrett	Mconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	
3	Ciaran	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 8 x

Output

Action Output

#	Time	Action	Message
297	13:45:35	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_...	1 row(s) affected
298	13:45:42	SET foreign_key_checks = 1	0 row(s) affected
299	13:46:08	SELECT * FROM user	275 row(s) returned

b. Cambiamos el nombre del campo email a personal_email

Girol - S03* x

```

701 ###Cambiamos el nombre del campo email a personal_email
702 • SET SQL_SAFE_UPDATES = 0;
703 • ALTER TABLE user
704 CHANGE email personal_email VARCHAR(150);
705 • SET SQL_SAFE_UPDATES = 1;
706
707 ###corroboramos actualización del nombre del campo
708 • DESCRIBE user;

```

Result Grid

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

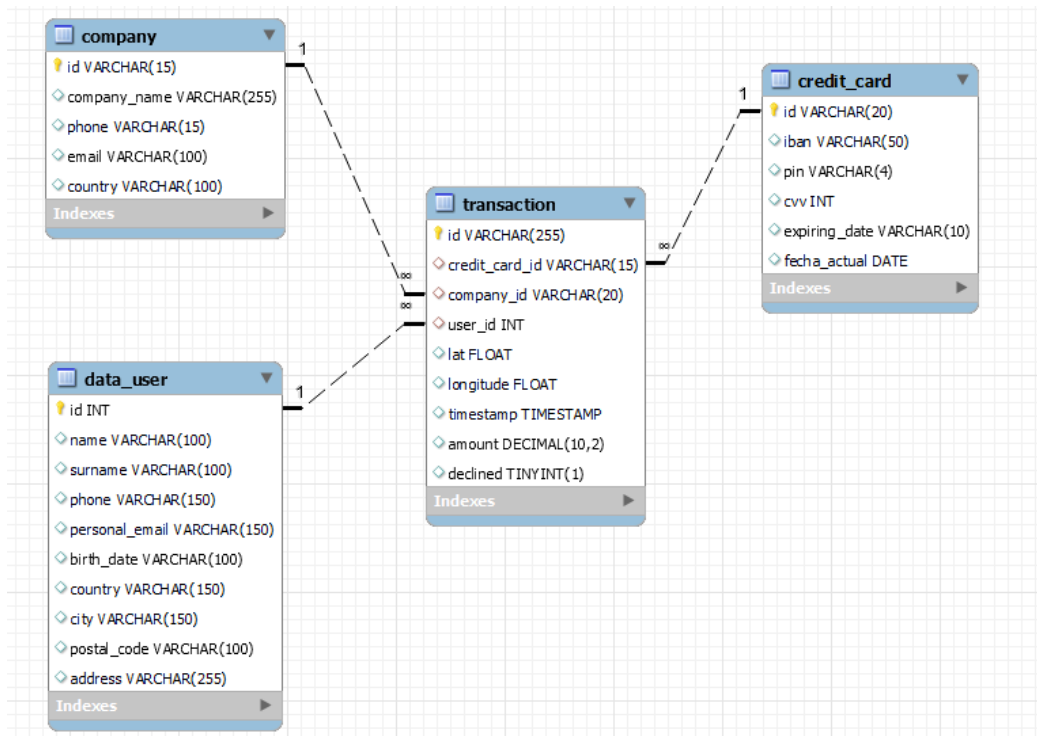
Result 9 x

Output

Action Output

#	Time	Action	Message
301	13:51:22	ALTER TABLE user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
302	13:51:24	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
303	13:52:05	DESCRIBE user	10 row(s) returned

4) Hacemos Reverse Data Engineer para obtener un diagrama del modelo.



Ejercicio 2

Creamos la vista InformeTecnico haciendo una Left Join para incluir todas las transacciones:

The screenshot shows the SQL Server Enterprise Manager interface with the following SQL script executed:

```

736
737  #Decidimos eliminar la vista creada y crear una nueva que incluya todas las transacciones usando LEFT JOIN
738  • DROP VIEW InformeTecnico;
739
740  • CREATE VIEW InformeTecnico AS
741  SELECT t.id as transaction_id, u.name as user_name, u.surname as user_surname, cc.iban, c.company_name
742  FROM transaction AS t
743  LEFT JOIN company AS c
744  ON c.id=t.company_id
745  LEFT JOIN credit_card AS cc
746  ON cc.id=t.credit_card_id
747  LEFT JOIN user AS u
748  ON u.id=t.user_id;
749
750
751  ###Corroboramos Vista creada
752
753  • SELECT * FROM InformeTecnico
754  ORDER BY transaction_id DESC;
  
```

The Output pane shows the results of the execution:

#	Time	Action	Message
352	15:33:33	DROP VIEW InformeTecnico	0 row(s) affected
353	15:33:38	CREATE VIEW InformeTecnico AS SELECT t.id as transaction_id, u.name as user_n...	0 row(s) affected

Verificamos la vista creada:

The screenshot shows the SQL Server Enterprise Manager interface with the following SQL script executed:

```

749
750
751  ###Corroboramos Vista creada
752
753  • SELECT * FROM InformeTecnico
754  ORDER BY transaction_id DESC;
755
  
```

The Output pane shows the results of the execution:

#	Time	Action	Message
353	15:33:38	CREATE VIEW InformeTecnico AS SELECT t.id as transaction_id, u.name as user_n...	0 row(s) affected
354	15:34:23	SELECT * FROM InformeTecnico ORDER BY transaction_id DESC	587 row(s) returned

The Result Grid shows the data returned by the query:

transaction_id	user_name	user_surname	iban	company_name
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E46468	Molly	Gilliam	SE2813123487163628531121	Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	MD1234119525145401270486	Nunc Interdum Incorporated
FBD7E0D6-BA6B-F5BC-0CA9-EA4B8760100C	Hedwig	Gilbert	MU4132333444534342541344788855	Mauris Id Inc.
FAC76A80-8448-69AA-E892-426C2F12621C	Slade	Poole	MT05JWCF58868200575771634583813	Arcu LLP
FAAD3FFC-1A17-E141-43D3-359A5BA7CB38	Hedwig	Gilbert	GE90157928843338134463	Lorem Eu Incorporated
FA053936-75D8-85FA-490D-9B624E1B920A	Hedwig	Gilbert	GT02497653655330848247645975	Non Justo Corp.