

Tasca S3.01. Manipulació de taules

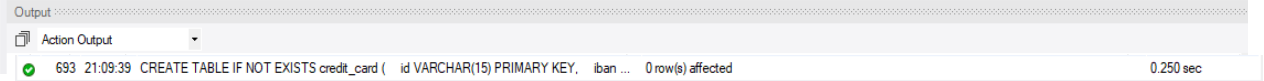
NIVEL 1

Ejercicio 1

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 ingrese la información del documento denominado "datos_introducir_credit". Recuerda mostrar el diagrama y realizar una breve descripción del mismo.

Para este primer ejercicio usaremos el código que se usa en el script de estructura de datos que proporciona la web, CREATE TABLE IF NOT EXISTS credit_card (vamos introduciendo cada variable con su tipo de dato). La primera variable (id) es la clave primaria y todas las columnas, a excepción de cvv que se establece como variable numérica INT, tienen formato VARCHAR para almacenar cadenas de caracteres de longitud variable.

```
1  #NIVEL 1
2  #Ejercicio 1
3  CREATE TABLE IF NOT EXISTS credit_card (
4      id VARCHAR(15) PRIMARY KEY,
5      iban VARCHAR(34),
6      pan VARCHAR(19),
7      pin VARCHAR(4),
8      cvv INT,
9      expiring_date VARCHAR(10)
10 );
```



Debemos asegurarnos que la tabla credit_card esté relacionada con transaction (1:N). Una credit_card tendrá varias transacciones pero una transacción está ligada a una credit_card. Para ello escribimos ALTER TABLE transaction y añadimos una restricción fk_credit_card seguido de FOREIGN KEY (credit_card_id) REFERENCES credit_card(id) para relacionar ambas tablas. La restricción se incluye para garantizar la integridad referencial entre dos tablas en una base de datos.

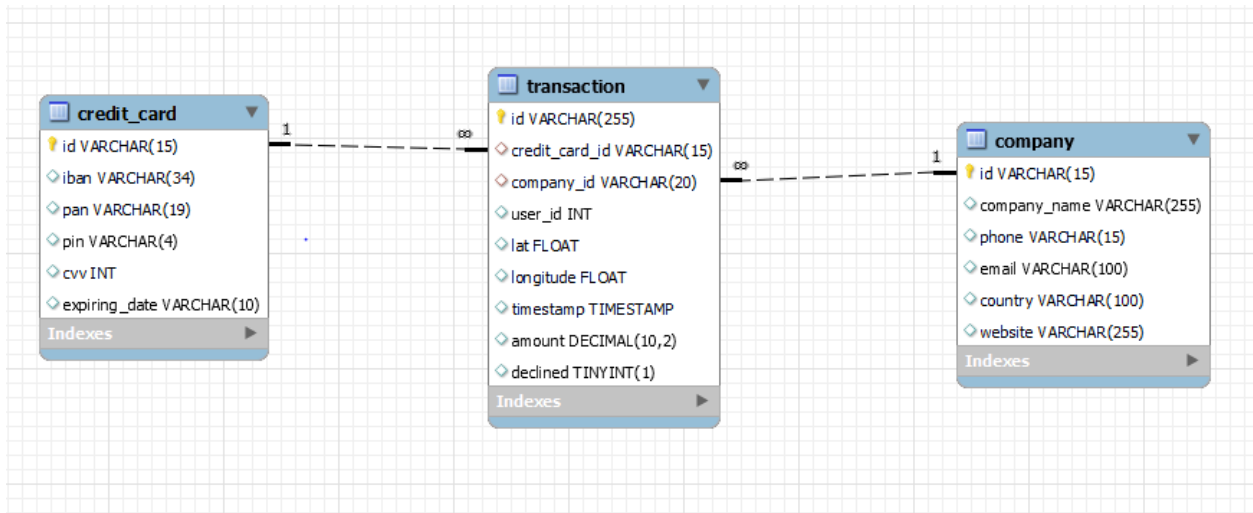
```

12 • ALTER TABLE transaction
13   ADD CONSTRAINT fk_credit_card
14   FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);

```

Output	
Action Output	
970 21:11:49 ALTER TABLE transaction ADD CONSTRAINT fk_credit_card FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0 2.594 sec

Para crear el diagrama iremos a Database, reverse engineer. Seleccionamos schema transactions para ver las tablas ya creadas en la tarea 2 junto a la nueva tabla credit_card. Hacemos click en next y finish hasta que se generen las 3 tablas.



Aquí vemos la tabla 'credit_card' ligada a transaction con la relación que queríamos (1:N). Ejecutamos el script datos_introducir_credit y hacemos una consulta rápida para ver que nuestra tabla creada almacena bien los valores a introducir.

```

12 • SELECT *
13 FROM credit_cards;
14

```


id	iban	pan	pin	cvv	expiring_date
CCU-2938	TR301950312213576817638661	5424465566613633	3257	984	10/30/22
CCU-2945	DO26854763748537475216568689	5142423821948828	9080	887	08/24/23
CCU-2952	BG451VQL52710525608255	4556 453 55 5287	4598	438	06/29/21
CCU-2959	CR724247224335841535	372461377949375	3583	667	02/24/23
CCU-2966	BG724721562762837763	448566 886747 7265	4900	130	10/29/24
CCU-2973	PT8780422813590242946436	544 58654 54343 384	8760	887	01/30/25
CCU-2980	DE39241881883086277136	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
CCU-3008	LU507216693616119230	4485744464433884	1856	740	04/05/25
CCU-3015	PS119398216295715968342456821	3784 662233 17389	3246	822	01/31/22
CCU-3022	GT91695162850556977423121857	5164 1379 4842 3951	5610	342	04/25/25
CCU-3029	AZ62317413982441418123739746	3429 279566 77631	9708	505	09/02/23
CCU-3036	AZ39336002925842865843941994	3768 451596 48766	2232	565	10/27/25
CCU-3043	TH6488143310514852179535	455676 6437463635	5969	196	06/07/25
CCU-3050	FR5167744369173836831854477	4024007123722	4834	125	10/09/23
CCU-3057	LU931822574697545215	3484 621767 21237	6805	848	09/14/25
CCU-3064	PS146965545449253377627273133	3467 732741 26810	3865	498	06/03/25
CCU-3071	NO8923814763512	3464 789562 23352	6625	661	12/20/23
CCU-3078	IS025127145884623279548733	4539 322 74 2377	9405	720	03/08/23
CCU-3085	BE63114723972437	5266 3346 1135 1687	7241	413	05/10/23
CCU-3092	RO6LS001166122125447487	3488 754223 46253	9417	594	12/19/22
CCU-3099	PT26105275356823705537218	448 55418 98863 789	5612	564	01/22/23
CCU-3106	AT64251637751136592	349547146395283	9733	209	01/27/24
CCU-3113	IE26LCGT477321739572752	34183482877471	9011	287	06/12/21
CCU-3120	RS72655766666166237144	527946 533375 5577	7658	265	01/16/21
CCU-3127	PT83533461438644242816864	4716 443 46 4368	8038	924	01/16/23
CCU-3134	BG23MYJQ5568951824779	5146 3453 9766 2168	7260	935	08/24/25
CCU-3141	CH4437804777669672438	3775 626726 45261	2923	330	05/11/24
CCU-3148	FI626140324677114	3733 238351 51810	2326	333	09/28/21
CCU-3155	AD2777204763277722050982	4532263578421	3015	779	01/12/22
CCU-3162	HU56074054826233730628233311	455666 645685 4443	5898	603	05/18/20

Ejercicio 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. La información que debe mostrarse para este registro es: R323456312213576817699999. Recuerda mostrar que el cambio se realizó.

Basta con escribir UPDATE credit_card para actualizar la tabla, SET iban = 'R323456312213576817699999' para registrar el cambio de valor en esta columna y WHERE id = 'Ccu-2938 para que identifique la fila donde debe cambiarse dicho iban.

```
15 #Ejercicio 2
16 • UPDATE credit_card
17 SET iban = 'R323456312213576817699999'
18 WHERE id = 'CcU-2938';
19
```



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
longitude	-117.999
amount	111.11
declined	0

Antes de hacer la inserción de una fila nueva en la tabla transaction debemos deshabilitar las restricciones de clave foránea usando SET FOREIGN_KEY_CHECKS = 0, si no lo hacemos nos dará error. Consultando online vi que este paso se realiza por varias razones, cuando se está realizando una carga masiva de datos en una base de datos, deshabilitarlas temporalmente puede acelerar el proceso de inserción. A veces, los datos deben insertarse en un orden que no respeta las restricciones de clave foránea, deshabilitar las restricciones permite realizar estas inserciones. También, cuando se están restaurando datos desde una copia de seguridad, es posible que sea necesario deshabilitar las verificaciones de clave foránea para evitar errores debido a referencias temporales inexistentes.

Una vez deshabilitada la restricción, usamos el mismo código que se usa en los scripts para introducir datos. INSERT INTO transaction (nombres de las columnas separados por comas) VALUES (valores de cada columna separados por comas y siguiendo el mismo orden que los nombres de las columnas). Finalmente, volvemos a habilitar las restricciones con SET FOREIGN_KEY_CHECKS = 1;

```
20 #Deshabilitar foreign key constraints
21 • SET FOREIGN_KEY_CHECKS = 0;
22
23 #Ejercicio 3
24 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
25 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);
26
27
28 #Habilitar foreign key constraints
29 • SET FOREIGN_KEY_CHECKS = 1;
30
31 • SELECT *
32 FROM transaction
```

Output			
Action Output			
#	Time	Action	Message
1	12:47:37	SET FOREIGN_KEY_CHECKS = 0	0 row(s) affected
2	12:47:40	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A9...	1 row(s) affected
3	12:47:42	SET FOREIGN_KEY_CHECKS = 1	0 row(s) affected

Ejercicio 4

Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit_card. Recuerda mostrar el cambio realizado. Usando ALTER TABLE credit_card DROP COLUMN pan eliminamos la columna. Si escribimos DESCRIBE credit_card comprobaremos que se ha eliminado.

```
24 #Ejercicio 4
25 • ALTER TABLE credit_card
26 DROP COLUMN pan;
27
28 • DESCRIBE credit_card;
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	N/A	
iban	varchar(34)	NO		N/A	
pin	varchar(4)	NO		N/A	
cvv	int	NO		N/A	
expiring_date	varchar(10)	NO		N/A	

Result 6

Output

#	Time	Action	Message
1	11:03:38	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	11:03:44	DESCRIBE credit_card	5 row(s) returned

NIVEL 2

Ejercicio 1

Elimina de la tabla transaction el registro con ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de datos. En lugar de eliminar una columna, queremos eliminar una fila, por tanto, usamos la cláusula DELETE FROM transaction con el criterio:

WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02' para identificar el registro a eliminar. Tener en cuenta que en este caso también debemos deshabilitar y habilitar foreign key

```
42 # NIVEL 2
43 #Ejercicio 1
44 #Deshabilitar foreign key constraints
45 • SET FOREIGN_KEY_CHECKS = 0;
46
47 • DELETE FROM transaction
48 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
49
50 #Habilitar foreign key constraints
51 • SET FOREIGN_KEY_CHECKS = 1;
52
```

Output

#	Time	Action	Message
1	11:00:26	SET FOREIGN_KEY_CHECKS = 0	0 row(s) affected
2	11:00:28	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02'	1 row(s) affected
3	11:00:30	SET FOREIGN_KEY_CHECKS = 1	0 row(s) affected

constraints, si no lo hacemos nos saldrá error al compartir clave foránea con la tabla user.

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. Promedio de compras realizadas por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compras.

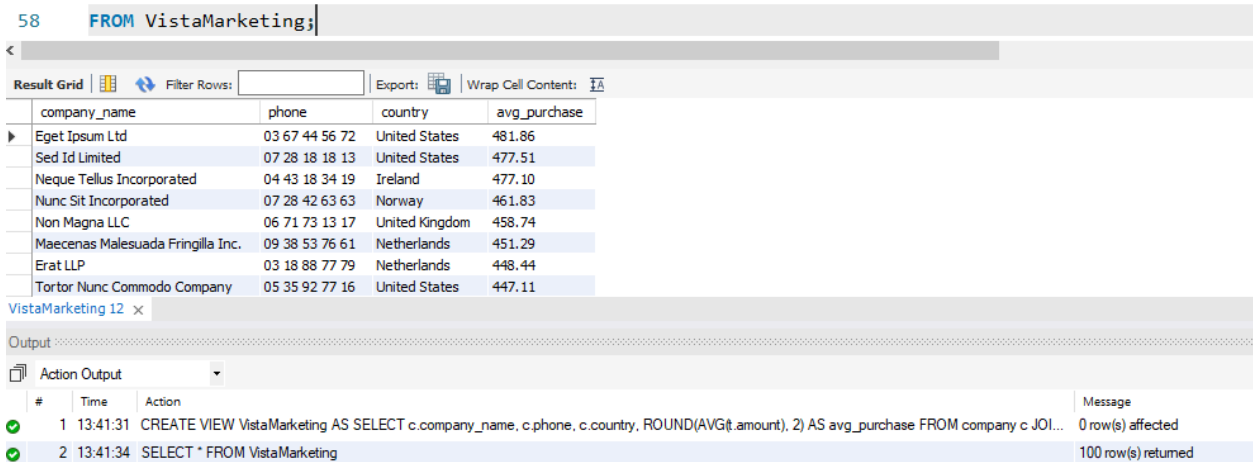
Para crear una vista, empezamos la query con la cláusula CREATE VIEW VistaMarketing AS. Después haremos la consulta como si fuese una query normal. Seleccionamos las columnas que nos indican precedidas por el alias de la tabla a la que pertenecen. Para el promedio de compras,

redondeamos a 2 decimales con `ROUND(,2)` y calculamos promedio con `AVG(t.amount)`. Renombramos el promedio usando alias `AS avg_purchase`. Usamos el mismo JOIN empleado en el sprint 2, enlazando por `id` y `company_id`. Nos aseguramos con el criterio `WHERE declined = 0` que cogemos solo las transacciones aceptadas. Agrupamos por nombre de compañía, teléfono y país. Ordenamos de mayor a menor promedio con `ORDER BY avg_purchase DESC`.

```

48 #Ejercicio 2
49 • CREATE VIEW VistaMarketing AS
50 SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount), 2) AS avg_purchase
51 FROM company c
52 JOIN transaction t ON c.id = t.company_id
53 WHERE t.declined = 0
54 GROUP BY c.company_name, c.phone, c.country
55 ORDER BY avg_purchase DESC;
56
57 • SELECT *
58 FROM VistaMarketing;

```



company_name	phone	country	avg_purchase
Eget Ipsum Ltd	03 67 44 56 72	United States	481.86
Sed Id Limited	07 28 18 18 13	United States	477.51
Neque Tellus Incorporated	04 43 18 34 19	Ireland	477.10
Nunc Sit Incorporated	07 28 42 63 63	Norway	461.83
Non Magna LLC	06 71 73 13 17	United Kingdom	458.74
Maecenas Malesuada Fringilla Inc.	09 38 53 76 61	Netherlands	451.29
Erat LLP	03 18 88 77 79	Netherlands	448.44
Tortor Nunc Commodor Company	05 35 92 77 16	United States	447.11

#	Time	Action	Message
1	13:41:31	CREATE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount), 2) AS avg_purchase FROM company c JOI...	0 row(s) affected
2	13:41:34	SELECT * FROM VistaMarketing	100 row(s) returned

Podemos comprobar la vista con `SELECT * FROM VistaMarketing;`

Ejercicio 3

Filtra la vista `VistaMarketing` para mostrar sólo las compañías que tienen su país de residencia en "Germany". Hacemos la misma consulta anterior incluyendo el criterio `WHERE country = 'Germany'`.

```
60 #Ejercicio 3
```

```
61 • SELECT *
```

```
62 FROM VistaMarketing
```

```
63 WHERE country = 'Germany';
```

```
64
```

Result Grid				
Filter Rows: Export: Wrap Cell Content:				
company_name	phone	country	avg_purchase	
Ac Industries	09 34 65 40 60	Germany	396.15	
Auctor Mauris Corp.	05 62 87 14 41	Germany	308.99	
Ac Fermentum Incorporated	06 85 56 52 33	Germany	293.57	
Aliquam PC	01 45 73 52 16	Germany	280.34	
Rutrum Non Inc.	02 66 31 61 09	Germany	266.90	
Nunc Interdum Incorporated	05 18 15 48 13	Germany	242.95	
Convallis In Incorporated	06 66 57 29 50	Germany	60.99	
Augue Foundation	06 88 43 15 63	Germany	15.05	

VistaMarketing 14 x

Output

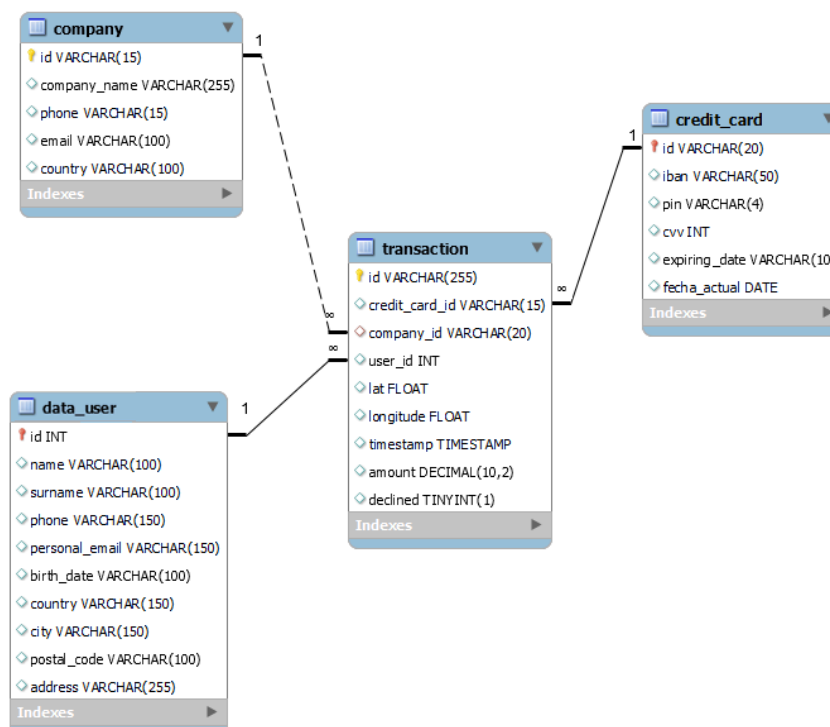
Action Output

#	Time	Action	Message
1	13:48:04	SELECT * FROM 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:



*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".

Lo primero que haremos será importar y cargar los archivos mencionados. Si generamos un diagrama, nos daremos cuenta que la relación entre la tabla user y transaction está invertida con respecto al diagrama con los cambios realizados por el compañero de trabajo. Debemos hacer un código que invierta dicha relación entre transaction y user de 1:N a N:1. Para ello, eliminamos primero la clave foránea id en la tabla user, eliminamos de la tabla transaction filas que no tienen correspondencia en la tabla user y luego haremos algo que ya hemos hecho anteriormente, añadir una nueva restricción y clave foránea en transaction. Para eliminar filas debemos desactivar primero el modo seguro con SET SQL_SAFE_UPDATES = 0 y luego reactivamos de nuevo.

```
79 # Eliminar la clave foránea existente en user
80 • SHOW CREATE TABLE user;
81 • ALTER TABLE user
82   DROP FOREIGN KEY user_ibfk_1;
83
84 # Desactivar el modo seguro
85 • SET SQL_SAFE_UPDATES = 0;
86 # Eliminar filas en 'transaction' con 'user_id' que no tienen correspondencia en 'user'
87 • DELETE FROM transaction
88   WHERE user_id NOT IN (SELECT id FROM user);
89 #Reactivar el modo seguro (opcional)
90 • SET SQL_SAFE_UPDATES = 1;
91
92 #Añadir restricción y clave foránea en transaction
93 • ALTER TABLE transaction
94   ADD CONSTRAINT fk_user
95   FOREIGN KEY(user_id) REFERENCES user(id);
96
```

Output

Action Output

#	Time	Action	Message
✓ 10	23:39:42	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✓ 11	23:39:44	DELETE FROM transaction WHERE user_id NOT IN (SELECT id FROM user)	1 row(s) affected
✓ 12	23:39:46	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
✓ 13	23:39:50	ALTER TABLE transaction ADD CONSTRAINT fk_user FOREIGN KEY(user_id) REFEREN...	586 row(s) affected Records: 586 Duplicates: 0 Warnings: 0

Aparte de tener una 4a tabla creada (user), los otros cambios que se perciben con respecto al diagrama anterior son:

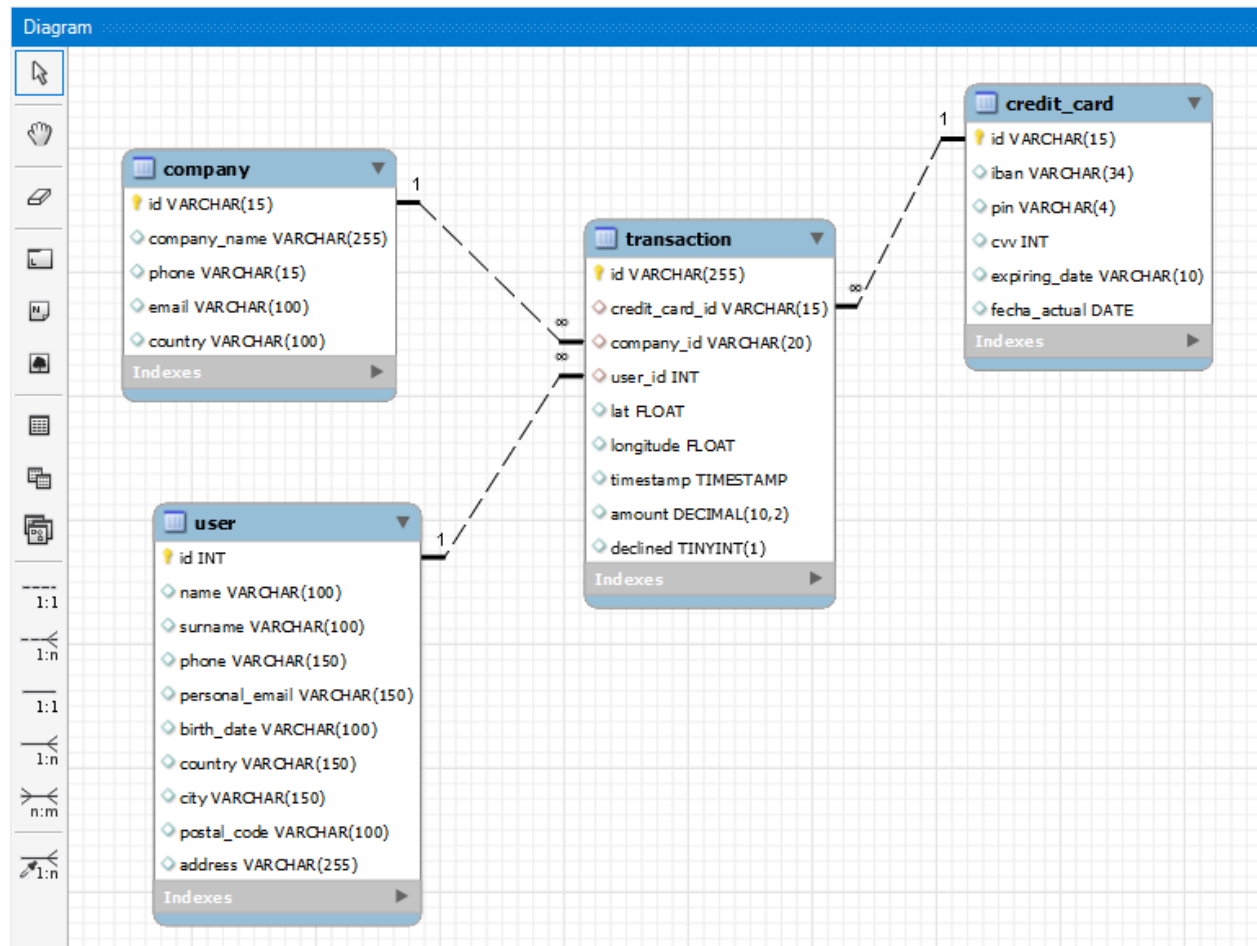
- 1) La columna 'website' de la tabla 'company' se ha eliminado, para ello usamos la cláusula `ALTER TABLE company` para informar que vamos a efectuar una modificación en dicha tabla y `DROP COLUMN website` para eliminar la columna que no aparece en el diagrama.
- 2) La tabla `credit_card` tiene la clave primaria `id` en rojo y una columna nueva llamada `fecha_actual` con `DATE` como tipo de dato. El color nos indica que se ha creado un índice para dicha variable, de modo que Usamos `ALTER TABLE credit_card` y añadimos columna con `ADD COLUMN fecha_actual DATE`.
- 3) La columna `email` de la tabla `user` se ha renombrado a `personal_email`. Por ello, usaremos de nuevo la cláusula `ALTER TABLE user` y `CHANGE nombre antiguo nombre nuevo` y el tipo de dato `VARCHAR(150)`.

Veamos como queda en MySQL y generamos el diagrama con Database, reverse engineer.

```
75 # Nivel 3
76 # Ejercicio 1
77 # Eliminar la columna 'website' de la tabla 'company'
78 • ALTER TABLE company
79 DROP COLUMN website;
80
81 # Agregar la columna 'fecha_actual' a la tabla 'credit_card'
82 • ALTER TABLE credit_card
83 ADD COLUMN fecha_actual DATE;
84
85 # Renombrar la columna 'email' a 'personal_email' en la tabla 'user'
86 • ALTER TABLE user
87 CHANGE email personal_email VARCHAR(150);
```

Output

#	Time	Action	Message
1	11:08:33	ALTER TABLE company DROP COLUMN website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	11:08:37	ALTER TABLE credit_card ADD COLUMN fecha_actual DATE	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	11:08:39	ALTER TABLE user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0



Ejercicio 2

La empresa también te solicita 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.

Asegúrate de incluir información relevante de ambas tablas y utiliza 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.

Creamos la vista con CREATE VIEW seleccionamos todas las variables que nos piden con el alias de cada tabla precediendo dicha variable. Hacemos 3 joins para unir las tablas user, credit_card y company con transaction por las variables que tengan en común y ordenamos por la id de la tabla transacción de mayor a menor valor usando DESC. Mostramos qué aspecto tiene la vista con SELECT * FROM InformeTecnico.

```
99 # Ejercicio 2
100 • CREATE VIEW InformeTecnico AS
101 SELECT t.id, u.name, u.surname, cc.iban, c.company_name
102 FROM transaction t
103 JOIN user u ON t.user_id = u.id
104 JOIN credit_card cc ON t.credit_card_id = cc.id
105 JOIN company c ON t.company_id = c.id
106 ORDER BY t.id DESC;
107
108 • SELECT *
109 FROM InformeTecnico
```

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

id	name	surname	iban	company_name
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B	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.
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries

InformeTecnico 7 x Read Only

Output

Action Output

#	Time	Action	Message
✓ 2	23:10:05	CREATE VIEW InformeTecnico AS SELECT t.id, u.name, u.surname, cc.iban, c.company_...	0 row(s) affected
✓ 3	23:10:23	SELECT * FROM InformeTecnico LIMIT 0, 1000	586 row(s) returned