

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 ingreses 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 pin y cvv que sabemos que tendrán siempre la misma longitud, 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 CHAR(4),
8      cvv CHAR(3),
9      expiring_date VARCHAR(10)
10 );
```

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
693	21:09:39	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, iban ...	0 row(s) affected	0.250 sec

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.

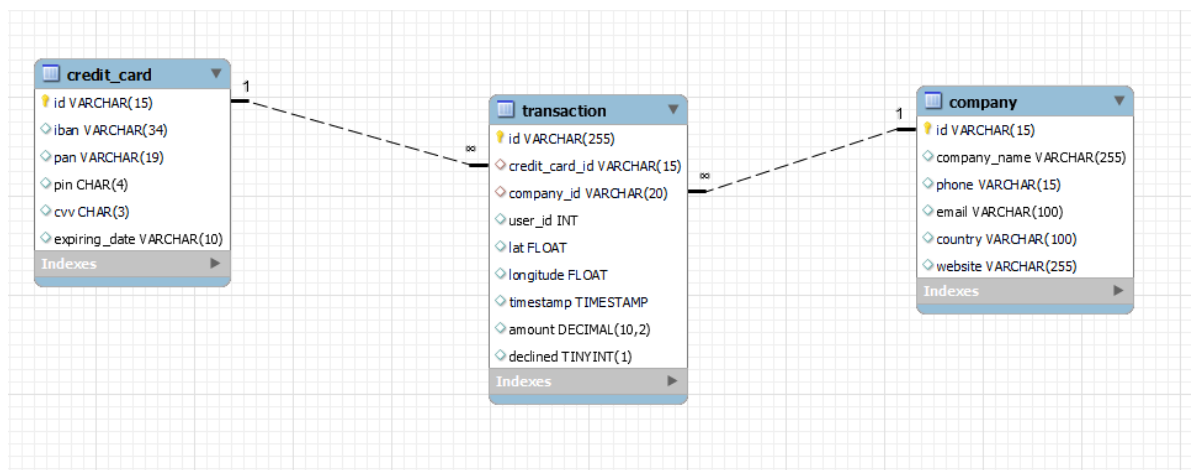
```

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_card;
14

```

id	iban	pan	pin	cvv	expiring_date
CU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22
CU-2945	DO26854763748537475216568689	514242821948828	9080	887	08/24/23
CU-2952	BO45102327105256063255	4556 453 55 5287	4598	438	06/29/21
CU-2959	CR724247724335841535	372461377349375	3583	667	02/24/23
CU-2966	BG724KTQ15627628377363	448566 886747 7265	4900	130	10/29/24
CU-2973	PT8780628135092429456346	544 58654 54343 384	8760	887	01/30/25
CU-2980	DE39241881883086277126	402400 7148845969	5075	596	07/24/22
CU-2987	ES89681434837746781813	3763 747687 76666	2298	797	10/31/23
CU-2994	BH62714428368066765294	344283273252593	7545	595	02/28/22
CU-3001	CY49087426654774581266832110	511722 924833 2244	9562	867	09/16/22
CU-3008	LU507216659816119230	4485744464433884	1856	740	04/05/25
CU-3015	PS151598216295715968342456821	3784 662233 1789	3246	822	01/31/22
CU-3022	GT91695162850556977423121857	5164 1379 4842 3951	5610	342	04/25/25
CU-3029	A262317413982441418123739746	3429 279566 77631	9708	505	09/02/23
CU-3036	A23923602929384085849941994	3768 451556 48766	2232	565	10/27/25
CU-3043	TN6488143310514852179535	455676 6437463635	5969	196	06/07/25
CU-3050	FR1516744369175836831854477	4024007123722	4834	126	10/09/23
CU-3057	LU9318222574697546215	3484 621767 21237	6805	848	09/14/25
CU-3064	PS146853454949253377627273133	3467 732741 26810	3865	498	06/03/25
CU-3071	NO8923814763512	3464 789562 23352	6625	661	12/20/23
CU-3078	IS025127145884623279548733	4539 322 74 2377	9405	720	03/08/23
CU-3085	BE63114729972437	5266 3346 1135 1687	7241	413	05/10/23
CU-3092	RO6385200116612125447487	3488 754223 46253	9417	594	12/19/22
CU-3099	PT2610525356823705537218	448 55418 98863 789	5612	564	01/22/23
CU-3106	AT684251637751136592	349547146395283	9733	209	01/27/24
CU-3113	IE26LCGT47732173572752	341834822877471	9011	287	06/12/21
CU-3120	RS7265576666166237144	527646 533375 6577	7658	265	01/16/21
CU-3127	PT35333461438644342816864	4716 443 46 4368	8038	924	01/16/23
CU-3134	BG23MYJQ52668951824779	5146 3453 9766 2168	7260	935	08/24/25
CU-3141	CH4437804777669672438	3775 626726 45261	2923	330	05/11/24
CU-3148	FR261403224677114	3733 238351 51810	2326	333	09/28/21
CU-3155	AD277720476327722050982	4532263578421	3015	779	01/12/22
CU-3162	HU5607405482623730628233311	455666 645685 4443	5898	603	05/18/20
CU-3169	AT658218806589843788	5396 2876 7721 4764	6102	420	06/23/24
CU-3176	LV84ASV1039958872222	448538 587942 4778	4457	707	03/04/22
CU-3183	GE9015708843338134463	516318 372677 5641	6198	327	10/01/21
CU-3190	CZ810282412635255629363	4532 563 85 3433	4359	849	05/07/22
CU-3197	GT0249763655330848247645975	402 40071 75272 977	6921	562	05/26/21
CU-3204	DK5124134241868111	4716486677272	5714	175	10/30/22
CU-3211	HR5688747222861164805	556665 6465764236	8013	321	04/04/22

credit_card 4 x


#	Time	Action	Message
279	10:30:03	SELECT * FROM credit_card	275 row(s) returned

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 insertar en transaction este nuevo registro, vamos a cargar la tabla user y los datos a introducir de ésta, ya que vamos a tener que insertar los valores correspondientes a id en las tablas credit_card, user y company. Para evitar errores por restricciones en la tabla user, eliminamos la clave foránea 'user_ibfk_1'. Para introducir valores usamos el mismo código que se usa en los scripts para introducir datos. INSERT INTO nombre de la tabla (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).

```

27 #Ejercicio 3
28 • SHOW CREATE TABLE user;
29 • ALTER TABLE user DROP FOREIGN KEY user_ibfk_1;
30
31 • INSERT INTO user (id) VALUES ('9999');
32 • INSERT INTO company(id) VALUES ('b-9999');
33 • INSERT INTO credit_card(id) VALUES('CcU-9999');
34
35 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
36 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);
37
38 # comprobación de registro
39 • SELECT *
40 FROM transaction
41 WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';
42

```

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	2024-11-11 11:11:11	111.11	0

transaction 10 x

Output

#	Time	Action	Message
1264	12:23:58	ALTER TABLE user DROP FOREIGN KEY user_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
1265	12:24:01	INSERT INTO user (id) VALUES (9999)	1 row(s) affected
1266	12:24:03	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES ('108B1D1D-5B23-A76C-55EF-C568E49...	1 row(s) affected
1267	12:33:02	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD'	1 row(s) returned

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;
```



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

Result 6 x

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.

```
50 # NIVEL 2
51 #Ejercicio 1
52 • DELETE FROM transaction
53   WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
54
```



Output

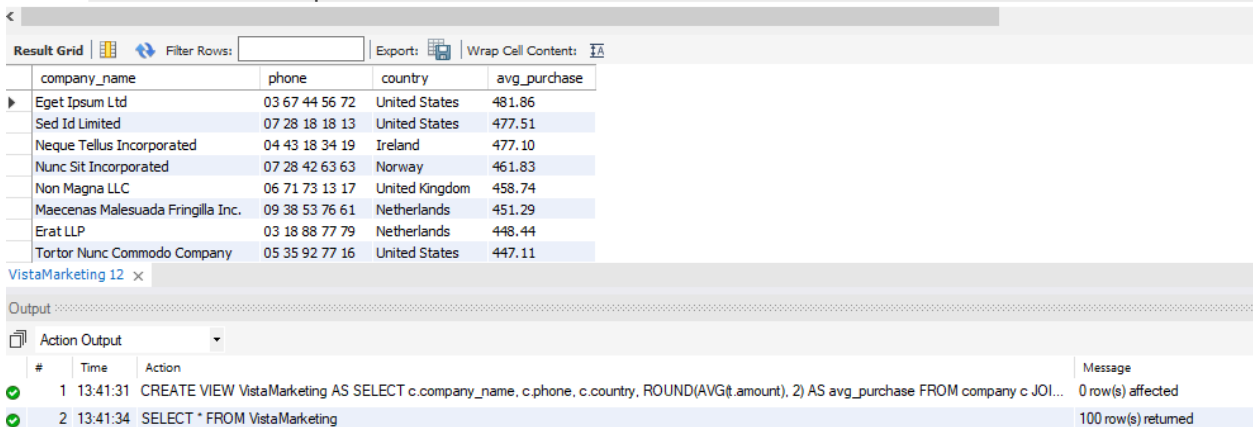
#	Time	Action	Message
1	12:38:40	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02'	1 row(s) affected

Ejercicio 2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesaria que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. 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;
```



The screenshot shows a database interface with a 'Result Grid' and an 'Output' section. The 'Result Grid' displays the results of the SQL queries, showing columns for company_name, phone, country, and avg_purchase. The 'Output' section shows the execution of the queries, with the first query creating the view and the second query selecting all data from the view.

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 Commodo Company	05 35 92 77 16	United States	447.11

VistaMarketing 12 x

Output

Action Output

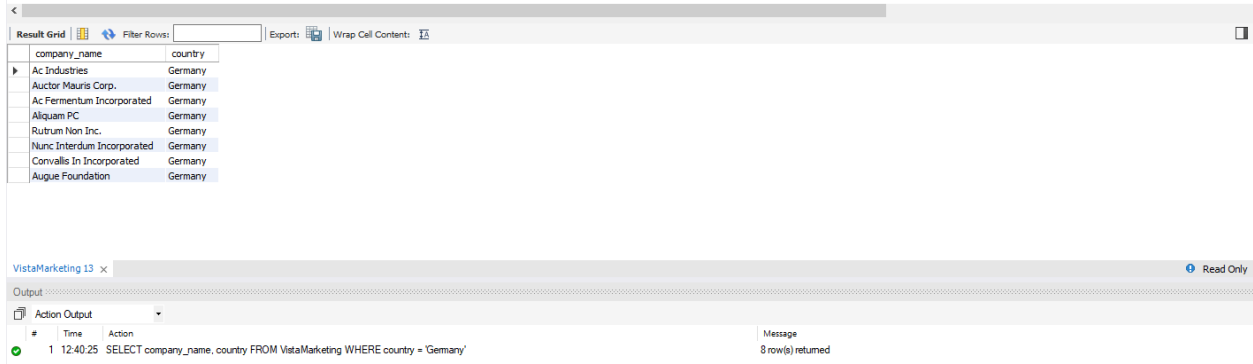
#	Time	Action	Message
1	13:41:31	CREATE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount), 2) AS avg_purchase FROM company c JOIN...	0 row(s) affected
2	13:41:34	SELECT * FROM VistaMarketing	100 row(s) returned

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". Para ello seleccionamos `company_name` y `country` de la vista e incluimos el criterio `WHERE country = 'Germany'`.

```
70 #Ejercicio 3
71 • SELECT company_name, country
72 FROM VistaMarketing
73 WHERE country = 'Germany';
74
```



company_name	country
Ac Industries	Germany
Auctor Mauris Corp.	Germany
Ac Fermentum Incorporated	Germany
Aliquam PC	Germany
Rutrum Non Inc.	Germany
Nunc Interdum Incorporated	Germany
Convallis In Incorporated	Germany
Augue Foundation	Germany

VistaMarketing 13 x Read Only

Output:

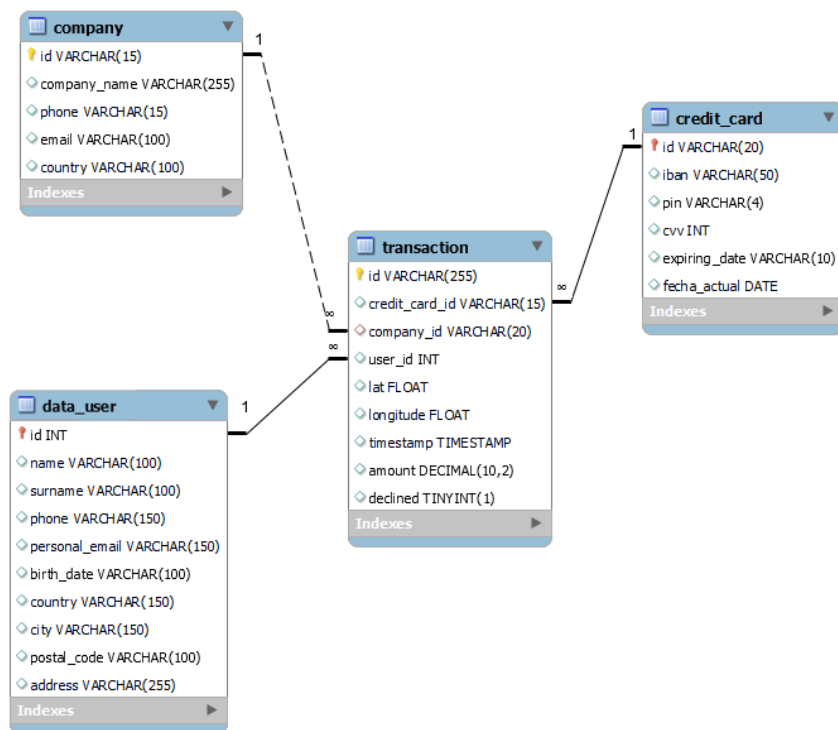
Action Output

#	Time	Action	Message
1	12:40:25	SELECT company_name, country 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".

Los cambios que se perciben son:

- 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.
- La tabla user se ha renombrado a data_user. Usamos la cláusula RENAME TABLE user TO data_user.

- Hay una nueva columna llamada fecha_actual, de modo que usamos ALTER TABLE credit_card y añadimos columna con ADD COLUMN fecha_actual DATE DEFAULT(CURRENT_DATE) para ver como el nombre indica, la fecha actual.
- La columna email de la tabla user se ha renombrado a personal_email. Por ello, usaremos de nuevo la cláusula ALTER TABLE data_user y CHANGE nombre antiguo nombre nuevo y el tipo de dato VARCHAR(150).
- Los siguientes cambios son de tipo de dato en credit_card, usaremos las mismas cláusulas ALTER TABLE credit_card MODIFY COLUMN nombre de la variable y el tipo de dato nuevo.

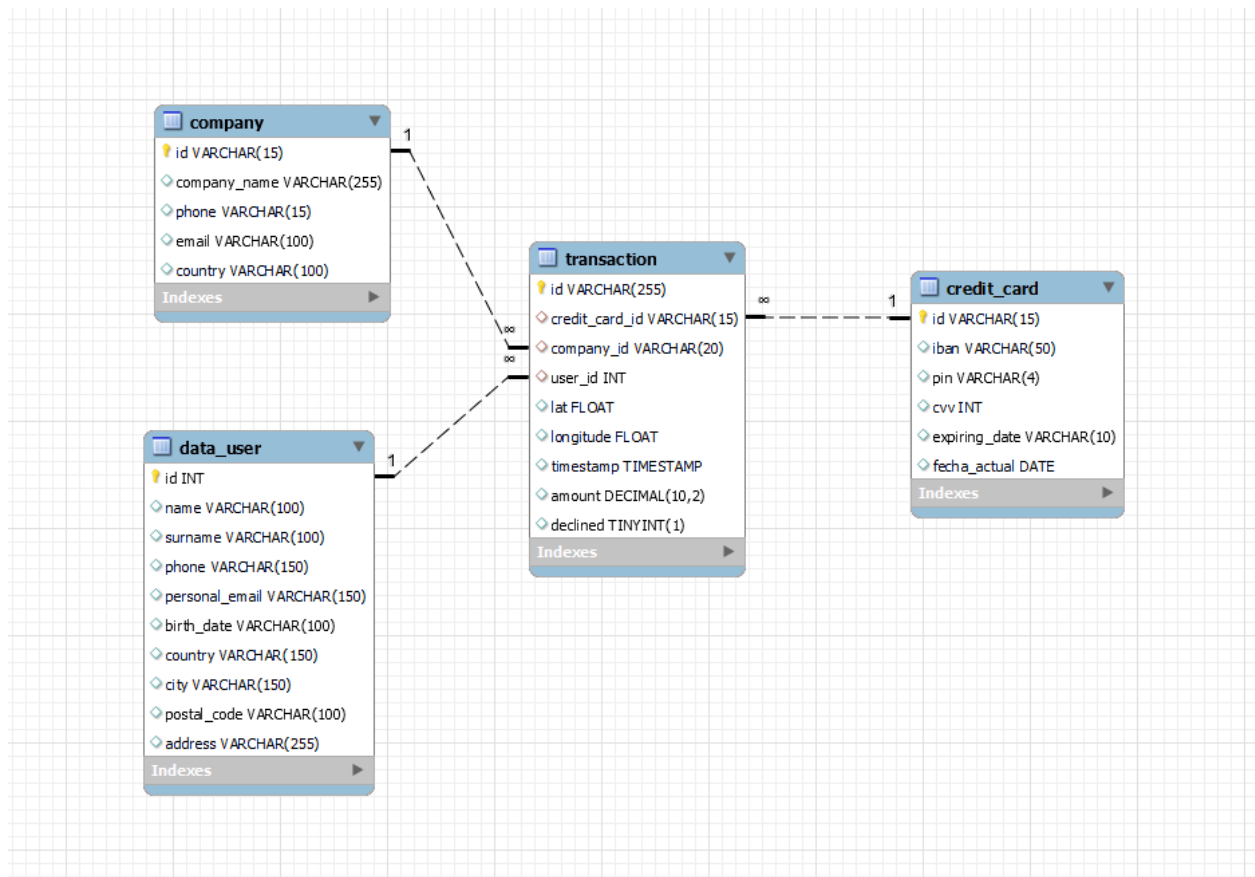
Veamos como queda en MySQL y generamos el diagrama con Database, reverse engineer.

```

76 # Ejercicio 1
77 # Eliminar la columna 'website' de la tabla 'company'
78 • ALTER TABLE company
79 DROP COLUMN website;
80
81 #Cambiar nombre de tabla user a data_user
82 • RENAME TABLE user TO data_user;
83
84 # Agregar la columna 'fecha_actual' a la tabla 'credit_card'
85 • ALTER TABLE credit_card
86 ADD COLUMN fecha_actual DATE DEFAULT(CURRENT_DATE);
87
88 # Renombrar la columna 'email' a 'personal_email' en la tabla 'user'
89 • ALTER TABLE data_user
90 CHANGE email personal_email VARCHAR(150);
91
92 #Cambiar tipo de dato de la variable pin
93 • ALTER TABLE credit_card
94 MODIFY COLUMN pin VARCHAR(4);
95
96 #Cambiar tipo de dato de iban
97 • ALTER TABLE credit_card
98 MODIFY COLUMN iban VARCHAR(50);
99
100 #Cambiar tipo de dato cvv
101 • ALTER TABLE credit_card
102 MODIFY COLUMN cvv INT;

```

Output			
Action Output			
#	Time	Action	Message
5	12:50:56	ALTER TABLE credit_card MODIFY COLUMN pin VARCHAR(4)	276 row(s) affected Records: 276 Duplicates: 0 Warnings: 0
6	12:51:53	ALTER TABLE data_user CHANGE email personal_email VARCHAR(150)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
7	12:52:00	ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(4)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
8	12:52:06	ALTER TABLE credit_card MODIFY COLUMN iban VARCHAR(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
9	12:52:10	ALTER TABLE credit_card MODIFY COLUMN cvv INT	276 row(s) affected Records: 276 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.

Creemos 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 data_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.

```
105 # Ejercicio 2
106 • CREATE VIEW InformeTecnico AS
107 SELECT t.id, u.name, u.surname, cc.iban, c.company_name
108 FROM transaction t
109 JOIN data_user u ON t.user_id = u.id
110 JOIN credit_card cc ON t.credit_card_id = cc.id
111 JOIN company c ON t.company_id = c.id
112 ORDER BY t.id DESC;
113
114 • SELECT *
115 FROM InformeTecnico;
```

Result Grid					
Filter Rows: [] Exports: [] Wrap Cell Content: []					
id	name	surname	iban	company_name	
FE96CE47-8D59-381C-4E18-E3CA3D44E8FF	Keriyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries	
FE809ED4-2D86-55AC-C915-929516E46468	Molly	Gilliam	SE2813123487163628531121	Nunc Interdum Incorporated	
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated	
FD89D51B-AE8D-77DC-E450-88083FBD3187	Hilda	Levy	LT053237077744661475	Malesuada PC	
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.	
FCE2A89A-271D-2BDC-9E49-8DD92A373391	Hakeem	Alford	MD1234119525145401270486	Nunc Interdum Incorporated	
FBD7E0D6-8A68-F9BC-CC49-EA4B8760100C	Hedwig	Gilbert	MJ4133333444534342541344788855	Mauris Id Inc.	
FAC76A80-8448-69AA-E892-426C2F12621C	Slade	Poole	MT053WCF58868200575771634583813	Arcu LLP	
FAAD3FFC-1A17-E141-43D3-359A5BA7CB3B	Hedwig	Gilbert	GE90157928843338134463	Lorem Eu Incorporated	
FA053936-75D8-83FA-490D-98624E1B920A	Hedwig	Gilbert	GT02497653655330848247645975	Non Justo Corp.	
F85A7D75-2778-9D75-D776-3F41A828DE88	Sarah	Beck	VG1468087984174645729577	Ut Semper Foundation	
F843DC08-CCB5-2444-1B4E-5966289FBA88	Jasper	Landry	VG1468087984174645729577	Ut Semper Foundation	
F5ACD74B-4275-SAA1-2414-6EF417636B98	Nora	Reeves	MD1234119525145401270486	Nunc Interdum Incorporated	
F56FCA4A-0039-9F64-7376-85632B91121B	Lynn	Riddle	CR7242477244335841535	Ut Semper Foundation	

InformeTecnico 14 x

Output

Action Output			Message
#	Time	Action	
11	13:29:55	CREATE VIEW InformeTecnico AS SELECT t.id, u.name, u.surname, cc.iban, c.company_name FROM transaction t JOIN data_user u ON t.user_id = ...	0 row(s) affected
12	13:29:58	SELECT * FROM InformeTecnico	587 row(s) returned