Creación de bases de datos S4.01

Elian Daghoum Dorado

Índice

Nivel 1 03

Nivel 2 14

Nivel 3 16

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Empezamos creando la base de datos llamda t4.

En realidad la cree dándole a create schema pero te incluyo el código para el registro.

Creamos la tabla dimension Users.

A la hora de cargar los datos nos encontramos con una dificultad,, debemos averiguar primero en que carpeta podemos dejar los csv. Con esa consulta obtenemos la ruta.

```sql
CREATE DATABASE IF NOT EXISTS t4;
USE t4;
```

```sql
CREATE TABLE IF NOT EXISTS users (
    id CHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date DATE,
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);
```

```sql
SHOW VARIABLES LIKE 'secure_file_priv';
```

Result Grid | Filter Rows: | Export: | Wrap Cel

Variable_name	Value
secure_file_priv	C:\ProgramData\MySQL\MySQL Server 8.0\Upl...

Result 2

Output

Action Output

#	Time	Action
38	12:27:18	SHOW VARIABLES LIKE 'secure_file_priv'

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Vamos a unir las dos bases de datos de usuarios en una misma tabla, para hacer el modelo más eficiente y simplificar las consultas.

Cambiaremos las fechas de string a date para facilitar las consultas.

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(
  id, name, surname, phone, email, @birth_date, country, city, postal_code, address
)
SET birth_date = STR_TO_DATE(@birth_date, '%b %e, %Y');
```

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(
  id, name, surname, phone, email, @birth_date, country, city, postal_code, address
)
SET birth_date = STR_TO_DATE(@birth_date, '%b %e, %Y');
```

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Añadimos a la tabla una columna nueva que identifica el continente de cada usuario.

Para ello editamos al tabla añadiendo la columna nueva. Listamos los paises que tenemos y los usamos como filro para añadir los usuarios a cada región.

Esto se podría haber hecho más fácil a la hora de insertar los datos en el paso anterior, pero me lo han comentado cuando me corrigieron el ejercicio y ya lo hice así

```sql
ALTER TABLE users
ADD COLUMN region ENUM('Europa', 'America') DEFAULT NULL;

SELECT DISTINCT country
FROM users
ORDER BY country;

-- Asignar Europa
UPDATE users
SET region = 'Europa'
WHERE country IN (
    'France', 'Germany', 'Italy',
    'Netherlands', 'Poland', 'Portugal',
    'Spain', 'Sweden', 'United Kingdom')
    AND id IS NOT NULL;

-- Asignar América
UPDATE users
SET region = 'America'
WHERE country IN ('Canada', 'United States')
 AND id IS NOT NULL;
```

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Revisamos que la tabla se haya creado bien

```
18 •  SELECT * FROM t4.users;
19
```

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

id	name	surname	phone	email	birth_date	region	country
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	1985-11-17	America	United State
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	1984-04-30	America	United State
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca	1989-09-15	America	United State
1000	Amkjrv	Qbulrxbp	+48-258-9936	amkjrv.qbulrxbp@example.com	1970-05-17	Europa	Germany
1001	Nfvrlb	Oydaiwbg	+94-121-2522	nfvrlb.oydaiwbg@example.com	1994-03-04	Europa	Germany
1002	Ijbfmd	Jbddzhvp	+70-120-3668	ijbfmd.jbddzhvp@example.com	2001-09-27	Europa	Germany
1003	Uyciig	Sfbdymzj	+58-123-6968	uyciig.sfbdymzj@example.com	1981-01-20	Europa	Germany
1004	Yjqurq	Ojizvgqi	+77-944-2340	yjqurq.ojizvgqi@example.com	1954-07-27	Europa	Germany
1005	Mnlqtu	Glofenwk	+54-801-2627	mnlqtu.glofenwk@example.com	1962-11-15	Europa	Portugal

users 1

Output

Action Output

#	Time	Action	Message
10	10:09:16	SELECT * FROM t4.users	5000 row(s) returned

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Creamos la tabla dimension company y cargamos sus datos.

Revisamos que funciona bien.

```sql
CREATE TABLE IF NOT EXISTS company (
    company_id VARCHAR(15) PRIMARY KEY,
    company_name VARCHAR(255),
    phone VARCHAR(15),
    email VARCHAR(100),
    country VARCHAR(100),
    website VARCHAR(255)
);
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'
INTO TABLE company
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
( company_id, company_name, phone, email, country, website);

SELECT * FROM t4.company;
```

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

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@icloud.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

company 3

Output

Action Output

#	Time	Action	Message
12	10:18:35	SELECT * FROM t4.company	100 row(s) returned

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Creamos la tabla dimension credit card y cargamos sus datos.

Hice una conexión a la tabla users, aunque creo que no hacía falta.

```sql
CREATE TABLE IF NOT EXISTS credit_cards (
        id VARCHAR(20) PRIMARY KEY,
        user_id CHAR(10),
        iban VARCHAR(34) NOT NULL,
        pan VARCHAR(50) NOT NULL,
        pin CHAR(4)NOT NULL,
        cvv CHAR(3)NOT NULL,
        track1 VARCHAR(250) NOT NULL,
        track2 VARCHAR(250) NOT NULL,
        expiring_date DATE,
        FOREIGN KEY (user_id) REFERENCES users(id)
        );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
( id, user_id, iban, pan, pin, cvv, track1, track2, @expiring_date )
SET expiring_date = STR_TO_DATE(@expiring_date, '%m/%d/%y');

SELECT * FROM t4.credit_cards;
```

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

id	user_id	iban	pan	pin	cvv	track1
CcS-4857	276	XX4857591835292505850771	2314242385113924	1819	467	%B2314242385113924^LWC
CcS-4858	277	XX8581768137002436094025	6582720299715533	3964	817	%B6582720299715533^TIQM
CcS-4859	278	XX7826930491423553609370	8861684536289642	4983	277	%B8861684536289642^COFE
CcS-4860	279	XX5559590368835304645299	2481155515498459	6876	661	%B2481155515498459^TIUJ
CcS-4861	280	XX2035182877195191627307	1308930301149557	5710	398	%B1308930301149557^HPOE
CcS-4862	281	XX4774721462463645409758	6715617009807829	4042	174	%B6715617009807829^LDMV

credit_cards 5

Output

Action Output

#	Time	Action	Message
14	10:23:12	SELECT * FROM t4.credit_cards	5000 row(s) returned

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Creamos la tabla dimension products y cargamos sus datos

```sql
CREATE TABLE IF NOT EXISTS products (
        id VARCHAR(20) PRIMARY KEY,
        product_name VARCHAR(200),
        price DECIMAL(10, 2),
        colour VARCHAR(10),
        weight DECIMAL(10, 2),
        warehouse_id VARCHAR(50)
        );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY '""'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, product_name, @price, colour, weight, warehouse_id)
SET price = CAST(SUBSTRING(@price, 2) AS DECIMAL(10,2));

 SELECT * FROM t4.products;
```

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

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	161.11	#7c7c7c	1.00	WH-4
10	Karstark Dorne	119.52	#f4f4f4	2.40	WH--5
100	south duel	40.43	#6d6d6d	3.00	WH--95
11	Karstark Dorne	49.70	#141414	2.70	WH--6
12	duel Direwolf	181.60	#a8a8a8	2.10	WH--7

products 8

Output

Action Output

#	Time	Action	Message
17	10:28:23	SELECT * FROM t4.products	100 row(s) returned

Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

Creamos la tabla de hechos transactions_t4, la conectamos al resto de tablas a través de las id y cargamos sus datos.

A diferencia de las otras tablas, que usaban comas (,) como delimitador, este archivo venía exportado con punto y coma (;)

```sql
CREATE TABLE IF NOT EXISTS transactions_t4 (
    id VARCHAR(255) PRIMARY KEY,
    card_id VARCHAR(15),
    business_id VARCHAR(100),
    timestamp TIMESTAMP,
    amount DECIMAL(10, 2),
    declined BOOLEAN,
    product_ids VARCHAR(255),
    user_id CHAR(10),
    lat FLOAT,
    longitude FLOAT,
    FOREIGN KEY (card_id) REFERENCES credit_cards(id),
    FOREIGN KEY (business_id) REFERENCES company(company_id),
    FOREIGN KEY (user_id) REFERENCES users(id) );

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
INTO TABLE transactions_t4
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, card_id, business_id, timestamp, amount, declined, product_ids, user_id, lat, longitude) ;

SELECT * FROM t4.transactions_t4;
```

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

id	card_id	business_id	timestamp	amount	declined	product_id
00043A49-2949-494B-A5DD-A5BAE3BB19DD	CcS-9294	b-2458	2024-08-28 07:16:46	395.43	0	16, 26, 97,
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	2016-12-21 20:07:18	155.63	0	66, 69, 87
00045D6B-ED2E-4F2F-8186-CEE074D875D0	CcS-6699	b-2390	2020-07-14 15:37:45	326.01	0	30, 11, 16,
000481C3-1C26-4FEF-83A0-4CD0EB004BBD	CcS-6696	b-2230	2017-09-04 19:44:53	161.60	0	72
00051AA4-9CBE-4268-B070-C38062A1B3E2	CcS-7606	b-2266	2017-01-05 18:19:25	148.91	0	18

transactions_t4 10

Output

Action Output

#	Time	Action	Message
20	10:32:36	SELECT * FROM t4.transactions_t4	100000 row(s) returned

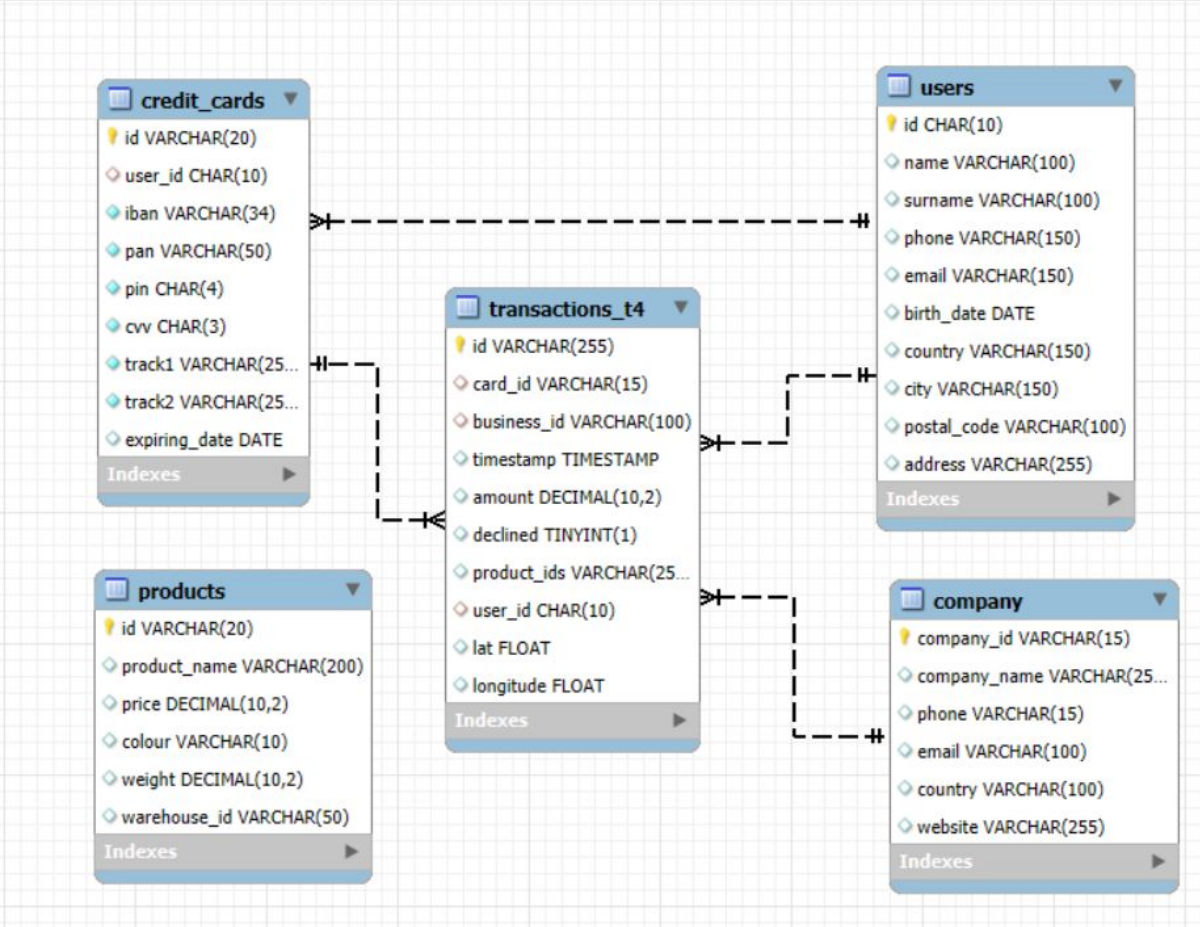
Nivel 1

Creación de tablas

Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas.

El conjunto de tablas, con un esquema de estrella. La tabla transactions_t4 al centro y el resto al rededor.

La tabla products está desconectada de momento.



Nivel 1

Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 80 transacciones utilizando al menos 2 mesas.

Creamos una consulta que busca los id de los que hayan hecho más compras de 80 de la tabla transactions.

Esta la incluimos a modo de subconsulta en una consulta que rescata el nombre y apellido de la tabla users.

```sql
SELECT u.id, u.name, u.surname, recuento.compras_total
FROM (
  SELECT user_id, COUNT(id) AS compras_total
  FROM transactions_t4
  GROUP BY user_id
  HAVING COUNT(id) > 80
  ) AS Recuento
JOIN users u ON Recuento.user_id = u.id;
```

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

id	name	surname	compras_total
185	Molly	Gilliam	110
289	Dxwgi	Hwcru	94
318	Bnyr	Astuw	91
454	Sfzzoh	Xgvfridxs	81

Result 20

Output

Action Output

#	Time	Action	Message
32	12:08:17	SELECT u.id, u.name, u.surname, recuento.compras_total FROM (SELECT ...	4 row(s) returned

Nivel 1

Ejercicio 2

Muestra la media de amount por IBAN de las tarjetas de crédito a la compañía Donec Ltd, utiliza al menos 2 mesas.

Juntamos las tablas transactions, credit cards y company en una consulta para crear la media de gasto. Agrupamos por IBAN y filtramos para que la compañía sea la exigida.

```sql
SELECT ROUND(AVG(t.amount),2) AS 'Media de gasto', cc.iban, c.company_name
FROM transactions_t4 t
JOIN credit_cards cc ON t.card_id = cc.id
JOIN company c ON t.business_id = c.company_id
WHERE c.company_name = 'Donec Ltd'
GROUP BY cc.iban;
```

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

Media de gasto	iban	company_name
356.25	XX9114064011255863075868 05	Donec Ltd
142.96	SK9446370242474562577506	Donec Ltd
257.37	XX7767529178459529755556 40	Donec Ltd
139.59	XX4138273622897193049089 90	Donec Ltd
240.41	XX3477872460707696107803 08	Donec Ltd
[illegible]	[illegible]	[illegible]

Result 22

Output

Action Output

#	Time	Action	Message
34	12:09:50	SELECT ROUND(AVG(t.amount),2) AS 'Media de gasto', cc.iban, c.company_...	371 row(s) returned

⭐⭐

Nivel 2

Creación de la tabla

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas.

```sql
CREATE TABLE Tarjetas_status AS
SELECT card_id AS Tarjeta,
CASE
        WHEN SUM(CASE WHEN declined = 1 THEN 1 ELSE 0 END) = 3 THEN 'inactiva'
        ELSE 'activa' END AS Estado
FROM ( SELECT card_id, declined, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY t.timestamp DESC) AS fila
        FROM transactions_t4 t) AS tabla
WHERE fila <= 3
GROUP BY card_id
;
```

Esta consulta tiene dos puntos interesantes:

Creamos una subconsulta que permite ordenar por tiempo las transacciones dentro de cada tarjeta. Le asigna a cada una un número y gracias a eso podemos seleccionar las 3 primeras.

Además, hay un case que va sumando las veces que cada tarjeta tiene una operación declinada, y cuando llega a 3 la declara como inactiva.

Esta consulta ha sido difícil de hacer y he tenido que seguir muchos tutoriales para hacerla pero creo que he llegado a entenderla.

```sql
15 • SELECT * FROM t4.tarjetas_status;
```

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

Tarjeta	Estado
CcS-4857	activa
CcS-4858	activa
CcS-4859	activa
CcS-4860	activa
CcS-4861	activa
CcS-4862	activa

tarjetas_status 2

Output

Action Output

#	Time	Action	Message
4	11:02:46	SELECT * FROM t4.tarjetas_status	5000 row(s) returned

Nivel 2

Ejercicio 1

¿Cuántas tarjetas están activas?

FIltramos por el estado activa y tenemos la respuesta.

```
45 •  SELECT COUNT(*) AS Tarjetas_activas
46    FROM tarjetas_status
47    WHERE Estado = 'activa';
48
```

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

	Tarjetas_activas
▶	4995

Result 6 ×

Output

Action Output

#	Time	Action	Message
7	11:04:57	SELECT COUNT(*) AS Tarjetas_activas FRO...	1 row(s) returned

Nivel 3

Creación de la tabla

Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes product_id

El principal problema viene de que tenemos varios IDs en un mismo campo y hay que separarlos en distintas filas.

Generamos una tabla auxiliar con los números del 0 al 5 usando UNION SELECT, para acceder a cada posición en la lista de productos.

Con las funciones `SUBSTRING_INDEX`, se extrae cada producto.

La función (LENGTH...) calcula cuántos productos hay dentro del campo para no hacer repeticiones de más.

Finalmente unimos esta tabla al resto.

Este ejercicio ha sido el que más me ha costado. Sabía lo que quería hacer pero no cómo hacerlo. Y cuando lo he comentado con compañeros he visto que tenían soluciones más sencillas que me han gustado más.

```sql
CREATE TABLE transacciones_productos AS
SELECT
    t.id AS transaccion_id,
    TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.n + 1), ',', -1)) AS producto_id
FROM transactions_t4 t
JOIN (
    SELECT 0 AS n UNION SELECT 1 UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5
) n ON n.n < LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1
WHERE
    SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.n + 1), ',', -1) != '';

ALTER TABLE transacciones_productos
ADD CONSTRAINT fk_transaccion
FOREIGN KEY (transaccion_id) REFERENCES transactions_t4(id);

ALTER TABLE transacciones_productos
ADD CONSTRAINT fk_producto
FOREIGN KEY (producto_id) REFERENCES products(id);
```

transaccion_id	producto_id
00043A49-2949-494B-A5DD-A5BAE3BB19DD	87
00043A49-2949-494B-A5DD-A5BAE3BB19DD	97
00043A49-2949-494B-A5DD-A5BAE3BB19DD	26
00043A49-2949-494B-A5DD-A5BAE3BB19DD	16
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	87

transacciones_productos 2

Output

Action Output

#	Time	Action	Message
11	11:18:32	SELECT * FROM t4.transacciones_productos	253391 row(s) returned

Nivel 3

Ejercicio 1

Necesitamos conocer el número de veces que se ha vendido cada producto.

Usamos la tabla recién creada como puente para unir products y transactions_t4. Contamos las id para obtener la cantidad de ventas de cada producto y descartamos las transacciones declined.

```sql
SELECT tp.producto_id, p.product_name, COUNT(transaccion_id) AS ventas_totales
FROM transacciones_productos tp
JOIN products p ON tp.producto_id = p.id
JOIN transactions_t4 t ON tp.transaccion_id = t.id
WHERE declined = 0
GROUP BY tp.producto_id
ORDER BY ventas_totales DESC;
```

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

producto_id	product_name	ventas_totales
52	riverlands the duel	2642
29	Tully maester Tarly	2627
21	duel Direwolf	2603
16	the duel warden	2602
33	duel warden	2593
87	sith Jade	2591

Result 2

Output

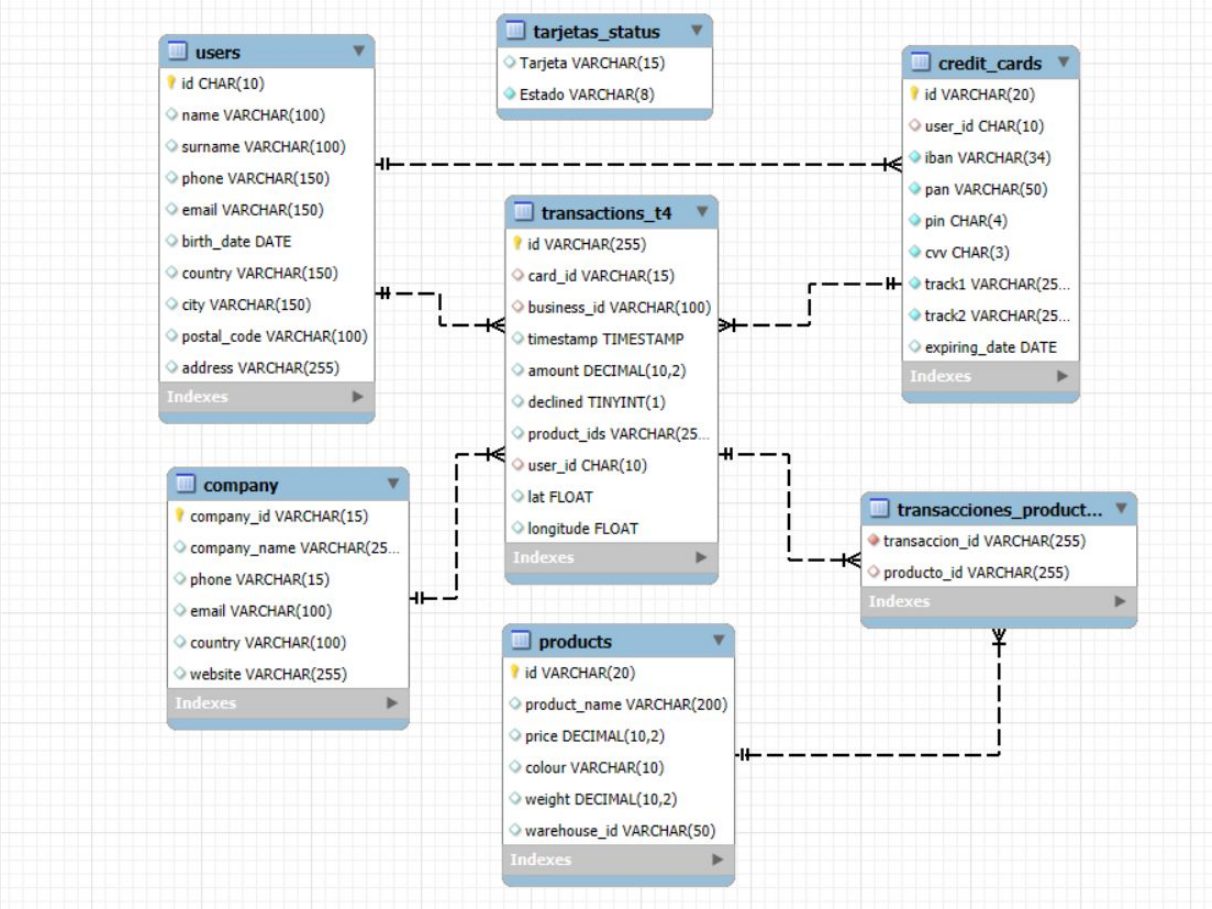
Action Output

#	Time	Action	Message
8	16:57:36	SELECT tp.producto_id, p.product_name, COUNT(transaccion_id) AS ventas...	100 row(s) returned

⭐⭐⭐

Nivel 3

Esquema final de tablas



Gracias.