SPRINT 4

Nivel 1 ----------

Descarga los archivos CSV, estudiales y diseña una base de datos con un esquema de **estrella** que contenga, al menos **4 tablas** de las que puedas realizar las siguientes consultas:

Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.

Ejercicio 2

Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.

Creación de la BBDD

```sql
-- 1. CREAR BBDD

CREATE DATABASE SPRINT4;

-- 2. CREAR TABLAS

CREATE TABLE companies(
    company_id VARCHAR(20),
    company_name VARCHAR(100) NOT NULL,
    phone VARCHAR(30),
    email VARCHAR(100) UNIQUE,
    country VARCHAR(50),
    website VARCHAR(150),
    PRIMARY KEY (company_id)
);

CREATE TABLE credit_cards (
    id VARCHAR(20),
    user_id INT NOT NULL,
    iban VARCHAR(34) NOT NULL UNIQUE, -- IBAN, formato estándar con longitud máxima de 34 caracteres
    pan VARCHAR(30) NOT NULL UNIQUE, -- Número de tarjeta, longitud fija de 16 caracteres - pero hay datos con espacios - mas largos en el csv
    pin CHAR(4) NOT NULL, -- PIN credit_cards de 4 dígitos
    cvv CHAR(3) NOT NULL, -- CVV de 3 dígitos
    track1 VARCHAR(80) NOT NULL,
    track2 VARCHAR(40) NOT NULL,
    expiring_date  VARCHAR(10),            -- ver luego de la carga de transformar en DATE en formato YYYY-MM-DD
    PRIMARY KEY (id),
    CONSTRAINT chk_pin CHECK (LENGTH(pin) = 4), -- Restricción para asegurar que el PIN tenga 4 caracteres
    CONSTRAINT chk_cvv CHECK (LENGTH(cvv) = 3) -- Restricción para asegurar que el CVV tenga 3 caracteres
);

CREATE TABLE products (
    id INT,
    product_name VARCHAR(255),
    price VARCHAR(20)  -- tiene simbolo $ en el archivo a cargar - se puede modificar luego
    colour CHAR(7) CHECK (colour REGEXP '^#[0-9a-fA-F]{6}$'), -- restricción CHECK que valida que el campo colour contenga un código de color hexadecimal correcto.
    weight INT CHECK (weight > 0), -- Peso en unidades enteras, debe ser positivo
    warehouse_id VARCHAR(10),
    PRIMARY KEY (id)
    );
```

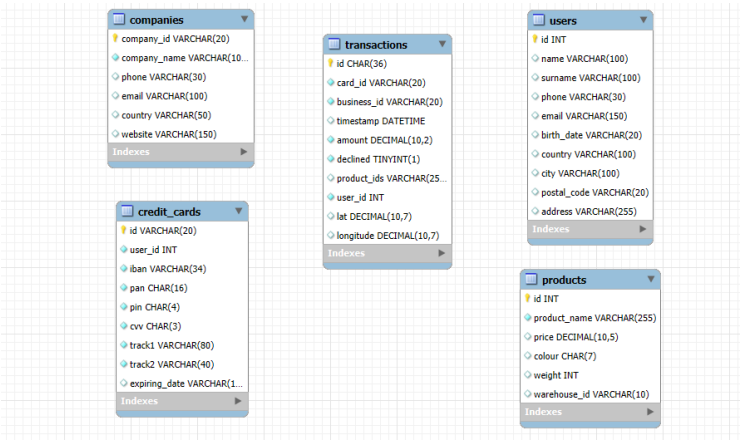
```sql
CREATE TABLE transactions (
    id CHAR(36),
    card_id VARCHAR(20) NOT NULL,
    business_id VARCHAR(20) NOT NULL,
    timestamp DATETIME,
    amount DECIMAL(10,2) NOT NULL CHECK (amount > 0), -- Monto de la transacción, debe ser positivo
    declined BOOLEAN NOT NULL, -- Indica si la transacción fue rechazada (0 = aprobada, 1 = rechazada)
    product_ids VARCHAR(255), -- Lista de productos por  transaccion
    user_id INT NOT NULL,
    lat DECIMAL(10,7) CHECK (lat BETWEEN -90 AND 90), -- Latitud con precisión de 7 decimales
    longitude DECIMAL(10,7) CHECK (longitude BETWEEN -180 AND 180), -- Longitud con precisión de 7 decimales
    PRIMARY KEY (id)
    );


-- CREAR UNA TABLA UNIFICADA A PARTIR DE LAS 3 TABLAS DE USERS DE CANADA, UK Y US YA QUE TODAS TIENEN MISMOS CAMPOS Y ESTRUCTURA

CREATE TABLE users (
    id INT,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(30),
    email VARCHAR(150) UNIQUE,
    birth_date VARCHAR(20), -- transformar después de la carga a formato date
    country VARCHAR(100),
    city VARCHAR(100),
    postal_code VARCHAR(20),
    address VARCHAR(255),
    PRIMARY KEY (id)
);

-- CARGAR LOS CSV E INSERTARLOS EN LAS TABLAS RESPECTIVAS

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv"
INTO TABLE companies


  -- CARGAR LOS CSV E INSERTARLOS EN LAS TABLAS RESPECTIVAS

  LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv"
  INTO TABLE companies
  FIELDS TERMINATED BY ','
  ENCLOSED BY '"'
  LINES TERMINATED BY '\n'
  IGNORE 1 ROWS
  (company_id,company_name,phone,email,country,website);

  LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\credit_cards.csv"
  INTO TABLE credit_cards
  FIELDS TERMINATED 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');  -- en la CARGA SE TRANSFORMA el tipo de dato de expiring_date a DATE (YYY-MM-DD)


101
102 •  LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv"
103    INTO TABLE products
104    FIELDS TERMINATED BY ','
105    LINES TERMINATED BY '\n'
106    IGNORE 1 ROWS
107    (id,product_name,price,colour,weight,warehouse_id);
108
109    -- Transformaciones en la tablas/columnas
110
111    -- Quitar el signo $ de price
112
113 •  UPDATE products
114    SET price = REPLACE(price, '$', '');
115
116    -- Modificar en la TABLA el tipo de dato de esta columna
117 •  ALTER TABLE products MODIFY price DECIMAL(10,2);
118


  -- Tabla transactions

  -- Aparece un Warning de datos truncados en las cols lat y longitud
  -- Modificar en la tabla creada ambas cols para que coincidan con el .csv

•  ALTER TABLE transactions MODIFY lat DECIMAL(14,10);  -- DECIMAL(14,10) es un formato común para representar las coordenadas de latitud y longitud en bases de datos
•  ALTER TABLE transactions MODIFY longitude DECIMAL(14,10);

  -- truncate table - Borrar todos los datos y volver a cargar

•  TRUNCATE TABLE transactions;

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

```sql
-- Tabla users

-- Los 3 archivos de usuarios .csv tienen la misma estructura, por lo tanto se pueden ir cargando una a una a la tabla users.
-- NO repiten los id.

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_usa.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(id,name,surname,phone,email,birth_date,country,city,postal_code,address);

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_uk.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(id,name,surname,phone,email,birth_date,country,city,postal_code,address);
```

```sql
-- al repetirse un mail de este archivo que ya está en la carga desde el archivo anterior, se genera el error code 1062.
-- Tomo la opcion de eliminar de la tabla users la restriccion impuesta inicialmente de UNIQUE en el campo email

ALTER TABLE users DROP INDEX  email;

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_uk.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(id,name,surname,phone,email,birth_date,country,city,postal_code,address);

LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_ca.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS
(id,name,surname,phone,email,birth_date,country,city,postal_code,address);
```

```sql
-- Chequeo la tabla completa- Son 275 registros total

SELECT * FROM users;
```

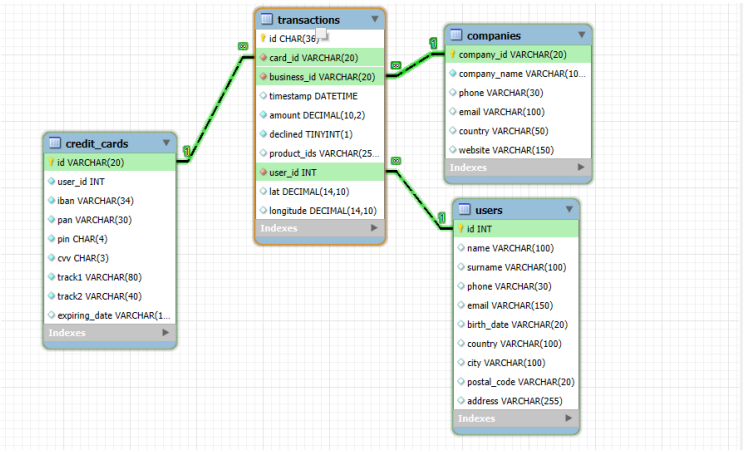
Tablas creadas y cargadas



Crear las relaciones entre las tablas

```sql
-- Agregar las relaciones entre tablas

ALTER TABLE transactions
ADD FOREIGN KEY (business_id) REFERENCES companies (company_id),
ADD FOREIGN KEY (card_id) REFERENCES credit_cards (id),
ADD FOREIGN KEY (user_id) REFERENCES users (id);
```

Las 4 tablas relacionadas



Ejercicio 1

```sql
-- Ejercicio 1

-- Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.

SELECT u.id, u.name, u.surname
FROM users u
WHERE u.id IN (SELECT t.user_id
               FROM transactions t
               GROUP BY t.user_id
               HAVING COUNT(t.id) > 30)
ORDER BY u.id;
```

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

id	name	surname
92	Lynn	Riddle
267	Ocean	Nelson
272	Hedwig	Gilbert
275	Kenyon	Hartman

Ejercicio 2

```sql
-- Ejercicio 2
-- Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.

-- Buscar el nombre correcto de la compañia
SELECT * FROM companies WHERE company_name LIKE '%Donec%';

SELECT cc.iban, ROUND(AVG(t.amount),2)
FROM transactions t INNER JOIN companies c ON  t.business_id = c.company_id
INNER JOIN credit_cards cc ON t.card_id = cc.id
WHERE c.company_name = 'Donec Ltd'
GROUP BY cc.iban;
```

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

iban	ROUND(AVG(t.amount),2)
PT87806228135092429456346	203.72

Result 67

Output

Action Output

#	Time	Action	Message
140	18:00:49	SELECT u.id, u.name, u.surname FROM users u WHERE u.id IN (SELECT t.user_id FROM tra...	4 row(s) returned
141	18:03:33	SELECT cc.iban, ROUND(AVG(t.amount),2) FROM transactions t INNER JOIN companies c ON t.business_id = c...	1 row(s) returned

Nivel 2 ----------

```sql
-- Nivel 2 ----------

/* Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones
 fueron declinadas y genera la siguiente consulta:

Ejercicio 1
¿Cuántas tarjetas están activas? */

-- Ordenar una query con funcion ventana ROW_NUMBER() PARTICIONADA POR IBAN Y ORDENADA POR TIMESTAMP DESC (primero las ultimas)
-- Filtrar por hasta 3 ROWS de cada particion por IBAN
-- Clasificar las ultimas 3 transacciones por IBAN en Activas o Inactivas segun esas ultimas 3 por tarjeta hayan sido todas rechazadas (suma= 3) o no.
-- Contar cuantas clasifican como Activas
```

Ejercicio 1

```sql
SELECT COUNT(*)  AS Cantidad_Activas
FROM( SELECT s3.iban, SUM(s3.estado) AS total_declined,
        CASE WHEN SUM(s3.estado) = 3 THEN 'inactiva' ELSE 'activa' END AS clasification
FROM ( SELECT *
FROM ( SELECT ROW_NUMBER() OVER (PARTITION BY cc.iban ORDER BY s1.fecha DESC) AS row_n, s1.iban, s1.estado
FROM ( SELECT cc.iban AS iban, t.declined AS estado, t.timestamp AS fecha
        FROM transactions t
        JOIN credit_cards cc ON cc.id = t.card_id
        ORDER BY cc.iban, t.timestamp DESC ) AS s1) s2
WHERE s2.row_n <= 3) s3
GROUP BY s3.iban) s4
WHERE clasification = 'activa';

-- Bajo este criterio de clasificación todas las tarjetas estan activas
```

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

iban	ROUND(AVG(t.amount),2)
PT87806228135092429456346	203.72

```sql
-- Crear Tabla nueva

CREATE TABLE credit_card_status (
    iban VARCHAR(34) PRIMARY KEY,
    classification ENUM('activa', 'inactiva'));

```

```sql
-- Insertar los datos de la query

INSERT INTO credit_card_status (iban, classification)
SELECT s3.iban,        CASE WHEN SUM(s3.estado) = 3 THEN 'inactiva' ELSE 'activa' END AS classification
FROM ( SELECT *
FROM ( SELECT ROW_NUMBER() OVER (PARTITION BY cc.iban ORDER BY s1.fecha DESC) AS row_n, s1.iban, s1.estado
FROM ( SELECT cc.iban AS iban, t.declined AS estado, t.timestamp AS fecha
        FROM transactions t
        JOIN credit_cards cc ON cc.id = t.card_id
        ORDER BY cc.iban, t.timestamp DESC ) AS s1) s2
WHERE s2.row_n <= 3) s3
GROUP BY s3.iban;
```

```sql
-- Chequeo tabla -- ************************
select * from credit_card_status;

-- Nivel 3 ------------------------------------------------------------------------

/* 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_ids.*/
```

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

iban	classification
AD2777204763277722050982	activa
AD5031514560453613215340	activa
AD7964354272148656432616	activa
AE2251854054380117697777	activa
AE4918271423028872663 69	activa

credit_card_status 68

-- Nivel 3 ----------

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_ids.

-- Ejercicio 1

```sql
-- Nivel 3 ------------------------------------------------------------------------
/* 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_ids.*/

-- Ejercicio 1
-- Necesitamos conocer el número de veces que se ha vendido cada producto.

-- situacion inicial col product_ids

SELECT id, product_ids
FROM transactions;
```

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

id	product_ids
02C6201E-D90A-1859-B4EE-88D2986D3B02	71, 1, 19
0466A42E-47CF-8D24-FD01-C0B689713128	47, 97, 43
063FBA79-99EC-66FB-29F7-25726D1764A5	47, 67, 31, 5
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89, 83, 79
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43, 31

transactions 49 | Apply

Output

Action Output

#	Time	Action	Message	Duration / Fe
166	17:10:04	select * from credit_card_status LIMIT 0, 1000	275 row(s) returned	0.000 sec / (
167	17:25:36	SELECT id, product_ids FROM transactions LIMIT 0, 1000	587 row(s) returned	0.000 sec / (

```sql
-- Transformar la columna product_ids
-- Crear una tabla puente o intermedia: transactions-products para poder unir por su intermedio
-- las tablas originales transactions y products.

-- Chequear de tipos de datos de col product_ids para reconvertir
SHOW COLUMNS FROM transactions WHERE Field = 'product_ids';  -- es varchar(255)
```

```sql
-- Extraer valores de cada product_ids de la tabla transactions y transformarlo en filas cada una con un solo id de producto

SELECT id AS transaction_id, SUBSTRING_INDEX(product_ids, ',', 1) AS product_id
FROM transactions WHERE product_ids IS NOT NULL
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%'
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 3), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%,%'
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 4), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%,%,%';
```

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

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
0466A42E-47CF-8D24-FD01-C0B689713128	47
063FBA79-99EC-66FB-29F7-25726D1764A5	47
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43

```sql
-- Crear la tabla puente transaction_products

CREATE TABLE transaction_products (
transaction_id CHAR(36) NOT NULL,
product_id INT NOT NULL,
FOREIGN KEY (transaction_id) REFERENCES transactions(id),
FOREIGN KEY (product_id) REFERENCES products(id),
PRIMARY KEY (transaction_id, product_id)) ;            -- Evita duplicados
```

```sql
SELECT transaction_id, product_id
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transaction_products_1.csv'
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
FROM ( SELECT id AS transaction_id, SUBSTRING_INDEX(product_ids, ',', 1) AS product_id
FROM transactions WHERE product_ids IS NOT NULL
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%'
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 3), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%,%'
UNION ALL
SELECT id, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 4), ',', -1) AS product_id
FROM transactions WHERE product_ids LIKE '%,%,%,%'
) AS transct_prod;

 -- Cargar a la tabla transaction_products desde el archivo .csv guardado- sin INSERT INTO.

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transaction_products_1.csv'
INTO TABLE transaction_products
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(transaction_id, product_id);

-- Chequear la carga
select * from transaction_products;
```

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

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
122DC333-E19F-D629-DCD8-9C54CF1EBB9A	1
1753A288-9FC1-52E6-5C39-A1FFB97B0D3A	1
1A6CECFB-2E3A-65A3-72D9-2FDB58A1E4BA	1
1EA2B262-D507-AD14-4374-4D532967113F	1

transaction_products 70

```sql
-- Ejercicio 1
-- Necesitamos conocer el número de veces que se ha vendido cada producto.

SELECT tp.product_id, p.product_name , COUNT(*) AS cant_ventas_producto
FROM transaction_products tp
JOIN products p ON tp.product_id = p.id
JOIN transactions t ON t.id = tp.transaction_id
WHERE declined = 0
GROUP BY tp.product_id, p.product_name
ORDER BY cant_ventas_producto DESC;
```

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

product_id	product_name	cant_ventas_producto
23	riverlands north	60
67	Winterfell	59
2	Tarly Stark	56
43	duel	54
17	skywalker ewok sith	54
97	jinn Winterfell	53
79	Direwolf riverlands the	52

Result 72

Output

Action Output

#	Time	Action	Message	Duration /
145	18:19:36	SELECT tp.product_id, p.product_name , COUNT(tp.transaction_id) AS cant_ventas_producto FROM transaction...	26 row(s) returned	0.015 sec

Modelo BBDD final