SPRINT 4
MODELAT SQL
TASCA S4.01. CREACIÓ DE BASE DE DADES
HAILE JACOBO MENESES MORENO

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules.

TAREAS REALIZADAS

Crear la Base de Datos

```
CREATE DATABASE transacciones;
```

Crear las Tablas

TABLA COMPANY

```
CREATE TABLE company (
        id VARCHAR(10) UNIQUE,
        company_name VARCHAR(255),
        phone VARCHAR(25),
        email VARCHAR(100),
        country VARCHAR(100),
        website VARCHAR(100),
        PRIMARY KEY (id)
    );
```

Incidencia: Al momento de crear la tabla no permitía la creación de la misma declarando la Foreign Key, solo era posible agregándolas posteriormente mediante, primero indexando company_id en transactions y declarando la Foreign Key.

```
/*Se declara un índice en transactions para poder agregar la FK company_id*/
CREATE INDEX idx_company
ON transactions (company_id);


/* Se declara la FK en company*/
ALTER TABLE company
        ADD FOREIGN KEY (id) REFERENCES transactions(company_id);
```

TABLA CREDIT_CARD

```
CREATE TABLE credit_card (
        id VARCHAR(15),
        user_id VARCHAR(50),
        iban VARCHAR(50),
        pan VARCHAR(50),
        pin VARCHAR(4),
        cvv INT,
        track1 VARCHAR(255),
        track2 VARCHAR(255),
        expiring_date VARCHAR(10),
        PRIMARY KEY (id)
        );
```

Incidencia: La misma que la anterior, se soluciona con el siguiente procedimiento:

```
/*Se declara un índice en transactions para poder agregar la FK company_id*/
CREATE INDEX idx_creditcard
ON transactions (credit_card_id);

 /* Se declara la FK en credit_card*/
ALTER TABLE credit_card
        ADD FOREIGN KEY (id) REFERENCES transactions(credit_card_id);
```

TABLA USERS

```
CREATE TABLE users (
        id int,
        name VARCHAR(100),
        surname VARCHAR(100),
        phone VARCHAR(150),
        email VARCHAR(150),
        birth_day VARCHAR(50),
        country VARCHAR(100),
        city VARCHAR(150),
        postal_code VARCHAR(50),
        address VARCHAR(150),
        PRIMARY KEY(id)
  );
```

Incidencia: La misma que la anterior, se soluciona con el siguiente procedimiento:

```
/*Se declara un índice en transactions para poder agregar la FK users_id*/
CREATE INDEX idx_users
ON transactions (users_id);

 /* Se declara la FK en users*/
ALTER TABLE users
        ADD FOREIGN KEY(id) REFERENCES transactions(users_id);
```

TABLA PRODUCTS

```
CREATE TABLE products (
      id VARCHAR(255),
      product_name VARCHAR(255),
      price VARCHAR(25),
      colour VARCHAR(25),
      weight VARCHAR(10),
      warehouse_id VARCHAR(10),
      PRIMARY KEY(id)
      );
```

Incidencia: La misma que la anterior, se soluciona con el siguiente procedimiento:

```
/*Se declara un índice en transactions para poder agregar la FK product_ids*/
CREATE INDEX idx_products
ON transactions (product_ids);

 /* Se declara la FK en products*/
ALTER TABLE products
        ADD FOREIGN KEY(id) REFERENCES transactions(product_ids);
```

TABLA TRANSACTIONS

```
CREATE TABLE transactions (
        id VARCHAR(255),
        credit_card_id VARCHAR(15),
        company_id VARCHAR(10),
        timestamp timestamp,
        amount decimal(10,2),
        declined tinyint,
        product_ids VARCHAR(255),
        user_id int,
        latitude float,
        longitude float,
        PRIMARY KEY(id)
   );
```

Nivel 1. Exercici 1. “Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.”

Ejecución de la solicitud realizada mediante la siguiente consulta(utilizando subconsulta y una JOIN implícita):

```
SELECT users.id AS Usuario_a,
CONCAT(users.name, ' ', users.surname) AS Nombre,
(SELECT COUNT(*)
 FROM transactions
 WHERE transactions.users_id = users.id) AS Transacciones
FROM users
WHERE users.id IN (
            SELECT users_id
            FROM transactions
           GROUP BY users_id
           HAVING COUNT(id) > 30
)
ORDER BY Transacciones DESC;
```

Datos presentados en la consulta:

Usuario_a	Nombre	Transacciones
272	Hedwig Gilbert	76
267	Ocean Nelson	52
275	Kenyon Hartman	48
92	Lynn Riddle	39

Sobre esta query he de comentar que mediante la utilización de una subconsulta el costo de la query es casi el doble que utilizando JOIN (revisado en MySQL Workbench 303.75 vs 186.99) doy por presentado el primero pero aclaro que el siguiente procedimiento obtiene el mismo resultado y con menor coste:

```
SELECT users.id AS Usuario_a,
CONCAT(users.name," ", .users.surname) AS Nombre,
COUNT(transactions.id) AS Transacciones
FROM transactions
JOIN users ON transactions.users_id = users.id
GROUP BY usuario_a
HAVING COUNT(transactions.id) > 30
ORDER BY transacciones DESC
;
```

Nivel 1. Exercici 2. "Mostra la mitjana de la suma de transaccions per IBAN de les targetes de crèdit en la companyia Donec Ltd. utilitzant almenys 2 taules."

Ejecución de la solicitud realizada mediante la siguiente consulta:

```
SELECT company_name AS Compañia, credit_card.iban AS IBAN,
ROUND(AVG(transactions.amount), 2) AS Mediana_Transacciones
FROM transactions
JOIN company ON company.id = transactions.company_id
JOIN credit_card ON transactions.credit_card_id = credit_card.id
WHERE company_name = "Donec Ltd"
GROUP BY credit_card.iban;
```

Datos presentados en la consulta:

Compañia	IBAN	Mediana_Transacciones
Donec Ltd	PT87806228135092429456346	203.72

Nivell 2

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades i genera la següent consulta:

Exercici 1
Quantes targetes estan actives?

PRIMER PASO: Creo una consulta de prueba que me permita ordenar por una parte los credit_card_id y sus operaciones de más reciente a antigua por timestamp, utilizando la función de ventana row_number, diviendo en particiones por el credit_card_id, ordenándolos por timestamp de manera descendente y con esto poder realizar el siguiente paso.

```
select credit_card_id, timestamp, declined,
row_number() over (partition by credit_card_id order by timestamp DESC) as operacion
from transactions
order by credit_card_id ASC, timestamp DESC
;
```

Datos presentados en la consulta:

credit_card_id	timestamp	declined	operacion
CcU-2938	2022-03-12 09:23:10	0	1
CcU-2938	2022-03-09 20:53:59	0	2
CcU-2938	2022-02-24 11:01:42	0	3
CcU-2938	2021-10-24 01:29:53	0	4
CcU-2938	2021-10-17 03:52:48	0	5
CcU-2938	2021-09-28 02:24:34	0	6
CcU-2938	2021-09-24 08:33:44	0	7
CcU-2938	2021-09-18 00:31:49	0	8
CcU-2938	2021-09-15 05:23:32	0	9
CcU-2938	2021-08-28 23:42:24	0	10
CcU-2938	2021-07-27 17:14:52	0	11
CcU-2938	2021-07-25 12:34:59	0	12
CcU-2938	2021-07-18 08:20:59	0	13
CcU-2938	2021-07-11 00:19:27	0	14
CcU-2938	2021-07-07 17:43:16	0	15
CcU-2938	2021-07-03 19:56:27	0	16
CcU-2938	2021-05-28 15:21:36	0	17
CcU-2938	2021-05-19 01:05:28	0	18
CcU-2938	2021-05-09 10:25:08	1	19

El resultado permite ver que el orden es el necesario para poder realizar la siguiente parte de la tarea solicitada.

Se crea la Tabla Tarjetas_Activas con una CTE mediante WITH con la consulta anterior nombrándola Operaciones y posteriormente creo otra CTE llamada ConteoDeclinadas donde se selecciona el credit_card_id y se cuentan el número de transacciones declinadas teniendo en cuenta las 3 transacciones más recientes indicadas en Operaciones.

Posteriormente se realiza una consulta tomando los resultados de ConteoDeclinadas y con otra declaración CASE se determina el estado de la tarjeta, considerando que si la suma es 3 el estado será "Tarjeta Inactiva", de lo contrario "Tarjeta Activa"

```
CREATE TABLE tarjetas_activas
WITH Operaciones AS (
        SELECT credit_card_id, declined,
        ROW_NUMBER() OVER (PARTITION BY credit_card_id ORDER BY timestamp DESC) as
        operacion
        FROM transactions
),
ConteoDeclinadas AS (
        SELECT credit_card_id,
        SUM(CASE WHEN declined = 1 THEN 1 ELSE 0 END) AS No_Aceptada_1
        FROM Operaciones
        WHERE operacion <= 3
        GROUP BY credit_card_id
)
SELECT credit_card_id AS Número_Tarjeta,
        CASE WHEN No_Aceptada_1 = 3 THEN "Tarjeta Inactiva"
        ELSE "Tarjeta Activa"
        END AS Estado_Tarjeta
FROM ConteoDeclinadas
```

;

Datos Presentados

Número_Tarjeta	Estado_Tarjeta
CcU-2938	Tarjeta Activa
CcU-2945	Tarjeta Activa
CcU-2952	Tarjeta Activa
CcU-2959	Tarjeta Activa
CcU-2966	Tarjeta Activa
CcU-2973	Tarjeta Activa
CcU-2980	Tarjeta Activa
CcU-2987	Tarjeta Activa
CcU-2994	Tarjeta Activa
CcU-3001	Tarjeta Activa
CcU-3008	Tarjeta Activa
CcU-3015	Tarjeta Activa
CcU-3022	Tarjeta Acti

Tarjet

Se muestra una porción de todas las tarjetas que aparecen en el resultado, que son 275, coincidiendo con el total de tarjetas de la tabla credit_card y transactions (todas tienen al menos 1 operación).

Finalmente se realiza la consulta indicada en el ejercicio, mediante:

```
SELECT COUNT(*) AS Cantidad_Tarjetas_Activas
FROM tarjetas_activas
WHERE Estado_Tarjeta = "Tarjeta Activa"
;
```

Cantidad_Tarjetas_Activas
275

NIVEL 3: Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

Exercici 1: Necessitem conèixer el nombre de vegades que s'ha venut cada producte.

Consideración previa.

He realizado dos formas de realizar esta tarea: Por un lado, he realizado una tabla para poder hacer la relación entre products y transactions, con el fin de poder tener un listado donde separar los id contenidos dentro del campo product_ids en la tabla transactions y poder realizar lo indicado en el ejercicio 1.

Realizando pruebas de métodos para poder hacer la extracción de los datos contenidos me encontré con que no realizaba conteos o separaciones completas de los id aninados (mediante row_number, por ejemplo), revisando la columna observé que dentro de cada fila existían espacios antes y después de las comas, detalle que me hizo pensar en la posibilidad de eliminarlos para poder hacer la consulta de manera más efectiva, realicé el siguiente procedimiento:

1. Para poder realizar la modificación de la tabla

   ```
   SET SQL_SAFE_UPDATES=0;
   ```

2. La actualización de la tabla eliminando los espacios en blanco antes y después y además del espacio antes y después de las comas.

   ```
   UPDATE transactions
   SET product_ids = REPLACE(TRIM(product_ids), ' ', '');
   ```

PRIMER MÉTODO. CREANDO UNA TABLA.

Mediante el siguiente código creo la tabla, cuyo objetivo es poder separar todos los id contenidos en el campo product_ids, segregados por el id de la operación repetido tantas veces como id estén contenidos en cada registro.

```
SELECT t.id, SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.n), ',', -1) as product_id
FROM (
        SELECT 1 as n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
   ) n
JOIN transactions t
ON n.n <= 1 + (LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')))
ON n.n <= 1 + (LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')))
ORDER BY t.id, n.n;
```

Explicación de cada porción del código.

```
(
 SELECT 1 as n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4
   ) n
```

La subconsulta con alias “n” genera una serie de números utilizando el operador UNION ALL que representan las posiciones de la lista que están separadas por las comas y serán tantas uniones como id se contengan (revisando y probando se han visto que son necesarias hasta un máximo de 4)

```
JOIN transactions t
```

La consulta principal une la serie de números “n” con la tabla transactions (con el alias “t”)

ON n.n <= 1 + (LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')))

La condición de unión ON asegura que el número “n” sea igual o menor que el recuento de ids dentro de la columna product_ids.

La función SUBSTRING_INDEX se utiliza para extraer los IDs de productos individuales de la lista separada por comas.

SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.n), ',', -1) as product_id

SUBSTRING_INDEX(t.product_ids, ',', n.n)` extrae la subcadena hasta la n-ésima ocurrencia de la coma (posición específica).

La porción (',', -1)` extrae la subcadena después de la última coma, lo que proporciona el n-ésimo ID de producto.

Se le asigna al resultado el alias product_id. El resultado final incluye el ID de transacción (t.id) y el ID de producto extraído (product_id).

El resultado obtenido incluye 1457 elementos enlistados donde se puede apreciar por ejemplo que la transacción con id 02C6201E-D90A-1859-B4EE-88D2986D3B02 tiene 3 elementos, que comparado con la tabla transactions tiene anidados los id 71, 1, 19

id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
0466A42E-47CF-8D24-FD01-C0B689713128	47
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	43
063FBA79-99EC-66FB-29F7-25726D1764A5	47
063FBA79-99EC-66FB-29F7-25726D1764A5	67
063FBA79-99EC-66FB-29F7-25726D1764A5	31
063FBA79-99EC-66FB-29F7-25726D1764A5	5
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	83
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	79
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	31
07A46D48-31A3-7E87-65B9-0DA902AD109F	47
07A46D48-31A3-7E87-65B9-0DA902AD109F	23
09DE92CE-6F27-2BB7-13B5-9385B2B3B8E2	67
09DE92CE-6F27-2BB7-13B5-9385B2B3B8E2	7

Muestra de tabla origen (transactions) donde se aprecia la id 02C6201E-D90A-1859-B4EE-88D2986D3B02 y sus product_ids anidados.

id	credit_card_id	company_id	timestamp	amount	declined	product_ids
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71,1,19

Se pueden comprobar que todas transacciones están correctamente enlistadas según el criterio esperado para poder realizar el conteo solicitado en el ejercicio 1 que se consigue a través de la siguiente consulta.

```
SELECT products.id AS id_prod, products.product_name as Producto, count(product_id) as
Compras_Realizadas
FROM products
LEFT JOIN comprados ON products.id = comprados.product_id
GROUP BY id_prod, Producto;
```

Obteniendo el siguiente resultado donde se enlistan los 100 productos existentes dentro de la tabla Products. El resultado arroja que son 26 productos los que han sido comprados y los productos más vendidos son: riverlands north(id 23) y winterfell(id 67) con 68 uds.
:

id_prod	Producto	Compras_Realizadas
1	Direwolf Stannis	61
10	Karstark Dorne	0
100	south duel	0
11	Karstark Dorne	48
12	duel Direwolf	0
13	palpatine chewbacca	60
14	Direwolf	0
15	Stannis warden	0
16	the duel warden	0
17	skywalker ewok sith	61
18	Karstark warden	0
19	dooku solo	49
2	Tarly Stark	65
20	warden Karstark	0
21	duel Direwolf	0

SEGUNDO MÉTODO. SIN CREAR UNA TABLA NUEVA

```
SELECT products.id AS id, products.product_name AS Nombre_Producto,
COUNT(transactions.id) AS Unidades_Vendidas
FROM products
LEFT JOIN transactions ON FIND_IN_SET(products.id, transactions.product_ids)
GROUP BY id, Nombre_Producto
;
```

Mediante esta consulta se utiliza la función FIND_IN_SET dentro de la condición ON de la JOIN que devolverá los registros de la tabla transactions en el campo product_ids que contiene las id de la tabla products que vaya encontrando dentro de él.

Mediante esta función se buscará la posición de la cadena dentro de una lista de cadenas, los valores que puede devolver serán según el caso:

Si la cadena no se encuentra en la lista, la función devuelve **0**.
Si la cadena o la lista de cadenas es **NULL**, la función devuelve **NULL**.
Si la lista de cadenas está vacía (""), la función también devuelve **0**.

Es por ello que el método de eliminación de espacios entre las comas es necesario, porque puede devolver resultados cero, por ejemplo en un registro que tenga (2, 1,5) donde el 1 no sería sumado al haber un espacio después de la coma (de hecho esto fue comprobado al principio de esta tarea al obtenerse resultados incompletos y después de estudiar las opciones se encontró el método con TRIM para conseguir la modificación de los registros dentro de la columna product_ids.

El resultado obtenido mediante esta consulta es el siguiente:

id	Nombre_Producto	Unidades_Vendidas
1	Direwolf Stannis	61
10	Karstark Dorne	0
100	south duel	0
11	Karstark Dorne	48
12	duel Direwolf	0
13	palpatine chewbacca	60
14	Direwolf	0
15	Stannis warden	0
16	the duel warden	0
17	skywalker ewok sith	61
18	Karstark warden	0
19	dooku solo	49
2	Tarly Stark	65
20	warden Karstark	0

Utilizando LEFT JOIN asegura que se obtengan todos los id de la tabla products aunque no haya coincidencias en la tabla transactions; esto puede permitirnos saber si la consulta funciona adecuadamente. Dejando únicamente JOIN solo se mostrarían los resultados que tengan coincidencias.