Tasca S4.01. Creació de Base de Dades

Nivell 1

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules de les quals puguis realitzar les següents consultes:

Teniendo en cuenta la consigna, se procedió a crear el esquema, con las correspondientes tablas. Considerando el diseño del modelo dimensional de estrella, con una tabla de hechos, (transactions) y sus correspondientes tablas de dimensiones (products, users, companies, credit_cards).

```sql
CREATE DATABASE top_toys_inc; -- Creamos una nueva base de datos para alojar la información de las diferentes tablas

CREATE TABLE products (  -- Creamos la tabla products, teniendo en cuenta los campos y tipos de datos a cargar
id INT PRIMARY KEY AUTO_INCREMENT,
product_name VARCHAR (255),
price FLOAT,
colour VARCHAR (150),
weight FLOAT,
warehouse_id VARCHAR (100));

CREATE TABLE companies ( -- Creamos la tabla companies, teniendo en cuenta los campos y tipos de datos a cargar
company_id VARCHAR (15) PRIMARY KEY,
company_name VARCHAR (255),
phone VARCHAR (15),
email VARCHAR (100),
website VARCHAR (255)
);
```

```sql
CREATE TABLE credit_cards ( -- Creamos la tabla credit_cards, teniendo en cuenta los campos y tipos de datos a cargar
id VARCHAR (15) PRIMARY KEY,
user_id INT,
iban VARCHAR (50),
pan INT,
pin INT,
cvv INT,
track1 VARCHAR (255),
track2 VARCHAR (255),
expiring_date VARCHAR (255)
);
```

```sql
######### -- Creamos la tabla users, para luego unificar los datos de european_users y american_users,
-- teniendo en cuenta los campos y tipos de datos a cargar
CREATE TABLE users (
id INT PRIMARY KEY,
name VARCHAR (100),
surname VARCHAR (100),
phone VARCHAR (150),
email VARCHAR (150),
birth_date VARCHAR (100),
country VARCHAR (150),
city VARCHAR (150),
postal_code VARCHAR (100),
address VARCHAR (255)
);
```

```sql
######### -- Creamos la tabla transactions, teniendo en cuenta los campos y tipos de datos a cargar,
-- con sus respectivas FK y asi crear un modelo estrella

CREATE TABLE transactions (
id VARCHAR (255) PRIMARY KEY,
card_id VARCHAR (15),
business_id VARCHAR (50),
timestamp TIMESTAMP,
amount DECIMAL (10,2),
declined TINYINT,
products_id CHAR (50),
user_id INT,
lat FLOAT,
longitude FLOAT,
FOREIGN KEY (card_id) REFERENCES credit_cards (id),
FOREIGN KEY (business_id) REFERENCES companies (company_id),
FOREIGN KEY (user_id) REFERENCES users (id),
FOREIGN KEY (products_id) REFERENCES products (id) ON UPDATE CASCADE
);
```

Por otra parte, al crear la tabla ‘‘transactions”, se asignaron las correspondientes FK mediante la instrucción FOREIGN KEY / REFERENCES, y para la asignación particular de la FK ‘product_ids’, referenciada a la tabla ‘products’, se estableció una condición ON UPDATE CASCADE.

La cláusula ON UPDATE CASCADE indica que, si el identificador de un producto (id) se actualiza en la tabla products, la actualización se realizará en cascada en la tabla transactions. Cada operación que contenga un valor de identificador de producto correspondiente se actualizará automáticamente con el nuevo valor, ya que esta relación debe estar definida en la cláusula CONSTRAIN de la clave externa. (FK).

Luego de creada la base de datos y sus correspondientes tablas, al iniciar el proceso de carga de datos en cada una de ellas, se obtuvo un ‘Error Code’, en la ejecución de la instrucción LOAD DATA INFILE.

---Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement---

Por lo que mediante diferentes pruebas de ensayo y error, se detectó y resolvió la problemática, mediante el siguiente tutorial:

https://platzi.com/tutoriales/1272-sql-mysql2018/6354-cargar-datos-por-archivos-csv-o-de-texto-desde-la-terminal-y-solucion-error-gt-error-1290-hy000/

Teniendo que modificar un archivo (.ini) de configuración de MySQL para habilitar el directorio de carga, desde la carpeta localizada en la raiz de instalación del programa. (C:/MySQL/MySQL Server 8.0/Uploads) Y alojar en esta carpeta local, los archivos CSV con la información correspondiente para cada una de las tablas creadas.

```
82 •    SHOW VARIABLES LIKE "secure_file_priv";
```

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

Variable_name	Value
secure_file_priv	C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

```
72    ##########CARGAR DATOS TABLA PRODUCTS
73    # La sentencia LOAD DATA INFILE requiere que se especifique el archivo fuente y la tabla de destino
74    -- los parámetros añadidos especifican que los campos están separados por comas,
75    -- que las cadenas están encerradas por comillas dobles
76    -- y que la primera fila debe ser ignorada porque contiene cabeceras de columna.
77
78 •  LOAD DATA INFILE
79    'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' -- usar forward slash
80    INTO TABLE products
81    FIELDS TERMINATED BY ','
82    ENCLOSED BY '"'
83    LINES TERMINATED BY '\n'
84    IGNORE 1 ROWS;
```

#Al cargar los datos, para cada una de las tablas, se cambio el sentido del backslash, para que permitiese la lectura del archivo, desde su ruta original de acceso.

#También se tuvo que modificar la configuración de los valores de las variables de tiempo, y setear a la opción regional “ES”.

```
108 •   SHOW VARIABLES LIKE 'lc_time_names'; -- # Variable_name, Value 'lc_time_names', 'en_US'
109
110 •   SET lc_time_names = 'es_ES'; -- se cambia la configuración a regional ES
```

Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.

En esta consulta se concatenan las columnas, ‘name’ y ‘ surname’ para obtener un solo campo con el nombre y apellido de cada ‘user’. Luego, mediante una subconsulta, se obtienen los id de los usuarios que hayan realizado operaciones, por un monto mayor a 80. A partir de los datos obtenidos de la tabla ‘transactions’. Para luego en la consulta externa, vincular mediante la instrucción JOIN la columna con los nombres y apellidos, de la tabla ‘users’, a partir los usuarios solicitados en la consigna.

```sql
#Nivell 1 -- Exercici 1
SELECT CONCAT(users.name, " ", users.surname) AS user_name, plus_t.total_transactions
FROM users
JOIN
(
SELECT user_id, COUNT(id) AS total_transactions
FROM transactions
GROUP BY user_id
HAVING COUNT(id) >80
) as plus_t
ON users.id = plus_t.user_id;
```

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

user_name	total_transactions
Molly Gilliam	110
Dxwgi Hwcru	94
Bnyr Astuw	91
Sfzzoh Xgvfridxs	81

#Observaciones: en la consulta anterior no se consideró filtrar mediante la instrucción WHERE, si alguna de las operaciones realizadas por los usuarios, no ha obtenido un declined, en alguna de ellas. En el caso que se considerase este filtrado de la información, el resultado de la consulta variará en la cantidad de usuarios a obtener.

```sql
#Nivell 1 -- Exercici 1
SELECT CONCAT(users.name, " ", users.surname) AS user_name, plus_t.total_transactions
FROM users
JOIN
(
SELECT user_id, COUNT(id) AS total_transactions
FROM transactions
WHERE declined = 0
GROUP BY user_id
HAVING COUNT(id) >80
) as plus_t
ON users.id = plus_t.user_id;
```

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

user_name	total_transactions
Molly Gilliam	107
Dxwgi Hwcru	91
Bnyr Astuw	86

Exercici 2

Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.

```
38 •  SELECT ROUND(AVG(t.amount),2) AS media_amount, cc.iban
39    FROM transactions t
40    JOIN companies c
41    ON t.business_id = c.company_id
42    JOIN credit_cards cc
43    ON cc.id = t.card_id
44    WHERE c.company_name = 'Donec Ltd'
45    GROUP BY cc.iban
46    ORDER BY media_amount DESC;
```

Se realiza una consulta, vinculando mediante la instrucción JOIN las tablas ‘transactions’, ‘companies’, y ‘credit_cards’. Para luego agrupar, por la columna ‘iban’ de la tabla ‘credit_cards’, la media del monto de las operaciones efectuadas. Con la condición WHERE, donde de la columna ‘company_name’ la variable sea ‘Donect Ltd’.

media_amount	iban
680.69	XX3830178139196201993663 52
680.01	XX6377063573975703949739 13
645.46	XX9713939714652922023122 59
628.89	XX1718471169288923759693 07
608.68	XX2254246388185424062235 75
607.29	XX7488907290571957117660 71
605.41	TN9614563570667381893122
605.36	XX4819080340373642425911 85
597.19	XX1946755197392563357535 08
594.26	XX2159627660619671954934 37

Action	Message
SELECT ROUND(AVG(t.amount),2) AS media_amount, cc.iban FROM transactions t JOIN...	371 row(s) returned

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?

Para esta consulta, como primera acción se crea una CTE, (‘cc_activity_t) y mediante la instrucción ROW_NUMBER poder asignar a cada ‘transactions’ efectuada por una tarjeta de credito, un número. Las operaciones son ordenadas desde la ultima a la mas antigua, a traves de la instrucción ORDER BY, y mediante PARTITION BY de la columna ‘card_id’, se ha particionado la vinculación para cada una de ellas.

```sql
WITH cc_activity_t AS
(
SELECT transactions.card_id, transactions.timestamp, transactions.declined,
ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS num_transaction
FROM transactions
)
SELECT * FROM cc_activity_t;
```

card_id	timestamp	declined	num_transaction
CcS-4857	2024-10-25 13:11:54	0	1
CcS-4857	2024-10-07 16:43:17	0	2
CcS-4857	2024-07-27 10:50:49	0	3
CcS-4857	2024-02-08 14:55:10	1	4
CcS-4857	2024-01-15 20:18:34	0	5
CcS-4857	2022-09-21 22:52:27	0	6
CcS-4857	2022-05-20 08:36:18	0	7
CcS-4857	2021-10-02 03:58:28	0	8
CcS-4857	2021-09-30 07:08:18	0	9
CcS-4857	2021-06-30 22:34:08	0	10

#WITH cc_activity_t AS (SELECT transactions.card_id, transactions.timestamp, transactions.declined, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS num_transaction FROM transactions) SELECT * FROM cc_activity_t

/// 100000 row(s) returned

Una vez contamos con la CTE creada, procedemos a obtener los datos solicitados en la consigna. Mediante la instrucción SELECT de la columna card_id de la tabla temporal creada, (cc_activity_t) y utilizando una función condicional, CASE, indicar cuando se cumpla que la suma de los valores declinados sea menor o igual a 2, asignar el valor ‘Activa. Cuando no se cumpla esta condición, se asignará el valor ‘Inactiva’. Esto permitirá diferenciar aquellas transaccions que contabilicen menos de 3 operaciones declinadas. Tal lo solicitado en la consigna, y mediante la instrucción WHERE se filtran aquellas operaciones que cumplan con la condición de tener la asignación de num_transaction menor o igual a 3. Esto permitirá obtener solo hasta las tres ultimas operaciones efectuadas con cada una de las tarjetas de crédito. Es decir las mas recientes.

Por ultimo, se realiza mediante la función GROUP BY la agregación de los datos, a partir de la columna, ‘card_id’ correspondiente a la CTE creada.

```sql
WITH cc_activity_t AS
(
SELECT transactions.card_id, transactions.timestamp, transactions.declined,
ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS num_transaction
FROM transactions
)
SELECT card_id,
    CASE
        WHEN SUM(declined) <=2 THEN 'Activa'
        ELSE 'Inactiva'
    END AS cc_state
FROM cc_activity_t
WHERE num_transaction <=3
GROUP BY card_id;

```

Por ultimo, creamos una tabla temporal, que contenga los datos de la consulta anterior. Para luego aplicar la consulta solicitada en la consigna, a la información obtenida. Con la condición que el estado de las tarjetas sea igual a la variable ‘Activa’.

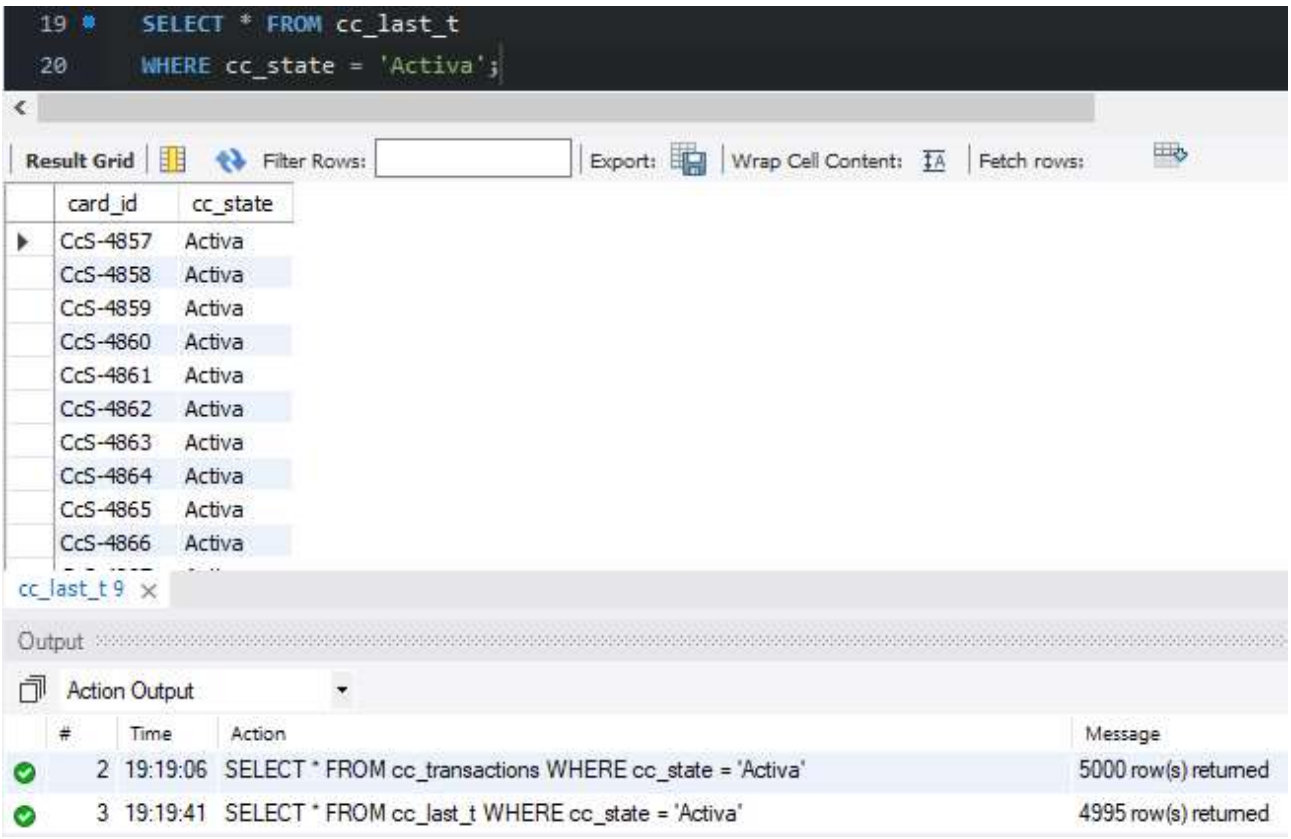
```sql
CREATE TEMPORARY TABLE IF NOT EXISTS cc_last_t AS (
WITH cc_activity_t AS
(
SELECT transactions.card_id, transactions.timestamp, transactions.declined,
ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS num_transaction
FROM transactions
)
SELECT card_id,
    CASE
        WHEN SUM(declined) <=2 THEN 'Activa'
        ELSE 'Inactiva'
    END AS cc_state
FROM cc_activity_t
WHERE num_transaction <=3
GROUP BY card_id
);
```

#SELECT * FROM cc_transactions

/// 5000 row(s) returned

#SELECT * FROM cc_last_t WHERE cc_state = 'Activa'

/// 4995 row(s) returned



Nivell 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.

Para este ejercicio, se han utilizado las siguientes funciones:

Función MySQL FIND_IN_SET()

Función REPLACE() de MySQL

Dada la complejidad de la consulta, y debido a la necesidad de realizar una operación para desanidar datos de una lista, para cada fila de la columna ‘product_ids’ de la tabla ‘transactions’. Se buscó la información para poder ejectutar esta acción en un asistente de IA, y además considerar la consulta con otros compañeros de la especialización.

Creamos una consulta que devuelva, una tabla con una asignación unica de cada producto con su correspondiente id de cada transaction. Tomando como referenciada los valores de la columna ‘product_ids’. Mediante la instrucción JOIN y utilizando la función FIND_IN_SET, localizamos la posición de cada subcadena, en la columna id de la tabla products, dentro de cada lista separada por comas. Luego, mediante la función REPLACE, se reemplazan caracteres separados por coma y se asignan “” como string individual para cada valor que encuentre en la lista de la columna product_ids de la tabla transactions. Con la condición que los indices de cada cadena sean mayores a 0.

```
SELECT transactions.id AS transaction_id, transactions.products_id, products.id AS unique_product_id
        FROM transactions
        JOIN products
        ON   FIND_IN_SET((products.id),
        REPLACE(transactions.products_id, ' ', '')) > 0;
```

#SELECT transactions.id AS transaction_id, transactions.products_id, products.id AS unique_product_id FROM transactions JOIN products ON FIND_IN_SET((products.id), REPLACE(transactions.products_id, ' ', '')) > 0

/// 253391 row(s) returned

transaction_id	products_id	unique_product_id
00E9A7E4-9FA1-42EC-85BC-ECBE44AAE558	1	1
011692F7-7669-48AF-B492-2A7466AD977A	1	1
02EF2169-FBB8-45B7-9827-FC60E70624E1	1	1
0333CE96-0EBA-40E2-8E27-506EFA41FCFB	1	1
0366EBA2-B53F-436C-A917-6310C826DD1F	1	1
04AE0AB9-4885-4FD0-9A70-F388EA546CA7	1	1
04CF444A-5A20-41A2-BA80-7E9BF75AB097	1	1
04DC619F-319A-4937-B183-FD9C4A1ED068	1	1
063E155E-B9B5-4610-AEBE-DC6C07B7D0DC	1	1
06F4599A-9063-4A16-9369-BBB16401B69E	1	1
082260F2-782F-4B8F-9D6D-84D73C34D6A3	1	1

Una vez que tenemos esta tabla temporal, con la asignación de cada transaction_id a un único product_id, creamos una tabla denominada transactions_products, a partir de la consulta anterior.

```
CREATE TABLE IF NOT EXISTS transactions_products AS (
SELECT transactions.id AS transaction_id, transactions.products_id, products.id AS unique_product_id
        FROM transactions
        JOIN products
        ON   FIND_IN_SET((products.id),
        REPLACE(transactions.products_id, ' ', '')) > 0
        );

SELECT unique_product_id, COUNT( transaction_id) AS salesxp
FROM transactions_products
GROUP BY unique_product_id;
```

CREATE TABLE IF NOT EXISTS transactions_products AS (SELECT transactions.id AS transaction_id, transactions.products_id, products.id AS unique_product_id FROM transactions JOIN products ON FIND_IN_SET((products.id), REPLACE(transactions.products_id, ' ', '')) > 0)

/// 253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

Una vez creada la tabla transactions_products, realizamos la consulta solicitada en la consigna.

	unique_product_id	salesxp
▶	1	2468
	10	2571
	11	2573
	30	2468
	12	2558
	62	2549
	14	2496
	90	2500
	94	2479
	100	2517
	49	2531
	13	2560
	56	2537
	76	2507
	18	2552

En esta consulta, realizamos una función de agregación COUNT, para contabilizar la cantidad de transacciones por producto, para cada uno de los valores unicos de la columna de productos (unique_product_id) y agrupamos por cada id de producto, que figuran en la nueva tabla creada.

#SELECT COUNT(transaction_id) AS salesxp, unique_product_id FROM transactions_products GROUP BY unique_product_id

/// 100 row(s) returned