

SPRINT 4: Creación y modelado de bases de datos

Descripción

Partiendo de algunos archivos CSV diseñarás y crearás tu base de datos.

NIVEL 1

Descarga los archivos CSV, estúdialos 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:

CREACIÓN BASE:

- 1 • `CREATE DATABASE sprint4;`
- 2 • `USE sprint4;`

CREACIÓN TABLAS:

- Creación tabla companies:

```
4 • CREATE TABLE companies(  
5     company_id VARCHAR(255) PRIMARY KEY,  
6     company_name VARCHAR(255),  
7     phone VARCHAR(255),  
8     email VARCHAR(255),  
9     country VARCHAR(255),  
10    website VARCHAR(255)  
11 );
```

```
/*-----  
Alternativas de escritura para la creación de tablas  
con FK y PK  
  
CREATE TABLE transactions(  
    id VARCHAR(255) PRIMARY KEY,  
    card_id VARCHAR(255),  
    business_id VARCHAR(255),  
    timestamp VARCHAR(255),  
    amount VARCHAR(255),  
    declined VARCHAR(255),  
    product_ids VARCHAR(255),  
    user_id VARCHAR(255),  
    lat VARCHAR(255),  
    longitude VARCHAR(255)  
    FOREIGN KEY fk_(user_id) REFERENCES users (id);  
);
```

```

CREATE TABLE transactions(
    id VARCHAR(255),
    card_id VARCHAR(255),
    business_id VARCHAR(255),
    timestamp VARCHAR(255),
    amount VARCHAR(255),
    declined VARCHAR(255),
    product_ids VARCHAR(255),
    user_id VARCHAR(255),
    lat VARCHAR(255),
    longitude VARCHAR(255),
    PRIMARY KEY (id),
    FOREIGN KEY fk_(user_id) REFERENCES users (id);
);*/

```

```

/*
#Codigo para insertar datos en una tabla creada

INSERT INTO nombre_tabla(nombre_de_los_campos_separados_por_comas) VALUES('STR',INT)
*/

```

- Creación tabla credit_cards

```

58 • CREATE TABLE credit_cards(
59     id VARCHAR(255) PRIMARY KEY,
60     user_id VARCHAR(255),
61     iban VARCHAR(255),
62     pan VARCHAR(255),
63     pin VARCHAR(255),
64     cvv VARCHAR(255),
65     track1 VARCHAR(255),
66     track2 VARCHAR(255),
67     expiring_date VARCHAR(255)
68 );

```

- Creación tabla products

```
70 • CREATE TABLE products(  
71     id VARCHAR(255) PRIMARY KEY,  
72     product_name VARCHAR(255),  
73     price VARCHAR(255),  
74     colour VARCHAR(255),  
75     weight VARCHAR(255),  
76     warehouse_id VARCHAR(255)  
77 );
```

- Creación tabla transactions

```
79 • CREATE TABLE transactions(  
80     id VARCHAR(255) PRIMARY KEY,  
81     card_id VARCHAR(255),  
82     business_id VARCHAR(255),  
83     timestamp VARCHAR(255),  
84     amount VARCHAR(255),  
85     declined VARCHAR(255),  
86     product_ids VARCHAR(255),  
87     user_id VARCHAR(255),  
88     lat VARCHAR(255),  
89     longitude VARCHAR(255)  
90 );
```

- Creación tabla users

```
92 • CREATE TABLE users(  
93     id VARCHAR(255) PRIMARY KEY,  
94     name VARCHAR(255),  
95     surname VARCHAR(255),  
96     phone VARCHAR(255),  
97     email VARCHAR(255),  
98     birth_date VARCHAR(255),  
99     country VARCHAR(255),  
100    city VARCHAR(255),  
101    postal_code VARCHAR(255),  
102    address VARCHAR(255)  
103  
104 );
```

Determino cuantas tablas hay creadas en la base datos.

```
107 #Cuenta cuantas tablas hay para determinar si he agregado el número correcto de datasets = 5.
108
109 • SELECT COUNT(*) AS total_tables
110 FROM information_schema.tables
111 WHERE table_schema = 'sprint4';
```

MODIFICO LOS TIPOS DE DATOS: LAS LLAVES FORÁNEAS Y LAS LLAVES PRIMARIAS DEBEN TENER EL MISMO TIPO DE DATO:

- Modificación el tipo de datos de la tabla de hechos [transactions]:

```
117 • ALTER TABLE transactions
118 MODIFY id VARCHAR(255);
119
120 • ALTER TABLE transactions
121 MODIFY card_id VARCHAR(15);
122
123 • ALTER TABLE transactions
124 MODIFY business_id VARCHAR(10);
125
126 • ALTER TABLE transactions
127 MODIFY timestamp TIMESTAMP;

129 • ALTER TABLE transactions
130 MODIFY amount DECIMAL(5,2);
131
132 • ALTER TABLE transactions
133 MODIFY declined BOOL;
134
135 • ALTER TABLE transactions
136 MODIFY product_ids VARCHAR(255);
137
138 • ALTER TABLE transactions
139 MODIFY user_id INT;
140
141 • ALTER TABLE transactions
142 MODIFY lat FLOAT;
143
144 • ALTER TABLE transactions
145 MODIFY longitude FLOAT;
146
147 • DESCRIBE transactions;
```

- Modificación tipo de datos dimensión [companies]

```
151 • ALTER TABLE companies
152     MODIFY company_id VARCHAR(10);
153
154 • ALTER TABLE companies
155     MODIFY company_name VARCHAR(255);
156
157 • ALTER TABLE companies
158     MODIFY phone VARCHAR(20);
159
160 • ALTER TABLE companies
161     MODIFY email VARCHAR(100);
162
163 • ALTER TABLE companies
164     MODIFY country VARCHAR(50);
165
166 • ALTER TABLE companies
167     MODIFY website VARCHAR(100);
168
169 • DESCRIBE companies;
```

- Modificación tipo de datos dimensión [credit_cards]

```
173 • ALTER TABLE credit_cards
174     MODIFY id VARCHAR(15);
175
176 • ALTER TABLE credit_cards
177     MODIFY user_id INT;
178
179 • ALTER TABLE credit_cards
180     MODIFY iban VARCHAR(100);
181
182 • ALTER TABLE credit_cards
183     MODIFY pan VARCHAR(50);
184
185 • ALTER TABLE credit_cards
186     MODIFY pin INT;
187
188 • ALTER TABLE credit_cards
189     MODIFY cvv INT;
190
191 • ALTER TABLE credit_cards
192     MODIFY track1 VARCHAR(255);
193
194 • ALTER TABLE credit_cards
195     MODIFY track2 VARCHAR(255);
196
197 • ALTER TABLE credit_cards
198     MODIFY expiring_date VARCHAR(50);
199
200 • DESCRIBE credit_cards;
```

- Modificación tipo de datos dimensión [products]

```
204 • ALTER TABLE products
205     MODIFY id INT;
206
207 • ALTER TABLE products
208     MODIFY product_name VARCHAR(255);
209
210 • ALTER TABLE products
211     MODIFY price VARCHAR(25);
212
213 • ALTER TABLE products
214     MODIFY colour VARCHAR(50);
215
216 • ALTER TABLE products
217     MODIFY weight DECIMAL(5,1);
218
219 • ALTER TABLE products
220     MODIFY warehouse_id VARCHAR(25);
221
222 • DESCRIBE products;
```

- Modificación tipo de datos dimensión [users]

```
227 ● ALTER TABLE users
228     MODIFY id INT;
229
230 ● ALTER TABLE users
231     MODIFY name VARCHAR(50);
232
233 ● ALTER TABLE users
234     MODIFY surname VARCHAR(100);
235
236 ● ALTER TABLE users
237     MODIFY phone VARCHAR(50);
238
239 ● ALTER TABLE users
240     MODIFY email VARCHAR(100);
241
242 ● ALTER TABLE users
243     MODIFY birth_date VARCHAR(25);
244
245 ● ALTER TABLE users
246     MODIFY country VARCHAR(50);
247
248 ● ALTER TABLE users
249     MODIFY city VARCHAR(50);
250
251 ● ALTER TABLE users
252     MODIFY postal_code VARCHAR(25);
253
254 ● ALTER TABLE users
255     MODIFY address VARCHAR(255);
256
257 ● DESCRIBE users;
```


- **Creo un Entitie Relationship Diagram EER Diagram con el Reverse Engineer para visualizar las relaciones.**

El análisis del EER Diagram permite definir visualmente el modelo de datos indicado, que en este caso corresponde a un modelo en estrella, por su particularidad de tener una sola tabla de hechos y varias dimensiones. A primera vista se puede decir, respecto a la dimensión (users_'country'), que se puede unir las tres tablas en una sola, si es necesario, agrupando a canada, estados unidos y reino unido. Por otra parte, conviene advertir que si se unen están tablas, la granularidad de los datos no es la misma, debido a que no se precisan datos de si esta estrella, hará parte de una galaxia y la granulidad de las entidades que representan a objetos del mundo real; en este caso una variable compleja como es el 'pais', sea necesaria para conocer con mayor profundidad detalles que permitan afinar la estrategia empresarial.

El término "EER Diagram" significa "Enhanced Entity-Relationship Diagram" en inglés, y se traduce como "Diagrama de Entidad-Relación Extendido/Mejorado" en español. Un diagrama EER es una representación gráfica de las entidades y las relaciones entre ellas en una base de datos, utilizando los conceptos de la modelización de datos de entidad-relación.

El modelo de entidad-relación (ER) es una técnica utilizada para modelar los datos en sistemas de bases de datos. Define entidades como objetos en el mundo real y las relaciones entre estas entidades. El modelo EER extiende el modelo ER básico al incluir conceptos adicionales como la generalización/especialización, la herencia y las relaciones de atributo.

En un diagrama EER, las entidades se representan como rectángulos y las relaciones entre ellas como líneas que conectan los rectángulos. Además, el modelo EER puede incluir otras notaciones gráficas para representar conceptos como la herencia, las restricciones de cardinalidad, los atributos, etc.

En el contexto de SQL, un diagrama EER se utiliza como una herramienta de diseño para visualizar la estructura de la base de datos y las relaciones entre las entidades, lo que ayuda a los desarrolladores y diseñadores a comprender y diseñar el esquema de la base de datos antes de implementarlo.

⊖ /*Una vez establecida la estrella, continuo con la cardinalidad y con la modificación del tipo de dato (ALTER) que inicialmente se estableció como VARCHAR(255).

Ordinalidad/Cardinalidad				Ordinalidad/Cardinalidad		Fact_Table(FT)	Dimension_Table(DT)	FT/DT
MIN:MAX				MIN:MAX	relación	PK	PK	FK
transactions	1:1	companies	0:N	transactions	1:N	id	company_id	business_id
transactions	1:1	credit_cards	0:N	transactions	1:N	id	id	card_id
transactions	1:N	products	0:N	transactions	N:N	id	id	N/A
transactions	1:1	users	0:N	transactions	1:N	id	id	user_id

*/

Existe una relación de N:M entre transactions y products, debido a que una transacción está compuesta por varios productos y un producto puede estar en varias transacciones, en estos casos se hace uso de una tabla intermedia o puente para romper la relación de N:M, esto se logra creando una tabla que contenga la PK de cada una de las dos tablas garantizando la identidad referencial y convirtiendo el modelo en estrella en un modelo en copo de nieve o Snowflake.

Una vez creada la tabla puente se establecen las FK en dicha tabla con referencia a la clave primaria de las tablas que presentan la relación N:M

Los tres tipos de modelos de datos más comunes en el diseño de bases de datos son:

Modelo de Datos en Estrella (Star Schema): En este modelo, los datos están organizados en una estructura central de hechos rodeada por varias tablas dimensionales. La tabla de hechos contiene las métricas o medidas de interés, mientras que las tablas dimensionales contienen atributos descriptivos que proporcionan contexto a las medidas. Esta estructura se asemeja a una estrella, con la tabla de hechos en el centro y las tablas dimensionales alrededor de ella.

Modelo de Datos Copo de Nieve (Snowflake Schema): Similar al modelo de estrella, el modelo de copo de nieve también tiene una tabla de hechos central y tablas dimensionales. Sin embargo, en el modelo de copo de nieve, las tablas dimensionales se normalizan aún más, dividiéndose en subdimensiones. Esto puede resultar en una estructura más compleja pero también puede ofrecer ventajas en términos de eficiencia de almacenamiento y mantenimiento.

Modelo de Datos Galaxia (Galaxy Schema): Aunque menos común que los anteriores, el modelo de datos galaxia surge cuando hay múltiples tablas de hechos que comparten las mismas dimensiones. En este modelo, hay varias tablas de hechos que se conectan a las mismas tablas dimensionales, formando una estructura similar a una galaxia con múltiples sistemas estelares.

Estos modelos de datos son importantes herramientas en el diseño de bases de datos para representar y organizar la información de manera eficiente y comprensible.

- **Creación de tabla puente, o intermedia para eliminar relación N:M**

```
338 • CREATE TABLE transaction_product(  
339     id_transaction VARCHAR(100),  
340     id_product INT,  
341     FOREIGN KEY(id_transaction) REFERENCES transactions(id),      -- FK tabla puente y tabla hechos  
342     FOREIGN KEY(id_product) REFERENCES products(id)              -- FK tabla puente y tabla dimensión products  
343 );
```

- Establezco la relaciones de clave foránea en la tabla de hechos (transactions) las cuales presentan una cardinalidad de 1:N. Con esta acción garantizo la integridad referencial entre las tablas.

```

350 • ALTER TABLE transactions
351     ADD FOREIGN KEY fk_credit_cards(card_id)
352     REFERENCES credit_cards(id);
353
354 • ALTER TABLE transactions
355     ADD FOREIGN KEY fk_companies(business_id)
356     REFERENCES companies(company_id);
357
358 • ALTER TABLE transactions
359     ADD FOREIGN KEY fk_users(user_id)
360     REFERENCES users(id);
361
362 • SHOW CREATE TABLE transactions;

```

CARGA BASES DE DATOS AL MODELO

Determino en donde se encuentra configurado por defecto el almacenaje de los archivos que puedo cargar como bases de datos en MYSQL y lo modifico para que admita cualquier ubicación.

```

368 • SELECT @@secure_file_priv;

```

Antes de cargar los data sets con código no con el import wizard, debo modificar la configuración del MYSQL, esto lo logro ingresando a la carpeta oculta (ver/mostrar carpetas ocultas) en el disco C donde se encuentran alojada la configuración del MYSQL, "C:\ProgramData\MySQL\MySQL Server 8.0\my.ini". Puedo consultar esta ruta con [SELECT @@secure_file_priv;]. El objetivo es modificar el archivo 'my.ini' es un bloc de notas, debo buscar con ctrl+b la palabra 'secure', esto me lleva al código que indica de donde pueden provenir las bases de datos por defecto secure-file-priv="C:/ProgramData/MySQL/MySQL Server 8.0/Uploads", coloco un '#' antes de esta sentencia para que quede como un comentario y en la línea siguiente copio el siguiente código que me permitirá extraer datos de cualquier ubicación secure-file-priv="" . Desde 'Servicios' detengo a MYSQL y posteriormente guardo este archivo en el escritorio, una vez guardado lo corto y lo pego en "C:\ProgramData\MySQL\MySQL Server 8.0\my.ini" lo que me permitirá remplazar el archivo que hemos modificado previamente. Reestablezco el servicio de MYSQL para poder ejecutar el código para cargar los datos. NOTA: tener en cuenta que dependiendo del sistema Windows, Linux o macOS se puede trabajar para la ruta de un archivo, con el carácter de separación '\' o '/', en este sentido se debe intentar con estas versiones de ruta:

Ejemplo:

'\': "D:\DIEGO\Desktop\BOOTCAMP\ESPECIALIDAD ANALISIS DE DATOS\MySQL\Sprint 4\Bases S4\companies.csv"

O

'/': "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/companies.csv"

Dependerá del sistema operativo como se mencionó anteriormente.

- **Carga dataset (companies)**

```
400 • LOAD DATA
401   INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/companies.csv"
402   INTO TABLE companies
403   FIELDS TERMINATED BY ','
404   #ENCLOSED BY '"' -- no utilizo esta sentencia, porque la base no lo demanda, la puedo eliminar en este caso
405   LINES TERMINATED BY '\n' -- \n significa que el salto de una fila define el fin de una serie de datos a la siguiente fila o serie de datos.
406   IGNORE 1 LINES;
407
408 • SELECT * FROM companies;
```

- **Carga dataset (credit_cards)**

```
421 • LOAD DATA
422   INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/credit_cards.csv"
423   INTO TABLE credit_cards
424   FIELDS TERMINATED BY ','
425   LINES TERMINATED BY '\n'
426   IGNORE 1 LINES;
427
428 • SELECT * FROM credit_cards;
```

- **Carga dataset (products)**

```
446 • LOAD DATA
447   INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/products.csv"
448   INTO TABLE products
449   FIELDS TERMINATED BY ','
450   LINES TERMINATED BY '\n'
451   IGNORE 1 LINES;
452
453 • SELECT * FROM products;
```

- **Carga dataset (users)**

En este caso contamos con los usuarios de tres países diferentes, Canada, Reino Unido y Estados Unidos unifico estos tres países en la tabla users.

- **Data Set Users Canada : (users_ca)**

```
479 • LOAD DATA
480 INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/users_ca.csv"
481 INTO TABLE users
482 FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''''
483 LINES TERMINATED BY '\r\n'
484 IGNORE 1 LINES;
```

\r\n Es una secuencia de escape que representa un salto de línea en muchos sistemas informáticos.

Aquí está su significado y utilidad:

Carácter de retorno (\r): Este carácter, conocido como "carriage return" en inglés, es un control de caracteres que mueve el cursor de escritura al principio de la línea actual. En algunos sistemas antiguos, como los sistemas basados en máquinas de escribir, mover el cursor al principio de la línea permitía escribir sobre el texto existente en esa línea.

Carácter de nueva línea (\n): Este carácter, conocido como "line feed" en inglés, es otro control de caracteres que mueve el cursor de escritura a la siguiente línea en un documento de texto.

\r\n como secuencia de fin de línea: En muchos sistemas operativos modernos, como Windows, la combinación \r\n se utiliza como la secuencia de caracteres estándar para representar un salto de línea en archivos de texto. Esto significa que al final de cada línea en un archivo de texto en Windows, se espera encontrar estos dos caracteres para indicar el fin de la línea.

La utilidad de \r\n radica en que proporciona una convención estándar para representar saltos de línea en archivos de texto en sistemas Windows. Esto es importante para asegurar la portabilidad de archivos entre diferentes plataformas y para garantizar que los archivos de texto se muestren correctamente en editores de texto y otros programas.

- **Data Set Users UK : (users_uk)**

```
526 • LOAD DATA
527 INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/users_uk.csv"
528 INTO TABLE users
529 FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''''
530 LINES TERMINATED BY '\r\n'
531 IGNORE 1 LINES;
```

- **Data Set Users USA : (users_usa)**

```
538 • LOAD DATA
539 INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/users_usa.csv"
540 INTO TABLE users
541 FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''''
542 LINES TERMINATED BY '\r\n'
543 IGNORE 1 LINES;
```

- **Carga dataset (transactions)**

```
552  /* Fue necesario abrir el archivo en el bloc de notas, (formato txt) para poder ver como estaban delimitados los datos
553  en este caso, se evidencia que aunque aparentemente esta dividido por columnas, realmente los datos estan dividos por
554  punto y coma, esta es la ventaja de ver los datos como txt.*/
555
556  • LOAD DATA
557  INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint 4/Bases S4/transactions.csv"
558  INTO TABLE transactions
559  FIELDS TERMINATED BY ';'
560  LINES TERMINATED BY '\n'
561  IGNORE 1 LINES;
562
563  • SELECT * FROM transactions;
```

- **Carga dataset (transaction_product): Tabla Puente**

Previamente se creó la tabla (transaction_product) con el objetivo de poder relacionar la tabla de hechos transactions y la dimensión products, y romper la relación de muchos a muchos existente entre estas dos entidades, sin embargo a la hora de cargar los datos, en la base de datos proporcionada como csv, en la tabla transactions se puede observar que los valores de la columna products_ids de la tabla transactions presentan diferentes id de productos separados por comas dentro de un mismo campo, el objetivo para poder llevar estos datos a la tabla puente consiste en presentar el id de transacción con un producto no con múltiples. Esto permitirá generar en la tabla puente, registros de una misma transacción con diferentes productos, para poder lograrlo se requiere realizar las siguientes acciones:

1) Crear una tabla temporal que almacene los datos en su forma pura, es decir con varios productos en una misma transacción.

TABLA TEMPORAL:

Se trata de una tabla que se utiliza para almacenar datos temporalmente, y que se borra automáticamente al cerrar la sesión en MySQL. Las tablas temporales son útiles para almacenar información de manera temporal, como resultados de una consulta compleja.

Las tablas temporales son una herramienta muy útil en el mundo de la gestión de base de datos. Sin embargo, es importante recordar que los datos almacenados en una tabla temporal se borran automáticamente al cerrar la sesión en MySQL. Es fundamental utilizar esta funcionalidad con precaución y siempre tener una copia de seguridad de los datos importantes.

Las tablas temporales son herramientas útiles para aquellos que necesitan manejar grandes cantidades de información y almacenarla de manera efectiva durante un período corto de tiempo. Gracias a ellas, podrás ordenar los datos de manera fácil y rápida, lo que hará más sencillo su posterior tratamiento.

Ejemplo de creación de un tabla temporal:

```
CREATE TEMPORARY TABLE temp_empleados (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(50),  
    edad INT  
);
```

2) Insertar el dataset en la tabla temporal desde la tabla de hechos transactions

Una forma alternativa de cargar la data consiste en utilizar el LOAD DATA INFILE así:

```
LOAD DATA INFILE "D:/DIEGO/Desktop/BOOTCAMP/ESPECIALIDAD ANALISIS DE DATOS/MySQL/Sprint  
4/Bases S4/transactions.csv"  
INTO TABLE temp_T_P_VARCHAR  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\r\n'  
IGNORE 1 ROWS  
(temp_transaction, @dummy, @dummy, @dummy, @dummy, @dummy, temp_product, @dummy,  
@dummy, @dummy);
```

Es importante tener en cuenta en esta sentencia, el uso del @dummy:

El uso de @dummy en esta sentencia de carga de datos LOAD DATA INFILE en MySQL y es una convención para manejar columnas en el archivo CSV que no se van a cargar en la tabla de destino.

En este contexto, @dummy es simplemente un nombre de variable de usuario. Se usa como marcador de posición para indicar que se está ignorando el valor en esa posición del archivo CSV.

Por ejemplo, en la parte (temp_transaction, @dummy, @dummy, @dummy, @dummy, @dummy, temp_product, @dummy, @dummy, @dummy), indica que hay ocho columnas en el archivo CSV que no se están cargando en la tabla temp_T_P_VARCHAR, ya que están representadas por @dummy. Estas columnas adicionales en el CSV se están "saltando" durante el proceso de carga de datos.

Esta técnica es útil cuando el archivo CSV tiene más columnas de las que se necesitan o cuando la estructura del archivo no coincide completamente con la estructura de la tabla de destino, permitiendo que solo las columnas relevantes se carguen en la tabla mientras se omiten las demás.

3) Insertar el dataset en la tabla puente desde la tabla temporal, realizando la división de productos.

4) Eliminar la tabla temporal, aunque al cerrar MYSQL se elimina automáticamente como ya se ha mencionado.

- **PASO 1:** Creo una tabla temporal con tipo de dato VARCHAR debido a necesito cargar una cadena de números separados por comas y por lo tanto se leen como un string.

```
653 • CREATE TEMPORARY TABLE temp_T_P(
654     temp_transaction VARCHAR(255),
655     temp_product VARCHAR(255)
656 );
```

- **PASO 2:** Inserto el dataset para cada fila creada en la tabla temporal desde la tabla de hechos transactions. Como se emocionó una alternativa es con LOAD DATA INFILE.

```
667 • INSERT INTO temp_T_P(temp_transaction, temp_product)
668     SELECT id, product_ids
669     FROM transactions;
```

- **PASO 3:** Inserto el dataset en la tabla puente desde la tabla temporal, realizando la división de productos.

```
675 • INSERT INTO transaction_product(id_transaction, id_product)
676     SELECT temp_transaction,
677            SUBSTRING_INDEX(SUBSTRING_INDEX(temp_product, ',', numbers.n), ',', -1) AS product_id
678     FROM temp_T_P
679     JOIN (SELECT 1 AS n
680           UNION ALL SELECT 2
681           UNION ALL SELECT 3
682           UNION ALL SELECT 4) AS numbers -- Esta SELECT genera un tabla derivada que sirve para
683                                           -- organizar los registros en una sola columna de 1 a 4 porque
684                                           -- cada transacción tiene maximo 4 productos.
685     ON CHAR_LENGTH(temp_product) - CHAR_LENGTH(REPLACE(temp_product, ',', '')) >= n - 1; -- Resto la longitud de la cadena
686     -- temp_product y la longitud de la misma cadena sin comas, y genero una condición que debe ser mayor o igual a
687     -- n-1 para poder indicar cual es el numero maximo de elementos que tiene una cadena.
```

SUBSTRING_INDEX() En MySQL Workbench es útil para dividir una cadena en subcadenas basadas en un delimitador específico y devolver una parte específica de esas subcadenas.

Ejemplo:

```
SELECT SUBSTRING_INDEX(CADENA, DELIMITADOR, CONTEO)
SELECT SUBSTRING_INDEX('Juan,María,Pedro', ',', 1);
```

SUBSTRING En MySQL se utiliza para extraer una parte de una cadena de texto. Tiene varias formas de uso, pero la forma más común es la siguiente:

SUBSTRING(cadena, inicio [, longitud])

Cadena: La cadena de texto de la que deseas extraer una parte.

Inicio: La posición de inicio desde la cual comenzar a extraer caracteres. La primera posición es 1.
Longitud (opcional): La cantidad de caracteres que deseas extraer. Si no se proporciona, SUBSTRING extraerá todos los caracteres desde la posición de inicio hasta el final de la cadena.

Aquí hay algunos ejemplos para ilustrar cómo funciona SUBSTRING:

```
SELECT SUBSTRING('Hello World', 7); -- Devuelve 'World'  
SELECT SUBSTRING('Hello World', 7, 5); -- Devuelve 'World'
```

En el primer ejemplo, SUBSTRING('Hello World', 7) extrae todos los caracteres de la cadena comenzando desde la posición 7, por lo que devuelve 'World'. En el segundo ejemplo, SUBSTRING('Hello World', 7, 5) extrae 5 caracteres comenzando desde la posición 7, por lo que también devuelve 'World'.

Es importante tener en cuenta que las posiciones en SUBSTRING comienzan desde 1, no desde 0 como en algunos otros lenguajes de programación.

CHAR_LENGTH():

Se utiliza para devolver la longitud de una cadena de texto en términos de caracteres, no de bytes.

Detalles de esta función y sus atributos:

Cadena: La cadena de texto de la cual deseas obtener la longitud en caracteres.

Esta función es útil cuando se necesita determinar la longitud real de una cadena de texto que puede contener caracteres multibyte, como UTF-8, donde un solo carácter puede ocupar más de un byte.

Ejemplo de uso:

```
SELECT CHAR_LENGTH('Hola, mundo!');
```

Esto devolverá 11, ya que la cadena 'Hola, mundo!' contiene 11 caracteres.

Es importante tener en cuenta que CHAR_LENGTH() cuenta los caracteres, no los bytes. Si se necesita contar los bytes en lugar de los caracteres, se puede usar la función LENGTH() en lugar de CHAR_LENGTH(). La diferencia radica en cómo manejan los caracteres multibyte. CHAR_LENGTH() los cuenta como un solo carácter, mientras que LENGTH() cuenta los bytes.* /

REPLACE() :En MySQL Workbench se utiliza para reemplazar todas las ocurrencias de una subcadena dentro de una cadena más grande con otra subcadena especificada. Aquí los detalles de esta función y sus atributos:

Atributos de REPLACE():

Cadena_original: La cadena de la que se desea realizar el reemplazo.

Subcadena_a_reemplazar: La subcadena que deseas reemplazar.

Nueva_subcadena: La subcadena que se desea usar para reemplazar la subcadena original.

La función REPLACE() busca todas las ocurrencias de subcadena_a_reemplazar dentro de cadena_original y las reemplaza con nueva_subcadena.

Ejemplo de uso:

```
SELECT REPLACE('Hola mundo, hola todos', 'hola', 'adiós');
```

Este comando devolverá 'Adiós mundo, adiós todos', ya que todas las ocurrencias de 'hola' en 'Hola mundo, hola todos' han sido reemplazadas por 'adiós'.

Es importante destacar que REPLACE() distingue entre mayúsculas y minúsculas. Si se necesita realizar un reemplazo sin tener en cuenta la distinción entre mayúsculas y minúsculas, se puede usar la función REPLACE() junto con las funciones LOWER() o UPPER() para convertir la cadena a minúsculas o mayúsculas antes de realizar el reemplazo.



- **PASO 4)** Eliminar la tabla temporal, aunque al cerrar mysql se elimina automáticamente.

```
776 • DROP TEMPORARY TABLE temp_T_P;
```

Ejercicio 1

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

```
787 • SELECT users.id,  
788         name,  
789         surname,  
790         city,  
791         country,  
792         COUNT(transactions.user_id) AS no_transactions  
793 FROM users  
794 JOIN transactions ON users.id = transactions.user_id  
795 GROUP BY  
796     users.id,  
797     users.name,  
798     users.surname,  
799     users.city,  
800     users.country  
801 HAVING no_transactions >= 30;  
802
```

Result Grid						
Filter Rows: <input type="text"/>						
Exports: 						
Wrap Cell Content: 						
	id	name	surname	city	country	no_transactions
▶	92	Lynn	Riddle	Bozeman	United States	39
	267	Ocean	Nelson	Charlottetown	Canada	52
	272	Hedwig	Gilbert	Tuktoyaktuk	Canada	76
	275	Kenyon	Hartman	Richmond	Canada	48

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.

```
860 • SELECT company_name, iban, ROUND(AVG(amount),2) AS media
861 FROM credit_cards
862 JOIN transactions ON transactions.card_id = credit_cards.id
863 JOIN companies ON transactions.business_id = companies.company_id
864 WHERE company_name = 'Donec Ltd'
865 GROUP BY company_name, iban;
866
```

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content:			
	company_name	iban	media
▶	Donec Ltd	PT87806228135092429456346	203.72

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:

Paso 1: Creo la tabla que va a contener los datos requeridos.

```
874 • CREATE TABLE credit_cards_status(
875     card_id VARCHAR(15) PRIMARY KEY,
876     status VARCHAR(15),
877     FOREIGN KEY fk_credit_cards_status(card_id) REFERENCES credit_cards(id)
878 );
```

Paso 2: Inserto en la tabla los datos

```
882 • INSERT INTO credit_cards_status (card_id, status)
883 SELECT
884     card_id,
885     CASE
886         WHEN SUM(declined) >= 3 THEN 'Inactiva'
887         ELSE 'Activa'
888     END AS status
889 FROM (
890     SELECT
891         card_id,
892         declined,
893         ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS fr
894     FROM
895         transactions
896 ) AS last_transactions
897 WHERE
898     fr <= 3
899 GROUP BY
900     card_id;
```

Ejercicio 1

¿Cuántas tarjetas están activas?

```
908 • SELECT COUNT(card_id) AS tarjetas_activas
909 FROM credit_cards_status;
910
```

Result Grid		Filter Rows:	Export:	Wrap Cell Cor
	tarjetas_activas			
▶	275			

NIVEL 3

Ejercicio 1

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

```
923 • SELECT product_name, COUNT(id_product) total_products
924 FROM products p
925 JOIN transaction_product t_p ON t_p.id_product = p.id
926 GROUP BY product_name
927 ORDER BY total_products DESC;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Contents:
	product_name	total_products			
►	Direwolf Stannis	106			
	skywalker ewok	100			
	riverlands north	68			
	Winterfell	68			
	Direwolf riverlands the	66			
	Tarly Stark	65			
	duel	65			
	Tully	62			
	skywalker ewok sith	61			
	jinn Winterfell	61			
	palpatine chewbacca	60			
	kingsblood Littlefinger...	58			
	Winterfell Lannister	57			
	duel tourney	57			
	north of Casterly	54			
	Tully Dorne	54			
	Lannister Barratheon ...	53			
	duel tourney Lannister	51			
	Direwolf Littlefinger	51			
	dooku solo	49			
	Tully maester Tarly	49			
	Karstark Dorne	48			
	Lannister	47			
	Dorne bastard	47			