### Nivel 1

### La base de datos creada se llama “TransactionsT4”. Inicialmente tendrá 4 tablas:

### Companies

### Credit\_cards

### Users\_all

### Transactions

### Las tablas siguen un esquema de tipo estrella, con la tabla Transactions como tabla de hechos y las otras como tabla de dimensiones. Todas estas tienen una relación de 1 a N con la tabla central, ya que cada compañía, usuario y tarjeta de crédito puede realizar muchas transacciones.

### DB TransactionsT4

### 

### 

### Para una explicación de la creación de la base de datos y de las tablas ver la sección ANEXO al final del documento.

### Ejercicio 1

### Los usuarios con más de 30 transacciones son los que tienen los siguientes ID:

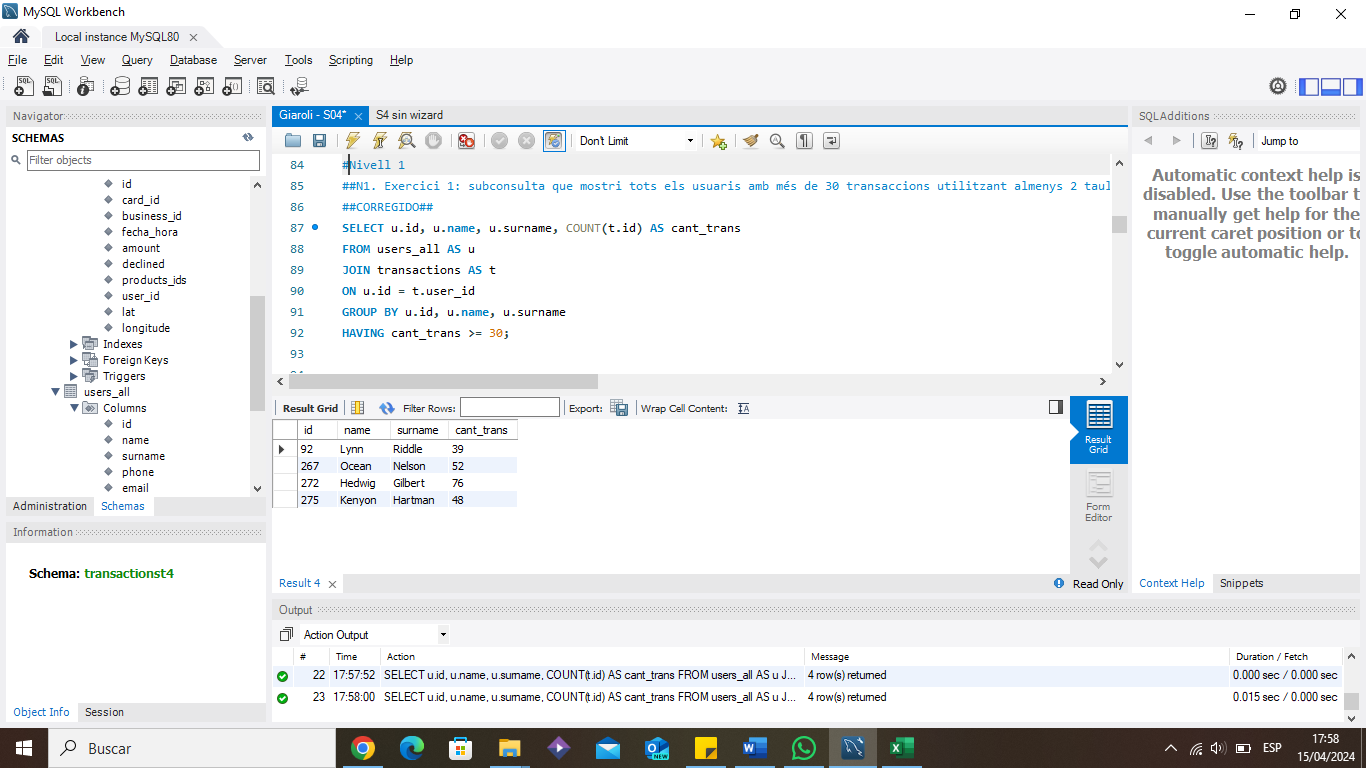
### 92

### 267

### 272

### 275

Corregido: reemplazamos subquery por join para hacer más eficiente la consulta



### Ejercicio 2

### La media de la suma de transacciones por IBAN de la compañía Donec Ltd. es de 203.715 ya que ha realizado dos transacciones con la misma tarjeta.

### Nivel 2

### Antes de crear la tabla que refleje el estado de las tarjetas, cambiaremos el nombre del campo timestamp a fecha\_hora para evitar errores y le daremos formato de tipo datetime para poder ordenar las transacciones por fecha.

### 

### Utilizamos una tabla temporal (t1) para poder crear la tabla. La función row\_number nos permite asignar un número a cada transacción y partition by nos permite hacer una partición de la enumeración para separar por cada tarjeta. Con order by ordenamos las transacciones por fecha realizada.

### Corregido: simplificamos la tabla

### Por último, añadimos una columna “Status” que analiza cada caso (CASE WHEN) y asigna el status “Inactiva” sólo si la suma de las últimas tres transacciones es igual que 3, es decir si las últimas 3 transacciones han sido declinadas.

### Como todas las transacciones por tarjeta salen Activas, editaremos dos transacciones para que se cumpla el caso Inactiva, para ver si las analiza correctamente.

### 

### Comprobamos con las transacciones editadas que el código funciona:

### Volvemos a colocar la tabla Transactions con sus datos originales, acabamos la comprobación del código de la tabla card\_status.

### 

### Ejercicio 1

### Hay 275 tarjetas activas, es decir todas las tarjetas. Ya que de las 587 transacciones realizadas, 87 fueron declinadas, pero cada operación declinada ha sido con diferentes tarjetas. Con lo que ninguna tarjeta ha tenido las últimas tres transacciones declinadas como para considerarse inactiva.

### 

### Nivel 3

### Primero creamos la tabla Products e importamos los datos:

### 

### Luego creamos la tabla puente “Products\_per\_transactions” entre Products y Transactions para romper la relación N-N entre ambas.

### 

### Esta tabla puente se compone de dos campos:

### uno es id\_transaccion que contiene los id de cada transacción realizada y actúa como FK con la tabla Transaccions, con la que tiene una relación de N-1.

### Otro es id\_product que contiene los id de cada producto y actúa como FK con la tabla Products, con la que tiene una relación de N-1.

### Añadiremos las FK luego de insertar los datos en la tabla para facilitar la inserción.

### A continuación, insertamos los datos en la tabla puente a partir de las columnas que ya existen en la tabla Transacions.

### 

### Para ello, seguiremos los siguientes pasos:

### Separamos cada producto del campo products\_ids con la función SUBSTRING\_INDEX, la aplicamos dos veces porque sino nos quedaría el primer producto sin separar

### Haremos una JOIN con una tabla llamada numbers, creada por un producto cartesiano (es decir una CROSS JOIN) entre una secuencia de números generada con la función ROW\_NUMBER para cada fila de la tabla transactions y una tabla ficticia de 10 números, para asegurar que al menos habrá 10 filas disponibles para los productos de cada transacción

### La JOIN la hacemos teniendo en cuenta el cálculo de la cantidad de productos que hay en cada registro del campo products\_ids, para ello restamos la función LENGTH para la longitud de la cadena con comas menos la sin comas, así sabremos la cantidad de comas que hay y le sumamos uno para tener el total de productos por registro.

### Al comprobar la creación de la tabla encontramos que el campo id\_product tiene espacios en blanco, los quitamos con la función TRIM:

### 

### Ahora creamos las FK y añadimos un índice en la tabla products:

### Ejercicio 1

### Para obtener un listado de la cantidad de veces que se ha “vendido” cada producto tendremos en cuenta que la transacción no haya sido declinada (declined = 0). Obtenemos el listado con la siguiente consulta:

### 

### Para que el resultado pueda ser ordenado por id de producto, cambiaremos el formato de varchar a integer. Como es FK la desactivamos momentáneamente:

### 

### Corremos de nuevo la misma consulta que ahora por defecto sale ordenada por id de producto:

### 

### Para interpretar los resultados, podemos ver el primer renglón que el producto con id = 1 se ha vendido 51 veces. Si el producto no aparece en el listado es porque aún no se ha vendido ninguna vez.

### Si quisiéramos obtener un listado tanto con los productos que se han vendido, como los que no se han vendido nunca, podemos hacer una subquery con la consulta anterior:

### ESTRUCTURA DE LA BASE DE DATOS AL FINAL DEL SPRINT:

### 

ANEXO

### Para crear la base de datos y el modelo realizamos los siguientes comandos:

### Creamos la Base de datos y la tabla companies

### 

### Creamos las tablas credit\_Cards, users\_all y transactions

### 

### 

### 

### Insertamos los datos en algunas tablas con "Table Date Import Wizard". Y en otras lo hacemos con código. Para ello:

1. Movemos el archivo CSV a la ubicación segura especificada por la opción --secure-file-priv. Podemos encontrar esta ubicación ejecutando la siguiente consulta en MySQL:

SHOW VARIABLES LIKE 'secure\_file\_priv';

1. Una vez tenemos el archivo en esta ubicación, copiamos su ruta de acceso y cambiamos las barras hacia la otra dirección para que MySQL la lea correctamente.

### 

### Insertamos datos en la tabla credit\_cards:

### 