

Sprint	4
Alumno	Rodrigo Padilla
Revisado Por	Joseph Tapia
Fecha	16-10-2024
Objetivo	Partint d'alguns arxius CSV dissenyaràs i crearàs la teva 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:

Como primera tarea borrar la BD con la que estaba trabajando, para crear la nueva BD del ejercicio.

CREANDO LA BASE DE DATOS

```
19 • DROP DATABASE TRANSACTIONS;
20 • SHOW DATABASES
21
22 -- CREAM TABLAS
23
```

<

Output

📄 Action Output ▼

#	Time	Action	Message
✓ 1	10:25:10	SHOW DATABASES	10 row(s) returned
✓ 2	10:28:54	DROP DATABASE TRANSACTIONS	9 row(s) affected

Ya no está la BD transactions.

12 • `SHOW DATABASES;`

13

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [↕](#)

Database
▶ bdd_union
beijing_upc
facultad
information_schema
mysql
olympics
performance_schema
sys
world

Ahora vamos a crear la BD nueva, la nombraremos **SALES**.

21 • `CREATE DATABASE sales;`

22 • `SHOW DATABASES;`

23

24 `-- CREAR TABLAS`

25

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [↕](#)

Database
olympics
performance_schema
sales
sys
world

Result 3 ×








Output

Action Output

#	Time	Action	Message
✓ 3	10:31:02	SHOW DATABASES	9 row(s) returned
✓ 4	10:53:35	CREATE DATABASE sales	1 row(s) affected
✓ 5	10:53:49	SHOW DATABASES	10 row(s) returned

El nombre **SALES** obedece a las etapas exploratorias de los datos ya acontecidas hasta este punto. La BD refleja transacciones de ventas, con estado exitoso (declined = 0) o fallido (declined = 1)

Se crean los Scripts para las tablas a crear que provienen desde los CSV del Sprint.. Estos son:

 companies.csv	06 de febrer 2024, 4:25 PM
 credit_cards.csv	06 de febrer 2024, 4:25 PM
 products.csv	06 de febrer 2024, 4:25 PM
 transactions.csv	06 de febrer 2024, 4:25 PM
 users_ca.csv	06 de febrer 2024, 4:25 PM
 users_uk.csv	06 de febrer 2024, 4:25 PM
 users_usa.csv	06 de febrer 2024, 4:25 PM

Consideraciones en la creación de las tablas.

1.- Para los archivos de users, se creará solo una tabla USERS y se agregara un atributo(columna) que describa su "owner" **ELIMINAR LA COLUMNA OWNER PUEDE SER REDUNDANTE CON EL PAIS**

2.- La tabla TRANSACTIONS, al ser FACT TABLE, se creará al último por tener FKs que dependen de las otras tablas.

CREANDO LAS TABLAS

Es importante mencionar acá un concepto importante. El orden de creación.

Para un modelo de estrella siempre debemos primero crear las DIMENSIONES y en último lugar los FACT. Se seguirá este orden.

CREATE TABLE COMPANIES

```

12 • CREATE TABLE companies(
13     company_id    VARCHAR(6) PRIMARY KEY,
14     company_name  VARCHAR(255) NOT NULL,
15     phone         VARCHAR(30),
16     email         VARCHAR(50),
17     country       VARCHAR(50),
18     website       VARCHAR(50)
19 );
20

```

Output

Action Output

#	Time	Action	Message
245	11:13:28	DROP TABLE COMPANIES	Error Code: 1051. Unknown table 'sales.companies'
246	11:13:40	CREATE TABLE companies(company_id VARCHAR(6) PRIMARY KEY, c...	0 row(s) affected

CREATE TABLE CREDIT_CARDS

```

21 • CREATE TABLE credit_cards(
22     card_id    VARCHAR(8) PRIMARY KEY,
23     user_id    INT NOT NULL,
24     iban       VARCHAR(50) NOT NULL,
25     pan        VARCHAR(50) NOT NULL,
26     pin        VARCHAR(4) NOT NULL,
27     cvv        VARCHAR(4) NOT NULL,
28     track1     VARCHAR(255),
29     track2     VARCHAR(255),
30     expiring_date VARCHAR(15)
31 );
32

```

Output

Action Output

#	Time	Action	Message
246	11:13:40	CREATE TABLE companies(company_id VARCHAR(6) PRIMARY KEY, c...	0 row(s) affected
247	11:15:52	CREATE TABLE credit_cards(card_id VARCHAR(8) PRIMARY KEY, user...	0 row(s) affected

CREATE TABLE PRODUCTS

La tabla PRODUCTS será creada bajo una relación NO permanente, dado que dependerá de una tabla que se generará de manera temporal por cada consulta y permitirá ser relacionada con detalles.

```
33 • CREATE table products
34 (
35     product_id INT PRIMARY KEY,
36     product_name VARCHAR(255),
37     price FLOAT,
38     colour VARCHAR(7),
39     weight FLOAT,
40     warehouse_id VARCHAR(10)
41 );
42
```

Output :

Action Output

#	Time	Action	Message
✓ 247	11:15:52	CREATE TABLE credit_cards(card_id VARCHAR(8) PRIMARY KEY, user...	0 row(s) affected
✓ 248	11:17:32	CREATE table products (product_id INT PRIMARY KEY, product_name VARC...	0 row(s) affected

CREATE TABLE USERS

```
43 • CREATE TABLE users (
44     user_id INT PRIMARY KEY,
45     first_name VARCHAR(255),
46     last_name VARCHAR(255),
47     phone VARCHAR(30),
48     email VARCHAR(50),
49     birth_date VARCHAR(30),
50     country VARCHAR(50),
51     city VARCHAR(50),
52     postal_code VARCHAR(30),
53     address VARCHAR(255)
54 );
55
```

Output :

Action Output

#	Time	Action	Message
✓ 314	20:59:44	CREATE table products (product_id INT PRIMARY KEY, product_name VARCHAR(255), price F...	0 row(s) affected
✓ 315	21:00:19	CREATE TABLE users (user_id INT PRIMARY KEY, first_name VARCHAR(255), last_nam...	0 row(s) affected

CREATE TABLE TRANSACTIONS

```
57 • CREATE TABLE transactions
58 (
59     transac_id VARCHAR(50) PRIMARY KEY,
60     card_id VARCHAR(8) NOT NULL,
61     business_id VARCHAR(6) NOT NULL,
62     timestamp_aud VARCHAR(30),
63     amount DOUBLE, -- aca cambiaste a FLOAT el dato 030
64     declined TINYINT(1),
65     product_ids VARCHAR(512),
66     user_id INT,
67     lat DOUBLE,
68     longitude DOUBLE,
69     FOREIGN KEY (card_id) REFERENCES credit_cards(card_id),
70     FOREIGN KEY (business_id) REFERENCES companies(company_id),
71     FOREIGN KEY (user_id) REFERENCES users(user_id)
72 );
73
```

Output

Action Output

#	Time	Action	Message
✓ 249	11:18:21	CREATE TABLE users (user_id INT PRIMARY KEY, first_name VARCHAR(...	0 row(s) affected
⚠ 250	11:20:27	CREATE TABLE transactions (transac_id VARCHAR(50) PRIMARY KEY, ...	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will ...

En TRANSACTIONS tuve algunos problemas por FK, se resuelven pero ejecuta la creación con un warning. La tabla TRANSACTION de igual forma es creada.

⚠ 35 12:07:43 CREATE TABLE transactions(transac_id VARCHAR(50) PRIMARY KEY, c... 0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will ... 0.218 sec

Ahora revisamos todas las tablas de la BD Sales

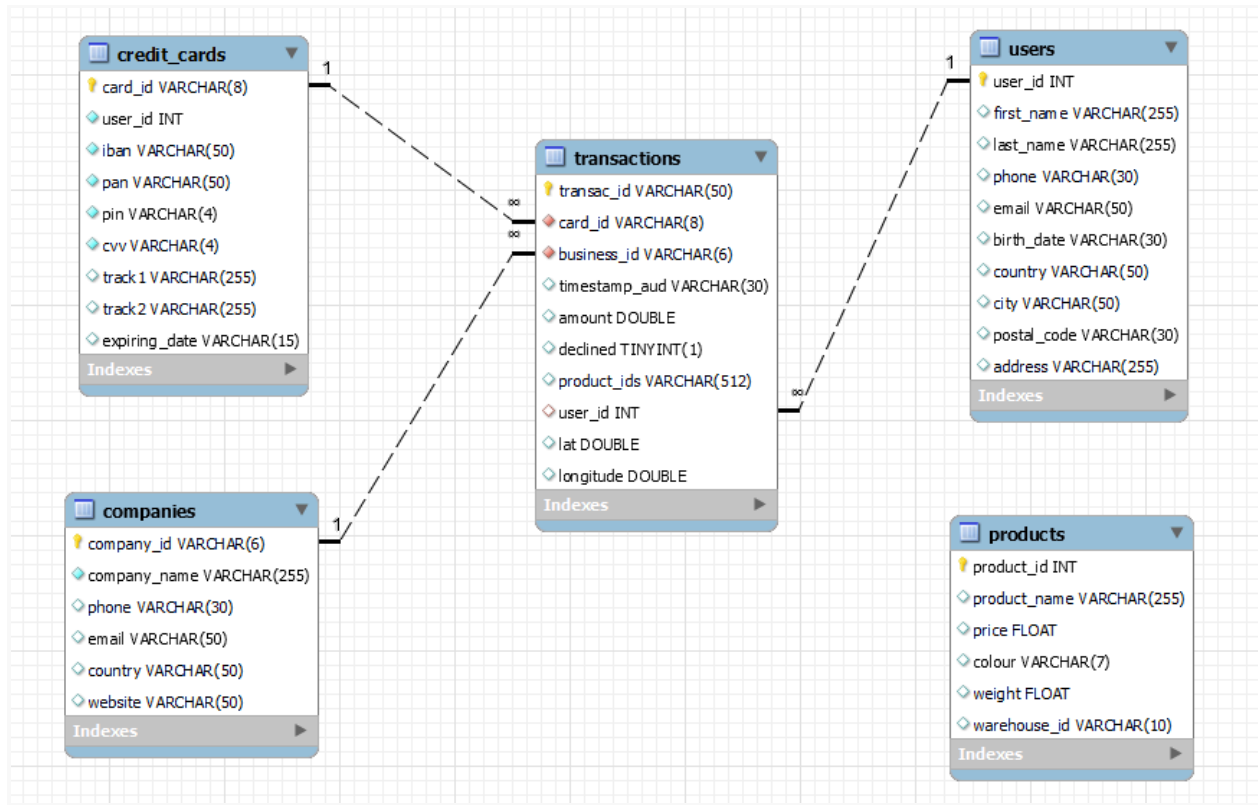
```
67 • SHOW FULL TABLES FROM SALES;
```

68

Result Grid | Filter Rows: | Export:

	Tables_in_sales	Table_type
▶	companies	BASE TABLE
	credit_cards	BASE TABLE
	products	BASE TABLE
	transactions	BASE TABLE
	users	BASE TABLE

Ahora vemos el resultado con el diagrama EERR de la BD en Workbench con opción Base de Datos, Ingeniería Inversa.



El modelo que nos muestra si obedece a una estrella, donde vemos al centro la tabla de hechos FACT TABLE, rodeado de las DIM TABLES que se vinculan de 1 en DIM a muchos en FACT, DIM → FACT. La tabla PRODUCTS no queda relacionada porque dependerá de una tabla temporal que será creada posteriormente en procedimiento almacenado.

CSV A LAS TABLAS (LOAD DATA)

CONSIDERACIÓN CARPETA DE SEGURIDAD DE MySQL

Ahora se realizará la carga de los datos con LOAD DATA de SQL. Para esto previamente nos aseguramos de respetar las normas de seguridad de MySQL, esto implica dejar los archivos solo en la “carpeta segura” que está definida y por ende permite utilizar. Para ello usamos:

```

8 • SHOW VARIABLES LIKE 'secure_file_priv';
9
10

```

Result Grid

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

Ahora podemos realizar la importación de los datos por cada tabla:

LOAD DATA TABLE COMPANIES

```

12 -- 1 LOAD DATA TABLE COMPANIES
13 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'
14 INTO TABLE companies
15 FIELDS TERMINATED BY ','
16 ENCLOSED BY '"'
17 LINES TERMINATED BY '\n'
18 IGNORE 1 ROWS;
19

```

Output

Action Output

#	Time	Action	Message
251	11:24:41	SHOW FULL TABLES FROM SALES	7 row(s) returned
252	11:30:38	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/co...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

LOAD DATA TABLE CREDIT_CARDS

```

20 -- 2 LOAD DATA TABLE CREDIT_CARDS
21 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
22 INTO TABLE credit_cards
23 FIELDS TERMINATED BY ','
24 ENCLOSED BY '"'
25 LINES TERMINATED BY '\n'
26 IGNORE 1 ROWS;
27

```

Output

Action Output

#	Time	Action	Message
252	11:30:38	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/co...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
253	11:31:29	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cre...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0

LOAD DATA TABLE PRODUCTS

Al realizar la importación de la tabla PRODUCTS, debemos agregar una modificación en el LOAD DATA que permita reemplazar los caracteres '\$' que contiene la columna price, de lo contrario nos da error.

```
28 -- 3 LOAD DATA PRODUCTS
29 -- aca se agrega una modificación previa a la carga para reemplazar el $ y respetar el tipo de datos FLOAT
30 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
31 INTO TABLE products
32 FIELDS TERMINATED BY ','
33 OPTIONALLY ENCLOSED BY '"'
34 LINES TERMINATED BY '\n'
35 IGNORE 1 ROWS
36 (product_id, product_name, @price, colour, weight, warehouse_id)
37 set price = REPLACE(@precio, '$', '') * 1.0;
38
```

Output

#	Time	Action	Message	Duration / Fetch
✓ 319	21:13:20	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' INTO TABL...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 ...	0.047 sec

LOAD DATA TABLE USERS

Con la tabla users, apuntamos algunas consideraciones:

- 1) Acá se debió cambiar la instrucción de finalización de la línea, porque de lo contrario da problemas la importación. esta modificación nos permitió resolverlo
LINES TERMINATED BY '\r\n'

a) CA

```

39 -- 4 LOAD DATA USERS
40 -- El LOAD de users contempla un UPDATE posterior a cada carga segun su "owner" CA, UK, USA
41
42 -- a) CA
43 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv'
44 INTO TABLE users
45 FIELDS TERMINATED BY ','
46 OPTIONALLY ENCLOSED BY '"'
47 LINES TERMINATED BY '\r\n'
48 IGNORE 1 ROWS
49 (user_id, first_name, last_name, phone, email, birth_date, country, city, postal_code, address);
50

```

Output

Action Output

#	Time	Action	Message
✓ 319	21:13:20	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' INTO TABL...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 ...
✓ 320	21:13:58	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv' INTO TABL...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 ...

b) USA

```

52 -- b) USA
53 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv'
54 INTO TABLE users
55 FIELDS TERMINATED BY ','
56 OPTIONALLY ENCLOSED BY '"'
57 LINES TERMINATED BY '\r\n'
58 IGNORE 1 ROWS
59 (user_id, first_name, last_name, phone, email, birth_date, country, city, postal_code, address);
60

```

Output

Action Output

#	Time	Action	Message
✓ 320	21:13:58	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv' INTO TABL...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 ...
✓ 321	21:19:00	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv' INTO TABL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 ...

c) UK. Por último los registros de UK,

```

61 -- c) UK
62 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'
63 INTO TABLE users
64 FIELDS TERMINATED BY ','
65 OPTIONALLY ENCLOSED BY '"'
66 LINES TERMINATED BY '\r\n'
67 IGNORE 1 ROWS
68 (user_id, first_name, last_name, phone, email, birth_date, country, city, postal_code, address);
69

```

Output

Action Output

#	Time	Action	Message
✓ 321	21:19:00	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv' INTO TABL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 ...
✓ 322	21:20:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv' INTO TABL...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 ...

LOAD DATA TABLE TRANSACTIONS

Por último importamos en la tabla de hechos FACT TABLE denominada TRANSACTIONS

```
71 -- 5 LOAD DATA TRANSACTIONS
72 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
73 INTO TABLE transactions
74 FIELDS TERMINATED BY ';'
75 OPTIONALLY ENCLOSED BY '"'
76 LINES TERMINATED BY '\r\n'
77 IGNORE 1 ROWS
78 (transac_id, card_id, business_id, timestamp_aud, amount, declined, product_ids, user_id, lat, longitude);
79
80 -- -----
```

Output

Action Output

#	Time	Action	Message
✓ 322	21:20:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv' INTO TABL...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 ...
✓ 323	21:20:44	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv' INTO T...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 ...

DESARROLLO DE EJERCICIOS

- Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.

```
37 • SELECT CONCAT(U.FIRST_NAME, " ", U.LAST_NAME) NOMBRE_USUARIO, TMP.CUENTA_TRANSAC
38 FROM USERS U
39 JOIN
40 (
41     SELECT USER_ID, COUNT(TRANSAC_ID) CUENTA_TRANSAC
42     FROM TRANSACTIONS
43     GROUP BY USER_ID
44     HAVING CUENTA_TRANSAC > 30
45 ) TMP
46 ON U.USER_ID = TMP.USER_ID
47 ORDER BY 2 DESC
```

Result Grid

	NOMBRE_USUARIO	CUENTA_TRANSAC
▶	Hedwig Gilbert	76
	Ocean Nelson	52
	Kenyon Hartman	48
	Lynn Riddle	39

Result 27 x

Output

Action Output

#	Time	Action	Message
✓ 272	12:05:35	SELECT CONCAT(U.FIRST_NAME, " ", U.LAST_NAME) NOMBRE_USUARIO, TMP.CUENTA_...	4 row(s) returned
✓ 273	12:05:47	SELECT CONCAT(U.FIRST_NAME, " ", U.LAST_NAME) NOMBRE_USUARIO, TMP.CUENTA_...	4 row(s) returned

- Exercici 2

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

```
52 • SELECT CC.IBAN IBAN, PROMEDIO_AMOUNT
53 FROM credit_cards CC
54 JOIN
55 (SELECT T.card_id TC, ROUND(AVG(T.amount),2) PROMEDIO_AMOUNT
56 FROM transactions T
57 JOIN
58 ( SELECT C.COMPANY_ID ID_CIA
59 FROM COMPANIES C
60 WHERE C.COMPANY_NAME LIKE '%Donec Ltd%'
61 ) TMP
62 ON TMP.ID_CIA = T.BUSINESS_ID
63 GROUP BY 1) TMP_B
64 ON TMP_B.TC = CC.CARD_ID;
```

Result Grid

IBAN	PROMEDIO_AMOUNT
PT87806228135092429456346	203,72

Result 33 x

Output

Action Output

#	Time	Action	Message
✓ 281	13:21:59	SELECT CC.IBAN IBAN, PROMEDIO_AMOUNT FROM credit_cards CC JOIN (SELECT T.card_id...	1 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?

Explicación del desarrollo del ejercicio

- Creemos la tabla CARD_STATES con CREATE TABLE, con ID de tarjeta y su STATUS,
- posterior a ello realizamos un INSERT de todas las tarjetas de crédito quedando por defecto como ACTIVA, entendiendo una relación de 1 a 1 entre ambas tablas.
- En un tercer paso se actualiza su estado final en caso de encontrar las últimas 3 transacciones como DECLINED, actualizando a "NO ACTIVA". Este paso se realiza con un UPDATE basado en un SELECT ANIDADO que busca esta condición en las 3 últimas transacciones.
- De acá en más, se debería agregar un Trigger a la tabla CREDIT_CARDS, a la creación un registro nuevo, que cree otro registro en CARD_STATES en ACTIVO.
- De acá en más el UPDATE de estado se podía programar para que corra cada día actualizando las tarjetas en su estado.
- Por último solo queda consultar de manera directa la tabla de estados CARD_STATES y nos responderá las tarjetas NO ACTIVAS con fecha de corte del día anterior.

```
71 • CREATE TABLE card_states
72 (
73     card_id VARCHAR(8) PRIMARY KEY,
74     card_status VARCHAR(20) default 'ACTIVA'
75 )
76 ;
77
78 • INSERT INTO CARD_STATES (card_id, card_status)
79     SELECT card_id, 'ACTIVA' FROM credit_cards
```

Output

#	Time	Action	Message
✓ 104	13:19:41	CREATE TABLE card_states (card_id VARCHAR(8) PRIMARY KEY, card_stat...	0 row(s) affected
✓ 105	13:20:01	SELECT * FROM sales.card_states LIMIT 0, 50000	0 row(s) returned
✓ 106	13:21:01	INSERT INTO CARD_STATES (card_id, card_status) SELECT card_id, 'ACTIV...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

81 • `SELECT * FROM CARD_STATES`

82

Result Grid

	card_id	card_status
▶	CcU-2938	ACTIVA
	CcU-2945	ACTIVA
	CcU-2952	ACTIVA
	CcU-2959	ACTIVA
	CcU-2966	ACTIVA

CARD_STATES 3 x

Output

Ahora se creará la consulta en SQL de actualización de la tabla CARD_STATUS.

UPDATE que actualiza por JOIN bajo un SELECT que cumple la condición.

71 • `UPDATE CARD_STATES`

72 `JOIN`

73 `(`

74 `SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP_AUD, TEMP.TRANSACTION_RANKING, TEMP.DECLINED`

75 `FROM`

76 `(`

77 `SELECT T.TRANSACTION_ID, T.CARD_ID, T.DECLINED, T.TIMESTAMP_AUD,`

78 `DENSE_RANK() OVER(PARTITION BY T.CARD_ID ORDER BY T.TIMESTAMP_AUD DESC) TRANSACTION_RANKING`

79 `FROM TRANSACTIONS T) TEMP -- genero ranking para enumerar las transacciones por tarjeta`

80 `WHERE TEMP.TRANSACTION_RANKING BETWEEN 1 AND 3`

81 `AND TEMP.CARD_ID IN`

82 `(SELECT T.CARD_ID -- Acá solo incluyo tarjetas con al menos 3 transacciones`

83 `FROM TRANSACTIONS T`

84 `WHERE T.DECLINED = 1`

85 `GROUP BY 1`

86 `HAVING COUNT(t.transaction_id) >= 3)`

87 `) TEMP2 -- A este nivel solo 27 registros que cumplen con 3 transacciones`

88 `ON CARD_STATES.CARD_ID = TEMP2.CARD_ID`

89 `SET CARD_STATES.CARD_STATUS = 'ACTIVA';`

90

Output

Action Output

#	Time	Action	Message
348	22:08:42	(SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP_AUD, TEMP.TRANSACTION_RANKING, TEMP.DECLINED	Error Code: 1064. You have an error in your SQL syntax; ...
349	22:08:58	SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP_AUD, TEMP.TRANSACTION_RANKING, TEMP.DECLINED	0 row(s) returned
350	22:14:08	UPDATE CARD_STATES JOIN(SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP_AUD, TEMP.TRANSACTION_RANKING, TEMP.DECLINED	0 row(s) affected Rows matched: 0 Changed: 0 Warning...

Resultado

La consulta UPDATE, no actualiza registros porque no existe ninguna tarjeta con sus últimos 3 movimientos DECLINED = 1.

Posterior a ello solo queda consultar de manera directa sobre la tabla, sabiendo que fue actualizada hace un día, por el estado = 'NO ACTIVA'

```

92 • SELECT C.CARD_ID CARD_ID, C.IBAN IBAN, CS.card_status ESTADO
93 FROM credit_cards C
94 JOIN card_states CS ON C.CARD_ID = CS.CARD_ID
95 WHERE CS.card_status = 'NO ACTIVA'

```

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

CARD_ID	IBAN	ESTADO
---------	------	--------

✓ 233 12:33:47 SELECT C.CARD_ID CARD_ID, C.IBAN IBAN, CS.card_status ESTADO FRO... 0 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.

STORED PROCEDURE Y CURSORES COMO SOLUCIÓN

El presente ejercicio significó un gran desafío al requerir resolverlo solo en el ámbito de SQL. Al evaluarlo se decidió como mejor alternativa el uso de procedimientos almacenados y cursores en SQL.

Por lo anterior explicamos su forma de funcionamiento.

- El procedimiento almacenado lee la tabla TRANSACTIONS y transforma cada IDS separado por comas en un registro nuevo
- mediante bucles, recorre cada String IDS para ir por cada caracter, buscando las comas para separar e insertar registro nuevo
- mediante el primer bucle y creando el cursor, recorre la tabla TRANSACTIONS leyendo cada registro

- cada vez que se ejecuta, elimina y vuelve a crear la tabla temporal TMP_DETALLE_PRODUCTOS

La tabla temporal creada TMP_DETALLE_PRODUCTOS permitirá así relacionarse con cada registro de la tabla PRODUCTS y además con la tabla TRANSACTIONS mediante el ID del producto y también mediante la ID de transacción.

El procedure **Crea_Detalle_Productos()** se entregará en archivo separado.

Para ejecutar la consulta SQL que responde a las veces que se ha vendido cada producto, primero se debe ejecutar este Procedure con un **CALL**.

Primero se creó el procedure y después se realizó el CALL para ejecutarlo.

```

98
99 • Call Crea_Detalle_Productos();
100 --
101 • SELECT P.PRODUCT_ID ID_PRODUCTO, P.PRODUCT_NAME NOMBRE_PRODUCTO, COUNT(DP.ID_TRANSAC) VECES_VENDIDO
102 FROM TRANSACTIONS T, TMP_DET_PRODUCTS DP, PRODUCTS P
103 WHERE T.TRANSACTION_ID = DP.ID_TRANSACTION
104 AND DP.ID_PRODUCT = P.PRODUCT_ID
105 AND T.DECLINED = 0
106 GROUP BY 1, 2
107 ORDER BY 1
108
109
110
111
112

```

Output

#	Time	Action	Message
✓ 349	22:08:58	SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP_AUD, TEMP.TRANSACTION_RAN...	0 row(s) returned
✓ 350	22:14:08	UPDATE CARD_STATES JOIN(SELECT TEMP.TRANSACTION_ID, TEMP.CARD_ID, TEMP.TIMESTAMP...	0 row(s) affected Rows matched: 0 Changed: 0 Warning...
4 351	22:20:53	Call Crea_Detalle_Productos()	Running...

Después de correr el Procedure, ha insertado los registros. Lo comprobamos con el número total, que previamente se sumó manualmente cada ID en Excel y **son 1457 IDs** de productos desde la tabla **TRANSACTIONS**..

12 • `SELECT * FROM sales.tmp_det_products;`

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

	id_product	id_transac	timestamp_created
▶	67	69D90229-AD26-43C3-5AAF-8D332D3B3E62	2022-03-16 14:01:36 -- 2024-10-15 11:53:41
	29	69D90229-AD26-43C3-5AAF-8D332D3B3E62	2022-03-16 14:01:36 -- 2024-10-15 11:53:41
	3	FD89D51B-AE8D-77DC-E450-B8083FBD3187	2022-03-16 02:35:05 -- 2024-10-15 11:53:41
	2	FD89D51B-AE8D-77DC-E450-B8083FBD3187	2022-03-16 02:35:05 -- 2024-10-15 11:53:41
	73	FD89D51B-AE8D-77DC-E450-B8083FBD3187	2022-03-16 02:35:05 -- 2024-10-15 11:53:41

det_products 1 x

Output

Action Output

#	Time	Action	Message
✓ 225	11:53:41	CALL Crea_Detalle_Productos()	0 row(s) affected
✓ 226	11:55:22	SELECT * FROM sales.tmp_det_products LIMIT 0, 50000	1457 row(s) returned

Si coinciden las cantidades en TMP_DET_PRODUCT.

Ahora creamos la consulta que vincula los detalles de los productos con los productos y con transacciones. Al solicitar ventas realizadas, se considera solamente los DECLINED = 0, las que están en 1 no fueron realizadas exitosamente, por tanto no se deben contar.

111 • `SELECT P.PRODUCT_ID ID_PRODUCTO, P.PRODUCT_NAME NOMBRE_PRODUCTO, COUNT(DP.ID_TRANSAC) VECES_VENDIDO`
 112 `FROM TRANSACTIONS T, TMP_DET_PRODUCTS DP, PRODUCTS P`
 113 `WHERE T.TRANSACTION_ID = DP.ID_TRANSAC`
 114 `AND DP.ID_PRODUCT = P.PRODUCT_ID`
 115 `AND T.DECLINED = 0`
 116 `GROUP BY 1, 2`
 117 `ORDER BY 1`
 118

Result 18 x

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

	ID_PRODUCTO	NOMBRE_PRODUCTO	VECES_VENDIDO
▶	1	Direwolf Stannis	51
	2	Tarly Stark	56
	3	duel tourney Lannister	43
	5	skywalker ewok	42
	7	north of Casterly	44
	11	Karstark Dorne	40
	13	nalnatine chwhacca	51

Output

Action Output

#	Time	Action	Message
✓ 226	11:55:22	SELECT * FROM sales.tmp_det_products LIMIT 0, 50000	1457 row(s) returned
✓ 227	12:00:17	SELECT P.PRODUCT_ID ID_PRODUCTO, P.PRODUCT_NAME NOMBRE_P...	26 row(s) returned

La consulta nos responde LAS VECES que ocurrió la venta, lo consideramos como el evento de la venta, es decir cada vez que el ID sea parte del registro de transacción.

Este Stored Procedure, pertenece a la base de datos SALES y podría ser programado para que se ejecute todos los días cada una hora por ejemplo o bien cada noche.
