

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:

Per a importar les dades a MySQL Workbench, primer creem l'estructura de les taules. Especifiquem el nom de les diferents columnes, i de quin tipus seran. També indiquem quina és la primary key i, si hi ha foreign keys, amb quines columnes de quines taules es relacionen.

Un cop creades les taules, importem les dades. La primera dificultat s'ha presentat quan deia que no tenia permís per accedir al directori on es trobaven guardats els csv. Per solucionar això hem utilitzat el cmd "show variables like 'secure_file_priv';". D'aquest directori sí que pot importar arxius. Per tant, copiem els csv en aquesta carpeta. L'altra cosa que hem hagut de fer ha estat esborrar 'LOCAL' de la instrucció 'LOAD DATA LOCAL INFILE'.

A continuació hem importat les dades. Al codi hem d'especificar, entre d'altres coses, el delimitador i el salt de línia. A companies i credit_cards, aquests eren coma i \n, respectivament. No obstant, a transactions hem hagut de canviar la coma per punt i coma i a users hem hagut de canviar \n per \r\n.

Per users, en lloc de crear tres taules i fer union, el que hem fet és crear una única taula i, com que els csv tenien la mateixa estructura, els hem anat incorporant a la taula un darrere l'altre. No hem hagut d'afegir cap columna adicional per mantenir un registre de quin csv provenien les dades perquè ja hi havia la columna 'country'.

A continuació copio el codi per fer tot això:

```
## Creem la base de dades
```

```
CREATE DATABASE sprint4db;
```

```
## Creem l'estructura de les taules
```

```
### Taula company
```

```
CREATE TABLE sprint4db.companies (  
    company_id VARCHAR(255) PRIMARY KEY,  
    company_name VARCHAR(255),  
    phone VARCHAR(255),  
    email VARCHAR(255),  
    country VARCHAR(255),  
    website VARCHAR(255)  
);
```

Taula credit_cards

```
CREATE TABLE sprint4db.credit_cards (  
  id VARCHAR(255) PRIMARY KEY,  
  user_id INT,  
  iban VARCHAR(255),  
  pan VARCHAR(255),  
  pin VARCHAR(255),  
  cvv VARCHAR(255),  
  track1 VARCHAR(255),  
  track2 VARCHAR(255),  
  expiring_date varchar(255)  
);
```

Taula users

```
CREATE TABLE sprint4db.users (  
  id INT PRIMARY KEY,  
  name VARCHAR(255),  
  surname VARCHAR(255),  
  phone VARCHAR(255),  
  email VARCHAR(255),  
  birth_date VARCHAR(255),  
  country VARCHAR(255),  
  city VARCHAR(255),  
  postal_code VARCHAR(255),  
  address VARCHAR(255)  
);
```

Taula transactions

```
CREATE TABLE sprint4db.transactions (  
  id VARCHAR(255) PRIMARY KEY,  
  card_id VARCHAR(255),  
  business_id VARCHAR(255),  
  timestamp TIMESTAMP,  
  amount DECIMAL(10, 2),  
  declined BOOLEAN,  
  product_ids VARCHAR(255),  
  user_id INT,  
  lat FLOAT,  
  longitude FLOAT,  
  FOREIGN KEY (business_id) REFERENCES companies(company_id),  
  FOREIGN KEY (card_id) REFERENCES credit_cards(id),  
  FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

Importem les dades dels csvs

Passos previs per identificar per que no podia accedir als csvs
#show global variables like 'local_infile';

```
#show variables like 'secure_file_priv';  
#set global local_infile=true;
```

```
### companies
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'  
INTO TABLE companies  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES;
```

```
## credit_cards
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'  
INTO TABLE credit_cards  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES;
```

```
### users
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv'  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\r\n'  
IGNORE 1 LINES;
```

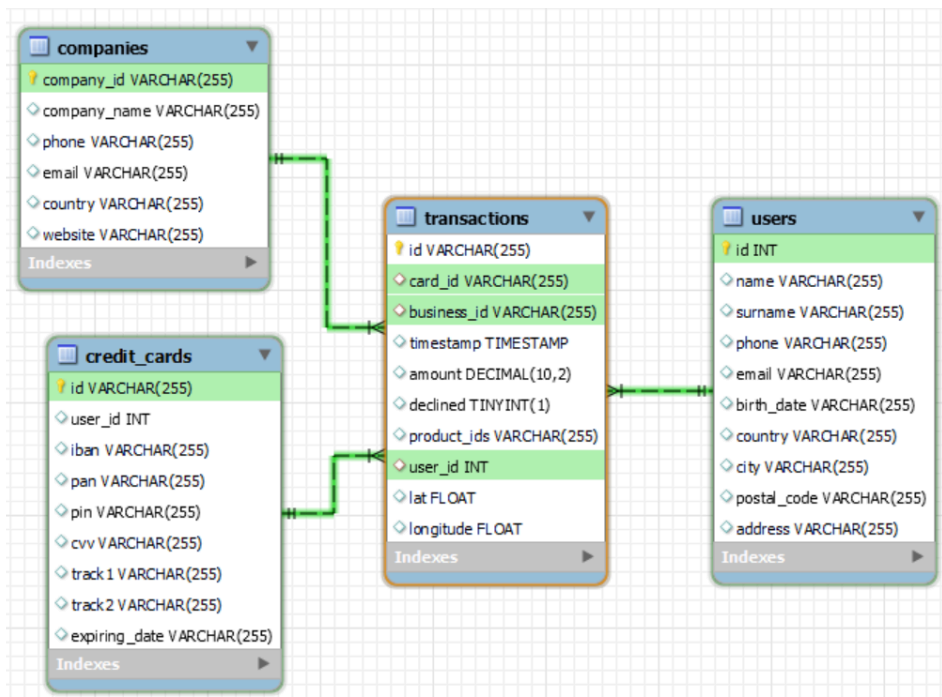
```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\r\n'  
IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv'  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\r\n'  
IGNORE 1 LINES;
```

```
### transactions
```

```
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'  
INTO TABLE transactions  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
LINES TERMINATED BY '\n'  
IGNORE 1 LINES;
```

2	15:40:51	CREATE DATABASE sprint4db	1 row(s) affected	0.015 sec
3	15:40:51	CREATE TABLE sprint4db.companies (company_id VARCHAR(255) PRIMARY ...	0 row(s) affected	0.047 sec
4	15:40:51	CREATE TABLE sprint4db.credit_cards (id VARCHAR(255) PRIMARY KEY, use...	0 row(s) affected	0.031 sec
5	15:40:51	CREATE TABLE sprint4db.users (id INT PRIMARY KEY, name VARCHAR(...	0 row(s) affected	0.063 sec
6	15:40:51	CREATE TABLE sprint4db.transactions (id VARCHAR(255) PRIMARY KEY, card_j...	0 row(s) affected	0.078 sec
8	15:42:04	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companie...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
9	15:42:04	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_car...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0	0.391 sec
10	15:42:04	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
11	15:42:05	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
12	15:42:05	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
13	15:42:05	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactio...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0	0.218 sec



- Exercici 1

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

```

128 • SELECT name, surname
129 FROM users
130 WHERE id IN (
131     SELECT user_id
132     FROM transactions
133     WHERE declined = 0
134     GROUP BY user_id
135     HAVING COUNT(id) > 30
136 );
  
```

Result Grid			Filter Rows:
	name	surname	
▶	Lynn	Riddle	
	Ocean	Nelson	
	Hedwig	Gilbert	

16 11:00:57 SELECT name, surname FROM users WHERE id IN (SELECT user_id FROM t... 3 row(s) returned 0.015 sec / 0.000 sec

- Exercici 2

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

```

129 • SELECT iban, ROUND(AVG(amount),2) as meanAmount, company_name
130      FROM transactions
131     LEFT JOIN credit_cards
132      ON card_id = credit_cards.id
133     LEFT JOIN companies
134      ON company_id = business_id
135     WHERE declined = 0 AND company_name = "Donec Ltd"
136     GROUP BY iban, company_name;

```

Result Grid				Filter Rows:	Export:
	iban	meanAmount	company_name		
▶	PT87806228135092429456346	42.82	Donec Ltd		

34 10:19:38 SELECT iban, ROUND(AVG(amount),2) as meanAmount, company_name FR... 1 row(s) returned 0.031 sec / 0.000 sec

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:

Creemos la tabla:

```
144 • CREATE TABLE credit_cards_status (  
145     card_id VARCHAR(255) PRIMARY KEY,  
146     card_status VARCHAR(255)  
147 );
```

206 13:11:41 CREATE TABLE credit_cards_status (card_id VARCHAR(255) PRIMARY ... 0 row(s) affected 0.047 sec



Incorporamos los datos:

```
149 • INSERT INTO credit_cards_status  
150     SELECT card_id,  
151     CASE  
152         WHEN SUM(declined) = 3 THEN 'not active'  
153         ELSE 'active'  
154     END AS card_status  
155     FROM (SELECT card_id, declined  
156     FROM (SELECT card_id, declined, timestamp,  
157         ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS rn  
158         FROM transactions) as orderedTransac  
159     WHERE rn <= 3) as threeLastTransac  
160     GROUP BY card_id;
```

207 13:12:36 INSERT INTO credit_cards_status SELECT card_id, CASE WHE... 275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0 0.047 sec

El razonamiento detrás de esta query es el siguiente. Necesito las tres últimas transacciones de cada tarjeta. Para ello, lo que hago es crear una columna que me enumere las transacciones realizadas por cada tarjeta. Para ello uso ROW_NUMBER. Esta función devuelve el número secuencial de una fila dentro de una partición de un conjunto de datos, empezando por 1 para la primera fila de cada partición. Así pues, en el PARTITION BY pongo card_id. Además, aprovecho para numerar las transacciones en función de la fecha en que se realizaron. Para ello uso el ORDER BY y especifico que sea DESC. De esta forma, las primeras transacciones de cada partición serán las más recientes. Una vez realizado este cálculo (subquery orderedTransac), me quedo únicamente con aquellos registros en que el número de fila sea menor o igual a 3 (subquery threeLastTransac). Finalmente, sumo los valores de declined por cada tarjeta (con SUM()) y GROUP BY y le asigno una etiqueta u otra en función de si suma tres o no (utilizando CASE).

Así es cómo luce la tabla:

Result Grid   Filter Rows:		
	card_id	card_status
▶	CcU-2938	active
	CcU-2945	active
	CcU-2952	active
	CcU-2959	active
	CcU-2966	active

La relacionamos con credit_cards.

```

164 • ALTER TABLE credit_cards_status
165     ADD FOREIGN KEY (card_id)
166     REFERENCES credit_cards(id);

```

✓ 208 13:16:44 ALTER TABLE credit_cards_status ADD FOREIGN KEY (card_id) REFEREN... 275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0 0.093 sec

En SQL no hay una instrucción específica para hacer una relación 1 a 1, porque no se contempla esa opción, ya que se trata de la misma tabla partida en dos. Así pues, cuando hacemos el diagrama, aparece una relación de 1 a muchos. Esto sólo lo podemos modificar manualmente a través de la interfaz gráfica.



Exercici 1

Quantes targetes estan actives?

```

159 • SELECT COUNT(*) as numberActiveCards
160     FROM credit_cards_status
161     WHERE card_status = 'active';

```

Result Grid   Filter Rows:	
	numberActiveCards
▶	275

✓ 50 11:38:34 SELECT COUNT(*) as numberActiveCards FROM credit_cards_status WHER... 1 row(s) returned 0.000 sec / 0.000 sec

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:

Primero creamos la tabla. En esta tabla, como es una tabla intermedia que nos debe permitir relacionar dos tablas mediante sus respectivas primary keys, la primary key será la combinación de ambas columnas (transaction_id y product_id). Por lo tanto, especificaremos que es una primary key compuesta.

```
178 • CREATE TABLE ProductsPerTransaction (  
179     transaction_id VARCHAR(255),  
180     product_id INT,  
181     PRIMARY KEY (transaction_id, product_id)  
182 );
```

209 13:18:07 CREATE TABLE ProductsPerTransaction (transaction_id VARCHAR(255), ... 0 row(s) affected

0.047 sec

A continuació introduïm els dades:

INSERT INTO ProductsPerTransaction

WITH RECURSIVE ProductsPerTransaction AS (

SELECT

id,

TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) AS split_value,

IF(LOCATE(',', product_ids) > 0, TRIM(SUBSTRING(product_ids, LOCATE(',', product_ids) + 1)), NULL) AS remaining_values

FROM

transactions

UNION ALL

SELECT

id,

TRIM(SUBSTRING_INDEX(remaining_values, ',', 1)) AS split_value,


```
        IF(LOCATE(',', remaining_values) > 0, TRIM(SUBSTRING(remaining_values,  
LOCATE(',', remaining_values) + 1)), NULL)
```

```
FROM
```

```
    ProductsPerTransaction
```

```
WHERE
```

```
    remaining_values IS NOT NULL
```

```
)
```

```
SELECT
```

```
    id as transaction_id,
```

```
    split_value as product_id
```

```
FROM
```

```
    ProductsPerTransaction;
```

210 13:19:39 INSERT INTO ProductsPerTransaction WITH RECURSIVE ProductsPerTra... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0 0.485 sec

Para hacer esto necesitamos un código recursivo.

En el primer bloque de código tenemos dos pasos. Primero, el SUBSTRING_INDEX lo que hace es buscar en el string especificado (en este caso, product_ids) el delimitador especificado (en este caso, una coma) tantas veces como le indiquemos (en este caso, 1) y quedarse con todo lo que hay a la izquierda. Si el número que le indicamos es positivo, empieza la búsqueda por la izquierda y se queda con todo lo que haya a la izquierda del delimitador. Por el contrario, si es negativo, empieza la búsqueda por la derecha y se queda con todo lo que haya a la derecha. Así pues, en este caso lo que está haciendo es quedarse con el primer valor de product_ids y lo coloca en una nueva columna llamada split_value. El segundo paso es poner el resto de productos que aparecen en el string en una nueva columna remaining_values. Concretamente, le decimos: busca si hay comas en product_ids (LOCATE(',',product_ids) > 0) y, si las hay, ponme todo lo que hay después de la coma (SUBSTRING(product_ids, LOCATE(',', product_ids) + 1) en una nueva columna llamada remaining_values. Si no las hay, ponme NULL. Estas columnas, junto con la columna id, constituyen la nueva tabla ProductsPerTransaction. La Todo esto lo mete en

A continuación, en el segundo bloque, hace lo mismo pero con la nueva tabla ProductsPerTransaction que incluye split_values y remaining_values. Como la columna split_values ya existe, lo que hace es poner el nuevo registro debajo. Y este código lo va repitiendo (y la tabla se va actualizando) hasta que remaining_values es NULL (WHERE remaining_values IS NOT NULL). Cuando es NULL, sale del loop.

Finalmente nos quedamos sólo con las columnas id (a la cual le cambio el nombre por transaction_id) y split_values (a la cual le cambio el nombre por product_id).

El TRIM lo que hace es eliminar los espacios que había en la columna original y quedarse únicamente con el número de producto.

Result Grid			Filter Rows:	Export:
	transaction_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		

15 10:58:14 CREATE TABLE ProductsPerTransaction AS WITH RECURSIVE ProductsP... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0 1.672 sec

Esta es la tabla ProductsPerTransaction resultante.

Ahora que tenemos la tabla, la linkamos con transactions y products.

Primero creamos la tabla products:

```

192 • CREATE TABLE sprint4db.products (
193     id INT PRIMARY KEY,
194     product_name VARCHAR(255),
195     price VARCHAR(255),
196     colour VARCHAR(255),
197     weight FLOAT,
198     warehouse_id VARCHAR(255)
199 );

```

16 11:05:01 CREATE TABLE sprint4db.products (id INT PRIMARY KEY, product_name... 0 row(s) affected 0.609 sec

A continuación cargamos los datos:

```

201 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
202 INTO TABLE products
203 FIELDS TERMINATED BY ','
204 ENCLOSED BY '"'
205 LINES TERMINATED BY '\n'
206 IGNORE 1 LINES;

```

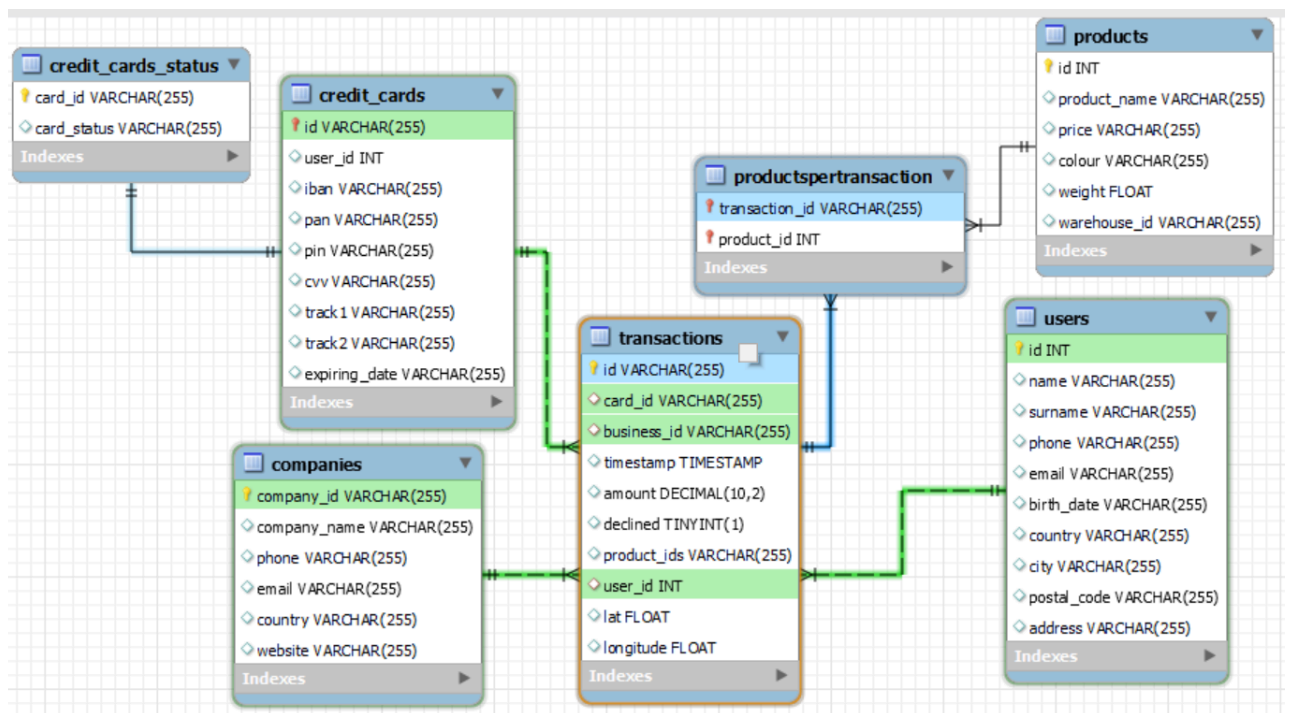
18 11:08:02 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/pr... 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0 0.016 sec

Ahora establecemos la relación entre tablas.

```
227      # Ara relacionem les taules
228      • ALTER TABLE productsperttransaction
229      ADD FOREIGN KEY (product_id)
230      REFERENCES products(id);
231
232      • ALTER TABLE productsperttransaction
233      ADD FOREIGN KEY (transaction_id)
234      REFERENCES transactions(id);
```

```
213 13:22:31 ALTER TABLE productsperttransaction ADD FOREIGN KEY (product_id) REFERENCE... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0 0.250 sec
214 13:22:33 ALTER TABLE productsperttransaction ADD FOREIGN KEY (transaction_id) REFEREN... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0 0.296 sec
```



Este es el resultado final:



Exercici 1

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

```
238 • SELECT product_id, product_name, count(*) as soldItems
239 FROM (SELECT product_id, product_name
240        FROM productsperttransaction
241        LEFT JOIN transactions
242        ON transaction_id = id
243        LEFT JOIN products
244        ON product_id = products.id
245        WHERE declined = 0) as soldProducts
246 GROUP BY product_id
247 ORDER BY soldItems DESC;
```

Result Grid   Filter Rows: <input type="text"/>			
	product_id	product_name	soldItems
▶	23	riverlands north	60
	67	Winterfell	59
	2	Tarly Stark	56
	43	duel	54
	17	skywalker ewok sith	54