Laura Cuscurita

SPRINT 4

Descripció

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:

```sql
CREATE DATABASE IF NOT EXISTS sprint4;

CREATE TABLE IF NOT EXISTS companies (
    company_id VARCHAR(100) PRIMARY KEY,
    company_name VARCHAR(255),
    phone VARCHAR(100),
    email VARCHAR(100),
    country VARCHAR(255),
    website VARCHAR(100)
);
```

En primer lloc, es crea el schema anomenat sprint 4 mitjançant el CREATE DATABASE.

Per a importar les taules, s'escull la taula *companies* per a importar-la mitjançant queries. S'utilitza la funció CREATE TABLE , i es descriuen totes les columnes,

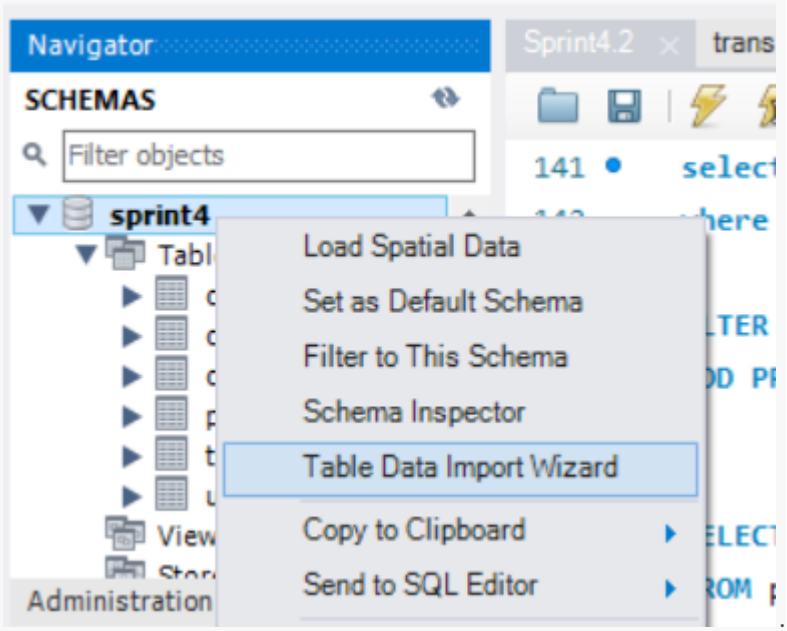
Per a procedir amb la importació, es fa ús de la funció SET GLOBAL, per activar la ruta d'origen. Si no es procedeix a aquest pas, donaria error.

```sql
SHOW VARIABLES LIKE 'secure_file_priv';
set global local_infile = 'ON';
```

Després d'establir les local_infiles on, es procedeix amb la següent funció, que el que fa és buscar la ruta d'origen del arxiu cvs:

```
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 ROWS;
```

Les demés taules es fan via import wizard per motius d'optimització de temps:

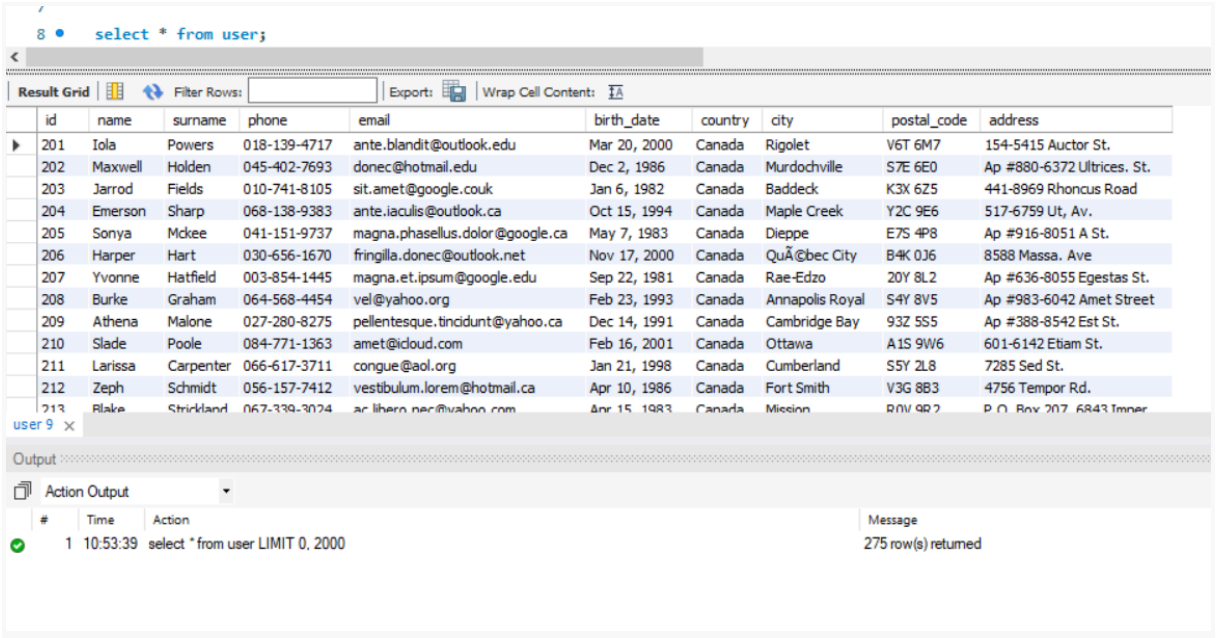


Un cop tenim totes les taules importades, es pot observar que hi han 3 taules *d'users*, totes elles amb la mateixa estructura. Per a optimitzar i millorar l'eficiència del model que es crearà, es procedeix a unir aquestes 3 taules mitjançant la consulta UNION.

Es pot fer ús d'aquesta funció ja que les yaules son completament identiques, sinó, no es podria procedir amb la query i s'haurien de buscar alternatives.

```
1 •   create table if not exists user
2     as select * from users_ca
3     union
4     select * from users_uk
5     union
6     select * from users_usa;
7
```

Un cop unides les taules, la taula única resultant anomenada *user*:



```
8 •   select * from user;
```

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

id	name	surname	phone	email	birth_date	country	city	postal_code	address
201	Iola	Powers	018-139-4717	ante.blandit@outlook.edu	Mar 20, 2000	Canada	Rigolet	V6T 6M7	154-5415 Auctor St.
202	Maxwell	Holden	045-402-7693	donec@hotmail.edu	Dec 2, 1986	Canada	Murdochville	S7E 6E0	Ap #880-6372 Ultrices. St.
203	Jarrod	Fields	010-741-8105	sit.amet@google.couk	Jan 6, 1982	Canada	Baddeck	K3X 6Z5	441-8969 Rhoncus Road
204	Emerson	Sharp	068-138-9383	ante.iaculis@outlook.ca	Oct 15, 1994	Canada	Maple Creek	Y2C 9E6	517-6759 Ut, Av.
205	Sonya	Mckee	041-151-9737	magna.phasellus.dolor@google.ca	May 7, 1983	Canada	Dieppe	E7S 4P8	Ap #916-8051 A St.
206	Harper	Hart	030-656-1670	fringilla.donec@outlook.net	Nov 17, 2000	Canada	QuÃ©bec City	B4K 0J6	8588 Massa. Ave
207	Yvonne	Hatfield	003-854-1445	magna.et.ipsum@google.edu	Sep 22, 1981	Canada	Rae-Edzo	20Y 8L2	Ap #636-8055 Egestas St.
208	Burke	Graham	064-568-4454	vel@yahoo.org	Feb 23, 1993	Canada	Annapolis Royal	S4Y 8V5	Ap #983-6042 Amet Street
209	Athena	Malone	027-280-8275	pellentesque.tincidunt@yahoo.ca	Dec 14, 1991	Canada	Cambridge Bay	93Z 5S5	Ap #388-8542 Est St.
210	Slade	Poole	084-771-1363	amet@icloud.com	Feb 16, 2001	Canada	Ottawa	A1S 9W6	601-6142 Etiam St.
211	Larissa	Carpenter	066-617-3711	congue@aol.org	Jan 21, 1998	Canada	Cumberland	S5Y 2L8	7285 Sed St.
212	Zeph	Schmidt	056-157-7412	vestibulum.lorem@hotmail.ca	Apr 10, 1986	Canada	Fort Smith	V3G 8B3	4756 Tempor Rd.
213	Blake	Strickland	067-339-3024	ac.libero.nec@yahoo.com	Apr 15, 1983	Canada	Mission	R0V 9R2	P.O. Box 207, 6843 Imper

user 9 ×

Output

Action Output

#	Time	Action	Message
1	10:53:39	select * from user LIMIT 0, 2000	275 row(s) returned

Com ja tenim una única taula de *users* , es borren les tres taules d'origen:

```
10 •   drop table users_ca;
11 •   drop table users_uk;
12 •   drop table users_usa;
```

A continuació, es presenten els canvis realitzats a les taules, ja que les taules d'origen tenien moltes columnes com a text, i per a millorar l'eficiència d'aquestes, les definim com a varchar(x).

Per altra banda, es defineixen les pk de cada taula, per a procedir més endavant amb les relacions.

```sql
alter table transactions
modify column id varchar(100);


alter table products
modify column id varchar(100);

alter table user
modify column id varchar(100);

alter table companies
modify column company_id varchar(100);

alter table credit_cards
modify column id varchar(100);

```

```sql
alter table transactions
modify column card_id varchar(100);

alter table transactions
modify column business_id varchar(100);

alter table transactions
modify column product_ids varchar(100);

alter table transactions
modify column user_id varchar(100);

```

```sql
alter table transactions
add primary key (id);

alter table companies
add primary key (company_id);
```

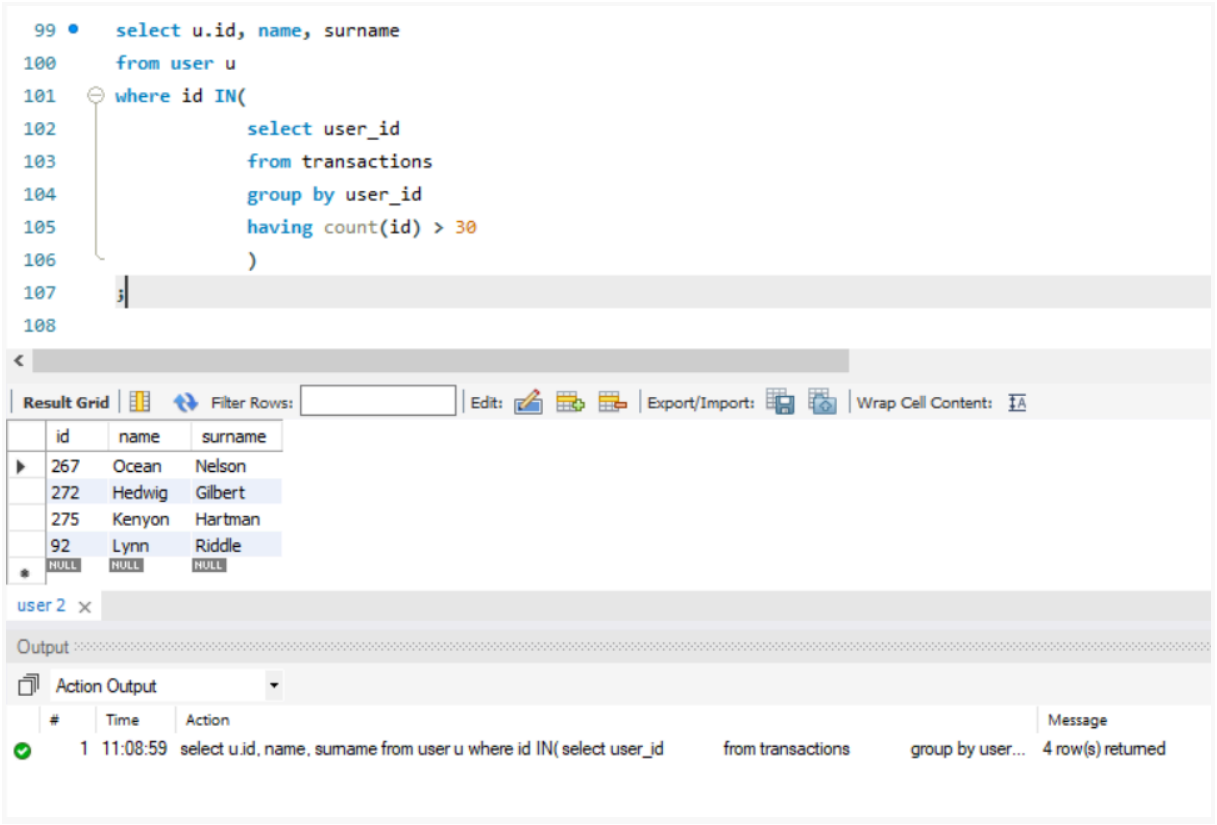
A continuació, es presenten l'assignació de relacions entre les taules. Utilitzant sempre la taula *transactions* com origen, i establint-ne les foreign key de les altres taules.

```sql
alter table transactions
add constraint fk_card
foreign key (card_id)
references credit_cards(id);

alter table transactions
add constraint fk_company
foreign key (business_id)
references companies(company_id);

alter table transactions
add constraint fk_user
foreign key (user_id)
references user(id);
```

- Exercici 1

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

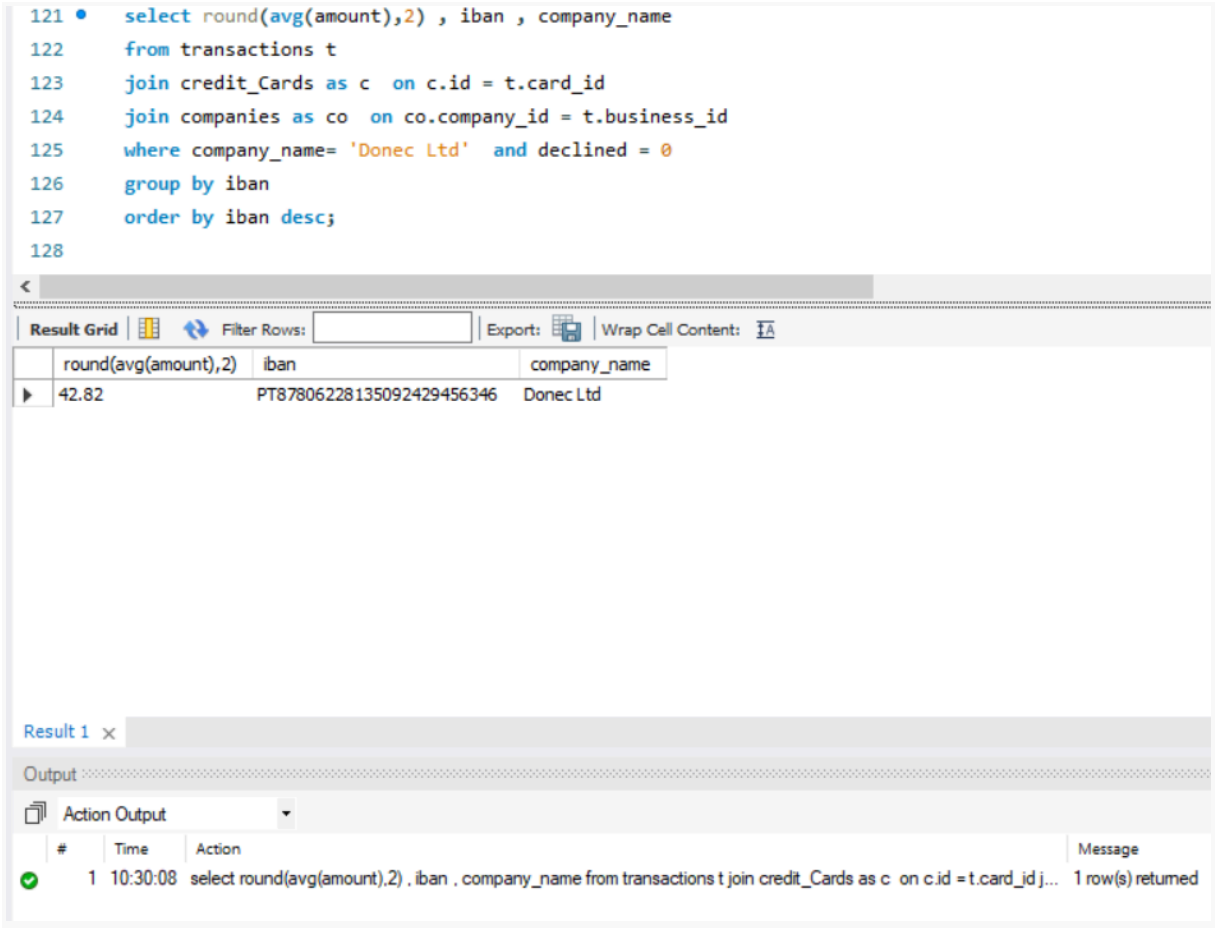
Es procedeix a fer un select amb la condició que el id contingui la següent informació : que els usuaris tinguin més de 30 transaccions. Podem veure que en total hi han 4 .



- Exercici 2

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

Es procedeix a seleccionar el la mitjana *d'amount* de la companyia, amb la funció AVG juntament amb la ROUND, ja que com tractem dades monetàries, sol es fan servir 2 decimals. S'utilitza la funció JOIN com a pont per connectar les respectives taules, i s'aplica la condició amb WHERE de que l'empresa s'anomeni " Donec Ltd" juntament amb declined =0 . S'entén que declined= 0 significa que la transacción ha sigut realitzada, i es pot considerar transacció.



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:

Aquest procediment crea una taula anomenat *card_status* que indica l'estat de les targetes basat en les últimes tres transaccions. Primer, es seleccionen les transaccions més recents per a cada *card_id* i els assigna un rang amb la funcio ROW_NUMBER() per a identificar-les fàcilment. Després, es compta el nombre de transaccions i el nombre de transaccions declinades per a cada targeta. Finalment, si hi ha menys de tres transaccions, la targeta es marca com 'Activa'; si totes les tres transaccions estan declinades, es marca com 'Inactiva'; en cas contrari, es manté 'Activa'. L'ús de ROW_NUMBER() és crucial per assegurar que només es consideren les tres transaccions més recents de cada targeta en aquesta anàlisi.

```sql
CREATE TABLE card_status AS
SELECT
    card_id,
    CASE
        WHEN COUNT(*) < 3 THEN 'Activa'
        WHEN SUM(declined) = 3 THEN 'Inactiva'
        ELSE 'Activa'
    END AS Status
FROM (
    SELECT
        card_id,
        declined,
        ROW_NUMBER() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS Rank_Transaccion
    FROM transactions
) AS Transacciones_Ordenadas
WHERE Rank_Transaccion <= 3
GROUP BY card_id;
```

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

card_id	Status
CcU-2938	Activa
CcU-2945	Activa
CcU-2952	Activa
CcU-2959	Activa
CcU-2966	Activa

card_status 41

Output

Action Output

#	Time	Action	Message
1	10:30:41	CREATE TABLE card_status AS SELECT card_id, CASE WHEN COUNT(*) < 3 THEN 'Activa' W...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
2	10:30:56	select * from card_status LIMIT 0, 2000	275 row(s) returned

Exercici 1

Quantes targetes estan actives?

Com es pregunta quantes estan actives, es realitza un count amb la condició de que estiguin en estat 'Activa':

```sql
select count(*) from card_status
where status = 'Activa';
```

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

count(*)
275

Result 44

Output

Action Output

#	Time	Action	Message
1	10:35:20	select count(*) from card_status where status = 'Activa' LIMIT 0, 2000	1 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.

Per a la realització d'aquest exercici, hi han diferents mètodes per arribar al resultat esperat. Es poden utilitzar les funcions FIND_IN_SET o amb la funció CAST i JOIN. No obstant això, com l'enunciat et demana que es creï una taula, s'ha obtat per crear-la des de 0 i introduirla al model.

En primer lloc, es fa un pas previ per a poder continuar amb la creació (ja que em donava error 3780). Es mira com està composta la taula *products*, i s'opta per cambiar la composició del id (estava en format VARCHAR(100) i es canvia a INT). AMB aquest pas previ, es podrà procedir amb l'exericci sense complicacions.

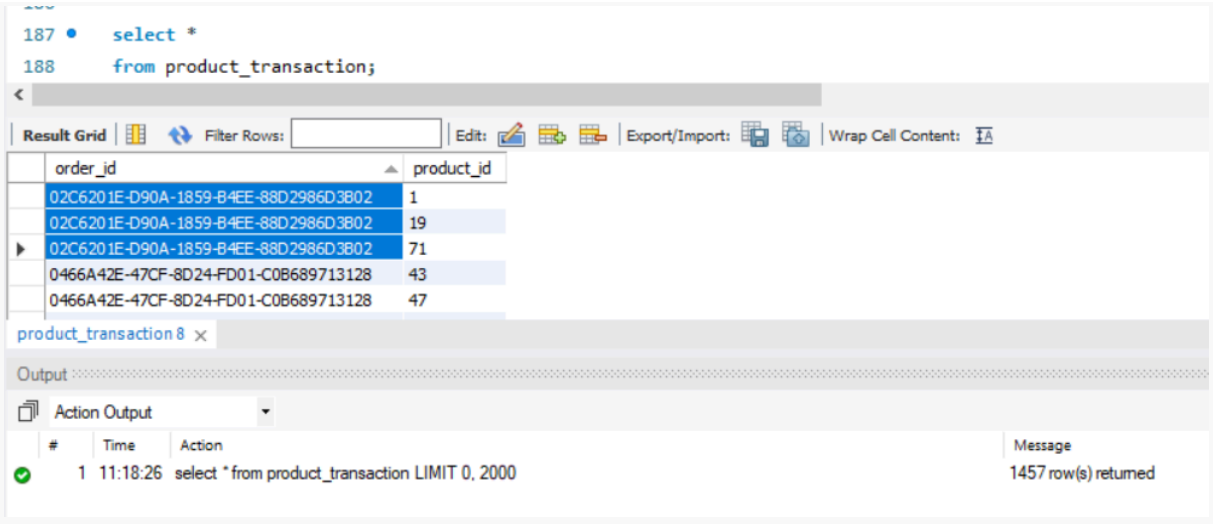
Seguidament, es crea la taula *product_transaction*, utilitzant la funció CAST per a identificar cada número separat per comes de la columna *product_ids*.

S'utilitza la funció join , per a descomposar cada número del 1 al 10 (per a evitar possibles errors).

S'aplica la condició CHAR_LENGHT per a calcular la el número de comes i aixi obtenir la quantitat total de productes.

```
ALTER TABLE products MODIFY COLUMN id INT; #como era VAR(100) cambio a INT el id para q no me de error

CREATE TABLE product_transaction AS
SELECT
    t.id AS order_id,
    CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', numbers.n), ',', -1) AS UNSIGNED) AS product_id
FROM
    transactions t
JOIN (
    SELECT 1 AS n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5
    UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9 UNION ALL SELECT 10
) numbers
ON CHAR_LENGTH(t.product_ids) - CHAR_LENGTH(REPLACE(t.product_ids, ',', '')) >= numbers.n - 1
ORDER BY order_id, product_id;   #utilizo la funcion cast para descomponer los numeros separados por comas, y l
```

Per verificar que la taula ens retorna lo desitjat, es procedeix a verificar els resultats d'aquesta. S'observa que ara hi han ids repetits, pero cada un assignat a un id de producte:



Un cop creada la taula, es procedeix a establir les PK i les FK per a tal de relacionar les taules.

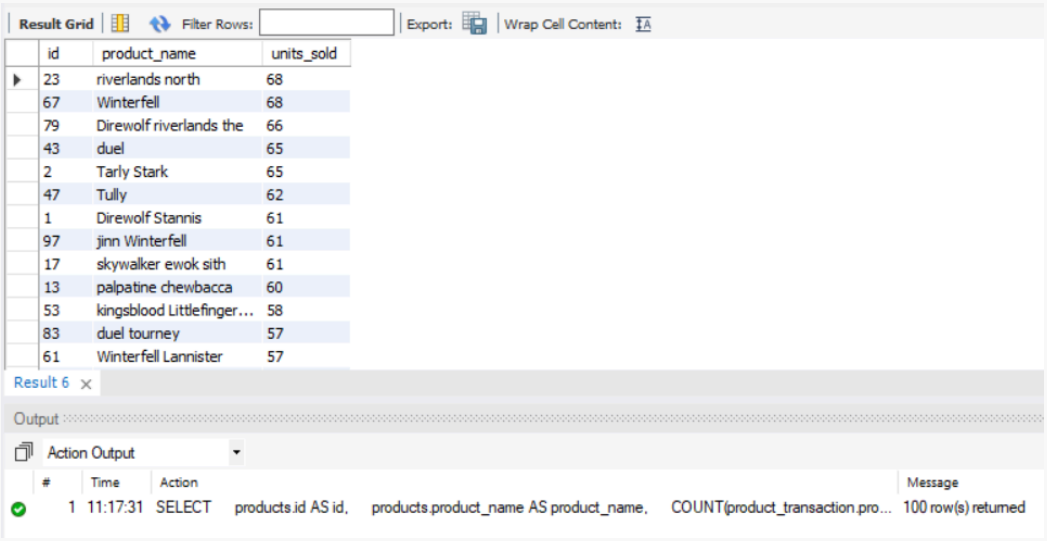
```
ALTER TABLE product_transaction
MODIFY COLUMN product_id INT,
ADD PRIMARY KEY (order_id, product_id),
ADD FOREIGN KEY (order_id) REFERENCES transactions(id),
ADD FOREIGN KEY (product_id) REFERENCES products(id);
```

Per últim, un cop creada la taula i establertes les PK'S i FK'S, es procedeix a escriure el comando que ens donarà el resultat esperat. Es fa una selecció del id del producte, el nom del producte i un count de les unitats venudes de la taula products. S'utilitza el join per a unir-la amb la taula creada prèviament.

El resultat es mostra el id del producte, el nom del producte, i la quantitat de vegades que s'ha venut cada un.

Laura Cuscurita

```sql
SELECT
    products.id AS id,
    products.product_name AS product_name,
    COUNT(product_transaction.product_id) AS units_sold
FROM
    products
LEFT JOIN
    product_transaction ON products.id = product_transaction.product_id
GROUP BY
    products.id, products.product_name
ORDER BY
    units_sold DESC, product_name;
```



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

id	product_name	units_sold
23	riverlands north	68
67	Winterfell	68
79	Direwolf riverlands the	66
43	duel	65
2	Tarly Stark	65
47	Tully	62
1	Direwolf Stannis	61
97	jinn Winterfell	61
17	skywalker ewok sith	61
13	palpatine chewbacca	60
53	kingsblood Littlefinger...	58
83	duel tourney	57
61	Winterfell Lannister	57

Result 6

Output

Action Output

#	Time	Action	Message
1	11:17:31	SELECT products.id AS id, products.product_name AS product_name, COUNT(product_transaction.pro...	100 row(s) returned