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:

Generarem en primer lloc un nou SCHEMA anomenat *sprint4*:

```
CREATE DATABASE Sprint4;
```

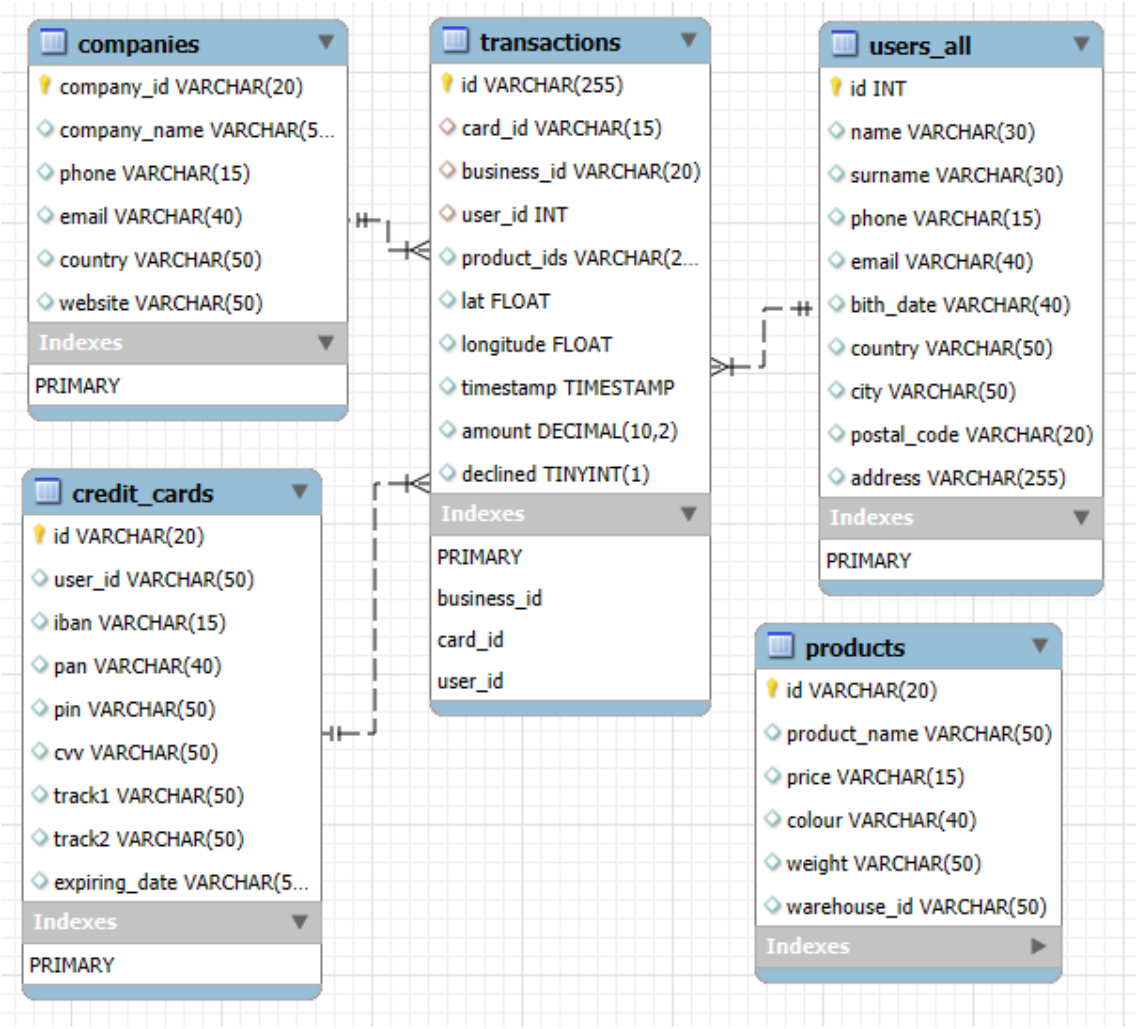
Les taules *users_ca, users_uk* i *users_usa* es poden unir en una sola taula *users_all*. Per això importarem en sèrie els .csv de les tres taules des de la carpeta *secure_file_priv* que en el cas del meu ordinador personal correspon a l'adreça *C:/ProgramData/MySQL/MySQL Server 8.0/Uploads* obtinguda amb el codi SHOW VARIABLES LIKE "secure_file_priv". També crearem de forma anàloga quatre taules més corresponents als fitxers *companies, transactions, credit_cards* i *products* i importarem de la mateixa forma les dades corresponents.

```
CREATE TABLE `users_all` (
  `id` int NOT NULL,
  `name` varchar(30) DEFAULT NULL,
  `surname` varchar(30) DEFAULT NULL,
  `phone` varchar(15) DEFAULT NULL,
  `email` varchar(40) DEFAULT NULL,
  `bith_date` varchar(40) DEFAULT NULL,
  `country` varchar(50) DEFAULT NULL,
  `city` varchar(50) DEFAULT NULL,
  `postal_code` varchar(20) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
;

SHOW VARIABLES LIKE "secure_file_priv";

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

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

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

La relació de les taules és la següent. La taula *products* a priori no es pot relacionar directament amb la taula *transactions* ja que la columna *product_ids* conté els valors de diversos id's de productes en format .csv per cada transacció (els separarem a l'exercici 3).



- Exercici 1

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

Un cop obtinguda aquesta taula podem trobar els usuaris amb més de 30 transaccions amb un JOIN amb la taula transactions i un GROUP BY. Mostrarem el user_id i els seus noms i cognoms:

```
142 •  SELECT name, surname, users_all.id, count(transactions.id) nTransaction FROM users_all
143    JOIN transactions ON users_all.id = transactions.user_id
144    GROUP BY users_all.name, users_all.surname, users_all.id
145    HAVING nTransaction > 30
146    ORDER BY nTransaction DESC;
147
```

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

name	surname	id	nTransaction
Hedwig	Gilbert	272	76
Ocean	Nelson	267	52
Kenyon	Hartman	275	48
Lynn	Riddle	92	39

Result 138

Output

Action Output

#	Time	Action	Message
34	20:56:12	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL S...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0
35	20:56:25	CREATE TABLE `transactions` (`id` varchar(255) NOT ...	0 row(s) affected, 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future release.
36	20:56:29	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL S...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0
37	20:56:38	SELECT name, surname, users_all.id, count(transactions.i...	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.

Per trobar la mitjana per IBAN de cada targeta hem fet un JOIN amb les taules transaction (per poder obtenir l'amount), companies (per poder filtrar pel nom 'Donec Ltd') i credit_cards (per poder agrupar per iban):

```
SELECT credit_cards.iban, ROUND(avg(amount),2) avgSales FROM transactions
JOIN companies ON companies.company_id = transactions.business_id
JOIN credit_cards ON credit_cards.id = transactions.card_id
WHERE companies.company_name = 'Donec Ltd'
GROUP BY  credit_cards.iban;
```

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

iban	avgSales
PT87806228135092429456346	203.72

Result 96

Output

Action Output

#	Time	Action	Message
1	11:39:24	SELECT credit_cards.iban, avg(amount) avgSales FROM transactions JOI...	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?

Creem una taula credit_card_state que retornarà una columna Card_Activity amb valor 1 si la targeta es troba activa (és a dir, que alguna de les tres últimes transaccions no s'han denegat) o 0 si es troba inactiva (el cas contrari). Utilitzarem una sèrie de subquerys. Per poder obtenir les transaccions de cada targeta ordenades temporalment per timestamp la subquery és la següent:

```
SELECT *, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS n
FROM transactions
```

id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude	n
AD85A78A-8829-5746-93A0-8B7A792EBC18	CcU-2938	b-2362	2022-03-12 09:23:10	263.59	0	2, 3, 29	92	25.5502450688	-142.6244872192	1
F1A598A2-86C5-50A9-F1CE-FB1D69866C39	CcU-2938	b-2362	2022-03-09 20:53:59	229.65	0	79, 73	84	-65.1240208384	31.0336632832	2
55166D02-D74C-6A63-6C54-867846769B4	CcU-2938	b-2362	2022-02-24 11:01:42	186.12	0	97, 89, 83	89	-31.5170302976	-168.5745020928	3
6575E457-EF8B-B50A-AD85-68ED6FE98BE1	CcU-2938	b-2362	2021-10-24 01:29:53	297.16	0	19	92	-79.988224512	-45.7529433088	4
C9185110-96F7-193C-D4B4-D1086A63744A	CcU-2938	b-2362	2021-10-17 03:52:48	390.67	0	53, 89, 23	86	-39.3218968576	-10.4055994368	5
E6873589-5339-4189-E984-9F4B54C844A4	CcU-2938	b-2362	2021-09-28 02:24:34	199.83	0	37, 29	88	-70.210527232	-46.1508482048	6
18B2472B-DA8B-36F7-B0F2-7DDA688B73D1	CcU-2938	b-2362	2021-09-24 08:33:44	391.38	0	59, 23	87	-41.772207616	98.2897368064	7
C3516B1E-4923-B2DE-E6E2-B07548480CCE	CcU-2938	b-2362	2021-09-18 00:31:49	247.13	0	17	91	27.7145867264	133.7281160192	8

Result 107

Output

Action Output

#	Time	Action	Message
1	12:15:07	SELECT *, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY t...	587 row(s) returned

La següent subquery filtrarà les últimes tres transaccions que s'han ordenat per timestamp per cada targeta i les agruparà segons si han estat declinades o no contant el nombre de vegades que han o no han estat declinades. Si declined=1 i nDeclined es=3, aleshores la targeta es trobarà inactiva:

```
SELECT card_id, declined, count(*) nDeclined
  FROM (SELECT *, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS n
        FROM transactions) AS b1 WHERE n <= 3
        GROUP BY card_id, declined
```

card_id	declined	nDeclined
CcU-4835	0	1
CcU-4842	0	1
CcU-4849	0	3
CcU-4856	0	1
CcU-2945	1	1
CcU-2952	1	1
CcU-2966	1	1
CcU-2973	1	1

13 12:09:51 SELECT card_id, declined, count(*) nDeclined FROM (SELECT *, ROW_... 359 row(s) returned

A continuació farem servir l'anterior subquery a dins d'una QUERY CASE WHEN que generarà la columna Card_Activity a la nova taula:

```
45  CREATE TABLE IF NOT EXISTS credit_card_state SELECT card_id,
46
47      CASE WHEN card_id IN (SELECT card_id FROM (SELECT card_id, declined, count(*) nDeclined
48                                                 FROM (SELECT *, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS n
49                                                       FROM transactions) AS a1 WHERE n <= 3
50                                                       GROUP BY card_id, declined
51                                                       HAVING (declined=1 and nDeclined>2)) AS a2) THEN 0
52
53      WHEN card_id NOT IN (SELECT card_id FROM (SELECT card_id, declined, count(*) nDeclined
54                                                 FROM (SELECT *, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS n
55                                                       FROM transactions) AS b1 WHERE n <= 3
56                                                       GROUP BY card_id, declined
57                                                       HAVING (declined=1 and nDeclined>2)) AS b2) THEN 1
58      END AS Card_Activity
59      FROM transactions;
```

card_id	Card_Activity
CcU-2938	1
CcU-2945	1
CcU-2952	1
CcU-2959	1
CcU-2966	1
CcU-2973	1
CcU-2980	1
CcU-2987	1
CcU-2994	1

Action Output

#	Time	Action	Message
1	11:57:42	CREATE TABLE IF NOT EXISTS credit_card_state SELECT card_id, CAS...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
2	11:58:07	SELECT * FROM credit_card_state LIMIT 0, 50000	587 row(s) returned

Repassant el resultat de la següent query es pot veure que no hi ha cap targeta que es trobi inactiva (Card_Activity=0 => inactiva).

```
177  SELECT Card_Activity, count(*) FROM credit_card_state
178  GROUP BY Card_Activity
179  HAVING Card_Activity=0;
180
```

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

Card_Activity	count(*)

Result 156

Output

Action Output

#	Time	Action	Message
1	11:15:15	SELECT Card_Activity, count(*) FROM credit_card_state GROUP BY Card_...	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:

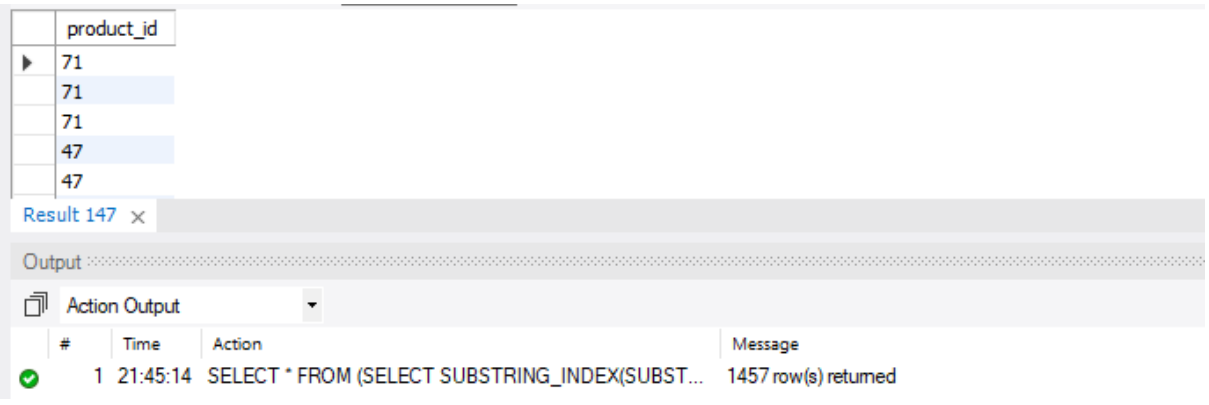
A la taula transactions hi ha una columna products_id on hi consten els diversos productes que s'han ordenat a aquella transacció en format csv. Caldrà separar aquests valors en files individuals per poder fer un JOIN amb la taula products en funció del seu id. Per resoldre aquest problema m'he basat en la següent pregunta d'stack overflow: https://stackoverflow.com/questions/17942508/sql-split-values-to-multiple-rows

Primer cal crear una taula *numbers* que contingui 1:n sent n el màxim nombre de camps a separar. En aquest cas agafarem n=6:

```
CREATE TABLE numbers (
  n INT PRIMARY KEY);

INSERT INTO numbers VALUES (1),(2),(3),(4),(5),(6);
```

La següent subquery ens permet obtenir cada producte separat en una fila individual. La funció SUBSTRING_INDEX permet destriar els k primers substrings separats en aquest cas per una ',' . A continuació el codi recorrerà amb una recurrència en funció del nombre d'elements continguts a cada valor de la columna *product_ids* i anirà restant cada element (fins a sis elements) (CHAR_LENGTH(tra…) - CHAR_LENGTH(REP...) per poder accedir en cada cas a l'últim element de cada valor: SUBSTRING_INDEX((SUBSTRING…), ' , ' ,-1):

```
(SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(transactions.product_ids, ',',1), ',', -1) product_id FROM numbers
 INNER JOIN transactions
 ON CHAR_LENGTH(transactions.product_ids)
 -CHAR_LENGTH(REPLACE(transactions.product_ids, ',', ''))>=numbers.n-1
 ORDER BY
  id, n) AS t
```



Caldrà aleshores crear una taula *prodplustrans* on farem un JOIN de la subquery anterior amb la taula products. Reanomenarem la columa id de la taula transaccions com a t_id per evitar conflictes d'homonímia en fer el JOIN. També farem DROP a la columna products.id ja que apareix com a duplicat de la columna product_id.

```sql
CREATE TABLE IF NOT EXISTS prodplustrans
SELECT * FROM (SELECT tr.id t_id, card_id, business_id, timestamp, amount, declined, SUBSTRING_INDEX(SUBSTRING_INDEX(tr.product_ids, ',',1), ',', -1)
              product_id, user_id, lat, longitude FROM numbers
              INNER JOIN transactions tr
              ON CHAR_LENGTH(tr.product_ids)
              -CHAR_LENGTH(REPLACE(tr.product_ids, ',', ''))>=numbers.n-1
              ORDER BY
               id, n) AS t
          JOIN products
          ON t.product_id = products.id;
```

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

t_id	card_id	business_id	timestamp	amount	declined	product_id	user_id	lat	longitude	product_name	price	colour	weight	warehou
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71	92	81.9185	-12.5276	Tully Dorne	$103.73	#424242	2.7	WH--66
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71	92	81.9185	-12.5276	Tully Dorne	$103.73	#424242	2.7	WH--66
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71	92	81.9185	-12.5276	Tully Dorne	$103.73	#424242	2.7	WH--66
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47	170	-43.9695	-117.525	Tully	$82.15	#919191	2.7	WH--42
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47	170	-43.9695	-117.525	Tully	$82.15	#919191	2.7	WH--42

prodplustrans 151

Output

Action Output

#	Time	Action	Message
1	21:59:53	CREATE TABLE IF NOT EXISTS prodplustrans SELECT * FROM (SELECT...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
2	21:59:53	ALTER TABLE prodplustrans DROP COLUMN id	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	21:59:57	SELECT * FROM prodplustrans LIMIT 0, 50000	1457 row(s) returned

Exercici 1

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

Per últim, un cop creada la taula agruparem per product_id i product_name contant el nombre de productes venuts i els ordenarem per nombre de vendes:

```sql
SELECT count(t_id) nSales ,product_id, product_name FROM prodplustrans
GROUP BY product_id, product_name
ORDER BY nSales DESC;
```

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

nSales	product_id	product_name
80	7	north of Casterly
77	23	riverlands north
74	61	Winterfell Lannister
72	47	Tully
72	79	Direwolf riverlands the
68	43	duel

Result 153

Output

Action Output

#	Time	Action	Message
1	22:09:49	SELECT count(t_id) nSales ,product_id, product_name FROM prodplustran...	26 row(s) returned