

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:

```
1 • CREATE DATABASE IF NOT EXISTS sprint4;
2
3 • CREATE TABLE IF NOT EXISTS companies (
4     company_id VARCHAR(100) PRIMARY KEY,
5     company_name VARCHAR(255),
6     phone VARCHAR(100),
7     email VARCHAR(100),
8     country VARCHAR(255),
9     website VARCHAR(100)
10 );
```

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,

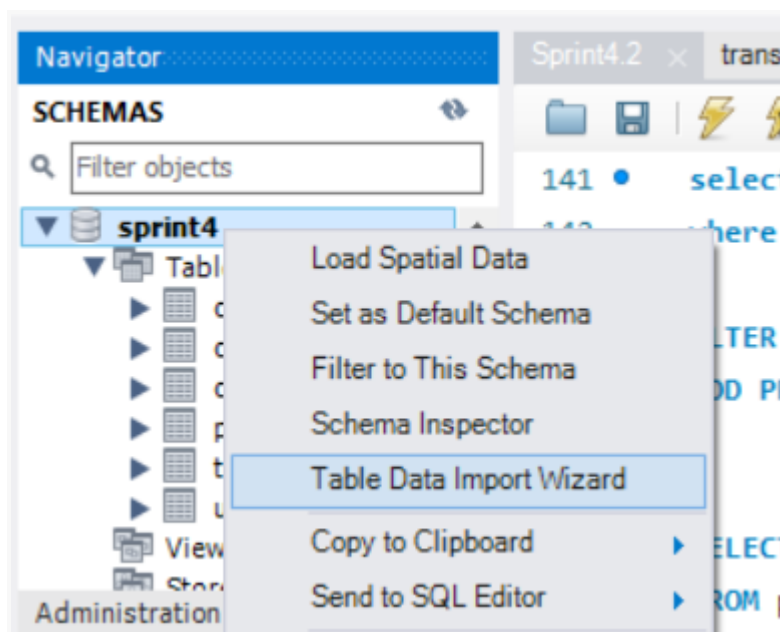
Per a procedir amb la importació, es fa ús de la funció SET GLOBAL, per activar la ruta d'origen. Si no es procedeix a aquest pas, donaria error.

```
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 taules 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;
```

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.co.uk	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	QuÃbec 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@idoud.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 Tempus Rd.
213	Blake	Strickland	067-339-3074	ac.literna@yahoo.com	Apr 15, 1983	Canada	Mission	R0V 9R2	P.O. Box 207, 6R43 Timber

user 9 x

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.

```
14 • alter table transactions
15     modify column id varchar(100);
16
17
18 • alter table products
19     modify column id varchar(100);
20
21 • alter table user
22     modify column id varchar(100);
23
24 • alter table companies
25     modify column company_id varchar(100);
26
27 • alter table credit_cards
28     modify column id varchar(100);
29
```

```
30 • alter table transactions
31     modify column card_id varchar(100);
32
33 • alter table transactions
34     modify column business_id varchar(100);
35
36 • alter table transactions
37     modify column product_ids varchar(100);
38
39 • alter table transactions
40     modify column user_id varchar(100);
41
```

```
• 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.

- 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 .

```
99 • select u.id, name, surname
100   from user u
101  where id IN(
102      select user_id
103      from transactions
104      group by user_id
105      having count(id) > 30
106  )
107 ;
108
```

id	name	surname
267	Ocean	Nelson
272	Hedwig	Gilbert
275	Kenyon	Hartman
92	Lynn	Riddle
*	NULL	NULL

user 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	11:08:59	select u.id, name, surname from user u where id IN(select user_id from transactions group by user...	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.

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 transacció ha sigut realitzada, i es pot considerar transacció.

```

121 • select round(avg(amount),2) , iban , company_name
122 from transactions t
123 join credit_Cards as c on c.id = t.card_id
124 join companies as co on co.company_id = t.business_id
125 where company_name= 'Donec Ltd' and declined = 0
126 group by iban
127 order by iban desc;
128

```

round(avg(amount),2)	iban	company_name
42.82	PT87806228135092429456346	Donec Ltd

Result 1 x

Output

Action Output

#	Time	Action	Message
✓ 1	10:30:08	select round(avg(amount),2) , iban , company_name from transactions t join credit_Cards as c on c.id = t.card_id j...	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:

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 funció 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.

```

129 • CREATE TABLE card_status AS
130 SELECT
131     card_id,
132     CASE
133         WHEN COUNT(*) < 3 THEN 'Activa'
134         WHEN SUM(declined) = 3 THEN 'Inactiva'
135         ELSE 'Activa'
136     END AS Status
137 FROM (
138     SELECT
139         card_id,
140         declined,
141         ROW_NUMBER() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS Rank_Transaccion
142     FROM transactions
143 ) AS Transacciones_Ordenadas
144 WHERE Rank_Transaccion <= 3
145 GROUP BY card_id;

```

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

card_status 41 x

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':

```

147 • select count(*) from card_status
148 where status = 'Activa';

```

count(*)
275

Result 44 x

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

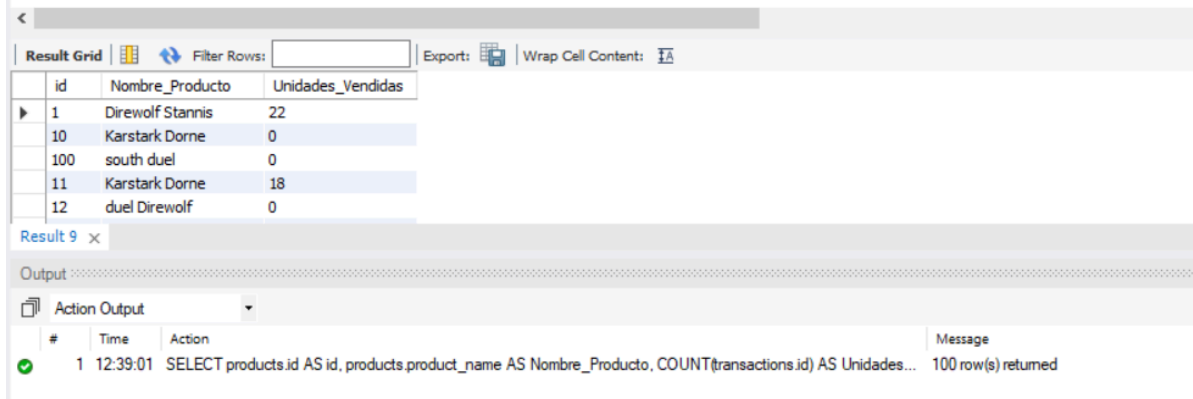
Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

Exercici 1

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

Per a la realització d'aquest exercici, es procedeix a fer un COUNT de les transaccions, i s'utilitza un LEFT JOIN amb la taula *transactions*. Per a poder fer una recerca dels *products_ids*, s'utilitza la funció FIND_IN_SET la qual permet buscar dins la columna *products_ids* de la taula *transactions*, i veure si coincideix amb el *id* de la taula *products*. Aquesta funció s'utilitza quan hi han més d'un valor per columna, i estan separats per una coma.

```
148 • SELECT products.id AS id, products.product_name AS Nombre_Producto, COUNT(transactions.id) AS Unidades_Vendidas
149 FROM products
150 LEFT JOIN transactions ON FIND_IN_SET(products.id, transactions.product_ids)
151 GROUP BY id, Nombre_Producto
152
153
154
```



The screenshot shows a database query interface. At the top, the SQL query is displayed with line numbers 148 to 154. Below the query, there is a 'Result Grid' section with a table showing the results of the query. The table has three columns: 'id', 'Nombre_Producto', and 'Unidades_Vendidas'. The results are as follows:

id	Nombre_Producto	Unidades_Vendidas
1	Direwolf Stannis	22
10	Karstark Dorne	0
100	south duel	0
11	Karstark Dorne	18
12	duel Direwolf	0

Below the result grid, there is an 'Output' section with a table showing the execution details of the query. The table has three columns: '#', 'Time', and 'Action'. The results are as follows:

#	Time	Action
1	12:39:01	SELECT products.id AS id, products.product_name AS Nombre_Producto, COUNT(transactions.id) AS Unidades_Vendidas...