

Tasca S4.01. Creació de Base de Dades

Nivell 1

Exercici 1

En esta tarea, se decidió añadir a la base de datos todos los archivos CSV excepto product.csv. Todas las tablas creadas tenían las columnas iniciales de tipo VARCHAR, para evitar problemas en la lectura, y posteriormente se modificaron los tipos de valores de las columnas dependiendo de la tabla. También se añadieron las primary keys y foreign keys más adelante. Las primeras tablas creadas fueron american_users y european_users, que posteriormente se combinaron en la tabla users.

The screenshot shows the MySQL Workbench interface. The top window displays the SQL query used to create the 'american_users' table and load data from a CSV file. The bottom window shows the resulting data grid and the log of executed actions.

```
6 • CREATE TABLE IF NOT EXISTS american_users (
7     id VARCHAR(255) NULL,
8     name VARCHAR(255) NULL,
9     surname VARCHAR(255) NULL,
10    phone VARCHAR(255) NULL,
11    email VARCHAR(255) NULL,
12    birth_date VARCHAR(255) NULL,
13    country VARCHAR(255) NULL,
14    city VARCHAR(255) NULL,
15    postal_code VARCHAR(255) NULL,
16    address VARCHAR(255) NULL
17 );
18
19 -- Añadimos los datos a la tabla
20 • LOAD DATA
21   INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\american_users.csv"
22   INTO TABLE american_users
23   FIELDS TERMINATED BY ','
24   ENCLOSED BY ""
25   IGNORE 1 ROWS;
26
27 -- Hacemos cambios a la tabla american_users;
28 • ALTER TABLE american_users
29   MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;
30
31 • SELECT * FROM american_users;
32
33 /*ALTER TABLE american_users*/
```

	id	name	surname	phone	email	birth_date	country	city
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia
	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix

erican_users 1 X

Output:

#	Time	Action	Message	Duration / Fetch
4	19:11:32	ALTER TABLE american_users MODIFY COLUM...	Error Code: 1068. Multiple primary key defined	0.000 sec
5	19:15:20	SELECT * FROM american_users	1010 row(s) returned	0.000 sec / 0.000 sec

```

39 • CREATE TABLE IF NOT EXISTS european_users (
40     id VARCHAR(255) NULL,
41     name VARCHAR(255) NULL,
42     surname VARCHAR(255) NULL,
43     phone VARCHAR(255) NULL,
44     email VARCHAR(255) NULL,
45     birth_date VARCHAR(255) NULL,
46     country VARCHAR(255) NULL,
47     city VARCHAR(255) NULL,
48     postal_code VARCHAR(255) NULL,
49     address VARCHAR(255) NULL
50 );
51
52 -- Añadimos los datos a la tabla
53 • LOAD DATA
54 INFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\S4\\\\european_users.csv"
55 INTO TABLE european_users
56 FIELDS TERMINATED BY ','
57 ENCLOSED BY ""
58 IGNORE 1 ROWS;
59
60 -- Hacemos cambios a la tabla european_users;
61 • ALTER TABLE european_users
62     MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;
63
64 • SELECT * FROM european_users;

```

Result Grid

	id	name	surname	phone	email	birth_date	country	city
▶	151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	Jul 2, 1980	United Kingdom	London
	152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google.edu	Sep 30, 1979	United Kingdom	Birmingham
	153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	London
	154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Manche

european_users 2 X

Output:

Action Output

#	Time	Action	Message	Duration / Fetch
✖	8	19:18:19 ALTER TABLE european_users MODIFY COLUM...	Error Code: 1068. Multiple primary key defined	0.000 sec
✓	9	19:18:21 SELECT * FROM european_users	3990 row(s) returned	0.000 sec / 0.031 sec

66 -- Confirmamos que los id de las tablas american_user and european_user son diferentes
67 • SELECT *
68 FROM american_users
69 WHERE id IN (SELECT id FROM european_users);
70
71 -- Combinamos las dos tablas en la tabla users
72 • CREATE TABLE IF NOT EXISTS users AS
73 SELECT * FROM american_users
74 UNION
75 SELECT * FROM european_users;
76
77 -- Hacemos cambios a la tabla users;
78 • ALTER TABLE users
79 MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;
80
81 • SELECT * FROM users;

	id	name	surname	phone	email	birth_date	country	city
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia
	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix
	5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Philadelphia
	6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	San Jose
	7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Chicago
	8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	New York
	9	Mannix	Mcclain	(590) 883-2184	aliquam.nisl@outlook.com	Jan 24, 1987	United States	San Antonio
	10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	San Jose
	11	Joan	Baird	(981) 429-8106	et@outlook.net	Feb 25, 1990	United States	Los Angeles
	12	Benedict	Wheeler	1-515-824-2855	tincidunt.donec.vitae@hotmail.co.uk	Aug 6, 1999	United States	Phoenix
	13	Allegra	Stanton	1-927-753-6488	proin.eget@protonmail.ca	May 19, 1990	United States	New York
	14	Sara	Flynn	1-311-646-9333	integer@outlook.net	Dec 27, 1988	United States	Los Angeles
	15	Noelani	Patrick	1-723-488-5894	sem.magna@google.com	Sep 17, 1993	United States	Los Angeles
	16	Eric	Roth	1-218-549-8253	lorem.sit@yahoo.net	Sep 7, 1988	United States	San Diego

users 3 × Apply Revert

Output

Action Output		
#	Time	Action
9	19:18:21	SELECT * FROM european_users
10	19:19:08	SELECT * FROM users

Añadimos el resto de las tablas: credit_cards, companies y transactions.

```
83 -- Creamos la tabla credit_cards
84 • CREATE TABLE IF NOT EXISTS credit_cards (
85     id VARCHAR(255) NULL,
86     user_id VARCHAR(255) NULL,
87     iban VARCHAR(255) NULL,
88     pan VARCHAR(255) NULL,
89     pin VARCHAR(255) NULL,
90     cvv VARCHAR(255) NULL,
91     track1 VARCHAR(255) NULL,
92     track2 VARCHAR(255) NULL,
93     expiring_date VARCHAR(255) NULL
94 );
95
96 -- Añadimos los datos a la tabla
97 • LOAD DATA
98    INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\credit_cards.csv"
99    INTO TABLE credit_cards
100   FIELDS TERMINATED BY ','
101   IGNORE 1 ROWS;
102
103 -- Hacemos cambios a la tabla credit_cards;
104 • ALTER TABLE credit_cards
105   MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
106   MODIFY COLUMN user_id INT;
107
108 -- Añadimos foreign key con users
109 • ALTER TABLE credit_cards
110   ADD CONSTRAINT fk_user
111     FOREIGN KEY (user_id)
112       REFERENCES users(id);
113
114 • SELECT * FROM credit_cards;
115
116 -- Creamos la tabla companies
```

Output

Action Output				
#	Time	Action	Message	Duration / Fetch
13	19:20:39	ALTER TABLE credit_cards MODIFY COLUMN i...	Error Code: 1068. Multiple primary key defined	0.000 sec
14	19:20:42	ALTER TABLE credit_cards ADD CONSTRAINT ...	Error Code: 1826. Duplicate foreign key constraint...	0.000 sec

```

89     pin VARCHAR(255) NULL,
90     cvv VARCHAR(255) NULL,
91     track1 VARCHAR(255) NULL,
92     track2 VARCHAR(255) NULL,
93     expiring_date VARCHAR(255) NULL
94   );
95
96 -- Añadimos los datos a la tabla
97 • LOAD DATA
98   INFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\S4\\\\credit_cards.csv"
99   INTO TABLE credit_cards
100  FIELDS TERMINATED BY ','
101  IGNORE 1 ROWS;
102
103 -- Hacemos cambios a la tabla credit_cards;
104 • ALTER TABLE credit_cards
105   MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
106   MODIFY COLUMN user_id INT;
107
108 -- Añadimos foreign key con users
109 • ALTER TABLE credit_cards
110   ADD CONSTRAINT fk_user
111     FOREIGN KEY (user_id)
112     REFERENCES users(id);
113
114 • SELECT * FROM credit_cards;

```

Result Grid

	id	user_id	iban	pan	pin	cvv	track1
▶	CcS-4857	276	XX4857591835292505850771	2314242385113924	1819	467	%B2314242385113924^LWCBDLWCE
▶	CcS-4858	277	XX8581768137002436094025	6582720299715533	3964	817	%B6582720299715533^TIQMVTIQMV
▶	CcS-4859	278	XX7826930491423553609370	8861684536289642	4983	277	%B8861684536289642^COFBGDCOFB
▶	CcS-4860	279	XX5559590368835304645299	2481155515498459	6876	661	%B2481155515498459^TIUTUTIUTL

credit_cards 4 ×

Output :

Action Output

#	Time	Action	Message	Duration / Fetch
✖	14	19:20:42	ALTER TABLE credit_cards ADD CONSTRAINT ... Error Code: 1826. Duplicate foreign key constraint...	0.000 sec
✓	15	19:21:23	SELECT * FROM credit_cards 5000 row(s) returned	0.000 sec / 0.032 sec

```

116  -- Creamos la tabla companies
117 • CREATE TABLE IF NOT EXISTS companies (
118      company_id VARCHAR(255) NULL,
119      company_name VARCHAR(255) NULL,
120      phone VARCHAR(255) NULL,
121      email VARCHAR(255) NULL,
122      country VARCHAR(255) NULL,
123      website VARCHAR(255) NULL
124 );
125
126  -- Añadimos los datos a la tabla
127 • LOAD DATA
128    INFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\S4\\\\companies.csv"
129    INTO TABLE companies
130    FIELDS TERMINATED BY ','
131    IGNORE 1 ROWS;
132
133  -- Hacemos cambios a la tabla companies;
134 • ALTER TABLE companies
135    MODIFY COLUMN company_id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL;
136
137 • SELECT * FROM companies;
138

```

Result Grid | Filter Rows: | Edit: Export/Import: | Wrap Cell Content:

	company_id	company_name	phone	email	country	website
▶	b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://in
	b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@idoud.org	Australia	https://wt
	b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pir
	b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.co.uk	Germany	https://cn
	b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.pron@outlook.ca	New Zealand	https://ne
	b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.co.uk	Norway	https://ny
	b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cn
	b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://ne
	b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@icloud.net	United States	https://eb

companies 5 × Apply Revert

Output :

Action Output		
#	Time	Action
15	19:21:23	SELECT * FROM credit_cards
16	19:21:56	SELECT * FROM companies

Message Duration / Fetch

5000 row(s) returned 0.000 sec / 0.032 sec

100 row(s) returned 0.000 sec / 0.000 sec

139 -- Creamos la tabla transactions

140 • CREATE TABLE IF NOT EXISTS transactions (

141 id VARCHAR(255) NULL,

142 card_id VARCHAR(255) NULL,

143 business_id VARCHAR(255) NULL,

144 timestamp VARCHAR(255) NULL,

145 amount VARCHAR(255) NULL,

146 declined VARCHAR(255) NULL,

147 product_ids VARCHAR(255) NULL,

148 user_id VARCHAR(255) NULL,

149 lat VARCHAR(255) NULL,

150 longitude VARCHAR(255) NULL

151);

152

153 -- Añadimos los datos a la tabla

154 • LOAD DATA

155 INFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\S4\\\\transactions.csv"

156 INTO TABLE transactions

157 FIELDS TERMINATED BY ','

158 IGNORE 1 ROWS;

159

160 -- Hacemos cambios a la tabla transactions;

161 • ALTER TABLE transactions

162 MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,

163 MODIFY COLUMN amount DECIMAL(10,2),

164 MODIFY COLUMN declined TINYINT(1),

165 MODIFY COLUMN user_id INT;

166

167 -- Añadimos foreign key con card_id

168 • ALTER TABLE transactions

169 ADD CONSTRAINT fk_credit_cards

170 FOREIGN KEY (card_id)

171 REFERENCES credit_cards(id);

172

Output :

Action Output

#	Time	Action	Message	Duration / Fetch
15	19:21:23	SELECT * FROM credit_cards	5000 row(s) returned	0.000 sec / 0.032 sec
16	19:21:56	SELECT * FROM companies	100 row(s) returned	0.000 sec / 0.000 sec

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text area containing SQL code. The code includes modifications to the 'transactions' table, adding foreign keys for 'card_id', 'business_id', and 'user_id'. It also includes a SELECT statement to retrieve all data from the 'transactions' table.

```

164    MODIFY COLUMN declined TINYINT(1),
165    MODIFY COLUMN user_id INT;
166
167    -- Añadimos foreign key con card_id
168 • ALTER TABLE transactions
169    ADD CONSTRAINT fk_credit_cards
170        FOREIGN KEY (card_id)
171        REFERENCES credit_cards(id);
172
173    -- Añadimos foreign key con companies
174 • ALTER TABLE transactions
175    ADD CONSTRAINT fk_companies_transactions
176        FOREIGN KEY (business_id)
177        REFERENCES companies(company_id);
178
179    -- Añadimos foreign key con users
180 • ALTER TABLE transactions
181    ADD CONSTRAINT fk_user_transactions
182        FOREIGN KEY (user_id)
183        REFERENCES users(id);
184
185 • SELECT * FROM transactions;
186

```

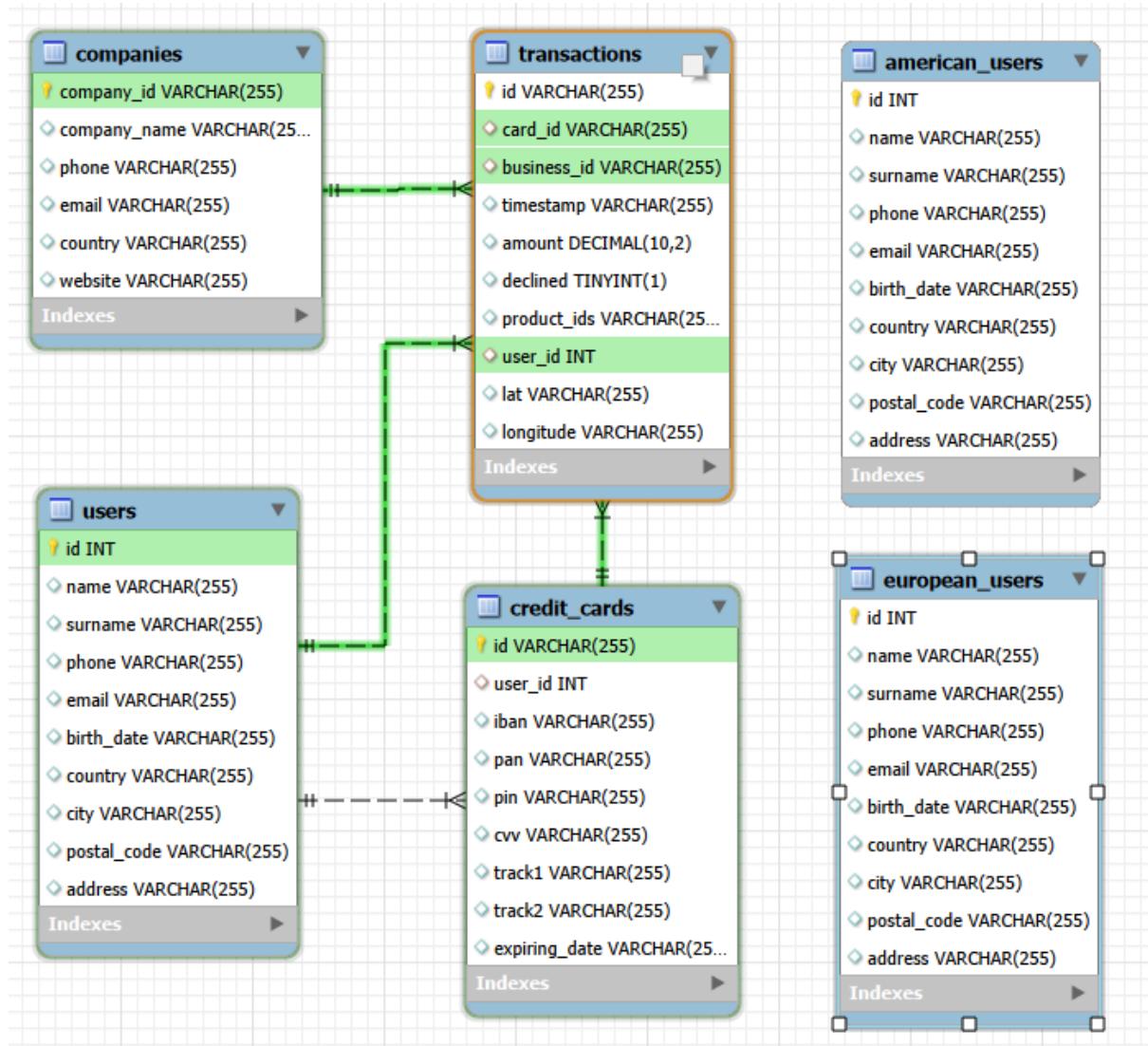
Below the code is a 'Result Grid' section displaying the results of the SELECT query. The grid has columns for id, card_id, business_id, timestamp, amount, declined, and product_ids. There are 10 rows of data. To the right of the grid is a vertical toolbar with buttons for Result Grid, Form Editor, and Field Types.

	id	card_id	business_id	timestamp	amount	declined	product_ids
▶	00043A49-2949-494B-A5DD-A5BAE3BB19DD	CcS-9294	b-2458	2024-08-28 07:16:46	395.43	0	16, 26, 97, 18
▶	000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	2016-12-21 20:07:18	155.63	0	66, 69, 87
▶	00045D6B-ED2E-4F2F-8186-CEE074D875D0	CcS-6699	b-2390	2020-07-14 15:37:45	326.01	0	30, 11, 16, 18
▶	000481C3-1C26-4FEF-83A0-4CD0EB004BBD	CcS-6696	b-2230	2017-09-04 19:44:53	161.60	0	72
▶	00051AA4-9CBE-4268-B070-C38062A1B3E2	CcS-7606	b-2266	2017-01-05 18:19:25	148.91	0	18
▶	0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	CcU-3358	b-2598	2023-09-23 04:51:43	294.59	0	35, 33, 19
▶	0009A151-9BCF-4E31-9053-A468FF77FAAB	CcS-7509	b-2546	2023-12-31 00:06:36	383.63	0	93, 55, 28, 18
▶	0009D494-6245-4DF9-955D-2C084191CFB	CcS-8483	b-2526	2017-07-18 07:52:02	197.80	0	55, 8, 72
▶	000A1DEC-CDB6-4AB2-A619-71DAB8D4A262	CcS-6467	b-2558	2018-09-08 05:29:58	339.94	0	46, 56, 73

Below the grid is a 'transactions 6 x' section with 'Apply' and 'Revert' buttons. Underneath is an 'Output' section titled 'Action Output' with a table showing execution details:

#	Time	Action	Message	Duration / Fetch
16	19:21:56	SELECT * FROM companies	100 row(s) returned	0.000 sec / 0.000 sec
17	19:23:43	SELECT * FROM transactions	100000 row(s) returned	0.015 sec / 0.219 sec

En la siguiente figura se muestran las relaciones entre las 6 tablas. Podemos observar una relación n-to-1 entre las tablas transactions y companies, unidas por transactions.business_id=companies.id. Esto se explica a que una misma compañía puede haber realizado diversas transacciones. También observamos la relación n-to-1 entre las tablas transactions y credit_cards, unidas por transactions.card_id=credit_cards.id; las tablas transactions y users, unidas por transactions.user_id=users.id; y entre las tablas credit_cards y users, unidas por credit_cards.user_id=users.id . Esto último indica que un usuario puede tener múltiples tarjetas de crédito.



Una vez el esquema de estrella ha sido elaborado, realizamos la primera consulta de todos los usuarios con más de 80 transacciones.

The screenshot shows the MySQL Workbench interface. At the top is a toolbar with various icons. Below it is a query editor window containing the following SQL code:

```
185 •  SELECT * FROM transactions;
186
187 -- Exercici 1
188 •  SELECT user_id, name, surname, country, COUNT(amount) AS num_transa
189   FROM transactions AS t
190   INNER JOIN users AS u
191     ON t.user_id = u.id
192   WHERE declined = 0
193   GROUP BY user_id
194   HAVING num_transa > 80
195   ORDER BY num_transa DESC;
196
197 -- Exercici 2
```

Below the query editor is a "Result Grid" window displaying the results of the query. The results are as follows:

	user_id	name	surname	country	num_transa
▶	185	Molly	Gilliam	United Kingdom	107
	289	Dxwgi	Hwcru	Germany	91
	318	Bnyr	Astuw	Italy	86

On the right side of the interface, there is a vertical sidebar with several tabs: "Result Grid" (which is selected), "Form Editor", "Field Types", "Query Stats", and "Execution Plan".

At the bottom, there is a "Result 9" window showing the execution details:

#	Time	Action	Message	Duration / Fetch
19	19:26:14	SELECT user_id, name, surname, country, COUN...	3 row(s) returned	0.531 sec / 0.000 sec
20	19:26:45	SELECT user_id, name, surname, country, COUN...	3 row(s) returned	0.531 sec / 0.000 sec

Exercici 2

Hacemos la segunda consulta para la media de cantidad de cada tarjeta para la compañía indicada.

The screenshot shows the Oracle SQL Developer interface. The SQL Editor at the top contains the following code:

```
197  -- Exercici 2
198 •  SELECT iban, ROUND(AVG(amount),2) AS media_cantidad
199   FROM transactions AS t
200   LEFT JOIN credit_cards AS cc
201     ON t.card_id = cc.id
202   LEFT JOIN companies AS c
203     ON t.business_id = c.company_id
204   WHERE company_name = 'Donec Ltd' AND declined = 0
205   GROUP BY iban
206   ORDER BY media_cantidad DESC;
207
208  -- Nivell 2
209  -- Exercici 1
```

The Result Grid below displays the query results:

iban	media_cantidad
XX383017813919620199366352	680.69
XX637706357397570394973913	680.01
XX971393971465292202312259	645.46
XX171847116928892375969307	628.89
XX225424638818542406223575	608.68
XX748890729057195711766071	607.29
TN9614563570667381893122	605.41
XX481908034037364242591185	605.36
XX194675519739256335753508	597.19
XX215962766061967195493437	594.26
XX449322320826890721001443	591.61
XX535185492735704229474237	570.09
CH9552373968796160224	566.38
XX347605377125637880303131	561.80
XX688471446697921912860304	543.42
XX605533964582458704105956	542.00
PL76249283566852676343404...	541.56
XX258862585706063154381130	539.81
XX651270526010893179119477	535.59
CR2918135947128138635	535.11
XX353434833721483641741327	518.96
VVC21407072407452750105450	518.71

The Result Grid panel on the right shows various options: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan. The status bar at the bottom right indicates "Read Only".

The Action Output panel at the bottom shows the following log entries:

#	Time	Action	Message	Duration / Fetch
21	19:27:18	SELECT iban, ROUND(AVG(amount),2) AS media...	370 row(s) returned	0.000 sec / 0.000 sec
22	19:28:12	SELECT iban, ROUND(AVG(amount),2) AS media...	370 row(s) returned	0.016 sec / 0.000 sec

Nivell 2

Exercici 1

Para elaborar la tabla indicada, credit_card_state, utilizamos la función de tabla ROW_NUMBER() con PARTITION BY, lo que nos permite tener una numeración de las fechas de transacciones que reinicia para cada tarjeta. De esta tabla resultado, se decidió si la tarjeta estaba activa o no sumando declined en los tres últimos días (last_days <= 3), de tal manera que si había estado declinada 3 veces (SUM(declined)=3), la tarjeta se considera inactiva.

Con la nueva tabla credit_card_state, se realizó la consulta indicada.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is the SQL editor window containing the following code:

```
208  -- Nivell 2
209  -- Exercici 1
210 • CREATE TABLE IF NOT EXISTS credit_cards_state AS
211   SELECT card_id, IF(SUM(declined)=3,'inactiva','activa') AS estado
212   FROM (SELECT card_id, DATE(timestamp), declined,
213            -- Utilizamos una función de ventana
214            ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY DATE(timestamp) DESC) AS last_days
215            FROM transactions) AS declined_date
216   WHERE last_days <= 3
217   GROUP BY card_id;
218
219 • SELECT COUNT(card_id) AS tarjetas_activas
220   FROM credit_cards_state
221   WHERE estado = 'activa';
222
```

Below the SQL editor is the Result Grid pane, which displays the result of the final query:

tarjetas_activas
4995

To the right of the Result Grid is a vertical toolbar with several icons: Result Grid (selected), Form Editor, Field Types, Query Stats, and Execution Plan.

At the bottom, there's a Results pane titled "Result 13" with a "Read Only" status. It shows the execution log:

Action Output
Time Action Message Duration / Fetch
24 19:29:36 CREATE TABLE IF NOT EXISTS credit_cards_st... 0 row(s) affected, 1 warning(s): 1050 Table 'credit...' 0.000 sec
25 19:29:39 SELECT COUNT(card_id) AS tarjetas_activas FR... 1 row(s) returned 0.000 sec / 0.000 sec

Nivell 3

Exercici 1

Antes de crear la nueva tabla, introducimos la tabla con datos de products.csv.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text editor containing a series of SQL commands. The commands include creating a table named 'products' with columns id, product_name, price, colour, weight, and warehouse_id, all set to VARCHAR(255) and NULL. It then loads data from a CSV file into this table using LOAD DATA INFILE. Finally, it alters the table to make the 'id' column an INT PRIMARY KEY UNIQUE NOT NULL. A 'Result Grid' tab is open at the bottom, displaying the 8 rows of data inserted into the 'products' table. The 'Output' tab shows the log of actions taken, including the ALTER TABLE command which failed with an error message: 'Error Code: 1068. Multiple primary key defined'. The 'Action Output' section also shows the successful SELECT * FROM products query.

```
223 -- Nivell 3
224 -- Creamos la tabla products
225 • CREATE TABLE IF NOT EXISTS products (
226     id VARCHAR(255) NULL,
227     product_name VARCHAR(255) NULL,
228     price VARCHAR(255) NULL,
229     colour VARCHAR(255) NULL,
230     weight VARCHAR(255) NULL,
231     warehouse_id VARCHAR(255) NULL
232 );
233
234 -- Añadimos los datos a la tabla
235 • LOAD DATA
236     INFILE "C:\\\\ProgramData\\\\MySQL\\\\MySQL Server 8.0\\\\Uploads\\\\S4\\\\products.csv"
237     INTO TABLE products
238     FIELDS TERMINATED BY ','
239     IGNORE 1 ROWS;
240
241 -- Hacemos cambios a la tabla products;
242 • ALTER TABLE products
243     MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;
244
245 • SELECT * FROM products;
246
```

	id	product_name	price	colour	weight	warehouse_id
▶	1	Direwolf Stannis	\$161.11	#7c7c7c	1	WH-4
	2	Tarly Stark	\$9.24	#919191	2	WH-3
	3	duel tourney Lannister	\$171.13	#d8d8d8	1.5	WH-2
	4	warden south duel	\$71.89	#111111	3	WH-1
	5	skywalker ewok	\$171.22	#dbdbdb	3.2	WH-0
	6	dooku solo	\$136.60	#c4c4c4	0.8	WH-1
	7	north of Casterly	\$63.33	#b7b7b7	0.6	WH-2
	8	Winterfell	\$32.37	#383838	1.4	WH-3

products 14 ×

Output:

#	Time	Action	Message	Duration / Fetch
✗	28 19:32:21	ALTER TABLE products MODIFY COLUMN id IN...	Error Code: 1068. Multiple primary key defined	0.000 sec
✓	29 19:32:24	SELECT * FROM products	100 row(s) returned	0.000 sec / 0.000 sec

Ahora con ya todas las tablas, elaboramos la tabla transaction_product. Para elaborarla, usamos una CTE recursiva, con la cual extraemos el primer índice de la lista de productos de cada fila de transacción con TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) y guardabamos el resto con SUBSTRING(product_ids, LENGTH(SUBSTRING_INDEX(product_ids, ',', 1)) + 2). Continuamos extrayendo índices de productos hasta que la lista esté vacía.

```

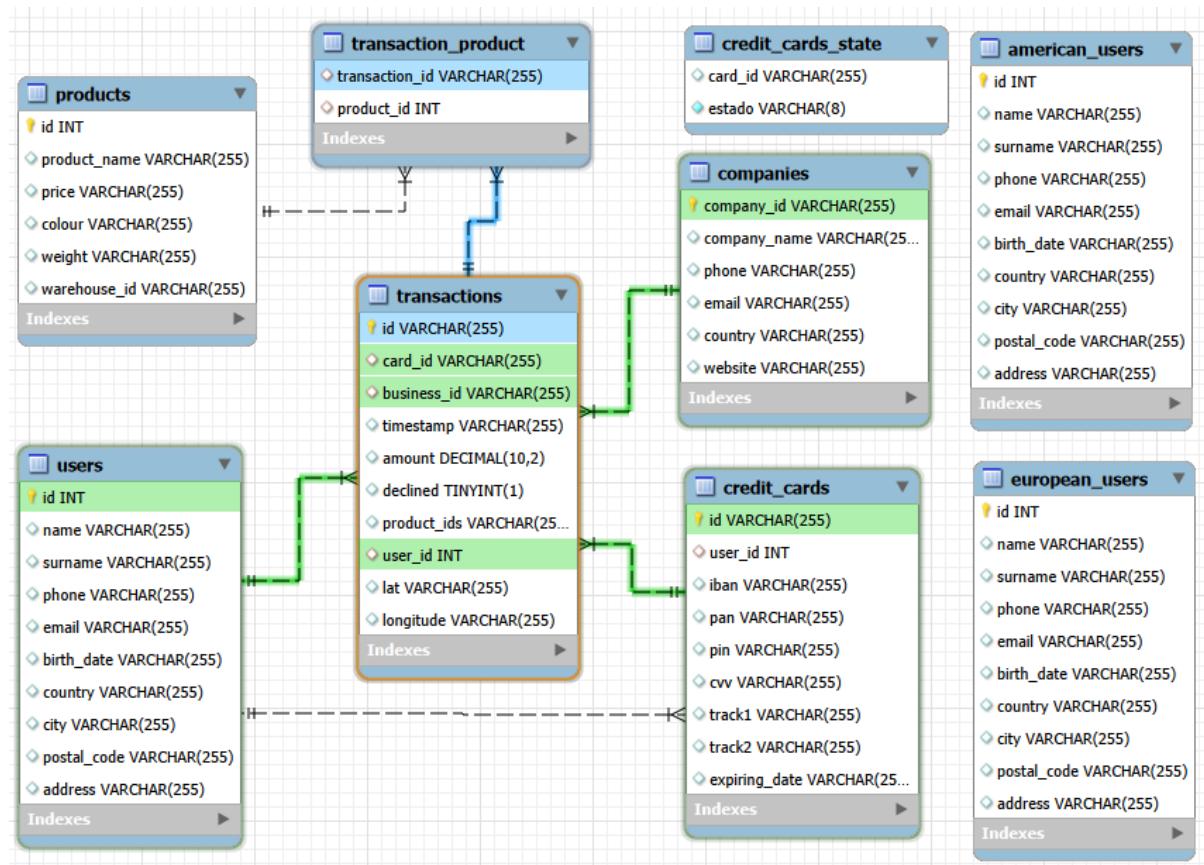
247 -- Creamos tabla que relaciona productos con transacciones, transaction_product
248 • CREATE TABLE IF NOT EXISTS transaction_product AS
249     -- Usamos una CTE recursiva
250     WITH RECURSIVE list_products AS (
251         SELECT id, TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) AS product_id,
252             SUBSTRING(product_ids, LENGTH(SUBSTRING_INDEX(product_ids, ',', 1)) + 2) AS rest_product_ids
253         FROM transactions
254
255         UNION ALL
256
257         SELECT id, TRIM(SUBSTRING_INDEX(rest_product_ids, ',', 1)),
258             SUBSTRING(rest_product_ids, LENGTH(SUBSTRING_INDEX(rest_product_ids, ',', 1)) + 2)
259         FROM list_products
260         WHERE rest_product_ids <> ''
261     )
262     SELECT id AS transaction_id, CAST(product_id AS UNSIGNED) AS product_id
263     FROM list_products
264     ORDER BY id;
265
266     -- Hacemos cambios a la tabla transaction_product;
267 • ALTER TABLE transaction_product
268     MODIFY COLUMN product_id INT;
269
270     -- Añadimos foreign key con transactions
271 • ALTER TABLE transaction_product
272     ADD CONSTRAINT fk_tp_t
273     FOREIGN KEY (transaction_id)
274     REFERENCES transactions(id);
275
276     -- Añadimos foreign key con products
277 • ALTER TABLE transaction_product
278     ADD CONSTRAINT fk_tp_p
279     FOREIGN KEY (product_id)
280     REFERENCES products(id);

```

Output:

Action Output	#	Time	Action	Message	Duration / Fetch
	29	19:32:24	SELECT * FROM products	100 row(s) returned	0.000 sec / 0.000 sec
	30	19:34:00	CREATE TABLE IF NOT EXISTS transaction_pro...	0 row(s) affected, 1 warning(s): 1050 Table 'transa...	0.000 sec

Tras los cambios indicados, obtuvimos el siguiente diagrama de estrella.



Elaboramos la consulta para saber el número de veces que se ha vendido cada producto.

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a code editor window containing the following SQL script:

```
269
270  -- Añadimos foreign key con transactions
271 • ALTER TABLE transaction_product
272   ADD CONSTRAINT fk_tp_t
273     FOREIGN KEY (transaction_id)
274       REFERENCES transactions(id);
275
276  -- Añadimos foreign key con products
277 • ALTER TABLE transaction_product
278   ADD CONSTRAINT fk_tp_p
279     FOREIGN KEY (product_id)
280       REFERENCES products(id);
281
282  -- Exercici 1
283 • SELECT product_id, product_name, COUNT(transaction_id) AS num_ventas
284   FROM transaction_product AS tp
285   LEFT JOIN products AS p
286     ON tp.product_id = p.id
287   GROUP BY product_id
288   ORDER BY num_ventas DESC;
```

Below the code editor is a "Result Grid" pane showing the query results:

product_id	product_name	num_ventas
52	riverlands the duel	2654
29	Tully maester Tarly	2635
21	duel Direwolf	2609
16	the duel warden	2608
66	mustafar jinn	2601
87	sith Jade	2598
48	rock Renly in	2597
33	duel warden	2597
23	riverlands north	2593
68	Stark Karstark	2589
88	Stannis warden so...	2587
4	warden south duel	2584
28	chewbacca mustafar	2584

At the bottom of the interface is an "Output" pane showing the execution log:

#	Time	Action	Message	Duration / Fetch
⚠ 30	19:34:00	CREATE TABLE IF NOT EXISTS transaction_pro...	0 row(s) affected, 1 warning(s): 1050 Table transa...	0.000 sec
✓ 31	19:34:49	SELECT product_id, product_name, COUNT(tran...	100 row(s) returned	0.562 sec / 0.000 sec

Revisión peer-to-peer

Revisado por Minu Campoy

- Tener en cuenta que al unir american_users y european_users en la tabla users, se pierde información si no se crea una nueva columna que indique de qué tabla procede cada usuario. No es un problema en esta actividad porque ningún ejercicio pide tener en cuenta la diferencia, pero se debe considerar por si fuera necesario.
- Se podría conectar la tabla users con american_users y european_users, como forma de mantener la separación que se pierde en users haciendo un JOIN con cualquiera de las dos.
- También se podría unir la tabla credit_cards con credit_cards_state.