Level 1

Download the csv files, study them, and design a database with a star schema containing at least 4 tables, with which you can perform the following queries:

First, I create a new database: **store**. I set it as default.

```
5 •  CREATE DATABASE IF NOT EXISTS store;
6
7 •  USE store;
```

```
1  11:57:01  CREATE DATABASE IF NOT EXISTS store        1 row(s) affected   0.109 sec
2  11:57:09  USE store                                  0 row(s) affected   0.000 sec
```

The **schema** I'll design has a **fact** table (transactions) and five **dimension** tables (american_users, european_users, credit_cards, companies and products).

Following best practices, I first create the dimension tables and, at the end, the fact table. I initially create tables all with datatypes VARCHAR(255), then I analyse data and make the necessary modifications in each of the tables. I finally define primary keys and fk constrains to relate the tables and obtain the schema.

Dimension table american_users:

```
9  • CREATE TABLE IF NOT EXISTS american_users (
10                                    id VARCHAR(255),
11                                    name VARCHAR(255),
12                                    surname VARCHAR(255),
13                                    phone VARCHAR(255),
14                                    email VARCHAR(255),
15                                    birth_date VARCHAR(255),
16                                    country VARCHAR(255),
17                                    city VARCHAR(255),
18                                    postal_code VARCHAR(255),
19                                    address VARCHAR(255)
20                                   );
```

```
4  11:12:54  CREATE TABLE IF NOT EXISTS american_users (id VARCHAR(255),  name ...  0 row(s) affected  0.093 sec
```

To be able to load data from a csv stored in any of the folders, I've made some changes to the permissions:

```
78 •  SET GLOBAL local_infile = 1;
79 •  SHOW GLOBAL VARIABLES LIKE 'local_infile';
```

```
103  12:10:52  SET GLOBAL local_infile = 1                    0 row(s) affected    0,000 sec
104  12:10:57  SHOW GLOBAL VARIABLES LIKE 'local_infile'      1 row(s) returned    0,000 sec / 0,000 sec
```

I now load the data to table american_users from the csv file:

```
25 •  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprint 4\\american_users.csv"
26    INTO TABLE american_users
27    FIELDS TERMINATED BY ','
28    OPTIONALLY ENCLOSED BY '"'
29    LINES TERMINATED BY '\n'
30    IGNORE 1 ROWS;
```

```
5  11:21:58  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin...  1010 row(s) affected Records: 1010  Deleted: 0  Skipped: 0  Warnings: 0  0.125 sec
```

I visualize the table to see if the data was loaded correctly and to identify modifications I need to make, if any:

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

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@prot...	Nov 17, 1985	United States	New York	10001	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@p...	Aug 23, 1992	United States	Philadelphia	19101	903 Sit Ave
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@ao...	Apr 29, 1998	United States	Houston	77001	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-3269	ornare.egestas@iclo...	Feb 18, 1989	United States	Phoenix	85001	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames...	Sep 26, 1998	United States	Philadelphia	19101	341-2821 Ultrices Av.

```
7  11:53:05  SELECT * FROM store.american_users  1010 row(s) returned  0.016 sec / 0.000 sec
```

I repeat the same steps, this time for **dimension table european_users**, which has the same structure as american_users:

```
32 •  CREATE TABLE IF NOT EXISTS european_users (
33        id VARCHAR(255),
34        name VARCHAR(255),
35        surname VARCHAR(255),
36        phone VARCHAR(255),
37        email VARCHAR(255),
38        birth_date VARCHAR(255),
39        country VARCHAR(255),
40        city VARCHAR(255),
41        postal_code VARCHAR(255),
42        address VARCHAR(255)
43    );
```

```
8  11:57:24  CREATE TABLE IF NOT EXISTS european_users (id VARCHAR(255),  na...  0 row(s) affected  0.125 sec
```

I load the data to table european_users from the csv file:

```
LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprint 4\\european_users.csv"
INTO TABLE european_users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
9  11:58:40  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin...  3990 row(s) affected Records: 3990  Deleted: 0  Skipped: 0  Warnings: 0  0.172 sec
```

I visualize the table to see if the data loaded correctly and to identify modifications I need to make, if any:

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

id	name	surname	phone	email	birth_date	country	city	postal_code	address
151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	Jul 2, 1980	United Kingdom	London	EC1A 1BB	Ap #432-4493 Aliquet Rd.
152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google....	Sep 30, 1979	United Kingdom	Birmingham	B1 1AA	551-8930 Lobortis Street
153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	London	EC1A 1BB	Ap #312-5898 Consectetuer St.
154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Manchester	M1 1AE	872-1866 Pede Rd.
155	Joshua	Russell	055 4409 5286	justo.nec.ante@outlook....	Jan 23, 1984	United Kingdom	Manchester	M1 1AE	Ap #285-4727 Auctor. Av.

```
10  11:59:37  SELECT * FROM store.european_users  3990 row(s) returned  0.000 sec / 0.015 sec
```

Dimension table credit_cards:

```
CREATE TABLE IF NOT EXISTS credit_cards (
                id VARCHAR(255),
                user_id VARCHAR(255),
                iban VARCHAR(255),
                pan VARCHAR(255),
                pin VARCHAR(255),
                cvv VARCHAR(255),
                track1 VARCHAR(255),
                track2 VARCHAR(255),
                expiring_date VARCHAR(255)
            );
```

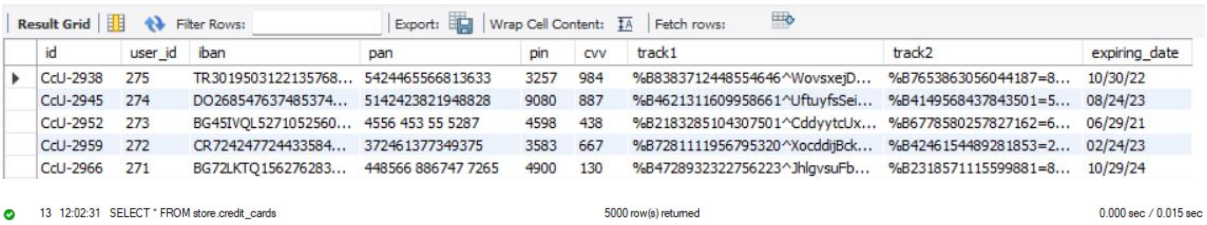
```
11  12:00:30  CREATE TABLE IF NOT EXISTS credit_cards ( id VARCHAR(255),  user_i...  0 row(s) affected  0.078 sec
```

I load the data to table credit_cards from the csv file:

```
LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprint 4\\credit_cards.csv"
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
12  12:01:54  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin...  5000 row(s) affected Records: 5000  Deleted: 0  Skipped: 0  Warnings: 0  0.234 sec
```

I visualize the table to see if the data loaded correctly and to identify modifications I need to make, if any:



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

id	user_id	iban	pan	pin	cvv	track1	track2	expiring_date
CcU-2938	275	TR3019503122135768...	5424465566813633	3257	984	%B8383712448554646^WovsxejD...	%B7653863056044187=8...	10/30/22
CcU-2945	274	DO268547637485374...	5142423821948828	9080	887	%B4621311609958661^UftuyfsSei...	%B4149568437843501=5...	08/24/23
CcU-2952	273	BG45IVQL5271052560...	4556 453 55 5287	4598	438	%B2183285104307501^CddyytcUx...	%B6778580257827162=6...	06/29/21
CcU-2959	272	CR7242477244335 84...	372461377349375	3583	667	%B7281111956795320^XocddjBck...	%B4246154489281853=2...	02/24/23
CcU-2966	271	BG72LKTQ156276283...	448566 886747 7265	4900	130	%B4728932322756223^JhlgvsuFb...	%B2318571115599881=8...	10/29/24

13 12:02:31 SELECT * FROM store.credit_cards 5000 row(s) returned 0.000 sec / 0.015 sec

Dimension table companies:

```
71 • CREATE TABLE IF NOT EXISTS companies (
72                     company_id VARCHAR(255),
73                     company_name VARCHAR(255),
74                     phone VARCHAR(255),
75                     email VARCHAR(255),
76                     country VARCHAR(255),
77                     website VARCHAR(255)
78                 );
```

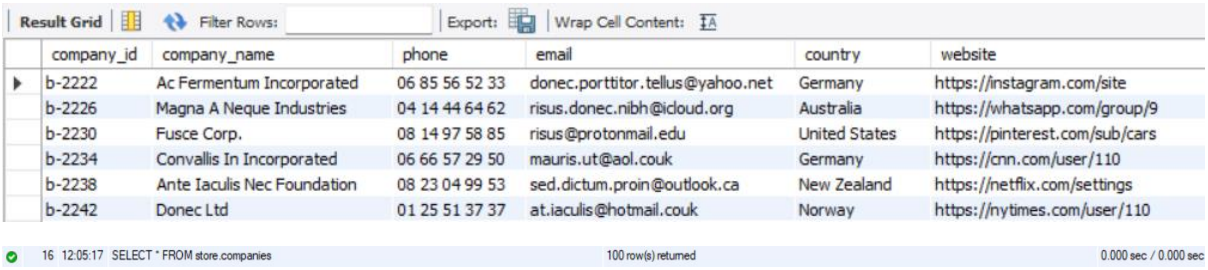
14 12:03:50 CREATE TABLE IF NOT EXISTS companies (company_id VARCHAR(255). ... 0 row(s) affected 0.062 sec

I load the data to table companies from the csv file:

```
83 • LOAD DATA LOCAL INFILE "C:\\Users\\Asus\\Documents\\IT\\Especialidad IT Academy\\Sprint 4\\csv iniciales\\companies.csv"
84   INTO TABLE companies
85   FIELDS TERMINATED BY ','
86   OPTIONALLY ENCLOSED BY '"'
87   LINES TERMINATED BY '\n'
88   IGNORE 1 ROWS;
```

15 12:04:50 LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin... 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0 0.047 sec

I visualize the table to see if the data loaded correctly and to identify modifications I need to make, if any:



Result Grid | Filter Rows: | Export: | 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://instagram.com/site
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/group/9
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/sub/cars
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany	https://cnn.com/user/110
b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.proin@outlook.ca	New Zealand	https://netflix.com/settings
b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.couk	Norway	https://nytimes.com/user/110

16 12:05:17 SELECT * FROM store.companies 100 row(s) returned 0.000 sec / 0.000 sec

Dimension table products:

```sql
CREATE TABLE IF NOT EXISTS products (
    id VARCHAR(255),
    product_name VARCHAR(255),
    price VARCHAR(255),
    colour VARCHAR(255),
    weight VARCHAR(255),
    warehouse_id VARCHAR(255)
);
```

```
17  12:07:08  CREATE TABLE IF NOT EXISTS products (id VARCHAR(255), product_name VARCHAR(255),  ...  0 row(s) affected  0.078 sec
```

I load the data from the csv file:

```sql
LOAD DATA LOCAL INFILE "C:\\Users\\Asus\\Documents\\IT\\Especialidad IT Academy\\Sprint 4\\csv iniciales\\products.csv"
INTO TABLE products
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
18  12:09:07  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin...  100 row(s) affected Records: 100  Deleted: 0  Skipped: 0  Warnings: 0  0.031 sec
```

I visualize the table to see if the data loaded correctly and to identify any modifications I need to make:

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

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

```
19  12:09:40  SELECT * FROM store.products  100 row(s) returned  0.000 sec / 0.000 sec
```

Fact table transactions:

```sql
CREATE TABLE IF NOT EXISTS transactions (
    id VARCHAR(255),
    card_id VARCHAR(255),
    business_id VARCHAR(255),
    timestamp VARCHAR(255),
    amount VARCHAR(255),
    declined VARCHAR(255),
    product_ids VARCHAR(255),
    user_id VARCHAR(255),
    lat VARCHAR(500),
    longitude VARCHAR(500)
);
```

```
20  12:11:21  CREATE TABLE IF NOT EXISTS transactions (id VARCHAR(255), card_id VARCHAR(255), business...  0 row(s) affected  0.078 sec
```

I load the data to table transactions from the csv file:

```
105 •  LOAD DATA LOCAL INFILE "C:\\Users\\Asus\\Documents\\IT\\Especialidad IT Academy\\Sprint 4\\csv iniciales\\transactions.csv"
106    INTO TABLE transactions
107    FIELDS TERMINATED BY ';'
108    OPTIONALLY ENCLOSED BY '"'
109    LINES TERMINATED BY '\n'
110    IGNORE 1 ROWS;
```

```
21  12:12:40  LOAD DATA LOCAL INFILE "C:\\Users\\Usuari\\Desktop\\Dades\\Especialidad IT Academy\\Sprin...  100000 row(s) affected Records: 100000  Deleted: 0  Skipped: 0  Warnings: 0  1.641 sec
```

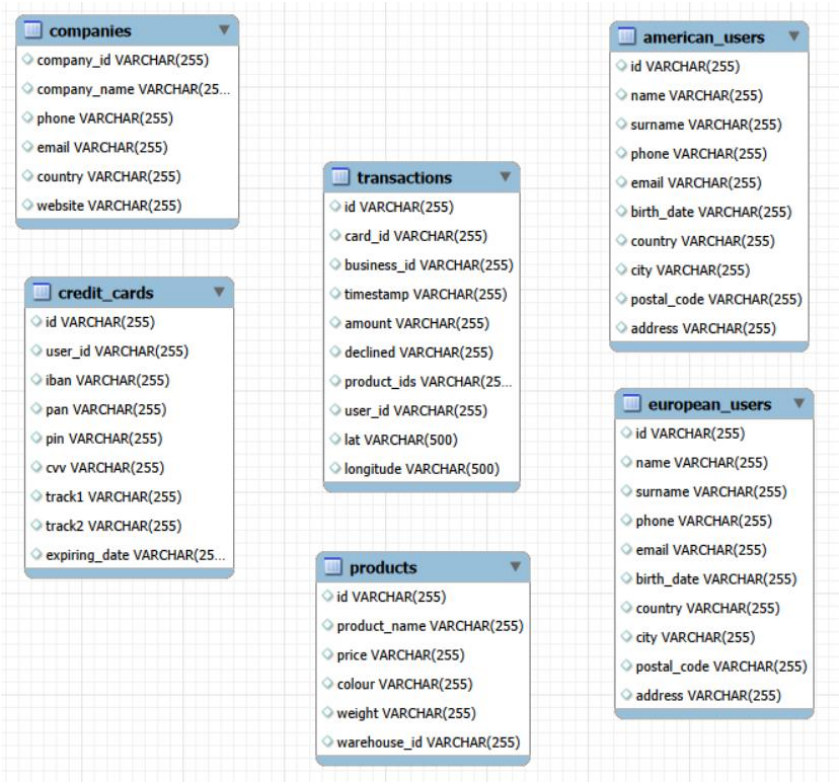
I visualize the table to see if the data loaded correctly and to identify modifications I need to make, if any:

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

id	card_id	business_	timestamp	amount	declined	product_ids	user_id	lat	longitude
CDDA7E40-544D-47BB-A4ED-671D...	CcS-6894	b-2466	2018-12-12 08:0...	161.88	0	75, 73, 98	2313	59.62050974356...	16.559977155728436
09456357-8E9B-475A-8257-87A02...	CcS-5135	b-2342	2024-05-20 22:4...	171.13	0	3	554	45.76458841901...	4.843056518287656
C47C7C84-C174-4973-A76B-825A...	CcS-8415	b-2250	2018-11-04 23:1...	497.29	0	92, 85, 36, 23	3834	52.06849608409...	4.301099382555438
2FB526AA-3844-4DDF-AC09-2EBC...	CcS-6553	b-2610	2022-06-17 09:1...	344.15	0	5, 27	1972	39.47598513179...	-0.3764194439184784

```
22  12:12:57  SELECT * FROM store.transactions  100000 row(s) returned  0.016 sec / 0.187 sec
```

Right now, the tables are unrelated, they don't have PK or FK defined and all the variables are VARCHAR(255):



Before defining the relations to create the schema, I make all the necessary modifications to the data and the structure of the database:

users

As american_users and european_users have the same structure, the same data granularity, I create a new dimension table with all the users, adding a column that specifies whether the continent is America or Europe.

```
124 •  CREATE TABLE IF NOT EXISTS users AS
125                                 SELECT *, 'America' AS continent
126                                 FROM american_users
127                                 UNION ALL
128                                 SELECT *, 'Europe' AS continent
129                                 FROM european_users
130                                 ;
```

```
23  12:34:24  CREATE TABLE IF NOT EXISTS users AS SELECT *, 'America' AS continent FROM american_users ...  5000 row(s) affected Records: 5000  Duplicates: 0  Warnings: 0  0.640 sec
```

I visualize the table to see if the data loaded correctly and to identify extra modifications I need to make, if any:

```
1 •  SELECT * FROM store.users;
```

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

id	name	surname	phone	email	birth_date	country	city	postal_code	address	continent
1	Zeus	Gamble	1-282-581-0551	interdum.enim@proto...	Nov 17, 1985	United States	New York	10001	348-7818 Sagittis St.	America
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@pr...	Aug 23, 1992	United States	Philadelphia	19101	903 Sit Ave	America
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol...	Apr 29, 1998	United States	Houston	77001	736-2063 Tellus St.	America
4	Howard	Stafford	1-411-740-3269	ornare.egestas@iclou...	Feb 18, 1989	United States	Phoenix	85001	Ap #545-2244 Erat. Rd.	America

```
24  12:35:32  SELECT * FROM store.users  5000 row(s) returned  0.000 sec / 0.016 sec
```

I've been making some research on how variable length could affect performance and I basically conclude that the reason to limit length is if I have a specific need for it to be smaller. Otherwise, it's better to make them all 255 or maintain a reasonable length.

```
132 •  SELECT 'id' AS column_name, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id) * 1.2)) AS 'plus_20'
133    FROM users
134    UNION ALL
135    SELECT 'phone', MAX(LENGTH(phone)), CEIL(MAX(LENGTH(phone) * 1.2))
136    FROM users
137    UNION ALL
138    SELECT 'email', MAX(LENGTH(email)), CEIL(MAX(LENGTH(email) * 1.2))
139    FROM users
140    UNION ALL
141    SELECT 'country', MAX(LENGTH(country)), CEIL(MAX(LENGTH(country) * 1.2))
142    FROM users
143    UNION ALL
144    SELECT 'city', MAX(LENGTH(city)), CEIL(MAX(LENGTH(city) * 1.2))
145    FROM users
```

Result Grid | Filter Rows:

column_name	maximum_length	plus_20
id	4	5
phone	15	18
email	40	48
country	14	17
city	12	15
postal_code	8	10
continent	7	9

2 21:18:18 SELECT 'id' AS column_name, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id) * 1.2)) A... 7 row(s) returned 0.125 sec / 0.000 sec

I decide to modify the length and type of some of the variables. I first measure the maximum length of VARCHAR variables I'm interested in, and I modify them according to specific needs. I also set the PK and some variables as NOT NULL.

```
153 •  ALTER TABLE users
154        MODIFY COLUMN id INT UNSIGNED PRIMARY KEY,
155        MODIFY COLUMN name VARCHAR(255) NOT NULL,
156        MODIFY COLUMN surname VARCHAR(255) NOT NULL,
157        MODIFY COLUMN phone VARCHAR(20),
158        MODIFY COLUMN email VARCHAR(100),
159        MODIFY COLUMN country VARCHAR(60),
160        MODIFY COLUMN city VARCHAR(60),
161        MODIFY COLUMN postal_code VARCHAR(10),
162        MODIFY COLUMN address VARCHAR(100),
163        MODIFY COLUMN continent VARCHAR(30);
```

3 21:27:44 ALTER TABLE users MODIFY COLUMN id INT UNSIGNED PRIMARY KEY, MODIFY COLUMN name ... 5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

In the case of variable birth_date, for MySQL to accept date format "Nov 17, 1985", I need to make a conversion using STR_TO_DATE(), disactivating and activating the safe mode.

```
165 •  SET SQL_SAFE_UPDATES = 0;
166 •  UPDATE users
167    SET birth_date = STR_TO_DATE(birth_date, '%b %d, %Y');
168 •  SET SQL_SAFE_UPDATES = 1;
```

6	21:38:12	SET SQL_SAFE_UPDATES = 0	0 row(s) affected	0.016 sec
7	21:38:17	UPDATE users SET birth_date = STR_TO_DATE(birth_date, '%b %d, %Y')	5000 row(s) affected Rows matched: 5000 Changed: 5000 Warnings: 0	0.313 sec
8	21:38:21	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec

Finally, I modify the datatype:

```
170 •  ALTER TABLE users
171    MODIFY COLUMN birth_date DATE;
```

```
10  21:40:13  ALTER TABLE users  MODIFY COLUMN birth_date DATE    5000 row(s) affected Records: 5000  Duplicates: 0  Warnings: 0    0.391 sec
```

companies

I repeat the same process to measure the maximum length of the data and make the necessary modifications, according to the specific needs.

```
SELECT 'company_id' AS column_name, MAX(LENGTH(company_id)) AS maximum_length, CEIL(MAX(LENGTH(company_id)) * 1.2) AS 'plus_20%'
FROM companies
UNION ALL
SELECT 'phone', MAX(LENGTH(phone)), CEIL(MAX(LENGTH(phone)) * 1.2)
FROM companies
UNION ALL
SELECT 'email', MAX(LENGTH(email)), CEIL(MAX(LENGTH(email)) * 1.2)
FROM companies
UNION ALL
SELECT 'country', MAX(LENGTH(country)), CEIL(MAX(LENGTH(country)) * 1.2)
FROM companies
UNION ALL
SELECT 'website', MAX(LENGTH(website)), CEIL(MAX(LENGTH(website)) * 1.2)
FROM companies;
```

Result Grid | Filter Rows:

column_name	maximum_length	plus_20%
company_id	6	8
phone	14	17
email	38	46
country	14	17
website	32	39

```
11  21:49:13  SELECT 'company_id' AS column_name, MAX(LENGTH(company_id)) AS maximum_length, CEIL(MA...  5 row(s) returned    0.016 sec / 0.000 sec
```

I modify the length of some of the variables and set the PK.

```
ALTER TABLE companies
    MODIFY COLUMN company_id VARCHAR(8) PRIMARY KEY,
    MODIFY COLUMN company_name VARCHAR(255) NOT NULL,
    MODIFY COLUMN phone VARCHAR(20),
    MODIFY COLUMN email VARCHAR(50),
    MODIFY COLUMN country VARCHAR(60),
    MODIFY COLUMN website VARCHAR(255);
```

```
13  21:52:58  ALTER TABLE companies MODIFY COLUMN company_id VARCHAR(8) PRIMARY KEY,-- modify it in...  100 row(s) affected Records: 100  Duplicates: 0  Warnings: 0    0.187 sec
```

credit_cards

I repeat the same process to measure the maximum length of the data and make the necessary modifications, according to the specific needs.

```sql
SELECT 'id' AS card_id, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id)) * 1.2) AS 'plus_20%'
FROM credit_cards
UNION ALL
SELECT 'user_id', MAX(LENGTH(user_id)), CEIL(MAX(LENGTH(user_id)) *  1.2)
FROM credit_cards
UNION ALL
SELECT 'iban', MAX(LENGTH(iban)), CEIL(MAX(LENGTH(iban)) * 1.2)
FROM credit_cards
UNION ALL
SELECT 'pan', MAX(LENGTH(pan)), CEIL(MAX(LENGTH(pan)) *  1.2)
FROM credit_cards
UNION ALL
SELECT 'pin', MAX(LENGTH(pin)), CEIL(MAX(LENGTH(pin)) *  1.2)
FROM credit_cards
```

Result Grid | Filter Rows:

card_id	maximum_length	plus_20%
id	8	10
user_id	4	5
iban	31	38
pan	19	23
pin	4	5
cvv	3	4
track1	52	63
track2	36	44

14 21:54:42 SELECT 'id' AS card_id, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id)) * 1.2) AS 'pl... 8 row(s) returned 0.078 sec / 0.000 sec

Modifications of the length and type of some variables, as well as setting the PK.

```sql
ALTER TABLE credit_cards
    MODIFY COLUMN id VARCHAR(20) PRIMARY KEY,
    MODIFY COLUMN user_id INT UNSIGNED NOT NULL,
    MODIFY COLUMN iban VARCHAR(34) NOT NULL,
    MODIFY COLUMN pan VARCHAR(20) NOT NULL,
    MODIFY COLUMN pin CHAR(4) NOT NULL,
    MODIFY COLUMN cvv CHAR(4) NOT NULL,
    MODIFY COLUMN track1 VARCHAR(100),
    MODIFY COLUMN track2 VARCHAR(100),
    MODIFY COLUMN expiring_date VARCHAR(255);
```

15 21:58:36 ALTER TABLE credit_cards MODIFY COLUMN id VARCHAR(20) PRIMARY KEY,-- modify it in transa... 5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0 0.453 sec

For MySQL to accept date format "10/30/22 ", again I need to make a conversion using STR_TO_DATE(), disactivating and activating the safe mode.

```sql
SET SQL_SAFE_UPDATES = 0;
UPDATE credit_cards
SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y');
SET SQL_SAFE_UPDATES = 1;
```

```
18  22:03:59  SET SQL_SAFE_UPDATES = 0                                                    0 row(s) affected                                                  0.000 sec
19  22:04:02  UPDATE credit_cards  SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y')   5000 row(s) affected Rows matched: 5000  Changed: 5000  Warnings: 0   0.266 sec
20  22:04:11  SET SQL_SAFE_UPDATES = 1                                                    0 row(s) affected                                                  0.000 sec
```

Finally, I modify the datatype:

```
238 •  ALTER TABLE credit_cards
239        MODIFY COLUMN expiring_date DATE;
```

```
21  22:05:39  ALTER TABLE credit_cards MODIFY COLUMN expiring_date DATE   5000 row(s) affected Records: 5000  Duplicates: 0  Warnings: 0   0.375 sec
```

transactions

I repeat the same process to measure the maximum length of the data and make the necessary modifications, according to the specific needs.

```
230 •  SELECT 'id' AS id_transaction, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id) * 1.2)) AS 'plus_20%'
231    FROM transactions
232    UNION ALL
233    SELECT 'lat', MAX(LENGTH(LAT)), CEIL(MAX(LENGTH(lat) * 1.2))
234    FROM transactions
235    UNION ALL
236    SELECT 'longitude', MAX(LENGTH(longitude)), CEIL(MAX(LENGTH(longitude) * 1.2))
237    FROM transactions;
```

Result Grid | Filter Rows:

id_transaction	maximum_length	plus_20%
id	36	44
lat	18	22
longitude	19	23

```
1  09:30:05  SELECT 'id' AS id_transaction, MAX(LENGTH(id)) AS maximum_length, CEIL(MAX(LENGTH(id) * 1.2)) A...  3 row(s) returned   0.922 sec / 0.000 sec
```

Modifications of the length, type and name of some variables (in some cases, to match the same variables in other tables), as well as setting the PK.

In the case of ‘latitude’ and ‘longitude’, I’ve been making some research, and it seems the consensus for storing this data is using DECIMAL(9,6). Of course, it’d depend on the context. For this particular purpose, I decide to follow that rule, since including more decimals would mean a minimal distance difference that wouldn’t affect the tasks I’m conducting. That’s why I get a warning when updating the table (the data in latitude and longitude is truncated).

```
251 •  ALTER TABLE transactions
252        MODIFY COLUMN id VARCHAR(50) PRIMARY KEY,
253        MODIFY COLUMN card_id VARCHAR(20) NOT NULL,
254        CHANGE COLUMN business_id company_id VARCHAR(8) NOT NULL,
255        MODIFY COLUMN timestamp TIMESTAMP,
256        MODIFY COLUMN amount DECIMAL(10,2),
257        MODIFY COLUMN declined BOOLEAN,
258        MODIFY COLUMN user_id INT UNSIGNED NOT NULL,
259        CHANGE COLUMN lat latitude DECIMAL(9,6),
260        MODIFY COLUMN longitude DECIMAL(9,6);
```

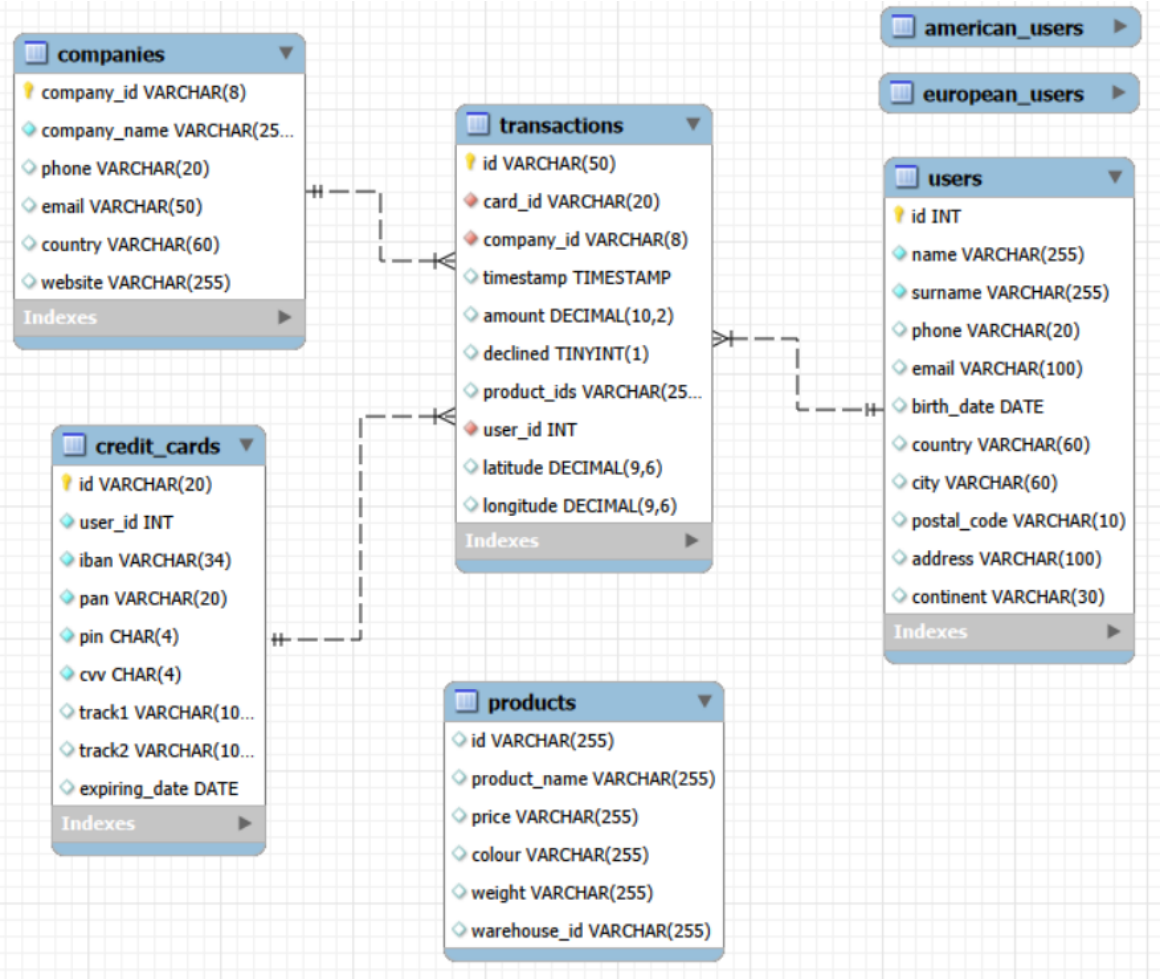
```
3  09:42:48  ALTER TABLE transactions  MODIFY COLUMN id VARCHAR(50) PRIMARY KEY,  MODIFY COL...  3.921 sec
```

Finally, I define the **fk constraints** to relate the tables:

```
262 •  ALTER TABLE transactions
263    ADD CONSTRAINT fk_transactions_card_id FOREIGN KEY (card_id) REFERENCES credit_cards(id),
264    ADD CONSTRAINT fk_transactions_company_id FOREIGN KEY (company_id) REFERENCES companies(company_id),
265    ADD CONSTRAINT fk_transactions_user_id FOREIGN KEY (user_id) REFERENCES users(id);
```

```
5  10:00:50  ALTER TABLE transactions ADD CONSTRAINT fk_transactions_card_id FOREIGN KEY (card_id) RE...  100000 row(s) affected Records: 100000  Duplicates: 0  Warnings: 0  5.734 sec
```

Now, the **schema** looks like this:



Unrelated tables: european_users and american_users. I decide to keep these tables unrelated and minimized. And table products is unrelated, too, for now.

Level 1

Exercise 1

Make a subquery that shows all users with more than 80 transactions, using at least 2 tables.

```sql
SELECT u.id
FROM users AS u
WHERE EXISTS    (   SELECT t.user_id
                    FROM transactions AS t
                    WHERE t.user_id = u.id
                    HAVING COUNT(t.id) >= 80
                );
```

Result Grid

id
185
289
318
454

8 10:05:58 SELECT u.id FROM users AS u WHERE EXISTS(SELECT t.user_id FROM transactions AS t WHERE... 4 row(s) returned 0.094 sec / 0.000 sec

Level 1

Exercise 2

Show the average amount per IBAN of credit cards from the company Donec Ltd. Use at least 2 tables.

```sql
SELECT cc.iban AS iban, ROUND(AVG(t.amount),2) AS average
FROM transactions AS t
JOIN credit_cards AS cc
ON t.card_id = cc.id
JOIN companies AS c
ON t.company_id = c.company_id
WHERE c.company_name = 'Donec Ltd'
AND t.declined = 0
GROUP BY cc.iban
ORDER BY average;
```

Result Grid | Filter Rows:

iban	average
XX2957431476652107...	3.83
XX5151076178352999...	6.90
HR1647261369237 56...	6.90
XX8238042338901741...	9.24
XX1829176385104138...	13.14
XX2463215962424695...	14.10
XX1733698916552286...	14.57
XX8272625659136146...	20.92

9 10:12:58 SELECT cc.iban AS iban, ROUND(AVG(t.amount),2) AS average FROM transactions AS t JOIN credit... 370 row(s) returned 0.047 sec / 0.000 sec

I've considered 'not declined' transactions to calculate the real average per IBAN (I'll maintain the same criterium throughout all the document). And I've also decided to order the results per average to facilitate interpretation of data.

Level 2

Exercise 1

Create a new table that reflects the status of credit cards: if the last three transactions were declined, the status is inactive; if at least one was not declined, the status is active.

```
CREATE TABLE IF NOT EXISTS card_status (
WITH ranked_transactions AS (
        SELECT t.id, t.card_id, t.timestamp, t.declined, ROW_NUMBER() OVER (
                                                                PARTITION BY t.card_id
                                                                ORDER BY t.timestamp DESC
                                                                ) AS row_num
        FROM transactions AS t
                            ),
last3transactions AS    (
        SELECT rt.id, rt.card_id, rt.timestamp, rt.declined
        FROM ranked_transactions AS rt
        WHERE rt.row_num <= 3
                        )
SELECT l3t.card_id, CASE
                        WHEN SUM(declined) = 3 THEN 'inactive'
                        ELSE 'active'
                    END AS card_status
FROM last3transactions AS l3t
GROUP BY l3t.card_id
);
```

```
10  10:16:02  CREATE TABLE IF NOT EXISTS card_status ( WITH ranked_transactions AS( SELECT t.id, t.card_id...  5000 row(s) affected Records: 5000  Duplicates: 0  Warnings: 0  0.953 sec
```

I visualize the new table card_status:

```
SELECT * FROM store.card_status;
```

Result Grid | Filter R

card_id	card_status
CcS-4857	active
CcS-4858	active
CcS-4859	active
CcS-4860	active
CcS-4861	active
CcS-4862	active
CcS-4863	active

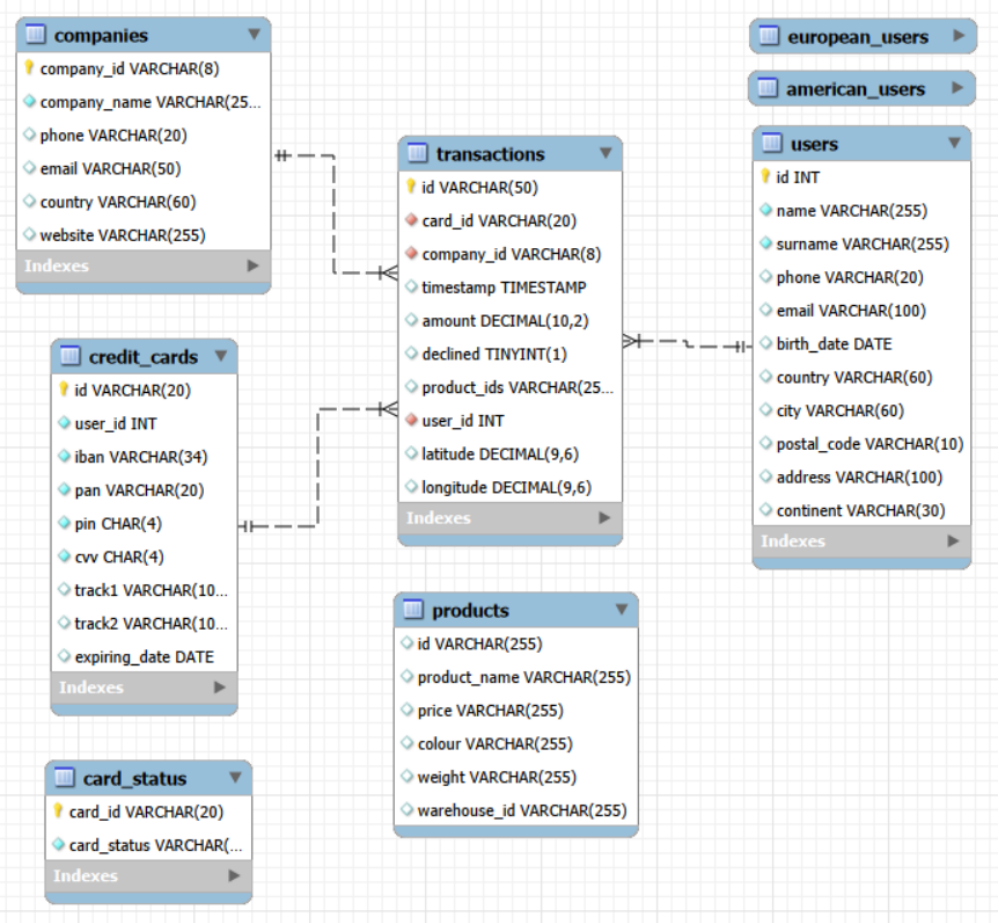
```
11  10:16:58  SELECT * FROM store.card_status  5000 row(s) returned  0.015 sec / 0.000 sec
```

I define the PK of the new table:

```
329 •   ALTER TABLE card_status
330         MODIFY COLUMN card_id VARCHAR(20) PRIMARY KEY;
```

```
12  10:20:15  ALTER TABLE card_status MODIFY COLUMN card_id VARCHAR(20) PRIMARY KEY   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.360 sec
```

Now, the **schema** looks like this:



Based on this table, answer:

Exercise 1

How many cards are active?

```sql
SELECT COUNT(cs.card_id) AS active_cards
FROM card_status AS cs
WHERE cs.card_status = 'active';
```

Result Grid

active_cards
4995

```
13 10:23:42 SELECT COUNT(cs.card_id) AS active_cards FROM card_status AS cs WHERE cs.card_status = 'ac... 1 row(s) returned 0.015 sec / 0.000 sec
```

Level 3

Create a new table that will allow us to join the data from the new products.csv file with the existing database, considering that the transaction table contains product_ids.

Before creating the new table, I make the necessary modifications to the table products:

I remove the symbol $ from the values in price, before changing the datatype from VARCHAR(255) to DECIMAL(10,2), first disactivating the safe mode.

```sql
SET SQL_SAFE_UPDATES = 0;
UPDATE products
SET price = REPLACE(price, '$', '');
```

```
16 10:32:52 SET SQL_SAFE_UPDATES = 0 0 row(s) affected 0.031 sec
17 10:32:55 UPDATE products SET price = REPLACE(price, '$', '') 100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0 0.125 sec
```

In transactions, as the product_ids are separated by a comma and a space after the comma, I execute this query to eliminate the spaces and be able then to convert the chain of values. Then I activate the safe mode again.

```sql
UPDATE transactions
SET product_ids = REPLACE(product_ids, ', ', ',');
SET SQL_SAFE_UPDATES = 1;
```

```
19 10:35:23 UPDATE transactions SET product_ids = REPLACE(product_ids, ', ', ',') 75993 row(s) affected Rows matched: 100000 Changed: 75993 Warnings: 0 2.797 sec
20 10:36:03 SET SQL_SAFE_UPDATES = 1 0 row(s) affected 0.016 sec
```

Together with price modified to DECIMAL(10,2), I change colour datatype to VARCHAR(10) because colour codes are standard (I left some extra characters, just in case).

In the case of the variable warehouse, I've noticed there're some registers that may be inconsistent: WH-4 and WH—4 (the same with numbers 3, 2 and 1). As I don't know whether they refer to the same warehouse or not, I decide to keep the values as they are, and just modify the length of the length allowed. In a real-life situation, I'd consult with my client to have more accurate information.

```
359 •   ALTER TABLE products
360         MODIFY COLUMN id INT PRIMARY KEY,
361         MODIFY COLUMN price DECIMAL (10,2),
362         MODIFY COLUMN colour VARCHAR(10),
363         MODIFY COLUMN warehouse_id VARCHAR(20);
```

```
26  10:58:34  ALTER TABLE products MODIFY COLUMN id INT PRIMARY KEY,  MODIFY COLUMN price DECI...  100 row(s) affected Records: 100  Duplicates: 0  Warnings: 0  0.281 sec
```

The new table allowing me to relate transactions and products should have the id of the transaction and the id of the products, each product appearing individually in each row. For that, I need to extract the values in transactions.product_ids.

I create a table called transactions_products and I use JSON_TABLE to convert a chain into an array, and load the data into the new table:

```
364  CREATE TABLE transactions_products AS (
365  SELECT t.id AS transaction_id, value AS product_id
366  FROM transactions AS t
367  JOIN JSON_TABLE (
368      CONCAT('["', REPLACE(t.product_ids, ',', '","'), '"]'), '$[*]' COLUMNS (
369                                                                  value INT PATH '$'
370                                                                  )
371                  ) AS jt
372  WHERE t.product_ids IS NOT NULL
373                                      );
```

Result Grid | Filter Rows: | Edit:

transaction_id	product_id
00043A49-2949-494B-A5DD-A5BAE3BB19DD	16
00043A49-2949-494B-A5DD-A5BAE3BB19DD	26
00043A49-2949-494B-A5DD-A5BAE3BB19DD	87
00043A49-2949-494B-A5DD-A5BAE3BB19DD	97
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	66
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	69
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	87
00045D6B-ED2E-4F2F-8186-CEE074D875D0	11
00045D6B-ED2E-4F2F-8186-CEE074D875D0	16
00045D6B-ED2E-4F2F-8186-CEE074D875D0	30

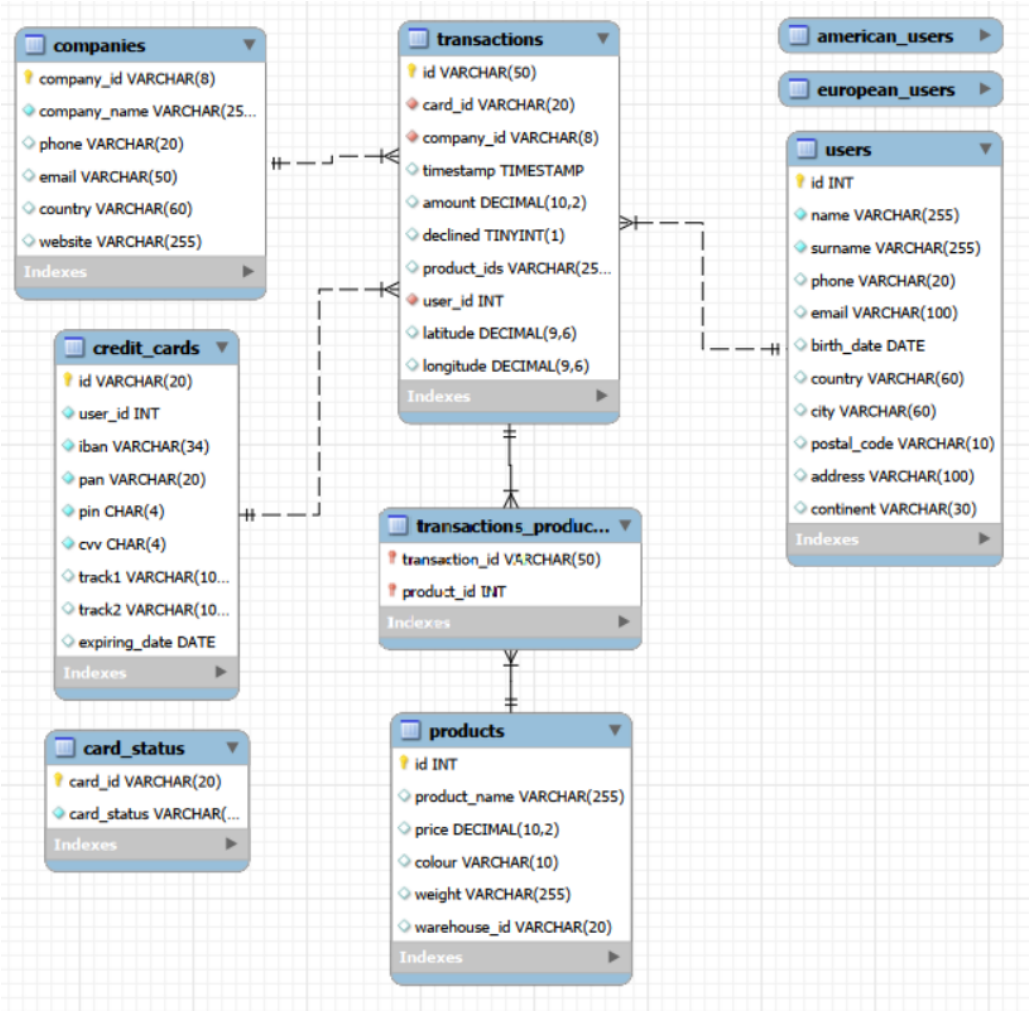
```
23  10:49:05  CREATE TABLE transactions_products AS ( SELECT t.id AS transaction_id, value AS product_id FRO...  253391 row(s) affected Records: 253391  Duplicates: 0  Warnings: 0  3.750 sec
```

Finally, I crate the pk with both variables because each of the combinations is unique and define the fk constraints to relate the new table to the schema:

```
376  ALTER TABLE transactions_products
377  ADD PRIMARY KEY (transaction_id, product_id),
378  ADD CONSTRAINT fk_transactions_products_products
379      FOREIGN KEY (product_id) REFERENCES products(id),
380  ADD CONSTRAINT fk_transactions_products_transactions
381      FOREIGN KEY (transaction_id) REFERENCES transactions(id);
```

```
27  10:59:14  ALTER TABLE transactions_products ADD PRIMARY KEY (transaction_id, product_id), ADD CONST...  253391 row(s) affected Records: 253391  Duplicates: 0  Warnings: 0  4.921 sec
```

Now, the **schema** looks like this:



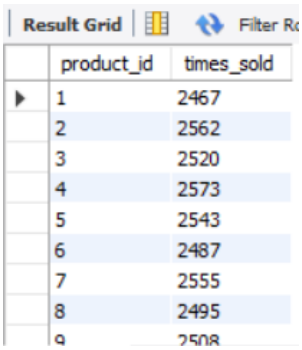
Generate the following query:

Exercise 1

We need to know how many times each product has been sold.

```
387 •   SELECT tp.product_id, COUNT(tp.product_id) AS times_sold
388     FROM transactions_products AS tp
389     INNER JOIN transactions AS t
390     ON tp.transaction_id = t.id
391     WHERE t.declined = 0
392     GROUP BY tp.product_id
393     ORDER BY tp.product_id;
```

I've considered transactions that haven't been declined to obtain effectively sold products.



Result Grid | Filter Ro

product_id	times_sold
1	2467
2	2562
3	2520
4	2573
5	2543
6	2487
7	2555
8	2495
9	2508

28 11:02:33 SELECT tp.product_id, COUNT(tp.product_id) AS times_sold FROM transactions_products AS tp INN... 100 row(s) returned 0.468 sec / 0.000 sec