

# Task S4.01. Database Creation

## LEVEL 1

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

- Created the database sprintstar:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Administration' and 'Schemas' are selected. The main pane displays the command: 'CREATE DATABASE sprintstar;'. Below the command, the 'Action Output' pane shows a single row: '1 11:31:24 CREATE DATABASE sprintstar' with a status of '1 row(s) affected'.

- Created the table companies:

The screenshot shows the MySQL Workbench interface. The left sidebar shows the 'sprintstar' schema with 'Tables' expanded, showing 'companies' selected. The main pane displays the creation of the 'companies' table and its data loading. The command starts with '#LEVEL 1' and includes:  
1 #LEVEL 1  
2  
3 • CREATE DATABASE sprintstar;  
4  
5 • CREATE TABLE companies (  
6 company\_id VARCHAR(100) PRIMARY KEY,  
7 company\_name VARCHAR(100),  
8 phone VARCHAR(50),  
9 email VARCHAR(100),  
10 country VARCHAR(100),  
11 website VARCHAR(100)  
12 );  
13  
14 • LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona\_activa/companies.csv'  
15 INTO TABLE companies  
16 FIELDS TERMINATED BY ','  
17 ENCLOSED BY '\"'  
18 LINES TERMINATED BY '\n'  
19 IGNORE 1 ROWS;  
20

The 'Action Output' pane shows two rows of logs: '1 15:39:25 CREATE TABLE companies (' and '2 15:39:29 LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona\_activa/companies.csv' INTO TABLE companies FILE...'. Both rows show a status of '0 row(s) affected'.

-Created the table credit\_cards:

The screenshot shows the MySQL Workbench interface with the 'Sprint4\*' connection selected. In the left sidebar, under the 'sprintstar' schema, the 'Tables' section contains 'companies', 'credit\_cards', and 'Transactions'. The 'credit\_cards' table has columns: id, user\_id, iban, pan, pin, cvv, track1, track2, and expiring\_date. The main pane displays the SQL code for creating the table and loading data from a CSV file:

```
20
21
22
23 • CREATE TABLE IF NOT EXISTS credit_cards (
24     id VARCHAR(20) PRIMARY KEY,
25     user_id INT,
26     iban VARCHAR(50) ,
27     pan VARCHAR(19),
28     pin VARCHAR(4),
29     cvv VARCHAR(4),
30     track1 VARCHAR(255),
31     track2 VARCHAR(255),
32     expiring_date VARCHAR(20)
33 );
34
35 • LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/credit_cards.csv'
36     INTO TABLE credit_cards
37     FIELDS TERMINATED BY ','
38     ENCLOSED BY ""
39     LINES TERMINATED BY '\n'
40     IGNORE 1 ROWS;
41
```

The 'Action Output' pane shows two successful operations:

Action	Time	Response
CREATE TABLE IF NOT EXISTS credit_cards ( id VARCHAR(20) PRIMARY KEY, user_id INT, iban VAR... )	15:43:10	0 row(s) affected
LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/credit_cards.csv' INTO TABLE credit_cards...	15:43:13	5000 row(s) affected Records: 5000 Deleted: 0 Skipped: 0 Warnings: 0

-Created the table products:

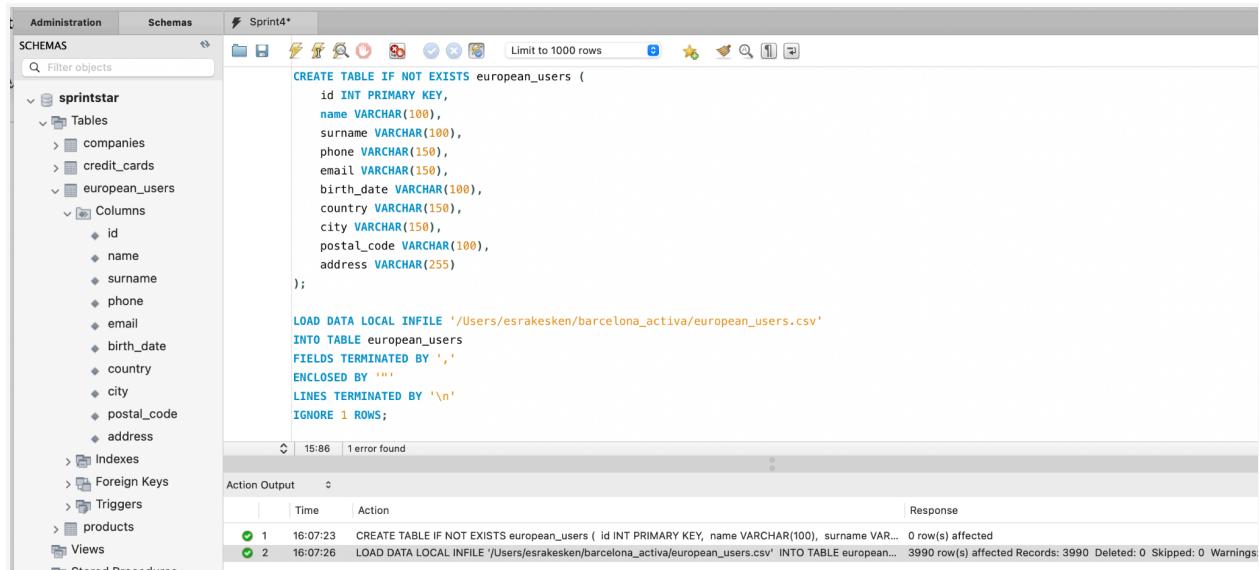
The screenshot shows the MySQL Workbench interface with the 'Sprint4\*' connection selected. In the left sidebar, under the 'sprintstar' schema, the 'Tables' section contains 'companies', 'credit\_cards', 'products', and 'Transactions'. The 'products' table has columns: id, product\_name, price, colour, weight, and warehouse\_id. The main pane displays the SQL code for creating the table and loading data from a CSV file:

```
41
42
43 • CREATE TABLE IF NOT EXISTS products (
44     id INT PRIMARY KEY,
45     product_name VARCHAR(50),
46     price DECIMAL(10,2),
47     colour VARCHAR(20),
48     weight FLOAT,
49     warehouse_id VARCHAR(25)
50 );
51
52 • LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/products (1).csv'
53     INTO TABLE products
54     FIELDS TERMINATED BY ','
55     ENCLOSED BY ""
56     LINES TERMINATED BY '\n'
57     IGNORE 1 ROWS
58     (id, product_name, @price, colour, weight, warehouse_id)
59     SET price = REPLACE(@price, '$', '');
60
61
```

The 'Action Output' pane shows two successful operations:

Action	Time	Response
CREATE TABLE IF NOT EXISTS products ( id INT PRIMARY KEY, product_name VARCHAR(50), price D... )	16:00:17	0 row(s) affected
LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/products (1).csv' INTO TABLE products FILE...	16:00:21	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

-Created the tables american\_users and european\_users:



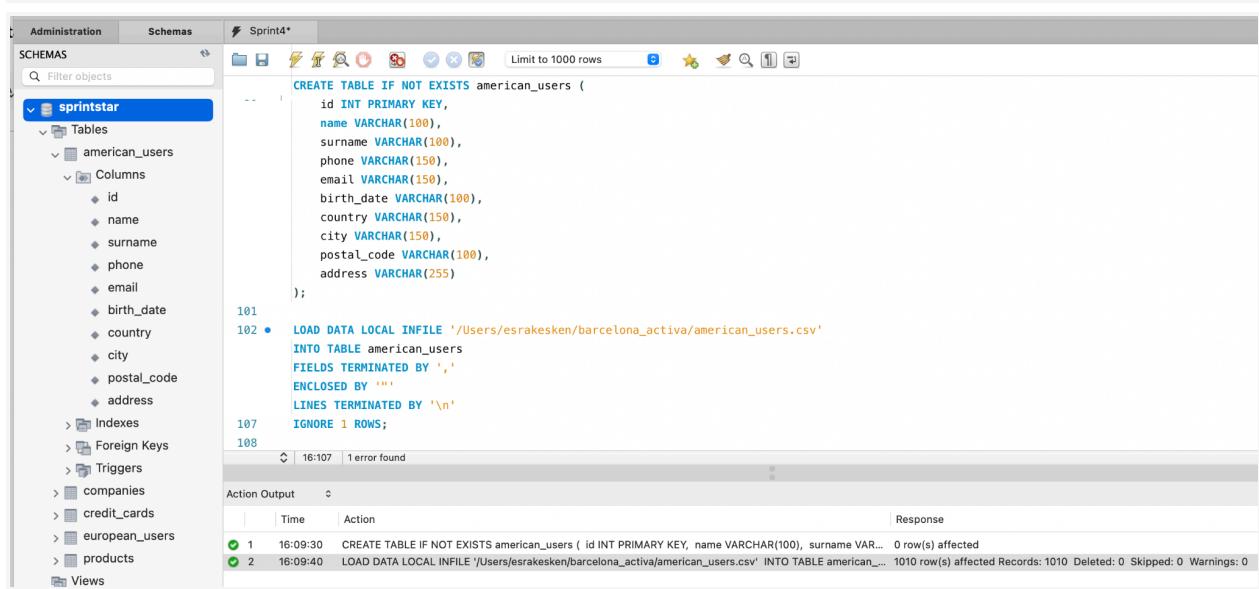
Screenshot of MySQL Workbench showing the creation of the `european_users` table. The schema `sprintstar` is selected. The table `european_users` is created with the following SQL:

```
CREATE TABLE IF NOT EXISTS european_users (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);

LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/european_users.csv'
INTO TABLE european_users
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Action Output shows two successful operations:

Time	Action	Response
16:07:23	CREATE TABLE IF NOT EXISTS european_users ( id INT PRIMARY KEY, name VARCHAR(100), surname VAR... 0 row(s) affected	
16:07:26	LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/european_users.csv' INTO TABLE european_... 3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0	

Screenshot of MySQL Workbench showing the creation of the `american_users` table. The schema `sprintstar` is selected. The table `american_users` is created with the following SQL:

```
CREATE TABLE IF NOT EXISTS american_users (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);

101
102 • LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/american_users.csv'
INTO TABLE american_users
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
107 IGNORE 1 ROWS;
108
```

Action Output shows two successful operations:

Time	Action	Response
16:09:30	CREATE TABLE IF NOT EXISTS american_users ( id INT PRIMARY KEY, name VARCHAR(100), surname VAR... 0 row(s) affected	
16:09:40	LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/american_users.csv' INTO TABLE american_... 1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0	

Now, to combine all users into one table, we need to create a new users table and insert all data from two tables and then delete the two tables:

```

108
109 • CREATE TABLE IF NOT EXISTS users (
110     id INT PRIMARY KEY AUTO_INCREMENT,
111     source_id INT NOT NULL,
112     region ENUM('american','european') NOT NULL,
113     name VARCHAR(100),
114     surname VARCHAR(100),
115     phone VARCHAR(150),
116     email VARCHAR(150),
117     birth_date VARCHAR(100),
118     country VARCHAR(150),
119     city VARCHAR(150),
120     postal_code VARCHAR(100),
121     address VARCHAR(255)
122 );
123
124 • INSERT INTO users (source_id, region, name, surname, phone, email, birth_date, country, city, postal_code, address)
125     SELECT id, 'american', name, surname, phone, email, birth_date, country, city, postal_code, address FROM american_users
126     UNION
127     SELECT id, 'european', name, surname, phone, email, birth_date, country, city, postal_code, address FROM european_users;
128
129
130
131
132

```

Action Output

Time	Action	Response
1 10:17:54	CREATE TABLE IF NOT EXISTS users ( id INT PRIMARY KEY AUTO_INCREMENT, source_id INT NOT NULL,...	0 row(s) affected
2 10:18:04	INSERT INTO users (source_id, region, name, surname, phone, email, birth_date, country, city, postal_code, address FROM american_users	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

```

CREATE TABLE IF NOT EXISTS users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    source_id INT NOT NULL,
    region ENUM('american','european') NOT NULL,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);

• INSERT INTO users (source_id, region, name, surname, phone, email, birth_date, country, city, postal_code, address)
SELECT id, 'american', name, surname, phone, email, birth_date, country, city, postal_code, address FROM american_users
UNION
SELECT id, 'european', name, surname, phone, email, birth_date, country, city, postal_code, address FROM european_users;

• DROP TABLE american_users;
• DROP TABLE european_users;

```

Action Output

Time	Action	Response
1 10:24:25	DROP TABLE american_users	0 row(s) affected
2 10:24:29	DROP TABLE european_users	0 row(s) affected

-I created the **transactions** table and set up its relationships with the relevant tables in the database:

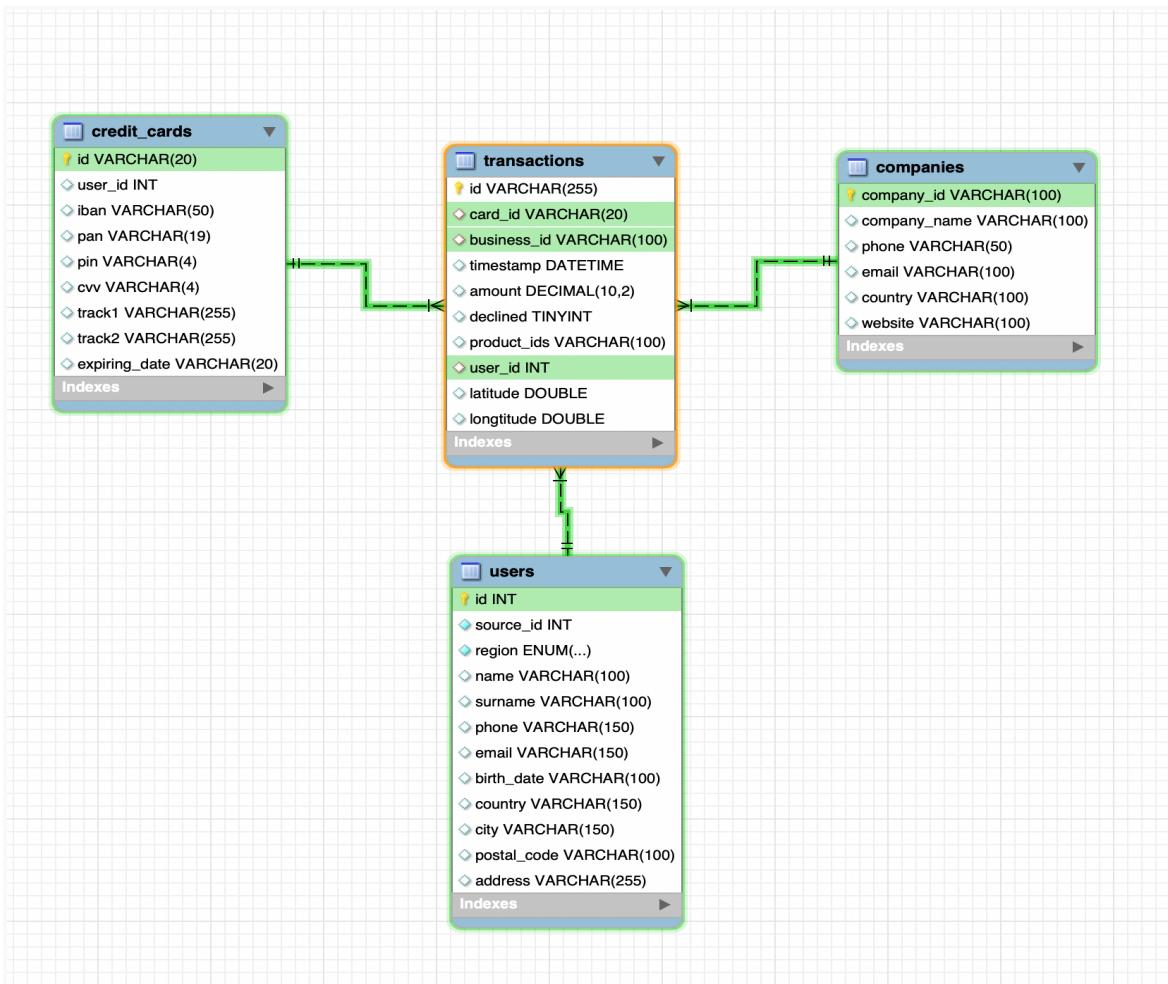
The screenshot shows the MySQL Workbench interface. In the top-left pane, the schema is selected as 'sprintstar'. The 'Tables' section contains 'companies', 'credit\_cards', 'products', and 'transactions'. The 'Columns' section for 'transactions' lists fields: id, card\_id, business\_id, timestamp, amount, declined, product\_ids, user\_id, latitude, and longitude. Below this, the 'Script' tab displays the SQL code for creating the table and loading data from a CSV file. The 'Logs' tab shows two successful operations: the creation of the table and the loading of 1000000 rows of data.

```

CREATE TABLE IF NOT EXISTS `transactions` (
  `id` VARCHAR(255) PRIMARY KEY,
  `card_id` VARCHAR(20),
  `business_id` VARCHAR(100),
  `timestamp` DATETIME,
  `amount` DECIMAL(10,2),
  `declined` TINYINT,
  `product_ids` VARCHAR(100),
  `user_id` INT,
  `latitude` DOUBLE,
  `longitude` DOUBLE,
  FOREIGN KEY (card_id) REFERENCES credit_cards(id),
  FOREIGN KEY (business_id) REFERENCES companies(company_id),
  FOREIGN KEY (user_id) REFERENCES users(id)
);

LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ','
ENCLOSED BY "'"
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
  
```

Action Output	Time	Action	Response
1	10:25:24	CREATE TABLE IF NOT EXISTS transactions ( `id` VARCHAR(255) PRIMARY KEY, `card_id` VARCHAR(20),...	0 row(s) affected
2	10:25:45	LOAD DATA LOCAL INFILE '/Users/esrakesken/barcelona_activa/transactions.csv' INTO TABLE transactions...	1000000 row(s) affected Records: 1000000 Deleted: 0 Skipped: 0 Warnings: 0



**Exercise 1:** Perform a subquery 😊 that shows all users with more than 80 transactions, using at least 2 tables.

```

163  #Exercise 1:Perform a subquery that shows all users with more than 80 transactions, using at least 2 tables.
164 •  SELECT id,
165      name,
166      surname
167  FROM users
168  WHERE id IN (
169      SELECT user_id
170      FROM transactions
171      GROUP BY user_id
172      HAVING COUNT(id) > 80
173  );
    
```

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code above is pasted into the editor. Below the editor is the 'Result Grid' pane, which displays a table with three columns: 'id', 'name', and 'surname'. The data shows 24 rows of user information. At the bottom of the Result Grid, it says 'users 24'. Below the Result Grid is the 'Action Output' pane, which shows a single query execution log entry:

Action	Time	Query	Response
1	17:35:14	SELECT id, name, surname FROM users WHERE id IN ( SELECT user_id FROM transactions... )	4 row(s) returned

**Exercise 2:** Show the average amount per IBAN of credit cards in the company Donec Ltd, using at least 2 tables.

```

172
173  #Exercise 2: Show the average amount per IBAN of credit cards in the company Donec Ltd, using at least 2 tables.
174 •  SELECT c.company_name,
175      cc.iban,
176      AVG(t.amount) AS avg_amount
177  FROM transactions t
178  JOIN credit_cards cc ON t.card_id = cc.id
179  JOIN companies c ON t.business_id = c.company_id
180  WHERE c.company_name = 'Donec Ltd'
181  GROUP BY cc.iban
182  ORDER BY avg_amount DESC;
    
```

The screenshot shows the MySQL Workbench interface with the SQL editor tab open. The code above is pasted into the editor. Below the editor is the 'Result Grid' pane, which displays a table with three columns: 'company\_name', 'iban', and 'avg\_amount'. The data shows 20 rows of credit card information for the company 'Donec Ltd'. At the bottom of the Result Grid, it says 'Result 20'. Below the Result Grid is the 'Action Output' pane, which shows a single query execution log entry:

Action	Time	Query	Response
1	12:18:24	SELECT c.company_name, cc.iban, AVG(t.amount) AS avg_amount FROM transactions t JOIN credit_cards cc ON t.card_id = cc.id JOIN companies c ON t.business_id = c.company_id WHERE c.company_name = 'Donec Ltd' GROUP BY cc.iban ORDER BY avg_amount DESC;	371 row(s) returned

## LEVEL 2

Create a new table that reflects the status of credit cards based on whether the last three transactions were declined, and generate the following query:

The screenshot shows the MySQL Workbench interface. The left sidebar shows the schema 'sprintstar' with its tables, columns, indexes, and foreign keys. The main pane displays the SQL code for creating the 'card\_status' table and inserting data based on the last three transactions. The action output shows two successful operations: creating the table and inserting data.

```

#LEVEL 2
#Create a new table that reflects the status of credit cards based on whether the last three transactions were declined
CREATE TABLE IF NOT EXISTS card_status (
    id VARCHAR(20) PRIMARY KEY,
    status VARCHAR(20)
);

INSERT INTO card_status (id, status)
SELECT card_id,
CASE
    WHEN SUM(declined) = 3 THEN 'declined_last_3'
    ELSE 'active'
END AS status
FROM (
    SELECT card_id,
    declined,
    ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS rn
    FROM transactions
) AS ranked
WHERE rn <= 3
GROUP BY card_id;
  
```

Action	Time	Response
CREATE TABLE IF NOT EXISTS card_status ( id VARCHAR(20) PRIMARY KEY, status VARCHAR(20) )	22:25:16	0 row(s) affected
INSERT INTO card_status (id, status) SELECT card_id, CASE WHEN SUM(declined) = 3 THEN 'de... rn <= 3 GROUP BY card_id;	22:26:10	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

### Exercise 1: How many cards are active?

The screenshot shows the MySQL Workbench interface. The left sidebar shows the schema 'sprintstar' with its tables, columns, indexes, and foreign keys. The main pane displays the SQL code to count active cards. The result grid shows the count of active cards as 4995. The action output shows four successful operations: creating the table, inserting data, selecting the count, and another select statement.

```

GROUP BY card_id;

#Exercise 1:How many cards are active?
SELECT COUNT(*) AS active_card_count
FROM card_status
WHERE status = 'active';
  
```

Result Grid
active_card_count
4995

Action	Time	Response
CREATE TABLE IF NOT EXISTS card_status ( id VARCHAR(20) PRIMARY KEY, status VARCHAR(20) )	22:25:16	0 row(s) affected
INSERT INTO card_status (id, status) SELECT card_id, CASE WHEN SUM(declined) = 3 THEN 'de... rn <= 3 GROUP BY card_id;	22:26:10	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
SELECT COUNT(*) AS active_card_count FROM sprintstar.card_status LIMIT 0, 5000	22:28:32	5000 row(s) returned
SELECT COUNT(*) AS active_card_count FROM card_status WHERE status = 'active' LIMIT 0, 5000	23:33:02	1 row(s) returned

-Set up a relationship between the credit\_cards and the table card\_status:

The screenshot shows the MySQL Workbench interface. The left sidebar shows the schema 'HEMAS' with its tables, columns, indexes, and foreign keys. The main pane displays the SQL code to set up a relationship between the 'credit\_cards' table and the 'card\_status' table. The action output shows one successful operation: adding a foreign key constraint.

```

WHERE status = 'active';

#Set up a relationship between the credit_cards and the table card_status:
ALTER TABLE credit_cards
ADD FOREIGN KEY(id) REFERENCES card_status(id);
  
```

Action	Time	Response
ALTER TABLE credit_cards ADD FOREIGN KEY(id) REFERENCES card_status(id)	17:46:09	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

## LEVEL 3

Create a table with which we can join the data from the new products.csv file with the created database, taking into account that from the transaction, you have product\_ids. Generate the following query:

We need to create a bridge table to join the data from the **products** table and establish relationships between the relevant tables. To achieve this, we also need to split the values in the **product\_ids** column, as it contains multiple entries.

The screenshot shows the MySQL Workbench interface with the 'Sprint4\*' schema selected. In the left pane, the 'Tables' section shows the 'transaction\_products' table. In the main pane, the SQL editor contains the following code:

```
#LEVEL 3
CREATE TABLE transaction_products (
    transaction_id VARCHAR(255),
    product_id INT,
    PRIMARY KEY (transaction_id,product_id)
);
```

The status bar at the bottom indicates '0 row(s) affected'.

The screenshot shows the MySQL Workbench interface with the 'Sprint4\*' schema selected. In the left pane, the 'Tables' section shows the 'transaction\_products' table. In the main pane, the SQL editor contains a complex recursive query:

```
INSERT INTO transaction_products (transaction_id, product_id)
WITH RECURSIVE split_products AS (
    SELECT
        id AS transaction_id,
        TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) AS product_id,
        SUBSTRING(product_ids, LENGTH(SUBSTRING_INDEX(product_ids, ',', 1)) + 2) AS rest
    FROM transactions
    WHERE product_ids IS NOT NULL AND product_ids > ''
)
UNION ALL
SELECT
    transaction_id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)) AS product_id,
    SUBSTRING(rest, LENGTH(SUBSTRING_INDEX(rest, ',', 1)) + 2) AS rest
FROM split_products
WHERE rest IS NOT NULL AND rest > ''
)
SELECT
    transaction_id,
    CAST(product_id AS UNSIGNED) AS product_id
FROM split_products;
```

The status bar at the bottom indicates '253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0'.

Screenshot of MySQL Workbench showing the results of a query on the `sprintstar.transaction_products` table.

**Schema Browser:**

- SCHEMAS: sprintstar
- Tables:
  - transaction\_products

**Result Grid:**

transaction_id	product_id
00043A49-2949-494B-ASD0-A5BAE3B819DD	16
00043A49-2949-494B-ASD0-A5BAE3B819DD	26
00043A49-2949-494B-ASD0-A5BAE3B819DD	87
00043A49-2949-494B-ASD0-A5BAE3B819DD	97
000447FE-B650-4DCF-85DE-C7E0EE1CAAD	66
000447FE-B650-4DCF-85DE-C7E0EE1CAAD	69
000447FE-B650-4DCF-85DE-C7E0EE1CAAD	87
00045D6B-ED2E-4F2F-81B6-CEE074D875D0	11
00045D6B-ED2E-4F2F-81B6-CEE074D875D0	16
00045D6B-ED2E-4F2F-81B6-CEE074D875D0	30
00045D6B-ED2E-4F2F-81B6-CEE074D875D0	81
000481C3-1C26-4FFC-83A0-4CD0EB04BB0	72
00051AA4-9CBE-426B-B070-C30082A1B3E2	18
0008A312-EDFE-4AF-BC99-E9C92EC3CA4D	19
0008A312-EDFE-4AF-BC99-E9C92EC3CA4D	33
0008A312-EDFE-4AF-BC99-E9C92EC3CA4D	35

**Action Output:**

Time	Action	Response
1 11:45:04	INSERT INTO transaction_products (transaction_id, product_id) WITH RECURSIVE split_products AS (	SE... 253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0
2 11:45:40	SELECT * FROM sprintstar.transaction_products LIMIT 0, 10000	10000 row(s) returned

-To set up a relationship between the table `transaction_products` and the relevant tables `transactions` and `products`:

Screenshot of MySQL Workbench showing the execution of an ALTER TABLE statement to add foreign key constraints to the `transaction_products` table.

**Schema Browser:**

- SCHEMAS: sprintstar
- Tables:
  - transaction\_products

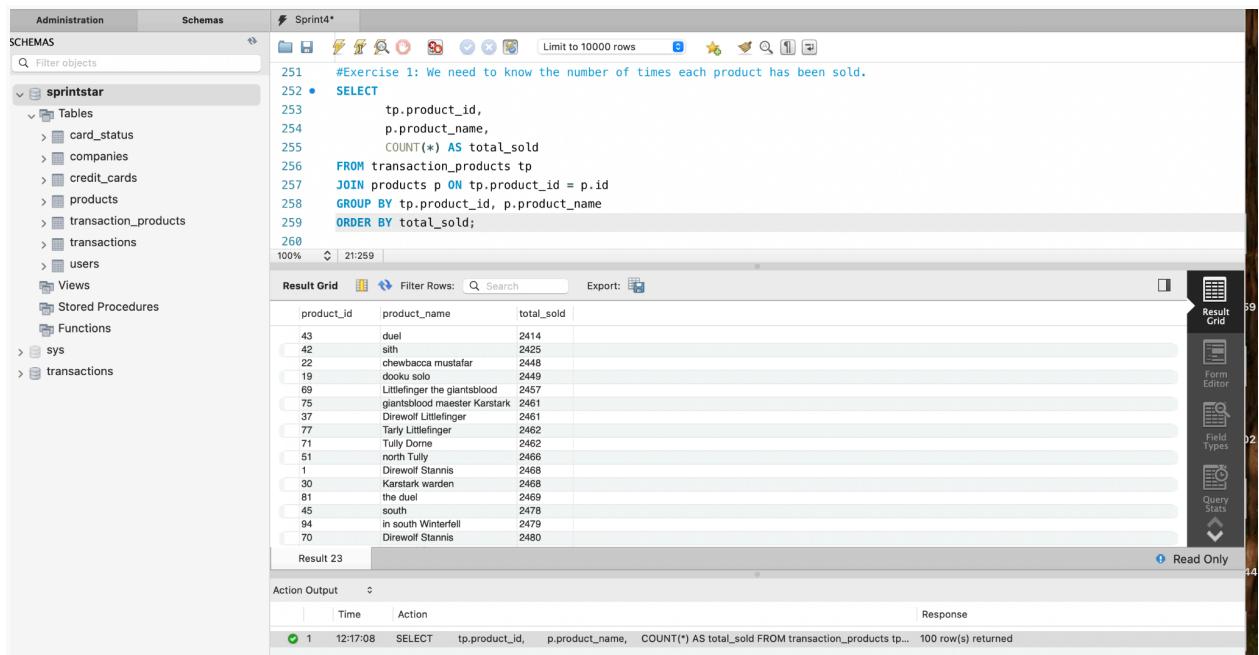
**Result Grid:**

```
247 • ALTER TABLE transaction_products
248 ADD FOREIGN KEY (transaction_id) REFERENCES transactions(id),
249 ADD FOREIGN KEY (product_id) REFERENCES products(id);
250
251
```

**Action Output:**

Time	Action	Response
1 11:53:46	ALTER TABLE transaction_products ADD FOREIGN KEY (transaction_id) REFERENCES transactions(id), ADD...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

**Exercise 1:** We need to know the number of times each product has been sold.



The screenshot shows a MySQL Workbench interface with the following details:

- Schemas:** Administration, Schemas, Sprint4\*
- Schema:** sprintstar
- Tables:** card\_status, companies, credit\_cards, products, transaction\_products, transactions, users
- Views:** None
- Stored Procedures:** None
- Functions:** None
- sys:** None
- transactions:** None

**Query:**

```
251 #Exercise 1: We need to know the number of times each product has been sold.
252 • SELECT
253     tp.product_id,
254     p.product_name,
255     COUNT(*) AS total_sold
256     FROM transaction_products tp
257     JOIN products p ON tp.product_id = p.id
258     GROUP BY tp.product_id, p.product_name
259     ORDER BY total_sold;
260
```

**Result Grid:**

product_id	product_name	total_sold
43	duel	2414
42	sith	2425
22	chewbacca mustafar	2448
19	dooku solo	2449
69	Littlefinger the giantsblood	2457
75	giantsblood maester Karstark	2461
37	Direwolf Littlefinger	2461
77	Tarly Littlefinger	2462
71	Tully Dorne	2462
51	north Tully	2466
1	Direwolf Stannis	2468
30	Karstark warden	2468
81	the duel	2469
45	south	2478
94	In south Winterfall	2479
70	Direwolf Stannis	2480

**Action Output:**

Time	Action	Response
12:17:08	SELECT tp.product_id, p.product_name, COUNT(*) AS total_sold FROM transaction_products tp...	100 row(s) returned

And the diagram between the tables:

