Question #: 1

The database stores information about customers, credit cards, merchants, and transactions.
The database stores the unique social security number, first and last name, and country of customers.
The database stores the unique number of each credit card and the social security number of its unique cardholder.
The database stores the unique code, name, and country of each merchant.
For each transaction, the database stores the credit card number used, the merchant code, the date and time of the transaction, and the amount of the transaction.
Create the database and populate it with the example instance using the files available in Canvas Files >Cases >Credit Cards.

Your answers should work for the example and other valid instances of the database. Do not give more answers than required. Unnecessary comments are penalised. Answers in SQL that yield syntax errors or do not produce the expected result with this example instance and PostgreSQL may be awarded zero mark. Prefer simple SQL queries to aggregate and nested queries. Do not include unnecessary SQL constructs. Do not use SQL constructs not discussed in the lecture and tutorials. Simplify the queries with the knowledge of the schema and constraints.

Indicate below whether you have read this description of the case.

Question #: 3
Give the SQL code for an insertion to the table merchants that violates a not null constraint.
Indicate "Impossible" if it is not possible.

Rationale:

INSERT INTO merchants VALUES ('91-3366100', null, 'Thailand');

Question #: 4
Give the SQL code for an update to the table transactions that violates a constraint on the table merchants
Indicate "Impossible" if it is not possible.

Rationale: Impossible: the foreign key is on the table <tt>transaction</tt>.

Question #: 5
Give the SQL code for an insertion into the table merchants that violates a primary key constraint.
Indicate "Impossible" if it is not possible.

Rationale:

INSERT INTO merchants VALUES ('91-3366138', 'Central', 'Thailand');

Question #: 6
Give the SQL code for an update to the table transactions that violates a primary key constraint. Indicate "Impossible" if it is not possible.

Rationale:
UPDATE transactions SET identifier = 1;

Question #: 7

Find the last and first names of the different Singaporean customers. Print the result in alphabetical order of the last and first names.

Rationale:
```
SELECT  c.last_name, c.first_name
FROM customers c
WHERE c.country='Singapore'
ORDER BY c.last_name, c.first_name;
```

Question #: 8

For each Singaporean customer, find his or her first and last name and total expenditure. Implicitly ignore customers who did not use their credit cards or do not have a credit card.

Rationale:
```
SELECT  c.first_name, c.last_name, SUM(t.amount)
FROM customers c, transactions t, credit_cards cc
WHERE c.ssn = cc.ssn AND cc.number = t.number AND c.country='Singapore'
GROUP BY c.ssn, c.first_name, c.last_name;
```

Question #: 9

Find the social security number of the different customers who purchased something on Christmas day 2017 with their Visa credit card (the credit card type is "visa".)

Rationale:

```
SELECT DISTINCT  cc.ssn
FROM transactions t, credit_cards cc
WHERE cc.number = t.number AND cc.type='visa' AND
t.datetime BETWEEN '2017-12-25 00:00:00' AND '2017-12-26 00:00:00';
```

Question #: 10

For each customer and for each credit card type, find how many credit cards of that type the customer owns. Print the customer's social security number, the credit card type and the number of credit cards of the given type owned. Print zero if a customer does not own a credit card of the given type.

Rationale:

```
SELECT c1.ssn, c1.type, COUNT(cc1.number)<BR>
FROM
(SELECT DISTINCT c2.ssn, cc2.type<BR>
 FROM customers c2, credit_cards cc2) AS c1<BR>
LEFT OUTER JOIN credit_cards cc1 <BR>
 ON c1.ssn = cc1.ssn <BR>
 AND c1.type = cc1.type<BR>
GROUP BY c1.ssn, c1.type<BR>
```

Question #: 11

Find the codes and names of the different merchants who did not entertain transactions for every type of credit card. Do not use aggregate functions.

Rationale:

```
SELECT DISTINCT   m1.code, m1.name
FROM merchants m1, credit_cards cc1
WHERE NOT EXISTS (
SELECT *
FROM merchants m, transactions t, credit_cards cc
WHERE m.code = t.code AND t.number = cc.number
    AND cc.type = cc1.type AND t.code = m1.code);
```

Question #: 12

Find the first and last names of the different customers from Thailand who do not have a JCB credit card (the credit card type is "jcb"). Propose five (5) different SQL queries.

Rationale:

```
SELECT c.ssn
FROM customers c
WHERE c.country = 'Thailand'
AND c.ssn NOT IN (
SELECT cc.ssn
FROM credit_cards cc
WHERE cc.type = 'jcb');
```

```
SELECT c.ssn
FROM customers c
WHERE c.country = 'Thailand'
AND c.ssn <> ALL ( SELECT
cc.ssn
FROM credit_cards cc
WHERE cc.type = 'jcb');
```

```
SELECT c.ssn
FROM customers c
WHERE c.country = 'Thailand'
AND NOT EXISTS ( SELECT
1
FROM credit_cards cc
WHERE cc.ssn = c.ssn AND
cc.type = 'jcb');
```

```
SELECT c.ssn
FROM customers c
WHERE c.country = 'Thailand'
EXCEPT
SELECT cc.ssn
FROM credit_cards cc
WHERE cc.type = 'jcb';
```

```
SELECT c.ssnFROM customers c
LEFT OUTER JOIN  credit_cards cc
ON c.ssn = cc.ssn AND cc.type = 'jcb'
WHERE cc.ssn IS NULL
AND country = 'Thailand';
```