Project: Credit Cards

Instructions

- This project is an individual project.
- Submit your answers by **Friday 10 September 2021, 17:00** to Luminus.
- There is strictly no possibility of late submission.
- Download the SQL files
  - “`CCSchema.sql`”,
  - “`CCCustomers.sql`”,
  - “`CCCreditCards.sql`”,
  - “`CCMerchants.sql`”,
  - “`CCTransactions.sql`”, and
  - “`CCClean.sql`”

  from Luminus

  ‘`Files > Credit Cards > Code`”.
- Download the template answer file “`answers.sql`” from Luminus

  “`Files > Projects > Credit Cards`”.
- Write your answers in the indicated sections of the file “`answers.sql`” .
- Submit the completed file “`answers.sql`” to Luminus

  “`Files > Projects > Credit Cards > Submissions`”.
- Do not submit other files.
- Follow the naming requirements, include your student number and write your answers the file “`answers.sql`” as instructed.
- The SQL code that you submit should run without error in PostgreSQL version 13.

---

1. (8 points) Translate the following queries.

   (a) Find the social security number of the different customers who purchased something on Christmas day 2017 with their visa card.

Solution:

```
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';
```

There is no need to use the table "`customers`". Intersection and subqueries are unnecessarily complicated for this question.

   (b) Find the first and last names of the different customers in Singapore who own both a JCB and an Visa credit card (the credit card type is "`jcb`" or "`visa`".) Make sure that the same customer is not printed twice (note that your answer should cater for the fact that there could be, in this or in future instances of the database, different customers with the same name: each different customer must be printed.)

Solution: Result has 21 rows. The two customers called "Dukey" "Malthus" are printed.

```
SELECT c.first_name, c.last_name
FROM customers c, credit_cards cc1, credit_cards cc2
WHERE c.ssn = cc1.ssn
 AND c.ssn = cc2.ssn
 AND cc1.type = 'jcb'
 AND cc2.type = 'visa'
 AND c.country = 'Singapore'
GROUP BY c.ssn, c.first_name, c.last_name;
```

Duplicates can be properly eliminated using GROUP BY as above as well as using a subquery.

```
SELECT c1.first_name, c1.last_name
FROM customers c1
WHERE c1.ssn IN
  (SELECT c2.ssn
 FROM customers c2, credit_cards cc1, credit_cards cc2
 WHERE c2.ssn = cc1.ssn
   AND c2.ssn = cc2.ssn
   AND cc1.type = 'jcb'
   AND cc2.type = 'visa'
   AND c.country = 'Singapore');
```

One can also use a subquery with INTERSECT.
The following query is wrong: it prints "Dukey" "Malthus" only once (result has 20 rows).

```
SELECT DISTINCT c.first_name, c.last_name
FROM customers c, credit_cards cc1, credit_cards cc2
WHERE c.ssn = cc1.ssn
AND c.ssn = cc2.ssn
AND cc1.type = 'jcb'
AND cc2.type = 'visa'
AND c.country = 'Singapore'
```

   (c) For each customer, find how many credit cards the customer owns. Print the customer's social security number and the number of credit cards owned. Print zero if a customer does not own any credit card.

Solution:

```
SELECT c.ssn, COUNT(cc.number)
FROM customers c LEFT OUTER JOIN credit_cards cc ON c.ssn = cc.ssn
GROUP BY c.ssn;
```

Customer "John" "Doe" with ssn from Singapore "111-11-1111" does not own any credit card.

Add `ORDER BY COUNT(cc.number) ASC` to check the result (1301 rows)

It can also be done with two different queries and a union.

The following answer ignores the customers who do not own a credit card of a given type.

```
SELECT cc.ssn, COUNT(*)
FROM credit_cards cc
GROUP BY cc.ssn;
```

(d) 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 any credit card of the given type.

Solution:

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

Add `ORDER BY c1.ssn` to check the result (20816 rows).

It can also be done with two different queries and a union.

(e) For each country, find the number of customers from this country who purchased something from a merchant from a different country. You may ignore countries for which there is no such customer.

Solution:

```
SELECT c.country, COUNT(DISTINCT  c.ssn)
FROM customers c, merchants m, transactions t, credit_cards cc
WHERE c.ssn = cc.ssn
AND cc.number = t.number
AND m.code = t.code
AND c.country <> m.country
GROUP BY c.country;
```

`!=` works as inequality but is not standard and should not be used.

```
"Indonesia" 850
"Malaysia" 41
"Singapore" 300
"Thailand" 109
```

(f) Print the identifier of the transactions with the largest amount among all other transactions using the same type of credit card. Use aggregate queries.

Solution:

```
SELECT  t1.identifier
FROM transactions t1, credit_cards cc1
WHERE t1.number = cc1.number AND
  (cc1.type, t1.amount) IN
   (SELECT cc2.type, MAX(t2.amount)
    FROM transactions t2, credit_cards cc2
    WHERE t2.number=cc2.number
    GROUP BY cc2.type);
```

The result has 16 rows.

The following queries are also correct but extremely slow.

```
SELECT  t1.identifier
FROM transactions t1, credit_cards cc1
WHERE t1.number = cc1.number
 AND t1.amount = ALL
   (SELECT MAX(t2.amount)
    FROM transactions t2, credit_cards cc2
    WHERE t2.number=cc2.number
    AND cc1.type = cc2.type);

SELECT  t1.identifier
FROM transactions t1, credit_cards cc1
WHERE t1.number = cc1.number
 AND EXISTS
   (SELECT MAX(t2.amount)
    FROM transactions t2, credit_cards cc2
    WHERE t2.number=cc2.number
      AND cc2.type = cc1.type
    HAVING t1.amount = MAX(t2.amount));
```

(g) Print the transaction identifier of the transactions with the largest amount among all other transactions using the same type of credit card. Do not use aggregate functions and queries.

Solution:

```
SELECT  t1.identifier FROM transactions t1, credit_cards cc1
WHERE t1.number = cc1.number AND t1.amount >= ALL
  (SELECT t2.amount
   FROM transactions t2, credit_cards cc2
   WHERE t2.number = cc2.number AND cc2.type = cc1.type);
```

(h) Find the codes and names of the different merchants who did not entertain transactions for an amount larger than or equal to 888 dollars for any kind of Visa or Diners Club credit card (the credit card type contains "`visa`" or "`diners-club`") .

Solution:

```
SELECT  m.code, m.name
FROM merchants m
WHERE NOT EXISTS (
 SELECT *
 FROM transactions t, credit_cards cc
 WHERE t.code = m.code
   AND t.number = cc.number
   AND (cc.type LIKE '%visa%'
    OR cc.type LIKE '%diners-club%')
   AND t.amount >= 888);
```

or

```
SELECT  m.code, m.name
FROM merchants m
WHERE m.code NOT IN (
 SELECT t.code
 FROM transactions t, credit_cards cc
 WHERE t.number = cc.number
   AND (cc.type LIKE '%visa%'
    OR cc.type LIKE '%diners-club%')
 AND t.amount >= 888);
```

The result has 7 rows.

– END OF PAPER –