Project 2: SQL Query - Marking Scheme and Some Answers

Marking Scheme

For each question, **1.5 mark** for a correct and precise query. **1 mark** for a correct query with minor error as follows:

1. Use GROUP BY or ORDER BY unnecessarily.
2. Use OUTER JOIN unnecessarily.
3. Use a table in the JOIN clause unnecessarily.
4. Use a subquery unnecessarily.
5. Use a subquery in the FROM clause. For example:

```
SELECT * FROM
    (SELECT cc.number, cc.type
     FROM customers c, credit_cards cc
     WHERE c.ssn = cc.ssn) subquery, transactions t
     WHERE subquery.number = t.number
```

6. Use WITH clause to create a subquery unnecessarily.
7. Use GROUP BY 1 or ORDER BY 1, 2 syntax. You should specify the name of the column when you use GROUP BY and ORDER BY.
8. Comparing to a subquery without a quantified ALL or ANY especially when the subquery returns one value. For example: In the HAVING clause of Question 2c,

```
HAVING COUNT(DISTINCT cc.type) < (SELECT COUNT(DISTINCT type) FROM credit_cards);
```

   is correct but not a good practice.
9. Use '!=' instead of '<>' to denote not equal. '<>' is preferred because it follows the SQL-92 standard.
10. Forgot to put all columns in the select statement except the aggregate function in the GROUP BY clause.
11. Use GROUP BY clause when DISTINCT clause can be used.
12. Other unnecessary complex and obscure syntax.

0.5 mark for Question 2b if you use ORDER BY DESC and LIMIT 1 to get the maximum value. This query will fail if you have more than one maximum value in one credit card type.
0 mark if the query is wrong or if the query is correct but you create a VIEW, create a new table or did not follow the instructions (e.g. use aggregate query in Question 2b).

Some Answers

Question 1 [9 marks]

(a) Find the first and last names of the customers in Singapore who own a visa credit card.

```
SELECT c.first_name, c.last_name
FROM customers c, credit_cards cc
WHERE c.ssn = cc.ssn AND cc.type = 'visa' AND c.country = 'Singapore';
```

(b) For each customer, find how many credit cards he or she owns. Print the customer's social security number and the number of credit cards owned.

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

Note: We were expecting answers that ignore customers who do not own a credit card, but we accepted answers who properly assigns 0 to those customers either with UNION clause or appropriate use of LEFT OUTER JOIN and COUNT clause.

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

It is incorrect to use COUNT(*) or COUNT(X) where X is one of the customers table column.

(c) For each Singaporean customer, find his or her first and last names and his or her total expenditure.

```
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;
```

(d) Find the social security numbers, first and last names of the different customers who purchased something from a merchand in a different country than the customer's country.

```
SELECT DISTINCT c.ssn, c.first_name, c.last_name
FROM customers c, merchands 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;
```

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

```
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';
```

(f) Find for each type of credit card the most expensive transactions. Print the credit card type and the amount of the transaction.

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

Question 2 [6 marks]

The next four questions are challenging, not everybody may be able to answer them correctly.

(a) Print the transaction identifier of the transactions that have the most expensive amount among all transactions using a credit card of the same type. Use aggregate queries (Hint: use the query in Question 1f).

```
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);
```

(b) Print the transaction identifier of the transactions that have the most expensive amount among all transactions using a credit card of the same type. Same question as in Question 2a but do not use aggregate queries.

Sample Answer 1:

```
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);
```

Sample Answer 2:

```
SELECT t1.identifier
FROM transactions t1, credit_cards cc1
WHERE t1.number = cc1.number AND NOT EXISTS
      (SELECT *
      FROM transactions t2,credit_cards cc2
      WHERE t2.number = cc2.number AND cc1.type=cc2.type AND t1.amount < t2.amount);
```

Note: Comparison of subquery should specify the use of ANY or ALL.

(c) Find the code and name of the different merchands who did not entertain transactions for every type of credit card. Use aggregate queries.

```
SELECT m.code, m.name
FROM merchands m, transactions t, credit_cards cc
WHERE m.code = t.code AND t.number = cc.number
GROUP BY m.code, m.name
HAVING COUNT(DISTINCT cc.type) < ALL (SELECT COUNT(DISTINCT type) FROM credit_cards);
```

Note: We give 0 mark if you hardcoded the number of credit card in the HAVING clause. We give 0.5 mark if you only consider credit card that have been used.

(d) Find the code and name of the different merchands who did not entertain transactions for every type of credit card. Same question as in Question 2c but do not use aggregate queries.

Sample Answer 1:

```
SELECT  DISTINCT m1.code, m1.name
FROM merchands m1, credit_cards cc1
WHERE NOT EXISTS (
      SELECT *
      FROM merchands 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);
```

Sample Answer 2:

```
SELECT DISTINCT m1.code,m1.name
FROM merchands as m1, credit_cards as cc1
WHERE (cc1.type, m1.code, m1.name) NOT IN (
      SELECT cc2.type, m2.code, m2.name
      FROM credit_cards as cc2, transactions as t2, merchands as m2
      WHERE t2.code = m2.code AND cc2.number = t2.number);
```

Sample Answer 3:

```
SELECT DISTINCT m.code, m.name
FROM merchands m
WHERE EXISTS (
      SELECT cc1.type
      FROM credit_cards cc1
      EXCEPT
      SELECT cc2.type
      FROM credit_cards cc2, transactions t2
      WHERE cc2.number = t2.number and t2.code = m.code
      );
```

– END OF PAPER –