BUAN 6320

Database Foundations for Business Analytics

Assignment 8

Problem 1

Create the following table in your database with the following schema:

Table: `Patients`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| patient_id   | int     |
| patient_name | varchar |
| conditions   | varchar |
+--------------+---------+
patient_id is the primary key for this table.
'conditions' contains 0 or more code separated by spaces.
This table contains information of the patients in the hospital.
```

Add the following data to your tables:

Input:

Patients table:

```
+------------+--------------+--------------+
| patient_id | patient_name | conditions   |
+------------+--------------+--------------+
| 1          | Daniel       | YFEV COUGH   |
| 2          | Alice        |              |
| 3          | Bob          | DIAB100 MYOP |
| 4          | George       | ACNE DIAB100 |
| 5          | Alain        | DIAB201      |
+------------+--------------+--------------+
```

Write an SQL query to report the patient_id, patient_name all conditions of patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix

Return the result table in any order.

The result should be:

Output:

```
+------------+--------------+--------------+
| patient_id | patient_name | conditions   |
+------------+--------------+--------------+
| 3          | Bob          | DIAB100 MYOP |
| 4          | George       | ACNE DIAB100 |
+------------+--------------+--------------+
```

Problem 2

Create the following table in your database with the following schema:

Table: `Sales`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| sale_id      | int     |
| product_name | varchar |
| sale_date    | date    |
+--------------+---------+
sale_id is the primary key for this table.
Each row of this table contains the product name and the date it was sold.
```

Add the following data to your tables:

Input:

Sales table:

```
+---------+--------------+------------+
| sale_id | product_name | sale_date  |
+---------+--------------+------------+
| 1       | LCPHONE      | 2000-01-16 |
| 2       | LCPhone      | 2000-01-17 |
| 3       | LcPhOnE      | 2000-02-18 |
| 4       | LCKeyCHAiN   | 2000-02-19 |
| 5       | LCKeyChain   | 2000-02-28 |
| 6       | Matryoshka   | 2000-03-31 |
+---------+--------------+------------+
```

Since table Sales was filled manually in the year 2000, product_name may contain leading and/or trailing white spaces, also they are case-insensitive.

Write an SQL query to report

- product_name in lowercase without leading or trailing white spaces.
- sale_date in the format ('YYYY-MM').
- total the number of times the product was sold in this month.

Return the result table ordered by product_name in ascending order. In case of a tie, order it by sale_date in ascending order.

Hint: you may need to use the following functions:

- LOWER()
- TRIM()
- DATE_FORMAT()

The results should be:

Output:

```
+--------------+-----------+-------+
| product_name | sale_date | total |
+--------------+-----------+-------+
| lckeychain   | 2000-02   | 2     |
| lcphone      | 2000-01   | 2     |
| lcphone      | 2000-02   | 1     |
| matryoshka   | 2000-03   | 1     |
+--------------+-----------+-------+
```

Problem 3

Create the following table in your database with the following schema:

Table: `Visits`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| visit_id    | int     |
| customer_id | int     |
+-------------+---------+
visit_id is the primary key for this table.
This table contains information about the customers who visited the mall.
```

Table: `Transactions`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| transaction_id | int     |
| visit_id       | int     |
| amount         | int     |
+----------------+---------+
transaction_id is the primary key for this table.
This table contains information about the transactions made during the
visit_id.
```

Add the following data to your tables:

Input:

```
Visits
+----------+-------------+
| visit_id | customer_id |
+----------+-------------+
| 1        | 23          |
| 2        | 9           |
| 4        | 30          |
| 5        | 54          |
| 6        | 96          |
| 7        | 54          |
| 8        | 54          |
+----------+-------------+
Transactions
+----------------+----------+--------+
| transaction_id | visit_id | amount |
+----------------+----------+--------+
| 2              | 5        | 310    |
| 3              | 5        | 300    |
| 9              | 5        | 200    |
| 12             | 1        | 910    |
| 13             | 2        | 970    |
+----------------+----------+--------+
```

Write an SQL query to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

Return the result table sorted in any order.

The results should be:

Output:

customer_id	count_no_trans
54	2
30	1
96	1

Explanation:
Customer with id = 23 visited the mall once and made one transaction during the visit with id = 12.
Customer with id = 9 visited the mall once and made one transaction during the visit with id = 13.
Customer with id = 30 visited the mall once and did not make any transactions.
Customer with id = 54 visited the mall three times. During 2 visits they did not make any transactions, and during one visit they made 3 transactions.
Customer with id = 96 visited the mall once and did not make any transactions.
As we can see, users with IDs 30 and 96 visited the mall one time without making any transactions. Also, user 54 visited the mall twice and did not make any transactions.

Problem 4

Create the following tables in your database with the following schema:

Table: `Users`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| account      | int     |
| name         | varchar |
+--------------+---------+
account is the primary key for this table.
Each row of this table contains the account number of each user in the bank.
```

Table: `Transactions`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| trans_id      | int     |
| account       | int     |
| amount        | int     |
| transacted_on | date    |
+---------------+---------+
trans_id is the primary key for this table.
Each row of this table contains all changes made to all accounts.
amount is positive if the user received money and negative if they
transferred money.
All accounts start with a balance of 0.
```

Add the following data to your tables:

Input:

```
Users table:
+------------+--------------+
| account    | name         |
+------------+--------------+
| 900001     | Alice        |
| 900002     | Bob          |
| 900003     | Charlie      |
+------------+--------------+
Transactions table:
+------------+------------+------------+---------------+
| trans_id   | account    | amount     | transacted_on |
+------------+------------+------------+---------------+
| 1          | 900001     | 7000       |  2020-08-01   |
| 2          | 900001     | 7000       |  2020-09-01   |
| 3          | 900001     | -3000      |  2020-09-02   |
| 4          | 900002     | 1000       |  2020-09-12   |
| 5          | 900003     | 6000       |  2020-08-07   |
| 6          | 900003     | 6000       |  2020-09-07   |
| 7          | 900003     | -4000      |  2020-09-11   |
+------------+------------+------------+---------------+
```

Write an SQL query to report the name and balance of users with a balance higher than 10000. The balance of an account is equal to the sum of the amounts of all transactions involving that account.

Return the result table in any order.

The results should be:

Output:

name	balance
Alice	11000

Explanation:
Alice's balance is (7000 + 7000 - 3000) = 11000.
Bob's balance is 1000.
Charlie's balance is (6000 + 6000 - 4000) = 8000.