BUAN 6320

Database Foundations for Business Analytics

Assignment 7

Problem 1

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

Table: `Users`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key for this table.
name is the name of the user.
```

Table: `Rides`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| user_id       | int     |
| distance      | int     |
+---------------+---------+
id is the primary key for this table.
user_id is the id of the user who traveled the distance "distance".
```

Add the following data to your tables:

Input:

```
Users table:
+------+-----------+
| id   | name      |
+------+-----------+
| 1    | Alice     |
| 2    | Bob       |
| 3    | Alex      |
| 4    | Donald    |
| 7    | Lee       |
| 13   | Jonathan  |
| 19   | Elvis     |
+------+-----------+
Rides table:
+------+----------+----------+
| id   | user_id  | distance |
+------+----------+----------+
| 1    | 1        | 120      |
| 2    | 2        | 317      |
| 3    | 3        | 222      |
| 4    | 7        | 100      |
| 5    | 13       | 312      |
| 6    | 19       | 50       |
| 7    | 7        | 120      |
| 8    | 19       | 400      |
| 9    | 7        | 230      |
+------+----------+----------+
```

Write an SQL query to report the distance traveled by each user.

Return the result table ordered by travelled_distance in descending order, if two or more users traveled the same distance, order them by their name in ascending order.

Hint: Use IFNULL(a,b) function in your select clause where a is the value that should be returned and b is the value that should be returned if a is NULL.

The result should be:

Output:

```
+----------+--------------------+
| name     | travelled_distance |
+----------+--------------------+
| Elvis    | 450                |
| Lee      | 450                |
| Bob      | 317                |
| Jonathan | 312                |
| Alex     | 222                |
| Alice    | 120                |
| Donald   | 0                  |
+----------+--------------------+
```

Explanation:

```
Elvis and Lee traveled 450 miles, Elvis is the top traveler as his name is
alphabetically smaller than Lee.
Bob, Jonathan, Alex, and Alice have only one ride and we just order them by the
total distances of the ride.
Donald did not have any rides, the distance traveled by him is 0.
```

Problem 2

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

Table: `Sessions`

```
+---------------------+---------+
| Column Name         | Type    |
+---------------------+---------+
| session_id          | int     |
| duration            | int     |
+---------------------+---------+
session_id is the primary key for this table.
duration is the time in seconds that a user has visited the application.
```

Add the following data to your tables:

Input:

Sessions table:

```
+-------------+---------------+
| session_id  | duration      |
+-------------+---------------+
| 1           | 30            |
| 2           | 199           |
| 3           | 299           |
| 4           | 580           |
| 5           | 1000          |
+-------------+---------------+
```

You want to know how long a user visits your application. You decided to create bins of "[0-5>", "[5-10>", "[10-15>", and "15 minutes or more" and count the number of sessions on it.

Write an SQL query to report the (bin, total).

Return the result table in any order.

The results should be:

Output:

```
+--------------+--------------+
| bin          | total        |
+--------------+--------------+
| [0-5>        | 3            |
| [5-10>       | 1            |
| [10-15>      | 0            |
| 15 or more   | 1            |
+--------------+--------------+
```

Explanation:

For session_id 1, 2, and 3 have a duration greater or equal than 0 minutes and less than 5 minutes.
For session_id 4 has a duration greater or equal than 5 minutes and less than 10 minutes.
There is no session with a duration greater than or equal to 10 minutes and less than 15 minutes.
For session_id 5 has a duration greater than or equal to 15 minutes.

Problem 3

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

Table `Activities`:

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| sell_date   | date    |
| product     | varchar |
+-------------+---------+
There is no primary key for this table, it may contain duplicates.
Each row of this table contains the product name and the date it was sold in
a market.
```

Add the following data to your tables:

Input:

```
Activities table:
+------------+------------+
| sell_date  | product    |
+------------+------------+
| 2020-05-30 | Headphone  |
| 2020-06-01 | Pencil     |
| 2020-06-02 | Mask       |
| 2020-05-30 | Basketball |
| 2020-06-01 | Bible      |
| 2020-06-02 | Mask       |
| 2020-05-30 | T-Shirt    |
+------------+------------+
```

Write an SQL query to find for each date the number of different products sold and their names.

The sold products names for each date should be sorted lexicographically.

Return the result table ordered by sell_date.

Hint: You can use the following function to aggregate a character column:

- GROUP_CONCAT(column)

The results should be:

Output:

```
+------------+----------+------------------------------+
| sell_date  | num_sold | products                     |
+------------+----------+------------------------------+
| 2020-05-30 | 3        | Basketball,Headphone,T-shirt |
| 2020-06-01 | 2        | Bible,Pencil                 |
| 2020-06-02 | 1        | Mask                         |
+------------+----------+------------------------------+
```

Explanation:

```
For 2020-05-30, Sold items were (Headphone, Basketball, T-shirt), we sort them
lexicographically and separate them by a comma.
For 2020-06-01, Sold items were (Pencil, Bible), we sort them lexicographically
and separate them by a comma.
For 2020-06-02, the Sold item is (Mask), we just return it.
```

Problem 4

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

Table: `Customers`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| customer_id   | int     |
| name          | varchar |
| country       | varchar |
+---------------+---------+
customer_id is the primary key for this table.
This table contains information about the customers in the company.
```

Table: `Product`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| product_id    | int     |
| description   | varchar |
| price         | int     |
+---------------+---------+
product_id is the primary key for this table.
This table contains information on the products in the company.
price is the product cost.
```

Table: `Orders`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| order_id      | int     |
| customer_id   | int     |
| product_id    | int     |
| order_date    | date    |
| quantity      | int     |
+---------------+---------+
order_id is the primary key for this table.
This table contains information on customer orders.
customer_id is the id of the customer who bought "quantity" products with id
"product_id".
Order_date is the date in format ('YYYY-MM-DD') when the order was shipped.
```

Add the following data to your tables:

Input:

Customers table:

customer_id	name	country
1	Winston	USA
2	Jonathan	Peru
3	Moustafa	Egypt

Product table:

product_id	description	price
10	LC Phone	300
20	LC T-Shirt	10
30	LC Book	45
40	LC Keychain	2

Orders table:

order_id	customer_id	product_id	order_date	quantity
1	1	10	2020-06-10	1
2	1	20	2020-07-01	1
3	1	30	2020-07-08	2
4	2	10	2020-06-15	2
5	2	40	2020-07-01	10
6	3	20	2020-06-24	2
7	3	30	2020-06-25	2
9	3	30	2020-05-08	3

Write an SQL query to report the customer_id and customer_name of customers who have spent at least $100 in each month of June and July 2020.

Return the result table in any order.

Hint: You need to use JOINS, GROUP BY, HAVING, CASE (in the HAVING clause) and MONTH() function to check for the month.

The results should be:

Output:

customer_id	name
1	Winston

Explanation:

Winston spent $300 (300 * 1) in June and $100 (10 * 1 + 45 * 2) in July 2020.
Jonathan spent $600 (300 * 2) in June and $20 (2 * 10) in July 2020.
Moustafa spent $110 (10 * 2 + 45 * 2) in June and $0 in July 2020.