**DATALEMUR SQL QUESTIONS**

1. **Highest Grossing Items**

Assume you are given the table containing information on Amazon customers and their spending on products in various categories.

Identify the top two highest-grossing products within each category in 2022. Output the category, product, and total spend.

Table: product\_spend

| **Column Name** | **Type** |
| --- | --- |
| category | string |
| product | string |
| user\_id | integer |
| spend | decimal |
| transaction\_date | timestamp |

### product\_spend Example Input:

| **category** | **product** | **user\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- | --- |
| appliance | refrigerator | 165 | 246.00 | 12/26/2021 12:00:00 |
| appliance | refrigerator | 123 | 299.99 | 03/02/2022 12:00:00 |
| appliance | washing machine | 123 | 219.80 | 03/02/2022 12:00:00 |
| electronics | vacuum | 178 | 152.00 | 04/05/2022 12:00:00 |
| electronics | wireless headset | 156 | 249.90 | 07/08/2022 12:00:00 |
| electronics | vacuum | 145 | 189.00 | 07/15/2022 12:00:00 |

### Example Output:

| **category** | **product** | **total\_spend** |
| --- | --- | --- |
| appliance | refrigerator | 299.99 |
| appliance | washing machine | 219.80 |
| electronics | vacuum | 341.00 |
| electronics | wireless headset | 249.90 |

**SOLUTION:**

WITH

tab1 AS

(

SELECT

category, product , sum(spend) as total\_spend

FROM product\_spend

WHERE

EXTRACT(YEAR FROM transaction\_date) = 2022

GROUP BY category,product

),

tab2 AS

(

SELECT \*,

RANK() OVER (PARTITION BY category ORDER BY total\_spend desc )

as ranking

FROM tab1

)

SELECT

category,product, total\_spend

FROM tab2

where ranking <= 2

ORDER BY category, ranking ;

1. **Odd and Even Measurements**

Assume you are given the table containing measurement values obtained from a Google sensor over several days. Measurements are taken several times within a given day.

Write a query to obtain the sum of the odd-numbered and even-numbered measurements on a particular day, in two different columns. Refer to the Example Output below for the output format.

Definition:

1st, 3rd, and 5th measurements taken **within a day** are considered odd-numbered measurements and the 2nd, 4th, and 6th measurements are even-numbered measurements.

| **Column Name** | **Type** |
| --- | --- |
| measurement\_id | integer |
| measurement\_value | decimal |
| measurement\_time | datetime |

### measurements Example Input:

| **measurement\_id** | **measurement\_value** | **measurement\_time** |
| --- | --- | --- |
| 131233 | 1109.51 | 07/10/2022 09:00:00 |
| 135211 | 1662.74 | 07/10/2022 11:00:00 |
| 523542 | 1246.24 | 07/10/2022 13:15:00 |
| 143562 | 1124.50 | 07/11/2022 15:00:00 |
| 346462 | 1234.14 | 07/11/2022 16:45:00 |

### Example Output:

| **measurement\_day** | **odd\_sum** | **even\_sum** |
| --- | --- | --- |
| 07/10/2022 00:00:00 | 2355.75 | 1662.74 |
| 07/11/2022 00:00:00 | 1124.50 | 1234.14 |

### Explanation

On 07/11/2022, there are only two measurements. In chronological order, the first measurement (odd-numbered) is 1124.50, and the second measurement(even-numbered) is 1234.14.

**SOLUTION:**

WITH Table1 AS

(

SELECT

measurement\_time,

CAST(measurement\_time AS DATE) AS meas\_day,

measurement\_value,

ROW\_NUMBER() OVER (

PARTITION BY CAST(measurement\_time AS DATE)

ORDER BY measurement\_time) AS meas\_num

FROM measurements

)

SELECT meas\_day,

SUM( CASE WHEN mod(meas\_num,2) != 0

THEN measurement\_value

ELSE 0

END ) as odd\_sum ,

SUM( CASE WHEN mod(meas\_num,2) = 0

THEN measurement\_value

ELSE 0

END ) as even\_sum

FROM table1

GROUP BY meas\_day;

1. **Signup Activation Rate**

New TikTok users sign up with their emails. They confirmed their signup by replying to the text confirmation to activate their accounts. Users may receive multiple text messages for account confirmation until they have confirmed their new account.

Write a query to find the activation rate of the users. Round the percentage to 2 decimal places.

Definitions:

emails table contain the information of user signup details.

texts table contains the users' activation information.

As of 29 Nov 2022, the term confirmation rate is changed to activation rate to reflect the nature of the new user activation process.

| **Column Name** | **Type** |
| --- | --- |
| email\_id | integer |
| user\_id | integer |
| signup\_date | datetime |

### emails Example Input:

| **email\_id** | **user\_id** | **signup\_date** |
| --- | --- | --- |
| 125 | 7771 | 06/14/2022 00:00:00 |
| 236 | 6950 | 07/01/2022 00:00:00 |
| 433 | 1052 | 07/09/2022 00:00:00 |

### texts Table:

| **Column Name** | **Type** |
| --- | --- |
| text\_id | integer |
| email\_id | integer |
| signup\_action | varchar |

### texts Example Input:

| **text\_id** | **email\_id** | **signup\_action** |
| --- | --- | --- |
| 6878 | 125 | Confirmed |
| 6920 | 236 | Not Confirmed |
| 6994 | 236 | Confirmed |

'Confirmed' in signup\_action means the user has activated their account and successfully completed the signup process.

### Example Output:

| **confirm\_rate** |
| --- |
| 0.67 |

### Explanation: 67% of users have successfully completed their signup and activated their accounts. The remaining 33% have not yet replied to the text to confirm their signup.

**SOLUTION 1**

WITH CTE1 AS

( SELECT COUNT(\*) as t FROM texts WHERE signup\_action='Confirmed')

,CTE2 AS

(SELECT COUNT(\*) as e FROM texts)

SELECT round (1.0\*t/e,2) from CTE2, CTE1;

1. **Card Launch Success**

Your team at JPMorgan Chase is soon launching a new credit card. You are asked to estimate how many cards you'll issue in the first month.

Before you can answer this question, you want to first get some perspective on how well new credit card launches typically do in their first month.

Write a query that outputs the name of the credit card, and how many cards were issued in its launch month. The launch month is the earliest record in the monthly\_cards\_issued table for a given card. Order the results starting from the biggest issued amount.

| **Column Name** | **Type** |
| --- | --- |
| issue\_month | integer |
| issue\_year | integer |
| card\_name | string |
| issued\_amount | integer |

### monthly\_cards\_issued Example Input:

| **issue\_month** | **issue\_year** | **card\_name** | **issued\_amount** |
| --- | --- | --- | --- |
| 1 | 2021 | Chase Sapphire Reserve | 170000 |
| 2 | 2021 | Chase Sapphire Reserve | 175000 |
| 3 | 2021 | Chase Sapphire Reserve | 180000 |
| 3 | 2021 | Chase Freedom Flex | 65000 |
| 4 | 2021 | Chase Freedom Flex | 70000 |

### Example Output:

| **card\_name** | **issued\_amount** |
| --- | --- |
| Chase Sapphire Reserve | 170000 |
| Chase Freedom Flex | 65000 |

### Explanation

Chase Sapphire Reserve card was launched on 1/2021 with an issued amount of 170,000 cards and the Chase Freedom Flex card was launched on 3/2021 with an issued amount of 65,000 cards.

**SOLUTION**

WITH tab1 AS

(

SELECT card\_name,issued\_amount,issue\_year,issue\_month,

ROW\_NUMBER()

OVER(PARTITION BY card\_name ORDER BY issue\_year,issue\_month)

AS row\_num

FROM monthly\_cards\_issued

)

Select card\_name, issued\_amount

FROM tab1

WHERE row\_num = 1

ORDER BY issued\_amount DESC;