1) User's Third Transaction [Uber SQL Interview Question]

Assume you are given the table below on Uber transactions made by users. Write a query to obtain the third transaction of every user. Output the user id, spend and transaction date.

`transactions` Table:

Column Name	Type
user_id	integer
spend	decimal
transaction_date	timestamp

`transactions` Example Input:

user_id	spend	transaction_date
111	100.50	01/08/2022 12:00:00
111	55.00	01/10/2022 12:00:00
121	36.00	01/18/2022 12:00:00
145	24.99	01/26/2022 12:00:00
111	89.60	02/05/2022 12:00:00

Example Output:

user_id	spend	transaction_date
111	89.60	02/05/2022 12:00:00

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

SELECT

```
USER_ID,
SPEND,
TRANSACTION_DATE
FROM
(
SELECT
USER_ID,
SPEND,
TRANSACTION_DATE,
ROW_NUMBER() OVER(PARTITION BY USER_ID ORDER BY TRANSACTION_DATE) AS
ROW_NUM
FROM TRANSACTIONS
) AS SIDDHARTH
WHERE ROW_NUM =3
```

2) Sending vs. Opening Snaps [Snapchat SQL Interview Question]

Assume you're given tables with information on Snapchat users, including their ages and time spent sending and opening snaps.

Write a query to obtain a breakdown of the time spent sending vs. opening snaps as a percentage of total time spent on these activities grouped by age group. Round the percentage to 2 decimal places in the output.

Notes:

- Calculate the following percentages:
    - time spent sending / (Time spent sending + Time spent opening)
    - Time spent opening / (Time spent sending + Time spent opening)
- To avoid integer division in percentages, multiply by 100.0 and not 100.

Effective April 15th, 2023, the solution has been updated and optimised.

`activities` Table

Column Name	Type
activity_id	integer
user_id	integer

activity_type	string ('send', 'open', 'chat')
time_spent	float
activity_date	datetime

`activities` Example Input

activity_id	user_id	activity_type	time_spent	activity_date
7274	123	open	4.50	06/22/2022 12:00:00
2425	123	send	3.50	06/22/2022 12:00:00
1413	456	send	5.67	06/23/2022 12:00:00
1414	789	chat	11.00	06/25/2022 12:00:00
2536	456	open	3.00	06/25/2022 12:00:00

`age_breakdown` Table

Column Name	Type
user_id	integer
age_bucket	string ('21-25', '26-30', '31-25')

`age_breakdown` Example Input

user_id	age_bucket
123	31-35
456	26-30
789	21-25

Example Output

age_bucket	send_perc	open_perc
26-30	65.40	34.60

31-35	43.75	56.25

Explanation

Using the age bucket 26-30 as example, the time spent sending snaps was 5.67 and the time spent opening snaps was 3.

To calculate the percentage of time spent sending snaps, we divide the time spent sending snaps by the total time spent on sending and opening snaps, which is 5.67 + 3 = 8.67.

So, the percentage of time spent sending snaps is 5.67 / (5.67 + 3) = 65.4%, and the percentage of time spent opening snaps is 3 / (5.67 + 3) = 34.6%.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
AGE_BREAKDOWN.AGE_BUCKET,
ROUND(100.0 * SUM(ACTIVITIES.TIME_SPENT) FILTER (WHERE
ACTIVITIES.ACTIVITY_TYPE = 'send') / SUM(ACTIVITIES.TIME_SPENT),2) AS
SEND_PERC,
ROUND(100.0 * SUM(ACTIVITIES.TIME_SPENT) FILTER (WHERE
ACTIVITIES.ACTIVITY_TYPE = 'open') / SUM(ACTIVITIES.TIME_SPENT),2) AS OPEN_PERC
FROM
ACTIVITIES
INNER JOIN
AGE_BREAKDOWN
ON
ACTIVITIES.USER_ID = AGE_BREAKDOWN.USER_ID
WHERE
ACTIVITIES.ACTIVITY_TYPE IN ('open', 'send')
GROUP BY AGE_BREAKDOWN.AGE_BUCKET
```

3) Highest-Grossing Items [Amazon SQL Interview Question]

Assume you're given a table containing data on Amazon customers and their spending on products in different category, write a query to identify the top two highest-grossing products within each category in the year 2022. The output should include the category, product, and total spend.

`product_spend` Table:

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

Explanation:

Within the "appliance" category, the top two highest-grossing products are "refrigerator" and "washing machine."

In the "electronics" category, the top two highest-grossing products are "vacuum" and "wireless headset."

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
CATEGORY,
PRODUCT,
TOTAL_SPEND
FROM
(SELECT
CATEGORY,
PRODUCT,
SUM(SPEND) AS TOTAL_SPEND,
RANK() OVER (
PARTITION BY CATEGORY
ORDER BY SUM(SPEND) DESC) AS RANKING
FROM
PRODUCT_SPEND
WHERE EXTRACT(YEAR FROM transaction_date) = 2022
GROUP BY
CATEGORY,PRODUCT) AS SIDDHARTH
WHERE RANKING<=2,
ORDER BY RANKING
```

4) Tweets' Rolling Averages [Twitter SQL Interview Question]

Given a table of tweet data over a specified time period, calculate the 3-day rolling average of tweets for each user. Output the user ID, tweet date, and rolling averages rounded to 2 decimal places.

Notes:

- A rolling average, also known as a moving average or running mean is a time-series technique that examines trends in data over a specified period of time.
- In this case, we want to determine how the tweet count for each user changes over a 3-day period.

Effective April 7th, 2023, the problem statement, solution and hints for this question have been revised.

`tweets` Table:

Column Name	Type
user_id	integer
tweet_date	timestamp
tweet_count	integer

`tweets` Example Input:

user_id	tweet_date	tweet_count
111	06/01/2022 00:00:00	2
111	06/02/2022 00:00:00	1
111	06/03/2022 00:00:00	3
111	06/04/2022 00:00:00	4
111	06/05/2022 00:00:00	5

Example Output:

user_id	tweet_date	rolling_avg_3d
111	06/01/2022 00:00:00	2.00
111	06/02/2022 00:00:00	1.50
111	06/03/2022 00:00:00	2.00
111	06/04/2022 00:00:00	2.67
111	06/05/2022 00:00:00	4.00

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
USER_ID,
TWEET_DATE,
ROUND(AVG(TWEET_COUNT) OVER (
PARTITION BY USER_ID
ORDER BY TWEET_DATE
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW),2) AS ROLLING_AVD_3D
FROM TWEETS
```

5) Top 5 Artists [Spotify SQL Interview Question]

Assume there are three Spotify tables: `artists`, `songs`, and `global_song_rank`, which contain information about the artists, songs, and music charts, respectively.

Write a query to find the top 5 artists whose songs appear most frequently in the Top 10 of the `global_song_rank` table. Display the top 5 artist names in ascending order, along with their song appearance ranking.

If two or more artists have the same number of song appearances, they should be assigned the same ranking, and the rank numbers should be continuous (i.e. 1, 2, 2, 3, 4, 5). If you've never seen a rank order like this before, do the rank window function tutorial.

`artists` Table:

Column Name	Type
artist_id	integer
artist_name	varchar
label_owner	varchar

`artists` Example Input:

artist_id	artist_name	label_owner

101	Ed Sheeran	Warner Music Group
120	Drake	Warner Music Group
125	Bad Bunny	Rimas Entertainment

`songs` Table:

Column Name	Type
song_id	integer
artist_id	integer
name	varchar

`songs` Example Input:

song_id	artist_id	name
55511	101	Perfect
45202	101	Shape of You
22222	120	One Dance
19960	120	Hotline Bling

`global_song_rank` Table:

Column Name	Type
day	integer (1-52)
song_id	integer
rank	integer (1-1,000,000)

`global_song_rank` Example Input:

day	song_id	rank
1	45202	5

3	45202	2
1	19960	3
9	19960	15

Example Output:

artist_name	artist_rank
Ed Sheeran	1
Drake	2

Explanation:

Ed Sheeran's song appeared twice in the Top 10 list of global song rank while Drake's song is only listed once. Therefore, Ed is ranked #1 and Drake is ranked #2.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT artist_name, artist_rank
FROM(
SELECT
  artists.artist_name,
  DENSE_RANK() OVER (
    ORDER BY COUNT(songs.song_id) DESC) AS artist_rank
FROM artists
INNER JOIN songs
  ON artists.artist_id = songs.artist_id
INNER JOIN global_song_rank AS ranking
  ON songs.song_id = ranking.song_id
WHERE ranking.rank <= 10
GROUP BY artists.artist_name) AS siddharth
WHERE artist_rank <= 5
```

6) Signup Activation Rate [TikTok SQL Interview Question]

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.

A senior analyst is interested to know the activation rate of specified users in the `emails` table. Write a query to find the activation rate. Round the percentage to 2 decimal places.

Definitions:

- `emails` table contain the information of user signup details.
- `texts` table contains the users' activation information.

Assumptions:

- The analyst is interested in the activation rate of specific users in the `emails` table, which may not include all users that could potentially be found in the `texts` table.
- For example, user 123 in the `emails` table may not be in the `texts` table and vice versa.

Effective April 4th 2023, we added an assumption to the question to provide additional clarity.

`emails` Table:

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.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
ROUND(COUNT(texts.email_id)::DECIMAL
    /COUNT(emails.email_id),2) AS activation_rate

FROM
EMAILS
```

```
LEFT JOIN
TEXTS
ON EMAILS.EMAIL_ID = TEXTS.EMAIL_ID
AND TEXTS.SIGNUP_ACTION = 'Confirmed'
```

7) Supercloud Customer [Microsoft SQL Interview Question]

A Microsoft Azure Supercloud customer is defined as a company that purchases at least one product from each product category.

Write a query that effectively identifies the company ID of such Supercloud customers.

As of 5 Dec 2022, data in the `customer_contracts` and `products` tables were updated.

`customer_contracts` Table:

Column Name	Type
customer_id	integer
product_id	integer
amount	integer

`customer_contracts` Example Input:

customer_id	product_id	amount
1	1	1000
1	3	2000
1	5	1500
2	2	3000
2	6	2000

`products` Table:

Column Name	Type

product_id	integer
product_category	string
product_name	string

`products` Example Input:

product_id	product_category	product_name
1	Analytics	Azure Databricks
2	Analytics	Azure Stream Analytics
4	Containers	Azure Kubernetes Service
5	Containers	Azure Service Fabric
6	Compute	Virtual Machines
7	Compute	Azure Functions

Example Output:

customer_id
1

Explanation:

Customer 1 bought from Analytics, Containers, and Compute categories of Azure, and thus is a Supercloud customer. Customer 2 isn't a Supercloud customer, since they don't buy any container services from Azure.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
DISTINCT CUSTOMER_ID AS CUSTOMER_ID
FROM(
SELECT
```

```
CC.CUSTOMER_ID,
PP.PRODUCT_CATEGORY
FROM
CUSTOMER_CONTRACTS AS CC
INNER JOIN
PRODUCTS AS PP
ON CC.PRODUCT_ID = PP.PRODUCT_ID
GROUP BY CC.CUSTOMER_ID, PP.PRODUCT_CATEGORY) AS SID
GROUP BY CUSTOMER_ID
HAVING COUNT(CUSTOMER_ID) >= 3
```

8) Odd and Even Measurements [Google SQL Interview Question]

This is the same question as problem #28 in the SQL Chapter of Ace the Data Science Interview!

Assume you're given a table with measurement values obtained from a Google sensor over multiple days with measurements taken multiple times within each day.

Write a query to calculate the sum of odd-numbered and even-numbered measurements separately for a particular day and display the results in two different columns. Refer to the Example Output below for the desired format.

Definition:

- Within a day, measurements taken at 1st, 3rd, and 5th times are considered odd-numbered measurements, and measurements taken at 2nd, 4th, and 6th times are considered even-numbered measurements.

Effective April 15th, 2023, the question and solution for this question have been revised.

`measurements` Table:

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

Based on the results,

- On 07/10/2022, the sum of the odd-numbered measurements is 2355.75, while the sum of the even-numbered measurements is 1662.74.
- On 07/11/2022, there are only two measurements available. The sum of the odd-numbered measurements is 1124.50, and the sum of the even-numbered measurements is 1234.14.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
MEASUREMENT_DAY,
SUM(MEASUREMENT_VALUE) FILTER (WHERE MEASUREMENT_NUM %2 != 0) AS
ODD_SUM,
SUM(MEASUREMENT_VALUE) FILTER (WHERE MEASUREMENT_NUM % 2 = 0) AS
EVEN_SUM
FROM
(
SELECT
CAST(MEASUREMENT_TIME AS DATE) AS MEASUREMENT_DAY,
```

```
MEASUREMENT_VALUE,
ROW_NUMBER() OVER(
PARTITION BY CAST(MEASUREMENT_TIME AS DATE)
ORDER BY MEASUREMENT_TIME) AS MEASUREMENT_NUM
FROM
MEASUREMENTS) AS SIDDHARTH
GROUP BY MEASUREMENT_DAY
```

9) Compressed Mode [Alibaba SQL Interview Question]

You're given a table containing the item count for each order on Alibaba, along with the frequency of orders that have the same item count. Write a query to retrieve the mode of the order occurrences. Additionally, if there are multiple item counts with the same mode, the results should be sorted in ascending order.

Clarifications:

- `item_count`: Represents the number of items sold in each order.
- `order_occurrences`: Represents the frequency of orders with the corresponding number of items sold per order.
- For example, if there are 800 orders with 3 items sold in each order, the record would have an `item_count` of 3 and an `order_occurrences` of 800.

Effective June 14th, 2023, the problem statement has been revised and additional clarification have been added for clarity.

`items_per_order` Table:

Column Name	Type
item_count	integer
order_occurrences	integer

`items_per_order` Example Input:

item_count	order_occurrences
1	500
2	1000

3	800

Example Output:

mode
2

Explanation:

Based on the example output, the `order_occurrences` value of 1000 corresponds to the highest frequency among all item counts. This means that item count of 2 has occurred 1000 times, making it the mode of order occurrences.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
ITEM_COUNT AS MODE
FROM
ITEMS_PER_ORDER
WHERE
ORDER_OCCURRENCES = (
SELECT
MAX(ORDER_OCCURRENCES)
FROM ITEMS_PER_ORDER)
ORDER BY
ITEM_COUNT
```

10) Card Launch Success [JPMorgan Chase SQL Interview Question]

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.

`monthly_cards_issued` Table:

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.

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
CARD_NAME,
ISSUED_AMOUNT
FROM(
SELECT
  card_name,
  issued_amount,
  MAKE_DATE(issue_year, issue_month, 1) AS issue_date,
  MIN(MAKE_DATE(issue_year, issue_month, 1)) OVER (
    PARTITION BY card_name) AS launch_date
FROM monthly_cards_issued) AS SIDDHARTH
WHERE ISSUE_DATE = LAUNCH_DATE
ORDER BY ISSUED_AMOUNT DESC
```

11) International Call Percentage [Verizon SQL Interview Question]

A phone call is considered an international call when the person calling is in a different country than the person receiving the call.

What percentage of phone calls are international? Round the result to 1 decimal.

Assumption:

- The `caller_id` in `phone_info` table refers to both the caller and receiver.

`phone_calls` Table:

Column Name	Type
caller_id	integer
receiver_id	integer
call_time	timestamp

`phone_calls` Example Input:

caller_id	receiver_id	call_time

1	2	2022-07-04 10:13:49
1	5	2022-08-21 23:54:56
5	1	2022-05-13 17:24:06
5	6	2022-03-18 12:11:49

`phone_info` Table:

Column Name	Type
caller_id	integer
country_id	integer
network	integer
phone_number	string

`phone_info` Example Input:

caller_id	country_id	network	phone_number
1	US	Verizon	+1-212-897-1964
2	US	Verizon	+1-703-346-9529
3	US	Verizon	+1-650-828-4774
4	US	Verizon	+1-415-224-6663
5	IN	Vodafone	+91 7503-907302
6	IN	Vodafone	+91 2287-664895

Example Output:

international_calls_pct
50.0

Explanation

There is a total of 4 calls with 2 of them being international calls (from caller_id 1 => receiver_id 5, and caller_id 5 => receiver_id 1). Thus, 2/4 = 50.0%

The dataset you are querying against may have different input & output - **this is just an example**!

ANSWER

```
SELECT
  ROUND(
    100.0 * SUM(CASE
      WHEN caller.country_id <> receiver.country_id THEN 1 ELSE NULL END)
    /COUNT(*) ,1) AS international_call_pct
FROM phone_calls AS calls
LEFT JOIN phone_info AS caller
  ON calls.caller_id = caller.caller_id
LEFT JOIN phone_info AS receiver
  ON calls.receiver_id = receiver.caller_id;
```