Q1. Laptop vs. Mobile Viewership

Assume you're given the table on user viewership categorised by device type where the three types are laptop, tablet, and phone.

Write a query that calculates the total viewership for laptops and mobile devices where mobile is defined as the sum of tablet and phone viewership. Output the total viewership for laptops as **laptop\_reviews** and the total viewership for mobile devices as **mobile\_views**.

*Effective 15 April 2023, the solution has been updated with a more concise and easy-to-understand approach.*

**viewership Table**

| **Column Name** | **Type** |
| --- | --- |
| user\_id | integer |
| device\_type | string ('laptop', 'tablet', 'phone') |
| view\_time | timestamp |

**viewership Example Input**

| **user\_id** | **device\_type** | **view\_time** |
| --- | --- | --- |
| 123 | tablet | 01/02/2022 00:00:00 |
| 125 | laptop | 01/07/2022 00:00:00 |
| 128 | laptop | 02/09/2022 00:00:00 |
| 129 | phone | 02/09/2022 00:00:00 |
| 145 | tablet | 02/24/2022 00:00:00 |

**Example Output**

| **laptop\_views** | **mobile\_views** |
| --- | --- |
| 2 | 3 |

**Explanation**

Based on the example input, there are a total of 2 laptop views and 3 mobile views.

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

Q2. Average Review Ratings

Given the reviews table, write a query to retrieve the average star rating for each product, grouped by month. The output should display the month as a numerical value, product ID, and average star rating rounded to two decimal places. Sort the output first by month and then by product ID.

P.S. If you've read the Ace the Data Science Interview, and liked it, consider writing us a review?

**reviews Table:**

| **Column Name** | **Type** |
| --- | --- |
| review\_id | integer |
| user\_id | integer |
| submit\_date | datetime |
| product\_id | integer |
| stars | integer (1-5) |

**reviews Example Input:**

| **review\_id** | **user\_id** | **submit\_date** | **product\_id** | **stars** |
| --- | --- | --- | --- | --- |
| 6171 | 123 | 06/08/2022 00:00:00 | 50001 | 4 |
| 7802 | 265 | 06/10/2022 00:00:00 | 69852 | 4 |
| 5293 | 362 | 06/18/2022 00:00:00 | 50001 | 3 |
| 6352 | 192 | 07/26/2022 00:00:00 | 69852 | 3 |
| 4517 | 981 | 07/05/2022 00:00:00 | 69852 | 2 |

**Example Output:**

| **mth** | **product** | **avg\_stars** |
| --- | --- | --- |
| 6 | 50001 | 3.50 |
| 6 | 69852 | 4.00 |
| 7 | 69852 | 2.50 |

**Explanation**

Product 50001 received two ratings of 4 and 3 in the month of June (6th month), resulting in an average star rating of 3.5.

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

Q3. Matching Skills

Given a table of candidates and their skills, you're tasked with finding the candidates best suited for an open Data Science job. You want to find candidates who are proficient in Python, Tableau, and PostgreSQL.

Write a query to list the candidates who possess all of the required skills for the job. Sort the output by candidate ID in ascending order.

**Assumption:**

* There are no duplicates in the **candidates** table.

**candidates Table:**

| **Column Name** | **Type** |
| --- | --- |
| candidate\_id | integer |
| skill | varchar |

**candidates Example Input:**

| **candidate\_id** | **skill** |
| --- | --- |
| 123 | Python |
| 123 | Tableau |
| 123 | PostgreSQL |
| 234 | R |
| 234 | PowerBI |
| 234 | SQL Server |
| 345 | Python |
| 345 | Tableau |

**Example Output:**

| **candidate\_id** |
| --- |
| 123 |

**Explanation**

Candidate 123 is displayed because they have Python, Tableau, and PostgreSQL skills. 345 isn't included in the output because they're missing one of the required skills: PostgreSQL.

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

Q4. Odd and Even Measurements

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**!

Q5. Y-on-Y Growth Rate

Assume you're given a table containing information about Wayfair user transactions for different products. Write a query to calculate the year-on-year growth rate for the total spend of each product, grouping the results by product ID.

The output should include the year in ascending order, product ID, current year's spend, previous year's spend and year-on-year growth percentage, rounded to 2 decimal places.

**user\_transactions Table:**

| **Column Name** | **Type** |
| --- | --- |
| transaction\_id | integer |
| product\_id | integer |
| spend | decimal |
| transaction\_date | datetime |

**user\_transactions Example Input:**

| **transaction\_id** | **product\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- |
| 1341 | 123424 | 1500.60 | 12/31/2019 12:00:00 |
| 1423 | 123424 | 1000.20 | 12/31/2020 12:00:00 |
| 1623 | 123424 | 1246.44 | 12/31/2021 12:00:00 |
| 1322 | 123424 | 2145.32 | 12/31/2022 12:00:00 |

**Example Output:**

| **year** | **product\_id** | **curr\_year\_spend** | **prev\_year\_spend** | **yoy\_rate** |
| --- | --- | --- | --- | --- |
| 2019 | 123424 | 1500.60 | NULL | NULL |
| 2020 | 123424 | 1000.20 | 1500.60 | -33.35 |
| 2021 | 123424 | 1246.44 | 1000.20 | 24.62 |
| 2022 | 123424 | 2145.32 | 1246.44 | 72.12 |

**Explanation:**

Product ID 123424 is analyzed for multiple years: 2019, 2020, 2021, and 2022.

* In the year 2020, the current year's spend is 1000.20, and there is no previous year's spend recorded (indicated by an empty cell).
* In the year 2021, the current year's spend is 1246.44, and the previous year's spend is 1000.20.
* In the year 2022, the current year's spend is 2145.32, and the previous year's spend is 1246.44.

To calculate the year-on-year growth rate, we compare the current year's spend with the previous year's spend.For instance, the spend grew by 24.62% from 2020 to 2021, indicating a positive growth rate.

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

Q6. Page With No Likes

Assume you're given two tables containing data about Facebook Pages and their respective likes (as in "Like a Facebook Page").

Write a query to return the IDs of the Facebook pages that have zero likes. The output should be sorted in ascending order based on the page IDs.

**pages Table:**

| **Column Name** | **Type** |
| --- | --- |
| page\_id | integer |
| page\_name | varchar |

**pages Example Input:**

| **page\_id** | **page\_name** |
| --- | --- |
| 20001 | SQL Solutions |
| 20045 | Brain Exercises |
| 20701 | Tips for Data Analysts |

**page\_likes Table:**

| **Column Name** | **Type** |
| --- | --- |
| user\_id | integer |
| page\_id | integer |
| liked\_date | datetime |

**page\_likes Example Input:**

| **user\_id** | **page\_id** | **liked\_date** |
| --- | --- | --- |
| 111 | 20001 | 04/08/2022 00:00:00 |
| 121 | 20045 | 03/12/2022 00:00:00 |
| 156 | 20001 | 07/25/2022 00:00:00 |

**Example Output:**

| **page\_id** |
| --- |
| 20701 |

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

Q7. Supercloud Customer

A Microsoft Azure Supercloud customer is defined as a customer who has purchased at least one product from every product category listed in the **products** table.

Write a query that identifies the customer IDs of these Supercloud customers.

**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**!

Q8. Sending vs. Opening Snap

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**!

Q9. User's Third Transaction

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**!

Q10. Repeated Payments

Sometimes, payment transactions are repeated by accident; it could be due to user error, API failure or a retry error that causes a credit card to be charged twice.

Using the transactions table, identify any payments made at the same merchant with the same credit card for the same amount within 10 minutes of each other. Count such repeated payments.

**Assumptions:**

* The first transaction of such payments should not be counted as a repeated payment. This means, if there are two transactions performed by a merchant with the same credit card and for the same amount within 10 minutes, there will only be 1 repeated payment.

**transactions Table:**

| **Column Name** | **Type** |
| --- | --- |
| transaction\_id | integer |
| merchant\_id | integer |
| credit\_card\_id | integer |
| amount | integer |
| transaction\_timestamp | datetime |

**transactions Example Input:**

| **transaction\_id** | **merchant\_id** | **credit\_card\_id** | **amount** | **transaction\_timestamp** |
| --- | --- | --- | --- | --- |
| 1 | 101 | 1 | 100 | 09/25/2022 12:00:00 |
| 2 | 101 | 1 | 100 | 09/25/2022 12:08:00 |
| 3 | 101 | 1 | 100 | 09/25/2022 12:28:00 |
| 4 | 102 | 2 | 300 | 09/25/2022 12:00:00 |
| 6 | 102 | 2 | 400 | 09/25/2022 14:00:00 |

**Example Output:**

| **payment\_count** |
| --- |
| 1 |

**Explanation**

Within 10 minutes after Transaction 1, Transaction 2 is conducted at Merchant 1 using the same credit card for the same amount. This is the only instance of repeated payment in the given sample data.

Since Transaction 3 is completed after Transactions 2 and 1, each of which occurs after 20 and 28 minutes, respectively hence it does not meet the repeated payments' conditions. Whereas, Transactions 4 and 6 have different amounts.

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

Q11. Compressed Mode

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**!

Q12. 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.

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**!

Q13. Highest-Grossing Items

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**!

Q14. International Call Percentage

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**!

Q15. Tweets' Rolling Average

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**!

Q16. Top Three Salaries

As part of an ongoing analysis of salary distribution within the company, your manager has requested a report identifying high earners in each department. A 'high earner' within a department is defined as an employee with a salary ranking among the top three salaries within that department.

You're tasked with identifying these high earners across all departments. Write a query to display the employee's name along with their department name and salary. In case of duplicates, sort the results of department name in ascending order, then by salary in descending order. If multiple employees have the same salary, then order them alphabetically.

Note: Ensure to utilize the appropriate ranking window function to handle duplicate salaries effectively.

*As of June 18th, we have removed the requirement for unique salaries and revised the sorting order for the results.*

**employee Schema:**

| **column\_name** | **type** | **description** |
| --- | --- | --- |
| employee\_id | integer | The unique ID of the employee. |
| name | string | The name of the employee. |
| salary | integer | The salary of the employee. |
| department\_id | integer | The department ID of the employee. |
| manager\_id | integer | The manager ID of the employee. |

**employee Example Input:**

| **employee\_id** | **name** | **salary** | **department\_id** | **manager\_id** |
| --- | --- | --- | --- | --- |
| 1 | Emma Thompson | 3800 | 1 | 6 |
| 2 | Daniel Rodriguez | 2230 | 1 | 7 |
| 3 | Olivia Smith | 2000 | 1 | 8 |
| 4 | Noah Johnson | 6800 | 2 | 9 |
| 5 | Sophia Martinez | 1750 | 1 | 11 |
| 6 | Liam Brown | 13000 | 3 |  |
| 7 | Ava Garcia | 12500 | 3 |  |
| 8 | William Davis | 6800 | 2 |  |
| 9 | Isabella Wilson | 11000 | 3 |  |
| 10 | James Anderson | 4000 | 1 | 11 |

**department Schema:**

| **column\_name** | **type** | **description** |
| --- | --- | --- |
| department\_id | integer | The department ID of the employee. |
| department\_name | string | The name of the department. |

**department Example Input:**

| **department\_id** | **department\_name** |
| --- | --- |
| 1 | Data Analytics |
| 2 | Data Science |

**Example Output:**

| **department\_name** | **name** | **salary** |
| --- | --- | --- |
| Data Analytics | James Anderson | 4000 |
| Data Analytics | Emma Thompson | 3800 |
| Data Analytics | Daniel Rodriguez | 2230 |
| Data Science | Noah Johnson | 6800 |
| Data Science | William Davis | 6800 |

The output displays the high earners in each department.

* In the Data Analytics deaprtment, James Anderson leads with a salary of $4,000, followed by Emma Thompson earning $3,800, and Daniel Rodriguez with $2,230.
* In the Data Science department, both Noah Johnson and William Davis earn $6,800, with Noah listed before William due to alphabetical ordering.

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

Q17. Histogram of Users and Purchases

Assume you're given a table on Walmart user transactions. Based on their most recent transaction date, write a query that retrieve the users along with the number of products they bought.

Output the user's most recent transaction date, user ID, and the number of products, sorted in chronological order by the transaction date.

**user\_transactions Table:**

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

**user\_transactions Example Input:**

| **product\_id** | **user\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- |
| 3673 | 123 | 68.90 | 07/08/2022 12:00:00 |
| 9623 | 123 | 274.10 | 07/08/2022 12:00:00 |
| 1467 | 115 | 19.90 | 07/08/2022 12:00:00 |
| 2513 | 159 | 25.00 | 07/08/2022 12:00:00 |
| 1452 | 159 | 74.50 | 07/10/2022 12:00:00 |

**Example Output:**

| **transaction\_date** | **user\_id** | **purchase\_count** |
| --- | --- | --- |
| 07/08/2022 12:00:00 | 115 | 1 |
| 07/08/2022 12:00:000 | 123 | 2 |
| 07/10/2022 12:00:00 | 159 | 1 |

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