1. **3-Topping Pizzas [McKinsey SQL Interview Question]**

You’re a consultant for a major pizza chain that will be running a promotion where all 3-topping pizzas will be sold for a fixed price and are trying to understand the costs involved.

Given a list of pizza toppings, consider all the possible 3-topping pizzas, and print out the total cost of those 3 toppings. Sort the results with the highest total cost on the top followed by pizza toppings in ascending order.

Break ties by listing the ingredients in alphabetical order, starting from the first ingredient, followed by the second and third.

\*\* pizza\_toppings :\*\*

| **Column Name** | **Type** |
| --- | --- |
| topping\_name | varchar(255) |
| ingredient\_cost | decimal(10,2) |

**Input Data:**

| **topping\_name** | **ingredient\_cost** |
| --- | --- |
| Pepperoni | 0.50 |
| Sausage | 0.70 |
| Chicken | 0.55 |
| Extra Cheese | 0.40 |

**Example Output:**

| **Pizza** | **total\_cost** |
| --- | --- |
| Chicken,Pepperoni,Sausage | 1.75 |
| Chicken,Extra Cheese,Sausage | 1.65 |
| Extra Cheese,Pepperoni,Sausage | 1.60 |

***Solution:***

SELECT CONCAT(t1.topping\_name, ',' , t2.topping\_name, ',' , t3.topping\_name ) AS pizza,

(t1.ingredient\_cost + t2.ingredient\_cost + t3.ingredient\_cost) AS pizza\_cost

FROM pizza\_toppings t1

JOIN pizza\_toppings t2

ON t1.topping\_name < t2.topping\_name

JOIN pizza\_toppings t3

ON t2.topping\_name < t3.topping\_name

ORDER BY pizza\_cost DESC, pizza ASC

1. **Repeated Payments [Stripe SQL Interview Question]**

Identify any payments made at the same merchant with the same credit card for the same amount within 10 minutes of each other and report the count of such repeated payments

Example Input: **transactions**

| **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 |

**Output:**

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

***Solution:***

SELECT COUNT(\*) AS payment\_count

FROM transactions t1

JOIN transactions t2

ON t1.merchant\_id = t2.merchant\_id

AND t1.credit\_card\_id = t2.credit\_card\_id

AND t1.amount = t2.amount

AND t1.transaction\_id < t2.transaction\_id -- Avoid self-matching the same transaction

AND ABS(TIMESTAMPDIFF(MINUTE, t1.transaction\_timestamp, t2.transaction\_timestamp)) <= 10;

1. **Median Google Search Frequency [Google SQL Interview Question]**

Google’s Marketing Team needed to add a simple statistic to their upcoming Superbowl Ad: the median number of searches made per year. You were given a summary table that tells you the number of searches made last year, write a query to report the median searches made per user

search\_frequency Table:

| **Column Name** | **Type** |
| --- | --- |
| Searches | integer |
| num\_users | integer |

**Example Input:**

| **searches** | **num\_users** |
| --- | --- |
| 1 | 2 |
| 2 | 2 |
| 3 | 3 |
| 4 | 1 |

**Example Output:**

| **median** |
| --- |
| 2.5 |

***Solution:***

SELECT PERCENTILE\_CONT(0.5) WITHIN GROUP (ORDER BY searches) AS median

FROM (

SELECT searches

FROM search\_frequency

CROSS JOIN generate\_series(1, num\_users) AS user\_id

) AS Expanded;

1. **Monthly Merchant Balance [Visa SQL Interview Question]**

Say you have access to all the transactions for a given merchant account. Write a query to print the cumulative balance of the merchant account at the end of each day, with the total balance reset back to zero at the end of the month. Output the transaction date and cumulative balance.

**transactions Table:**

| **Column Name** | **Type** |
| --- | --- |
| transaction\_id | Integer |
| type | string ('deposit', 'withdrawal') |
| amount | Decimal |
| transaction\_date | Timestamp |

**Example Input:**

| **transaction\_id** | **Type** | **amount** | **transaction\_date** |
| --- | --- | --- | --- |
| 19153 | deposit | 65.90 | 07/10/2022 10:00:00 |
| 53151 | deposit | 178.55 | 07/08/2022 10:00:00 |
| 29776 | withdrawal | 25.90 | 07/08/2022 10:00:00 |
| 16461 | withdrawal | 45.99 | 07/08/2022 10:00:00 |
| 77134 | deposit | 32.60 | 07/10/2022 10:00:00 |

**Example Output:**

| **transaction\_date** | **Balance** |
| --- | --- |
| 07/08/2022 12:00:00 | 106.66 |
| 07/10/2022 12:00:00 | 205.16 |

***Solution:***

WITH DailyBalance AS (

SELECT

DATE(transaction\_date) AS transaction\_day,

SUM(CASE

WHEN type = 'deposit' THEN amount

WHEN type = 'withdrawal' THEN -amount

END) AS daily\_balance

FROM transactions

GROUP BY DATE(transaction\_date)

)

SELECT transaction\_day,

SUM(daily\_balance) OVER (PARTITION BY EXTRACT(YEAR FROM transaction\_day), EXTRACT(MONTH FROM transaction\_day) ORDER BY transaction\_day) AS balance

FROM DailyBalance

ORDER BY transaction\_day;

1. **Server Utilization Time [Amazon SQL Interview Question]**

Fleets of server power Amazon Web Services (AWS). Senior management has requested data-driven solutions to optimize server usage.

Write a query that calculates the total time that the fleet of servers was running. The output should be in units of **full days**.

Assumptions:

* Each server might start and stop several times
* The total time in which the server fleet is running can be calculated as the sum of each server's uptime

**server\_utilization Table:**

| **Column Name** | **Type** |
| --- | --- |
| server\_id | integer |
| status\_time | timestamp |
| session\_status | string |

**Example Input:**

| **server\_id** | **status\_time** | **session\_status** |
| --- | --- | --- |
| 1 | 08/02/2022 10:00:00 | start |
| 1 | 08/04/2022 10:00:00 | stop |
| 2 | 08/17/2022 10:00:00 | start |
| 2 | 08/24/2022 10:00:00 | stop |

**Example Output:**

| **total\_uptime\_days** |
| --- |
| 21 |

***Solution:***

SELECT

SUM(DATEDIFF(CAST(b.status\_time AS DATE), CAST(a.status\_time AS DATE))) AS total\_uptime\_days

FROM

server\_utilization a

JOIN

server\_utilization b

ON a.server\_id = b.server\_id

WHERE

a.session\_status = 'start'

AND b.session\_status = 'stop'

AND a.status\_time < b.status\_time

ORDER BY

a.server\_id, a.status\_time;

1. **Uniquely Staffed Consultants [Accenture SQL Interview Questions]**

As a Data Analyst on the People Operations team at Accenture, you are tasked with understanding how many consultants are staffed to each client, and how many consultants are exclusively staffed to a single client.

Write a query that displays the outputs of client name and the number of uniquely and exclusively staffed consultants ordered by client name.

**employees Table:**

| **Column Name** | **Type** |
| --- | --- |
| employee\_id | integer |
| engagement\_id | integer |

**Example Input:**

| **employee\_id** | **engagement\_id** |
| --- | --- |
| 1001 | 1 |
| 1001 | 2 |
| 1002 | 1 |
| 1003 | 3 |
| 1004 | 4 |

**consulting\_engagements Table:**

| **Column Name** | **Type** |
| --- | --- |
| engagement\_id | Integer |
| project\_name | String |
| client\_name | String |

**consulting\_engagements Example Input:**

| **engagement\_id** | **project\_name** | **client\_name** |
| --- | --- | --- |
| 1 | SAP Logistics Modernization | Department of Defense |
| 2 | Oracle Cloud Migration | Department of Education |
| 3 | Trust & Safety Operations | Google |
| 4 | SAP IoT Cloud Integration | Google |

**Example Output:**

| **client\_name** | **total\_staffed** | **exclusive\_staffed** |
| --- | --- | --- |
| Department of Defense | 2 | 1 |
| Department of Education | 1 | 0 |
| Google | 2 | 2 |

***Solution:***

SELECT

ce.client\_name,

COUNT(DISTINCT e.employee\_id) AS total\_staffed,

SUM(CASE WHEN COUNT(DISTINCT ce.client\_name) = 1 THEN 1 ELSE 0 END) AS exclusive\_staffed

FROM employees e

JOIN consulting\_engagements ce ON e.engagement\_id = ce.engagement\_id

GROUP BY ce.client\_name

ORDER BY ce.client\_name;

1. **Find the month-over-month percentage change for monthly active users (MAU)**

**Table columns:** user\_id & date

***Solution***

WITH mau AS (

SELECT DATE\_TRUNC(month, date) AS month\_date,

COUNT(DISTINCT user\_id) AS mthly\_active\_users

FROM log\_table

GROUP BY DATE\_TRUNC(month, date) )

SELECT

a.month\_date AS previous\_month,

a.mthly\_active\_users AS previous\_mau,

b.month\_date AS current\_month,

b.mthly\_active\_users AS current\_mau,

ROUND(100\*(b.mthly\_active\_users - a.mthly\_active\_users)/ a.mthly\_active\_users , 2) AS MoM,

FROM mau a

JOIN mau b

ON a.month\_date = b.month\_date – INTERVAL ‘1 month’