Google SQL

Interview questions for Data Analysts

Part I

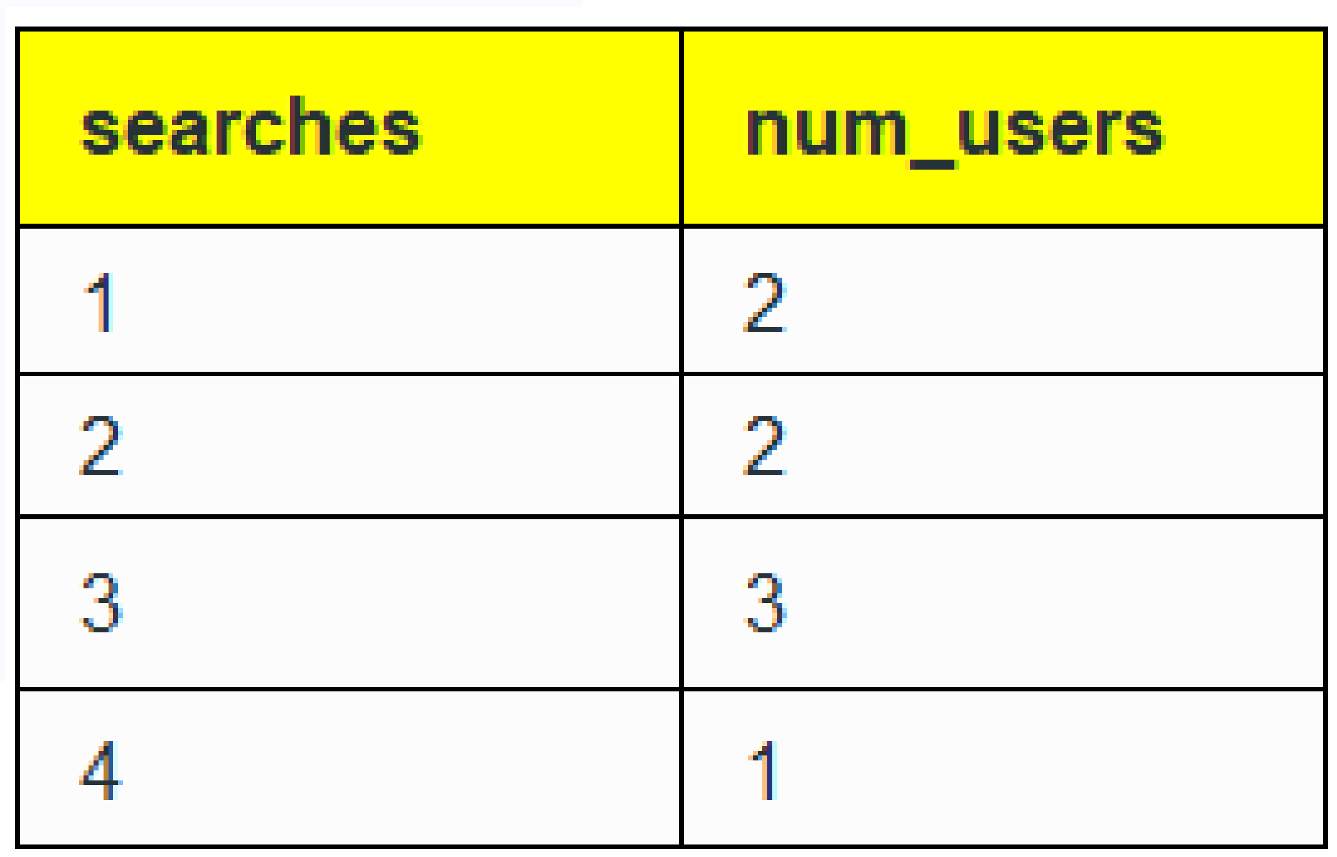
linkedin.com/in/ileonjose

1. Median Number of Searches

Problem Statement: For an IPL ad campaign, you need to determine the median number of searches made by fans last year. The data is stored in a summary table with columns *searches* (indicating the number of searches) and *num_users* (indicating the number of users who performed that many searches). Due to the large size of the dataset, a direct calculation of the median from the summary table is not feasible.

linkedin.com/in/ileonjose

search_frequency



searches	num_users
1	2
2	2
3	3
4	1

linkedin.com/in/ileonjose

How to Solve:

1. Expand Data:

- Create a detailed list where each search count is repeated according to the number of users. For example, if 10 users made 5 searches each, the list should include 10 entries of 5 searches.

1. Calculate Median:

- Use the expanded list to find the median value. The median is the middle value when all entries are ordered. If the number of entries is even, the median is the average of the two middle values.

linkedin.com/in/ileonjose

SQL

```sql
WITH expanded_searches AS (
  SELECT searches
  FROM search_frequency
  CROSS JOIN GENERATE_SERIES(1, num_users) AS g
)
SELECT
  ROUND(PERCENTILE_CONT(0.50) WITHIN
  GROUP (ORDER BY searches)::DECIMAL, 1) AS median
FROM expanded_searches;
```

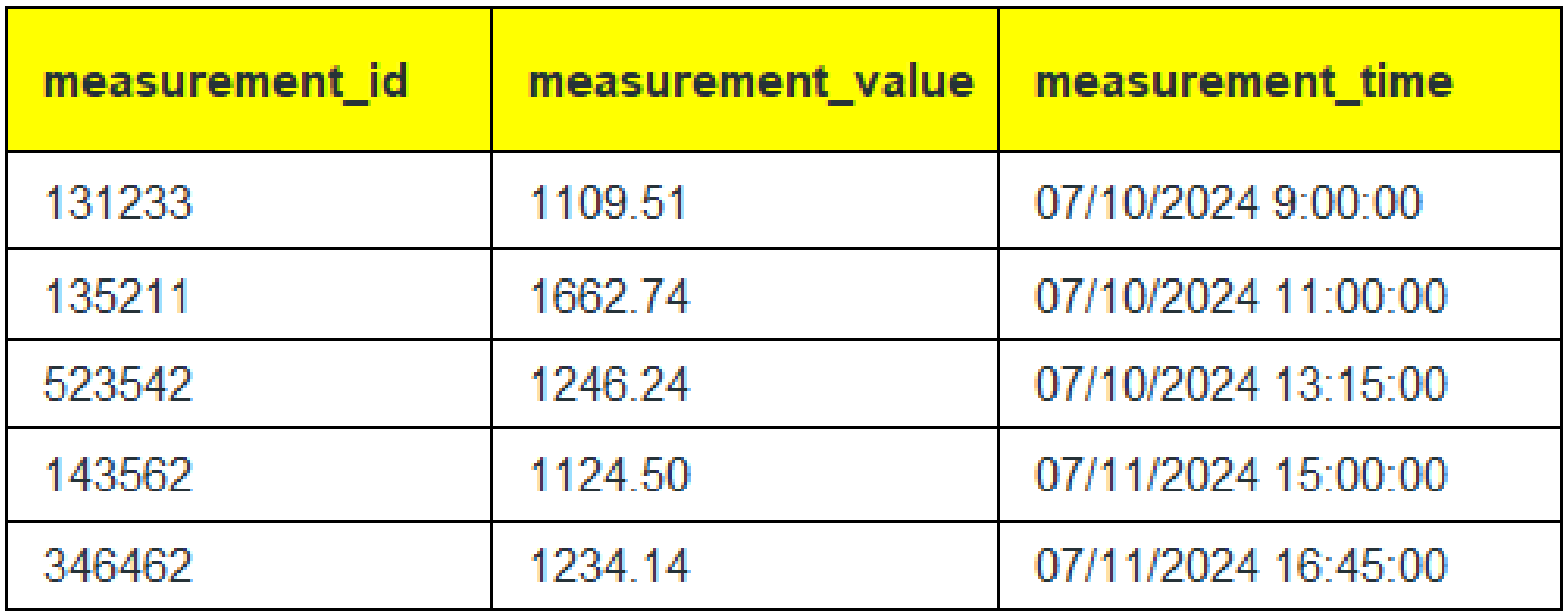
linkedin.com/in/ileonjose

2. Sum of Odd and Even Measurements

Problem Statement: You need to calculate the sum of measurements taken at various cricket matches, where measurements are categorized by odd and even row numbers. You have a table *measurements* with columns *measurement_time* and *measurement_value*.

linkedin.com/in/ileonjose

Measurements



measurement_id	measurement_value	measurement_time
131233	1109.51	07/10/2024 9:00:00
135211	1662.74	07/10/2024 11:00:00
523542	1246.24	07/10/2024 13:15:00
143562	1124.50	07/11/2024 15:00:00
346462	1234.14	07/11/2024 16:45:00

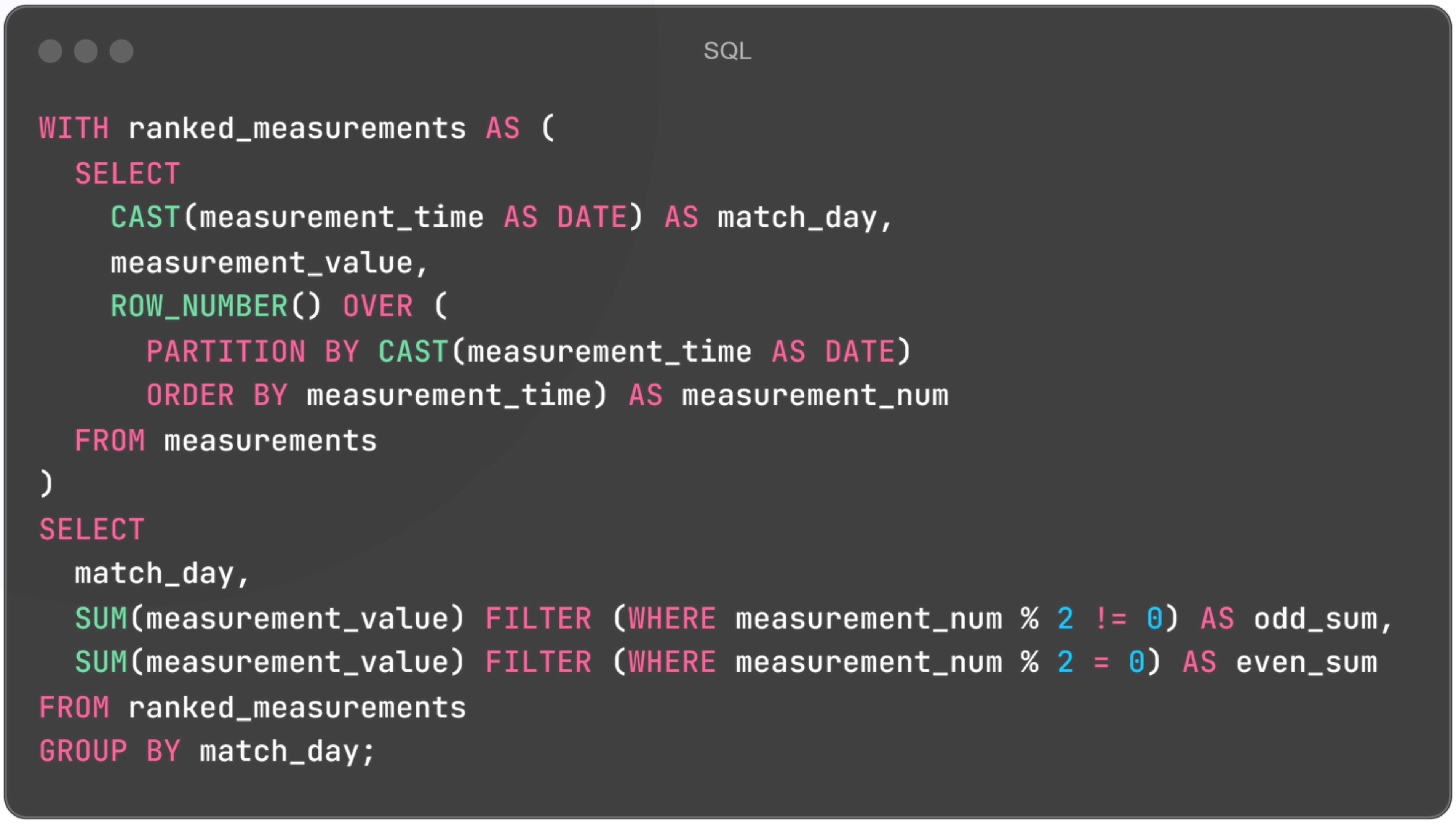
linkedin.com/in/ileonjose

How to Solve:

1. Assign Row Numbers:
   - Use the ROW_NUMBER() function to assign a unique row number to each measurement, partitioned by match date and ordered by measurement time.

2. Calculate Sums:
   - Use conditional aggregation to sum measurements based on whether their row number is odd or even.

linkedin.com/in/ileonjose



```sql
WITH ranked_measurements AS (
  SELECT
    CAST(measurement_time AS DATE) AS match_day,
    measurement_value,
    ROW_NUMBER() OVER (
      PARTITION BY CAST(measurement_time AS DATE)
      ORDER BY measurement_time) AS measurement_num
  FROM measurements
)
SELECT
  match_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 ranked_measurements
GROUP BY match_day;
```

linkedin.com/in/ileonjose

3. Google Maps - Most Off-Topic UGC

Problem Statement: As a Data Analyst on the Google Maps User Generated Content team, you and your Product Manager are investigating user-generated content (UGC) – photos and reviews that independent users upload to Google Maps.

Identify which venue type (e.g., Restaurant, Bar) has the highest amount of "off-topic" user-generated content (UGC). You have two tables: *place_info* (with place categories) and maps*_ugc_review* (with UGC details).

linkedin.com/in/ileonjose

place_info

place_id	place_name	place_category
1	Baar Baar	Restaurant
2	Rubirosa	Restaurant
3	Mr. Purple	Bar
4	La Caverna	Bar

maps_ugc_review

content_id	place_id	content_tag
101	1	Off-topic
110	2	Misinformation
153	2	Off-topic
176	3	Harassment
190	3	Off-topic

linkedin.com/in/ileonjose

How to Solve:

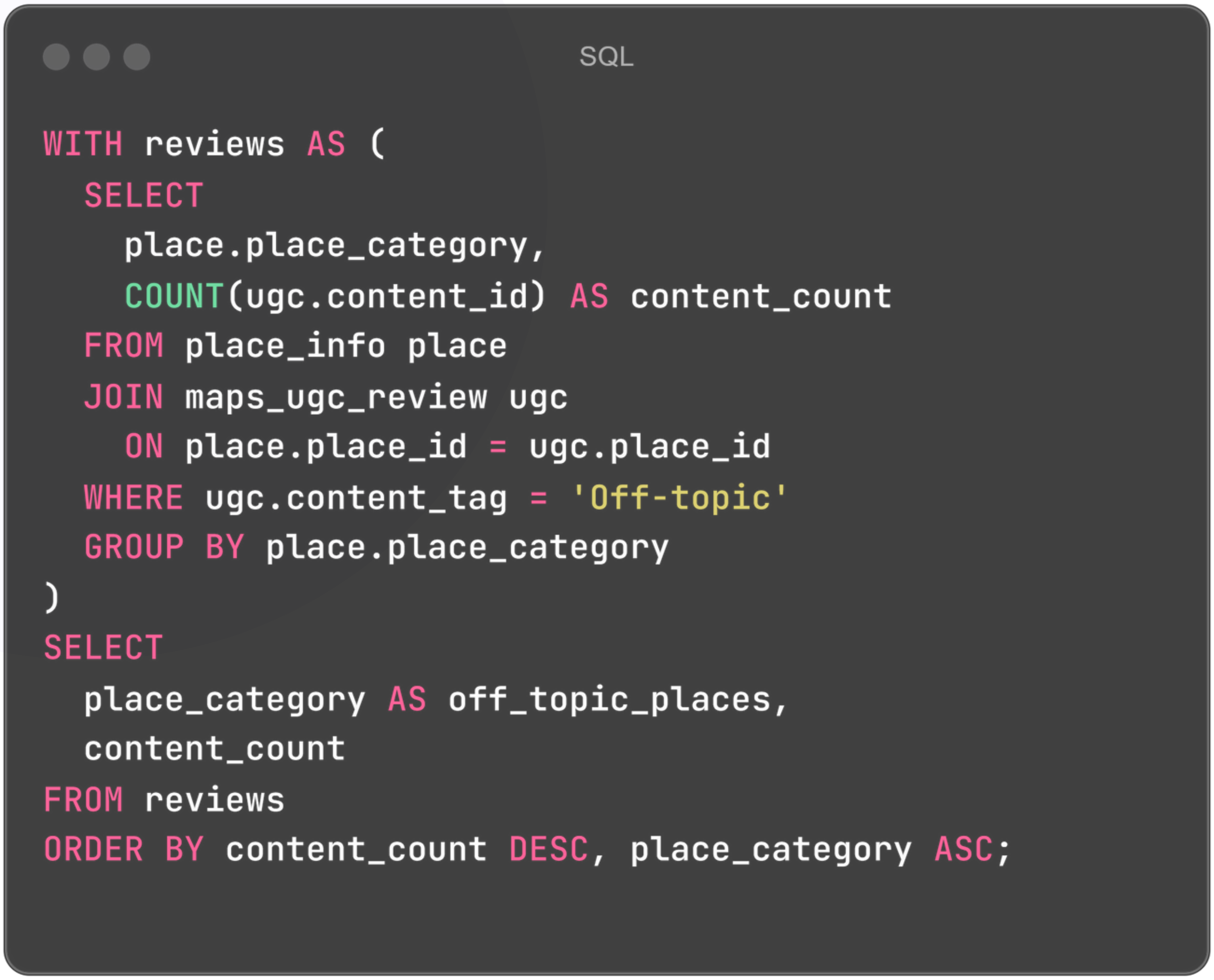
1. Count Off-Topic UGC:

   - Join place_*info* with maps_*ugc_review* on place ID. Filter UGC to include only those tagged as "Off-topic".
   - Count the occurrences of off-topic UGC for each venue category.

2. Find Top Venue Category:

   - Determine which category has the highest count of off-topic UGC.

linkedin.com/in/ileonjose



SQL

```sql
WITH reviews AS (
  SELECT
    place.place_category,
    COUNT(ugc.content_id) AS content_count
  FROM place_info place
  JOIN maps_ugc_review ugc
    ON place.place_id = ugc.place_id
  WHERE ugc.content_tag = 'Off-topic'
  GROUP BY place.place_category
)
SELECT
  place_category AS off_topic_places,
  content_count
FROM reviews
ORDER BY content_count DESC, place_category ASC;
```

linkedin.com/in/ileonjose

4. Popular Search Categories

Problem Statement: Find the total number of searches per category for the year 2024, and group the results by month. You have two tables: *searches* (with search details) and *categories* (with category names).

linkedin.com/in/ileonjose

Categories

search_id	user_id	search_date	category_id	query
1001	7654	06/01/2024 00:00:00	3001	"chicken recipe"
1002	2346	06/02/2024 00:00:00	3001	"vegan meal prep"
1003	8765	06/03/2024 00:00:00	2001	"google stocks"
1004	9871	07/01/2024 00:00:00	1001	"python tutorial"
1005	8760	07/02/2024 00:00:00	2001	"tesla stocks"

Searches

category_id	category_name
1001	"Programming Tutorials"
2001	"Stock Market"
3001	"Recipes"
4001	"Sports News"

linkedin.com/in/ileonjose

How to Solve:

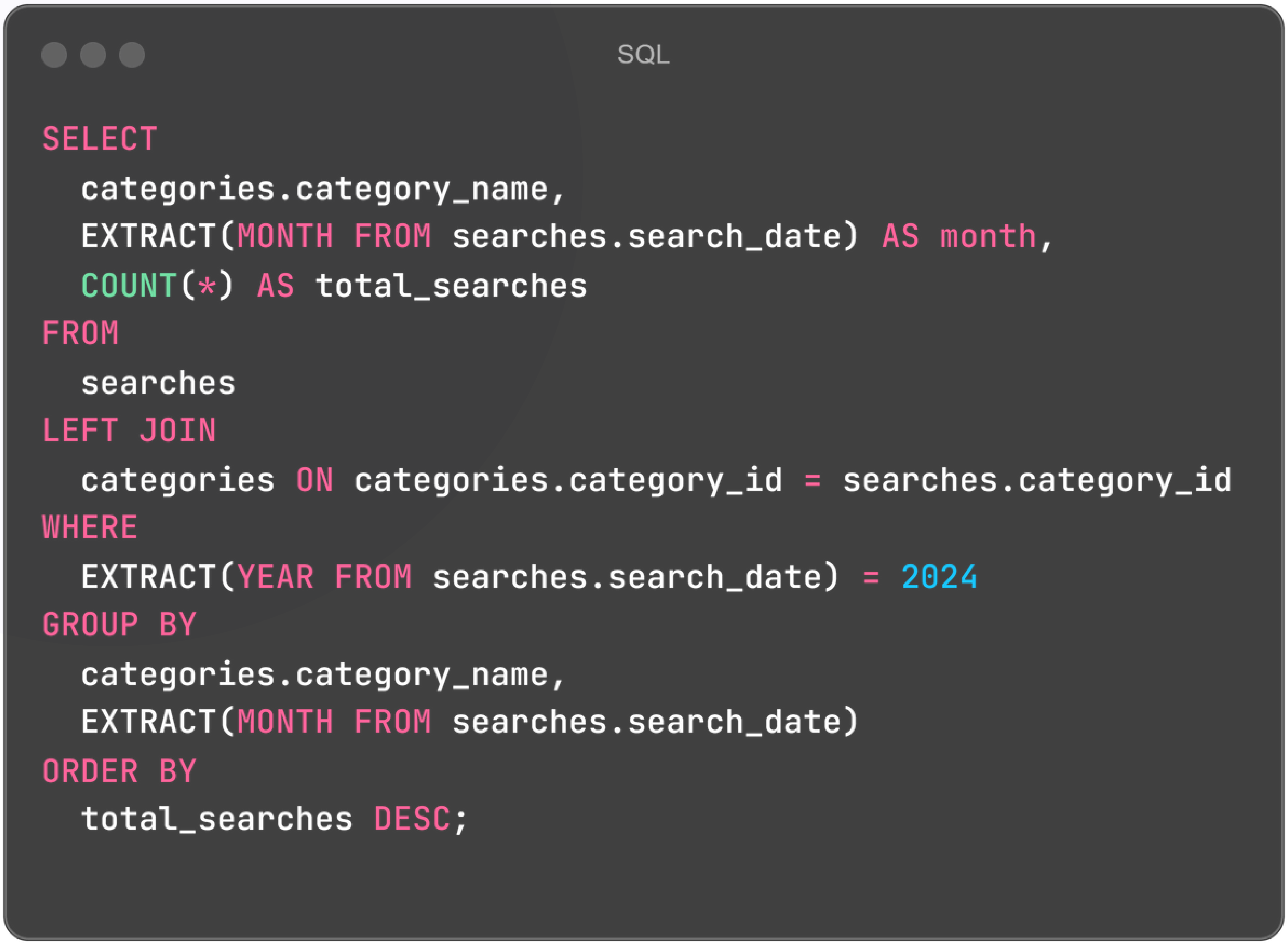
1. Join Tables:

- Combine the searches table with the categories table to include category names.

2. Count Searches:

- Aggregate the number of searches by category and month for the year 2024.

linkedin.com/in/ileonjose


SQL

```sql
SELECT
  categories.category_name,
  EXTRACT(MONTH FROM searches.search_date) AS month,
  COUNT(*) AS total_searches
FROM
  searches
LEFT JOIN
  categories ON categories.category_id = searches.category_id
WHERE
  EXTRACT(YEAR FROM searches.search_date) = 2024
GROUP BY
  categories.category_name,
  EXTRACT(MONTH FROM searches.search_date)
ORDER BY
  total_searches DESC;
```

linkedin.com/in/ileonjose

5. What is Database Denormalization?

Problem Statement: Explain the concept of denormalization in database design.

linkedin.com/in/ileonjose

Denormalization is a database design approach where tables are combined to simplify the schema and improve query performance.

This process involves introducing redundancy by merging tables, which reduces the need for complex joins and can speed up read operations.

While it may increase data redundancy, it can also improve performance and simplify certain queries.

linkedin.com/in/ileonjose

Found this helpful? Repost!

linkedin.com/in/ileonjose