SQL Practice Problems for Meta Data Science Interviews

This section provides complex SQL practice problems designed to simulate real-world scenarios at a social media company like Meta. These problems require a deeper understanding of SQL and analytical thinking. Solutions will be provided separately.

Problem Index

Problem	Difficulty	Type of Problem	Link
1	Medium	Joins, Aggregations	#problem-1-department-highest-salary
2	Medium	Self Joins, Consecutive Rows	#problem-2-consecutive-numbers
3	Medium	Subqueries, NOT IN	#problem-3-customers-who-never-order
4	Medium	Subqueries, MAX	#problem-4-second-highest-salary
5	Medium	Self Joins, Date Functions	#problem-5-rising-temperature
6	Advanced	Joins, Aggregations, Dates	#problem-6-engagement-rate-by-content-type
7	Advanced	Self Joins, Date Functions, Churn	#problem-7-churn-rate-by-user-segment
8	Advanced	Self Joins, Date Differences	#problem-8-average-time-between-posts
9	Advanced	String Manipulation, Window Functions, Dates	#problem-9-most-popular-hashtags-by-week
10	Advanced	Cohort Analysis, Date Functions	#problem-10-retained-users-by-cohort
11	Advanced	Self Joins, Counting Connections	#problem-11-mutual-connections
12	Advanced	Joins, Aggregations, Age Grouping	#problem-12-average-post-engagement-by-user-age-group
13	Advanced	Time Series Analysis, Percentage Change	#problem-13-trending-topics-over-time
14	Advanced	Funnel Analysis, Conversion Rates	#problem-14-user-engagement-funnel

Problem 1: Department Highest Salary (LeetCode 184)

Problem Statement: Write an SQL query to find employees who have the highest salary in each of their departments.

Table Schema:

- `Employee` table:
  - `Id` (int): Primary key.
  - `Name` (varchar): Employee name.
  - `Salary` (int): Employee salary.
  - `DepartmentId` (int): Foreign key referencing the `Department` table.
- `Department` table:
  - `Id` (int): Primary key.
  - `Name` (varchar): Department name.

Example:

`Employee` table:

Id	Name	Salary	DepartmentId
1	Joe	70000	1
2	Henry	80000	2
3	Sam	60000	2
4	Max	90000	1
5	Janet	69000	1
6	Randy	85000	1

`Department` table:

Id	Name
1	IT
2	Sales

Expected Output:

Department	Employee	Salary
IT	Max	90000
Sales	Henry	80000

Problem 2: Consecutive Numbers (LeetCode 180)

Problem Statement: Write an SQL query to find all numbers that appear at least three times consecutively.

Table Schema:

- `Logs` table:
  - `Id` (int): Primary key.
  - `Num` (int): The number in the log.

Example:

`Logs` table:

Id	Num
1	1
2	1
3	1
4	2
5	1
6	2
7	2

Expected Output:

Num
1

Problem 3: Customers Who Never Order (LeetCode 183)

Problem Statement: Write an SQL query to find all customers who never place an order.

Table Schema:

- `Customers` table:
  - `Id` (int): Primary key.
  - `Name` (varchar): Customer name.
- `Orders` table:
  - `Id` (int): Primary key.
  - `CustomerId` (int): Foreign key referencing the `Customers` table.

Example:

`Customers` table:

Id	Name
1	Joe
2	Henry
3	Sam
4	Max

`Orders` table:

Id	CustomerId
1	3
2	1

Expected Output:

Name
Henry
Max

Problem 4: Second Highest Salary (LeetCode 176)

Problem Statement: Write an SQL query to get the second highest salary from the `Employee` table. If there is no second highest salary, then the query should return `null`.

Table Schema:

- `Employee` table:
  - `Id` (int): Primary key.
  - `Salary` (int): Employee salary.

Example:

`Employee` table:

Id	Salary
1	100
2	200
3	300

Expected Output:

SecondHighestSalary
200

Problem 5: Rising Temperature (LeetCode 197)

Problem Statement: Write an SQL query to find all dates' `Id` with higher temperatures compared to its previous dates (yesterday's dates).

Table Schema:

- `Weather` table:
  - `Id` (int): Primary key.
  - `RecordDate` (date): Date of the weather record.
  - `Temperature` (int): Temperature on that day.

Example:

`Weather` table:

Id	RecordDate	Temperature
1	2015-01-01	10
2	2015-01-02	25
3	2015-01-03	20
4	2015-01-04	30

Expected Output:

Id
2
4

Problem 6: Engagement Rate by Content Type

Problem Statement: You are a data scientist at a social media platform. The product team is considering investing in more video content. You need to analyze user engagement across different content types (photo, video, text) to determine if this is a worthwhile investment. Write an SQL query to calculate the average engagement rate (likes + comments + shares / impressions) for each content type, broken down by month.

Table Schema:

- `Posts` table:
  - `post_id` (INT, PRIMARY KEY): Unique identifier for each post.
  - `user_id` (INT): ID of the user who created the post.
  - `content_type` (VARCHAR): Type of content ('photo', 'video', 'text').
  - `created_at` (TIMESTAMP): Timestamp of post creation.
  - `impressions` (INT): Number of times the post was shown to users.
- `Reactions` table:
  - `reaction_id` (INT, PRIMARY KEY): Unique identifier for each reaction.
  - `post_id` (INT): ID of the post that received the reaction.
  - `reaction_type` (VARCHAR): Type of reaction ('like', 'comment', 'share').

Example: (Simplified for brevity)

`Posts` table:

post_id	user_id	content_type	created_at	impressions
1	101	photo	2024-01-15 10:00:00	1000
2	102	video	2024-01-20 12:00:00	500
3	101	text	2024-02-01 09:00:00	2000
4	103	video	2024-02-10 15:00:00	1500

`Reactions` table:

reaction_id	post_id	reaction_type
1	1	like
2	1	comment
3	2	like
4	2	share
5	3	comment
6	4	like
7	4	like

reaction_id	post_id	reaction_type
8	4	share

Expected Output (Conceptual):

month	content_type	avg_engagement_rate
2024-01	photo	0.002
2024-01	video	0.004
2024-02	text	0.0005
2024-02	video	0.00133

Problem 7: Churn Rate by User Segment

Problem Statement: You are tasked with analyzing user churn. Calculate the churn rate for different user segments based on their signup date (users who signed up in the first half of the month vs. the second half). Churn is defined as users who were active in the previous month but did not have any activity in the current month.

Table Schema:

- `Users` table:
  - `user_id` (INT, PRIMARY KEY): Unique identifier for each user.
  - `signup_date` (DATE): Date when the user signed up.
- `UserActivity` table:
  - `user_id` (INT): ID of the user.
  - `activity_date` (DATE): Date of user activity.

Example: (Simplified)

`Users` table:

user_id	signup_date
1	2024-01-05
2	2024-01-20
3	2024-02-02
4	2024-02-18

`UserActivity` table:

user_id	activity_date
1	2024-01-10
1	2024-02-05
2	2024-01-25
3	2024-02-15

Expected Output (Conceptual for February 2024):

signup_segment	churn_rate
First Half	0.0
Second Half	1.0

Problem 8: Average Time Between Posts

Problem Statement: Analyze how frequently users post. Calculate the average time (in days) between posts for users who have made at least two posts.

Table Schema:

- `Posts` table:
  - `post_id` (INT, PRIMARY KEY): Unique identifier for each post.
  - `user_id` (INT): ID of the user who created the post.
  - `created_at` (TIMESTAMP): Timestamp of post creation.

Example:

`Posts` table:

post_id	user_id	created_at
1	101	2024-01-01 10:00:00
2	101	2024-01-05 12:00:00
3	102	2024-01-10 15:00:00
4	101	2024-01-12 08:00:00

Expected Output:

user_id	avg_days_between_posts
101	3

Problem 9: Most Popular Hashtags by Week

Problem Statement: Determine the most popular hashtags used each week. "Popularity" is defined by the number of posts containing the hashtag.

Table Schema:

- `Posts` table:
    - `post_id` (INT, PRIMARY KEY): Unique identifier for each post.
    - `content` (TEXT): The content of the post (may contain hashtags).
    - `created_at` (TIMESTAMP): Timestamp of post creation.

Example:

`Posts` table:

post_id	content	created_at
1	Check out this #amazing pic	2024-01-01 10:00:00
2	Another #amazing day	2024-01-03 12:00:00
3	Just a #random thought	2024-01-08 15:00:00
4	#amazing view	2024-01-10 08:00:00

Expected Output (Conceptual - assuming week starts on Monday):

week_start	hashtag	post_count
2023-12-31	#amazing	2
2024-01-07	#random	1

Problem 10: Retained Users by Cohort

Problem Statement: Analyze user retention by cohort. A cohort is defined by the month a user signed up. Calculate the percentage of users from each cohort who are still active (have at least one activity) in the current month (let's assume it's March 2024 for this example).

Table Schema:

- `Users` table:
  - `user_id` (INT, PRIMARY KEY): Unique identifier for each user.
  - `signup_date` (DATE): Date when the user signed up.
- `UserActivity` table:
  - `user_id` (INT): ID of the user.
  - `activity_date` (DATE): Date of user activity.

Example:

`Users` table:

user_id	signup_date
1	2024-01-15
2	2024-01-28
3	2024-02-05
4	2024-02-20
5	2024-03-10

`UserActivity` table:

user_id	activity_date
1	2024-01-20
1	2024-03-01
2	2024-01-30
3	2024-02-10
3	2024-03-15
5	2024-03-20

Expected Output (for March 2024):

signup_month	retention_rate
2024-01	50.0

signup_month	retention_rate
2024-02	50.0
2024-03	100.0

Clarification:

- The `signup_month` should be formatted as 'YYYY-MM'.
- The `retention_rate` should be a percentage (e.g., 50.0, 75.0, 100.0).
- If a cohort has no users active in the current month, the retention rate should be 0.0.

Problem 11: Mutual Connections

Problem Statement: You are analyzing user connections on the platform. Write an SQL query to find all pairs of users who have at least three mutual connections.

Table Schema:

- `Connections` table:
  - `user_id1` (INT): ID of the first user in the connection.
  - `user_id2` (INT): ID of the second user in the connection.
  - (Note: Connections are undirected, meaning if (1, 2) exists, (2, 1) does not need to exist. You should treat them as the same connection.)

Example:

`Connections` table:

user_id1	user_id2
1	2
1	3
1	4
2	3
2	4
2	5
3	4
3	5
4	5

Expected Output:

user1	user2
2	3
2	4
2	5
3	4
3	5
4	5

(Users 2, 3, and 4 have 3 mutual connections with user 5)

Problem 12: Average Post Engagement by User Age Group

Problem Statement: You are analyzing post engagement based on user demographics. Calculate the average number of likes, comments, and shares per post for different user age groups (e.g., 18-24, 25-34, 35+).

Table Schema:

- `Users` table:
  - `user_id` (INT, PRIMARY KEY): Unique identifier for each user.
  - `age` (INT): User's age.
- `Posts` table:
  - `post_id` (INT, PRIMARY KEY): Unique identifier for each post.
  - `user_id` (INT): ID of the user who created the post.
- `Reactions` table:
  - `reaction_id` (INT, PRIMARY KEY): Unique identifier for each reaction.
  - `post_id` (INT): ID of the post that received the reaction.
  - `reaction_type` (VARCHAR): Type of reaction ('like', 'comment', 'share').

Example: (Simplified)

(Example data would be provided similarly to previous problems)

Expected Output (Conceptual):

age_group	avg_likes	avg_comments	avg_shares
18-24	10.5	2.2	1.1
25-34	15.2	3.8	2.5
35+	8.7	1.5	0.8

Problem 13: Trending Topics Over Time

Problem Statement: Identify trending topics on the platform over time. A topic is represented by a keyword. A topic is considered trending in a given week if its usage (number of posts containing the keyword) increases by at least 20% compared to the previous week.

Table Schema:

- `Posts` table:
  - `post_id` (INT, PRIMARY KEY): Unique identifier for each post.
  - `content` (TEXT): The content of the post.
  - `created_at` (TIMESTAMP): Timestamp of post creation.

Example: (Simplified)

(Example data would be provided similarly to previous problems)

Expected Output (Conceptual - assuming week starts on Monday):

week_start	keyword	percent_increase
2024-01-07	#newtopic	25.0

Problem 14: User Engagement Funnel

Problem Statement: Analyze the user engagement funnel for a specific feature (e.g., creating a story). Calculate the conversion rates between different stages of the funnel:

1. Viewed the feature introduction.
2. Started creating a story.
3. Successfully posted a story.

Table Schema:

- `UserActions` table:
  - `user_id` (INT): ID of the user.
  - `action` (VARCHAR): Type of user action ('viewed_story_intro', 'started_story_creation', 'posted_story').
  - `action_timestamp` (TIMESTAMP): Timestamp of the action.

Example: (Simplified)

(Example data would be provided similarly to previous problems)

Expected Output (Conceptual):

stage	conversion_rate
Viewed Intro -> Started Creation	60.0
Started Creation -> Posted Story	75.0