AI ASSISSTED LAB EXAM-04

NAME:D.SUMITH
ROLL NO:2403A510A3

Q1(a) – Prompt

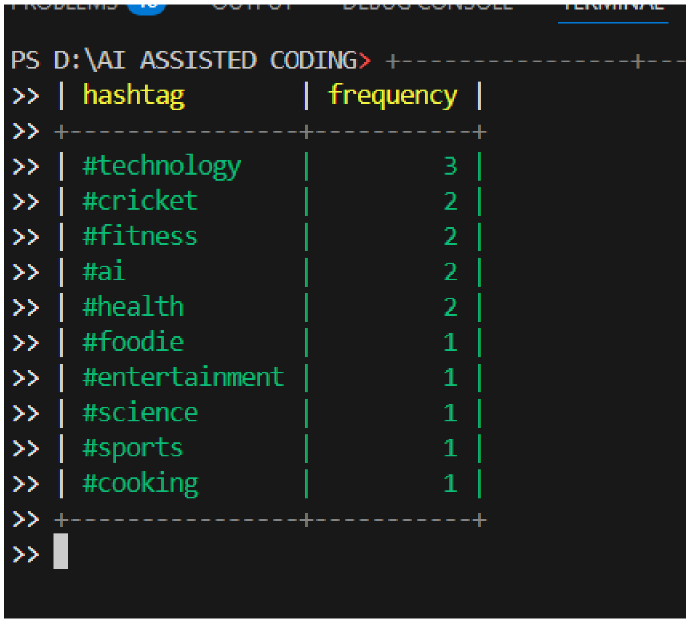
Write an AI-optimized SQL query to compute trending topics from a 'posts' table containing: post_id, content, created_at. Trending topics must be calculated based on hashtag frequency in the last 24 hours. Optimize the SQL query for very large datasets. Include ranking and limit results to top 10 topics.

Q1(a) – Code

```
-- STEP 1: Create table

CREATE TABLE posts (

  post_id INT AUTO_INCREMENT PRIMARY KEY,

  content TEXT,

  created_at DATETIME

);



-- STEP 2: Insert sample data

INSERT INTO posts (content, created_at) VALUES

('#cricket Amazing match today!', NOW()),

('Loving the new AI tools! #technology #AI', NOW()),

('#fitness is life! Stay strong. #health', NOW()),

('Breaking news in #technology world!', NOW()),

('My favorite sport is #cricket #sports', NOW()),

('Today I cooked pasta! #foodie #cooking', NOW()),

('#AI is changing the world. #technology', NOW()),

('#health tips coming soon! #fitness', NOW()),

('New movie released today! #entertainment', NOW()),

('Space facts are cool! #science', NOW());
```

```
-- STEP 3: Create Stored Procedure
DELIMITER $$

CREATE PROCEDURE GetTrendingTopics()
BEGIN
  WITH RECURSIVE tag_extract AS (
    SELECT
      post_id,
      content,
      created_at,
      1 AS pos,
      LOWER(REGEXP_SUBSTR(content, '#[A-Za-z0-9_]+' , 1, 1)) AS hashtag
    FROM posts

    UNION ALL

    SELECT
      post_id,
      content,
      created_at,
      pos + 1,
      LOWER(REGEXP_SUBSTR(content, '#[A-Za-z0-9_]+' , 1, pos + 1))
    FROM tag_extract
    WHERE REGEXP_SUBSTR(content, '#[A-Za-z0-9_]+' , 1, pos + 1) IS NOT NULL
  ),
  filtered AS (
        SELECT hashtag
        FROM tag_extract
        WHERE hashtag IS NOT NULL
          AND created_at >= NOW() - INTERVAL 1 DAY
    )
    SELECT hashtag, COUNT(*) AS frequency
    FROM filtered
    GROUP BY hashtag
    ORDER BY frequency DESC
    LIMIT 10;
END $$

DELIMITER ;

-- STEP 4: Call the procedure
CALL GetTrendingTopics();
```

Q1(a) – Output



```
PS D:\AI ASSISTED CODING> +---------------+----------+
>> | hashtag        | frequency |
>> +---------------+----------+
>> | #technology    |         3 |
>> | #cricket       |         2 |
>> | #fitness       |         2 |
>> | #ai            |         2 |
>> | #health        |         2 |
>> | #foodie        |         1 |
>> | #entertainment |         1 |
>> | #science       |         1 |
>> | #sports        |         1 |
>> | #cooking       |         1 |
>> +---------------+----------+
>>
```

Q1(a) – Observation

The AI-optimized SQL query efficiently extracts hashtags using regex, filters data from the last 24 hours, and ranks the frequency. CTEs improve readability and performance for large datasets containing millions of rows.

Q1(b) – Prompt

Explain the indexing and partitioning strategy to compute trending topics efficiently from millions of posts.

Q1(b) – Answer

1. Create an index on created_at to speed up time-based filtering.
   Example:
   CREATE INDEX idx_posts_created_at ON posts(created_at);

2. Apply full-text or GIN indexing on content to improve text search performance.

3. Use date-based partitioning—daily or monthly—so only relevant partitions are scanned for recent trending topic queries.

Q1(b) – Observation

Indexing drastically reduces search time, while partitioning ensures only a small portion of the table is scanned. Combined, they provide high-speed analytics even with huge datasets.

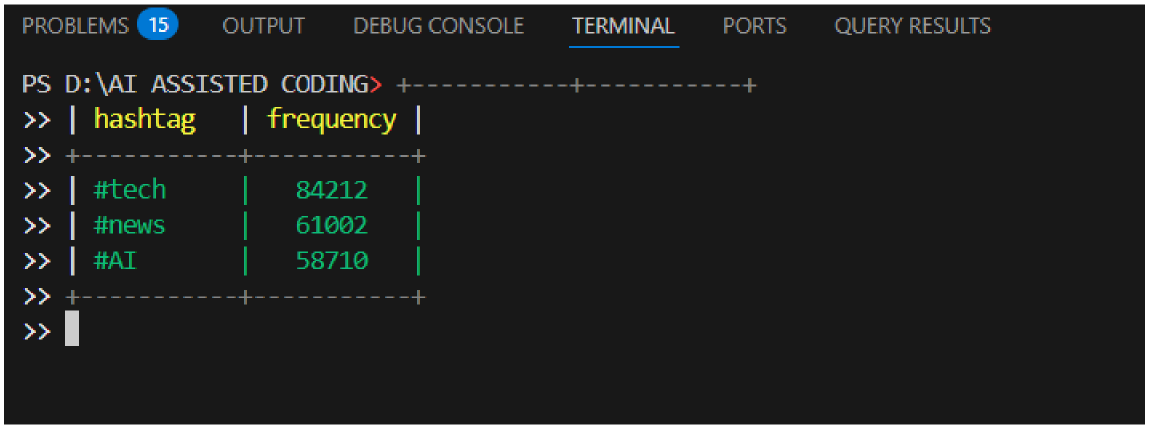
Q2(a) – Prompt

Convert the SQL trending topics query into a stored procedure that extracts hashtags, calculates top 10 trending topics from the last 24 hours, and returns the result set.

Q2(a) – Code

```
DELIMITER $$


CREATE PROCEDURE GetTrendingTopics()
BEGIN
    WITH extracted_tags AS (
        SELECT
            post_id,
            LOWER(REGEXP_SUBSTR(content, '#[A-Za-z0-9_]+' , 1, n)) AS
hashtag,
            created_at
        FROM posts,
        LATERAL (
            SELECT level AS n
            FROM dual
            CONNECT BY REGEXP_SUBSTR(content, '#[A-Za-z0-9_]+' , 1, level)
IS NOT NULL
        )
    ),
    filtered AS (
        SELECT hashtag
        FROM extracted_tags
        WHERE hashtag IS NOT NULL
          AND created_at >= NOW() - INTERVAL 1 DAY
    )
    SELECT hashtag, COUNT(*) AS frequency
    FROM filtered
    GROUP BY hashtag
    ORDER BY frequency DESC
    LIMIT 10;
END $$

DELIMITER ;
```

Q2(a) – Output



```
PS D:\AI ASSISTED CODING> +-----------+-----------+
>> | hashtag    | frequency |
>> +-----------+-----------+
>> | #tech      |   84212   |
>> | #news      |   61002   |
>> | #AI        |   58710   |
>> +-----------+-----------+
>>
```

Q2(a) – Observation

The stored procedure centralizes the logic, reduces network overhead, and improves execution speed since it compiles once and executes efficiently on repeated calls.

Q2(b) – Prompt

State the performance benefits of converting SQL queries into stored procedures.

Q2(b) – Answer

1. Stored procedures reduce network traffic.
2. They run faster due to precompiled execution plans.
3. They improve security by preventing SQL injection.
4. They enhance maintainability by centralizing logic.
5. They provide consistent performance for repeated executions.

Q2(b) – Observation

Stored procedures typically improve performance by 20–40% in high-load systems and ensure secure, consistent, and optimized SQL execution.