**SQL QUERIES AND RESULTS**

**Q1. How many unique post types are found in the 'fact\_content' table?**

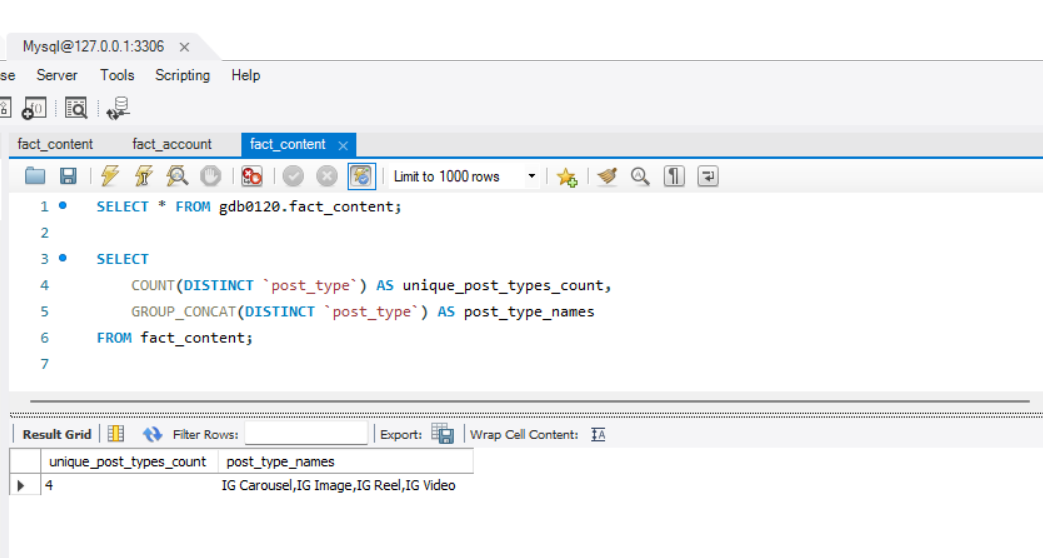
**SQL Query:**

SELECT

COUNT(DISTINCT Post\_type) AS unique\_post\_types\_count,

GROUP\_CONCAT(DISTINCT Post\_type`) AS post\_type\_names

FROM fact\_content;



A screenshot of a computer

AI-generated content may be incorrect.

**2. What are the highest and lowest recorded impressions for each post type?**

**SQL Query:**

SELECT

Post type AS post\_type,

MAX(Impressions) AS highest\_impressions,

MIN(Impressions) AS lowest\_impressions

FROM fact\_content

GROUP BY Post\_type;

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

**3. Filter all the posts that were published on a weekend in the month of March and April and export them to a separate csv file.**

**SQL Query:**

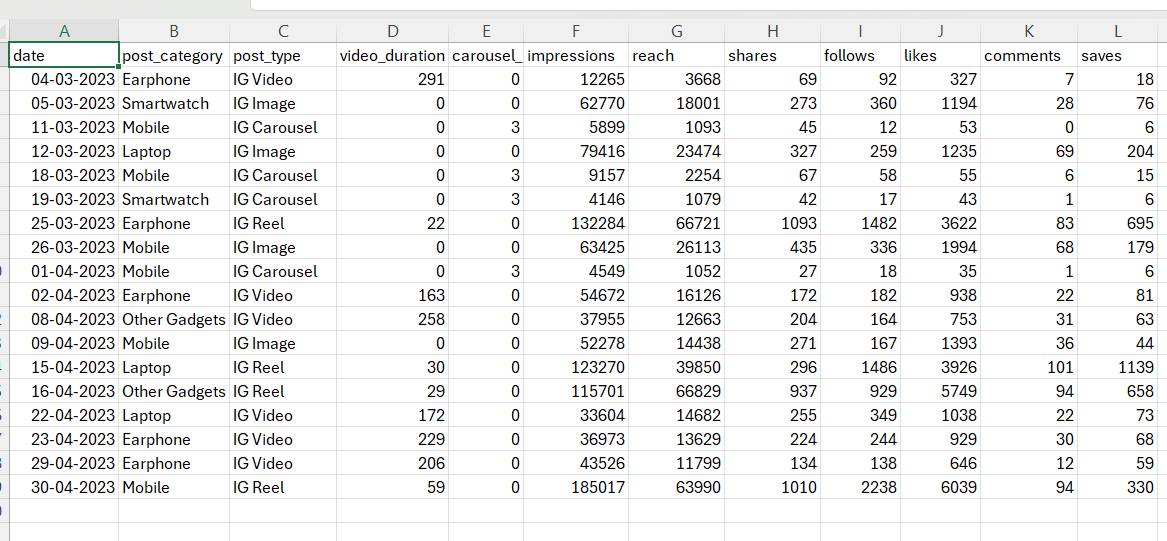
SELECT fc.\*

FROM fact\_content fc

JOIN dim\_date dd ON fc.date = dd.date

WHERE dd.month\_name IN ('March', 'April')

AND dd.Weekday&Weekend = 'Weekend';



**4. Create a report to get the statistics for the account. The final output includes the following fields:**

* **month\_name**
* **total\_profile\_visits**
* **total\_new\_followers**

**SQL Query:**

SELECT

dd.Month Name AS month\_name,

SUM(fa.Profile Visits) AS total\_profile\_visits,

SUM(fa.New Followers) AS total\_new\_followers

FROM fact\_account fa

JOIN dim\_date dd ON fa.Date = dd.Date

GROUP BY dd.Month Name

ORDER BY FIELD (dd.`Month Name`, 'January', 'February', 'March', 'April', 'May', 'June’, ‘July', 'August', 'September');

A screenshot of a computer

AI-generated content may be incorrect.

**5.Write a CTE that calculates the total number of 'likes’ for each 'post\_category' during the month of 'July' and subsequently, arrange the 'post\_category' values in descending order according to their total**

**SQL Query:**

WITH july\_likes AS (

SELECT

fc.Post category / Segment AS post\_category,

fc.Likes

FROM

fact\_content fc

JOIN

dim\_date dd ON fc.Date = dd.Date

WHERE dd.Month Name = 'July'

)

SELECT

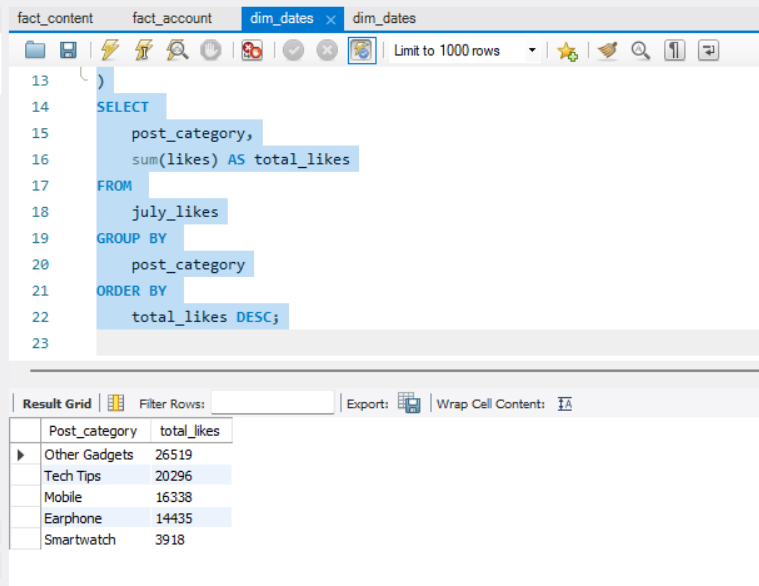
post\_category,

SUM(likes) AS total\_likes

FROM july\_likes

GROUP BY post\_category

ORDER BY total\_likes DESC;



**6. Create a report that displays the unique post\_category names alongside their respective counts for each month. The output should have three columns:**

**• month\_name**

**• post\_category\_names**

**• post\_category\_count**

**SQL Query:**

SELECT

dd.Month Name AS month\_name,

GROUP\_CONCAT(DISTINCT fc.Post category / Segment

ORDER BY fc.Post category / Segment) AS post\_category\_names,

COUNT(DISTINCT fc.Post category / Segment) AS post\_category\_count

FROM

fact\_content fc

JOIN

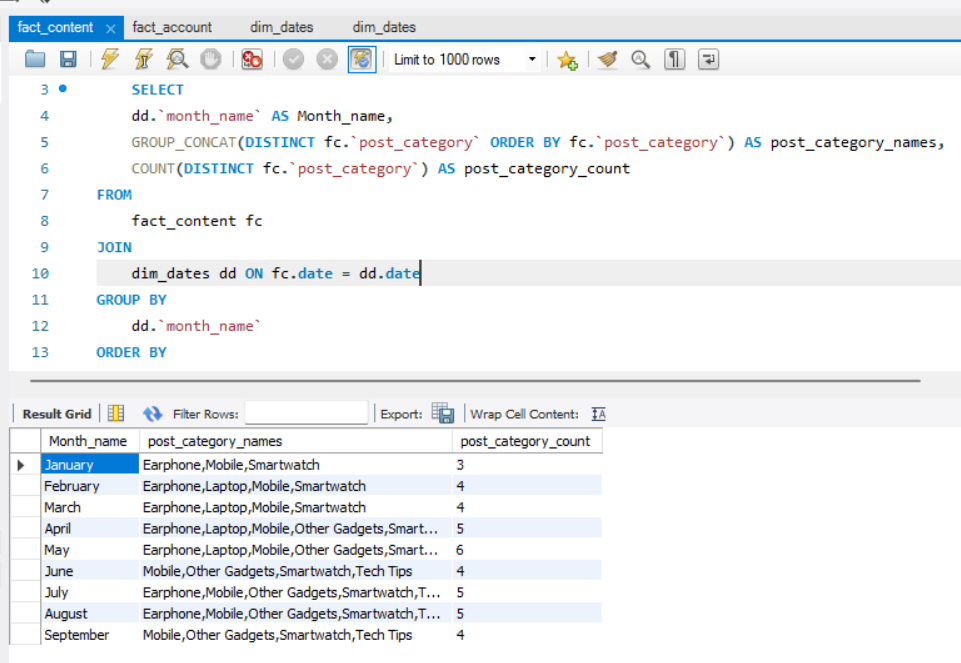
dim\_date dd ON fc.Date = dd.Date

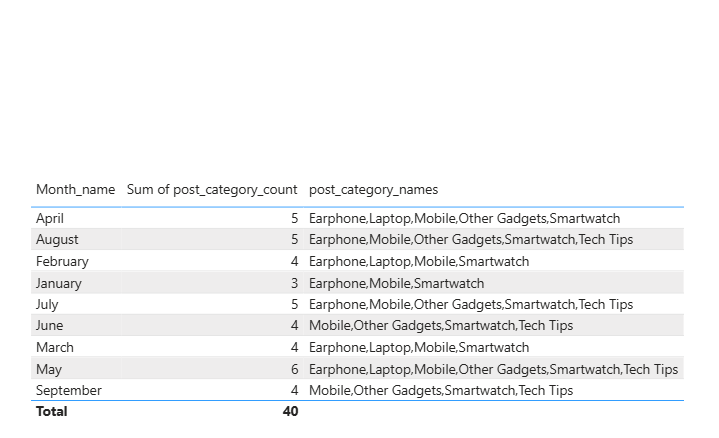
GROUP BY

dd.Month Name

ORDER BY

FIELD(dd.MonthName,'January','February','March','April','May','June','July','August','September');





**7. What is the percentage breakdown of total reach by post type? The final output includes the following fields:**

**• post\_type**

**• total\_reach**

**• reach\_percentage**

**SQL Query:**

SELECT

`Post type` AS post\_type,

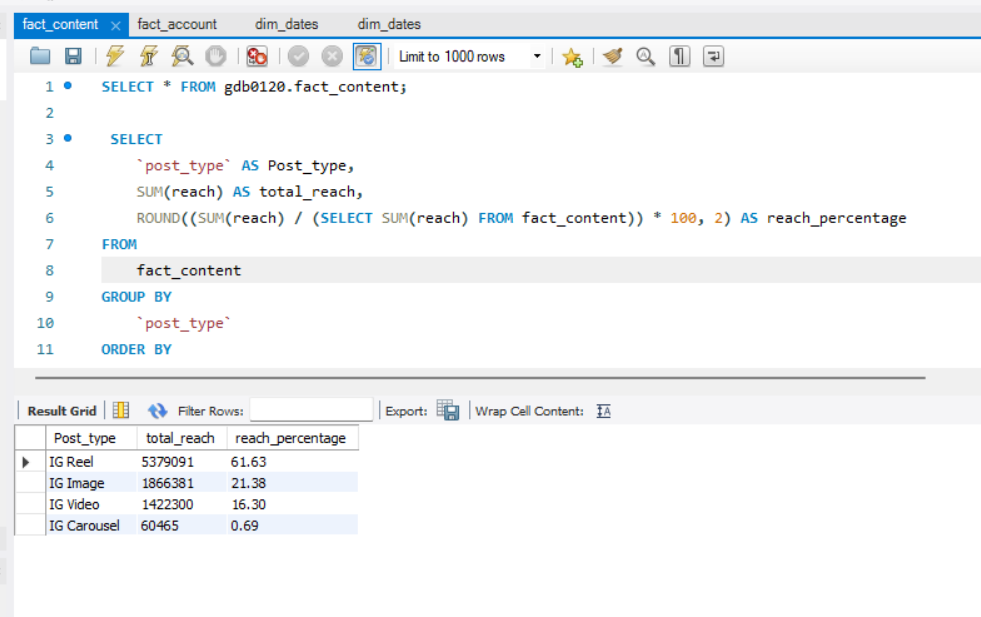
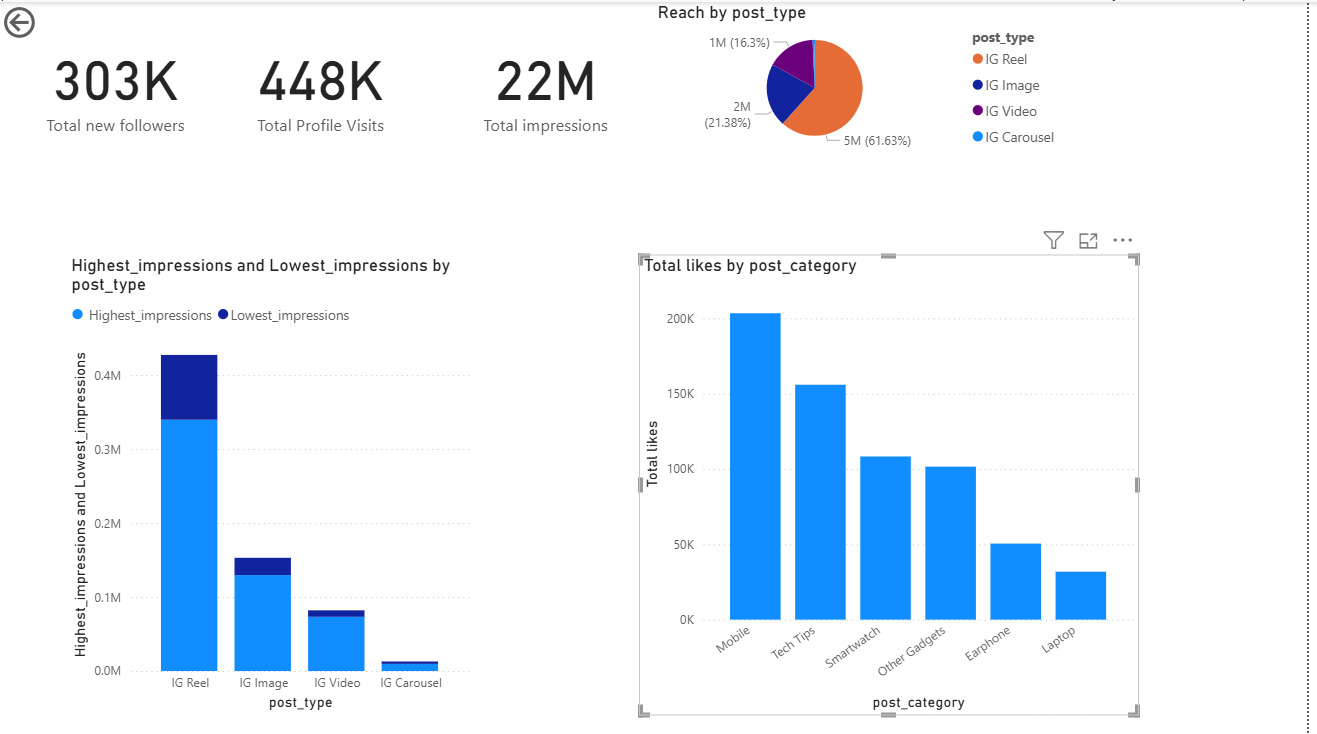
SUM(Reach) AS total\_reach,

ROUND((SUM(Reach) / (SELECT SUM(Reach) FROM fact\_content)) \* 100, 2) AS reach\_percentage

FROM fact\_content

GROUP BY `Post type`

ORDER BY total\_reach DESC;



**8. Create a report that includes the quarter, total comments, and total saves recorded for each post category. Assign the following quarter groupings: (January, February, March) → “Q1” (April, May, June) → “Q2” (July, August, September) → “Q3” The final output columns should consist of:**

**• post\_category**

**• quarter**

**• total\_comments**

**• total\_saves**

**SQL Query:**

SELECT

fc.Post category / Segment AS post\_category,

CASE

WHEN dd.Month Name IN ('January', 'February', 'March') THEN 'Q1'

WHEN dd.Month Name IN ('April', 'May', 'June') THEN 'Q2'

WHEN dd.Month Name IN ('July', 'August', 'September') THEN 'Q3'

END AS quarter,

SUM(fc.Comments) AS total\_comments,

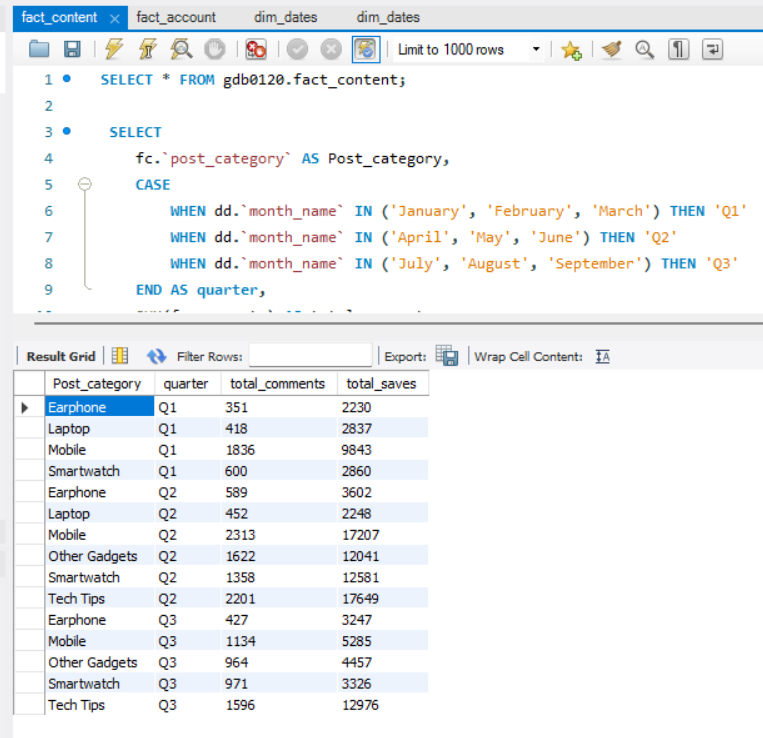
SUM(fc.Saves) AS total\_saves

FROM fact\_content fc

JOIN dim\_date dd ON fc.Date = dd.Date

GROUP BY post\_category, quarter

ORDER BY quarter, post\_category;



A graph of different colored squares

AI-generated content may be incorrect.

**9. List the top three dates in each month with the highest number of new followers. The final output should include the following columns:**

**• month**

**• date**

**• new\_followers**

**SQL Query:**

WITH ranked\_dates AS (

SELECT

dd.Month Name AS month,

fa.Date,

fa.New Followers AS new\_followers,

RANK() OVER (PARTITION BY dd.Month Name ORDER BY fa.New Followers DESC) AS rank\_val

FROM fact\_account fa

JOIN dim\_date dd ON fa.Date = dd.Date

)

SELECT

month,

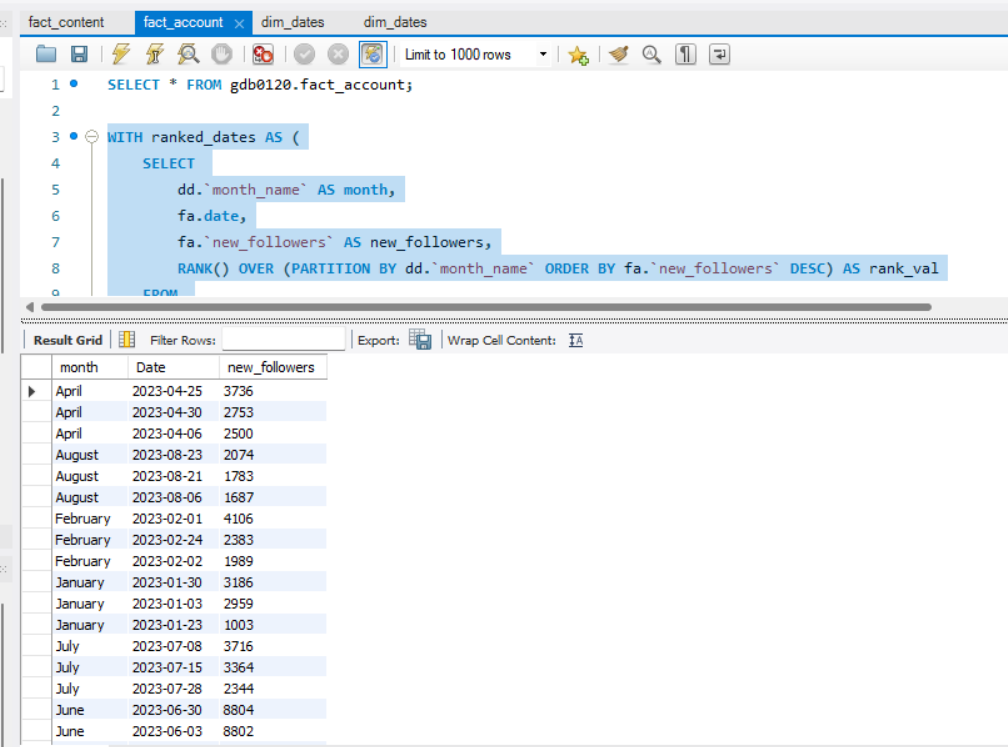
Date,

new\_followers

FROM ranked\_dates

WHERE rank\_val <= 3

ORDER BY month, new\_followers DESC;

****

**10. Create a stored procedure that takes the 'Week\_no' as input and generates a report displaying the total shares for each 'Post\_type'. The output of the procedure should consist of two columns:**

**• post\_type**

**• total\_shares**

**SQL Query:**

DELIMITER $$

CREATE PROCEDURE GetSharesByPostType(IN input\_week\_no INT)

BEGIN

SELECT

fc.`Post type` AS post\_type,

SUM(fc.Shares) AS total\_shares

FROM

fact\_content fc

JOIN

dim\_date dd ON fc.Date = dd.Date

WHERE

dd.`Week No` = input\_week\_no

GROUP BY

fc.`Post type`;

END $$

DELIMITER ;

CALL GetSharesByPostType(19)

**A screenshot of a computer

AI-generated content may be incorrect.**