Stack Overflow Post Analysis

The main objective of this project is to analyze the history of Stack Overflow dataset, including user activity, badges, edits and comments using SQL, focusing on user contributions, post engagement and content trends.

Tables Used

This dataset includes 10 rows per table. The tables are as follows:

- Badges – Tracks badges earned by users.
- Comments - Contains comments on posts.
- Post History - Tracks the history of edits, comments, and other changes made to posts.
- Post Links - Links between related posts.
- Posts - Information about posts.
- Posts Answers - Contains questions and answers.
- Tags - Information about tags associated with posts.
- Users - Details about Stack Overflow users.
- Votes - Tracks voting activity on posts.

- **Total records in each table:**
  - Badges = 10
  - Comments = 10
  - Post History = 10
  - Post Links = 10
  - Posts = 10
  - Posts Answers = 10
  - Tags = 10
  - Users = 10
  - Votes = 10

SQL Queries and Insights

- **Exploring Data**
  SQL Query

```
select * from badges;
select * from posts;
select * from users;
```

Insight = The above query includes the badges name, post activity and users name.

SQL Query

- **Most Engaging Posts (view count > 100)**

```
select *
from posts
where view_count > 100;
```

Result

	id	title	creation_date	score	view_count
▶	2002	Best practices for writing SQL queries	2023-01-02	15	150
	2003	Understanding INNER JOIN in SQL	2023-01-03	20	200
	2004	What is a LEFT JOIN?	2023-01-04	25	250
	2005	Database indexing techniques	2023-01-05	30	300
	2006	Explaining SQL subqueries	2023-01-06	35	350
	2007	How to optimize SQL queries?	2023-01-07	40	400
	2008	Database normalization concepts	2023-01-08	45	450
	2009	SQL Aggregate Functions explained	2023-01-09	50	500
	2010	Introduction to SQL Window Functions	2023-01-10	55	550
*	NULL	NULL	NULL	NULL	NULL

Insight = High Engagement posts belong to topics like **SQL**.

SQL Query

- **Most Earned Badges**

```
select name, count(id) as badges_count
from badges
group by name
order by badges_count desc;
```

Result

	name	badges_count
▶	Gold Contributor	4
	Silver Helper	3
	Bronze Reviewer	3

Insight =

- Badge (**Gold Contributor**) is the most frequently earned badge with the count of **4**.
- Users with higher engagement earn more badges.

SQL Query

- **Users with most Badges**

```
select u.id, u.display_name, count(b.id) as total_badges
from users u
join badges b on u.id = b.user_id
group by u.id
order by total_badges desc;
```

Result

	id	display_name	total_badges
▶	1001	Alice	4
	1002	Bob	2
	1003	Charlie	2
	1004	Dave	2

Insight = Alice has earned the most badges **(Gold Contributor)** with the count of **4**.

SQL Query

- **Users with highest reputation**

```
select id, display_name, reputation
from users
where reputation = (select max(reputation)
                    from users);
```

Result

	id	display_name	reputation
▶	1006	Frank	2000
*	NULL	NULL	NULL

Insight = Frank has the highest reputation of **2000**, is likely to be the top contributor.

SQL Query

- **Tags with High-Scoring Posts**

```
select t.tag_name, p.title, p.score
from tags t
join posts p on t.id = p.owner_user_id
where p.score = (select max(score) from posts);
```

Result

	tag_name	title	score
▶	Database	Introduction to SQL Window Functions	55

Insight = SQL Database is **among the high-scoring tags** on Stack Overflow.

SQL Query

- **Top Contributors (In terms of Comments, Edits and Votes)**

```
select u.id, u.display_name,
                    count(distinct c.id) as total_comments,
       count(distinct ph.id) as total_edits,
       count(distinct v.id) as total_votes,
       (count(distinct c.id) + count(distinct ph.id) + count(distinct v.id)) as top_contributors
from comments c
join users u on c.user_id = u.id
join post_history ph on c.user_id = ph.user_id
join votes v on c.post_id = v.post_id
group by u.id;
```

Result

	id	display_name	total_comments	total_edits	total_votes	top_contributors
▶	1001	Alice	3	3	4	10
	1002	Bob	3	3	4	10
	1003	Charlie	2	2	2	6
	1004	Dave	2	2	2	6

Insight =

- **Alice (user 1001) and Bob (user 1002)** are the most engaged users.
- They contribute through votes most rather than comments and edits.

- **Most discussed Posts**

```
select p.id, p.title, count(c.id) as total_comments
from posts p
join comments c on p.id = c.post_id
group by p.id
order by total_comments desc;
```

Result

	id	title	total_comments
▶	2001	How to solve SQL JOIN issues?	2
	2003	Understanding INNER JOIN in SQL	2
	2004	What is a LEFT JOIN?	2
	2005	Database indexing techniques	2
	2002	Best practices for writing SQL queries	1
	2006	Explaining SQL subqueries	1

Insight =

- Mostly, the posts discussion focus on the concepts like **SQL**.
- Posts about SQL Joins, Subqueries and indexing have the highest discussion.