**Social Media Analytics Backend**

**Project Report – Elevate Labs Internship**

**Name:** Prem Kumar

**Project Title**: Social Media Engagement Analytics Backend

**Organization**: Elevate Labs

**Duration**: July 2025

**Tools** **Used**: PostgreSQL, pgAdmin

**—————————————————————————————————————————**

## 1. Introduction

This project was part of my internship at Elevate Labs where I was required to build a backend system that captures and analyzes user engagement data from a simulated social media platform.  
  
As a beginner, my goal was to understand how real-world platforms track user activity like likes, comments, and follows. I chose SQL and PostgreSQL to build everything from the ground up without relying on external libraries or frameworks.

## 2. Abstract

To create a backend analytics system that:  
- Stores users, posts, comments, likes, and follow relationships  
- Automatically tracks engagement using triggers  
- Generates insights through views and analytical SQL queries  
  
This helped me practice how data flows and how to make sense of it through raw SQL.

**STEPS INVOLVED IN BUILDING THE PROJECT**

## 3. Database Schema Design

The project uses six interconnected tables:  
- **Users:** Stores user information  
- **Posts:** Contains content created by users  
- **Likes:** Tracks which users liked which posts  
- **Comments:** Tracks which users commented on which posts  
- **Followers:** Represents follower-following relationships  
- **PostAnalytics:** A dynamic table that stores engagement metrics for each post  
  
These tables were designed with proper foreign key relationships for integrity.

## 4. Automating Engagement Calculation

I used PostgreSQL triggers to automate updates in the PostAnalytics table.  
- **Trigger events:** AFTER INSERT on Likes and Comments  
**- Effect:** Automatically updates the corresponding post's engagement stats  
- **Formula used:**  
engagement score = likes\_count \* 1 + comments\_count \* 1.5  
  
This saved me from writing manual update queries.

## 5. Smart Views for Insight

I created the following SQL views to help summarize and explore the data:  
**- top\_5\_engaging\_posts:** Ranks posts based on engagement score  
**- most\_followed\_users:** Shows users with the highest follower count  
**- active\_users\_summary:** Aggregates user activity (likes, comments, posts)  
  
These views made querying and reporting very easy.

## 6. Analytical Queries with RANK() and Aggregates

To go beyond static views, I added:  
**- Ranked post engagement:** Using RANK () over engagement score  
**- Top users by total engagement:** Using SUM () and RANK ()  
**- Zero engagement post detection:** Identifying posts with no likes or comments  
  
These queries gave deeper insights into which users or posts were inactive or most engaging.

## 7. Results and Outputs

I ran multiple SQL queries and saved:  
- Screenshots of data tables and outputs (e.g., post analytics, top users)  
- Exported .csv files for every key result table  
  
This helped ensure that I could submit both visual and data proof of my work.

## 8. Learnings & Takeaways

- I now understand how SQL triggers automate backend logic  
- Learned how to write clean, readable analytical queries  
- Practiced creating views and working with relational data  
- Got better at problem-solving using just SQL and logic  
- Realized how important backend analytics is for any digital platform

## 9. Conclusion

This project helped me move from writing simple SELECT queries to building a full backend logic for engagement analytics. I followed all internship rules and worked independently. I'm confident this project built a strong foundation in SQL and real-world data logic.  
  
**Submitted by:**  
Prem Kumar Kurukanti.  
SQL Intern, Elevate Labs – July 2025.