**Data Analyst Technical Interview**

**Submitted By: - Sagar Purswani (purswanisagar60@gmail.com)**

## **Technical Round 1 – Analytics Proposal**

In this document I have proposed 20 analytics (with sample code in python) that can be performed on the given schema, covering Group Info, Member Info, and Message Info. These analytics provide insights into engagement, activity, and content trends, helping administrators optimize group management.

### **1. Daily Messages Sent**

* **Description**: Tracks the number of messages sent daily in each group.
* **Purpose:** Helps identify active days and peak engagement times.
* **Benefit:** Enables group admins to schedule important announcements or events during high-activity periods.
* **Sample Code (In Pyhton):**

daily\_messages = messages\_df.groupby([messages\_df["timestamp"].dt.date, "group\_id"]).size().reset\_index(name="message\_count")

### **2.Most Active Members**

* **Description**: Identifies the members who send the most messages in each group.
* **Purpose**: Helps determine key contributors and engagement leaders.
* **Benefit**: Admins can reward or engage top contributors to boost community interaction.
* **Sample Code (In Pyhton):**

active\_members = messages\_df.groupby(["sender\_id", "group\_id"]).size().reset\_index(name="message\_count")

### **3. Most Used Hashtags**

* **Description:** Identifies the most frequently used hashtags in messages.
* **Purpose:** Helps determine trending topics within the group.
* **Benefit:** Allows admins to tailor discussions, announcements, and content based on member interests.
* **Sample Code (In Python):**

all\_hashtags = sum([json.loads(x) for x in messages\_df["hashtags"]], []) hashtag\_counts = pd.Series(all\_hashtags).value\_counts()

print(hashtag\_counts)

### **4. Most Shared URLs**

* **Description:** Finds the most frequently shared links in messages.
* **Purpose:** Identifies popular shared resources and detects potential spam.
* **Benefit:** Admins can track valuable links and ensure the quality of shared content.
* **Sample Code (In Python):**

all\_urls = sum([json.loads(x) for x in messages\_df["urls"]], []) url\_counts = pd.Series(all\_urls).value\_counts()

print(url\_counts)

### **5. Message Views Analysis**

* **Description:** Calculates the average number of views per message in each group.
* **Purpose:** Measures message reach and engagement.
* **Benefit:** Helps admins understand which types of content attract the most attention.
* **Sample Code (In Python):**

average\_views = messages\_df.groupby("group\_id")["views"].mean().reset\_index()

print(average\_views)

### **6. Replies Per Message**

* **Description:** Determines the average number of replies per message in a group.
* **Purpose:** Measures engagement by analyzing how often members interact with messages.
* **Benefit:** Helps admins encourage more discussions and increase participation.
* **Sample Code (In Python):**

avg\_replies = messages\_df["replies"].mean()

print("Average replies per message:", avg\_replies)

### **7. Most Forwarded Messages**

* **Description:** Identifies the most frequently forwarded messages.
* **Purpose:** Highlights viral or high-value messages.
* **Benefit:** Admins can use this insight to amplify important messages.
* **Sample Code (In Python):**

top\_forwarded = messages\_df.sort\_values(by="forwards", ascending=False).head(3)

print(top\_forwarded)

### **8. New Members Per Day**

* **Description:** Tracks the number of new members joining daily.
* **Purpose:** Helps monitor group growth trends.
* **Benefit:** Enables admins to measure the effectiveness of promotions or invitations.
* **Sample Code (In Python):**

new\_members = members\_df.groupby(members\_df["join\_date"].dt.date).size().reset\_index(name="new\_members\_count")

print(new\_members)

### **9. Inactive Members**

* **Description:** Identifies members who have not sent any messages recently.
* **Purpose:** Helps track inactive users and re-engage them.
* **Benefit:** Admins can clean up inactive members or encourage them to participate.
* **Sample Code (In Python):**

last\_active = messages\_df.groupby("sender\_id")["timestamp"].max().reset\_index() inactive\_members = members\_df[~members\_df["user\_id"].isin(last\_active["sender\_id"])] print(inactive\_members)

### **10. Group Size Distribution**

* **Description:** Categorizes groups into small, medium, and large based on member count.
* **Purpose:** Helps segment groups based on engagement scale.
* **Benefit:** Admins can apply different engagement strategies based on group size.
* **Sample Code (In Python):**

conditions = [ (groups\_df["member\_count"] < 50), (groups\_df["member\_count"].between(50, 200)), (groups\_df["member\_count"] > 200) ]

categories = ["Small", "Medium", "Large"]

groups\_df["size\_category"] = pd.np.select(conditions, categories)

print(groups\_df)

### **11. Admin to Member Ratio**

* **Description:** Calculates the ratio of admins to regular members in each group.
* **Purpose:** Ensures proper moderation and governance in groups.
* **Benefit:** Helps admins determine if additional moderators are needed.
* **Sample Code (In Python):**

groups\_df["admin\_to\_member\_ratio"] = groups\_df["admin\_count"] / groups\_df["member\_count"]

### **12. Bots Presence in Groups**

* **Description:** Identifies the percentage of bots in each group.
* **Purpose:** Helps detect groups heavily influenced by bots.
* **Benefit:** Ensures that discussions are primarily human-driven.
* **Sample Code (In Python):**

bots\_percentage = (groups\_df["number\_of\_bots"] / groups\_df["member\_count"]) \* 100

groups\_df["bot\_percentage"] = bots\_percentage.fillna(0) print(groups\_df[["group\_id", "bot\_percentage"]])

### **13. Most Common Message Types**

* **Description:** Determines the most frequently used message types (text, media, etc.).
* **Purpose:** Helps admins understand content trends.
* **Benefit:** Assists in tailoring engagement strategies.
* **Sample Code (In Python):**

message\_type\_counts = messages\_df["message\_type"].value\_counts()

### **14. Most Popular Message Times**

* **Description:** Identifies peak hours when members are most active.
* **Purpose:** Helps optimize the timing of announcements and discussions.
* **Benefit:** Ensures maximum reach for important messages.
* **Sample Code (In Python):**

messages\_df["hour"] = messages\_df["timestamp"].dt.hour

popular\_hours = messages\_df["hour"].value\_counts().sort\_index() print(popular\_hours)

### **15. Engagement Score per Member**

* **Description:** Computes an engagement score based on messages sent, replies received, and messages forwarded.
* **Purpose:** Helps rank users based on their participation.
* **Benefit:** Encourages active engagement and helps identify community influencers.
* **Sample Code (In Python):**

messages\_df["engagement\_score"] = messages\_df["replies"] + messages\_df["forwards"]

member\_engagement = messages\_df.groupby("sender\_id")["engagement\_score"].sum().reset\_index()

print(member\_engagement)

### **16. Longest Active Members**

* **Description:** Identifies members who have been in the group the longest.
* **Purpose:** Recognizes and rewards loyal members.
* **Benefit:** Encourages member retention.
* **Sample Code (In Python):**

longest\_members = members\_df.sort\_values(by="join\_date").head(10)

### **17. Group Growth Rate**

* **Description:** Measures the rate at which groups are growing based on member joins over time.
* **Purpose:** Helps track community expansion trends.
* **Benefit:** Guides marketing and engagement efforts.
* **Sample Code (In Python):**

group\_growth = members\_df.groupby(["group\_id", members\_df["join\_date"].dt.date]).size().reset\_index(name="new\_members")

### **18. Most Discussed Topics (Based on Keywords in Messages)**

* **Description:** Extracts keywords from messages to determine trending topics.
* **Purpose:** Helps understand group interests.
* **Benefit:** Enables targeted discussions and content planning.
* **Sample Code (In Python):**

from collections import Counter

words = " ".join(messages\_df["text"].dropna()).lower().split()

word\_counts = Counter(words)

print(word\_counts.most\_common(10))

### **19. Message Response Time Analysis**

* **Description:** Calculates the average time taken for a message to receive its first reply.
* **Purpose:** Helps measure responsiveness and engagement speed.
* **Benefit:** Helps admins improve community interaction.
* **Sample Code (In Python):**

messages\_df["reply\_time"] = messages\_df.groupby("message\_id")["timestamp"].diff()

avg\_reply\_time = messages\_df["reply\_time"].mean()

print("Average reply time:", avg\_reply\_time)

### **20. Pinned Message Trends**

* **Description:** Analyzes pinned messages to understand the most important topics.
* **Purpose:** Tracks the types of messages that get pinned the most.
* **Benefit:** Helps admins optimize message pinning for better communication.
* **Sample Code (In Python):**

pinned\_trends = groups\_df["pinned\_messages"].value\_counts() print(pinned\_trends)

## **Technical Round 2 – Admin Dashboard Design**

In this I have stated 30 analytics that would be useful for an Admin Dashboard for managing Telegram groups: -

## Group Growth & Membership

1. **New Members Joined** – Tracks the number of new members joining daily, weekly, or monthly.
2. **Members Left** – Shows how many users left the group over a given period.
3. **Total Members** – Displays the total number of members in the group.
4. **Membership Growth Rate** – Percentage change in membership over time.
5. **New vs. Returning Members** – Compares new users to those who rejoin the group after leaving.

## Engagement & Activity

1. **Active vs. Inactive Members** – Percentage of members actively participating vs. inactive ones.
2. **Total Messages Sent** – The total number of messages posted in the group over a given period.
3. **Top Contributors** – Lists users who send the most messages.
4. **Peak Activity Time** – Identifies when the group is most active (by hour or day).
5. **Message Response Time** – Measures how quickly users respond to messages.

## Moderation & Security

1. **Deleted Messages** – Number of messages deleted by admins or Telegram.
2. **Reported Messages** – Messages flagged as spam or inappropriate.
3. **Banned Users** – List of users banned from the group.
4. **Admin Actions Summary** – Log of admin actions like bans, deletions, and pinned messages.
5. **Spam Detection** – Tracks the number of potential spam messages.

## User Behavior Analysis

1. **Most Mentioned Users** – Users who are tagged or mentioned the most in messages.
2. **Poll Engagement** – Tracks participation in group polls.
3. **Most Used Keywords/Topics** – Identifies trending discussion topics.
4. **Link Shares** – Number of external links shared in the group.
5. **Most Reacted Messages** – Messages that received the most reactions (emojis, likes).

## Multimedia & Content Analysis

1. **Image/Video Shares** – Tracks how many images or videos are shared.
2. **Document Shares** – Number of files (PDFs, Word docs, etc.) shared in the group.
3. **Pinned Messages Performance** – Tracks engagement (clicks, views) on pinned messages.
4. **Hashtag Usage** – Monitors hashtags used within the group.
5. **Top Shared Links** – Shows the most frequently shared links.

## Group Performance & Retention

1. **Daily/Weekly Engagement Score** – A calculated metric based on user activity.
2. **Retention Rate** – Percentage of users staying active over time.
3. **Event Participation** – Tracks how many members join scheduled events.
4. **Referral Growth** – Number of new members who joined via invites from existing users.
5. **Admin Response Rate** – Measures how quickly admins respond to user queries.

Thank You!