1. Objective

The goal of this assignment was to transform two append-only operational tables (`Messages` and `Statuses`) into a single analytics-ready dataset and generate engagement and delivery performance insights.

The final deliverable is a fact table with one row per message.

2. Raw Data Description

Messages Table

Contains message-level metadata:

- `uuid` – Unique identifier per message
- `direction` – inbound / outbound
- `masked_from_addr`
- `masked_addressees`
- `content`
- `last_status`
- `inserted_at`, `updated_at`

This table is append-only, meaning updates generate new rows.

Statuses Table

Contains message lifecycle events:

- `message_uuid`
- `status` (sent, delivered, read, failed)
- `timestamp`

Each message may have multiple status records.

3. Data Transformation Steps

Step 1 – Deduplication

Since the Messages table is append-only, multiple versions of the same message may exist.

We retained the latest version using:

- `MAX(updated_at)` per `uuid`

This ensured one clean row per message.

Step 2 – Status Aggregation

The Statuses table contains multiple events per message.

We transformed it using conditional aggregation:

- `sent_ts`
- `delivered_ts`
- `read_ts`
- `failed_ts`

Using SQL CASE logic and grouping by `message_uuid`. This converted multiple status rows into one structured lifecycle row.

Step 3 – Fact Table Creation

We joined deduplicated messages with aggregated statuses using LEFT JOIN.

Why LEFT JOIN? To preserve all message records even if some lifecycle events were missing.

The resulting table (`message_fact`) contains:

- One row per message
- Direction
- User identifier
- Lifecycle timestamps
- Message metadata

4. Data Validation

The following validation checks were performed:

4.1 Duplicate Check

Verified no duplicate `message_uuid` values exist in the final fact table.

Result: No duplicates found.

4.2 Logical Timestamp Validation

Checked for cases where:

- `read_ts < sent_ts`
- `delivered_ts < sent_ts`

Result:
5 minor millisecond-level inconsistencies identified.
These likely result from asynchronous event logging and timestamp precision.

No major logical violations observed.

4.3 Missing Lifecycle Records

Identified that 3.7% of outbound messages do not have corresponding status records in the Statuses table.This suggests partial lifecycle logging or incomplete export of status data.All records were retained to preserve data integrity.

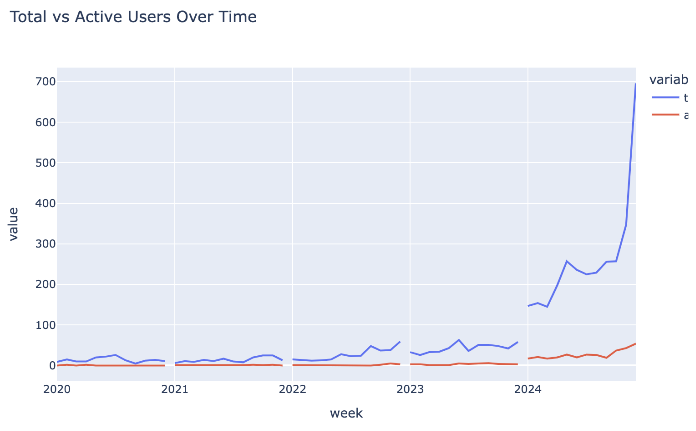
5. Business Metrics & Insights

5.1 Total vs Active Users

Tracked weekly:

- Total users (sent or received messages)
- Active users (sent inbound messages)

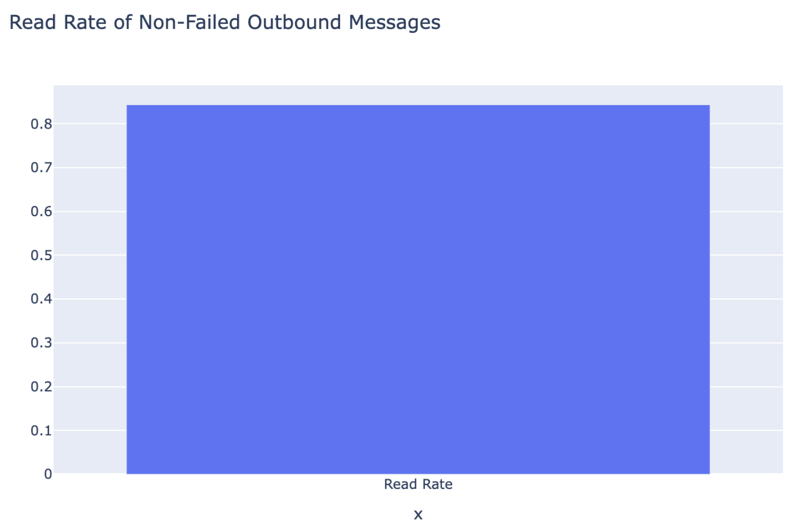
Observation:
Significant growth in total users in 2024. Active user growth is slower, suggesting outbound expansion is driving scale



5.2 Read Rate

Read Rate (non-failed outbound messages): **84.3%**

Indicates strong engagement and message effectiveness.

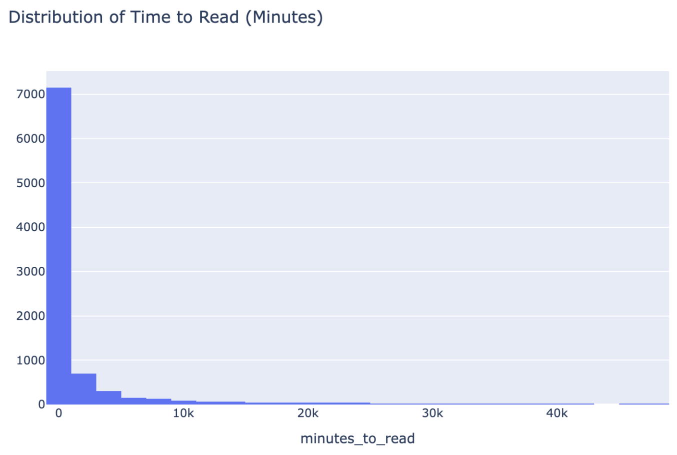


5.3 Time to Read

Distribution is heavily right-skewed:

- Majority of messages are read within minutes
- Long-tail behavior exists (some read days later)

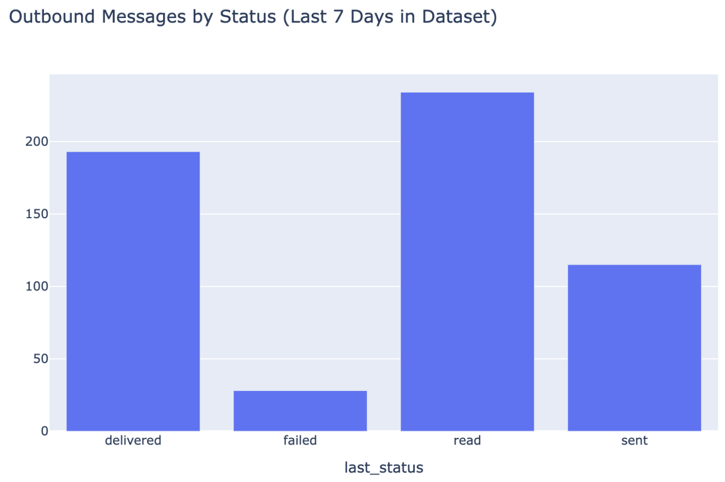
Median response time indicates strong near-real-time engagement.



5.4 Recent Delivery Performance

In the last 7 days of dataset:

- Majority of outbound messages were read
- Low failure count
- Delivery pipeline appears stable



6. Conclusion

The transformation pipeline successfully:

- Handles append-only data
- Aggregates event-level lifecycle information

- Produces an analytics-ready fact table
- Performs validation checks
- Generates business-relevant insights

The solution follows practical data engineering design principles and supports scalable analytics use cases.