MOS CIA 1 Assignment

Data Analysis Using Shell Scripting and Linux Commands

Name: Chrislo Tomy

Course: TY BSc-IT

UID: 2305072

Roll No: 57

1. Introduction

1.1 Overview of the Data Source and Reason for Selection

The dataset for this assignment comes from my personal Gmail archive, downloaded via **Google Takeout,** a service provided by Google that allows users to export data associated with their account.

- Source: Google Takeout → Gmail
- Data format: .mbox (Mailbox file format)
- Purpose of .mbox: Stores a collection of email messages in a single text file, often used for data backup and migration.
- File size: **~415 MiB**
- Total lines: **7,425,502**
- Total number of emails: **~7,230**
- Data range: From account creation until **August 6, 2025, 02:14 AM**
- Content types: Includes all folders (Inbox, Sent, Spam, Promotions, etc.)

I chose this dataset because it contains a large and diverse collection of emails accumulated over several years, making it highly suitable for identifying patterns, trends, and anomalies. The analysis will also be personally beneficial, as it will help me:

- Distinguish between spam and useful emails
- Identify the senders and topics that dominate my inbox
- Understand how my email usage has evolved over time

1.2 Nature and Structure of the Data

The .mbox format is semi-structured. Each email message starts with a special separator line and follows a pattern of headers and message content:

- Emails are separated by lines that begin with:

```
From sender@example.com Sat Aug 03 10:15:43 2025
```

Following this line are headers such as:

- Subject:
- Date:
- From:
- To:
- Message-ID:

Then comes the email body, which may include:

- Plain text
- HTML content
- Base64-encoded attachments

Sample (Raw View):

```
From noreply@xyz.com Wed Jun 26 16:21:35 2025
Date: Wed, 26 Jun 2025 16:21:35 +0530
From: XYZ Support <noreply@xyz.com>
Subject: Your support ticket has been updated
To: chrislo@gmail.com

Hello Chrislo,

Your ticket has been updated. Please check the portal.
```

1.3 Initial Observations

- Pattern for new emails is consistent (From delimiter).
- Key fields like From:, Subject:, and Date: are mostly present.
- Some headers (like Reply-To: or Cc:) are not always available.
- Spam and promotional content might affect the accuracy of some analysis.
- Emails may span multiple lines, and bodies might include noise (e.g., HTML, quoted replies, signatures, etc..).

1.4 Assumptions:

- Only top-level headers will be analyzed.
- Email body content may be ignored or minimally analyzed (e.g., for word count).
- Routing headers (Received) are skipped for simplicity.
- Some entries may be malformed and will be filtered during preprocessing.

2. Data cleaning & Preprocessing

I've created two shell scripts to process my .mbox email archive exported from Google Takeout and transform it into a clean, structured dataset suitable for further analysis and visualization.

2.1 Data Cleaning Script (data_clean.sh (github link))

This script is responsible for preprocessing and cleaning the raw .mbox file.

Purpose:

- Extract only the essential email header fields from each message:
  - FromName: Sender's display name.
  - FromEmail: Sender's email address.
  - ToEmail : Recipient's email address.
  - Date : The sent date, normalized to a consistent YYYY-MM-DD HH:MM:SS format.
  - Subject: Email subject decoded from MIME encodings (e.g., Base64, Quoted-Printable).
  - Labels: Gmail-specific labels such as Inbox, Spam, Promotions, etc.
- Remove extraneous metadata such as ARC seals, DKIM signatures, SMTP trace headers, and other internal Gmail details that do not add value to the analysis.
- Correct common formatting issues by normalizing line endings (CRLF to LF), handling multi-line headers, and trimming whitespace.
- Decode MIME-encoded headers to human-readable UTF-8 text.
- Output a clean, tab-separated values (TSV) file (emails_cleaned.tsv) with exactly six columns as described, ensuring uniform structure and easy parsing.

How it works:

- Uses sed to normalize line endings.
- Parses the .mbox file using awk, splitting on mbox message boundaries (From lines).
- Extracts and combines multi-line headers properly, trimming and cleaning them.
- Implements date parsing logic to convert varied date formats into a standard timestamp.
- Decodes encoded subjects and sender names directly in the script to avoid downstream decoding complexity.
- Writes the clean output to a TSV file ready for the next processing stage.

Output:

emails_cleaned.tsv with columns: FromName, FromEmail, ToEmail, Date, Subject, Labels.

2.2 Data Categorization Script (data_processed.sh(github link))

This script reads the cleaned dataset and assigns a category to each email based on its sender and Gmail labels.

Purpose:

- Add a Category column to the dataset to facilitate high-level grouping and filtering.
- Categorize emails into meaningful groups such as:
  - Job Alerts : Emails from job platforms or containing job-related keywords in the sender address.
  - Transactional/Finance: Emails related to banking, payments, invoices, or purchases.
  - Spam: Emails flagged as spam by Gmail labels.
  - Promotions: Marketing, promotional, or sales-related emails.
  - Social: Notifications from social media platforms.
  - Updates: General notifications and alerts.
  - Useful: A default fallback category for emails that don’t fit into any other group.

How it works:

- Uses awk to process each email record.
- Converts relevant fields (sender email and labels) to lowercase for case-insensitive matching.
- Applies pattern matching rules on sender domains and label contents to determine the category.
- Prints the original fields along with the newly assigned category into a new TSV file.

Output:

emails_categorized.tsv with all original columns plus a new Category column.

3. Data Analysis

Some meaningful insights from the processed email data, extracted using only shell scripts and standard Linux command-line utilities (e.g., awk, sed, grep, cut, sort, uniq, wc, etc..), are as follows:

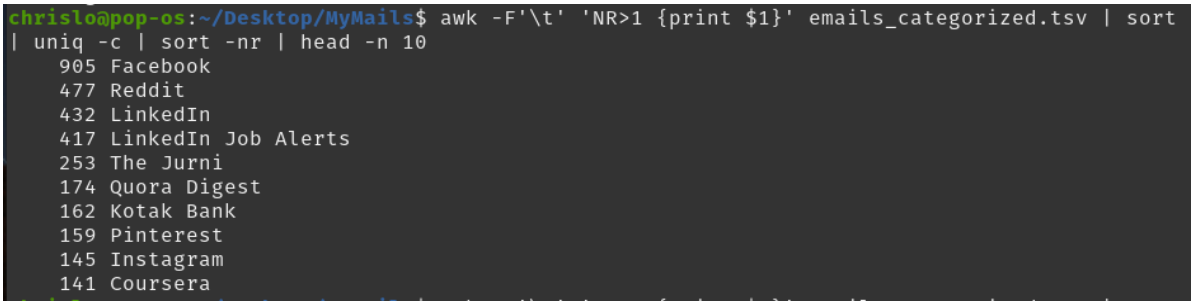
1.Overall Top 10 senders

Cmd: awk -F'\t' 'NR>1 {print $1}' emails_categorized.tsv | sort | uniq -c | sort -nr | head -n 10

Logic / Explanation:

- awk -F'\t' 'NR>1 {print $1}': Reads the file and prints only the sender's name (FromName column) from each row, skipping the header.
- sort: Groups all identical names together.
- uniq -c: Counts how many times each name appears consecutively.
- sort -nr: Sorts the results numerically (-n) and in reverse (-r), putting the highest counts at the top.
- head -n 10: Displays only the top 10 results.

Output:



2. .Top 3 Senders Within Each Category

Script: insight2.sh (github link)

Logic / Explanation:

- Stage 1 - awk (Count): The first awk command reads your data and counts every unique sender/category combination, just like before.
- Stage 2 - sort: This command takes the raw counts and sorts them, grouping all categories together and ranking the senders from highest to lowest within each group.
- Stage 3 - awk (Filter): This new final awk command reads the sorted list.
- while read ... loop: This loop now reads that sorted list line by line. It keeps a counter that resets to 1 every time it sees a new category. It only prints a line if its counter is 3 or less, effectively giving you the top 3 for each group.

Output:

output_to_insight2.tsv (github hyperlink)

3. Month-wise volume trends per category

Script: insight3.sh (github link)

Logic / Explanation:

- awk -F'\t' 'NR > 1 { ... }': Reads the file, skipping the header, and extracts the Category and Month (from the Date column).
- Inside awk, it counts occurrences of each unique Category-Month pair.
- END { print ... }: After reading all lines, prints the header and all counted pairs with their counts.
- sort -t' ' -k1,1 -k2,2: Sorts the output first by Category, then by Month alphabetically.

Output:

output_of_insight3.tsv (github hyperlink)

4. Number of emails per category (Spam, Promotions, Useful, etc.).

Cmd: awk -F'\t' 'NR>1 {print $7}' emails_categorized.tsv | sort | uniq -c | sort -nr

Logic / Explanation:

- awk -F'\t' 'NR>1 {print $7}': Reads the emails_categorized.tsv file and prints only the Category column (the 7th column) for every data row, skipping the header.
- sort: Takes the long list of categories and groups all identical entries together (e.g., all "Promotions" lines become adjacent).

- uniq -c: Counts how many times each category appears in the sorted list and prepends the count to the line.
- sort -nr: Sorts the final results numerically (-n) and in reverse (-r) to rank the categories from the highest count to the lowest.

Output:

```
chrislo@pop-os:~/Desktop/MyMails$ awk -F'\t' 'NR>1 {print $7}' emails_categorized.tsv | sort
| uniq -c | sort -nr
   2454 Updates
   1604 Promotions
   1053 Social
   1037 Job Alerts
    699 Transactional/Finance
    353 Useful
     26 Spam
```

5. Top 5 Most common keywords or phrases in subject lines per category.

Script: insight5.sh (github link)

Logic / Explanation:

a) Keyword Extraction (awk): The first awk script now does much more than count. For each subject line, it:
   - Converts the text to lowercase.
   - Removes all punctuation.
   - Splits the subject into individual words.
   - Ignores common "stop words" (like "the", "for", "is", etc.) and very short words to focus on the meaningful terms.
   - Finally, it counts the occurrences of each keyword within each category.

b) Ranking (sort): This stage takes the keyword counts and ranks them, placing the most frequent keywords at the top of each category group.

c) Filtering (awk): The final awk script reads the ranked list and, just like before, it smartly picks out only the top 5 entries for each category, giving you the output.

Ouptut:

output_of_insight5.tsv (github hyperlink)

6. Identify days with sudden email surges by category.
Script: insight6.sh (github link)
Logic / Explaination:

a) Get Daily Counts:
   - The first awk command reads emails_categorized.tsv, skips the header, and extracts the category ($7) and date ($4, trimmed to YYYY-MM-DD).
   - It counts how many emails appear for each (category, date) pair.

b) Sort Data:
   - The sort command arranges the counts by category first, then by date.
   - This ensures all days for the same category are in chronological order for proper spike detection

c) Detect High Volume Days & Spikes:
   - The second awk processes this sorted data.
   - For each category:
   - If `TodayCount >` EMAIL_COUNT_THRESHOLD (default: 5), it marks the row as "HighVolume".
   - If `TodayCount > SPIKE_MULTIPLIER × PrevCount` and `PrevCount ≥ MINIMUM_THRESHOLD` (defaults: 1.1 multiplier, 3 minimum), it marks the row as "Spike".
   - A day can be marked as both "HighVolume" and "Spike".

Output:

output_of_insight6.tsv (github hyperlink)

7. Day-wise volume of emails marked as Spam.
Script: insight7.sh (github link)
Logic / Explanation:

- Filter & Extract (awk): The first awk command reads your data and acts as a filter. It only pays attention to rows where the category is "Spam". For each of those rows, it prints out the month in which the email was received.
- Group & Count (sort | uniq -c): The list of days is then passed to sort, which groups all identical daystogether. uniq -c then counts how many emails occurred in each day.
- Output Format (awk): The final awk command takes these counts and reformats them into a clean, two-column output with a header.

Output:

output_of_insight7.tsv (github hyperlink)

8. Patterns in the subject header that correlate with spam.
Script: insight8.sh (github link)

Logic / Explanation:

- Keyword Counting: The first awk script creates a master list of every meaningful keyword and how many times it appeared in each category (e.g., "urgent" appeared 50 times in "Spam" and 2 times in "Useful").
- Scoring: The second awk script reads this master list. For every word, it calculates a Spam Score by comparing how often it appears in "Spam" emails versus "Useful" emails. A high score means the word is a strong indicator of spam.
- Ranking: The final sort and head commands rank the keywords by their Spam Score and display the top 20, giving you a clear list of the most common spam-related terms.

Output:

output_of_insight8.tsv (github hyperlink)

9.1. Peak email receiving hours of a day
Script: insight9.1.sh (github link)
Logic / Explanation:
The script extracts the hour (e.g., "14" for 2 PM) from the `Date` column for every email, then groups, counts, and sorts them to show the most active hours at the top.
Output:

output_of_insight9.1.tsv (github hyperlink)

9.2. Peak email receiving days of a week

Script: insight9.2.sh (github link)

Logic / Explaination:

For every email, the script uses the system's `date` command to find the corresponding day of the week (Mon, Tue, etc.). It then groups, counts, and sorts these days to rank them from busiest to quietest.

Output:

output_of_insight9.2.tsv (github hyperlink)

10. A Simple Sentiment analysis of subject lines (+ve ,-ve & neutral words)

Script: ~~insight10~~.sh (github link)

Logic / Explaination:

- Keyword Lists: The script defines two lists of words, POSITIVE_WORDS and NEGATIVE_WORDS. You can easily edit these lists to improve accuracy.
- Line-by-Line Processing: The script reads your emails_categorized.tsv file one email at a time.
- Counting: For each subject, it uses grep to count how many positive keywords and how many negative keywords it contains.
- Labeling: It compares the two counts. If there are more positive words, it labels the sentiment "Positive". If there are more negative words, it's "Negative". Otherwise, it's "Neutral".
- Output: It creates a new file, output_of_insight10.tsv, which is a copy of your input with the new Sentiment column added.

Output:

output_of_insight10.tsv (github hyperlink)

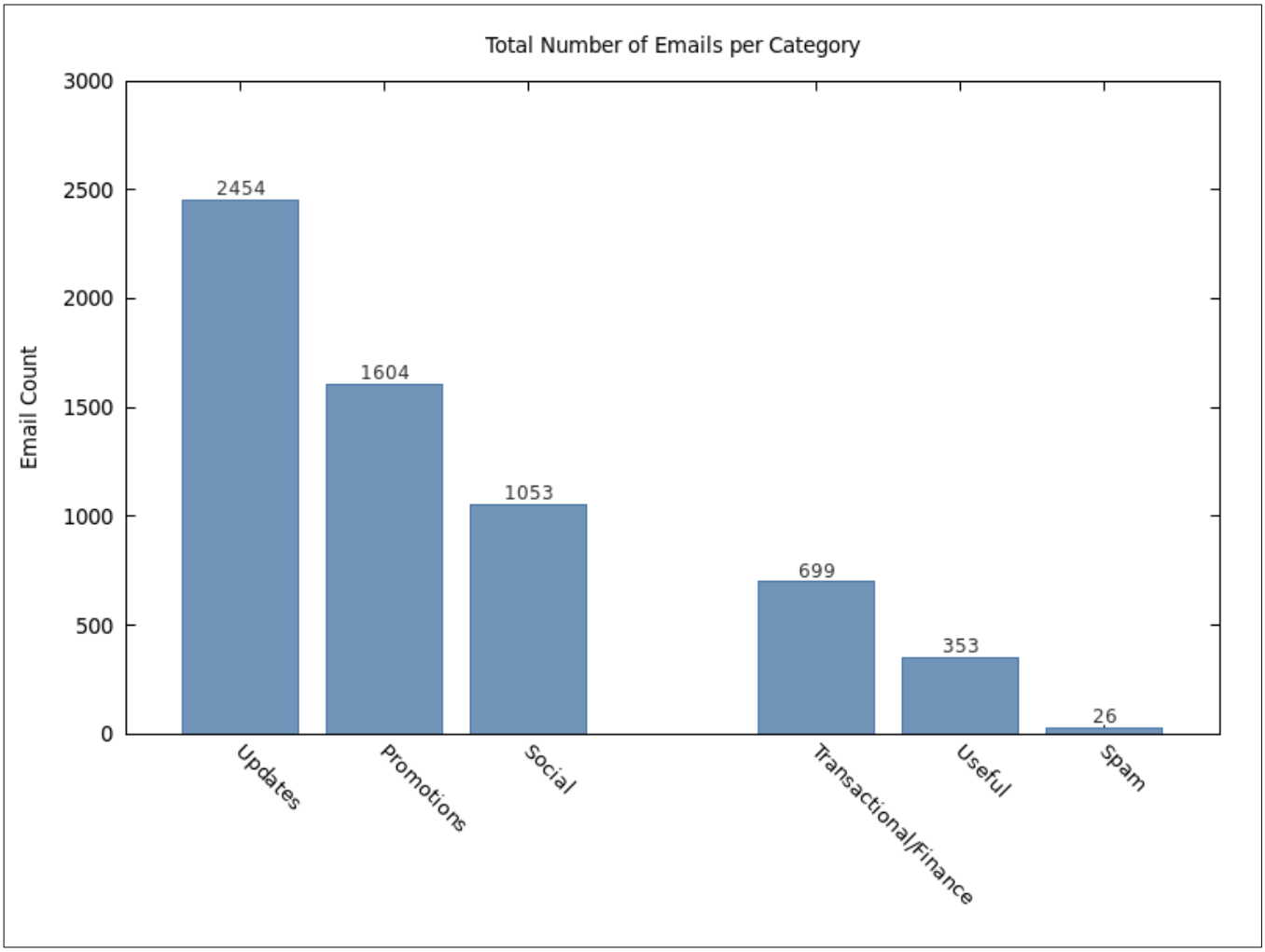
Visualization

Data visualization was performed using Gnuplot, a powerful open-source plotting utility that supports various chart types. It helped in identifying patterns and trends in the dataset that might not be evident from raw tabular data.

For this assignment, visualizations were created for four different questions/insights, using:

- Bar charts (2 questions)
- Line chart (1 question)
- Heatmap (1 question)

1. Bar Plot of Total number of emails per category (Spam, Promotions, Useful, etc



Plot script: plot_of_insight4gp (github link)
Explanation:

a) Output Setup:

- Exports the chart as a PNG (insight4_barchart.png) with an 800×600 resolution and a clean sans-serif font for readability.

b) Chart Style & Formatting:

- Histogram mode: set style data histograms produces the vertical bar layout.
- Color & fill: Bars are filled solid at 80% opacity with a consistent blue tone (#4e79a7).
- Gap and width: set style histogram clustered gap 1 and set boxwidth 0.8 balance spacing between bars.
- No legend: set key off hides the legend since the categories are already labeled on the X-axis.

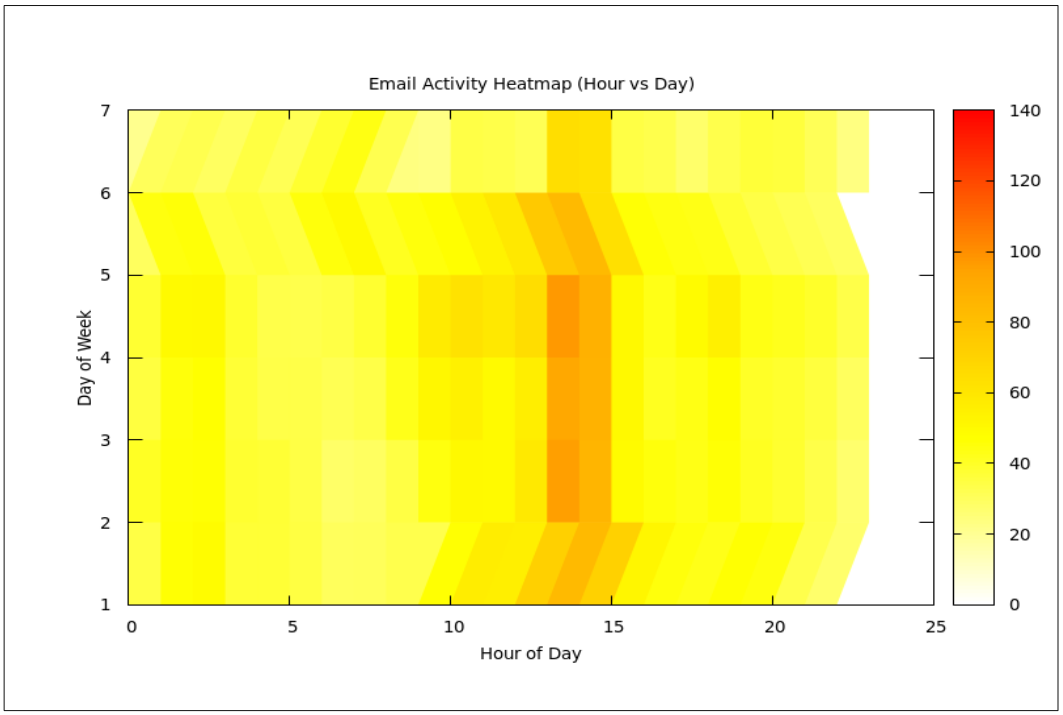
c) Labels & Readability Enhancements:

- Title & axis labels: Clearly indicate the purpose of the chart and the metric being measured.
- Rotated X-axis labels: Prevents text from overlapping for long category names.
- Offset padding: set offset adds extra space at the top so numeric labels above bars are not cut off.

d) Dual Plot Pass (Bars + Labels):

- First plot: Draws the colored bars with with boxes.
- Second plot: Reads the same file again to draw numeric labels above each bar, using:
  - Column 0 → X-position (bar index)
  - Column 2 → Label position (Y-value)
  - Labels are styled with a smaller font and darker text color for clarity

2. Heatmap of Email Actvity by Hours and Day



Plot script: [plot_of_insight9.gp (github link)]
Explanation:

a) Output Setup : The script outputs a PNG image (`insight9_heatmap.png`) at 800×600 pixels with clear font settings for readability.

b) Axes and Labels:

- X-axis: Hour of the day (0–23).
- Y-axis: Day of the week (numeric or mapped to names depending on the data).
- Z-axis (color scale): Represents email counts.
- Labels for each axis and a descriptive title are set for clarity.

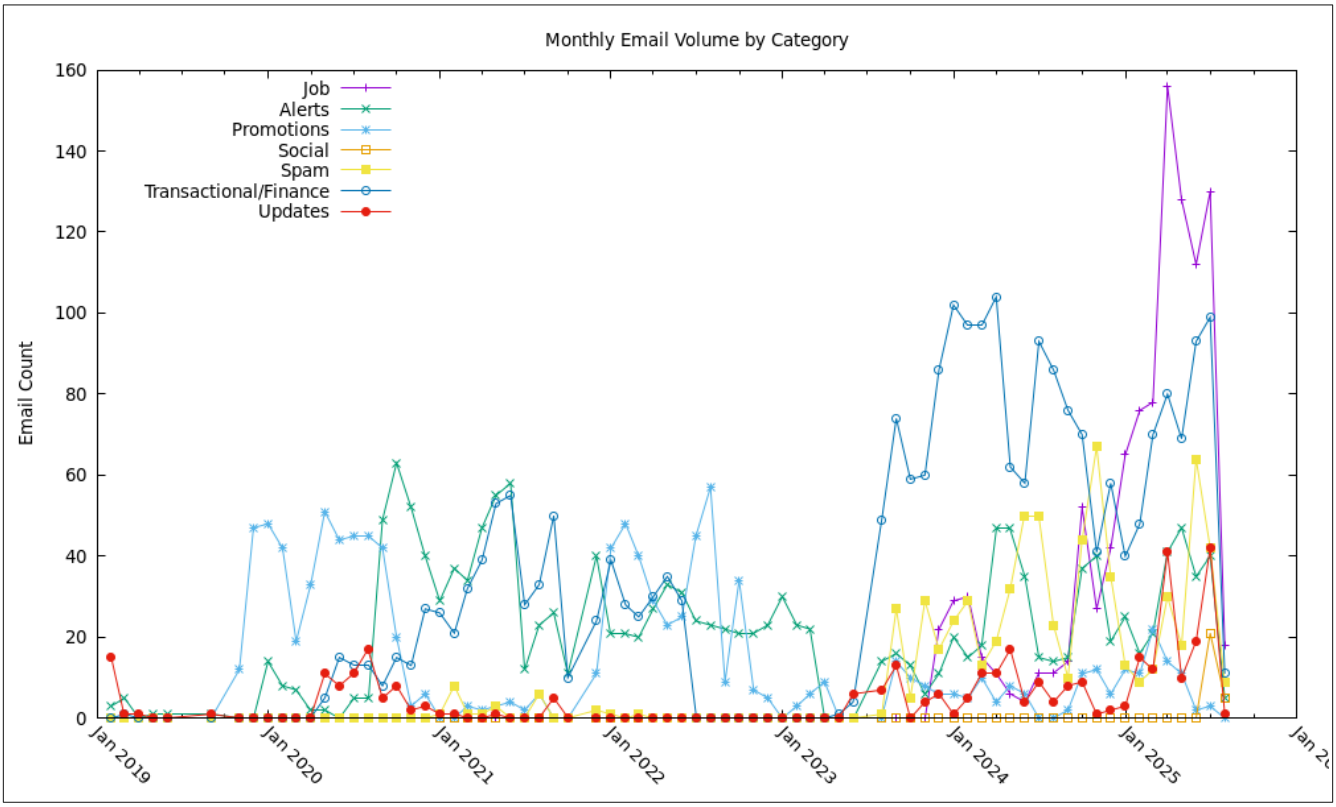
c) Heatmap Rendering:

- `set view map` flattens the 3D plot into a 2D top-down view.
- `set pm3d at b` uses a color gradient based on the Z-values.
- The custom color palette (`white → yellow → orange → red`) makes higher values stand out.

d) Plotting the Data:

- `splot` reads the processed data file, using:
  - Column 1 → Hour (X-axis)
  - Column 2 → Day (Y-axis)
  - Column 3 → Email Count (Color intensity).
- The unset key removes the legend since the color scale itself conveys the data range.

3. Line Plot Month-wise volume trends per category



Plot script: plot_of_insight3.gp (github link)

Explanation:

a) Output Setup : The script sets the output to a PNG image (insight3_linechart.png) with a defined font and size for clarity.

b) Dynamic Column Detection ; The system() function with head and awk is used to automatically detect how many columns are present in the TSV file. This makes the script adaptable to datasets with varying numbers of categories.

c) X-Axis Time Handling:

- set xdata time and set timefmt "%Y-%m" instruct Gnuplot to treat the first column as dates in YYYY-MM format.

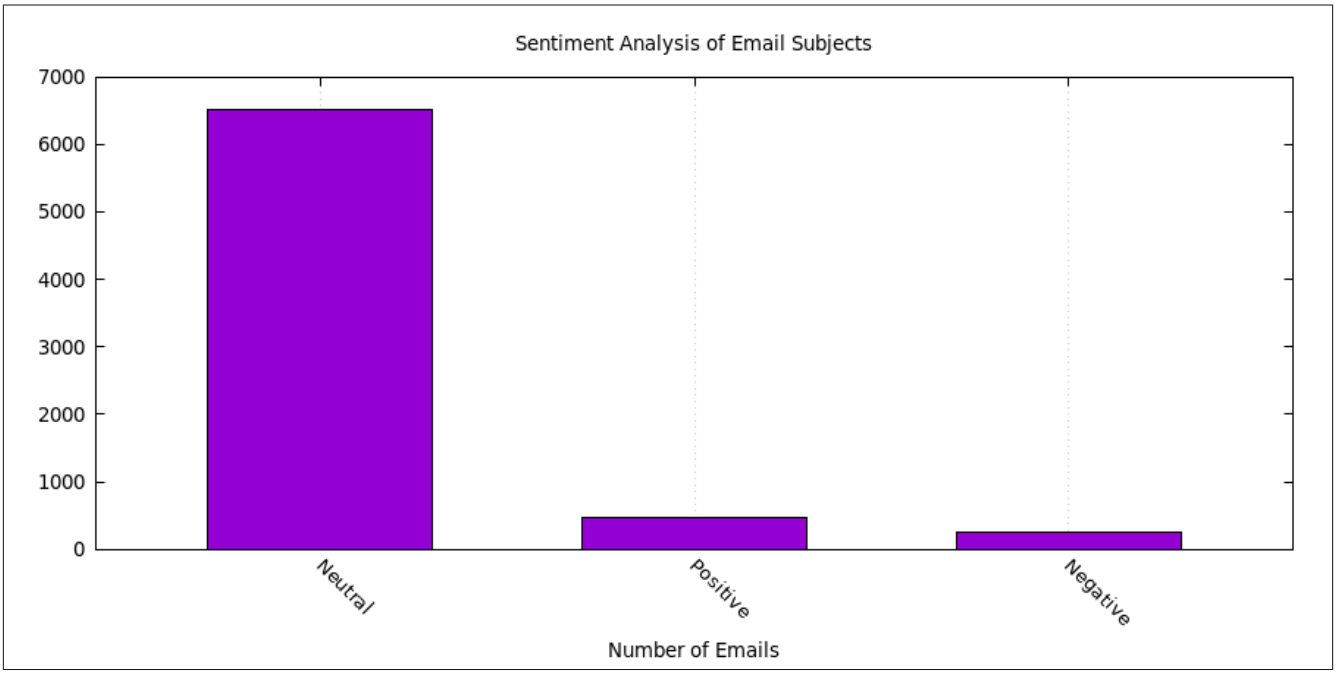
- set format x "%b %Y" formats the dates for human readability (e.g., "Jan 2023").
- set xtics rotate by -45 rotates labels to prevent overlap.

d) Multiple Series Plotting:

- The loop for [i=2:N] plots every column from the second onward (the first column is dates).
- with linespoints connects data points with lines and marks each point.
- Title columnhead uses the column header from the TSV file as the legend label.

e) Legend Positioning: The legend is placed in the top left to avoid overlapping with the plotted lines.

4. Bar Plot of the Sentiment analysis of the subject header



Plot script: plot_of_insight10.gp (github link)

Explanation:

a) Output Setup :

- Generates a PNG file (insight10_Barchart.png) at 900×450 pixels with a sans-serif font for clarity.

b) Chart Style & Layout:

- Title & axes: The X-axis shows sentiment categories, while the Y-axis measures the number of emails.
- No legend: Disabled via set key off since the categories are already labeled.

- Bar fill: set style fill solid 1.0 border -1 creates fully filled bars with a border color managed automatically.
- Bar width: Set to 0.6 to ensure bars are distinct but not too wide.
- Grid lines: set grid x adds vertical reference lines for easier comparison.

c) Color Assignment:

- No manual color mapping is used; Gnuplot automatically assigns default colors for each bar, ensuring a clean and minimal design.

d) Readability Features:

- Rotated X-axis labels: set xtics rotate by -45 prevents overlapping when sentiment category names are long.
- Simple presentation: No extra embellishments—keeps focus on the sentiment distribution.

e) Plot Command:

- Reads from plot_output_of_insight10.tsv where:
  - Column 1 → Sentiment category (X-axis).
  - Column 2 → Email count (Y-axis).
- Plots the bars directly with with boxes.

Note : If any of the scripts/outputs are missing from this report then its surely there on my gitub repo

Conclusion

This assignment helped me a lot to effectively use shell scripting and standard Linux command-line tools to process and analyze a large and complex email dataset. By extracting key metadata from a personal Gmail archive in `.mbox` format, the scripts were able to clean, categorize, and analyze over 7,000 emails spanning several years. The analysis revealed valuable insights including top senders to me, email volume trends by category and time, keyword distributions in subject lines, and sentiment trends, all achieved without relying on high-level programming languages or external libraries. Visualizations generated using Gnuplot further enhanced understanding by providing clear graphical representations of email patterns, peak activity periods, and sentiment distribution.

Possible Future Improvements

1. **Advanced Content Analysis Using High-Level Programming Languages:**
   While shell scripting is powerful for text processing, future enhancements could involve using languages such as Python or R that offer extensive libraries for natural language processing (NLP), machine learning, and data visualization. This would enable deeper analysis of email bodies, including semantic understanding, topic modeling, and more accurate sentiment classification.

2. **Integration of Specialized NLP Tools:**
   Employing NLP libraries such as NLTK, SpaCy, or transformer-based models (e.g., BERT) could improve keyword extraction, spam detection, and sentiment analysis accuracy by understanding context rather than just keyword frequency.

3. **Automated Attachment and Media Analysis:**
   Future work could explore extraction and analysis of attachments and embedded content (images, PDFs) within emails, which may reveal additional valuable insights not captured by header metadata alone.

4. **Scalability and Performance Optimization:**
   Processing larger datasets with shell scripts can be time-consuming and memory-intensive. Adopting parallel processing or moving to big data frameworks like Apache Spark could significantly reduce runtime and allow handling of even bigger email archives.

5. **Interactive Dashboards and Real-Time Monitoring:**
   Building an interactive web dashboard using tools such as Dash, Streamlit, can provide real-time analytics and more user-friendly exploration of email data trends and anomalies.

By adding these improvements, future analyses can provide clearer and deeper understanding of email patterns and how users interact, using modern tools together with basic shell scripting skills.