Automated Intelligent Data Analysis Pipeline

Project Documentation

1. Problem Statement

Data analysis is a critical but time-consuming process. When a data analyst receives a raw CSV file, they must manually:

- Understand the data structure and schema
- Clean and preprocess the data
- Explore distributions and relationships
- Detect patterns and anomalies
- Choose appropriate visualizations
- Generate charts and graphs
- Write a comprehensive summary report

This manual process typically takes 2–4 hours of skilled human work per dataset. For organizations handling multiple datasets daily, this represents a significant bottleneck in decision-making workflows.

2. Solution Overview

The Automated Intelligent Data Analysis Pipeline transforms a raw CSV file into a complete, professional HTML report in under 60 seconds with zero human intervention.

The solution chains two different AI tools via API:

1. Gemini API (Google) – Acts as the data analyst and report narrator
2. Groq API (Llama 3.3) – Generates executable Python analysis code

A single command (python pipeline.py dataset.csv) orchestrates all 6 stages, from data profiling to final report generation.

3. Tools Used

Tool	Provider	UX Type	Role in Pipeline	Cost
Gemini API	Google	Chat AI	Data Analyst	Free tier

(gemini-2.5-flash)			(Stage 1) + Report Narrator (Stage 4)	
Groq API (llama-3.3-70b-versatile)	Groq	Code Gen AI	Python Code Generator (Stage 2)	Free tier
Cerebras API (llama-3.3-70b)	Cerebras	Code Gen AI	Fallback for Stage 2	Free tier
OpenRouter (deepseek-chat-v3-0324)	OpenRouter	Chat AI	Fallback for Stages 1 & 4	Free tier
Python	–	CLI + Streamlit	Profiling, Execution, Compilation (Stages 0, 3, 5)	Free

4. Workflow Diagram

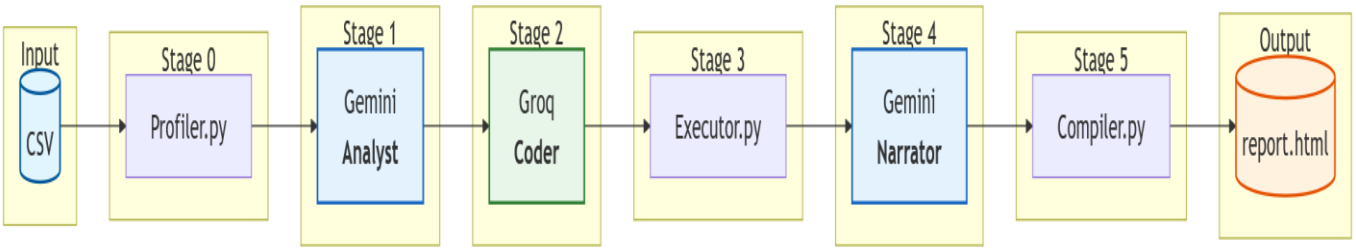

Figure 1: Pipeline Workflow Diagram

5. Step-by-Step Workflow Instructions

Stage 0: Data Profiling (Python – profiler.py)

- Reads the CSV file with pandas.
- For large datasets (over 2,000 rows), statistics are computed on a random sample of 2,000 rows; the full dataset is still used in Stage 3.

- Extracts schema: column names, data types, row count.
- Calculates statistics: mean, median, min, max, std, skewness, kurtosis, percentiles for numeric columns.
- Counts missing values per column and top value counts for categorical columns.
- Samples 3 representative rows.
- Output: data_profile (JSON).

Stage 1: AI Analysis Planning (Gemini API)

- Receives the data profile.
- Identifies what the dataset represents.
- Generates 4–5 analytical questions to answer.
- Specifies columns, analysis types, and chart types for each.
- Suggests data cleaning steps.
- Output: analysis_plan (JSON).

Stage 2: Code Generation (Groq API)

- Receives data profile and analysis plan.
- Generates a complete, runnable Python script.
- Script handles: data loading, cleaning, analysis, chart generation.
- Output: analysis_script.py (Python code string).

Stage 3: Auto-Execution (Python – executor.py)

- Clears old charts from previous runs to avoid stale results.
- Writes generated code to a file and executes via subprocess with 120-second timeout.
- Captures stdout (analysis results) and stderr (errors).
- Collects generated PNG charts.
- If the script crashes partway through but some charts were saved, the pipeline treats it as a partial success.
- Output: charts (PNG files) + raw_insights (JSON).

Stage 4: Narrative Generation (Gemini API)

- Receives raw analysis results and chart descriptions.
- Writes an executive summary (3–4 sentences).
- Generates key findings (bullet points).
- Provides actionable recommendations.
- Output: narrative (JSON).

Stage 5: Report Compilation (Python – compiler.py)

- Loads Jinja2 HTML template.
- Embeds charts as base64 data URIs.
- Injects narrative, findings, recommendations.
- Adds metadata (dataset name, timestamp, tools used).
- Output: report.html (self-contained HTML file).

6. Final Prompts Used at Each Stage

Prompt 1: The Analyst (Stage 1 – Gemini API)

```
You are a senior data analyst. I will give you a data profile (schema,
statistics, sample rows) of a CSV dataset.

Your job:
1. Identify what this dataset represents (domain, purpose)
2. List the 4-5 most interesting analytical questions this data can
answer
3. For each question, specify:
   - The exact columns to use
   - The type of analysis (correlation, distribution, comparison,
trend, anomaly detection)
   - The best chart type (bar, scatter, histogram, box, heatmap, line,
violin, pie)
   - A short description of what insight to look for
4. Identify data cleaning steps needed with SPECIFIC handling for null
values and edge cases:
   - For each column with null values, specify: "Drop rows with null in
[column_name]" OR "Fill null in [column_name] with
[value/mode/median/mean]"
   - For numeric columns: use mean, median, or a specific value
   - For categorical columns: use mode or "Unknown"
   - For columns with string patterns (like duration "2h 30m", "30m",
"5h" or stops "2 stops"), handle ALL edge cases
   - ALWAYS specify a robust parsing function that handles all formats
   - NEVER use operations that assume non-null values (like .split())
without first handling nulls and edge cases
   - Add a check: if converting strings to numeric, handle parsing
errors with try-except or pd.to_numeric with errors='coerce'

IMPORTANT: If any column has null values, you MUST include explicit
null-handling steps in cleaning_steps before any other operations.
```

```
Return your response as ONLY valid JSON with this structure:
{
  "dataset_description": "string",
  "cleaning_steps": ["step1", "step2"],
  "analyses": [
    {
      "question": "string",
      "columns": ["col1", "col2"],
      "analysis_type": "string",
      "chart_type": "string",
      "insight_hint": "string"
    }
  ]
}

Here is the data profile:
{profile_json}
```

Prompt 2: The Coder (Stage 2 – Groq API)

```
You are an expert Python data analyst. Generate a COMPLETE, RUNNABLE
Python script that performs the following analysis on a CSV file.

RULES:
- The CSV file path will be passed as the first command-line argument
(sys.argv[1])
- Use only: pandas, matplotlib, seaborn, sys, os, re
- Save each chart as a PNG file in an "output/charts/" directory
(create if needed)
- Name chart files using simple numbered names ONLY, like
"chart_1.png", "chart_2.png"
- Print a JSON object to stdout with the key results of each analysis
- Do NOT use plt.show() – only plt.savefig()
- Wrap EACH individual analysis in its own try/except block
- Close each figure after saving with plt.close()

CRITICAL DATA TYPE HANDLING – USE EXACT SYNTAX:
1. For null handling: df.loc[:, 'col'] = df['col'].fillna(value)
2. For converting categorical string columns to numeric:
   - Create mapping dict, apply mapping, pd.to_numeric, fillna,
astype(int)
3. For datetime conversion: df['col'] = pd.to_datetime(df['col'],
format='mixed', dayfirst=True)
```

```
4. For string operations like .split(), ALWAYS handle NaN first using
fillna
5. For Duration parsing: use a robust function handling "Xh Ym", "Ym",
"Xh", and empty values
6. Always verify column types after conversion using df.dtypes

DATA SCHEMA:
{data_profile}

ANALYSIS PLAN:
{analysis_plan}

Return ONLY the Python code, no markdown fences, no explanation.
```

Prompt 3: The Narrator (Stage 4 – Gemini API)

```
You are a senior data analyst writing a report for stakeholders.

I will give you the raw results of a data analysis (statistical
findings, chart descriptions, detected patterns).

Write:
1. An executive summary (3-4 sentences, the most important takeaway)
2. Key findings (bullet points, one per analysis performed)
3. Recommendations (2-3 actionable next steps based on the data)

Return your response as ONLY valid JSON:
{
  "executive_summary": "string",
  "key_findings": ["finding1", "finding2"],
  "recommendations": ["rec1", "rec2"]
}

Here are the analysis results:
{analysis_results}
```

7. Results

The pipeline successfully:

- Processes any CSV dataset in under 60 seconds
- Generates professional visualizations automatically
- Produces coherent, stakeholder-ready narratives
- Creates self-contained HTML reports (no external dependencies)

Time saved: 2–4 hours per dataset reduced to 45–60 seconds.