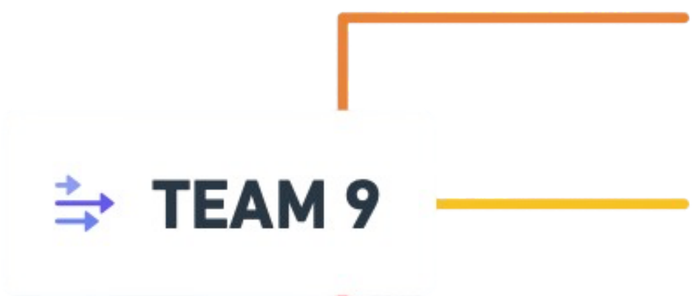
M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur, Tamilnadu.

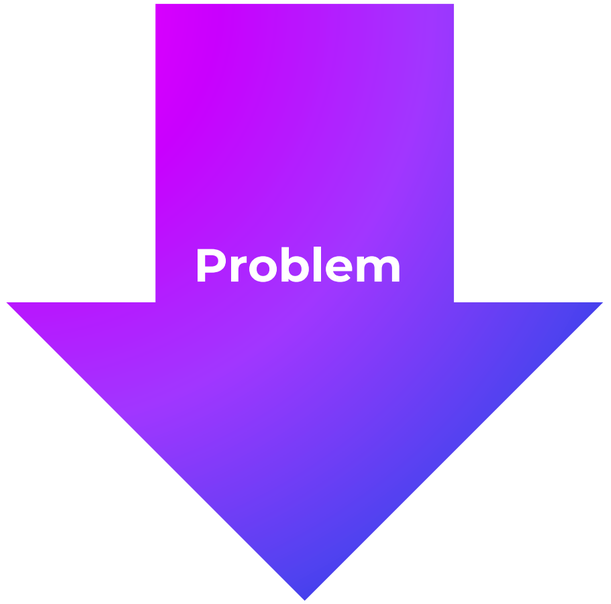
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Streamlined Data Processing Pipeline

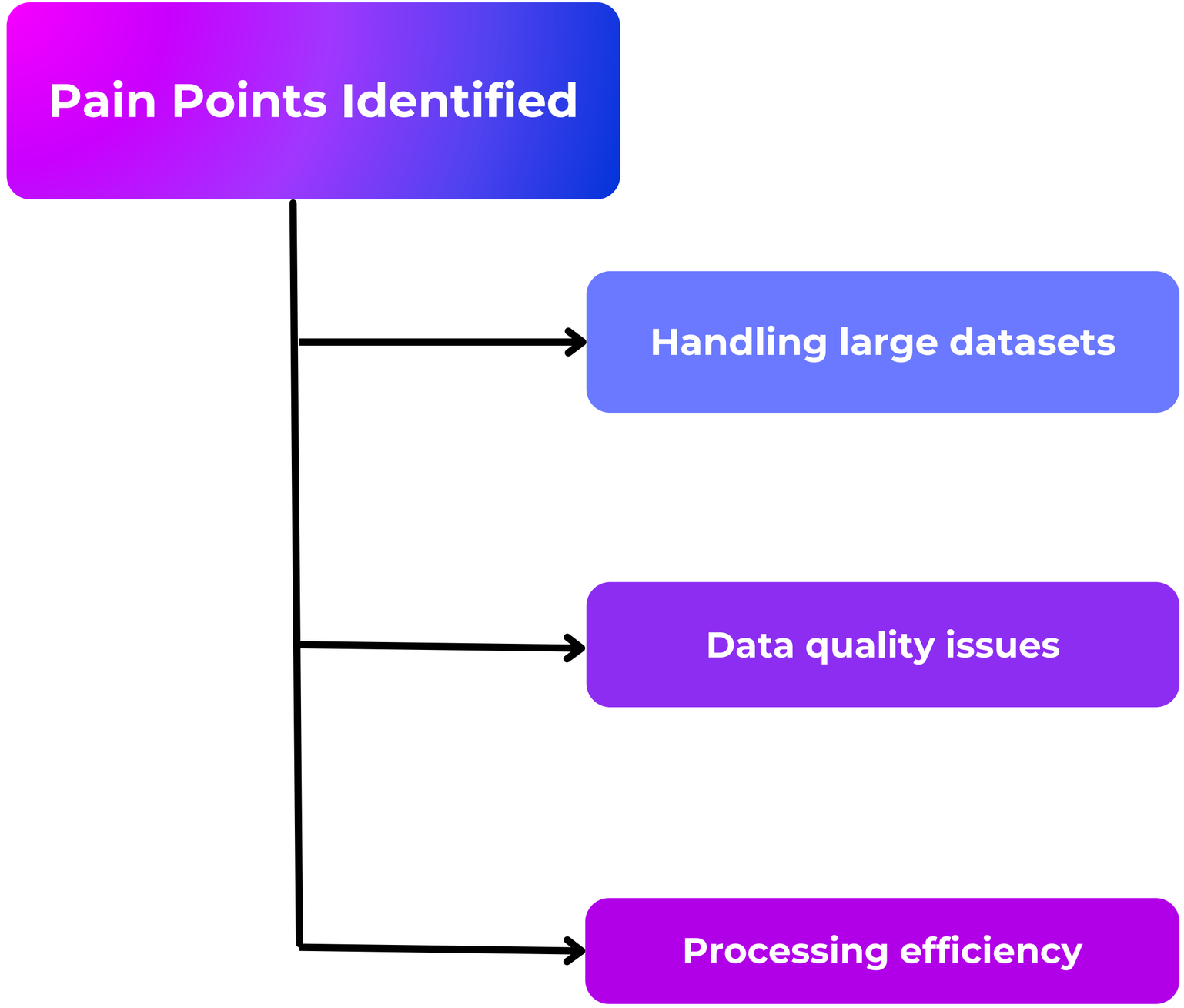


SrinithiT(927623BAD108)
Venkat Ragav N (927623BAD123)
Rithika G(927623BAD091

PandaPrep: Streamlined Data Processing Pipeline



The project aims to develop a scalable, efficient pipeline to process large CSV files with data quality issues


Pain Points Identified
Handling large datasets
Data quality issues
Processing efficiency

Significance of the Project

- **The project is significant because it addresses a common issue faced by data engineers and analysts—how to handle and process large datasets efficiently.**
- **By automating data preprocessing and making it scalable, the solution can save significant time and effort, enabling better data-driven decision-making.**

Original Objectives

01

Transform data for downstream analysis.

02

Handle data quality issues like missing values, duplicates, and formatting errors.

03

Develop a scalable pipeline for cleaning and preprocessing large CSV files.

Updated Objectives

01

Real-Time Processing : Explore the potential of adding real-time processing capabilities if the data comes from live systems

02

Output Formats: Enhance the pipeline to output cleaned data in different formats like JSON, SQL, and other structured formats besides just CSV.

03

Parallel Processing: Add support for parallel processing to handle large datasets faster.

04

Data Transformation: Enhance the pipeline to support more complex data transformations beyond just cleaning

Methodology & Approach

Methodologies:

Data Collection
&
Preprocessing

Data Sources

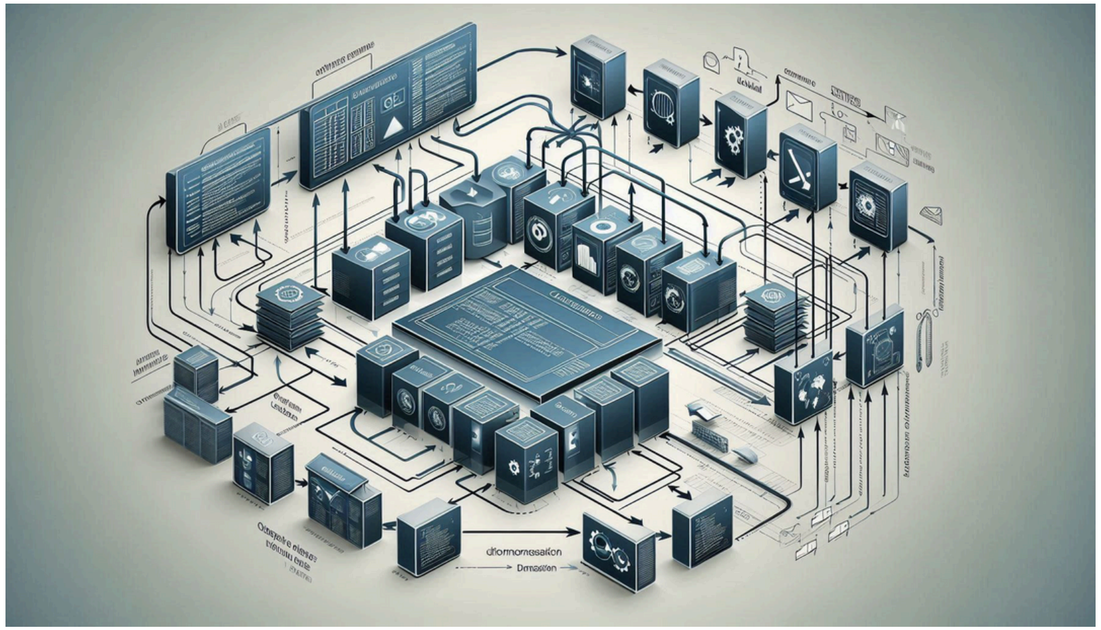
The data comes in the form of raw CSV files sourced from various internal databases or system logs, potentially containing millions of rows and multiple columns.

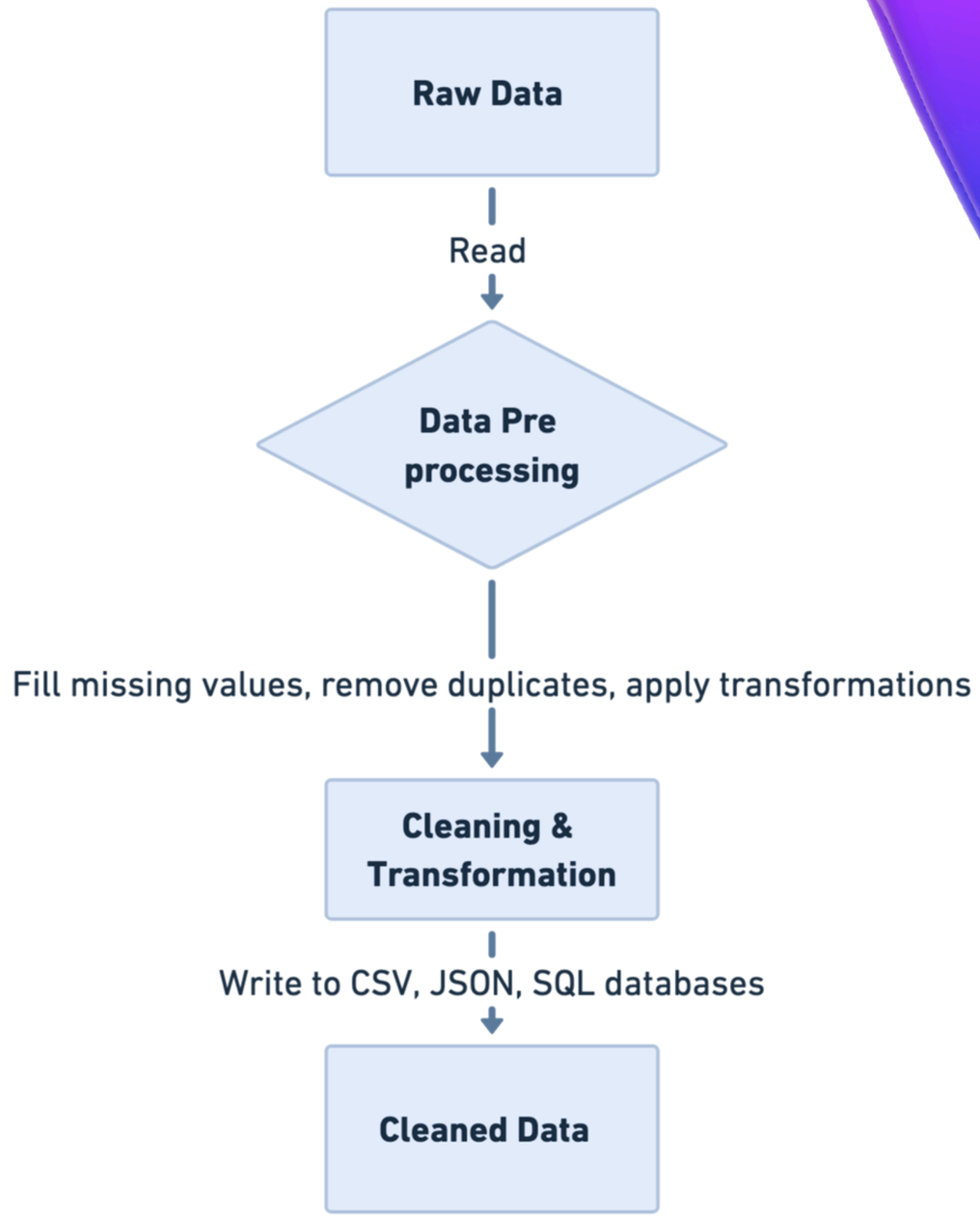
Data Cleaning Techniques

- Handling Missing Values
- Removing Duplicates
- Type Casting

System Architecture

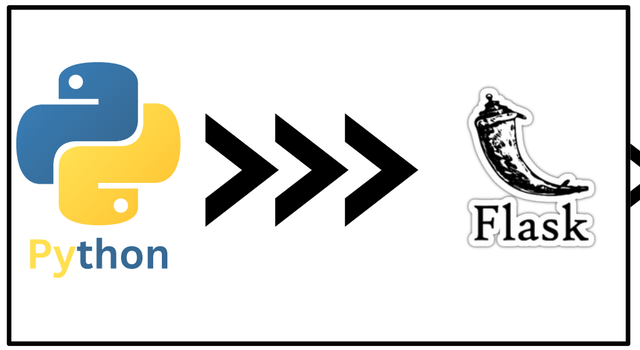
Flow Diagram





Technology Stack

Language:



Backend programming

Frontend programming

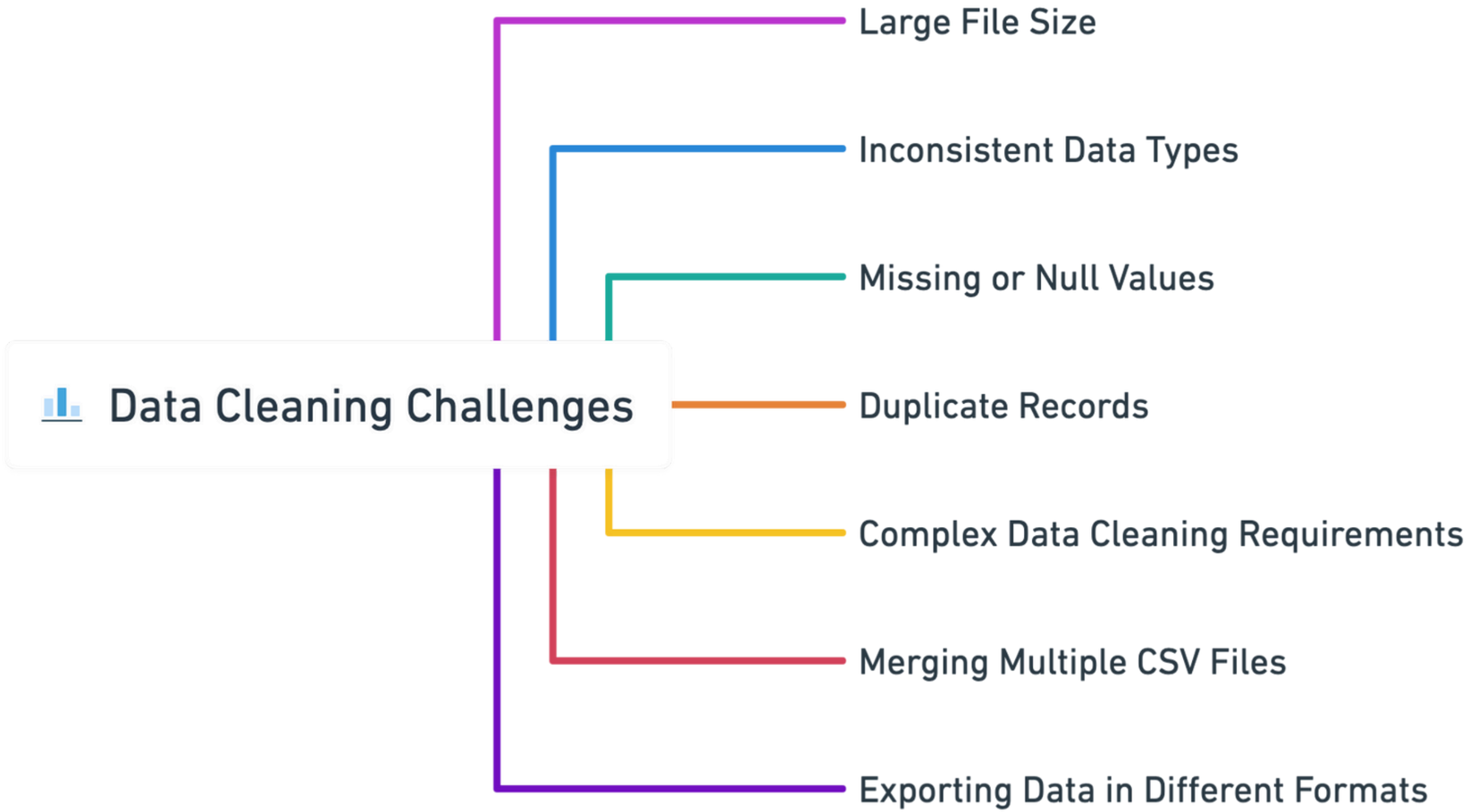
Tools:

Visual Studio Code

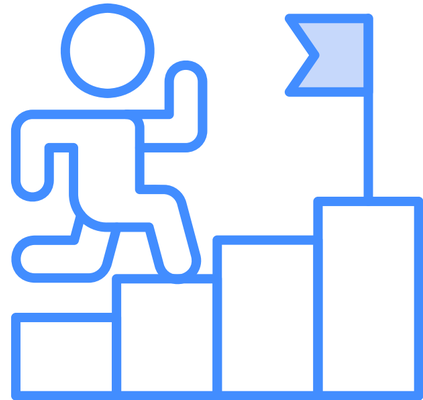
Google Sheets

Challenges & Solutions

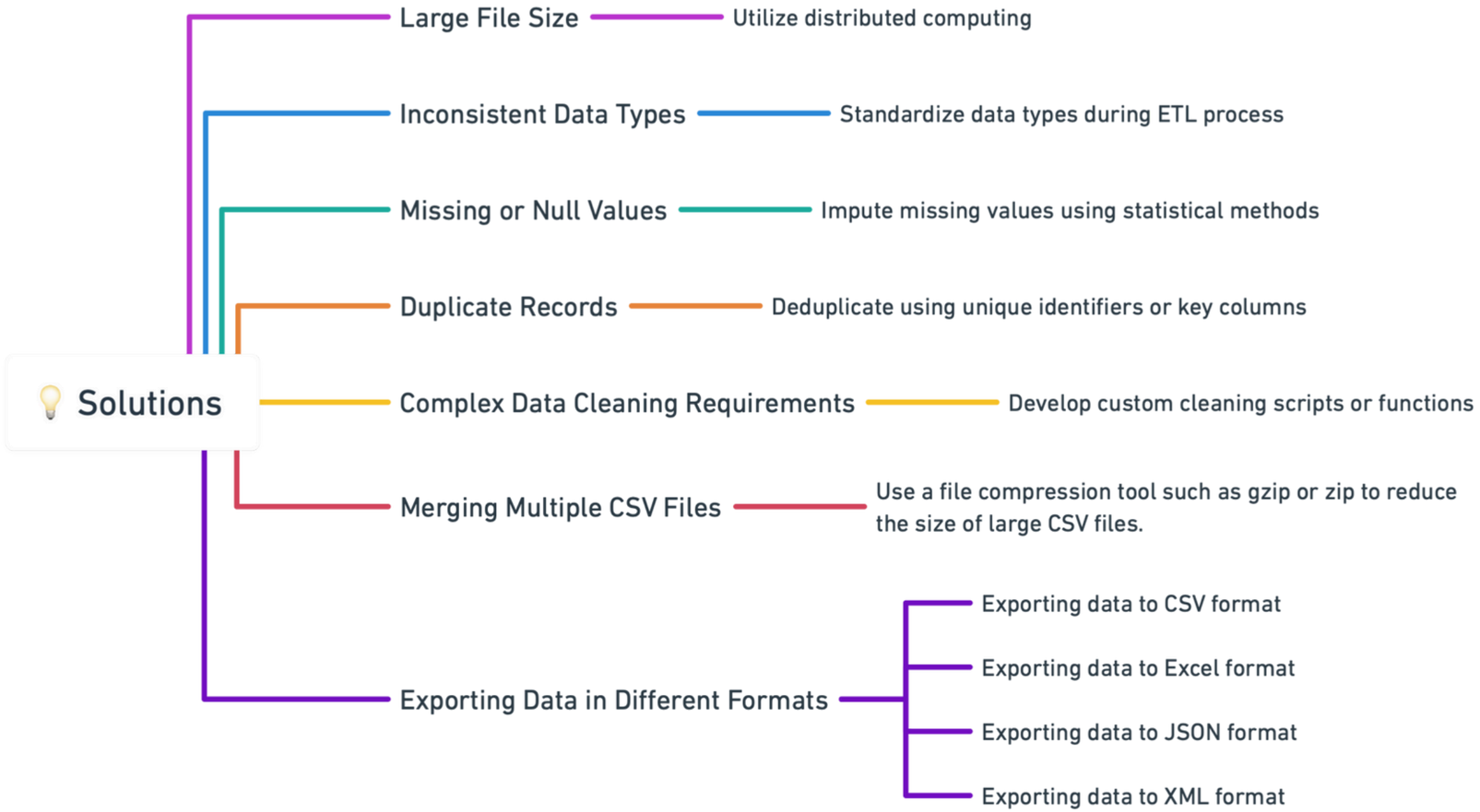
Challenges:



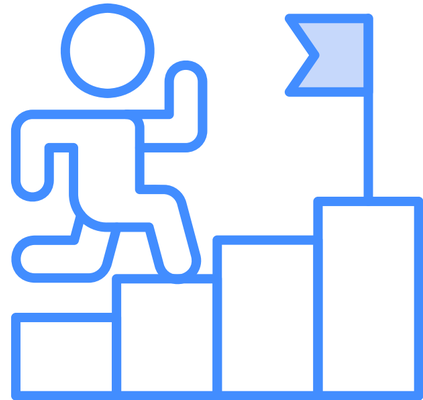
Google Sheets



Solutions:

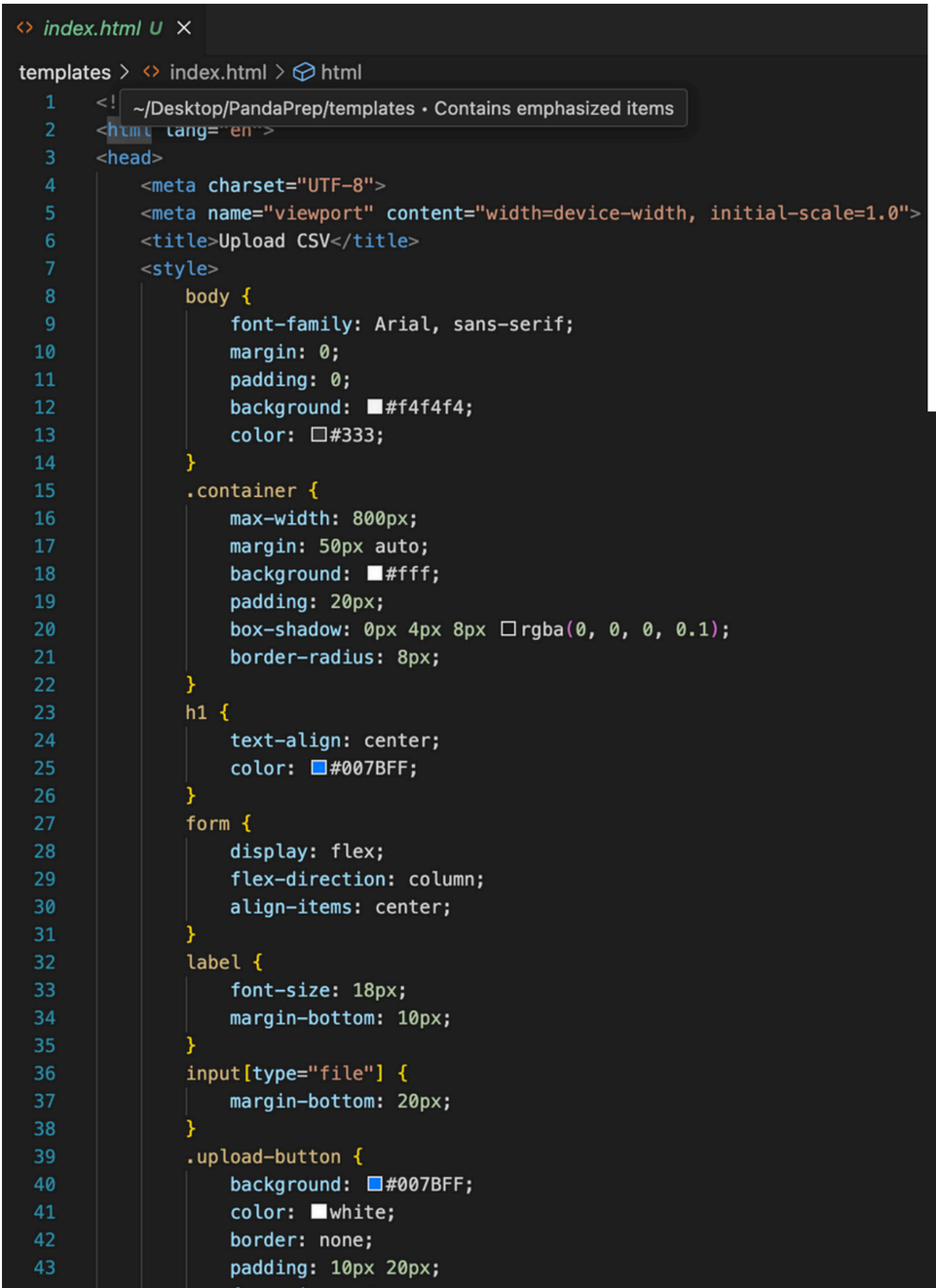


Google Sheets



Project Code 100%

index.html



Visual Studio Code

```
            font-size: 16px;
            border-radius: 4px;
            cursor: pointer;
        }
        .upload-button:hover {
            background: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Upload Your CSV File</h1>
        <form action="/clean" method="POST" enctype="multipart/form-data">
            <label for="file">Select CSV File:</label>
            <input type="file" id="file" name="file" accept=".csv" required>
            <button type="submit" class="upload-button">Upload and Clean</button>
        </form>
    </div>
</body>
</html>
```

result.html

Visual Studio Code

result.html U

templates > result.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
    <title>Cleaned Data</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background: #f4f4f4;
            color: #333;
        }
        .container {
            max-width: 1000px;
            margin: 30px auto;
            background: #fff;
            padding: 20px;
            box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
        }
        h1, h2 {
            text-align: center;
            color: #007BFF;
        }
        .table-container {
            margin: 20px auto;
            overflow-x: auto;
        }
        .styled-table {
            border-collapse: collapse;
            margin: 20px auto;
            font-size: 16px;
            width: 100%;
            border: 1px solid #ddd;
        }
        .styled-table th, .styled-table td {
            text-align: left;
            padding: 10px;
        }
        .styled-table th {
            background: #007BFF;
            color: white;
        }
        .styled-table tr:nth-child(even) {
            background: #f4f4f4;
        }
        .styled-table tr:hover {
            background: #ddd;
        }
        .chart-container {
            margin: 20px auto;
            text-align: center;
        }
        .button-group {
            text-align: center;
            margin-top: 20px;
        }
        .button-group a {
            display: inline-block;
            text-decoration: none;
            margin: 0 10px;
            background: #007BFF;
            color: white;
            padding: 10px 20px;
            border-radius: 4px;
            font-size: 16px;
        }
        .button-group a:hover {
            background: #0056b3;
        }
        .search-bar {
            text-align: center;
            margin-bottom: 20px;
        }
        .search-bar input {
            padding: 10px;
            width: 50%;
            font-size: 16px;
            border: 1px solid #ccc;
```

templates > result.html > ...

```
<html lang="en">
<head>
    <style>
        .search-bar input {
            border: 1px solid #ccc;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Cleaned Data Preview</h1>

        <h2>Preview of Cleaned Data</h2>
        <div class="search-bar">
            <input type="text" id="search" placeholder="Search in
        </div>
        <div class="table-container">
            {{ cleaned_preview|safe }}
        </div>

        <h2>Summary Statistics</h2>
        <div class="table-container">
            {{ stats_preview|safe }}
        </div>

        <h2>Data Visualization</h2>
        <div class="chart-container">
            {{ chart_html|safe }}
        </div>

        <div class="button-group">
            <a href="{{ download_link }}">Download Cleaned Data</a
            <a href="/">Back to Upload</a>
        </div>
    </div>
    <script>
        function filterTable() {
            let input = document.getElementById("search");
            let filter = input.value.toUpperCase();
```

Visual Studio Code

main.py

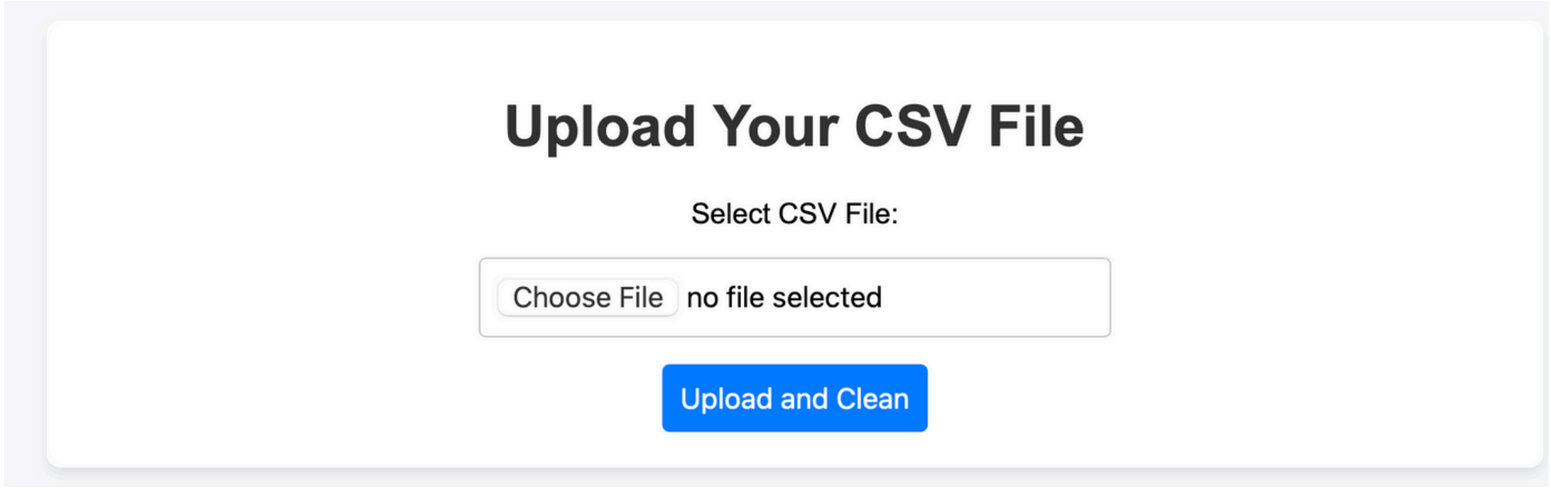
main.py > clean_data

```python
from flask import Flask, request, render_template, send_file
import pandas as pd
import io
import matplotlib
matplotlib.use('Agg')  # Use a non-GUI backend
import matplotlib.pyplot as plt
import base64
import os

app = Flask(__name__)

# Ensure the downloads directory exists
os.makedirs('downloads', exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/clean', methods=['POST'])
def clean_data():
    try:
        # Retrieve uploaded file
        file = request.files['file']
        if not file:
            return "No file uploaded", 400

        try:
            # Read the CSV file with UTF-8 encoding using the python engine
            df = pd.read_csv(file, on_bad_lines='skip', engine='python')
        except UnicodeDecodeError:
            # If UTF-8 fails, try with ISO-8859-1 encoding
            file.seek(0)  # Reset file pointer to the beginning
            df = pd.read_csv(file, encoding='ISO-8859-1', on_bad_lines='skip', engine='python')

        # Data Cleaning
        df.fillna(value='N/A', inplace=True)  # Fill missing values
        df.drop_duplicates(inplace=True)  # Remove duplicates

        # Generate cleaned data preview (first 10 rows)
        cleaned_preview = df.head(1000000).to_html(classes='styled-table', index=False)
```

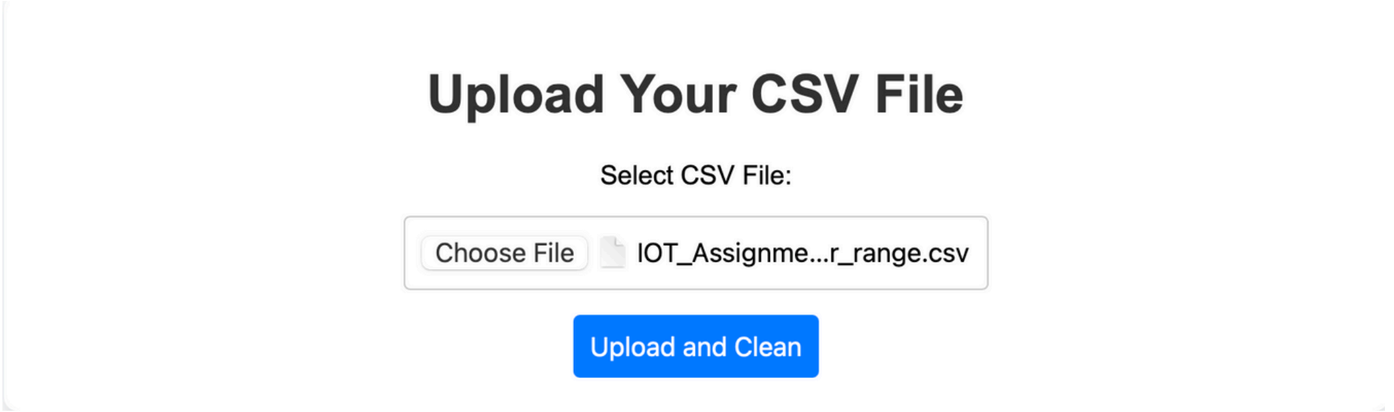
```python

        # Summary Statistics
        numeric_summary = df.describe().transpose()
        numeric_summary.reset_index(inplace=True)
        if numeric_summary.shape[1] == 8:
            numeric_summary.columns = ['Column', 'Count', 'Mean', 'Std', 'Min', '25%', '50%', '75%', 'Max']
        else:
            numeric_summary.columns = ['Column'] + [f'Stat_{i}' for i in range(1, numeric_summary.shape[1])]

        stats_preview = numeric_summary.to_html(classes='styled-table', index=False)

        # Data Visualization (example: column distribution)
        fig, ax = plt.subplots(figsize=(8, 6))
        if not df.select_dtypes(include=['number']).empty:
            df.select_dtypes(include=['number']).mean().plot(kind='bar', ax=ax, color='skyblue')
            ax.set_title('Mean of Numeric Columns', fontsize=14)
            ax.set_ylabel('Mean Value', fontsize=12)
            plt.tight_layout()

            # Save plot to a string buffer
            buf = io.BytesIO()
            plt.savefig(buf, format='png')
            buf.seek(0)
            chart_html = f'<img src="{base64.b64encode(buf.read()).decode()}" alt="Chart">'
            buf.close()

            # Save plot to a file in the downloads directory
            graph_path = os.path.join('downloads', 'mean_numeric_columns.png')
            plt.savefig(graph_path)
            plt.close(fig)
        else:
            chart_html = "<p>No numeric columns available for visualization.</p>"

        # Save cleaned data for download
        cleaned_data_path = os.path.join('downloads', 'cleaned_data.csv')
        df.to_csv(cleaned_data_path, index=False)

        # Provide a downloadable link
        download_link = "/download"
        download_graph_link = "/download-graph"
```

OUTPUT

INDEX PAGE



AFTER UPLOADING THE FILE



Cleaned Data Preview

Preview of Cleaned Data

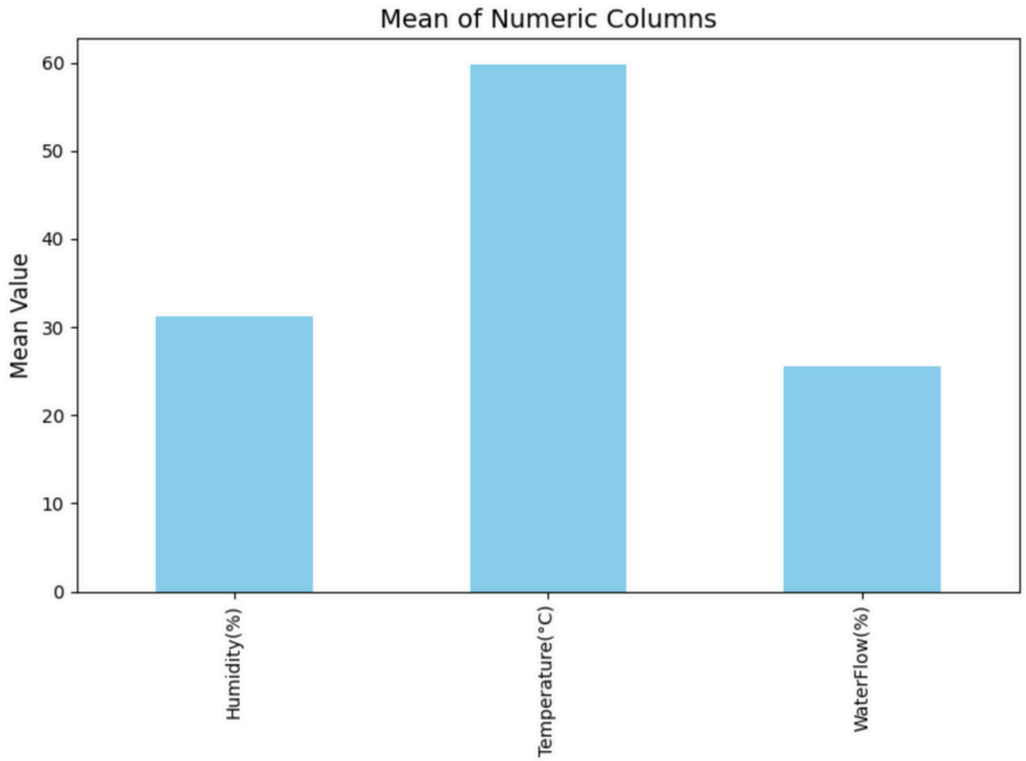
Search in table...

Humidity(%)	Temperature(°C)	WaterFlow(%)
25.045045	76.856857	67.432607
8.880881	31.771772	0.000000
24.956957	75.735736	66.105904
40.328328	88.308308	0.000000
38.742743	57.157157	0.000000
45.613614	24.164164	0.000000
33.237237	43.383383	0.000000
23.415415	81.421421	73.999716
10.026026	96.316316	100.000000
15.399399	58.838839	50.670417

Summary Statistics

Column	Count	Mean	Std	Min	25%	50%	75%	Max
Humidity(%)	200.0	31.137437	12.386344	8.132132	21.015015	30.616617	41.528529	51.867868
Temperature(°C)	200.0	59.747347	23.112767	20.000000	40.860861	58.638639	78.998999	100.000000
WaterFlow(%)	200.0	25.496364	34.953661	0.000000	0.000000	0.000000	61.884829	100.000000

Data Visualization



<THANK YOU>