End to End Pipeline for Chicago Crime Data

Data Engineering

April 19, 2025

Nishan Khanal

Contents

1. Project Overview

There are certain datasets for which we have up-to-date information in a single large file. However, if the instances in these datasets have temporal features and if the instances are bound to occur in the future, continuous integration is required to incorporate the future instances. I've decided to do this for the Chicago crime dataset. As of April 4, 2025, the dataset contains the crime records till "March 26, 2025". It is not suitable to download the entire dataset when up-to-date information is required, which is better suited for fetching smaller, newer portions through APIs. Using the API for fetching the entire dataset is also not suitable, as the dataset is huge, and we might exceed the rate limit quickly. The approach I suggest is to fetch the entire dataset once and use APIs to update the data as we move forward.

2. Data Source Selection

2.1 Dataset Choice

I've selected the **Chicago Crime Dataset** for this project. This dataset contains the reported incidents of crime in the City of Chicago from 2001 to present (minus the most recent 7 days). The dataset is deidentified in the spatial domain, i.e, the addresses are only accurate to the block level. This dataset will be helpful for analyzing crime patterns, and trends, and for predicting future crime hotspot. I will also be using supplementary datasets. I will be using the **Chicago Community Areas dataset**, which will aid in the visualization of the crime location in different communities. This dataset contains geographic information (GIS) about the different communities in Chicago.

2.2 Dataset Description

The Chicago Crime dataset contains ~ 8 million reported incidents and the following columns:

- **ID:** Unique identifier for each incident
- **Case Number:** Unique case number for each incident
- **Date:** Date and time of the incident
- **Block:** Block where the incident occurred
- **Primary Type:** Type of crime (e.g., theft, assault)
- **Description:** Description of the crime
- **Location Description:** Description of the location where the incident occurred
- **Arrest:** Indicates whether an arrest was made
- **Domestic:** Indicates whether the incident was domestic in nature
- **Beat:** Police beat where the incident occurred
- **District:** Police district where the incident occurred
- **Ward:** City ward where the incident occurred
- **Community Area:** Community area where the incident occurred
- **FBI Code:** FBI classification code for the crime
- **X Coordinate:** X coordinate of the incident location
- **Y Coordinate:** Y coordinate of the incident location
- **Year:** Year of the incident

- **Updated On:** Date and time when the record was last updated
- **Latitude:** Latitude of the incident location
- **Longitude:** Longitude of the incident location
- **Location:** Location of the incident (latitude and longitude)

The Chicago Community Areas dataset contains 77 rows - one for each community - and the following columns:

- **community area:** Unique identifier for each community area
- **shape_area:** Area of the community shape
- **perimeter:** Perimeter of the community shape
- **area_num_1:** Area id
- **area_numbe:** Area number
- **comarea_id:** Community area ID
- **comarea:** Community area name
- **shape_len:** Length of the community shape
- **geometry:** Geometry of the community area (multipolygon)

2.3 Selection Criteria and justification

While just downloading the large dataset would not carry appropriate weight in the extract phase of the ETL pipeline, my proposed approach of using an API to update the data periodically ensures that the extract phase is complex enough. As for the transformation, there are going to be a lot of steps involved, like handling missing values, mainly through imputation leveraging the help of a secondary dataset, removing outliers, etc., which will satisfy the complexity criteria of the project. And, for the Loading phase, I'll create an appropriate schema in PostgreSQL and load the data there.

3. Technology & Tool Stack

3.1 ETL Tools & Platforms

- **Python:** for accessing the dataset and implementing transformation logic. I plan on using the following libraries. (Some additional libraries may also be used along the way)
  - **pandas:** for data loading, manipulation and analysis
  - **NumPy:** for numerical operations
  - **sodapy:** for accessing the Chicago crime dataset API
  - **psycopg2:** for database connection
  - **geopandas:** for geospatial data manipulation
- **Apache Airflow/ Cron**: I'm planning to use Apache Airflow for workflow automation when accessing new data from the API and transforming it. I might end up just using cron if the workflow is simple.

3.2 Database & Storage Options

- **PostgreSQL:** I will load the transformed data into a local PostgreSQL database, which will then be used for analysis.

3.3 Visualization

- **Matplotlib/Seaborn:** for creating static, animated and interactive visualizations in Python.
- **GCP Looker Studio:** for creating interactive dashboards and reports.

4. Implementation Approach

Figure 1 shows the ETL workflow. The data will be extracted from the Chicago Crime dataset and the Chicago Community Areas dataset. The data will be transformed through cleaning, normalization, enrichment, and data type conversion. The transformed data will then be loaded into a PostgreSQL database. Finally, the data will be analyzed and visualized using Python libraries and GCP Looker Studio.

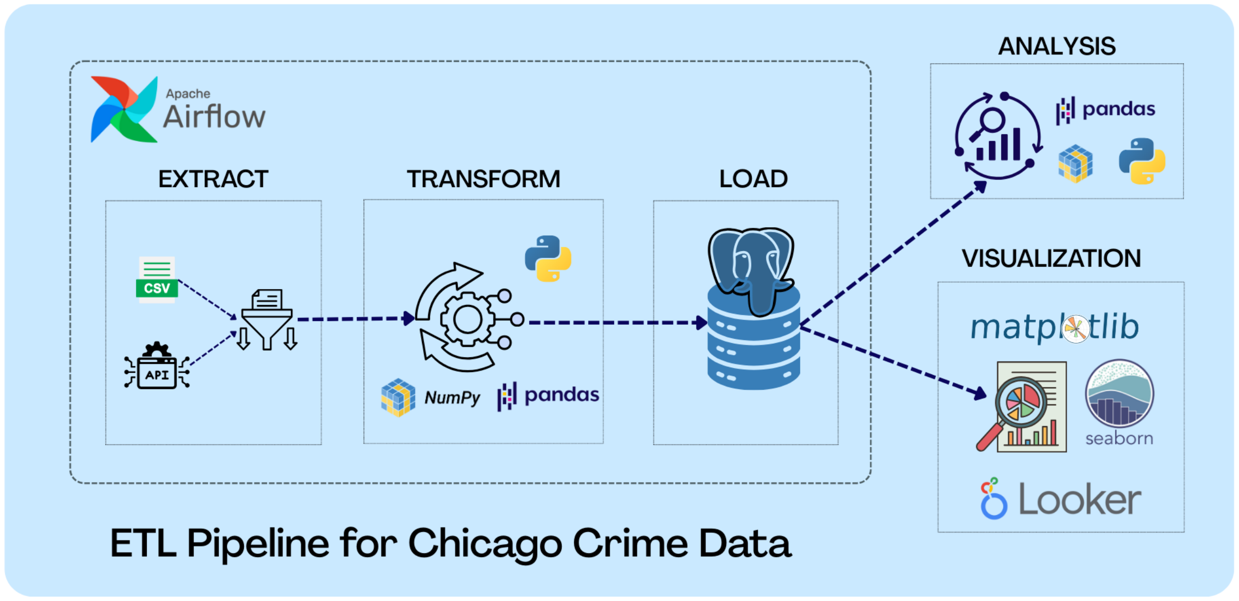

Figure 1: ETL Workflow

4.1 Data Extraction

The large crime dataset from 2001 to the present will be downloaded as a single CSV file from the link. This data will be updated through the use of the Chicago API, powered by Socrata. The supplementary Chicago Community Areas dataset will be downloaded as a single SHP/Geojson file, which includes the boundary data as multipolygons for each community.

The data extraction from the API will be automated and scheduled through the use of Apache Airflow or Cron.

4.2 Data Transformation

- **Cleaning:** In this phase, the missing values and outliers will be handled, and any duplicates will be removed. The columns will be converted to appropriate data types. The large dataset and the data coming in from the API have different names for columns, which will be handled by using a standardized naming convention. More specifically, in the cleaning phase, the missing values of the community areas will be imputed with the help of the Chicago Community Areas dataset by determining which crime location falls within which community by performing a spatial join (provided by geopandas).
- **Normalization:** The data will be stored as is. i.e. De-normalized. Since this database will be mostly used for querying rather than for transaction processing, I think a

- **Data Type Conversion:** The data types will be converted to appropriate types. For example, the date column will be converted to a date type, and the categorical columns will be converted to categorical types.denormalized table would be better.
- **Enrichment:** The main dataset will be accompanied by the Chicago Community Areas dataset. Aggregation will be mainly performed during the analysis phase. Temporal information will be decomposed for easy integration with another dataset in the future that only has aggregated information. I will store the data at the most granular level so that the aggregation can be done according to the purpose of the analysis.

4.3 Data Loading

- **Database Selection:** I first plan to load the data into a local PostgreSQL database. I chose a relational database because the dataset under consideration has a tabular structure.
- **Schema Design:** Since this dataset is huge and will be used mainly for analysis, it will be denormalized.
- **Loading Method:** psycopg2 will be used to make the connection, and bulk insert will be performed after the transformation.

4.4 Data Analysis & Visualization

- **Exploratory Data Analysis (EDA):** Jupyter Notebook, aided by Python libraries like matplotlib, seaborn, and plotly, will be used to explore data patterns, distributions, and correlations.
- **Visualization:** I will create visualizations using matplotlib and seaborn. I will also use Google Looker Studio to create interactive dashboards.

5. Project Timeline

Figure 2 shows the Gantt chart of the project timeline. The project is divided into four weeks, with each week focusing on a specific milestone. This is further described in table 1

Week	**Milestone**	**Description**
Week 1	Proposal Submission & Planning	- Submit the project proposal document, clearly outlining the data source, technology stack, tools, and implementation strategy - Finalize the dataset selection - Conduct initial data exploration - Outline ETL workflow structure (diagram or notes) to inform Week 2 development - Document the process
Week 2	Environment Setup & Initial Pipeline	- Set up development tools: Python environment, VS Code, and Git repo. - Begin scripting for data extraction (from API, file download, or scraping) - Implement and test initial transformation logic (cleaning, basic normalization) - Set up an orchestration tool (Airflow) to schedule data extraction jobs - Document the process

Week 3	Complete ETL Integration	- Expand transformation logic with enrichment and advanced cleaning/imputation tasks - Finalize data loading into the target storage system (PostgreSQL) - Design database schema and apply indexing - Begin testing for pipeline performance and data accuracy - Create automated workflows for recurring or batch ingestion - Document the process
Week 4	Visualization, Monitoring, & Final Documentation	- Create interactive dashboards using open-source BI tools (Metabase, Superset) or GCP's Looker Studio (if applicable) - Conduct final data quality checks and optimize transformation scripts - Document all implementation steps, architectural decisions, and visual outputs - Finalize project for submission/presentation, including README, code, dashboard links, and diagrams

Table 1: Project Timeline

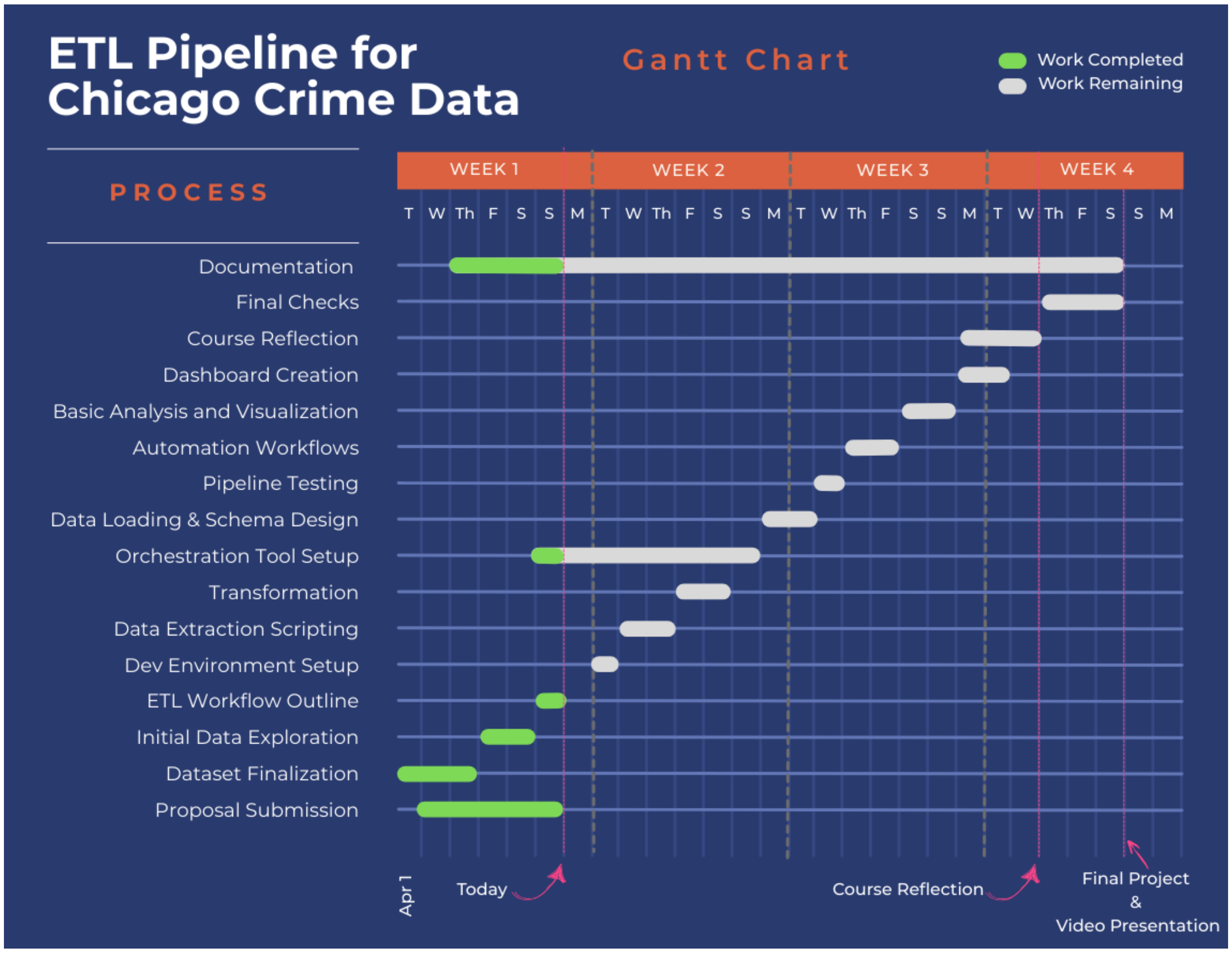

Figure 2: Gantt Chart of the Project Timeline

6. Expected Challenges

I am anticipating the following challenges. I've also proposed some strategies to overcome those challenges.

Challenge	Description	Proposed Strategy
Data Quality Issues	With the preliminary analysis, I've found that the data contains missing values, outliers, and invalid values.	I'll programmatically impute missing values for community_id from another dataset and remove instances that have invalid values.
API Issues	This is a public and freely available dataset, so with frequent requests, I can face a rate limitation.	All the available data till present will be downloaded as a static file. I plan to update the data daily, which won't have too many records, reducing the number of requests to the server.
Setting Environments with New Tools	I may face a problem in setting up Apache Airflow as I've never used it before.	I'll start familiarizing myself with new tools early on, as displayed on the Gantt chart.
Performance and Scalability	As the dataset contains $\sim$ 8 million rows, I might face some performance issues.	If such issues arise, I'll look into Spark and cloud-based processing.
Visualization Clarity and Usability	Creating visualizations that clearly convey insights without overwhelming users can be difficult.	I'll keep in mind that the visualization should be accessible to both technical and non-technical users. I'll focus on key performance indicators pertinent to my dataset.
Looker Studio Connection to Local Database	I may face issues connecting Looker Studio to my local PostgreSQL database.	I'll look into the documentation and community forums for solutions.

Table 2: Expected Challenges

References

[1] Apache Software Foundation. *Apache Airflow Documentation.* Apache Software Foundation, 2024. `https://airflow.apache.org/docs/`.

[2] City of Chicago. Boundaries - community areas (current). `https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Community-Areas-current-/cauq-8yn6`, 2024. Accessed: 2024-04-06.

[3] City of Chicago. Crimes - 2001 to present. `https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2`, 2024. Accessed: 2024-04-06.

[4] Federico Di Gregorio and Contributors. *psycopg: PostgreSQL database adapter for Python.* The Psycopg Project, 2024. `https://www.psycopg.org/`.

[5] Google LLC. *Looker Studio: Google's data visualization and reporting tool.* Google, 2024. `https://lookerstudio.google.com/`.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, et al. *NumPy: Fundamental package for scientific computing with Python.* NumPy Developers, 2024. `https://numpy.org/`.

[7] Kelsey Jordahl and Contributors. *GeoPandas: Python tools for geographic data.* GeoPandas Developers, 2024. `https://geopandas.org/`.

[8] Wes McKinney and The Pandas Development Team. *pandas: Powerful Python data analysis toolkit*. Pandas Community, 2024. `https://pandas.pydata.org/`.

[9] Socrata and Contributors. *Sodapy: A Python client for the Socrata Open Data API*. Socrata, 2024. `https://github.com/socrata/sodapy`.

[10] The PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open source relational database*. PostgreSQL Global Development Group, 2024. `https://www.postgresql.org/docs/`.

[11] Michael Waskom and the Seaborn Development Team. *Seaborn: Statistical Data Visualization*. Seaborn Developers, 2024. `https://seaborn.pydata.org/`.