National College of Ireland

MSc in Artificial Intelligence

24269522 – Smruti Pote

Programming for AI
MSCAI1, MSCAI1B
Due Date: Monday 15th Dec. 2025
Github Link :
https://github.com/smrutipote/Programming-for-AI

Table of Contents

Journal of Artificial Intelligence Programming Project

Project entry 1 - understanding of project and searching of the data set
Date: 20 Nov 2025
Time spent: ~2 hours

Tasks:

- Data retrieval Programmatic data retrieval.
- Pretreatment and posttreatment database use.
- Non-trivial preprocessing, analysis and visualisation.
- Well-supported ambitious project goals.
- Browse a range of possible data sets (a variety of CSV/JSON files) on the open databases.

Determined applicants according to:

- Data volume having records 1000.
- Quality of documentation and accessibility of API.

Reasoning / decisions:

A great number of candidate datasets were too small, too simple (not many columns, too limited time span), or not well documented, which would not sustain an exceptionally challenging project. The CJQ06 - Recorded Crime Incidents data set of the Central Statistics Office was interesting due to the following reason:

- It is official, high-quality, consistently coded and nationally covered in several years.
- It is massive 180k+ rows, and it has a number of dimensions (time, Garda division, offence type).
- It provides a CSO API endpoint in CSV format, available in an easily reachable way, and which is programmatically accessible.

Why this dataset was chosen

It is a formal and it is quality data of the Central statistics office, also it is well structured, coded in a consistent manner and covers all national areas in several years.

It is non-trivial and large (more than 180,000 rows) that comfortably meets the need of 1,000 records at minimum and provides the sufficiently challenging goals (more than one dimension: time, region, offence type, measures).

Entry 2 – Doing the first entry and into the database (SQLite and Neon)
Date: 27- 28 Nov 2025
Time spent: ~6 hours

Tasks:

Downloaded the CJQ06 data through Python with pd.readcsv on CSO CSV URL. Columns such as, STATISTIC, Statistic Label, TLIST(Q1), Quarter, Garda Division, Type of Offence, VALUE. Types of data (text/ integers / floats mix) and non null amounts.

One of them was created and the raw table stored in a local SQLite database file (criminaldatabase.db)

- Install a Neon Postgres and:
- Split the DSN string and got DBUSER, DBPass, DBhost, DBName and DB Port.
- Created a SQLAlchemy engine and put the criminalraw raw table in Neon.
- SELECT * of both databases are verified. LIMIT 5 using pd.readsql

Challenges and fixes:

Where is the database?
Misunderstanding of the SQLite in Colab (file vs server). Fixed by listing files (os.listdir()) and ensuring that the.db file was in the working directory.
The long DSN (with sslmode, etc.) initially appeared to be daunting. I resolved it by splitting it into user, password, host and DB by hand and reconstruction of a simpler SQLAlchemy URL (postgresql+psycopg2://user:pass@host:port/db).

Table naming:
Initially crimeraw and criminalraw were in use and this was confusing. Normalised on a well-defined standard naming scheme of raw and processed tables and confirmed them using sqlitemaster and informationschema.tables.

Issues/database design (SQLite vs Neon PostgreSQL).
It was necessary to have a local SQLite database as well as a cloud-based Neon PostgreSQL instance.
Neon PostgreSQL (cloud-based): Displays application of a complete, server based relational database. The situation that is more reflective of real-world AI/data engineering deployments where data is centred in a controlled DB service. It also supported the test of more realistic connection management like credentials, network connectivity, SQL dialect.

Problems and the manner in which they were addressed:

Knowledge of the location of the database:
At first there was misunderstanding on where SQLite resided in Colab. This was solved by enumerating the working directory and ascertaining that criminaldatabase.db was a local file.

Running SQL vs Python:
When trying to type in raw SELECT tablename ... statements, syntax errors occurred in a cell in Python. This was resolved by enclosing SQL in pd.readsql(...) such that python will send the query to the database instead of considering it as a code.

Entry 3 - Table Standardization and Semantic renaming.
Date: 1-2 Dec 2025
Time spent: ~5 hours

Tasks:

Preloaded the raw table retrieved out of the SQLite database into criminaldf.

Standardised column names: Stripped spaces, Lowercased, Eliminated non numeric characters.

Output names were statistic, statisticlabel, tlistq1, quarter/timeperiod, gardadivision, typeoffence, value.

- VALUE - valuegiven (counts of incidents).
- Columns of the year - yearofoccurrence / crimeyear.
- Quarter / Time -timeperiod (year-quarter codes).
- The names of confirmed columns after being renamed using print(criminaldf.columns).

Challenges and fixes:

KeyError on renamed columns:
I also used Old names (value, offencegroup, groupedoffence, year). This generated Key Error when using columns which were no longer there. The purpose of renaming is to make sure that criminal df columns are printed after each renaming.

Original column names including STATISTIC, Statistic Label, TLIST(Q1), Quarter, Garda Division,Type of Offence, VALUE were reformatted to a lower case with underscores (e.g. statistic, statisticlabel, timeperiod).

- The next step was a mapping step which gave semantic names suited to analysis: valuegiven in incident counts, crimeyear/yearofoccurrence in years, and timeperiod in quarter-encoded time.
- A periodbucket attribute was introduced to separate the records before 2015 and the ones on and after 2015, which will be useful in a comparative analysis in the future.

Missing data and smoothing:

- Valuegiven had missing values which were detected and dropped or imputed where needed.

- Missing entries of yearfromtime used yearfromtime to fill missing crimeyear entries.
- In order to smooth gaps and permit analysis of a continuous time-series of offenses by sequence and year, forward and backward filling within each offence group was employed.
- The dataset was then explored using info, describe, value counts and pivot tables and a picture of its distribution over time, offence type and geography was developed.

Entry 4 - Time features, missing data and smoothing.
Date: 4 - 5 Dec 2025
Time spent: ~5 hours

Tasks:

Added features:

- Modified the timeperiod yearfromtime of timeperiod (2003Q1 - 2003) and transformed to numeric.
- Calculated crimeyear based on the information on the year where the information is available; missing years were typed in based on yearfromtime.
- Created a periodbucket feature which marked every record with pre2015 or 2015andafter so that they could be compared later.
- Missing data was dtected with isna ().sum, primarily used with value given.

Handled missing values:

- The rows that had valuegiven as missing dropped as dependable counts in the analysis was needed.
- Applied forward and backward fill (ffill() and bfill()) on each group of offences by crimeyear in order to fill the gaps and allow continuous time series.

Challenges and fixes:

1. SettingWithCopyWarning:

In the updating of columns such as crimeyear and yearfromtime, pandas issued a SettingWithCopyWarning that suggested that operations may be operating a view and not a copy. This was resolved doing - criminaldf = criminaldf.copy()

2. KeyError on renamed columns:

Some of the code was left to refer to old names after standardisation and renaming of columns (e.g. VALUE - valuegiven, Type of Offence - typeofoffence, Quarter - timeperiod, year - crimeyear), including value, offencegroup, or groupedoffence. This resulted in the KeyError exceptions in attempt to access non-existent columns.

3. Problems for null and type conversion:
Conversion of valuegiven and year fields needed to be sensitive to non-numeric values and errors=coerce was used to make invalid values to NaN and allow operations on them to proceed safely later.

4. Code was failing to run, visualisation code did not run:
The output of some of the visualisation blocks did not display due to an error of not having the required conditions. Conditions and arguments were revised after confirming column names by using typeofoffence.

Entry 5 -Outlier Detection and Transformation
Date: 8-9 Dec 2025
Time spent: ~2 hours

Tasks:

Applied simple z-score based outlier detection on valuegiven:
Calculated z scores and flagged z scores Z>3 into an isoutlier column.

Introduced winsorisation:
Created value capped with clipping value provided between the 1 st and 99 th percentiles to minimize the impact of extreme values.

This was a log-transformed feature:
The skewness of the distribution of the number of incidences is dealt with by the use of logvalue = np.log1p(valuegiven). Took printed paired samples of valuegiven and logvalue to indicate the transform effect.

Challenges and fixes:
Realised that raw crime counts were very skewed therefore did the log transform and winsorisation to stabilize the variance and allow more useable visualisation.

Entry 6 - Trend Detection by EDA and Visualisation.
Date: 9-10 Dec 2025
Time spent: ~5 hours

Tasks:

- Cumulative number of incidences per year, as a line chart in orange colour, in order to emphasise the critical points.
- Offence-level analysis: Constructed pivot tables of typeofoffence vs crimeyear to construct multi year offence profiles.

- Created a bar chart of the 10 most common types of offences by incurred incidences shown with yellow bars to reveal the most common types.
- Created multi-series line plots of the top offence types per time to identify which categories increased or decreased in comparison to those.

Distribution and outliers:

- Graph of the distribution of valuegiven in the form of histograms.
- Diagrams of the logvalue to indicate how the log-transform equalizes the distribution.
- Boxplots of valuegiven to identify outliers identified above.

Geographical trends:
A series of geographical patterns has formed: Incidents by typeofoffence, incident by crimeyear: Heatmap to identify offence types with either repeated high activity or significant trends.

Comparison of the pre and post 2015 offence mix: Compared before 2015 and since 2015 by plotting the share of the top offences with pink bar chart to bring out the structural changes in crime mix.

Challenges and fixes:

Blocked visualisation non-executing processes: There were some if conditions that checked on groupedoffence which never occurred in the final schema (that is the correct column is typeofoffence). All these conditions were False and thus the plots did not appear.

This was fixed by: Printing the columns that are existing and ensuring that the necessary sets are observed. Substituting all instances of groupedoffence with typeofoffence in both states and plotting arguments.

Patterns and observations:

- The annual numbers revealed the obvious trends over time, and it was possible to distinguish the years when the incidences rose, leveled, or fell.

- The plots of offence type reflected the relative contribution of various types to the total crime followed by whether this was constant or was varying.

- Pre- vs post-2015 offence mix analysis implied the change in the relative weight of types of offences, which showed the structural changes of the crime profile.

- The region-offence showed that there are some areas of the Garda that consistently feature in the high ranking of on particular offences suggesting geographical concentration.

- Distribution plots supported heavy skew and transformations were necessary to avoid the dominance of a small number of extreme cells in the analysis.

Entry 7 -Limitations and work in future.
Date: 11-12 Dec 2025
Time spent: ~2.5 hours

Reflections: The project realized an end-to-end pipeline on a single complicated dataset (CJQ06), and met the most demanding aspects of the brief:

- Real-time API retrieval.
- Database are local Sqlite along with Neon Postgres
- Extreme preprocessing, such as time feature engineering, missing value treatment, outlier treatment, log transforms.
- Multi-angle EDA and visualisation with clear understanding of the territorial, offence-type and regional patterns.
- A large part of the current code is reusable and can be disaggregated into similar structured/semi-structured sources (e.g. other CSO cubes, JSON APIs).

Future work:

- A second crime related cube CSO crime or a socio-economic indicator dataset to be compared.
- Later can elaborate on more advanced modelling over the existing EDA: Basic prediction of occurrence by type of offence on the basis of time-series. A grouping of the areas in terms of offence patterns to find out some common Garda divisions.
- Enhance database design added indexing and basic normalisation in Postgres to optimise huge tables queries. Introduce a basic ETL architecture with the distinct layers of raw, cleaned and aggregated schema.
- Refined visual storytelling by integrating important charts. Reduced the number of high impact visuals to feature in the report and presentation like trends plots, offence mix changes, region hot-spots.

Additional Tasks Performed

- Created Powerpoint Presentation for the team.
- Created Visualizations for presentation.
- Along with local SQL lite Database, also used cloud Postgres Database.
- Made the Overleaf Latex Template in the format of IEEE.
- Researched all the related papers mentioned in the report to compare our work.