Syracuse 911 Demand Explorer – Architecture Review (Week 6)

Phase 3 – Development (Weeks 5–6)

1. System Overview

The Syracuse 911 Demand Explorer project analyzes calls-for-service and parking violations in the City of Syracuse to understand temporal patterns and prepare data for downstream reporting tools (e.g., Power BI).

During Weeks 5–6, the focus was on turning the exploratory work from Phase 2 into a reproducible analysis pipeline with basic quality checks and clear documentation.

2. System Architecture and Data Flow

The project follows a simple, linear data flow:

- **Raw Data (local CSV):**
  - Crime_Data_2023_Part_2_Offenses_With_Lat_and_Long.csv
  - Crime_Data_2024_Part_2_Offenses_With_Lat_and_Long_Info.csv
  - Crime_Data_2025_Part_1_Offenses_With_Lat_and_Long_Info.csv
  - Parking_Violations_-_2023_-_Present.csv
- **Phase 3 Notebook (Python / Jupyter)**
  - Reads raw CSVs from data_raw/.
  - Cleans and combines crime data.
  - Cleans parking data.
  - Applies sanity checks and lightweight tests.
  - Saves analysis-ready outputs to data_processed/.
  - Generates figures in figures/ for reports and slide decks.
- **Processed Outputs (analysis-ready)**
  - Crime: crime_clean.parquet, crime_clean.csv
  - Parking: parking_clean.parquet, parking_clean.csv
- **Downstream Consumption**
  - Python notebooks (future analysis or modeling) use Parquet files for fast, type-safe loading.
  - Power BI / Excel and other tools connect to the CSV files in data_processed/.
  - Visuals from figures/ can be dropped into reports and presentations.

Directory layout (current):

- notebooks/
  - phase2_exploration.ipynb – Week 3–4 EDA notebook
  - phase3_pipeline.ipynb – Week 5–6 pipeline notebook
- data_raw/ – raw CSVs (not committed if large)

- data_processed/ – cleaned Parquet and CSV outputs
- figures/ – PNG figures generated by the pipeline
- README.md – instructions for running the pipeline and understanding outputs

3. **Key Design Decisions and Rationale**

1. Separate notebooks for EDA vs pipeline

- Decision: Keep phase2_exploration.ipynb focused on exploratory analysis and create a dedicated phase3_pipeline.ipynb for the reproducible data pipeline.
- Rationale:
  - Preserves the "research trail" of Week 3–4 exploration.
  - Keeps the Week 5–6 notebook cleaner and easier to re-run top-to-bottom.
  - Aligns with the capstone requirement to distinguish exploration from production-style processing.

2. Use local CSVs as raw input, Parquet + CSV as processed output

- Decision: Read raw data from CSV, then save cleaned data in both Parquet and CSV formats.
- Rationale:
  - CSV is the format in which the city publishes data and is trivial to connect from Power BI or Excel.
  - Parquet provides faster read/write performance in Python and preserves data types (especially datetimes) more reliably
  - Saving both formats balances ease of use for reporting tools with efficiency for future Python analysis.

3. Minimal, transparent cleaning rules

Key cleaning steps include:

- Parsing DATEEND (crime) and issued_date (parking) as timezone-naive datetimes.
- Filtering out impossible future dates by removing crime records with DATEEND year greater than 2030 (e.g., a clear typo with year 2210).
- Converting parking amount to numeric and dropping negative values.
- Deriving simple features:
  - Crime: year, hour, dow (day of week), month (period).
  - Parking: date, month.

Rationale:

- The goal for Phase 3 is not heavy modeling but dependable aggregations and charts.

- Simple, explainable rules are easy to document and justify in the final report.

4. Light-weight tests instead of a full test suite

- Decision: Implement a small set of Python "test" functions at the end of the pipeline notebook rather than a separate, full unit-test framework.
- Current tests include:
  - test_no_future_crime_dates: checks that crime dates do not go beyond a reasonable year.
  - test_parking_amount_non_negative: ensures ticket amounts are non-negative after cleaning.
  - test_crime_lat_long_not_null: reports fraction of missing latitude/longitude and asserts that fewer than 50% are missing.
  - test_parking_year_range_soft: prints the min and max parking years and asserts only that the minimum year is reasonable (≥ 2000).
- Rationale:
  - Keeps effort low (consistent with project goals) while still adding a QA step to detect major data issues.
  - Tests are embedded directly in the notebook so they run automatically when the pipeline is executed.

5. Figures saved to disk with filenames tied to the analysis

- Decision: Save key visualizations as .png files with informative names, e.g.,
  - figures/calls_by_hour.png
  - figures/calls_by_dow.png
  - figures/top_incident_types.png
  - figures/parking_by_month.png
- Rationale:
  - Makes it easy to drop figures into reports, slides, and documentation without re-running code.
  - Provides a clear link between the pipeline outputs and the narrative in the final deliverables.

4. Dependencies

The pipeline relies on a small set of standard Python libraries:

- Python version: 3.x
- Core libraries:
  - pandas – data loading, cleaning, aggregation
  - numpy – basic numeric support

- matplotlib and seaborn – visualization
- pathlib – filesystem-safe path handling

No external APIs, databases, or LLM services are used at this stage. All data is local.

5. Execution and Reproducibility

To run the pipeline end-to-end:

1. Clone the GitHub repository.
2. Place the raw CSVs provided by the City of Syracuse into the data_raw/ folder using the filenames listed above.
3. Open Jupyter Lab / Notebook and run notebooks/phase3_pipeline.ipynb from top to bottom.
4. The notebook will:
   - Read the raw CSVs from data_raw/.
   - Perform cleaning and feature engineering.
   - Drop records with impossible dates and invalid ticket amounts.
   - Save cleaned outputs to data_processed/ in both Parquet and CSV formats.
   - Save the key figures to figures/.
   - Run basic QA tests and print confirmation messages.
5. After running the notebook, downstream tools can connect to:
   - data_processed/crime_clean.csv
   - data_processed/parking_clean.csv

for Power BI dashboards or further analysis.

6. Current Progress (End of Week 6)

By the end of Week 6, the project status is:

- Data pipeline:
  - Implemented in phase3_pipeline.ipynb.
  - Reads, cleans, and saves both crime and parking datasets.
  - Produces pre-aggregated tables and figures for temporal patterns.
- Quality assurance:
  - Basic tests implemented and run as part of the notebook.
  - Impossible crime dates and invalid parking amounts are filtered out
  - Lat/long and date ranges are checked with printed diagnostics.
- Architecture & documentation:

- System architecture and design decisions documented in this Week-6 architecture review.
- Project structure (notebooks, data folders, figures) aligned with reproducible analysis expectations.

7. Known Limitations and Blockers

- LLM integration:
  - No LLM-based components or prompt engineering are implemented yet.
  - This is planned for later phases (e.g., natural-language summaries of findings).
- Testing scope:
  - Tests are intentionally lightweight and live inside the notebook rather than a standalone test suite (e.g., pytest).
  - For production-grade deployment, more rigorous tests around edge cases and schema changes would be required.
- Coverage of relationships between datasets:
  - The current pipeline cleans and summarizes crime and parking data separately.
  - Deeper relational analysis (e.g., spatial or temporal correlations between the two datasets) is planned for later phases or the dashboard layer.

8. Next Steps (Looking Ahead to Week 8)

- Build an initial Power BI (or similar) prototype connected to crime_clean.csv and parking_clean.csv.
- Add a first draft of the interactive interface or dashboard that uses the figures and tables produced by the pipeline.
- Optionally refactor repeated notebook code into small reusable functions or a simple src/utils.py module if needed.