# Real Estate: Data Integration, Cleaning API & Modeling Project

## Important Notes (Read Carefully)

* **Individual Project** – This is strictly an individual assignment. Collaboration is not allowed.
* **Internet Use** – You may search for concepts or documentation online. However, copying code directly from external sources is prohibited.
* **Evaluation Format** – You will participate in a one-on-one review, where you must explain your process, decisions, and code functionality.
* **Timeline**
  + Start: Wednesday morning
  + Deadline: End of day Thursday
* **Early Submission** – If you complete and push your project to GitHub early, you’re free to leave the workspace.

## Project Objective

This project simulates a real-world data science task. You will **collect property data from two real estate APIs**, **clean and integrate the data**, and then **build a robust predictive model** based on pricing. The final deliverable will be a GitHub repository containing all project artifacts. The primary challenges in this project are to implement the data cleaning process and the predictive model as separate APIs.

## Assigned APIs

You’ll be assigned two real estate APIs. Each API provides listings for properties for rent and properties for sale.

* API 1 Endpoint: https://www.attomdata.com/solutions/property-data-api/
* API 2 Endpoint: https://www.rentcast.io/api

Your task is to collect data relevant to the pricing and characteristics of listed properties.

## Project Requirements

### 1. Data Collection from APIs

* Fetch relevant data (e.g., property title, location, number of rooms, price, size, listing type) from both APIs.
* Handle API authentication, rate limiting, and error handling as necessary.
* Save the raw data from each API in a structured format (e.g., JSON or CSV) before cleaning.

### 2. Data Cleaning & Integration (Mandatory, API is Challenge)

* **Implement data cleaning functions** in Python to address issues such as missing values, inconsistent formatting, and duplicates.
* **Combine data from both sources** into a single, cleaned dataset.
* Ensure consistency in column naming and data types.
* **Challenge:** Design and implement a RESTful API (e.g., using Flask or FastAPI) that exposes an endpoint for data cleaning.
  + The API should accept raw property data (e.g., as a JSON payload or a file upload).
  + It should apply your cleaning and integration logic.
  + The API should return the cleaned and integrated data in a structured format (e.g., JSON or a downloadable CSV).

### 3. Predictive Modeling (Mandatory, API is Challenge)

* Define a clear modeling objective, such as predicting property price.
* Prepare your dataset accordingly (split features/target, handle categorical variables, etc.).
* Implement and evaluate suitable Scikit-learn models for the task (e.g., Linear Regression, Decision Tree, Random Forest, Gradient Boosting).
* Justify your choice of models and evaluation metrics, and demonstrate the model’s performance on unseen data.
* **Challenge:** Design and implement a RESTful API (e.g., using Flask or FastAPI) that exposes an endpoint for making predictions.
  + The API should accept new, unseen property data (e.g., as a JSON payload).
  + It should use your trained model to generate predictions.
  + The API should return the predicted values.

### 4. GitHub Repository

Each student must create a public GitHub repository with the following:

* Data collection scripts or notebooks (for fetching from APIs)
* Data cleaning scripts/functions (the core cleaning logic)
* **Data Cleaning API code** (if attempting the challenge, including requirements.txt and clear instructions to run)
* Final cleaned & combined dataset (output from the cleaning process, or from the cleaning API if implemented)
* Modeling code and trained model artifacts (if applicable)
* **Predictive Modeling API code** (if attempting the challenge, including requirements.txt and clear instructions to run)
* A brief README.md file explaining:
  + Your objectives
  + APIs used for data collection
  + Steps taken in data collection and cleaning
  + How to run and interact with the cleaning API (if implemented)
  + Modeling approach and results
  + How to run and interact with the predictive modeling API (if implemented)

## Deliverables Summary

| Deliverable | Format |
| --- | --- |
| Raw API data (from each API) | JSON or CSV |
| Data Cleaning code (core logic) | .py script or notebook |
| **Data Cleaning API code (Challenge)** | .py scripts, Dockerfile (optional) |
| Final cleaned & combined dataset | CSV or JSON |
| Modeling code | Notebook or script |
| **Predictive Modeling API code (Challenge)** | .py scripts, Dockerfile (optional) |
| GitHub repository with all files | Public URL |

## Evaluation Criteria

| Component | Weight |
| --- | --- |
| Data Collection from APIs | 30% |
| Data Cleaning & Integration (Mandatory) | 30% |
| Predictive Modeling (Mandatory) | 25% |
| **Data Cleaning API & Predictive Modeling API (Challenge)** | +15% Bonus |
| Code Quality & GitHub Setup | 15% |

*The bonus challenges will not affect your core score but can significantly boost your final evaluation if completed thoughtfully and correctly.*

## Final Reminders

* Structure your code professionally and document your functions.
* Keep your GitHub repo clean and organized, providing clear instructions for running your APIs (if implemented) and modeling code.
* Be prepared to explain your choices in the evaluation interview.