# End-to-End Real Estate Data Engineering and Analytics Pipeline



This project demonstrates the development of a comprehensive data engineering pipeline that transforms raw real estate data into actionable business intelligence. Built specifically for the Hyderabad real estate market, the solution processes approximately 12,000 property listings through a modern data architecture, delivering insights through interactive dashboards for informed decision making



Kotha Pavan Kalyan

## 

## Problem Statement

The real estate market in Hyderabad generates vast amounts of property data across multiple platforms, but this information remains fragmented and difficult to analyze effectively. Real estate stakeholders including investors, buyers, agents, and market analysts face several challenges:

* **Data Fragmentation**: Property information scattered across multiple platforms with inconsistent formats
* **Manual Analysis**: Time-consuming manual processes for market research and property comparison
* **Lack of Real-time Insights**: Absence of dynamic dashboards for tracking market trends and pricing patterns
* **Inconsistent Data Quality**: Varying data standards across sources leading to unreliable analysis
* **Limited Geographic Analysis**: Difficulty in understanding locality-wise pricing trends and distribution patterns

This project addresses these challenges by creating an automated, scalable data pipeline that consolidates, cleans, and transforms real estate data into meaningful insights accessible through interactive dashboards.

## Solution Architecture

### Architecture Overview

The solution implements a modern medallion architecture with distinct layers for data processing:

**Bronze Layer (Raw Data)**: Initial data ingestion and storage in AWS S3

**Silver Layer (Processed Data)**: Transformed and cleaned data in Databricks

**Gold Layer (Analytics-Ready)**: Structured data warehouse in Snowflake with star schema design

### Data Source

### Data Flow Architecture

Housing.com → Apify (Web Scraping) → Python (Data Cleaning) → AWS S3 (Staging) → Databricks (Transformation) → Snowflake (Data Warehouse) → Power BI (Visualization)

The architecture ensures:

* **Scalability**: Cloud-native components handle growing data volumes
* **Reliability**: Multiple checkpoints and validation stages
* **Performance**: Distributed processing with Apache Spark
* **Flexibility**: Modular design allows component upgrades without system disruption

## 

## Technical Implementation

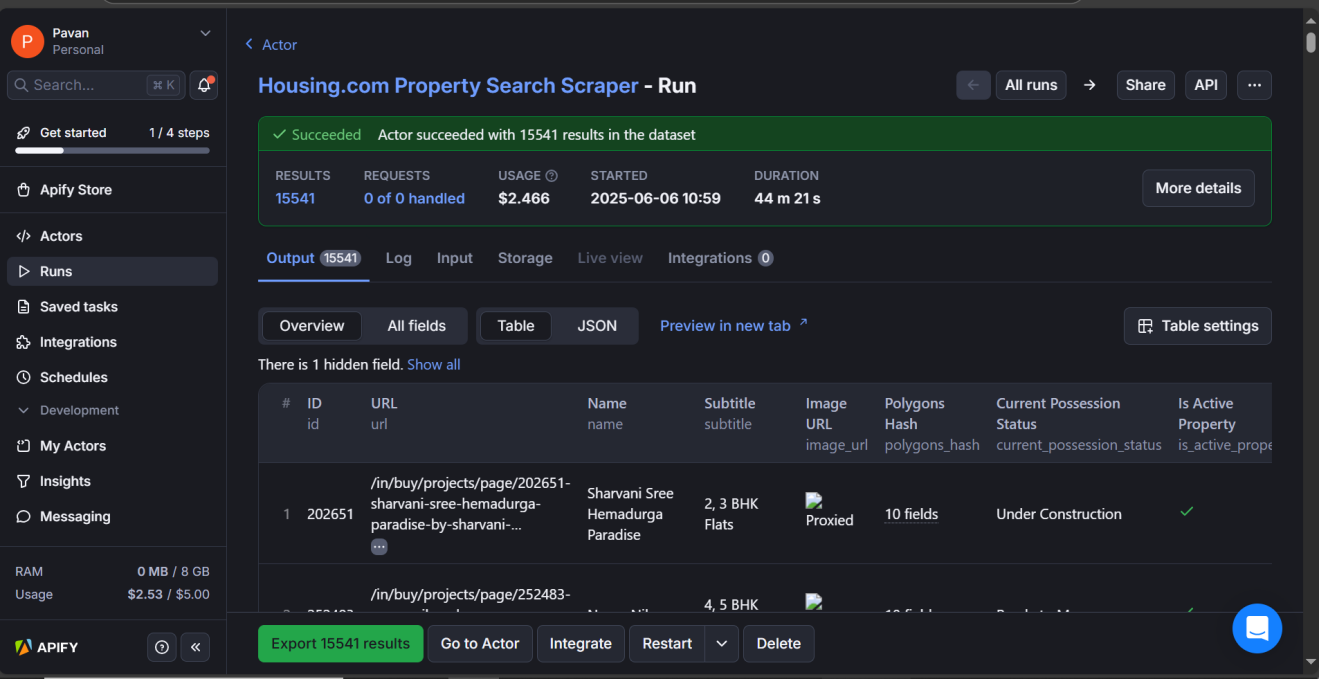
### Phase 1: Data Collection and Web Scraping

**Technology Stack**: Apify, Housing.com API

**Objective**: Extract comprehensive property listing data from Housing.com

**Implementation Details**:

* Configured Apify scrapers to collect approximately 12,000 property listings



* Captured essential property attributes including:
  + Location and locality information
  + Property configuration (1 BHK, 2 BHK, 3 BHK, etc.)
  + Pricing details and price ranges
  + Area measurements (built-up area, carpet area)
  + Amenities and features
  + Builder and developer information
  + Listing metadata and timestamps

**Data Quality Measures**:

* Implemented rate limiting to respect website policies
* Added retry mechanisms for failed requests
* Captured data lineage information for traceability

### 

### Phase 2: Data Cleaning and Preprocessing

**Technology Stack**: Python, Pandas, JSON, Regular Expressions

**Objective**: Transform raw scraped data into consistent, analysis-ready format

**Key Preprocessing Steps**:

1. **Data Standardization**:
   * Normalized price formats (removing currency symbols, converting to numeric)
   * Standardized area measurements to consistent units
   * Cleaned and categorized property configurations
2. **Derived Column Creation**:
   * Calculated price per square foot metrics
   * Created configuration categories (Studio, 1BHK, 2BHK, 3BHK+)
   * Generated locality-based geographic tags
   * Added property type classifications
3. **Data Structure Optimization**:
   * Converted complex amenities lists into structured JSON format
   * Created nested JSON structures for dynamic content
   * Implemented data type validation and conversion
4. **Quality Assurance**:
   * Handled missing values using domain-specific imputation strategies
   * Removed duplicate listings based on multiple criteria
   * Validated data ranges and business rules

**Output**: Clean, structured JSON files ready for cloud ingestion

### 

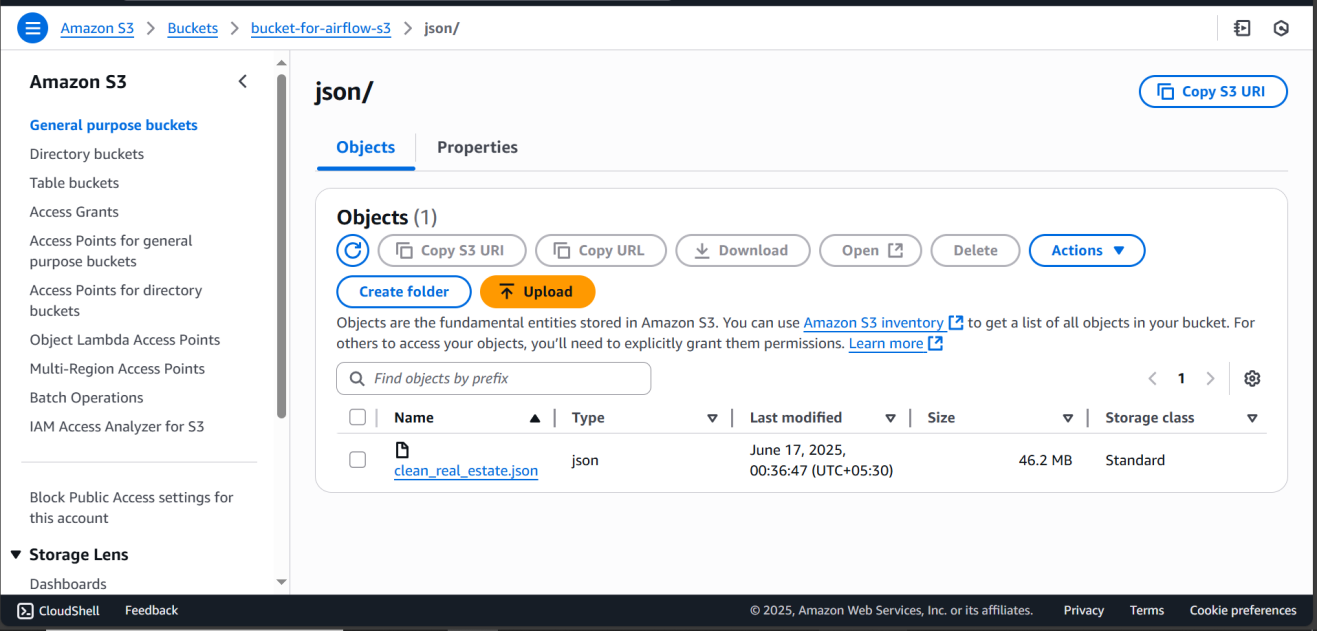
### Phase 3: Staging Layer - AWS S3

**Technology Stack**: AWS S3, JSON

**Objective**: Provide scalable, reliable staging storage for processed data

**Implementation Features**:

* **Partitioned Storage**: Organized data by date and locality for efficient querying
* **Schema Consistency**: Maintained consistent JSON schemas across all files
* **Access Control**: Implemented proper IAM roles and bucket policies
* **Versioning**: Enabled S3 versioning for data recovery and auditing



**Benefits**:

* Decoupled data processing from downstream systems
* Provided scalable storage solution supporting growth
* Enabled parallel processing in subsequent stages
* Served as reliable backup and recovery point

### 

### Phase 4: Data Transformation - Databricks with Apache Spark

**Technology Stack**: Databricks, Apache Spark, PySpark, SQL, Snowflake Connector **Objective**: Parse nested JSON data and perform comprehensive transformations before loading to Snowflake

**Implementation Architecture**: Two-notebook approach for modular data processing

#### Notebook 1: JSON Parsing and Table Creation

**Purpose**: Parse complex nested JSON files from S3 and create structured relational tables

**Input**: Staged JSON files from AWS S3 bucket

**Output**: Five normalized tables representing different aspects of property data

**Key Operations**:

1. **JSON Data Ingestion**:

* # Read JSON files from S3 staging area
* df = spark.read.option("multiline", "true").json("s3://bucket-for-airflow-s3/json/clean\_real\_estate.json")

1. **Nested JSON Parsing**:
   * **Properties Table**: Core property information (ID, price, area, configuration)
   * **Near by Places Table**: Details of Nearby landmarks.
   * **Amenities Table**: Property amenities and features (flattened from nested arrays)
   * **Tags Table**: Tags like under construction, Ready to Move,..
   * **Configurations Table :** Details of Configurations like 3bhk, 2bhk …
2. **Schema Normalization**:

* # Example: Extracting amenities from nested JSON
  + df\_places = df.select(
  + col("id"),
  + explode(col("nearby\_places")).alias("place")
  + ).select(
  + col("id"),
  + col("place.type").alias("place\_type"),
  + col("place.name").alias("place\_name"),
  + col("place.distance\_km"),
  + col("place.travel\_time\_min")
  + )

1. **Data Type Standardization**:
   * Converted string prices to numeric format
   * Standardized date formats across all tables
   * Applied consistent data types for join operations
2. **Primary and Foreign Key Establishment**:
   * Generated unique identifiers for each table
   * Established referential integrity relationships
   * Created surrogate keys for dimension tables

#### Notebook 2: Main Transformations and Snowflake Loading

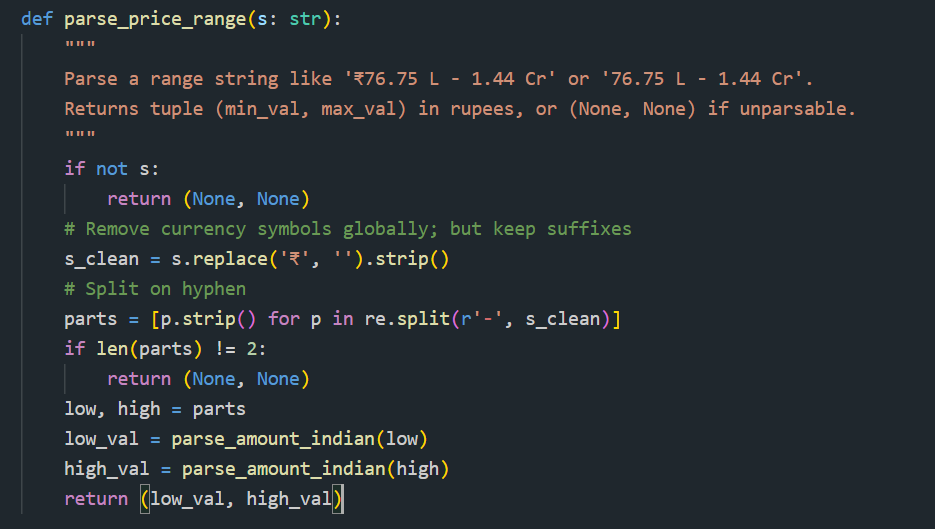
**Purpose**: Apply business logic transformations and load processed data directly to Snowflake

**Input**: Five structured tables from Notebook 1

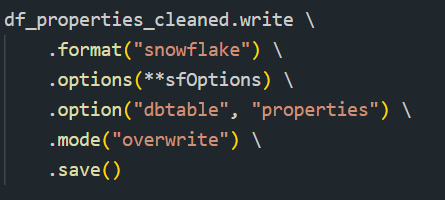
**Output**: Analytics-ready data in Snowflake data warehouse

**Transformation Operations**:

1. **UDF Logic Implementation**:



1. **Data Enrichment**:
   * **Configuration Standardization**: Parsed “2 BHK”, “3 BHK” into structured bedroom/bathroom counts
   * **Locality Scoring**: Created locality desirability scores based on amenities and connectivity
   * **Price Categorization**: Segmented properties into budget categories (Affordable, Mid-range, Premium, Luxury)
   * **Market Positioning**: Calculated percentile rankings within locality and configuration
2. **Derived Metrics Creation**:
   * Price appreciation indicators
   * Value-for-money scores
   * Investment potential ratings
   * Market competitiveness index
3. **Snowflake Integration**:

* # Direct write to Snowflake using connector
* 

**Snowflake Connector Configuration**:

* Established secure connections using Snowflake connector for Spark
* Implemented incremental loading strategies for daily updates
* Configured parallel writing for optimal performance
* Set up error handling and retry mechanisms for connection failures

**Table Loading Strategy**:

1. **Fact Table (Properties)**: Core transactional property data
2. **Dimension Tables**: Location, Builder, Configuration, Amenities hierarchies
3. **Aggregate Tables**: Pre-calculated summary statistics for dashboard performance
4. **Audit Tables**: Data lineage and transformation metadata

**Performance Optimization**:

* **Cluster Scaling**: Auto-scaling Databricks clusters based on data volume
* **Partition Strategy**: Partitioned large tables by locality and date for efficient processing
* **Caching Strategy**: Cached frequently joined dimension tables in memory
* **Parallel Processing**: Leveraged Spark’s distributed computing for concurrent table processing
* **Connection Pooling**: Optimized Snowflake connections to prevent bottlenecks

**Performance Optimization**:

* Utilized Spark’s distributed computing capabilities
* Implemented efficient join strategies for large datasets
* Applied columnar storage formats (Delta Lake) for improved performance
* Used caching for frequently accessed datasets

### 

### Phase 5: Data Warehousing - Snowflake Gold Layer

**Technology Stack**: Snowflake, SQL, Star Schema Design

**Objective**: Create analytics-optimized data warehouse with dimensional modeling

**Schema Design**:

**Fact Table - Property Listings**:

* Primary repository for all transactional property data
* Contains measures: price, area, price\_per\_sqft, listing\_date
* Foreign keys to all dimension tables
* Optimized for aggregation queries

****

**Dimension Tables**:

1. **Location Dimension**:
   * Hierarchical location data (City → Area → Locality → Sub-locality)
   * Geographic coordinates and boundary information
   * Demographic and infrastructure attributes
2. **Builder Dimension**:
   * Builder/developer information
   * Company profiles and historical data
   * Project portfolio and ratings
3. **Property Type Dimension**:
   * Property classifications (Apartment, Villa, Plot, etc.)
   * Configuration details and property features
   * Age and construction quality indicators
4. **Configuration Dimension**:
   * Bedroom/bathroom configurations
   * Property size categories
   * Layout and design specifications
5. **Locality Dimension**:
   * Micro-location details
   * Amenities and infrastructure scores
   * Transportation connectivity metrics

**Warehouse Optimization**:

* Implemented clustered tables for query performance
* Created materialized views for common analytical patterns
* Established automated data quality monitoring
* Set up role-based access control for different user types

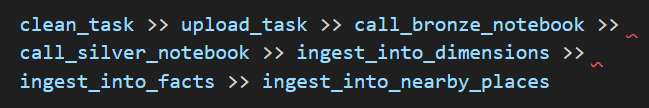
### Phase 6: Orchestration - Apache Airflow

**Technology Stack**: Apache Airflow, Python

**Objective**: Automate and monitor the entire data pipeline

**Workflow Orchestration**:

1. **DAG Structure**:



1. **Scheduling and Dependencies**:
   * It is a one time scheduled.
   * Task dependencies ensuring proper execution order
   * Conditional execution based on data availability
2. **Monitoring and Alerting**:
   * Real-time pipeline monitoring through Airflow UI
   * Email notifications for task failures or data quality issues
   * Performance metrics collection and reporting
3. **Error Handling**:
   * Comprehensive error handling at each pipeline stage
   * Automated retry mechanisms with exponential backoff
   * Data quality checkpoints preventing bad data propagation

### Phase 7: Business Intelligence - Power BI Dashboard

**Technology Stack**: Power BI, DAX, Snowflake Connector

**Objective**: Deliver actionable insights through interactive dashboards

**Dashboard Components**:

1. **Market Overview Dashboard**:
   * Total listings and market volume metrics
   * Average price trends over time
   * Geographic distribution of properties
   * Market share by builders and developers
2. **Locality Analysis Dashboard**:
   * Price per sqft comparison across localities
   * Locality-wise property distribution
   * Amenities availability heatmaps
   * Transportation connectivity scores
3. **Builder Performance Dashboard**:
   * Builder-wise listing volumes and pricing
   * Project delivery timelines and quality metrics
   * Market share and growth trends
   * Customer satisfaction indicators

**Advanced Analytics Features**:

* Dynamic filtering and cross-filtering capabilities
* Drill-down functionality from city to property level
* Real-time data refresh from Snowflake
* Mobile-responsive design for on-the-go access
* Export capabilities for detailed reporting

## 

## Technology Stack

| Layer | Technology | Purpose | Key Benefits |
| --- | --- | --- | --- |
| **Data Source** | Housing.com | Property listings platform | Comprehensive market coverage |
| **Web Scraping** | Apify | Automated data extraction | Scalable, reliable data collection |
| **Data Cleaning** | Python (Pandas, Regex) | Data preprocessing and standardization | Flexible data manipulation |
| **Staging (Bronze)** | AWS S3 | Raw data storage | Cost-effective, scalable storage |
| **Transformation (Silver)** | Databricks (Apache Spark) | Large-scale data processing | Distributed computing power |
| **Data Warehousing (Gold)** | Snowflake | Analytics-optimized storage | High-performance querying |
| **Orchestration** | Apache Airflow | Workflow automation | Reliable pipeline management |
| **Visualization** | Power BI | Business intelligence dashboards | Interactive analytics |

## 

## Challenges and Solutions

### Challenge 1: Data Volume and Velocity

**Problem**: Processing 12,000+ property listings with frequent updates

**Solution**: Implemented distributed processing with Apache Spark.

### Challenge 2: Data Quality and Consistency

**Problem**: Inconsistent data formats and missing values from web scraping

**Solution**: Developed comprehensive data cleaning pipelines with domain-specific validation rules

### Challenge 3: Real-time Analytics Requirements

**Problem**: Business users needed near real-time insights for decision making **Solution**: Implemented streaming data processing and optimized warehouse queries for sub-second response times

### Challenge 4: Scalability and Cost Management

**Problem**: Balancing performance requirements with cloud infrastructure costs **Solution**: Implemented auto-scaling clusters and intelligent data lifecycle management

## Conclusion

The End-to-End Real Estate Data Engineering and Analytics Pipeline represents a comprehensive solution that transforms raw property data into actionable business intelligence. By implementing modern data engineering practices and leveraging cloud-native technologies, the project delivers significant value to real estate stakeholders in the Hyderabad market.

The solution’s modular architecture ensures scalability and maintainability while providing reliable, high-performance analytics capabilities. The successful implementation demonstrates proficiency across the entire data engineering spectrum, from data acquisition through business intelligence visualization.

Key technical achievements include processing 12,000+ property listings through an automated pipeline, implementing dimensional data modeling for optimized analytics, and delivering interactive dashboards that enable data-driven decision making. The project serves as a strong foundation for expansion to additional markets and enhanced analytical capabilities.

The combination of modern cloud technologies, best-practice data engineering patterns, and comprehensive business intelligence capabilities positions this solution as a robust platform for real estate market analysis and investment decision support.