End-to-End ETL Pipeline Documentation

This document outlines the entire Extract, Transform, and Load (ETL) process implemented in the **`data_clean.py`**script. This single script prepares the raw luxury housing data and loads it into a MySQL database (`real_estate_db`) with two distinct tables: `luxury_housing_bangalore` and `microMarket_locations`.

1. Pipeline Summary

The pipeline is an integrated ETL process that avoids the need for an intermediate CSV file. It begins by ingesting the raw data, followed by **cleaning, standardization, and feature engineering** within a Pandas DataFrame. The script then dynamically creates the necessary database schema and performs high-speed, **batched SQL insertion** to populate the main housing table. Finally, it inserts pre-defined geographical coordinates into a separate locality table for spatial analysis.

2. Detailed Methods

2.1. Data Transformation and Standardization

All data cleaning and preparation steps happen in memory before loading to the database.

Column	Cleaning Action
Duplicate Removal	**Cleanup:** All duplicate records are identified and removed to ensure data integrity.
`Ticket_Price_Cr`	**Cleaned & Converted:** Removed non-numeric characters (₹, Cr, whitespace), converted to numeric (`float`), and rounded to 3 decimal places. **Handled Nulls:** Nan values (from conversion failures) were filled with `0`.

`Unit_Size_Sqft`	**Handled Erroneous Data:** Negative values were replaced with `0`. Null values were also replaced with `0`.
Text Fields	**Standardized:** `Micro_Market` and `Buyer_Type` were converted to **Title Case**. `Configuration` was converted to **UPPERCASE**. All were stripped of excess whitespace.
`Buyer_Comments`	**Handled Nulls:** Missing values were filled with an **empty string** (`""`).
`Amenity_Score`	**Imputation:** Missing values were imputed using the **global mean** of the existing, valid scores.
Validation	Infinite values created during division (e.g., in `Price_per_Sqft`) are identified and replaced with `0` before insertion.

2.2. Feature Engineering

The following calculated features were created to enhance the data's analytical utility:

- **`Booking_Status` (Categorical):** A binary flag derived from `Ticket_Price_Cr`. If `Ticket_Price_Cr > 0`, the status is set to **'Booked'**; otherwise, it is **'Not Booked'**.
- **`Price_per_Sqft` (Numeric):** TicketPrice /
- **`Quarter_Number` (Numeric):** Extracted the quarter number (1-4) from the `Purchase_Quarter` datetime object.
- **`BHK_Count` (Numeric):** Extracted the numeric BHK count from the standardized `Configuration` string using a regular expression.

2.3. Data Loading (ETL - Load Phase)

The pipeline dynamically creates the `real_estate_db` database and loads data into two separate tables:

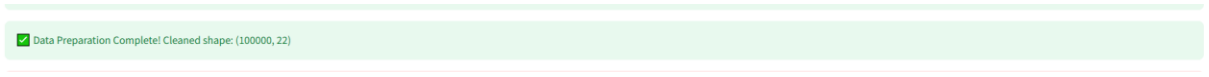
- **Phase 4A: Luxury Housing Data (`luxury_housing_bangalore` table)**
  - **Process:** The final cleaned DataFrame is streamed directly to MySQL.

  - **Method:** Insertion is performed using `cursor.executemany` with a configurable **batch size (default 1000)** for optimized network latency and performance during large data transfer.
- **Phase 4B: Geographical Data (`microMarket_locations` table)**
  - **Process:** A pre-defined list of **Locality coordinates** is inserted.
  - **Method:** The insertion uses an **`ON DUPLICATE KEY UPDATE`** clause (UPSERT) to ensure the table is updated if the script is run again, preventing primary key conflicts and ensuring data freshness.

3. Screenshot Evidence

Figure 1: Initial Data Audit and Missing Values

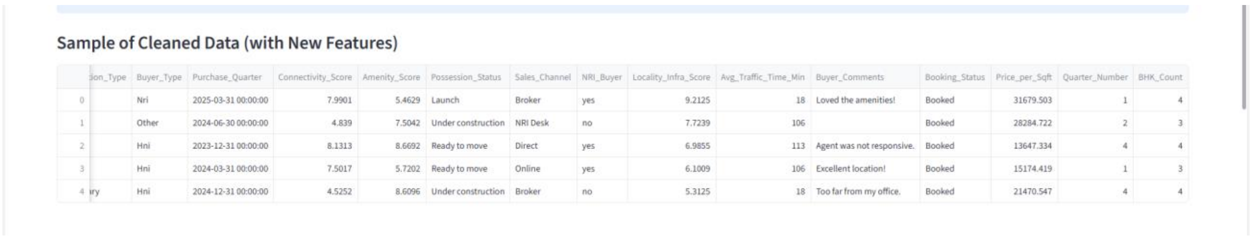
Content: Output from the beginning of the Python script showing: **Initial shape** (rows/columns). This validates the need for subsequent cleaning steps.



✅ Data Preparation Complete! Cleaned shape: (100000, 22)

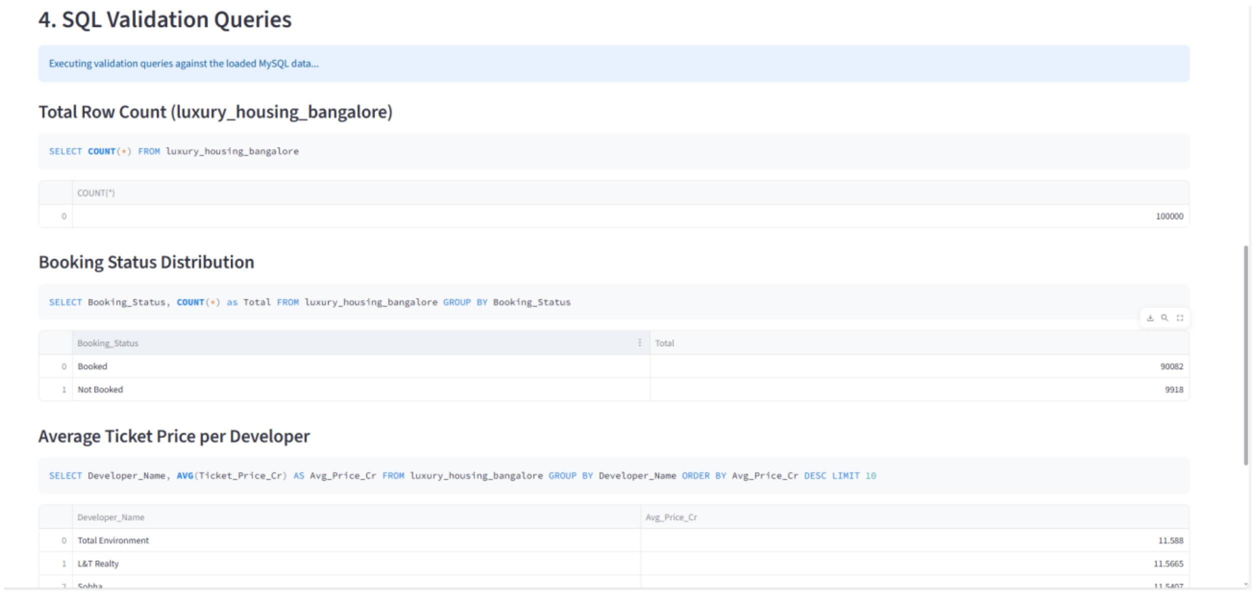
Figure 2: Final Data Structure

Content: A screenshot showing the output of a Pandas command (`df.head()` or similar) clearly displaying the new **calculated columns** such as `Price_per_Sqft` and `Booking_Status`, confirming feature engineering was successful.



Sample of Cleaned Data (with New Features)

	ion_Type	Buyer_Type	Purchase_Quarter	Connectivity_Score	Amenity_Score	Possession_Status	Sales_Channel	NRI_Buyer	Locality_Infra_Score	Avg_Traffic_Time_Min	Buyer_Comments	Booking_Status	Price_per_Sqft	Quarter_Number	BHK_Count
0		Nri	2025-03-31 00:00:00	7.9901	5.4629	Launch	Broker	yes	9.2125	18	Loved the amenities!	Booked	31679.503	1	4
1		Other	2024-06-30 00:00:00	4.839	7.5042	Under construction	NRI Desk	no	7.7239	106		Booked	28284.722	2	3
2		Hni	2023-12-31 00:00:00	8.1313	8.6692	Ready to move	Direct	yes	6.9855	113	Agent was not responsive.	Booked	13647.334	4	4
3		Hni	2024-03-31 00:00:00	7.5017	5.7202	Ready to move	Online	yes	6.1009	106	Excellent location!	Booked	15174.419	1	3
4	ary	Hni	2024-12-31 00:00:00	4.5252	8.6096	Under construction	Broker	no	5.3125	18	Too far from my office.	Booked	21470.547	4	4



4. SQL Validation Queries

Executing validation queries against the loaded MySQL data...

Total Row Count (luxury_housing_bangalore)

```
SELECT COUNT(*) FROM luxury_housing_bangalore
```

	COUNT(*)
0	100000

Booking Status Distribution

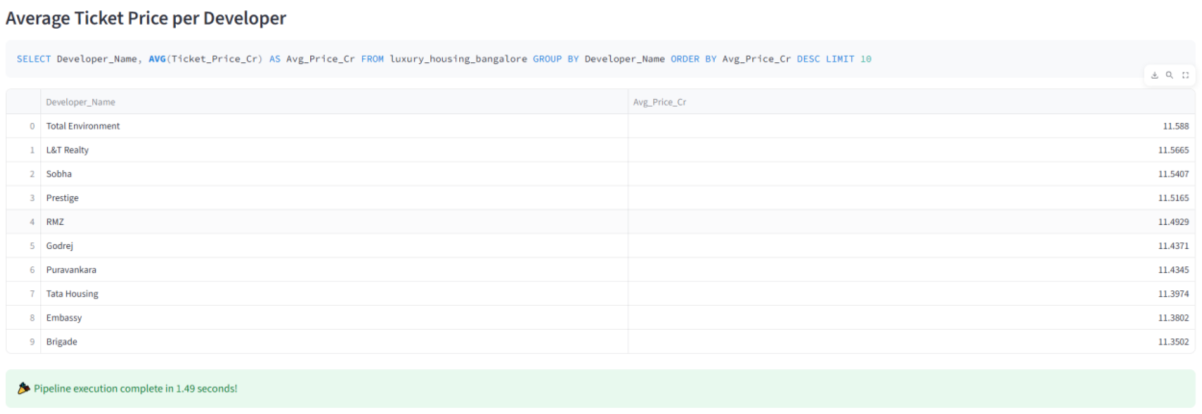
```
SELECT Booking_Status, COUNT(*) as Total FROM luxury_housing_bangalore GROUP BY Booking_Status
```

	Booking_Status	Total
0	Booked	90082
1	Not Booked	9918

Average Ticket Price per Developer

```
SELECT Developer_Name, AVG(Ticket_Price_Cr) AS Avg_Price_Cr FROM luxury_housing_bangalore GROUP BY Developer_Name ORDER BY Avg_Price_Cr DESC LIMIT 10
```

	Developer_Name	Avg_Price_Cr
0	Total Environment	11.588
1	L&T Realty	11.5665
2	Sobha	11.5407



Average Ticket Price per Developer

```
SELECT Developer_Name, AVG(Ticket_Price_Cr) AS Avg_Price_Cr FROM luxury_housing_bangalore GROUP BY Developer_Name ORDER BY Avg_Price_Cr DESC LIMIT 10
```

	Developer_Name	Avg_Price_Cr
0	Total Environment	11.588
1	L&T Realty	11.5665
2	Sobha	11.5407
3	Prestige	11.5165
4	RMZ	11.4929
5	Godrej	11.4371
6	Puravankara	11.4345
7	Tata Housing	11.3974
8	Embassy	11.3802
9	Brigade	11.3502

Pipeline execution complete in 1.49 seconds!