

AMES HOUSING PRICE PREDICTION

Phase 1: Data Acquisition & Exploration

Complete Educational Guide

Comprehensive cell-by-cell explanation with code breakdowns,
mathematical formulas, and Q&A for all team members

Document Generated: November 16, 2025 at 09:49

Coverage: Cells 1-28 (Data Acquisition Phase)

Notebook: Ames_Housing_Price_Prediction_EXECUTED.ipynb

Table of Contents

About the Dataset

SECTION 1: PROJECT INTRODUCTION

Cell 01: Project Title & Overview

Cell 03: Team Members

SECTION 2: ENVIRONMENT SETUP

Cell 08: Import Required Libraries

SECTION 3: DATA LOADING

Cell 10: Load Ames Housing Dataset

SECTION 4: INITIAL DATA EXPLORATION

Cell 12: Dataset Dimensions & Structure

Cell 14: Preview First Rows

Cell 14: Column Names & Data Types

SECTION 5: DATA QUALITY ASSESSMENT

Cell 16: Comprehensive Quality Checks

Cell 17: Schema Summary Table

SECTION 6: DATA DICTIONARY

Cell 18: Data Dictionary Cross-Reference

Cell 19: Load Data Dictionary

Cell 20: Key Features Overview

SECTION 7: SUMMARY STATISTICS

Cell 24: Comprehensive Summary Statistics

Cell 25: Understanding Summary Statistics (Educational)

SECTION 8: MISSING VALUE ANALYSIS

Cell 28: Calculate Missing Value Statistics

Cell 27: Understanding Missing Values (Educational)

About the Dataset

Ames Housing Dataset Overview

The Ames Housing dataset contains comprehensive information about residential properties sold in Ames, Iowa between 2006 and 2010. This dataset is widely used in machine learning and data science education due to its richness and real-world complexity.

Dataset Characteristics

- **Total Records:** 2,930 houses
- **Total Features:** 82 variables (80 predictors + 2 identifiers)
- **Target Variable:** SalePrice (final sale price in dollars)
- **Data Source:** Ames, Iowa Assessor's Office
- **Time Period:** 2006-2010

Feature Categories

The 82 features describe various aspects of residential homes:

- **Physical Characteristics:** Square footage, number of rooms, building type, architectural style
- **Quality & Condition:** Overall quality, material quality, condition ratings
- **Location:** Neighborhood, zoning classification, proximity to features
- **Age:** Year built, year remodeled
- **Lot Information:** Lot size, shape, contour, configuration
- **Basement:** Basement type, finish, quality, square footage
- **Garage:** Type, size, quality, year built
- **Utilities & Systems:** Heating, cooling, electrical, plumbing
- **External Features:** Pool, fence, deck, porch
- **Sale Details:** Sale type, condition, price

What We'll Cover in Phase 1

This document focuses on **Phase 1: Data Acquisition & Initial Exploration**, which includes:

- **Section 1:** Project Introduction & Setup
- **Section 2:** Importing Required Libraries
- **Section 3:** Loading the Dataset
- **Section 4:** Initial Data Exploration (shape, structure, columns)
- **Section 5:** Data Quality Assessment
- **Section 6:** Data Dictionary Reference
- **Section 7:** Summary Statistics
- **Section 8:** Missing Value Analysis
- **Section 9:** Educational Concepts (Statistics, Missing Values)

Why This Dataset?

The Ames Housing dataset offers several advantages for learning and practicing data science:

- **Real-World Complexity:** Contains missing values, outliers, and mixed data types - just like production data
- **Rich Features:** 82 variables provide many opportunities for feature engineering and selection
- **Documented:** Comprehensive data dictionary available with detailed feature descriptions
- **Challenging:** Requires thoughtful handling of categorical variables, missing data, and outliers
- **Practical:** House price prediction is relatable and has clear business value

Document Purpose: This guide explains every step of Phase 1 with code explanations, outputs, and Q&A to help team members understand the data acquisition and initial exploration process.

Cell 1 [MARKDOWN]

Ames Housing Price Prediction

Advanced Apex Project - Real Estate Price Modeling

A comprehensive machine learning approach to predicting residential property sale prices using multiple regression techniques and extensive feature engineering.

Cell 2 [MARKDOWN]

Project Information

Team: The Outliers

Course: Advanced Apex Project 1

Institution: BITS Pilani - Digital Campus

Academic Term: First Trimester 2025-26

Project Supervisor: Bharathi Dasari

Submission Date: November 2024

Cell 3 [MARKDOWN]

Team Members

Student Name BITS ID ----- -----	Anik Das 2025EM1100026	Adeetya Wadikar
2025EM1100384	Tushar Nishane 2025EM1100306	

Cell 4 [MARKDOWN]

Executive Summary

Problem Statement

Accurate real estate valuation is essential for buyers, sellers, and financial institutions. Traditional valuation methods can be subjective and time-consuming. This project develops machine learning models to predict house sale prices objectively based on property characteristics.

Business Objective

Develop a predictive regression model that estimates residential property sale prices with high accuracy. The model should help stakeholders:

- **Buyers:** Assess fair market value before purchase
- **Sellers:** Set competitive listing prices
- **Investors:** Identify undervalued properties
- **Lenders:** Support loan underwriting decisions

Dataset

Name: Ames Housing Dataset

Source: Kaggle (<https://www.kaggle.com/datasets/shashanknecrothapa/ames-housing-dataset>)

Size: 2,930 residential property sales transactions

Features: 82 variables describing:

- Physical characteristics (size, rooms, age)
- Quality ratings (construction, condition)
- Location attributes (neighborhood, zoning)
- Amenities (garage, basement, fireplace, pool)

Target Variable: SalePrice (in USD)

Time Period: Properties sold in Ames, Iowa from 2006-2010

Cell 5 [MARKDOWN]

Table of Contents

[Phase 1: Data Acquisition](#phase1)

1.1 [Environment Setup](#setup) 1.2 [Data Loading](#loading) 1.3 [Initial Data Inspection](#inspection) 1.4 [Schema Validation](#schema) 1.5 [Data Quality Assessment](#quality)

[Phase 2A: Data Preprocessing & Exploratory Analysis](#phase2a)

2.1 [Missing Value Analysis](#missing) 2.2 [Missing Value Treatment](#treatment) 2.3 [Univariate Analysis - Numerical](#univariate-num) 2.4 [Univariate Analysis - Categorical](#univariate-cat) 2.5 [Low-Variance Feature Removal](#lowvar) 2.6 [Bivariate Analysis - Correlations](#bivariate-corr) 2.7 [Bivariate Analysis - Visualizations](#bivariate-viz) 2.8 [Outlier Detection](#outliers)

[Phase 2B: Feature Engineering](#phase2b)

3.1 [Feature Creation](#creation) 3.2 [Feature Transformation](#transformation) 3.3 [Categorical Encoding](#encoding) 3.4 [Feature Importance](#importance)

[Phase 3: Model Development & Evaluation](#phase3)

4.1 [Data Preparation](#preparation) 4.2 [Simple Linear Regression](#simple-lr) 4.3 [Multiple Linear Regression](#multiple-lr) 4.4 [Model Comparison](#comparison) 4.5 [Conclusions & Recommendations](#conclusions)

Cell 6 [MARKDOWN]

Phase 1: Data Acquisition

Objective

Acquire the Ames Housing dataset and perform initial validation to ensure data integrity. This foundational phase establishes the quality and completeness of our data before proceeding to analysis.

Deliverables

- Successfully load dataset from CSV file
- Verify data structure and schema
- Conduct initial quality checks
- Document data characteristics and potential issues

Cell 7 [MARKDOWN]

1.1 Environment Setup

We import all necessary Python libraries for data manipulation, statistical analysis, visualization, and machine learning. Proper configuration ensures consistent behavior across different environments.

Cell 8 [CODE]

```
1 # Import core data manipulation libraries  
2 import pandas as pd  
3 import numpy as np  
4 import os  
5  
6 # Import visualization libraries  
7 import matplotlib.pyplot as plt  
8 import seaborn as sns  
9 import missingno as msno  
10  
11 # Import statistical libraries  
12 from scipy import stats  
13  
14 # Import machine learning libraries  
15 from sklearn.model_selection import train_test_split  
16 from sklearn.linear_model import LinearRegression  
17 from sklearn.preprocessing import LabelEncoder  
18 from sklearn.ensemble import RandomForestRegressor  
19 from sklearn.metrics import mean_squared_error, mean_absolute_error,  
r2_score
```

```
20  
  
21 # Configure environment  
  
22 import warnings  
  
23 warnings.filterwarnings('ignore')  
  
24  
  
25 # Set display options for better readability  
  
26 pd.set_option('display.max_columns', None)  
  
27 pd.set_option('display.max_rows', 100)  
  
28 pd.set_option('display.float_format', '{:.2f}'.format)  
  
29 pd.set_option('display.width', 1000)  
  
30  
  
31 # Set visualization defaults  
  
32 sns.set_style('whitegrid')  
  
33 plt.rcParams['figure.figsize'] = (12, 6)  
  
34 plt.rcParams['font.size'] = 10  
  
35  
  
36 # Print confirmation  
  
37 print("\u2713 All libraries imported successfully")  
  
38 print(f"\u2713 Pandas version: {pd.__version__}")  
  
39 print(f"\u2713 NumPy version: {np.__version__}")  
  
40 print(f"\u2713 Matplotlib version: {plt.matplotlib.__version__}")
```

```
41 print("\nEnvironment configured and ready for analysis.")
```

OUTPUT

- ✓ All libraries imported successfully
- ✓ Pandas version: 2.3.3
- ✓ NumPy version: 2.3.4
- ✓ Matplotlib version: 3.10.7

Environment configured and ready for analysis.

Cell 9 [MARKDOWN]

1.2 Data Loading

The Ames Housing dataset was downloaded from Kaggle and stored in the project's data directory. This dataset provides comprehensive information on residential properties sold in Ames, Iowa, making it an excellent resource for developing price prediction models.

Data Source: Kaggle - Ames Housing Dataset

Citation: Shashank Necrothapa. (n.d.). Ames Housing Dataset. Kaggle. <https://www.kaggle.com/datasets/shashanknecrothapa/ames-housing-dataset>

Cell 10 [CODE]**WHAT THIS CODE DOES**

Loads the Ames Housing dataset from a CSV file into a pandas DataFrame for analysis.

```
1 # Define the path to the dataset
2 data_path = "../data/AmesHousing.csv"
3
4 # Load the dataset into a pandas DataFrame
5 df = pd.read_csv(data_path)
6
7 # Display basic information
8 print("✓ Dataset loaded successfully!")
9 print(f"\nDataset Dimensions: {df.shape[0]} rows x {df.shape[1]} columns")
10 print(f"Memory Usage: {df.memory_usage(deep=True).sum() / 1024**2:.2f} MB")
11
12 # Display first few records
13 print("\nFirst 5 Records:")
14 df.head()
```

Code	Explanation
<code>data_path = '../data/ AmesHousing.csv'</code>	Sets the file path where our data is stored
<code>df = pd.read_csv(data_path)</code>	Reads CSV file and creates DataFrame (table structure)
<code>print('✓ Dataset loaded successfully')</code>	Confirms file was read without errors

🎯 WHY WE DO THIS

Can't analyze data without loading it first! CSV is standard format for tabular data.

📊 OUTPUT

✓ Dataset loaded successfully!

Dataset Dimensions: 2,930 rows × 82 columns

Memory Usage: 7.76 MB

First 5 Records:

```
Order      PID  MS SubClass MS Zoning Lot Frontage Lot Area Street
Alley Lot Shape Land Contour Utilities Lot Config Land Slope Neighborhood
Condition 1 Condition 2 Bldg Type House Style Overall Qual Overall Cond
Year Built Year Remod/Add Roof Style Roof Matl Exterior 1st Exterior 2nd Mas
Vnr Type Vnr Area Exter Qual Exter Cond Foundation Bsmt Qual Bsmt Cond
Bsmt E...
```

💡 What This Output Means:

Confirms the CSV file was found and loaded correctly into memory.

❓ COMMON QUESTIONS

Q: Why pandas instead of Excel?

A: Pandas handles large datasets better (1000s of rows), automates analysis, and integrates seamlessly with ML libraries.

Q: What if file path is wrong?

A: You'll get 'FileNotFoundException' - verify the path is correct relative to notebook location.

Q: What is a DataFrame?

A: A table-like structure (rows and columns) that makes data manipulation easy. Think Excel spreadsheet in Python.

Cell 11 [MARKDOWN]

1.3 Initial Data Inspection

Before conducting detailed analysis, we perform a high-level inspection to understand the dataset structure, identify data types, and spot any immediate quality concerns.

Cell 12 [CODE]**WHAT THIS CODE DOES**

Shows the dimensions of our dataset: how many houses (rows) and how many features (columns).

```
1 # Display comprehensive dataset information
2
3 print("Dataset Structure Overview:\n")
4
5 df.info()
6
7 print("\n" + "="*70)
8
9 print("Data Type Summary:")
10 print("="*70)
11
12 print(df.dtypes.value_counts())
13
14 print("\n" + "="*70)
15
16 print("Column Distribution:")
17 print("="*70)
18
19 print(f"Numerical columns (int64):"
20       f"\n{len(df.select_dtypes(include=['int64']).columns)}")
21
22 print(f"Numerical columns (float64):"
23       f"\n{len(df.select_dtypes(include=['float64']).columns)}")
24
25 print(f"Categorical columns (object):"
26       f"\n{len(df.select_dtypes(include=['object']).columns)}")
```

Code	Explanation
df.shape	Returns tuple: (number of rows, number of columns)

🎯 WHY WE DO THIS

Gives us a quick understanding of dataset size before diving into analysis.

📊 OUTPUT

Dataset Structure Overview:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 82 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Order              2930 non-null    int64  
 1   PID                2930 non-null    int64  
 2   MS SubClass        2930 non-null    int64  
 3   MS Zoning          2930 non-null    object  
 4   Lot Frontage       2440 non-null    float64 
 5   Lot Area           2930 non-null    int6... 
```

💡 What This Output Means:

(2930, 82) means 2,930 houses with 82 features each. That's 240,060 data points total!

❓ COMMON QUESTIONS

Q: Is 2,930 houses enough data?

A: Yes! For machine learning, 2,000+ samples is generally good. More data = better model accuracy.

Q: Why 82 features?

A: Each feature describes something about the house (size, quality, location, etc.). More features can improve predictions but also add complexity.

Cell 13 [MARKDOWN]

1.4 Schema Validation

We verify that all expected columns are present and properly formatted. This schema validation ensures data integrity and helps identify any structural anomalies early in the process.

Cell 14 [CODE]**WHAT THIS CODE DOES**

Shows the first 5 rows of the dataset to see what the actual data looks like.

```
1 # Display all column names
2
3 print(f"Total Features: {len(df.columns)}\n")
4
5
6 # Print in organized format (4 columns)
7
8 col_list = df.columns.tolist()
9
10 for i in range(0, len(col_list), 4):
11
12     row = col_list[i:i+4]
13
14     print(f"{i+1:2d}-{i+len(row):2d}: " + " | ".join(f"{col:20s}" for
15         col in row))
16
17
18 print("\n" + "="*70)
19
20 print("Key Columns Verified:")
21
22 print("=*70")
23
24
25 important_cols = ['Order', 'PID', 'SalePrice', 'Gr Liv Area', 'Overall Qual',
26 'Neighborhood']
```

```

16 for col in important_cols:
17     status = "✓" if col in df.columns else "✗"
18     print(f"{status} {col}")

```

Code	Explanation
df.head()	Returns first 5 rows by default (can specify different number)

⌚ WHY WE DO THIS

Lets us visually inspect data structure, column names, and sample values before processing.

📊 OUTPUT

Total Features: 82

All Column Names:

1- 4: Order MS Zoning	PID	MS SubClass	
5- 8: Lot Frontage Alley	Lot Area	Street	
9-12: Lot Shape Lot Config	Land Contour	Utilities	
13-16: Land Slope Condition 2	Neighborhood	Condition 1	
17-...			

💡 What This Output Means:

A table showing 5 houses with all their features. Each row = one house. Each column = one attribute.

❓ COMMON QUESTIONS

Q: Why only 5 rows?

A: Gives a quick preview without overwhelming output. For deeper inspection, use head(20) or head(50).

Q: What do column names mean?

A: They describe house attributes: 'Gr Liv Area' = above ground living area, 'Sale Price' = final price, etc.

Cell 15 [MARKDOWN]

1.5 Data Quality Assessment

We conduct initial quality checks to identify missing values, duplicate records, and verify the target variable integrity.

Cell 16 [CODE]**WHAT THIS CODE DOES**

Lists all 82 column (feature) names in our dataset.

```
1 # Perform comprehensive quality checks
2
3 print("Data Quality Assessment:")
4
5 print("*"*70)
6
7 # Check for missing values
8
9 total_missing = df.isnull().sum().sum()
10
11 cols_with_missing = df.isnull().any().sum()
12
13 print(f"\nMissing Value Check:")
14
15 print(f" Total missing values: {total_missing},")
16
17 print(f" Columns with missing data: {cols_with_missing} out of
{len(df.columns)}")
18
19
20 # Check for duplicates
21
22 duplicates = df.duplicated().sum()
23
24 print(f"\nDuplicate Check:")
25
26 print(f" Duplicate rows: {duplicates}")
27
28 if duplicates == 0:
```

```

17     print(" ✓ No duplicates found")

18

19 # Verify target variable

20 print(f"\nTarget Variable (SalePrice) Verification:")

21 print(f" Missing values: {df['SalePrice'].isnull().sum()}" )

22 print(f" Minimum: ${df['SalePrice'].min():,.2f}")

23 print(f" Maximum: ${df['SalePrice'].max():,.2f}")

24 print(f" Mean: ${df['SalePrice'].mean():,.2f}")

25 print(f" Median: ${df['SalePrice'].median():,.2f}")

26 print(f" Standard Deviation: ${df['SalePrice'].std():,.2f}")

27

28 print("*"*70)

```

Code	Explanation
df.columns.tolist()	Extracts column names and converts to a list

🎯 WHY WE DO THIS

Need to know what features are available before selecting which ones to use for predictions.

📊 OUTPUT

Data Quality Assessment:

=====

Missing Value Check:

Total missing values: 15,749

Columns with missing data: 27 out of 82

```
Duplicate Check:  
  Duplicate rows: 0  
  ✓ No duplicates found
```

```
Target Variable (SalePrice) Verification:  
  Missing values: 0  
  Minimum: $12,789  
  Maximum: $755,000  
  Mean: $180,796.06  
  Median: $160,000.00  
  Standard Deviation: $79,886.69
```

What This Output Means:

Complete inventory of all available features, from 'Order' to 'SalePrice'.

? COMMON QUESTIONS

Q: Do we use all 82 features?

A: No! Later we'll remove irrelevant ones (like 'Order' which is just an ID number) and handle missing values.

Q: How to remember what each means?

A: The data dictionary (in Phase 1) explains each feature. We'll focus on the most important ones.

Cell 17 [CODE]

```
1 # Create detailed schema summary table

2 schema_summary = pd.DataFrame({

3     'Column': df.columns,

4     'Data_Type': df.dtypes.values,

5     'Non_Null_Count': df.count().values,

6     'Null_Count': df.isnull().sum().values,

7     'Null_Percentage': (df.isnull().sum() / len(df) * 100).values,

8     'Unique_Values': [df[col].nunique() for col in df.columns]

9 })

10

11 # Sort by null percentage to see problematic columns first

12 schema_summary = schema_summary.sort_values('Null_Percentage',
ascending=False)

13

14 print("Schema Summary (Top 20 columns by missing data):")

15 print("*"*90)

16 schema_summary.head(20)
```

 **OUTPUT**

Schema Summary (Top 20 columns by missing data):

Unique_Values	Column	Data_Type	Non_Null_Count	Null_Count	Null_Percentage
73	PoolQC	object		13	2917
99.56			4		
75	MiscFeature	object		106	2824
96.38			5		
7	Alley	object		198	2732
93.24			2		
74	...				

Cell 18 [MARKDOWN]**1.5.1 Data Dictionary Cross-Reference**

We attempt to load the official data dictionary to cross-reference feature definitions and ensure our understanding aligns with the dataset documentation.

Cell 19 [CODE]

```
1 # Attempt to load the data dictionary
2
3     data_dict_path = "../docs/data_dictionary.xlsx"
4
5     data_dict = pd.read_excel(data_dict_path)
6
7     print(f"\u2708 Data dictionary loaded successfully")
8
9     print(f"  Total feature descriptions: {len(data_dict)}")
10
11    print(f"\nFirst 10 Feature Definitions:")
12
13    print("-"*70)
14
15    print(data_dict.head(10))
16
17
18 except FileNotFoundError:
19
20     print("i Data dictionary file not found at expected location")
21
22     print("  This is not critical - proceeding with dataset analysis")
23
24     print(f"  Expected path: {data_dict_path}")
25
26 except Exception as e:
27
28     print(f"i Could not load data dictionary: {str(e)}")
29
30
31     print("  Proceeding with dataset analysis")
```

 **OUTPUT**

- ✓ Data dictionary loaded successfully
- Total feature descriptions: 82

First 10 Feature Definitions:

	Feature	Data Type	Description
Primary Key (Yes/No)			
0	Order	int64	Observation number (sequential identifier for ... Yes)
1	PID	int64	Parcel Identification Number (unique property ... Yes)
2	MS SubClass	int64	Identifi...

Cell 20 [MARKDOWN]

Data Dictionary - Key Features

While a separate data dictionary file is not included, we document all critical features here for transparency and reproducibility.

Target Variable

| Feature | Description | Type | Range |-----|-----|-----|-----| | **SalePrice** | Property sale price in USD | Continuous | \$34,900 - \$755,000 |

Top Predictors (by correlation with SalePrice)

| Feature | Description | Type | Range/Values |-----|-----|-----|-----| | **Overall Qual** | Overall material and finish quality | Ordinal | 1-10 scale | | **Gr Liv Area** | Above grade living area | Continuous | 334 - 5,642 sq ft | | **Garage Cars** | Garage capacity | Discrete | 0-4 cars | | **Garage Area** | Garage size | Continuous | 0 - 1,418 sq ft | | **Total Bsmt SF** | Total basement area | Continuous | 0 - 6,110 sq ft | | **1st Flr SF** | First floor area | Continuous | 334 - 4,692 sq ft | | **Year Built** | Original construction year | Discrete | 1872 - 2010 | | **Full Bath** | Full bathrooms above grade | Discrete | 0-3 | | **Tot Rms AbvGrd** | Total rooms above grade | Discrete | 2-14 |

Feature Categories (82 total features)

1. **Physical Attributes** (28 features) - Size measurements: Living area, lot size, rooms - Floor areas: Basement, 1st floor, 2nd floor - Room counts: Bedrooms, bathrooms, total rooms
2. **Quality & Condition Ratings** (11 features) - Overall Quality (1-10) - Overall Condition (1-10) - Kitchen Quality, Basement Quality - External Quality, Heating Quality
3. **Location Features** (8 features) - Neighborhood (25 categories) - MS Zoning (5 categories) - Lot Configuration (5 categories)
4. **Amenities & Features** (35 features) - Garage: Type, finish, cars, area - Basement: Type, finish, area, bathrooms - Fireplace: Count, quality - Pool: Area, quality - Porch: Type, area

Data Sources

- **Primary Source:** Ames, Iowa Assessor's Office
- **Collection Period:** 2006-2010

- **Dataset:** Available on Kaggle - [Ames Housing Dataset](<https://www.kaggle.com/datasets/shashanknecrothapa/ames-housing-dataset>)
- **Original Research:** Dean De Cock (2011) - "Ames, Iowa: Alternative to the Boston Housing Data Set"

Feature Engineering Note

Additional features created during preprocessing:

- **Total_Bathrooms:** Sum of all bathroom types
- **Total_Porch_SF:** Combined porch areas
- **House_Age:** Years since construction
- **Years_Since_Remod:** Time since last remodel
- **Total_SF:** Combined living space

Documentation Philosophy

All features are documented through:

- Inline markdown explanations throughout this notebook
- Feature importance analysis (Section 3.4)
- Correlation analysis (Section 2.3)
- Statistical summaries (Section 2.1)
- Original Kaggle dataset documentation

This embedded documentation ensures **transparency** and **reproducibility** of our analysis without requiring external files.

Cell 21 [MARKDOWN]

Phase 1 Summary

Accomplishments

✓ Environment Configured

- All required libraries imported successfully
- Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn ready
- Display settings optimized for analysis

✓ Dataset Successfully Loaded

- **Source:** Ames Housing Dataset from Kaggle
- **Size:** 2,930 residential property records
- **Features:** 82 variables (28 int, 11 float, 43 categorical)
- **Memory:** ~2MB dataset size
- **Target:** SalePrice (range: \$12,789 - \$755,000)

✓ Data Quality Verified

- Schema matches expectations (82 columns present)
- No duplicate records identified
- Target variable has no missing values
- 27 features contain missing values (to be addressed in Phase 2)

✓ Initial Observations

- Mix of numerical and categorical features
- Some features have high missingness (>50%) - candidates for removal
- Price range suggests diverse property types
- Data appears well-structured and ready for analysis

Next Steps

Proceed to **Phase 2A: Data Preprocessing & Exploratory Analysis** where we will:

- Conduct comprehensive missing value analysis
- Implement systematic data cleaning procedures
- Perform univariate and bivariate analysis
- Identify and handle outliers
- Prepare data for feature engineering

Cell 22 [MARKDOWN]

Phase 2A: Data Preprocessing & Exploratory Analysis

Objective

Transform raw data into a clean, analysis-ready format through systematic preprocessing. Conduct comprehensive exploratory analysis to understand variable distributions, relationships, and data quality issues.

Key Activities

- Systematic missing value analysis and treatment
- Univariate analysis of all features
- Bivariate analysis to identify price predictors
- Low-variance feature identification and removal
- Outlier detection and assessment

Cell 23 [MARKDOWN]

2.1 Summary Statistics Overview

Before diving into detailed analysis, we establish a quantitative foundation by computing comprehensive descriptive statistics for all numerical features.

Objectives:

- Understand central tendency (mean, median)
- Measure spread and variability (std, IQR)
- Identify range boundaries (min, max)
- Detect potential data quality issues

This statistical overview guides our subsequent preprocessing decisions.

Cell 24 [CODE]**WHAT THIS CODE DOES**

Comprehensive summary statistics broken down by data type and displayed in organized sections.

```
1 # =====
2 # COMPREHENSIVE SUMMARY STATISTICS
3 # =====
4 print("=*70)
5 print("SUMMARY STATISTICS - NUMERICAL FEATURES")
6 print("=*70)
7 print("\nDescriptive Statistics for All Numerical Features:")
8 print(df.describe())
9
10 print("\n" + "=*70)
11 print("SUMMARY STATISTICS - TARGET VARIABLE (SalePrice)")
12 print("=*70)
13 target_stats = df['SalePrice'].describe()
14 print(target_stats)
15 print(f"\nPrice Range: ${df['SalePrice'].min():,.0f} to ${df['SalePrice'].max():,.0f}")
```

```
16 print(f"Price Spread (IQR): ${target_stats['75%'] - target_stats['25%']:.0f}")

17

18 # Key insights from statistics

19 print("\n" + "="*70)

20 print("KEY STATISTICAL INSIGHTS")

21 print("=*70)

22 print(f"1. SalePrice Distribution:")

23 print(f"    - Mean: ${df['SalePrice'].mean():,.0f}")

24 print(f"    - Median: ${df['SalePrice'].median():,.0f}")

25 print(f"    - Shows {'right' if df['SalePrice'].mean() > df['SalePrice'].median() else 'left'}-skewed distribution")

26 print(f"\n2. Living Area Variability:")

27 print(f"    - Range: {df['Gr Liv Area'].min():.0f} to {df['Gr Liv Area'].max():.0f} sq ft")

28 print(f"    - Coefficient of Variation: {(df['Gr Liv Area'].std()/df['Gr Liv Area'].mean())*100:.1f}%")

29 print(f"\n3. Age Distribution:")

30 print(f"    - Newest: {df['Year Built'].max()}")

31 print(f"    - Oldest: {df['Year Built'].min()}")

32 print(f"    - Span: {df['Year Built'].max() - df['Year Built'].min()} years")

33 print("\n\n Statistical foundation established for analysis")
```

Code	Explanation
<code>df.describe()</code>	Calculates statistics for numerical features
<code>df.describe(include=['object'])</code>	Calculates statistics for categorical (text) features

🎯 WHY WE DO THIS

Provides detailed statistical overview separated by numerical vs categorical features for better understanding.

📊 OUTPUT

```
=====SUMMARY=====
STATISTICS - NUMERICAL
FEATURES=====Descriptive
Statistics for All Numerical Features:          Order
PID ... Yr Sold     SalePrice count  2930.00000  2.930000e+03 ...
2930.000000  2930.000000 mean    1465.50000  7.144645e+08 ...
180796.060068 std   845.96247  1.887308e+08 ...      1.316613
79886.692357 min   1.000... .
```

💡 What This Output Means:

Shows statistical summaries for both numerical features (mean, std, quartiles) and categorical features (count, unique values, top value, frequency).

❓ COMMON QUESTIONS

Q: What's the difference between numerical and categorical stats?

A: Numerical: mean, std dev make sense. Categorical: we see most common values and how many unique categories exist.

Q: Why separate them?

A: Different types of features need different analysis methods. Can't calculate 'average' of text categories!

Cell 25 [EDUCATIONAL]: 🎓 Understanding Summary Statistics**📘 WHAT IS THIS?**

Summary statistics are numerical measures that describe the main characteristics of a dataset in a concise way.

🎯 WHY DO WE NEED THIS?

- Get a 'bird's eye view' of the data without looking at every single value
- Quickly spot unusual patterns or potential problems
- Compare different features on the same scale
- Make informed decisions about data cleaning and preprocessing

📐 Mean (Average)

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

In Plain English: Add all values together, then divide by how many values you have.

\bar{x} Mean (average value)

n Total number of values

x_i Each individual value

Σ Sum of all values

1 2
3 4 **Example Calculation:**

Data: House prices: \$150K, \$160K, \$170K, \$180K, \$190K

Calculation:

$$(\$150K + \$160K + \$170K + \$180K + \$190K) \div 5 = \$850K \div 5 = \$170K$$

Result: Mean price = \$170K

 **Standard Deviation**

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$

In Plain English: Measures how spread out the values are from the average. High = lots of variation, Low = values are similar.

σ Standard deviation (spread)

n Total number of values

x_i Each individual value

\bar{x} Mean value

1 2
3 4 **Example Calculation:**

Data: Prices: \$150K, \$160K, \$170K, \$180K, \$190K (Mean = \$170K)

Calculation:

Step 1: Find differences from mean: -20, -10, 0, 10, 20

Step 2: Square them: 400, 100, 0, 100, 400

Step 3: Average: $(400+100+0+100+400) \div 5 = 200$

Step 4: Square root: $\sqrt{200} \approx 14.14$

Result: Standard Deviation $\approx \$14,140$

Real-World Meaning

For Ames Housing: We see that SalePrice has mean ~\$180K and std ~\$80K. This tells us most houses are between \$100K-\$260K, with some outliers.

Quartiles help too: Q1=\$130K, Q2(median)=\$160K, Q3=\$213K means 50% of houses cost between \$130K-\$213K.

Cell 26 [MARKDOWN]

2.1 Missing Value Analysis

Missing data is common in real-world datasets. We systematically analyze missing value patterns to develop an appropriate treatment strategy.

Cell 27 [EDUCATIONAL]: Understanding Missing Values

WHAT IS THIS?

Missing values (also called NaN, null, or empty cells) are data points that weren't recorded or don't apply.

WHY DO WE NEED THIS?

- Machine learning models cannot handle missing values - they'll throw errors
- Missing data can introduce bias if not handled correctly
- Understanding WHY data is missing helps choose the right solution
- Large amounts of missing data might indicate a useless feature

MCAR (Missing Completely At Random)

Explanation: Data is missing for random reasons, unrelated to the data itself.

Example: Survey responses lost due to computer glitch - happens randomly to any respondent.

Solution: Safe to impute (fill with mean/median) or delete rows - no bias introduced.

MAR (Missing At Random)

Explanation: Missingness is related to other observed variables, but not the missing value itself.

Example: Older houses more likely to have missing 'Pool Quality' because pools weren't common back then.

Solution: Use group-based imputation (e.g., fill based on house age).

MNAR (Missing Not At Random)

Explanation: Missingness is related to the value itself.

Example: High-income people less likely to report income - the missing value IS related to the actual income.

Solution: Most challenging - may need advanced techniques or accept bias.

✓ Our Strategy

Step 1: Drop columns with >50% missing (too little data to trust)

Step 2: For categorical: impute with 'None' or 'Missing' category

Step 3: For numerical: impute with 0, mean, or median depending on feature

Step 4: Special case (Lot Frontage): use neighborhood median (grouped imputation)

Condition	Action	Reasoning
Missing > 50%	DROP the column entirely	Not enough data to be useful
Missing < 5%	IMPUTE safely	So few missing that method doesn't matter much
Missing 5-50%	ANALYZE carefully	Understand why it's missing before choosing method

Cell 28 [CODE]

```
1 # Calculate missing value statistics

2 missing_counts = df.isnull().sum()

3 missing_pct = (missing_counts / len(df)) * 100

4

5 missing_df = pd.DataFrame({ 

6     'Feature': missing_counts.index, 

7     'Missing_Count': missing_counts.values, 

8     'Missing_Percentage': missing_pct.values

9 })

10

11 # Filter to only features with missing values

12 missing_df = missing_df[missing_df['Missing_Count'] > 0]

13 missing_df = missing_df.sort_values('Missing_Percentage', ascending=False)

14

15 print(f"Features with Missing Values: {len(missing_df)} out of {len(df.columns)}")

16 print("\nTop 15 Features with Most Missing Data:")

17 print("=*70")

18 missing_df.head(15)
```

OUTPUT

Features with Missing Values: 27 out of 82

Top 15 Features with Most Missing Data:

	Feature	Missing_Count	Missing_Percentage
73	Pool QC	2917	99.56
75	Misc Feature	2824	96.38
7	Alley	2732	93.24
74	Fence	2358	80.48
26	Mas Vnr Type	1775	60.58
58	Fireplace Qu ...		