Redfin Housing Data for Business Insights

TEAM MEMBERS

Vasavi Surisetty - 11766524

Pooja Kondagoli - 11736183

Sindhu Putumbaka - 11635881

Dinakar Narahari - 11725613

TEAM DATA MINDERS

[BCIS 5110 SECTION 003 - PROGRAMMING LANGUAGES FOR BUSINESS ANALYTICS]

[ U N I V E R S I T Y O F N O R T H T E X A S ] [ 1 1 5 5 U N I O N C I R , D E N T O N , T X 7 6 2 0 5 ]

**Contents**

1. Project Members & Contribution

2. Executive summary

3. Project motivation/background

3. Problem Statement

4. Data description

4. Data transformation

5. Models and analysis

6. Findings & Managerial Implications

6. Conclusions

7. Appendix

8. References

**Project Members and Contributions**  
  
Vasavi Surisetty, Dinakar Narahari – Data collection, Data cleaning, Conclusion  
Pooja Kondagoli, Sindhu Putumbaka – Data analysis, Visualization

**Executive Summary**  
  
The project intended to bring out real estate data from Redfin using pandas packages in python. The given dataset, "Redfin\_housing\_data.csv" had a lot of qualities, for instance, price, features, and trends of the stipulated market. By exploiting data gathering, visualization, and analysis tools, the team tried to come up with the conclusion about changes in the housing market conditions, what trends influence prices and what factors play the most significant role when it comes to property value.

**Project Motivation/Background**  
  
The project's ideology was born from the rising need of people to find out how property prices are influenced and what trends in the housing are currently underway. Upon the acquisition of abundant housing data, from platforms like Redfin, we as a team tried to exploit this database for extracting revelations regarding the housing market mechanism, and they especially wanted to focus on geographical factors, property features and real estate trends.  
  
Fundamentally, the project endeavored to demonstrate that data-based strategies might help a better understanding of the challenges faced through the real estate market, with pandas data analysis and exploration as a particular tool.

The project aims to:

* Illustrate housing market basics including supply and demand factors, pricing patterns, and some of the differences between regions of interest.
* Bettering predictive power in property valuation via attributes like positions, house characteristics, and market conditions analysis.
* Amp stakeholders up to data-based decision skills, those for property acquisition, property disposal, and property investment.

**Problem Statement**

The real estate market, because of its complexity and movement is hard situation to handle. Stakeholders, for the most part, have not been able to acquire detailed information about the market trends, thus can calculate property valuation with good accuracy, and they also find themselves in the difficulty of making well-informed decisions. A common fact is that the common approach is not only extensive, but also short on details.

**Data Description:**

The dataset is called "Redfin\_housing\_dat\_2023" and it is formatted as a CSV file. The table consists of 98673 rows and 58 columns. Redfin housing data gives comprehensive data on home prices and amenities by region throughout the United States. The set of data, which is on real estate, contains property prices, approximately square footage, location, sale history, and other descriptive features of the property. Given the thoroughness of its scope and fresh data, the Redfin housing data is a beneficial source for individuals, real estate practitioners, and researchers in reference to the housing industry thereby enabling them to make qualified decisions.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| **Period\_begin/Period\_end/Period\_duration** | **Date** | Identifies how long the data is spanning, which are its start date, end date, and duration. |
| **Region/State/State\_code** | **String** | The region code along with the country code acts as a geographical identifier. |
| **property\_type/property\_type\_id** | **String** | Categorizes properties and assign unique value to each type. |
| **median\_sale\_price** | **Number** | The datapoint shows the middle price that established a property for the addressed region and time period. |
| **median\_list\_price** | **Number** | Median asking price (listing price) of properties within the designated area and period of time. |
| **median\_ppsf** | **Number** | Median price for each foot of properties. |
| **homes\_sold** | **Number** | The number of homes sold over the selected area and time period. |
| **pending\_sales** | **Number** | The number of pending sales subject to selected area for the designated period. |
| **new\_listings** | **Number** | The count of new properties being advertised in the designated area and the stipulated period. |
| **Inventory** | **Number** | Total amount of properties offered for sale within the admitted locations and time frame. |
| **months\_of\_supply** | **Number** | Prognosticated number of months that would have elapsed if we are selling the present inventory according to current pace of sales |
| **median\_dom** | **Number** | The average keeps time of properties in real estate market. |
| **avg\_sale\_to\_list** | **Number** | Sold price to list price ratio for properties. |
| **sold\_above\_list** | **Number** | Fee amount of the sale transaction that goes above the original price. |
| **price\_drops** | **Number** | It is a good representation of all the cases where prices of properties reduced between the specified period. |
| **off\_market\_in\_two\_weeks** | **Number** | The percentage of which go off-market within the two weeks following a listing. |
| **homes\_sold** | **Number** | **Is** a representation of the total number of home sales for the stated period. |

**Data transformation.**

Data transformation is a necessary step for preparing the Redfin housing dataset, concentrated on 2023 data, for analysis. It involves feature engineering, creating or modifying features; normalization for consistent scaling; encoding categorical variables; and dealing for missing values through imputation or removal. Date/time specification is standardized, and data processing is done as needed. These transformations enable the dataset to be ideal for discovering the trends in the housing market in 2023.

Below are the steps used in data transformation:

1. To extract for housing data for the year 2023 from the Redfin housing dataset, the file must be first read into the Jupyter notebook environment. With pandas, the dataset gets loaded and then filtered such that the records in it are from 2023 only using the period begin column. Changed the period begin column into datetime format using .to\_datatime and then filter the data using df[‘period\_begin’].dt.year ==2023 and then saved in temporary dataframe df\_2023.

A screenshot of a computer code

Description automatically generated

1. Identified the null values in each column by using .isnull() and .sum() is used to calculate the total null values in each column.

A screenshot of a computer

Description automatically generated

1. City column is drop as it is having complete null values using df\_2023.drop(columns=[‘city’].

A computer screen shot of a computer code

Description automatically generated

1. Calculated the missing values percentage. First, we have calculate the total 2023 records using df\_2023.shape[0]\*df\_2023.shape[1] and using (Total\_missing\_vales/Total\_2023\_records)\*100 the percentage is calculated.

A screenshot of a computer code

Description automatically generated

1. Identifying and handling null values is a critical step in data analysis. Below options are used to address null values:

* Replaced median sale price with the overall mean value using .fillna function. From the df\_2023, median sale price column is extracted and the null value in that column is filled by the mean of that column using .mean() function.

df\_2023[‘median\_sale\_price’].fillna(df\_2023[‘median\_sale\_price’].mean()

Similarly for median sale price mom and yoy the above code is used to fill the null value.

A screenshot of a computer

Description automatically generated

* Replaced median list price with the overall mean value. From the df\_2023, median list price column is extracted and the null value in that column is filled by the mean of that column using .mean() function.

df\_2023[‘median\_list\_price’].fillna(df\_2023[‘median\_list\_price’].mean() Similarly for median list price mom and yoy the above code is used to fill the null value.

A screen shot of a computer code

Description automatically generated

* Replaced median ppsf (price per square feet) price missing values by mean of each property type. First all the required columns are converted to numeric datatype using .to\_numeric. Using for loop, for each unique property type the mean is calculated and the mean value is filled into the null value.

df\_2023.loc[df\_2023['property\_type'] == property\_type, numeric\_cols] = df\_2023.loc[df\_2023['property\_type'] == property\_type, numeric\_cols].fillna(property\_means)

A screenshot of a computer code

Description automatically generated

* Replaced median list ppsf (price per square feet) price missing values by mean of each property type. First all the required columns are converted to numeric datatype using .to\_numeric. Using for loop, for each unique property type the mean is calculated and the mean value is filled into the null value.

df\_2023.loc[df\_2023['property\_type'] == property\_type, numeric\_cols] = df\_2023.loc[df\_2023['property\_type'] == property\_type,numeric\_cols].fillna(property\_means)

A screenshot of a computer

Description automatically generated

* Replaced home sold missing values by mean value of each month. The "Period Begin" column is converted to datetime format with `.to\_datetime()`, extracting the month. Using `.groupby()`, the dataset is grouped by month, and the mean is calculated for analysis. Further using for loop, the null value is replaced by mean value.

for month, mean\_homes\_sold in monthly\_means.items():

    df\_2023.loc[df\_2023['month'] == month, 'homes\_sold'] = df\_2023.loc[df\_2023['month'] == month, 'homes\_sold'].fillna(mean\_homes\_sold)

A screenshot of a computer code

Description automatically generated

1. Verified the missing value count.

A screenshot of a computer code

Description automatically generated

**Data Analysis:**  
  
Evaluating the Redfin housing data implies considering masses of data dealing with the housing business. Using statistical techniques and data visualization tools, we are able to observe relationships, trends, and comparisons that help us to uncover more about the dataset. Through this analysis we could perceive market dynamics changes and use this data for smart decisions related to real estate.

A screenshot of a computer code

Description automatically generated

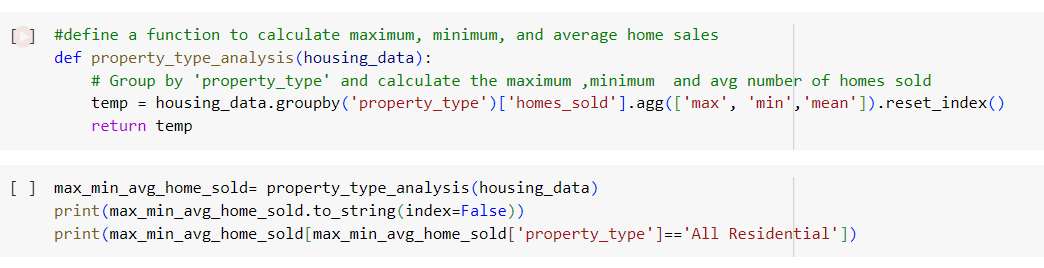
**Pandas:** Pandas is a robust Python library that allows for the manipulation and analysis of data. It does that using DataFrame and Series data structures as well as cleaning, filtering and transforming data functions. One of the favored tools for data science and machine learning tasks like data preprocessing, exploration, and visualization is pandas.

**Matplotlib**: Matplotlib is a modular library for plotting both static, animated, and interactive visualizations in Python. It has an extensive set of graphing functions that can be used for constructing line graphs, scatter charts, bar graphs, and histograms among others. Through Matplotlib's wide-spread applicability in data exploration and data visualization, it’s been seen to be useful in areas like sciences, engineering, and finance.

**NumPy**: NumPy is an important library for the numerical computing domain of Python, it gives support for multi-dimensional and mathematical arrays and functions. It empowers an effective use of the largescale datasets and works well for the scientific computing, data analysis, and machine learning. Numpy's multidimensional arrays through vectorized operations improve their performance and augment their general implement ability for mathematical computations.

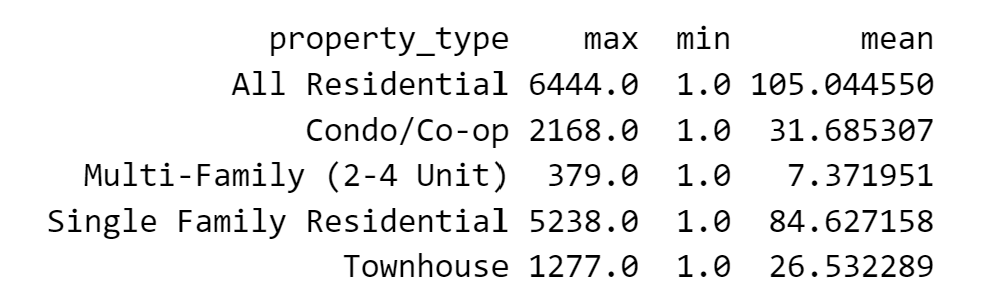
For analysing the dataset, we have imported Pandas, Matplotlib, Numpy and calendar libraries. The ‘Redfin housing’ dataset is read as a CSV file using pandas library.

Q1. In what way are the maximum, minimum, and average home sales vary across different property types?



The above code determines maximum, minimum, and average home sale values for each property type. The `Property\_type\_analysis` function is defined to analyze the data and to classify property types in the data with their home sold vales. This function analyze the data sets which contain property types and the number of houses sold as the input. Groupby function is used to group the data by ‘property\_type’ column, this step enables the function to isolate the values and analyze each property type separately. This results are stored in a new dataframe temp, which consists of columns for 'property\_type', 'max' (maximum homes sold), 'min' (minimum homes sold), and 'mean' (average homes sold).

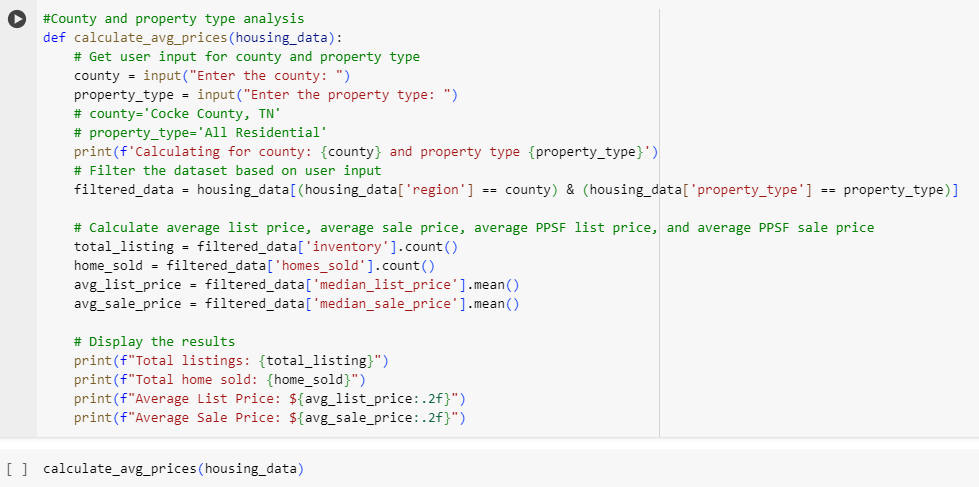
The code max\_min\_home\_sold stores the output DataFrame returned by the property\_type\_analysis function. The print statements display this DataFrame, presenting the analysis results in a tabular format. Additionally, a specific subset of the data is printed, showing the analysis for the 'All Residential' property type.

OUTPUT:  


Q2. How is the study of home listings and pricing vary based on the chosen county and property type?

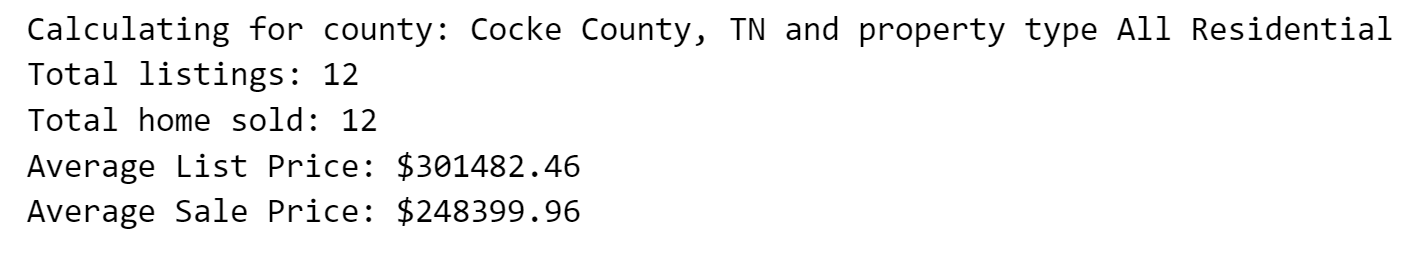
The python function ‘calculate\_avg\_prices’ is defined to analyze redfin real estate data for a specific county and property type provided by the user. Firstly, it prompts the user to input the county and property type we want to analyze. Once we provide this information, the function filters the dataset redfin\_data\_2023 based on the selections, isolating data entries corresponding to the specified county and property type.

After this the function calculates total count of home listing, total number of homes sold, average list price and average sale price using pandas aggregation like count and mean.



The print function displays the calculated results to the user, providing insights into the real estate market for the specified county and property type. The results include the total number of listings and homes sold, as well as the average list and sale prices. Overall, this function enables users to obtain essential information about the real estate market in a specific area and for a particular property type.

OUTPUT:



Q3. What is the regular monthly total sales for the period through the year 2023, and how do we visually present those using a line graph?

We have changed the housing data DataFrame into datetime format. The 'month' column is the one used to aggregate the DataFrame, and the'median\_sale\_price' values are the ones that will be used to get the global sale price by each month individually. As a result, it cuts the total by ten to the power of seven, it is inferred that the computation is for a price in billions of dollars. Secondly, the code illustrates plots of a line graph by using matplotlib. It depicts the variables (sale prices) by the index of the total\_sale\_price\_per\_month Series. The total\_sale\_price\_per\_month is a measure of the months. The plot title is changed "Total Sale Price Per Month," and the horizontal axis is marked "Months" while the vertical axis is marked "Total Sales Price (Billions)."



Finally, the function displays the graph as a visual presentation of the total sale prices trends across the different months, allowing users to inspect any fluctuations patterns that exist in housing sales over the timeline.

A graph with blue lines

Description automatically generated

Q4. Which state has maximum sales?

The function Maximum\_sales\_state is defined to determine which state has the maximum total sales for the year 2023. This function groups the data by state using the groupby function and calculates the total sales for each state by summing up the 'median\_sale\_price' column. This aggregated data gives the total sales amount for each state. the total sales values are converted to billions by dividing them by 1,000,000,000. Using idxmax function, we can get the maximum sales with the corresponding state and then print function prints the result indicating the state with the maximum sales and its corresponding total sales amount in billion dollars.

A screenshot of a computer program

Description automatically generated

Q5. How does the analysis of home listings vary on a monthly basis, and could you plot a side-by-side graph to visualize these differences?

The monthly\_analysis function defines a comprehensive analysis of monthly home listings using housing data which is provided as input. Initially, it converts the 'period\_begin' column to datetime format to extract the month and creates a new column named 'month', representing the abbreviated month names (e.g., Jan, Feb).

This function calculates the total inventory, total new listings, and total off-market listings for each month by grouping the data by month using the groupby function. These statistics are aggregated and stored in a DataFrame named monthly\_stats. Month order list is defined to make sure the correct order of the months and converts the 'month' column to categorical data type with the specified order.





For this type of property listing, it creates a bar graph illustration. The graph shows the total inventory, the total new listings, and the total off market listings graphically represented by the distinct bars of which thickness and width are predetermined for each category. The 'Month' component on the x-axis, and 'Inventory' on the y-axis it presents.

A graph of a home listing

Description automatically generated

Q6. List the top 10 states with maximum sales

*The state\_maximum\_sale\_price function is defined to analyze the top 10 states with the highest total housing sales prices based on the housing data. firstly, the function groups the data by state using the groupby function and calculates the total sales price for each state by summing up the 'median\_sale\_price' column. These total sales are then converted to billions and rounded to two decimal places. And then the function selects the top 10 states with the highest total sales prices using the nlargest function and sorts them in descending order. The result is then formatted to display the total sales prices in billions and sorted accordingly. We have used a bar graph to show the top 10 states with maximum sales.* The states are on the x-axis, and the sales in billions they had are shown on the y-axis. For bars visibility, they are painted with sky blue. To make comprehension easier, the labels of the x-axis states are rotated clockwise by 45 degrees. The states that have the largest sales volume among the states are put on the left side of the chart with the x-axis inverted. By classifying the states in terms of the total number of sales, this research provides perceptive data that analyzes the real estate market.

*A screenshot of a computer

Description automatically generated*

*A graph of blue bars with white text

Description automatically generated*

***Findings & Managerial Implications:***

***Q1. In what way are the maximum, minimum, and average home sales vary across different property types?***

***Findings:*** *Sales Distribution: Property types are heterogeneous with regards to sales’ volumes, Single-family residence properties being the leader with the greatest maximum sales (5238), and Multi-Family (2-4 residential units) properties being the least preferred with the lowest maximum sales values (379).* ***Managerial Implications:*** *Targeted Marketing: Make sales and marketing strategies tailored to property types that have higher sales volumes to be able to integrate the resources smoothly.*

*Diversification Strategy: Contemplate the option of a sample-stock ownership to offset risk and give eagerness to capitalize on sales possibilities across various styles of real estates.*

***Q3. What is the regular monthly total sales for the period through the year 2023, and how do we visually present those using a line graph?***

***Findings:***

*Seasonal Sales Variation: The statistics show that there are clearly some oscillations in the mainly volume of sales over different months of a year, which shows the presence of seasonal trends in the housing market.  
  
Peak Sales Months: Of particular importance, the month-wise sales volumes by groups are headed by June and August, reflecting their prominence as the timeframes with the highest market activity and customers’ engagement.****Managerial Implications:*** *Optimized Marketing Strategies: Real Estate industry practitioners can use the consumer behavioral knowledge they get through high sales months into designing their marketing campaigns; redistribution of input and resources to these periods to generate the highest visibility and exploitation of the great buying interests at that time.  
  
Strategic Inventory Management: Seasonal sales fluctuations are an integral factor for the project developers and investors to implement tactful techniques for inventory management. The move to line up the construction schedule and rental timeline with the high season will let stakeholders decrease inventory while at the same time setting out risk and eliminate associated costs.*

***Q5. How does the analysis of home listings vary on a monthly basis, and could you plot a side-by-side graph to visualize these differences?***

***Findings:***

*Monthly Inventory Trends: Each year there are those months when the inventory is higher than the others. August was the month with the highest total number of units (1,605,784). While the lowest total units were in the month of December (1,450,003).****Managerial Implications:*** *Strategic Planning: Secondly, real estate managers can make changes in their strategy of planning for the forthcoming months as well as proper allocation of resources in line with monthly inventory trends. One manner by which higher inventory months may call for vigorous marketing, promotion and staging is that complex will move effectively in the market.  
  
Seasonal Adjustments: Assessing monthly firms of inventory leads to better seasonal management of pricing, promotions and also inventory management, which combined provide optimal sales and carrying costs minimization.*

***Q6. List the top 10 states with maximum sales***

***Findings:***

*Top Performing States: Texas and California, stated the universities, will be the two states with most total sales with the figures reaching $1.74 billion and $1.79 billion, respectively.****Managerial Implications:*** *Market Focus: Allocate funds, emphasis and strategic orientation to the best performing states like California and Texas which have a high rate of sales, serviceable market and demand.  
  
Regional Strategies: Use marketing campaigns, pricing strategies and product planning to match with high-performing states main features and dynamics, thereby selectiveness will be ensured for the greater impact.*

***Conclusion:***

*The project presented a deep data collection, cleansing, analysis and visualization process which led to revelation of 2023 real estate market dynamics by utilization of the Redfin housing dataset. Using pandas for data manipulation, and Matplotlib for visualization, they managed the values that were missing, the specification of dates and times, and transformed the data to allow for meaningful analysis. Reported results include tendencies in the housing market, such as fluctuations in housing prices and patterns related to a particular area. Such revelations enable the stakeholders to have a clear perspective of whether to purchase, sell, or invest in properties. Altogether, the study shows how data-informed techniques can be used in exploring and resolving the intricate issues of the real estate market.*

*Appendix:*

*Below are the libraries used in our coding/analysis:*

#Import required libraries and read the file in CSV format.

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

housing\_data = pd.read\_csv('redfin\_data\_2023.csv')

*Q1. In what way are the maximum, minimum, and average home sales vary across different property types?*

#define a function to calculate maximum, minimum, and average home sales

def property\_type\_analysis(housing\_data):

# Group by 'property\_type' and calculate the maximum ,minimum and avg number of homes sold

temp = housing\_data.groupby('property\_type')['homes\_sold'].agg(['max', 'min','mean']).reset\_index()

return temp

max\_min\_avg\_home\_sold= property\_type\_analysis(housing\_data)

print(max\_min\_avg\_home\_sold.to\_string(index=False))

print(max\_min\_avg\_home\_sold[max\_min\_avg\_home\_sold['property\_type']=='All Residential'])

*Output:*

property\_type max min mean

All Residential 6444.0 1.0 105.044550

Condo/Co-op 2168.0 1.0 31.685307

Multi-Family (2-4 Unit) 379.0 1.0 7.371951

Single Family Residential 5238.0 1.0 84.627158

Townhouse 1277.0 1.0 26.532289

property\_type max min mean

0 All Residential 6444.0 1.0 105.04455

*Q2. How is the study of home listings and pricing vary based on the chosen county and property type?*

#County and property type analysis

def calculate\_avg\_prices(housing\_data):

# Get user input for county and property type

county = input("Enter the county: ")

property\_type = input("Enter the property type: ")

# county='Cocke County, TN'

# property\_type='All Residential'

print(f'Calculating for county: {county} and property type {property\_type}')

# Filter the dataset based on user input

filtered\_data = housing\_data[(housing\_data['region'] == county) & (housing\_data['property\_type'] == property\_type)]

# Calculate average list price, average sale price, average PPSF list price, and average PPSF sale price

total\_listing = filtered\_data['inventory'].count()

home\_sold = filtered\_data['homes\_sold'].count()

avg\_list\_price = filtered\_data['median\_list\_price'].mean()

avg\_sale\_price = filtered\_data['median\_sale\_price'].mean()

# Display the results

print(f"Total listings: {total\_listing}")

print(f"Total home sold: {home\_sold}")

print(f"Average List Price: ${avg\_list\_price:.2f}")

print(f"Average Sale Price: ${avg\_sale\_price:.2f}")

calculate\_avg\_prices(housing\_data)

Output:

Calculating for county: Cocke County, TN and property type All Residential

Total listings: 12

Total home sold: 12

Average List Price: $301482.46

Average Sale Price: $248399.96

*Q3. What is the regular monthly total sales for the period through the year 2023, and how do we visually present those using a line graph?*

#total sales for each month

def total\_Sales\_per\_month(housing\_data):

housing\_data['period\_begin'] = pd.to\_datetime(housing\_data['period\_begin'])

# Extract month from 'period\_begin'

housing\_data['month'] = housing\_data['period\_begin'].dt.month

# Group by month and calculate total sale price for each month

total\_sale\_price\_per\_month = housing\_data.groupby('month')['median\_sale\_price'].sum()/10000000

print(total\_sale\_price\_per\_month)

# Plot the line graph

plt.plot(total\_sale\_price\_per\_month.index, total\_sale\_price\_per\_month.values, marker='o', linestyle='-')

plt.title('Total Sale Price Per Month')

plt.xlabel('Month')

plt.ylabel('Total Sale Price (Billions)')

# Use month abbreviations on the x-axis

plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

# Format y-axis tick labels to display values in millions

plt.gca().yaxis.set\_major\_formatter('${:.0f}B'.format)

plt.grid(True)

plt.show()

total\_Sales\_per\_month(housing\_data)

*Output:*

month

1 215.802992

2 216.771846

3 234.164103

4 233.560431

5 245.445120

6 258.773824

7 246.967610

8 256.033984

9 251.251699

10 247.891097

11 240.814134

12 239.343868

Name: median\_sale\_price, dtype: float64

*A graph with blue lines and numbers

Description automatically generated*

*Q4.* Which state has maximum sales?

#Which state has maximum sales

def Maximum\_sales\_state(housing\_data):

# Group by 'state' and calculate the total sales for each state

total\_sales\_per\_state = housing\_data.groupby('state')['median\_sale\_price'].sum()

# Convert total sales to billions

total\_sales\_per\_state\_billion = total\_sales\_per\_state / 1000000000 # Convert to billions

# Find the state with maximum sales

max\_sales\_state = total\_sales\_per\_state\_billion.idxmax()

max\_sales\_amount = total\_sales\_per\_state\_billion.max()

print(f"The state with the maximum sales is {max\_sales\_state} with total sales of {max\_sales\_amount:.2f} billion dollars.")

Maximum\_sales\_state(housing\_data)

*Output:*

The state with the maximum sales is Texas with total sales of 1.79 billion dollars.

*Q5. How does the analysis of home listings vary on a monthly basis, and could you plot a side-by-side graph to visualize these differences?*

#Monthly home listing analysis

def monthly\_analysis(housing\_data):

# Convert 'period\_begin' to datetime format

housing\_data['period\_begin'] = pd.to\_datetime(housing\_data['period\_begin'])

# Extract month from 'period\_begin'

housing\_data['month'] = housing\_data['period\_begin'].dt.strftime('%b')

housing\_data['total\_off\_market\_count'] = housing\_data['off\_market\_in\_two\_weeks'] \* housing\_data['inventory']

# Group by month and calculate total inventory, total new listings, and total off-market listings

monthly\_stats = housing\_data.groupby('month').agg(

total\_inventory=('inventory', 'sum'),

total\_new\_listings=('new\_listings', 'sum'),

total\_off\_market=('total\_off\_market\_count', 'sum')

).reset\_index()

# Define the order of months

month\_order = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Convert month to categorical data type with the correct order

monthly\_stats['month'] = pd.Categorical(monthly\_stats['month'], categories=month\_order, ordered=True)

# Sort DataFrame based on month

monthly\_stats = monthly\_stats.sort\_values(by='month')

# Display the results

print(monthly\_stats.to\_string(index=False))

# Plotting

# Set the width of the bars

bar\_width = 0.3

# Define the x-axis positions for each category

x\_indexes = np.arange(len(monthly\_stats['month']))

#x\_indexes = monthly\_stats.index

# Plotting

plt.figure(figsize=(10, 6))

# Plot total inventory

plt.bar(x\_indexes - bar\_width, monthly\_stats['total\_inventory'], width=bar\_width, color='skyblue', label='Total Inventory')

# Plot total new listings

plt.bar(x\_indexes, monthly\_stats['total\_new\_listings'], width=bar\_width, color='orange', label='Total New Listings', alpha=0.7)

# Plot total off-market listings

plt.bar(x\_indexes + bar\_width, monthly\_stats['total\_off\_market'], width=bar\_width, color='green', label='Total Off-Market Listings', alpha=0.7)

plt.xlabel('Month')

plt.ylabel('Inventory')

plt.title('Monthly Home Listing Analysis')

plt.legend()

plt.xticks(ticks=x\_indexes, labels=monthly\_stats['month'], rotation=45)

plt.tight\_layout()

plt.show()

monthly\_analysis(housing\_data)

*Output:*

month total\_inventory total\_new\_listings total\_off\_market

Jan 1436640.0 561038.0 435590.910458

Feb 1394497.0 579665.0 511328.099577

Mar 1409105.0 768889.0 533234.701771

Apr 1424562.0 748404.0 549359.503064

May 1476311.0 850985.0 564750.354958

Jun 1534547.0 855081.0 549802.292864

Jul 1565108.0 773777.0 530879.690563

Aug 1605784.0 803489.0 533166.835462

Sep 1664589.0 742267.0 529536.182358

Oct 1685924.0 714924.0 507641.626145

Nov 1633104.0 579277.0 433817.799484

Dec 1450003.0 408703.0 344605.194070

*A graph of a home listing

Description automatically generated*

***Q6. List the top 10 states with maximum sales***

#top 10 state with max housing sale price

def state\_maximum\_sale\_price(housing\_data):

# Group by 'state' and calculate the total sales for each state

total\_sales\_per\_state = housing\_data.groupby('state')['median\_sale\_price'].sum() / 1000000000

total\_sales\_per\_state\_billion = total\_sales\_per\_state.round(2) # Round to two decimal places

# Sort states based on total sales and get the top 10 states

top\_10\_states = total\_sales\_per\_state\_billion.nlargest(10)

top\_10\_states = top\_10\_states.sort\_values(ascending=True) # Sort in ascending order

top\_10\_states = top\_10\_states.reset\_index().rename(columns={"median\_sale\_price": "Total Sales (Billion)"})

top\_10\_states['Total Sales (Billion)'] = top\_10\_states['Total Sales (Billion)'].apply(lambda x: '$' + str(x) + 'B')

# Display the top 10 states

print("Top 10 states with highest total sales (in billions):")

print(top\_10\_states.to\_string(index=False))

return top\_10\_states

def main():

housing\_data = pd.read\_csv('redfin\_data\_2023.csv')

top\_10\_states = state\_maximum\_sale\_price(housing\_data)

# Plotting

plt.figure(figsize=(10, 6))

plt.bar(top\_10\_states['state'], top\_10\_states['Total Sales (Billion)'], color='skyblue')

plt.xlabel('State')

plt.ylabel('Total Sales (Billion)')

plt.title('Top 10 States with Highest Total Sales')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.gca().invert\_xaxis() # Invert x-axis

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

main()

***Output:***

Top 10 states with highest total sales (in billions):

state Total Sales (Billion)

Washington $0.77B

Tennessee $0.9B

New York $1.0B

Florida $1.1B

Georgia $1.22B

North Carolina $1.22B

Colorado $1.38B

Virginia $1.59B

California $1.74B

Texas $1.79B

***A graph of blue bars with states

Description automatically generated***

***References:***

***Data set:*** *Data Center metrics definitions*. Redfin Real Estate News. (2023, October 27). https://www.redfin.com/news/data-center-metrics-definitions/

***Missing values:*** YouTube. (2021, May 20). *Handling missing values in pandas Dataframe | GeeksforGeeks*. YouTube. https://www.youtube.com/watch?v=uDr67HBIPz8