```
# Data Cleaning and Transformation in the Python script

# 1. Data Preparation:
# - Loading Data: The dataset was loaded into the notebook and the
structure of the first few rows was examined to understand the data.
# - Initial Cleaning: Preliminary cleaning steps were performed,
including type conversion and handling of missing values as per the
steps outlined in the Python script.

# 2. Data Exploration:
# - Visualization: Various techniques such as bar charts, line graphs,
and scatter plots were used to visualize data, identify trends, and
understand patterns.
# - Statistical Analysis: Statistical methods were employed to explore
relationships between variables, such as correlation analysis and
hypothesis testing.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from bokeh.io import output_notebook, show
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.transform import cumsum
from math import pi

# File location
df = pd.read_csv(r'C:\Users\ddiol\OneDrive\Python\Project\Housing\
Uk_House_1995_2023.csv')
df.head()
```

```
   {F887F88E-7D15-4415-804E-52EAC2F10958}  70000  1995-07-07 00:00
MK15 9HP  \
0  {40FD4DF2-5362-407C-92BC-566E2CCE89E9}  44500  1995-02-03 00:00
SR6 0AQ
1  {7A99F89E-7D81-4E45-ABD5-566E49A045EA}  56500  1995-01-13 00:00
CO6 1SQ
2  {28225260-E61C-4E57-8B56-566E5285B1C1}  58000  1995-07-28 00:00
B90 4TG
3  {444D34D7-9BA6-43A7-B695-4F48980E0176}  51000  1995-06-28 00:00
DY5 1SA
4  {AE76CAF1-F8CC-43F9-8F63-4F48A2857D41}  17000  1995-03-10 00:00
S65 1QJ

   D  N  F  31 Unnamed: 8     ALDRICH DRIVE          WILLEN  MILTON
KEYNES  \
0  T  N  F  50        NaN       HOWICK PARK     SUNDERLAND
SUNDERLAND
1  T  N  F  19        NaN  BRICK KILN CLOSE     COGGESHALL
```

```
COLCHESTER
2  T  N  F  37        NaN  RAINSBROOK DRIVE        SHIRLEY
SOLIHULL
3  S  N  F  59        NaN        MERRY HILL  BRIERLEY HILL  BRIERLEY
HILL
4  T  N  L  22        NaN     DENMAN STREET      ROTHERHAM
ROTHERHAM

  MILTON KEYNES.1  MILTON KEYNES.2  A A.1
0      SUNDERLAND    TYNE AND WEAR  A   A
1       BRAINTREE            ESSEX  A   A
2        SOLIHULL    WEST MIDLANDS  A   A
3          DUDLEY    WEST MIDLANDS  A   A
4       ROTHERHAM  SOUTH YORKSHIRE  A   A

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28276227 entries, 0 to 28276226
Data columns (total 16 columns):
 #   Column                                       Dtype
---  ------                                       -----
 0   {F887F88E-7D15-4415-804E-52EAC2F10958}  object
 1   70000                                        int64
 2   1995-07-07 00:00                             object
 3   MK15 9HP                                     object
 4   D                                            object
 5   N                                            object
 6   F                                            object
 7   31                                           object
 8   Unnamed: 8                                   object
 9   ALDRICH DRIVE                                object
 10  WILLEN                                       object
 11  MILTON KEYNES                                object
 12  MILTON KEYNES.1                              object
 13  MILTON KEYNES.2                              object
 14  A                                            object
 15  A.1                                          object
dtypes: int64(1), object(15)
memory usage: 3.4+ GB

# Copying the DataFrame: Creates a copy of the original DataFrame to
preserve the original data. This is a good practice to avoid
unintentional modifications.
# Renaming Columns: Standardizes column names to ensure consistency
and clarity, especially if the original names are unclear or not user-
friendly. The new names appear to be based on a standard naming
convention from the ONS (Office for National Statistics) website.
# Checking Data: df1.info() is used again to verify the changes and
ensure the renaming process has been correctly applied.
```

```
  Cell In[4], line 1
    > Copying the DataFrame: Creates a copy of the original DataFrame
to preserve the original data. This is a good practice to avoid
unintentional modifications.
    ^
SyntaxError: invalid syntax
```

```
# Rename the DataFrame
df1 = df.copy()

# Define the new column names based on ONS website
column_names = ['Transaction_unique_identifier', 'Price',
'Date_of_Transfer', 'Postcode', 'Property_Type', 'Old/New',
'Duration', 'PAON', 'SAON', 'Street', 'Locality', 'Town/City',
'District', 'County', 'Transaction_type', 'Record_status']

# Rename the columns
df1.columns = column_names

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28276227 entries, 0 to 28276226
Data columns (total 16 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Transaction_unique_identifier  object
 1   Price                           int64
 2   Date_of_Transfer                object
 3   Postcode                        object
 4   Property_Type                   object
 5   Old/New                         object
 6   Duration                        object
 7   PAON                            object
 8   SAON                            object
 9   Street                          object
 10  Locality                        object
 11  Town/City                       object
 12  District                        object
 13  County                          object
 14  Transaction_type                object
 15  Record_status                   object
dtypes: int64(1), object(15)
memory usage: 3.4+ GB
```

```
# Identifying Missing Values: Using isnull() or similar methods to
locate NaNs or missing entries.
# Filling Missing Values: Using techniques such as forward fill,
backward fill, or imputation based on statistical measures (mean,
```

```
median, mode).
# Dropping Missing Values: Removing rows or columns that contain a
significant amount of missing data.

# Drop missing values in 'Postcode' column
df1 = df1.dropna(subset=['Postcode']).reset_index(drop=True)

# Fill missing values in 'SAON', 'Street', and 'Locality' columns
using .loc
df1.loc[:, 'SAON'] = df1['SAON'].fillna('Unknown')
df1.loc[:, 'Street'] = df1['Street'].fillna('Unknown')
df1.loc[:, 'Locality'] = df1['Locality'].fillna('Unknown')

#Data Type Conversion
# Another essential step involves converting data types to appropriate
formats, such as ensuring dates are in datetime format and numerical
values are correctly identified. This ensures that subsequent analysis
and visualizations are accurate.

# Convert 'Date_of_Transfer' to datetime
df1['Date_of_Transfer'] = pd.to_datetime(df1['Date_of_Transfer'])

# Extract date components
df1['Date'] = df1['Date_of_Transfer'].dt.date
df1['Day'] = df1['Date_of_Transfer'].dt.day
df1['Month'] = df1['Date_of_Transfer'].dt.month
df1['Year'] = df1['Date_of_Transfer'].dt.year

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28230258 entries, 0 to 28230257
Data columns (total 20 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Transaction_unique_identifier  object
 1   Price                           int64
 2   Date_of_Transfer                datetime64[ns]
 3   Postcode                        object
 4   Property_Type                   object
 5   Old/New                         object
 6   Duration                        object
 7   PAON                            object
 8   SAON                            object
 9   Street                          object
 10  Locality                        object
 11  Town/City                       object
 12  District                        object
 13  County                          object
 14  Transaction_type                object
 15  Record_status                   object
```

```
 16  Date                                object
 17  Day                                 int32
 18  Month                               int32
 19  Year                                int32
dtypes: datetime64[ns](1), int32(3), int64(1), object(15)
memory usage: 3.9+ GB
```

```
# Calculate count of entries per county
county_count = df1['County'].value_counts().reset_index()

# Rename the columns
county_count.columns = ['County', 'count']

# Display DataFrame information
county_count.info()

# Display the DataFrame to see the counts
print(county_count)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   County  132 non-null    object
 1   count   132 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 2.2+ KB
                      County    count
0             GREATER LONDON  3616913
1         GREATER MANCHESTER  1255202
2              WEST MIDLANDS  1077364
3             WEST YORKSHIRE  1074166
4                       KENT   801625
..                       ...      ...
127          SOUTH GLAMORGAN     1770
128                CLEVELAND     1762
129          ISLES OF SCILLY      628
130               CUMBERLAND      448
131  WESTMORLAND AND FURNESS      340

[132 rows x 2 columns]
```

```
# List of Counties was printed for clustering by Region

# Get a list of unique entries in the 'County' column
unique_counties = df1['County'].unique().tolist()

# Display the list of unique counties
print(unique_counties)
```

```
['TYNE AND WEAR', 'ESSEX', 'WEST MIDLANDS', 'SOUTH YORKSHIRE',
'CAMBRIDGESHIRE', 'CHESHIRE', 'SWANSEA', 'NORFOLK', 'STAFFORDSHIRE',
'GLOUCESTERSHIRE', 'GREATER LONDON', 'MERSEYSIDE', 'BLACKPOOL',
'NORTHUMBERLAND', 'WORCESTERSHIRE', 'LINCOLNSHIRE', 'AVON',
'OXFORDSHIRE', 'SURREY', 'LUTON', 'GREATER MANCHESTER', 'EAST SUSSEX',
'LEICESTERSHIRE', 'NORTHAMPTONSHIRE', 'BUCKINGHAMSHIRE', 'DERBYSHIRE',
'HEREFORD AND WORCESTER', 'KENT', 'NOTTINGHAMSHIRE', 'SOMERSET',
'HUMBERSIDE', 'BOURNEMOUTH', 'HAMPSHIRE', 'WEST SUSSEX', 'GWENT',
'WOKINGHAM', 'NEWPORT', 'LANCASHIRE', 'STOKE-ON-TRENT', 'SUFFOLK',
'SOUTHEND-ON-SEA', 'WINDSOR AND MAIDENHEAD', 'STOCKTON-ON-TEES',
'NORTH YORKSHIRE', 'WEST YORKSHIRE', 'WARWICKSHIRE', 'SHROPSHIRE',
'WREXHAM', 'HERTFORDSHIRE', 'PORTSMOUTH', 'THURROCK', 'CUMBRIA',
'SLOUGH', 'BRACKNELL FOREST', 'SOUTH GLAMORGAN', 'CORNWALL', 'CLWYD',
'WARRINGTON', 'BEDFORDSHIRE', 'BERKSHIRE', 'MID GLAMORGAN', 'CONWY',
'ISLE OF WIGHT', 'SOUTHAMPTON', 'DEVON', 'MILTON KEYNES', 'DURHAM',
'DYFED', 'POOLE', 'POWYS', 'READING', 'GWYNEDD', 'LEICESTER',
'RUTLAND', 'DORSET', 'CLEVELAND', 'DARLINGTON', 'THAMESDOWN',
'HARTLEPOOL', 'CARDIFF', 'WILTSHIRE', 'TORFAEN', 'YORK', 'RHONDDA
CYNON TAFF', 'REDCAR AND CLEVELAND', 'TORBAY', 'CITY OF DERBY', 'CITY
OF KINGSTON UPON HULL', 'BRIGHTON AND HOVE', 'BATH AND NORTH EAST
SOMERSET', 'CITY OF PLYMOUTH', 'MIDDLESBROUGH', 'CHESHIRE EAST',
'MONMOUTHSHIRE', 'EAST RIDING OF YORKSHIRE', 'NORTH EAST
LINCOLNSHIRE', 'CARMARTHENSHIRE', 'NORTH LINCOLNSHIRE', 'THE VALE OF
GLAMORGAN', 'PEMBROKESHIRE', 'WEST GLAMORGAN', 'ISLE OF ANGLESEY',
'CEREDIGION', 'DENBIGHSHIRE', 'MERTHYR TYDFIL', 'BEDFORD', 'HALTON',
'BLACKBURN WITH DARWEN', 'BLAENAU GWENT', 'CITY OF NOTTINGHAM', 'CITY
OF BRISTOL', 'CITY OF PETERBOROUGH', 'FLINTSHIRE', 'CAERPHILLY',
'HEREFORDSHIRE', 'CHESHIRE WEST AND CHESTER', 'SWINDON', 'COUNTY
DURHAM', 'CENTRAL BEDFORDSHIRE', 'SOUTH GLOUCESTERSHIRE', 'NORTH
SOMERSET', 'NEATH PORT TALBOT', 'BOURNEMOUTH, CHRISTCHURCH AND POOLE',
'WEST BERKSHIRE', 'ISLES OF SCILLY', 'BRIDGEND', 'MEDWAY', 'WEST
NORTHAMPTONSHIRE', 'WREKIN', 'NORTH NORTHAMPTONSHIRE', 'WESTMORLAND
AND FURNESS', 'CUMBERLAND']

# Rename the DataFrame
df2 = df1.copy()

# Define regions
regions = {
    'North East England': ['TYNE AND WEAR', 'NORTHUMBERLAND',
'DURHAM', 'HARTLEPOOL', 'STOCKTON-ON-TEES', 'MIDDLESBROUGH', 'REDCAR
AND CLEVELAND', 'DARLINGTON', 'CLEVELAND'],
    'North West England': ['CUMBRIA', 'LANCASHIRE', 'GREATER
MANCHESTER', 'MERSEYSIDE', 'BLACKPOOL', 'BLACKBURN WITH DARWEN',
'HALTON', 'WARRINGTON'],
    'Yorkshire and the Humber': ['NORTH YORKSHIRE', 'WEST YORKSHIRE',
'SOUTH YORKSHIRE', 'EAST RIDING OF YORKSHIRE', 'HUMBERSIDE', 'CITY OF
KINGSTON UPON HULL', 'YORK'],
    'East Midlands': ['DERBYSHIRE', 'LEICESTERSHIRE', 'LINCOLNSHIRE',
'NORTHAMPTONSHIRE', 'NOTTINGHAMSHIRE', 'RUTLAND', 'CITY OF DERBY',
```

```
'CITY OF NOTTINGHAM', 'LEICESTER'],
    'West Midlands': ['SHROPSHIRE', 'STAFFORDSHIRE', 'WEST MIDLANDS',
'WARWICKSHIRE', 'WORCESTERSHIRE', 'HEREFORDSHIRE', 'WREKIN'],
    'East of England': ['BEDFORDSHIRE', 'CAMBRIDGESHIRE', 'ESSEX',
'HERTFORDSHIRE', 'NORFOLK', 'SUFFOLK', 'CITY OF PETERBOROUGH',
'THURROCK', 'SOUTHEND-ON-SEA'],
    'London': ['GREATER LONDON'],
    'South East England': ['BERKSHIRE', 'BUCKINGHAMSHIRE', 'EAST
SUSSEX', 'HAMPSHIRE', 'KENT', 'OXFORDSHIRE', 'SURREY', 'WEST SUSSEX',
'MILTON KEYNES', 'PORTSMOUTH', 'SOUTHAMPTON', 'ISLE OF WIGHT',
'BRACKNELL FOREST', 'WOKINGHAM', 'WINDSOR AND MAIDENHEAD', 'SLOUGH',
'MEDWAY', 'BRIGHTON AND HOVE'],
    'South West England': ['DORSET', 'SOMERSET', 'CORNWALL', 'DEVON',
'GLOUCESTERSHIRE', 'WILTSHIRE', 'BATH AND NORTH EAST SOMERSET',
'BOURNEMOUTH, CHRISTCHURCH AND POOLE', 'CITY OF BRISTOL', 'SOUTH
GLOUCESTERSHIRE', 'NORTH SOMERSET', 'PLYMOUTH', 'TORBAY', 'SWINDON',
'ISLES OF SCILLY'],
    'Wales': ['ISLE OF ANGLESEY', 'BRECONSHIRE', 'CAERNARFONSHIRE',
'CARDIFF', 'CARMARTHENSHIRE', 'CEREDIGION', 'CLWYD', 'CONWY',
'DENBIGHSHIRE', 'DYFED', 'FLINTSHIRE', 'GLAMORGAN', 'GWENT',
'GWYNEDD', 'MERTHYR TYDFIL', 'MONMOUTHSHIRE', 'NEATH PORT TALBOT',
'NEWPORT', 'PEMBROKESHIRE', 'POWYS', 'RHONDDA CYNON TAFF', 'SWANSEA',
'TORFAEN', 'VALE OF GLAMORGAN', 'WREXHAM'],
    'Scotland': [],  # No counties listed for Scotland in your
provided list
    'Northern Ireland': [],  # No counties listed for Northern Ireland
in your provided list
}

# Flatten the regions dictionary for easy lookup
county_to_region = {county: region for region, counties in
regions.items() for county in counties}

# Function to classify the county into regions
def classify_county(county):
    return county_to_region.get(county, 'Unknown')

# Apply the function to classify counties into regions
df2['Region'] = df2['County'].apply(classify_county)

# Display the DataFrame to verify the changes
print(df2[['County', 'Region']])
```

```
                County                     Region
0         TYNE AND WEAR        North East England
1                 ESSEX           East of England
2         WEST MIDLANDS             West Midlands
3         WEST MIDLANDS             West Midlands
4       SOUTH YORKSHIRE  Yorkshire and the Humber
```

```
...                    ...                          ...
28230253   GREATER LONDON                      London
28230254  SOUTHEND-ON-SEA           East of England
28230255            ESSEX           East of England
28230256            ESSEX           East of England
28230257            ESSEX           East of England

[28230258 rows x 2 columns]
```

```
df2.head()
```

```
             Transaction_unique_identifier  Price Date_of_Transfer
Postcode  \
0  {40FD4DF2-5362-407C-92BC-566E2CCE89E9}  44500       1995-02-03  SR6
0AQ
1  {7A99F89E-7D81-4E45-ABD5-566E49A045EA}  56500       1995-01-13  CO6
1SQ
2  {28225260-E61C-4E57-8B56-566E5285B1C1}  58000       1995-07-28  B90
4TG
3  {444D34D7-9BA6-43A7-B695-4F48980E0176}  51000       1995-06-28  DY5
1SA
4  {AE76CAF1-F8CC-43F9-8F63-4F48A2857D41}  17000       1995-03-10  S65
1QJ

  Property_Type Old/New Duration PAON     SAON             Street  ...
\
0             T       N        F   50  Unknown       HOWICK PARK  ...

1             T       N        F   19  Unknown  BRICK KILN CLOSE  ...

2             T       N        F   37  Unknown  RAINSBROOK DRIVE  ...

3             S       N        F   59  Unknown        MERRY HILL  ...

4             T       N        L   22  Unknown     DENMAN STREET  ...


       Town/City    District           County Transaction_type
Record_status  \
0     SUNDERLAND  SUNDERLAND    TYNE AND WEAR                A
A
1     COLCHESTER   BRAINTREE            ESSEX                A
A
2       SOLIHULL    SOLIHULL    WEST MIDLANDS                A
A
3  BRIERLEY HILL      DUDLEY    WEST MIDLANDS                A
A
4      ROTHERHAM   ROTHERHAM  SOUTH YORKSHIRE                A
A

         Date Day  Month  Year                     Region
```

```
0  1995-02-03   3      2  1995        North East England
1  1995-01-13  13      1  1995           East of England
2  1995-07-28  28      7  1995             West Midlands
3  1995-06-28  28      6  1995             West Midlands
4  1995-03-10  10      3  1995  Yorkshire and the Humber

[5 rows x 21 columns]
```

```
# Entries were rename to provide ration based on reseach from ONS
website

# Defining property type mapping
df2['Property_Type'] = df2['Property_Type'].map({
    'O': 'Other',
    'F': 'Flat/Maisonette',
    'D': 'Detached',
    'S': 'Semi-detached',
    'T': 'Terraced'
})

# Map the values based on ONS research
df2['Transaction_type'] = df2['Transaction_type'].map({
    'A': 'Standard Price Paid entry',
    'B': 'Additional Price Paid entry (includes transactions under
specific conditions like repossessions, buy-to-let, etc.)'
})

df2['Record_status'] = df2['Record_status'].map({
    'A': 'Add (a new entry)',
    'C': 'Change (an amendment to an entry)',
    'D': 'Delete (a deletion of an entry)'
})

# Renaming 'Duration' column 'F' and 'L' for Freehold and Leasehold
based on ONS research
df2['Duration'] = df2['Duration'].map({
    'F': 'Freehold',
    'L': 'Leasehold'
})

# Rename entries in 'Old/New' column
df2['Old/New'] = df2['Old/New'].map({
    'N': 'New',
    'Y': 'Old'
})

# List of relevant columns including the renamed ones
columns_of_interest = [
    'Duration', 'Property_Type', 'County', 'District', 'Town/City',
'Postcode',
```

```
    'Transaction_type', 'Record_status'
]

# Display the first few rows to verify the changes
df2.head()
```

```
            Transaction_unique_identifier  Price Date_of_Transfer
Postcode  \
0  {40FD4DF2-5362-407C-92BC-566E2CCE89E9}  44500       1995-02-03  SR6
0AQ
1  {7A99F89E-7D81-4E45-ABD5-566E49A045EA}  56500       1995-01-13  CO6
1SQ
2  {28225260-E61C-4E57-8B56-566E5285B1C1}  58000       1995-07-28  B90
4TG
3  {444D34D7-9BA6-43A7-B695-4F48980E0176}  51000       1995-06-28  DY5
1SA
4  {AE76CAF1-F8CC-43F9-8F63-4F48A2857D41}  17000       1995-03-10  S65
1QJ

   Property_Type Old/New   Duration PAON     SAON
Street  ...  \
0       Terraced     New   Freehold   50  Unknown       HOWICK
PARK  ...
1       Terraced     New   Freehold   19  Unknown  BRICK KILN
CLOSE  ...
2       Terraced     New   Freehold   37  Unknown  RAINSBROOK
DRIVE  ...
3  Semi-detached     New   Freehold   59  Unknown        MERRY
HILL  ...
4       Terraced     New  Leasehold   22  Unknown     DENMAN
STREET  ...

       Town/City    District           County
Transaction_type  \
0     SUNDERLAND  SUNDERLAND    TYNE AND WEAR  Standard Price Paid
entry
1     COLCHESTER   BRAINTREE            ESSEX  Standard Price Paid
entry
2       SOLIHULL    SOLIHULL    WEST MIDLANDS  Standard Price Paid
entry
3  BRIERLEY HILL      DUDLEY    WEST MIDLANDS  Standard Price Paid
entry
4      ROTHERHAM   ROTHERHAM  SOUTH YORKSHIRE  Standard Price Paid
entry

       Record_status        Date Day  Month  Year
Region
0  Add (a new entry)  1995-02-03   3      2  1995        North East
England
1  Add (a new entry)  1995-01-13  13      1  1995           East of
```

```
England
2  Add (a new entry)  1995-07-28  28      7  1995              West
Midlands
3  Add (a new entry)  1995-06-28  28      6  1995              West
Midlands
4  Add (a new entry)  1995-03-10  10      3  1995  Yorkshire and the
Humber

[5 rows x 21 columns]
```

```
entry_counts = df2.count()
print(entry_counts)
```

```
Transaction_unique_identifier    28230258
Price                            28230258
Date_of_Transfer                 28230258
Postcode                         28230258
Property_Type                    28230258
Old/New                          28230258
Duration                         28229725
PAON                             28226070
SAON                             28230258
Street                           28230258
Locality                         28230258
Town/City                        28230258
District                         28230258
County                           28230258
Transaction_type                 28230258
Record_status                    28230258
Date                             28230258
Day                              28230258
Month                            28230258
Year                             28230258
Region                           28230258
dtype: int64
```

```
# Checking if changes was successfull

# List of relevant columns including the renamed ones
columns_of_interest = [
    'Duration', 'Property_Type', 'County', 'District', 'Town/City',
'Postcode',
    'Transaction_type', 'Record_status'
]

# Ensure all columns exist in the DataFrame before counting unique
values
existing_columns_of_interest = [column for column in
columns_of_interest if column in df2.columns]
```

```python
# Count the unique values for each relevant column
unique_counts = {column: df2[column].nunique() for column in
existing_columns_of_interest}

# Display the unique counts
for column, count in unique_counts.items():
    print(f"{column}: {count} unique values")

# List the unique values for 'Transaction_type', 'Record_status', and
'Duration' if they exist
transaction_type_unique = df2['Transaction_type'].unique() if
'Transaction_type' in df2.columns else []
record_status_unique = df2['Record_status'].unique() if
'Record_status' in df2.columns else []
duration_unique = df2['Duration'].unique() if 'Duration' in
df2.columns else []

print("\nUnique values for 'Transaction_type':",
transaction_type_unique)
print("Unique values for 'Record_status':", record_status_unique)
print("Unique values for 'Duration':", duration_unique)
```

```
Duration: 2 unique values
Property_Type: 5 unique values
County: 132 unique values
District: 467 unique values
Town/City: 1172 unique values
Postcode: 1296549 unique values
Transaction_type: 2 unique values
Record_status: 1 unique values

Unique values for 'Transaction_type': ['Standard Price Paid entry'
 'Additional Price Paid entry (includes transactions under specific
conditions like repossessions, buy-to-let, etc.)']
Unique values for 'Record_status': ['Add (a new entry)']
Unique values for 'Duration': ['Freehold' 'Leasehold' nan]
```

```python
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28230258 entries, 0 to 28230257
Data columns (total 21 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Transaction_unique_identifier  object
 1   Price                           int64
 2   Date_of_Transfer                datetime64[ns]
 3   Postcode                        object
 4   Property_Type                   object
```

```
 5   Old/New                         object
 6   Duration                        object
 7   PAON                            object
 8   SAON                            object
 9   Street                          object
 10  Locality                        object
 11  Town/City                       object
 12  District                        object
 13  County                          object
 14  Transaction_type                object
 15  Record_status                   object
 16  Date                            object
 17  Day                             int32
 18  Month                           int32
 19  Year                            int32
 20  Region                          object
dtypes: datetime64[ns](1), int32(3), int64(1), object(16)
memory usage: 4.1+ GB
```

```python
#Saving data frame as csv for analysis
# Save df2 to a CSV file
df2.to_csv('Housing_Uk.csv', index=False)

# Provide the download link
print("The DataFrame has been saved to 'Housing_Uk.csv'.")
```

```
The DataFrame has been saved to 'Housing_Uk.csv'.
```