```python
import pandas as pd

import pandas as pd

filename = 'Property_Price_Train.csv'

# Read the file in binary mode and check for non-text content
with open(filename, 'rb') as file:
    content = file.read()

try:
    content.decode('utf-8')
    print("File appears to be valid UTF-8 text")
except UnicodeDecodeError:
    print("File contains non-text or binary data")

# If it's valid text, continue processing
try:
    df = pd.read_csv(filename, encoding='utf-8', on_bad_lines='skip')
    print("Data read successfully")
except pd.errors.ParserError as e:
    print(f"ParserError: {e}")

print(df.head())
```

```
File contains non-text or binary data
```

```
---------------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
last)
Cell In[26], line 17
     15 # If it's valid text, continue processing
     16 try:
---> 17     df = pd.read_csv(filename, encoding='utf-8',
on_bad_lines='skip')
     18     print("Data read successfully")
     19 except pd.errors.ParserError as e:

File D:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1026,
in read_csv(filepath_or_buffer, sep, delimiter, header, names,
index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
```

```
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File D:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:620,
in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File D:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1620,
in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File D:\anaconda\Lib\site-packages\pandas\io\parsers\readers.py:1898,
in TextFileReader._make_engine(self, f, engine)
   1895     raise ValueError(msg)
   1897 try:
-> 1898     return mapping[engine](f, **self.options)
   1899 except Exception:
   1900     if self.handles is not None:

File D:\anaconda\Lib\site-packages\pandas\io\parsers\
c_parser_wrapper.py:93, in CParserWrapper.__init__(self, src, **kwds)
     90 if kwds["dtype_backend"] == "pyarrow":
     91     # Fail here loudly instead of in cython after reading
     92     import_optional_dependency("pyarrow")
---> 93 self._reader = parsers.TextReader(src, **kwds)
     95 self.unnamed_cols = self._reader.unnamed_cols
     97 # error: Cannot determine type of 'names'

File parsers.pyx:574, in pandas._libs.parsers.TextReader.__cinit__()

File parsers.pyx:691, in pandas._libs.parsers.TextReader._get_header()

UnicodeDecodeError: 'utf-8' codec can't decode bytes in position 0-1:
invalid continuation byte
```

```
import magic

filename = 'Property_Price_Train.csv'
```

```
# Identify the file type
file_type = magic.from_file(filename, mime=True)
print(f"File type: {file_type}")

# Based on the file type, decide how to proceed
if 'zip' in file_type:
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)
    print("ZIP file extracted successfully")
elif 'tar' in file_type:
    with tarfile.open(filename, 'r:*') as tar_ref:
        tar_ref.extractall(extract_dir)
    print("TAR file extracted successfully")
else:
    print("Unsupported file type or not a compressed file")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[27], line 1
----> 1 import magic
      3 filename = 'Property_Price_Train.csv'
      5 # Identify the file type

ModuleNotFoundError: No module named 'magic'
```

```
import zipfile
import os

filename = 'Property_Price_Train.csv'
extract_dir = 'extracted_files'

# Check if the file is a ZIP file
if zipfile.is_zipfile(filename):
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)
    print("File extracted successfully")
else:
    print("File is not a ZIP archive")
```

```
File extracted successfully
```

```
import tarfile

# Check if the file is a tar archive (tar, tar.gz, tar.bz2)
if tarfile.is_tarfile(filename):
    with tarfile.open(filename, 'r:*') as tar_ref:
        tar_ref.extractall(extract_dir)
    print("File extracted successfully")
```

```
else:
    print("File is not a tar archive")
```

```
File is not a tar archive
```

```
pip install python-magic
```

```
Requirement already satisfied: python-magic in d:\anaconda\lib\site-
packages (0.4.27)Note: you may need to restart the kernel to use
updated packages.
```

```
import magic

filename = 'Property_Price_Train.csv'

# Identify the file type
file_type = magic.from_file(filename, mime=True)
print(f"File type: {file_type}")

# Based on the file type, decide how to proceed
if 'zip' in file_type:
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)
    print("ZIP file extracted successfully")
elif 'tar' in file_type:
    with tarfile.open(filename, 'r:*') as tar_ref:
        tar_ref.extractall(extract_dir)
    print("TAR file extracted successfully")
else:
    print("Unsupported file type or not a compressed file")
```

```
---------------------------------------------------------------------
-----
ImportError                               Traceback (most recent call
last)
Cell In[3], line 1
----> 1 import magic
      3 filename = 'Property_Price_Train.csv'
      5 # Identify the file type

File D:\anaconda\Lib\site-packages\magic\__init__.py:209
    206     return m.from_descriptor(fd)
    208 from . import loader
--> 209 libmagic = loader.load_lib()
    211 magic_t = ctypes.c_void_p
    214 def errorcheck_null(result, func, args):

File D:\anaconda\Lib\site-packages\magic\loader.py:49, in load_lib()
     46     pass
     47 else:
```

```
     48   # It is better to raise an ImportError since we are
importing magic module
---> 49   raise ImportError('failed to find libmagic.  Check your
installation')

ImportError: failed to find libmagic.  Check your installation
```

```
import os
import pandas as pd

extracted_dir = 'extracted_files'

# Find the CSV file in the extracted directory
csv_files = [f for f in os.listdir(extracted_dir) if
f.endswith('.csv')]
if csv_files:
    csv_file = os.path.join(extracted_dir, csv_files[0])
    try:
        df = pd.read_csv(csv_file, encoding='utf-8',
on_bad_lines='skip')
        print("Data read successfully from extracted CSV file")
        print(df.head())
    except pd.errors.ParserError as e:
        print(f"ParserError: {e}")
else:
    print("No CSV file found in the extracted files")
```

```
No CSV file found in the extracted files
```

```
import magic
import zipfile
import tarfile
import pandas as pd
import os

filename = 'Property_Price_Train.csv'
extract_dir = 'extracted_files'

# Identify the file type
file_type = magic.from_file(filename, mime=True)
print(f"File type: {file_type}")

# Handle ZIP files
if 'zip' in file_type:
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)
    print("ZIP file extracted successfully")

# Handle TAR files
elif 'tar' in file_type or 'gzip' in file_type or 'bzip2' in
file_type:
```

```
    with tarfile.open(filename, 'r:*') as tar_ref:
        tar_ref.extractall(extract_dir)
    print("TAR file extracted successfully")

# Handle Excel files
elif 'vnd.ms-excel' in file_type or 'vnd.openxmlformats-
officedocument.spreadsheetml.sheet' in file_type:
    df = pd.read_excel(filename)
    print("Excel file read successfully")
    print(df.head())

# Add handling for other file types if necessary
else:
    print("Unsupported file type or not a compressed file")

# If files were extracted, try reading them as CSV
if os.path.exists(extract_dir):
    csv_files = [f for f in os.listdir(extract_dir) if
f.endswith('.csv')]
    if csv_files:
        csv_file = os.path.join(extract_dir, csv_files[0])
        try:
            df = pd.read_csv(csv_file, encoding='utf-8',
on_bad_lines='skip')
            print("Data read successfully from extracted CSV file")
            print(df.head())
        except pd.errors.ParserError as e:
            print(f"ParserError: {e}")
    else:
        print("No CSV file found in the extracted files")
```

```
File type: application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet
Excel file read successfully
   Id  Building_Class Zoning_Class  Lot_Extent  Lot_Size Road_Type
Lane_Type  \
0   1              60          RLD        65.0      8450     Paved
NaN
1   2              20          RLD        80.0      9600     Paved
NaN
2   3              60          RLD        68.0     11250     Paved
NaN
3   4              70          RLD        60.0      9550     Paved
NaN
4   5              60          RLD        84.0     14260     Paved
NaN

  Property_Shape Land_Outline Utility_Type  ... Pool_Area Pool_Quality
\
0            Reg          Lvl       AllPub  ...         0          NaN
```

```
1            Reg          Lvl       AllPub  ...          0           NaN

2            IR1          Lvl       AllPub  ...          0           NaN

3            IR1          Lvl       AllPub  ...          0           NaN

4            IR1          Lvl       AllPub  ...          0           NaN


  Fence_Quality Miscellaneous_Feature Miscellaneous_Value
Month_Sold  \
0           NaN                   NaN                   0           2

1           NaN                   NaN                   0           5

2           NaN                   NaN                   0           9

3           NaN                   NaN                   0           2

4           NaN                   NaN                   0          12


  Year_Sold  Sale_Type  Sale_Condition  Sale_Price
0      2008         WD          Normal      208500
1      2007         WD          Normal      181500
2      2008         WD          Normal      223500
3      2006         WD         Abnorml      140000
4      2008         WD          Normal      250000

[5 rows x 81 columns]
No CSV file found in the extracted files

pip install python-magic-bin

Collecting python-magic-binNote: you may need to restart the kernel to
use updated packages.

  Downloading python_magic_bin-0.4.14-py2.py3-none-
win_amd64.whl.metadata (710 bytes)
Downloading python_magic_bin-0.4.14-py2.py3-none-win_amd64.whl (409
kB)
   ---------------------------------------- 0.0/409.3 kB ? eta -:--:--
   - -------------------------------------- 20.5/409.3 kB 330.3 kB/s
eta 0:00:02
   --- ------------------------------------ 41.0/409.3 kB 281.8 kB/s
eta 0:00:02
   --------- ---------------------------- 102.4/409.3 kB 590.8 kB/s
eta 0:00:01
   ----------- ------------------------- 122.9/409.3 kB 722.1 kB/s
eta 0:00:01
```

```
   ------------ ------------------------ 133.1/409.3 kB 563.7 kB/s
eta 0:00:01
   ------------------ ------------------ 204.8/409.3 kB 692.4 kB/s
eta 0:00:01
   ------------------------- ----------- 286.7/409.3 kB 842.9 kB/s
eta 0:00:01
   ----------------------------- ------- 327.7/409.3 kB 813.9 kB/s
eta 0:00:01
   ----------------------------------  399.4/409.3 kB 922.1 kB/s
eta 0:00:01
   ---------------------------------------- 409.3/409.3 kB 880.9 kB/s
eta 0:00:00
Installing collected packages: python-magic-bin
Successfully installed python-magic-bin-0.4.14
```

```
pip install python-magic
```

```
Requirement already satisfied: python-magic in d:\anaconda\lib\site-
packages (0.4.27)Note: you may need to restart the kernel to use
updated packages.
```

```
import magic

filename = 'Property_Price_Train.csv'

# Identify the file type
file_type = magic.from_file(filename, mime=True)
print(f"File type: {file_type}")
```

```
File type: application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet
```

```
import magic
import zipfile
import tarfile
import os

filename = 'Property_Price_Train.csv'
extract_dir = 'extracted_files'

# Identify the file type
file_type = magic.from_file(filename, mime=True)
print(f"File type: {file_type}")

# Handle ZIP files
if 'zip' in file_type:
    with zipfile.ZipFile(filename, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)
    print("ZIP file extracted successfully")
```

```
# Handle TAR files
elif 'tar' in file_type or 'gzip' in file_type or 'bzip2' in
file_type:
    with tarfile.open(filename, 'r:*') as tar_ref:
        tar_ref.extractall(extract_dir)
    print("TAR file extracted successfully")

# Add handling for other file types if necessary

# If files were extracted, continue processing them
if os.path.exists(extract_dir):
    # Example: Read CSV files if extracted
    csv_files = [f for f in os.listdir(extract_dir) if
f.endswith('.csv')]
    if csv_files:
        csv_file = os.path.join(extract_dir, csv_files[0])
        # Proceed with reading the CSV file
        print(f"Reading CSV file: {csv_file}")
    else:
        print("No CSV file found in the extracted files")
```

```
File type: application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet
No CSV file found in the extracted files
```

```
import pandas as pd

filename = 'Property_Price_Train.csv'  # Replace with your actual
filename

try:
    # Read the Excel file
    df = pd.read_excel(filename)

    # Display the first few rows of the DataFrame
    print("Data read successfully from Excel file:")
    print(df.head())

except Exception as e:
    print(f"Error reading Excel file: {e}")
```

```
Data read successfully from Excel file:
   Id  Building_Class Zoning_Class  Lot_Extent  Lot_Size Road_Type
Lane_Type  \
0   1              60          RLD        65.0      8450     Paved
NaN
1   2              20          RLD        80.0      9600     Paved
NaN
2   3              60          RLD        68.0     11250     Paved
NaN
3   4              70          RLD        60.0      9550     Paved
```

```
NaN
4   5              60          RLD         84.0     14260     Paved
NaN

  Property_Shape Land_Outline Utility_Type  ... Pool_Area Pool_Quality
\
0            Reg          Lvl       AllPub  ...         0          NaN

1            Reg          Lvl       AllPub  ...         0          NaN

2            IR1          Lvl       AllPub  ...         0          NaN

3            IR1          Lvl       AllPub  ...         0          NaN

4            IR1          Lvl       AllPub  ...         0          NaN


  Fence_Quality Miscellaneous_Feature Miscellaneous_Value
Month_Sold  \
0           NaN                   NaN                   0           2

1           NaN                   NaN                   0           5

2           NaN                   NaN                   0           9

3           NaN                   NaN                   0           2

4           NaN                   NaN                   0          12


  Year_Sold  Sale_Type  Sale_Condition  Sale_Price
0      2008         WD          Normal      208500
1      2007         WD          Normal      181500
2      2008         WD          Normal      223500
3      2006         WD         Abnorml      140000
4      2008         WD          Normal      250000

[5 rows x 81 columns]

df.head()

   Id  Building_Class Zoning_Class  Lot_Extent  Lot_Size Road_Type
Lane_Type  \
0   1              60          RLD        65.0      8450     Paved
NaN
1   2              20          RLD        80.0      9600     Paved
NaN
2   3              60          RLD        68.0     11250     Paved
NaN
3   4              70          RLD        60.0      9550     Paved
NaN
```

```
4   5              60          RLD         84.0     14260     Paved
NaN

  Property_Shape Land_Outline Utility_Type  ... Pool_Area Pool_Quality
\
0            Reg          Lvl       AllPub  ...         0           NaN

1            Reg          Lvl       AllPub  ...         0           NaN

2            IR1          Lvl       AllPub  ...         0           NaN

3            IR1          Lvl       AllPub  ...         0           NaN

4            IR1          Lvl       AllPub  ...         0           NaN


  Fence_Quality Miscellaneous_Feature Miscellaneous_Value
Month_Sold  \
0           NaN                  NaN                   0           2

1           NaN                  NaN                   0           5

2           NaN                  NaN                   0           9

3           NaN                  NaN                   0           2

4           NaN                  NaN                   0          12


  Year_Sold  Sale_Type  Sale_Condition  Sale_Price
0      2008         WD          Normal      208500
1      2007         WD          Normal      181500
2      2008         WD          Normal      223500
3      2006         WD         Abnorml      140000
4      2008         WD          Normal      250000

[5 rows x 81 columns]

df.shape

(1459, 81)

df.info

<bound method DataFrame.info of         Id  Building_Class
Zoning_Class  Lot_Extent  Lot_Size Road_Type  \
0        1              60          RLD        65.0      8450
Paved
1        2              20          RLD        80.0      9600
Paved
2        3              60          RLD        68.0     11250
```

```
Paved
3        4               70          RLD         60.0      9550
Paved
4        5               60          RLD         84.0     14260
Paved
...    ...              ...          ...          ...       ...       ..
.
1454  1455               20          FVR         62.0      7500
Paved
1455  1456               60          RLD         62.0      7917
Paved
1456  1457               20          RLD         85.0     13175
Paved
1457  1458               70          RLD         66.0      9042
Paved
1458  1459               20          RLD         68.0      9717
Paved

     Lane_Type Property_Shape Land_Outline Utility_Type  ... Pool_Area
\
0          NaN            Reg          Lvl       AllPub  ...         0

1          NaN            Reg          Lvl       AllPub  ...         0

2          NaN            IR1          Lvl       AllPub  ...         0

3          NaN            IR1          Lvl       AllPub  ...         0

4          NaN            IR1          Lvl       AllPub  ...         0

...        ...            ...          ...          ...  ...       ...

1454     Paved            Reg          Lvl       AllPub  ...         0

1455       NaN            Reg          Lvl       AllPub  ...         0

1456       NaN            Reg          Lvl       AllPub  ...         0

1457       NaN            Reg          Lvl       AllPub  ...         0

1458       NaN            Reg          Lvl       AllPub  ...         0


     Pool_Quality Fence_Quality Miscellaneous_Feature
Miscellaneous_Value  \
0             NaN           NaN                   NaN
0
1             NaN           NaN                   NaN
0
2             NaN           NaN                   NaN
```

```
0
3              NaN           NaN                    NaN
0
4              NaN           NaN                    NaN
0
...            ...           ...                    ...
...
1454           NaN           NaN                    NaN
0
1455           NaN           NaN                    NaN
0
1456           NaN         MnPrv                    NaN
0
1457           NaN         GdPrv                   Shed
2500
1458           NaN           NaN                    NaN
0
```

	Month_Sold	Year_Sold	Sale_Type	Sale_Condition	Sale_Price
0	2	2008	WD	Normal	208500
1	5	2007	WD	Normal	181500
2	9	2008	WD	Normal	223500
3	2	2006	WD	Abnorml	140000
4	12	2008	WD	Normal	250000
...	...	...	...	...	...
1454	10	2009	WD	Normal	185000
1455	8	2007	WD	Normal	175000
1456	2	2010	WD	Normal	210000
1457	5	2010	WD	Normal	266500
1458	4	2010	WD	Normal	142125

[1459 rows x 81 columns]>

```
df.dtypes
```

```
Id                   int64
Building_Class       int64
Zoning_Class        object
Lot_Extent         float64
Lot_Size             int64
                    ...
Month_Sold           int64
Year_Sold            int64
Sale_Type           object
Sale_Condition      object
Sale_Price           int64
Length: 81, dtype: object
```

```
null_counts = df.isnull().sum()
print("Columns with null values:")
print(null_counts[null_counts > 0])
```

```
Columns with null values:
Lot_Extent               259
Lane_Type               1368
Brick_Veneer_Type        871
Brick_Veneer_Area          8
Basement_Height           37
Basement_Condition        37
Exposure_Level            38
BsmtFinType1              37
BsmtFinType2              38
Electrical_System          1
Fireplace_Quality        689
Garage                    81
Garage_Built_Year         81
Garage_Finish_Year        81
Garage_Quality            81
Garage_Condition          81
Pool_Quality            1452
Fence_Quality           1178
Miscellaneous_Feature   1405
dtype: int64
```

```
df.columns
```

```
Index(['Id', 'Building_Class', 'Zoning_Class', 'Lot_Extent',
'Lot_Size',
       'Road_Type', 'Lane_Type', 'Property_Shape', 'Land_Outline',
       'Utility_Type', 'Lot_Configuration', 'Property_Slope',
'Neighborhood',
       'Condition1', 'Condition2', 'House_Type', 'House_Design',
       'Overall_Material', 'House_Condition', 'Construction_Year',
       'Remodel_Year', 'Roof_Design', 'Roof_Quality', 'Exterior1st',
       'Exterior2nd', 'Brick_Veneer_Type', 'Brick_Veneer_Area',
       'Exterior_Material', 'Exterior_Condition', 'Foundation_Type',
       'Basement_Height', 'Basement_Condition', 'Exposure_Level',
       'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2',
'BsmtUnfSF',
       'Total_Basement_Area', 'Heating_Type', 'Heating_Quality',
       'Air_Conditioning', 'Electrical_System', 'First_Floor_Area',
       'Second_Floor_Area', 'LowQualFinSF', 'Grade_Living_Area',
       'Underground_Full_Bathroom', 'Underground_Half_Bathroom',
       'Full_Bathroom_Above_Grade', 'Half_Bathroom_Above_Grade',
       'Bedroom_Above_Grade', 'Kitchen_Above_Grade',
'Kitchen_Quality',
       'Rooms_Above_Grade', 'Functional_Rate', 'Fireplaces',
       'Fireplace_Quality', 'Garage', 'Garage_Built_Year',
```

```
       'Garage_Finish_Year', 'Garage_Size', 'Garage_Area',
'Garage_Quality',
       'Garage_Condition', 'Pavedd_Drive', 'W_Deck_Area',
'Open_Lobby_Area',
       'Enclosed_Lobby_Area', 'Three_Season_Lobby_Area',
'Screen_Lobby_Area',
       'Pool_Area', 'Pool_Quality', 'Fence_Quality',
'Miscellaneous_Feature',
       'Miscellaneous_Value', 'Month_Sold', 'Year_Sold', 'Sale_Type',
       'Sale_Condition', 'Sale_Price'],
      dtype='object')

columns_to_drop =
['Lane_Type','Brick_Veneer_Type','Fireplace_Quality']
df.drop(columns=columns_to_drop, inplace=True)

df.shape

(1459, 78)

df.head(15)

    Id  Building_Class Zoning_Class  Lot_Extent  Lot_Size Road_Type  \
0    1              60          RLD        65.0      8450     Paved
1    2              20          RLD        80.0      9600     Paved
2    3              60          RLD        68.0     11250     Paved
3    4              70          RLD        60.0      9550     Paved
4    5              60          RLD        84.0     14260     Paved
5    6              50          RLD        85.0     14115     Paved
6    7              20          RLD        75.0     10084     Paved
7    8              60          RLD         NaN     10382     Paved
8    9              50          RMD        51.0      6120     Paved
9   10             190          RLD        50.0      7420     Paved
10  11              20          RLD        70.0     11200     Paved
11  12              60          RLD        85.0     11924     Paved
12  13              20          RLD         NaN     12968     Paved
13  14              20          RLD        91.0     10652     Paved
14  15              20          RLD         NaN     10920     Paved

   Property_Shape Land_Outline Utility_Type Lot_Configuration  ...
Pool_Area  \
0             Reg          Lvl       AllPub                 I  ...
0
1             Reg          Lvl       AllPub              FR2P  ...
0
2             IR1          Lvl       AllPub                 I  ...
0
3             IR1          Lvl       AllPub                 C  ...
0
4             IR1          Lvl       AllPub              FR2P  ...
```

```
0
5              IR1          Lvl       AllPub                   I  ...
0
6              Reg          Lvl       AllPub                   I  ...
0
7              IR1          Lvl       AllPub                   C  ...
0
8              Reg          Lvl       AllPub                   I  ...
0
9              Reg          Lvl       AllPub                   C  ...
0
10             Reg          Lvl       AllPub                   I  ...
0
11             IR1          Lvl       AllPub                   I  ...
0
12             IR2          Lvl       AllPub                   I  ...
0
13             IR1          Lvl       AllPub                   I  ...
0
14             IR1          Lvl       AllPub                   C  ...
0

   Pool_Quality Fence_Quality Miscellaneous_Feature
Miscellaneous_Value  \
0            NaN           NaN                     NaN
0
1            NaN           NaN                     NaN
0
2            NaN           NaN                     NaN
0
3            NaN           NaN                     NaN
0
4            NaN           NaN                     NaN
0
5            NaN         MnPrv                    Shed
700
6            NaN           NaN                     NaN
0
7            NaN           NaN                    Shed
350
8            NaN           NaN                     NaN
0
9            NaN           NaN                     NaN
0
10           NaN           NaN                     NaN
0
11           NaN           NaN                     NaN
0
12           NaN           NaN                     NaN
```

```
0
13          NaN           NaN                   NaN
0
14          NaN          GdWo                   NaN
0

   Month_Sold  Year_Sold  Sale_Type  Sale_Condition  Sale_Price
0           2       2008         WD          Normal      208500
1           5       2007         WD          Normal      181500
2           9       2008         WD          Normal      223500
3           2       2006         WD         Abnorml      140000
4          12       2008         WD          Normal      250000
5          10       2009         WD          Normal      143000
6           8       2007         WD          Normal      307000
7          11       2009         WD          Normal      200000
8           4       2008         WD         Abnorml      129900
9           1       2008         WD          Normal      118000
10          2       2008         WD          Normal      129500
11          7       2006        New         Partial      345000
12          9       2008         WD          Normal      144000
13          8       2007        New         Partial      279500
14          5       2008         WD          Normal      157000

[15 rows x 78 columns]
```

```
null_counts = df.isnull().sum()
print("Columns with null values:")
print(null_counts[null_counts > 0])
```

```
Columns with null values:
Lot_Extent                259
Brick_Veneer_Area           8
Basement_Height            37
Basement_Condition         37
Exposure_Level             38
BsmtFinType1               37
BsmtFinType2               38
Electrical_System           1
Garage                     81
Garage_Built_Year          81
Garage_Finish_Year         81
Garage_Quality             81
Garage_Condition           81
Pool_Quality             1452
Fence_Quality            1178
Miscellaneous_Feature    1405
dtype: int64
```

```
columns_to_drop =
['Pool_Quality','Fence_Quality','Miscellaneous_Feature']
df.drop(columns=columns_to_drop, inplace=True)

null_counts = df.isnull().sum()
print("Columns with null values:")
print(null_counts[null_counts > 0])
```

```
Columns with null values:
Lot_Extent            259
Brick_Veneer_Area       8
Basement_Height        37
Basement_Condition     37
Exposure_Level         38
BsmtFinType1           37
BsmtFinType2           38
Electrical_System       1
Garage                 81
Garage_Built_Year      81
Garage_Finish_Year     81
Garage_Quality         81
Garage_Condition       81
dtype: int64
```

```
df['Garage_Built_Year']
```

```
0       2003.0
1       1976.0
2       2001.0
3       1998.0
4       2000.0
         ...
1454    2004.0
1455    1999.0
1456    1978.0
1457    1941.0
1458    1950.0
Name: Garage_Built_Year, Length: 1459, dtype: float64
```

```
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

```
garage_counts = df['Garage'].value_counts()
garage_counts
```

```
Garage
Attchd     869
Detchd     387
BuiltIn     88
Basment     19
```

```
CarPort      9
2TFes        5
2Types       1
Name: count, dtype: int64
```

```
Garage_Condition_counts = df['Garage_Condition'].value_counts()
Garage_Condition_counts
```

```
Garage_Condition
TA    1325
Fa      35
Gd       9
Po       7
Ex       2
Name: count, dtype: int64
```

```
columns_of_interest = ['Garage',
'Garage_Built_Year','Garage_Quality','Garage_Condition','Sale_Price']
df[columns_of_interest].head(20)
```

```
     Garage  Garage_Built_Year Garage_Quality Garage_Condition
Sale_Price
0    Attchd             2003.0             TA                TA
208500
1    Attchd             1976.0             TA                TA
181500
2    Attchd             2001.0             TA                TA
223500
3    Detchd             1998.0             TA                TA
140000
4    Attchd             2000.0             TA                TA
250000
5    Attchd             1993.0             TA                TA
143000
6    Attchd             2004.0             TA                TA
307000
7    Attchd             1973.0             TA                TA
200000
8    Detchd             1931.0             Fa                TA
129900
9    Attchd             1939.0             Gd                TA
118000
10   Detchd             1965.0             TA                TA
129500
11  BuiltIn             2005.0             TA                TA
345000
12   Detchd             1962.0             TA                TA
144000
13   Attchd             2006.0             TA                TA
279500
```

```
14   Attchd             1960.0             TA                TA
157000
15   Detchd             1991.0             TA                TA
132000
16   Attchd             1970.0             TA                TA
149000
17  CarPort             1967.0             TA                TA
90000
18   Detchd             2004.0             TA                TA
159000
19   Attchd             1958.0             TA                TA
139000
```

```
columns_of_interest = ['Garage',
'Garage_Finish_Year','Garage_Built_Year','Garage_Quality','Garage_Cond
ition']

for column in columns_of_interest:
    if column in df.columns:
        mode_value = df[column].mode()[0]
        df[column].fillna(mode_value, inplace=True)

print(df[columns_of_interest].head())
```

```
   Garage Garage_Finish_Year  Garage_Built_Year Garage_Quality  \
0  Attchd                RFn             2003.0             TA
1  Attchd                RFn             1976.0             TA
2  Attchd                RFn             2001.0             TA
3  Detchd                Unf             1998.0             TA
4  Attchd                RFn             2000.0             TA

  Garage_Condition
0               TA
1               TA
2               TA
3               TA
4               TA

C:\Users\user\AppData\Local\Temp\ipykernel_9876\960025193.py:6:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

```
  df[column].fillna(mode_value, inplace=True)

null_counts = df.isnull().sum()
print("Columns with null values:")
print(null_counts[null_counts > 0])

Columns with null values:
Lot_Extent            259
Brick_Veneer_Area       8
Basement_Height        37
Basement_Condition     37
Exposure_Level         38
BsmtFinType1           37
BsmtFinType2           38
Electrical_System       1
dtype: int64

Lot_Extend_counts = df['Lot_Extent'].value_counts()
Lot_Extend_counts

Lot_Extent
60.0     143
70.0      70
80.0      69
50.0      57
75.0      52
         ...
137.0      1
141.0      1
38.0       1
140.0      1
46.0       1
Name: count, Length: 110, dtype: int64

columns_to_drop = ['Lot_Extent']
df.drop(columns=columns_to_drop, inplace=True)

Basement_Height_counts = df['Basement_Height'].value_counts()
Basement_Height_counts

Basement_Height
TA    648
Gd    618
Ex    121
Fa     35
Name: count, dtype: int64

columns_of_interest =
['Brick_Veneer_Area','Basement_Height','Basement_Condition','Exposure_
Level','BsmtFinType1','BsmtFinType2','Electrical_System']
```

```python
for column in columns_of_interest:
    if column in df.columns:
        mode_value = df[column].mode()[0]
        df[column].fillna(mode_value, inplace=True)

print(df[columns_of_interest].head(30))
```

```
    Brick_Veneer_Area Basement_Height Basement_Condition
Exposure_Level  \
0               196.0              Gd                  TA
No
1                 0.0              Gd                  TA
Gd
2               162.0              Gd                  TA
Mn
3                 0.0              TA                  Gd
No
4               350.0              Gd                  TA
Av
5                 0.0              Gd                  TA
No
6               186.0              Ex                  TA
Av
7               240.0              Gd                  TA
Mn
8                 0.0              TA                  TA
No
9                 0.0              TA                  TA
No
10                0.0              TA                  TA
No
11              286.0              Ex                  TA
No
12                0.0              TA                  TA
No
13              306.0              Gd                  TA
Av
14              212.0              TA                  TA
No
15                0.0              TA                  TA
No
16              180.0              TA                  TA
No
17                0.0              TA                  TA
No
18                0.0              TA                  TA
No
19                0.0              TA                  TA
No
```

```
20                380.0               Ex                   TA
Av
21                  0.0               TA                   TA
No
22                281.0               Gd                   TA
No
23                  0.0               Gd                   TA
No
24                  0.0               TA                   TA
Mn
25                640.0               Gd                   TA
No
26                  0.0               TA                   TA
Mn
27                200.0               Ex                   TA
No
28                  0.0               TA                   TA
Gd
29                  0.0               TA                   TA
No
```

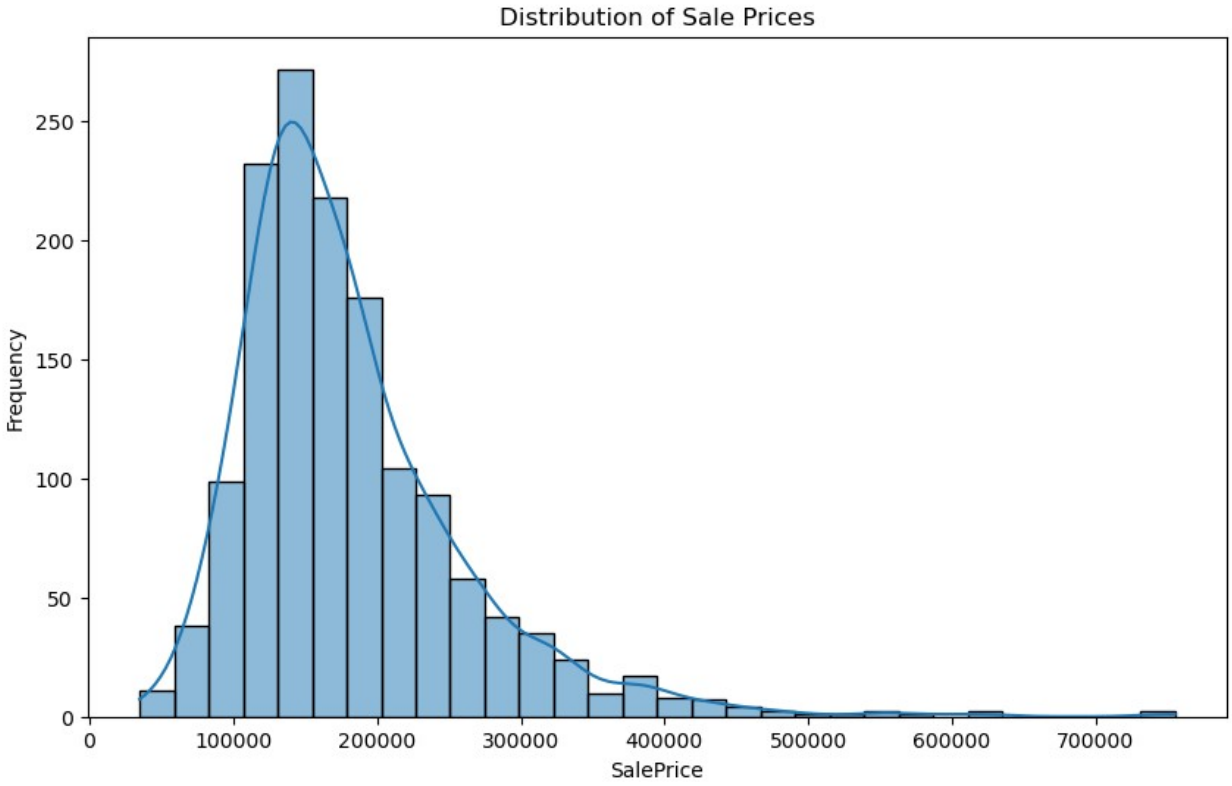
	BsmtFinType1	BsmtFinType2	Electrical_System
0	GLQ	Unf	SBrkr
1	ALQ	Unf	SBrkr
2	GLQ	Unf	SBrkr
3	ALQ	Unf	SBrkr
4	GLQ	Unf	SBrkr
5	GLQ	Unf	SBrkr
6	GLQ	Unf	SBrkr
7	ALQ	BLQ	SBrkr
8	Unf	Unf	FuseF
9	GLQ	Unf	SBrkr
10	Rec	Unf	SBrkr
11	GLQ	Unf	SBrkr
12	ALQ	Unf	SBrkr
13	Unf	Unf	SBrkr
14	BLQ	Unf	SBrkr
15	Unf	Unf	FuseA
16	ALQ	Unf	SBrkr
17	Unf	Unf	SBrkr
18	GLQ	Unf	SBrkr
19	LwQ	Unf	SBrkr
20	Unf	Unf	SBrkr
21	Unf	Unf	FuseF
22	Unf	Unf	SBrkr
23	GLQ	Unf	SBrkr
24	Rec	ALQ	SBrkr
25	Unf	Unf	SBrkr
26	BLQ	Rec	SBrkr

```
27          GLQ          Unf             SBrkr
28          BLQ          Unf             SBrkr
29          Unf          Unf             SBrkr

C:\Users\user\AppData\Local\Temp\ipykernel_9876\2464482277.py:6:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[column].fillna(mode_value, inplace=True)
```

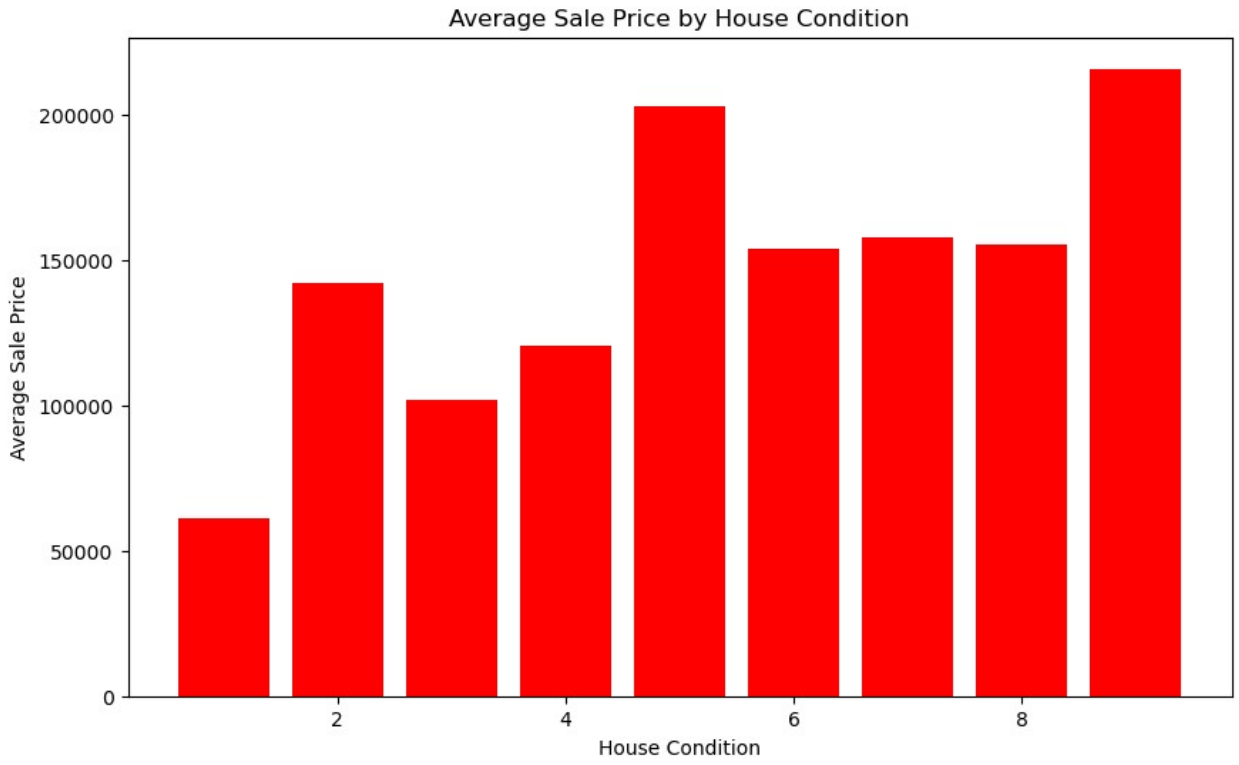
```
null_counts = df.isnull().sum()
print("Columns with null values:")
print(null_counts[null_counts > 0])
```

```
Columns with null values:
Series([], dtype: int64)
```

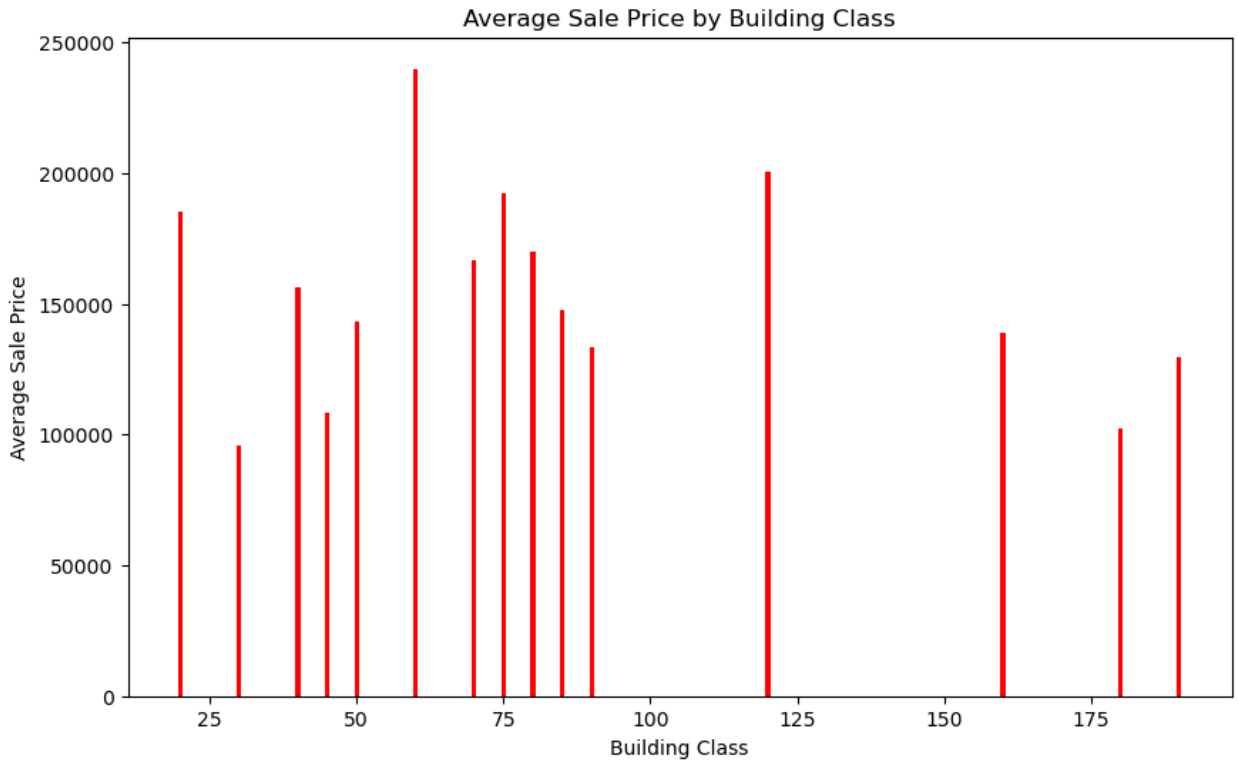
```
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.histplot(df['Sale_Price'], bins=30, kde=True)
plt.title('Distribution of Sale Prices')
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
plt.show()
```



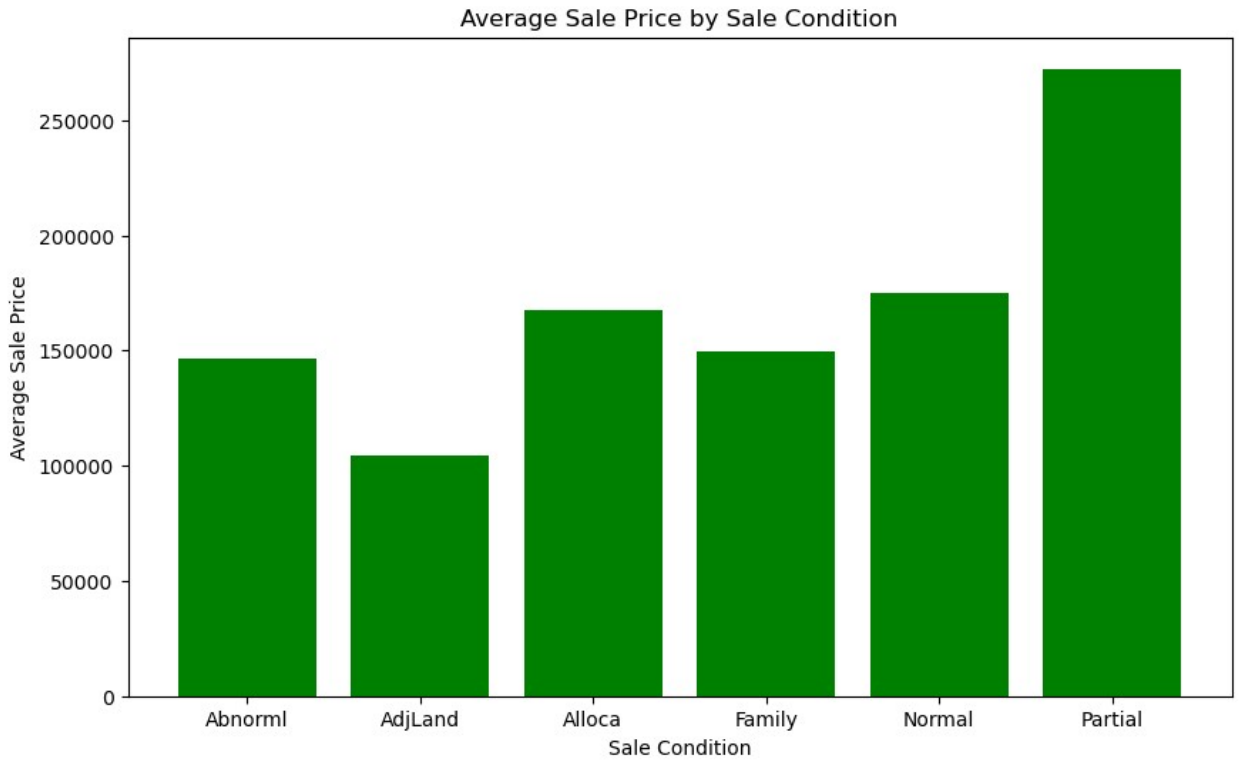

```
average_sale_price = df.groupby('House_Condition')
['Sale_Price'].mean().reset_index()

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(average_sale_price['House_Condition'],
average_sale_price['Sale_Price'], color='red')
plt.title('Average Sale Price by House Condition')
plt.xlabel('House Condition')
plt.ylabel('Average Sale Price')
plt.show()
```




```
average_sale_price = df.groupby('Building_Class')
['Sale_Price'].mean().reset_index()

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(average_sale_price['Building_Class'],
average_sale_price['Sale_Price'], color='red')
plt.title('Average Sale Price by Building Class')
plt.xlabel('Building Class')
plt.ylabel('Average Sale Price')
plt.show()
```

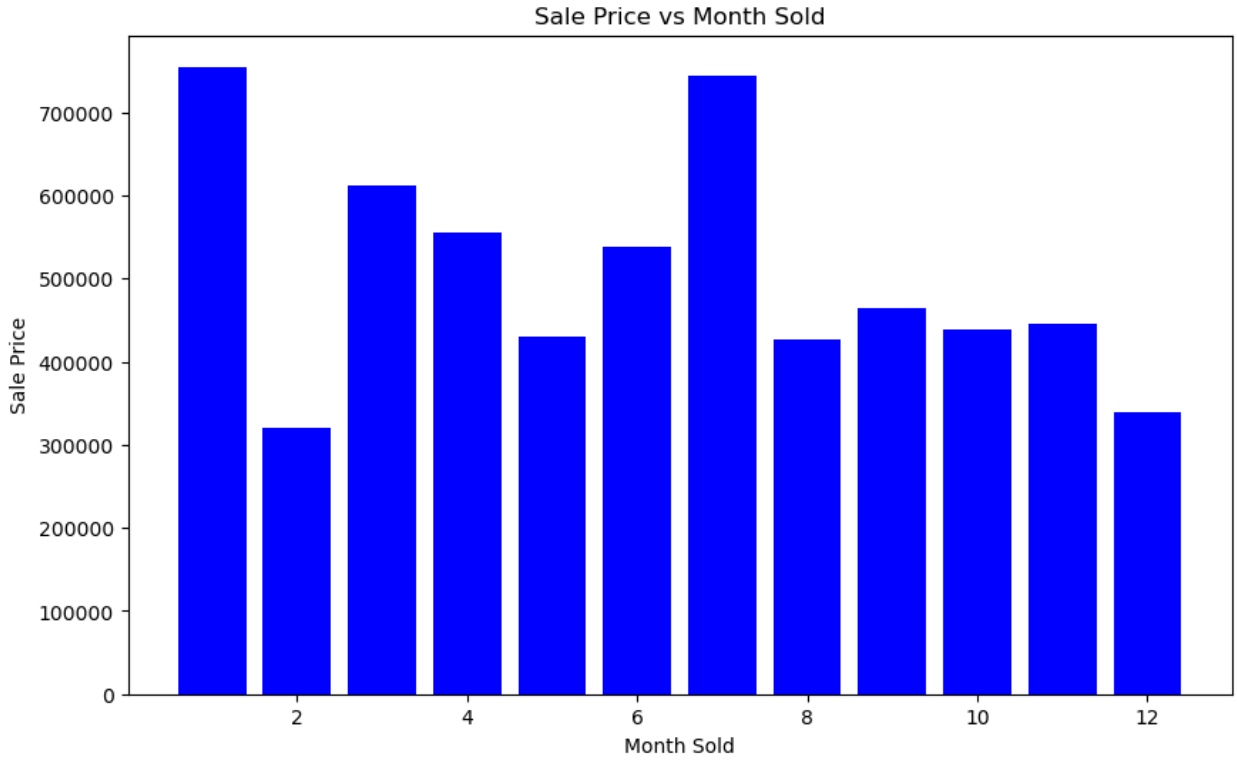



```
average_sale_price = df.groupby('Sale_Condition')
['Sale_Price'].mean().reset_index()

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(average_sale_price['Sale_Condition'],
average_sale_price['Sale_Price'], color='green')
plt.title('Average Sale Price by Sale Condition')
plt.xlabel('Sale Condition')
plt.ylabel('Average Sale Price')
plt.show()
```




```
plt.figure(figsize=(10, 6))
plt.scatter(df['Lot_Size'], df['Sale_Price'], alpha=0.5, color='r')
plt.title('Sale Price vs Lot Size')
plt.xlabel('Lot Size')
plt.ylabel('Sale Price')
plt.grid(True)
plt.show()
```


```
plt.figure(figsize=(10, 6))
plt.bar(df['Month_Sold'], df['Sale_Price'], color='b')
plt.title('Sale Price vs Month Sold')
plt.xlabel('Month Sold')
plt.ylabel('Sale Price')
plt.show()
```


Sale Price vs Month Sold
Sale Price
0
100000
200000
300000
400000
500000
600000
700000
2
4
6
8
10
12
Month Sold