In [1]:
```
#Importing relevant functions
import statsmodels.api as sm
import statsmodels as sms
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import os
%matplotlib inline
```

In [2]:
```
!pip install wooldridge
```

```
Requirement already satisfied: wooldridge in ./anaconda3/lib/python3.11/site-packages (0.4.4)
Requirement already satisfied: pandas in ./anaconda3/lib/python3.11/site-packages (from wooldridge) (2.1.
1)
Requirement already satisfied: numpy>=1.23.2 in ./anaconda3/lib/python3.11/site-packages (from pandas->woo
ldridge) (1.24.3)
Requirement already satisfied: python-dateutil>=2.8.2 in ./anaconda3/lib/python3.11/site-packages (from pa
ndas->wooldridge) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/site-packages (from pandas->wool
dridge) (2022.7)
Requirement already satisfied: tzdata>=2022.1 in ./anaconda3/lib/python3.11/site-packages (from pandas->wo
oldridge) (2023.3)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-packages (from python-dateutil>
=2.8.2->pandas->wooldridge) (1.16.0)
```

I. Panel Data Models

In [11]:
```
import wooldridge as woo
data1 = woo.data('KIELMC')
```

In [12]:
```
data1 = pd.DataFrame(data1)
data1
```

Out[12]:

	year	age	agesq	nbh	cbd	intst	lintst	price	rooms	area	...	lprice	y81	larea	lland	y81
0	1978	48	2304.0	4	3000.0	1000.0	6.9078	60000.0	7	1660	...	11.002100	0	7.414573	8.429017	0.000
1	1978	83	6889.0	4	4000.0	1000.0	6.9078	40000.0	6	2612	...	10.596635	0	7.867871	9.032409	0.000
2	1978	58	3364.0	4	4000.0	1000.0	6.9078	34000.0	6	1144	...	10.434115	0	7.042286	8.517193	0.000
3	1978	11	121.0	4	4000.0	1000.0	6.9078	63900.0	5	1136	...	11.065075	0	7.035269	9.210340	0.000
4	1978	48	2304.0	4	4000.0	2000.0	7.6009	44000.0	5	1868	...	10.691945	0	7.532624	9.210340	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
316	1981	25	625.0	0	10000.0	10000.0	9.2103	60000.0	6	1388	...	11.002100	1	7.235619	8.922658	9.735
317	1981	0	0.0	6	24000.0	26000.0	10.1660	144500.0	7	3140	...	11.881035	1	8.051978	10.795219	10.126
318	1981	20	400.0	0	15000.0	15000.0	9.6158	97000.0	6	1296	...	11.482467	1	7.167038	10.711458	9.966
319	1981	19	361.0	0	14000.0	15000.0	9.6158	95000.0	5	1148	...	11.461632	1	7.045776	10.696073	9.928
320	1981	0	0.0	0	13000.0	13000.0	9.4727	204080.0	7	2261	...	12.226268	1	7.723562	10.724698	9.873

321 rows × 25 columns

In [13]:
```
#Dropping log columns to clean the model

df_0= data1.drop(['lintst', 'lprice', 'ldist','larea','lland','lrprice'], axis=1)
df_0.head()
```

Out[13]:

	year	age	agesq	nbh	cbd	intst	price	rooms	area	land	baths	dist	wind	y81	y81ldist	lintstsq	near
0	1978	48	2304.0	4	3000.0	1000.0	60000.0	7	1660	4578.0	1	10700.0	3	0	0.0	47.717705	
1	1978	83	6889.0	4	4000.0	1000.0	40000.0	6	2612	8370.0	2	11000.0	3	0	0.0	47.717705	
2	1978	58	3364.0	4	4000.0	1000.0	34000.0	6	1144	5000.0	1	11500.0	3	0	0.0	47.717705	
3	1978	11	121.0	4	4000.0	1000.0	63900.0	5	1136	10000.0	1	11900.0	3	0	0.0	47.717705	
4	1978	48	2304.0	4	4000.0	2000.0	44000.0	5	1868	10000.0	1	12100.0	3	0	0.0	57.773682	

Question 1

The data is a panel dataset, as the years range from 1978 to 1981.

Discussing the data:

Temporal Aspect: The data spans multiple years (1978 to 1981). Consideration of time trends and changes over the years may be important.

Property Characteristics: Variables such as age, rooms, and area provide insights into property attributes. rooms and area might be crucial determinants of property prices.

Interest Rate and Central Business District: intst and cbd may indicate economic factors affecting property markets. The central business district (cbd) could influence property prices significantly.

Neighborhood Factors: The variable nbh suggests the presence of neighborhood-related information. Understanding its impact on property prices could be essential.

Categorical Variables: y81 and nearinc seem to be binary indicators, possibly capturing effects related to the year 1981 and proximity to income-generating areas.

Economic/Finance/Business Question:

The specific question we're trying to answer depends on our goals and the context of the data. However, potential questions might include:

What are the key determinants of property prices in this dataset?

How do interest rates and neighborhood characteristics impact property values?

Are there differences in property prices between the years 1981 and earlier years (y81)?

How well can we predict property prices using the provided features?

What factors contribute most to property prices?

How do interest rates and neighborhood characteristics impact property values?

How do interest rates and neighborhood characteristics impact property values?

Are there discernible trends or differences between the years 1978 and 1981?

Does the proximity to income-generating areas influence property prices?

Question 2

Descriptive Analysis

In [14]:
```
#Checking if there is any Null Value
print(df_0.isnull().values.any())
#Count the number of many missing obs per variable (if any)
print(df_0.isnull().sum())
```

```
False
year        0
age         0
agesq       0
nbh         0
cbd         0
intst       0
price       0
rooms       0
area        0
land        0
baths       0
dist        0
wind        0
y81         0
y81ldist    0
lintstsq    0
nearinc     0
y81nrinc    0
rprice      0
dtype: int64
```

There are no null values in the dataset.

In [15]:
```
df_0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 321 entries, 0 to 320
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   year      321 non-null    int64  
 1   age       321 non-null    int64  
 2   agesq     321 non-null    float64
 3   nbh       321 non-null    int64  
 4   cbd       321 non-null    float64
 5   intst     321 non-null    float64
 6   price     321 non-null    float64
 7   rooms     321 non-null    int64  
 8   area      321 non-null    int64  
 9   land      321 non-null    float64
 10  baths     321 non-null    int64  
 11  dist      321 non-null    float64
 12  wind      321 non-null    int64  
 13  y81       321 non-null    int64  
 14  y81ldist  321 non-null    float64
 15  lintstsq  321 non-null    float64
 16  nearinc   321 non-null    int64  
 17  y81nrinc  321 non-null    int64  
 18  rprice    321 non-null    float64
dtypes: float64(9), int64(10)
memory usage: 47.8 KB
```

In [16]:
```
#Part 1: Summary Statistics
df_0.describe()
```

Out[16]:

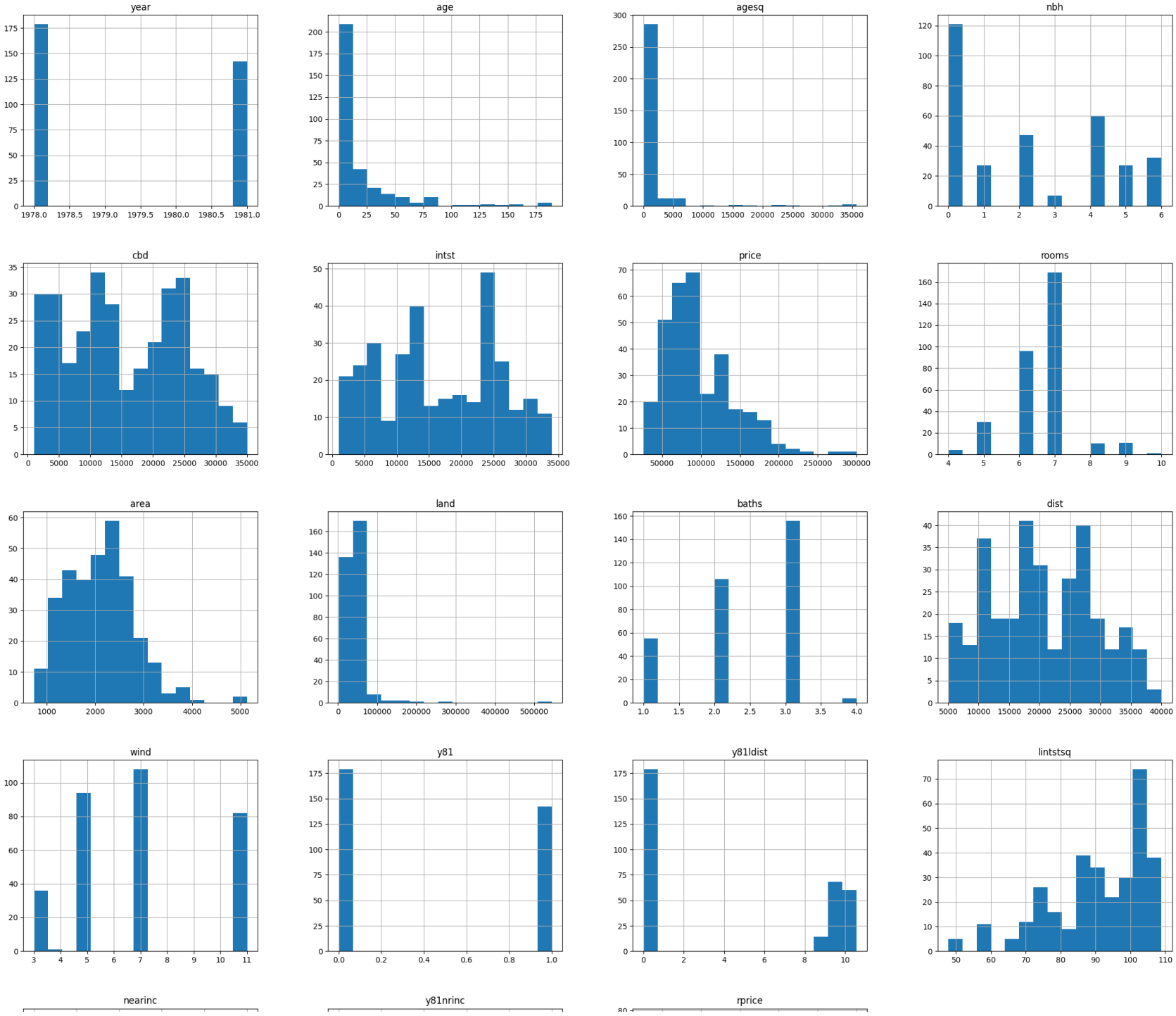
	year	age	agesq	nbh	cbd	intst	price	rooms	ar
count	321.000000	321.000000	321.000000	321.000000	321.000000	321.000000	321.000000	321.000000	321.0000
mean	1979.327103	18.009346	1381.566978	2.208723	15822.429907	16442.367601	96100.660436	6.585670	2106.7289
std	1.492329	32.565845	4801.788757	2.164353	8967.106296	9033.130652	43223.728867	0.901204	694.9579
min	1978.000000	0.000000	0.000000	0.000000	1000.000000	1000.000000	26000.000000	4.000000	735.0000
25%	1978.000000	0.000000	0.000000	0.000000	9000.000000	9000.000000	65000.000000	6.000000	1560.0000
50%	1978.000000	4.000000	16.000000	2.000000	14000.000000	16000.000000	85900.000000	7.000000	2056.0000
75%	1981.000000	22.000000	484.000000	4.000000	23000.000000	24000.000000	120000.000000	7.000000	2544.0000
max	1981.000000	189.000000	35721.000000	6.000000	35000.000000	34000.000000	300000.000000	10.000000	5136.0000

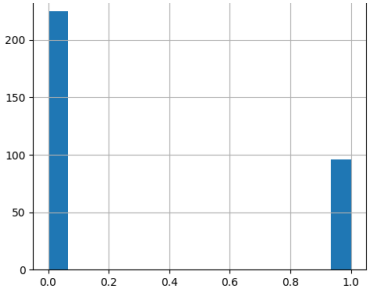
Interpretation:

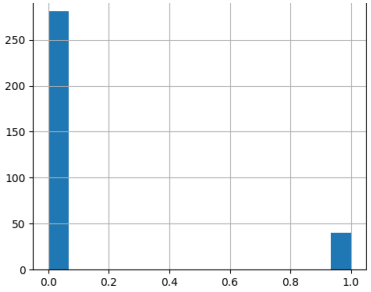
The dataset appears to be diverse, with a range of property ages, sizes, and prices.
Neighborhood values suggest a variety of locations.
Interest rates and land values show considerable variability.
The presence of indicator variables (y81 and nearinc) indicates potential categorical factors influencing
the dataset.
The range of property prices is substantial, reflecting different market segments.

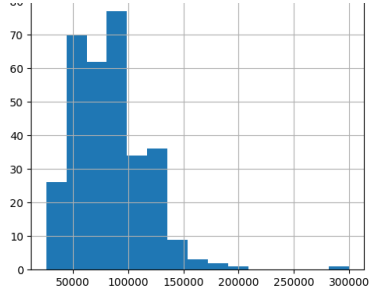
In [17]:
```
#Part 2: Plotting Histograms
df_0.hist(bins=15, figsize=(27, 28))
```

Out[17]:
```
array([[<Axes: title={'center': 'year'}>,
        <Axes: title={'center': 'age'}>,
        <Axes: title={'center': 'agesq'}>,
        <Axes: title={'center': 'nbh'}>],
       [<Axes: title={'center': 'cbd'}>,
        <Axes: title={'center': 'intst'}>,
        <Axes: title={'center': 'price'}>,
        <Axes: title={'center': 'rooms'}>],
       [<Axes: title={'center': 'area'}>,
        <Axes: title={'center': 'land'}>,
        <Axes: title={'center': 'baths'}>,
        <Axes: title={'center': 'dist'}>],
       [<Axes: title={'center': 'wind'}>,
        <Axes: title={'center': 'y81'}>,
        <Axes: title={'center': 'y81ldist'}>,
        <Axes: title={'center': 'lintstsq'}>],
       [<Axes: title={'center': 'nearinc'}>,
        <Axes: title={'center': 'y81nrinc'}>,
        <Axes: title={'center': 'rprice'}>, <Axes: >]], dtype=object)
```


year
age
agesq
nbh
cbd
intst
price
rooms
area
land
baths
dist
wind
y81
y81ldist
lintstsq
nearinc
y81nrinc
rprice







Analysis of the data through Histograms:

1.Most of the variables show skewness and have a long tail towards right.

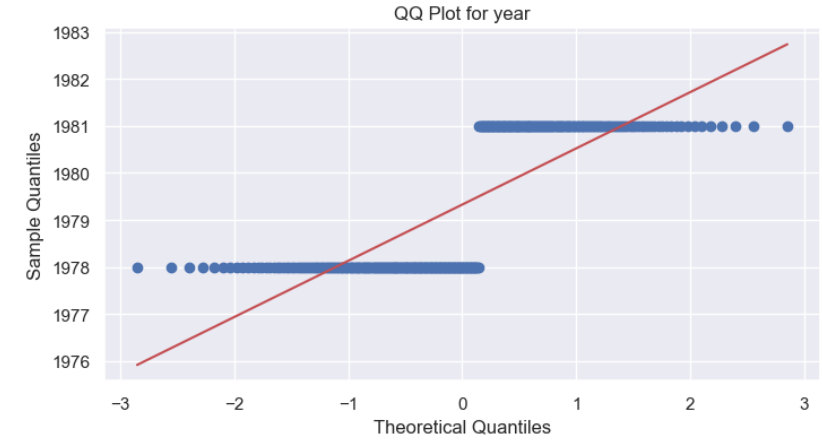
Variable that show Positively skewness are: age, cbd, price, area, land

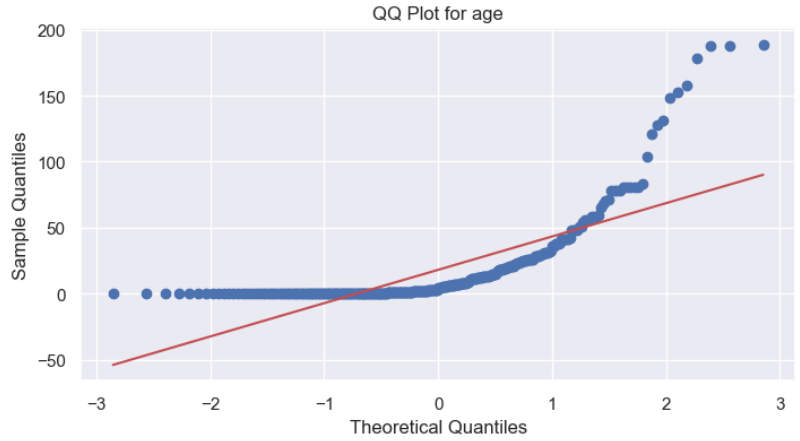
Variables that show Negatively skewness are: lintstsq

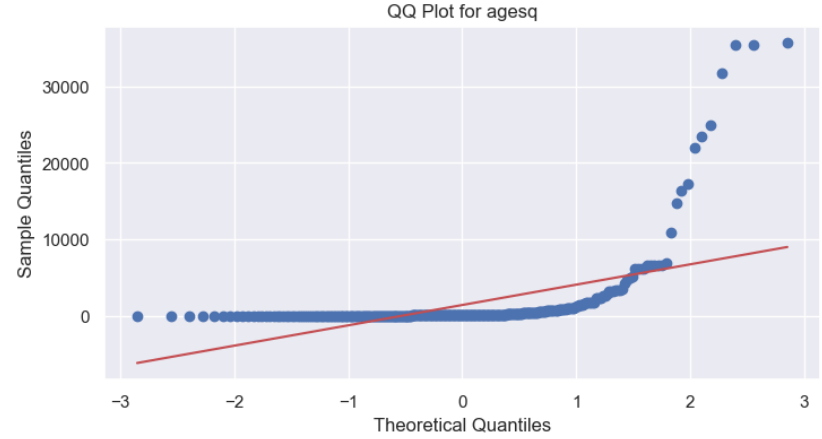
2.Outliers can also be observed in various variables such as area, land, rprice etc.

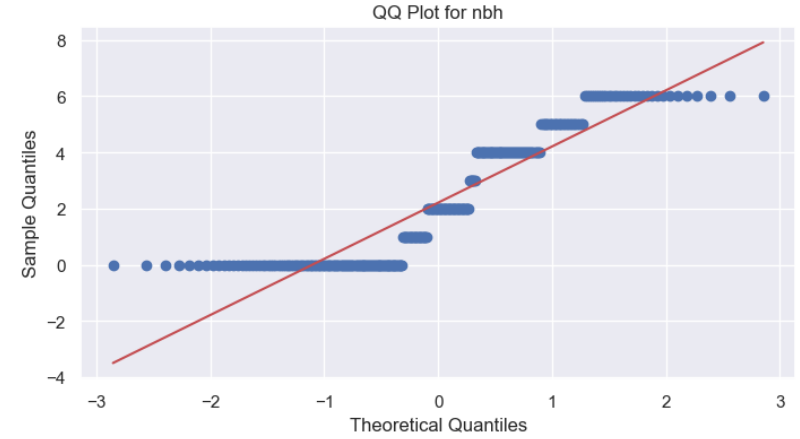
In [9]:
```
import scipy.stats as stats
```

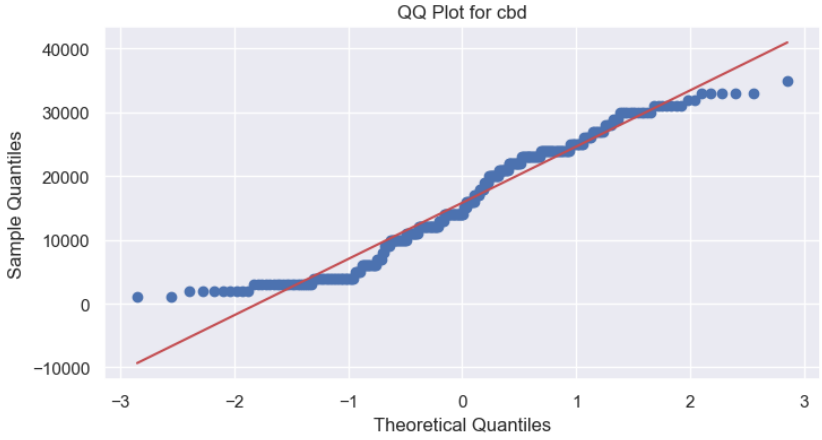
In [28]:
```
#Part 3: Plotting QQ Plots for each variable
selected_columns1 = df_0[["year","age","agesq","nbh","cbd","intst","price","rooms","area","land","baths","

# Create a figure with subplots for QQ plots
qqplot = len(selected_columns)
fig, axes = plt.subplots(10 ,2 , figsize=(15,40))
axes = axes.ravel()
#Step 1: Creating qq plot for each variable in the selected_columns list by using for function:
for x, y in enumerate(selected_columns):
    ax = axes[x]
    data = selected_columns[y]
# Create a QQ plot for the variable
    stats.probplot(data, dist="norm", plot=ax)
    ax.set_title(f'QQ Plot for {y}')
    ax.set_xlabel('Theoretical Quantiles')
    ax.set_ylabel('Sample Quantiles')
#Plot qq plots for each variable
plt.tight_layout()
plt.show()
```

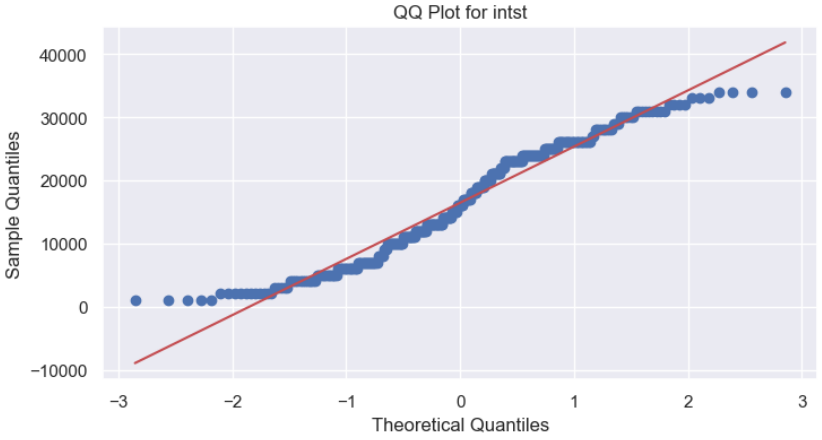

QQ Plot for year
Sample Quantiles
Theoretical Quantiles


QQ Plot for age
Sample Quantiles
Theoretical Quantiles

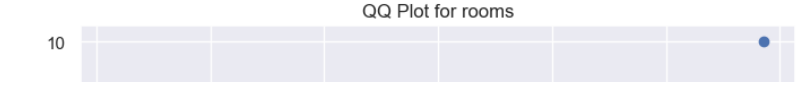

QQ Plot for agesq
Sample Quantiles
Theoretical Quantiles

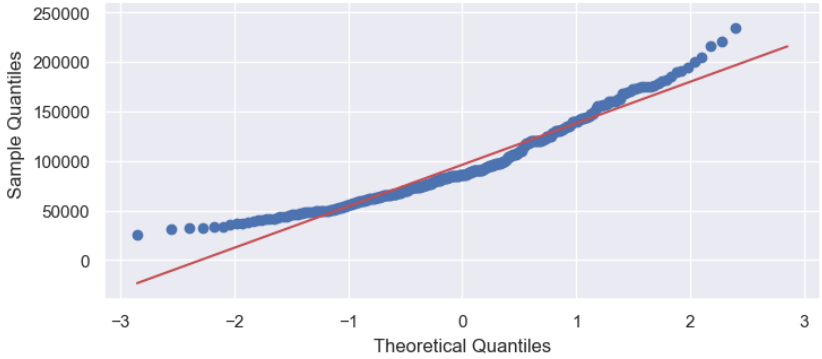

QQ Plot for nbh
Sample Quantiles
Theoretical Quantiles

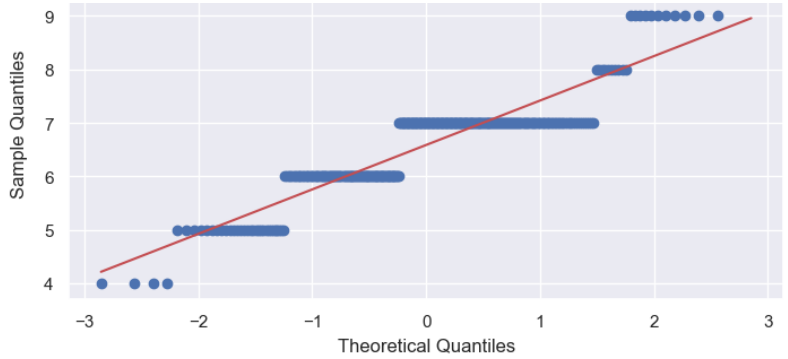

QQ Plot for cbd
Sample Quantiles
Theoretical Quantiles

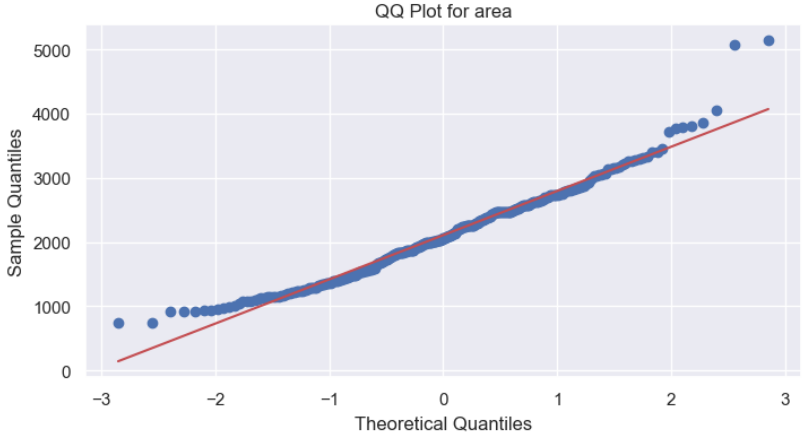

QQ Plot for intst
Sample Quantiles
Theoretical Quantiles

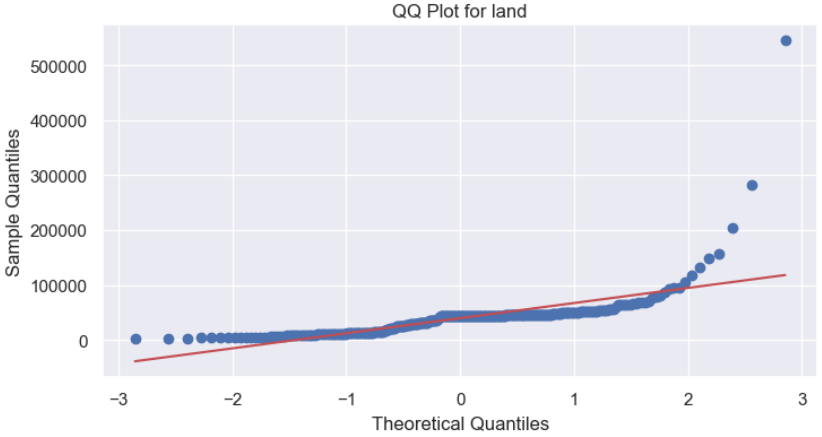

QQ Plot for price

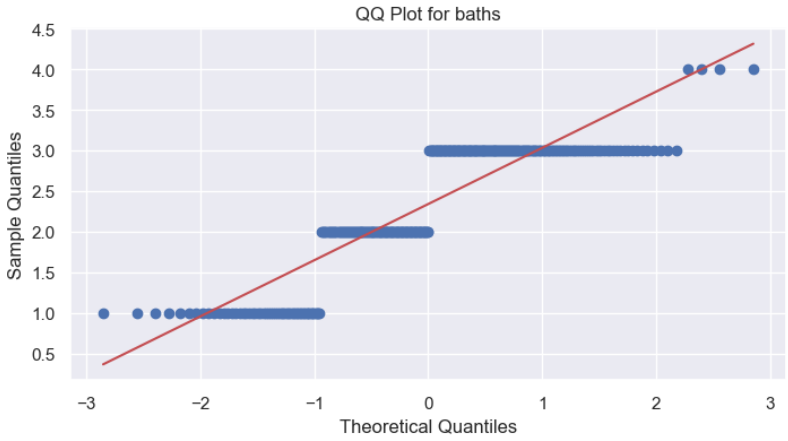

QQ Plot for rooms

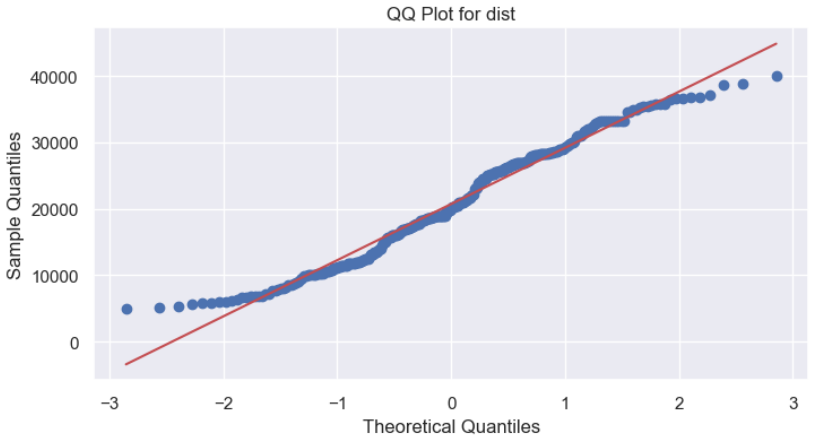

Sample Quantiles
Theoretical Quantiles

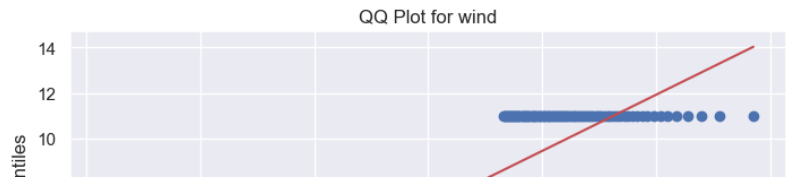

Sample Quantiles
Theoretical Quantiles

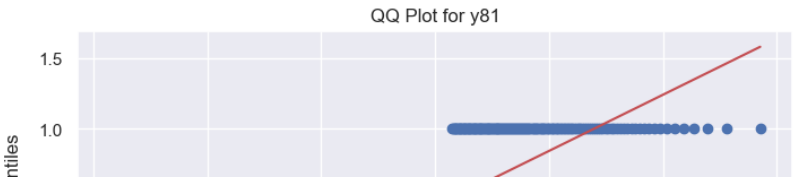

QQ Plot for area
Sample Quantiles
Theoretical Quantiles

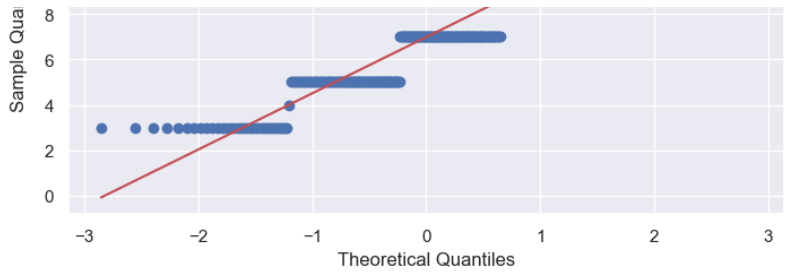

QQ Plot for land
Sample Quantiles
Theoretical Quantiles

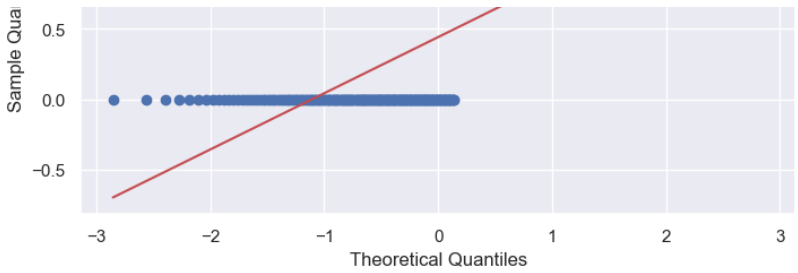

QQ Plot for baths
Sample Quantiles
Theoretical Quantiles


QQ Plot for dist
Sample Quantiles
Theoretical Quantiles

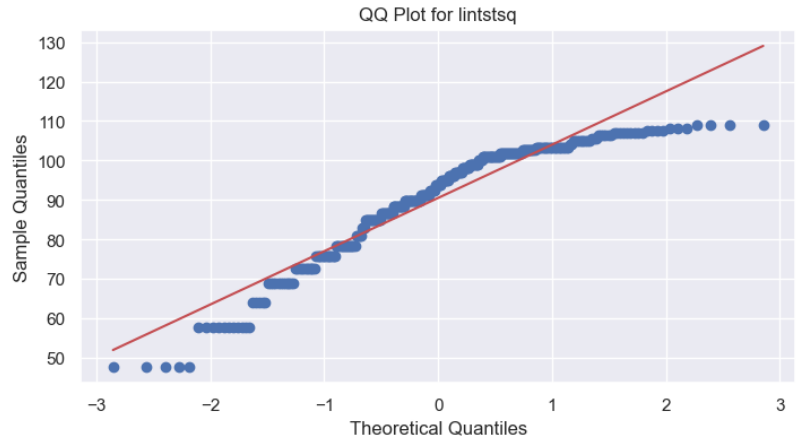

QQ Plot for wind
ntiles

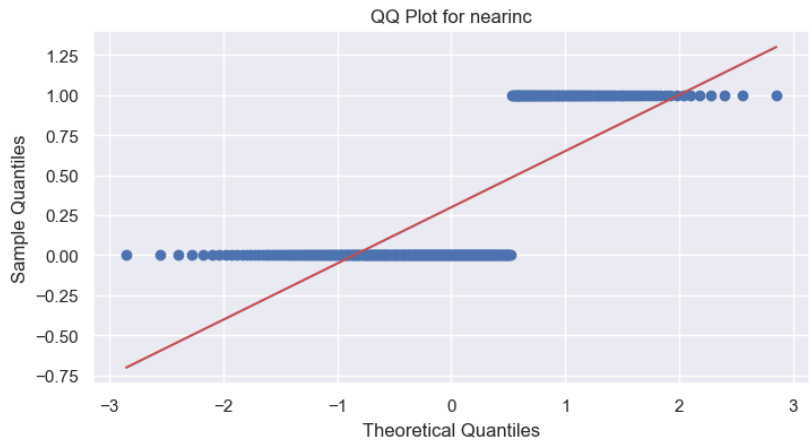

QQ Plot for y81
ntiles

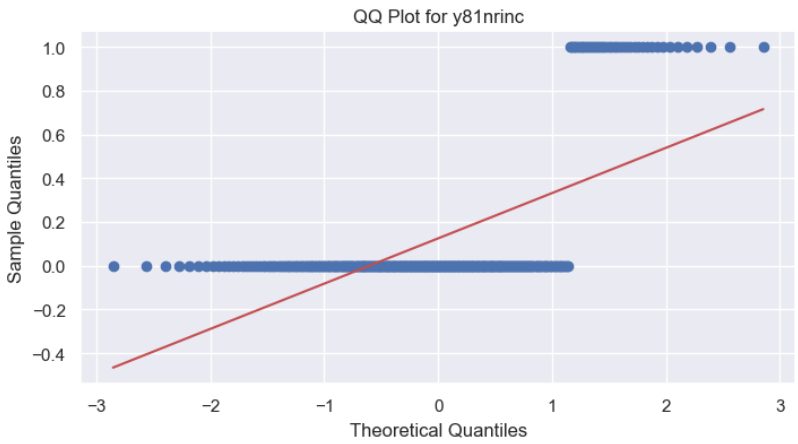

Sample Qua
Theoretical Quantiles

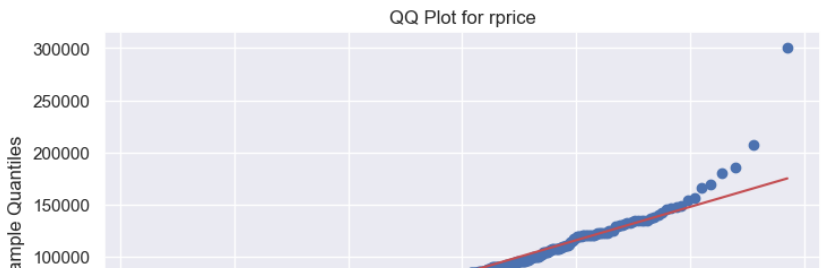

Sample Qua
Theoretical Quantiles

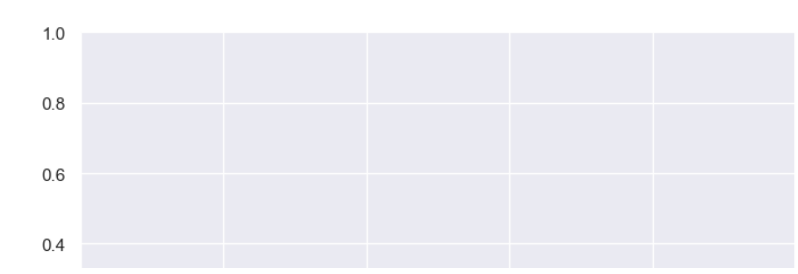

QQ Plot for y81ldist
Sample Quantiles
Theoretical Quantiles

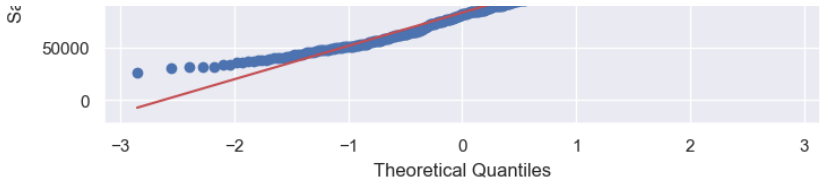

QQ Plot for lintstsq
Sample Quantiles
Theoretical Quantiles

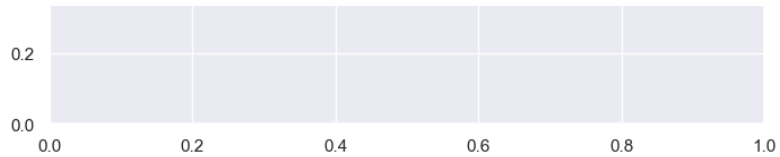

QQ Plot for nearinc
Sample Quantiles
Theoretical Quantiles


QQ Plot for y81nrinc
Sample Quantiles
Theoretical Quantiles


QQ Plot for rprice
ample Quantiles







Analysing QQ plots for each variable in the data set:

We have used this graphical tool used to assess whether the dataset follows a specific theoretical distribution, typically the normal distribution. If the points on the QQ plot fall along or near a straight line, it suggests that the data follows the theoretical distribution. If they deviate from a straight line, it indicates a departure from the theoretical distribution.

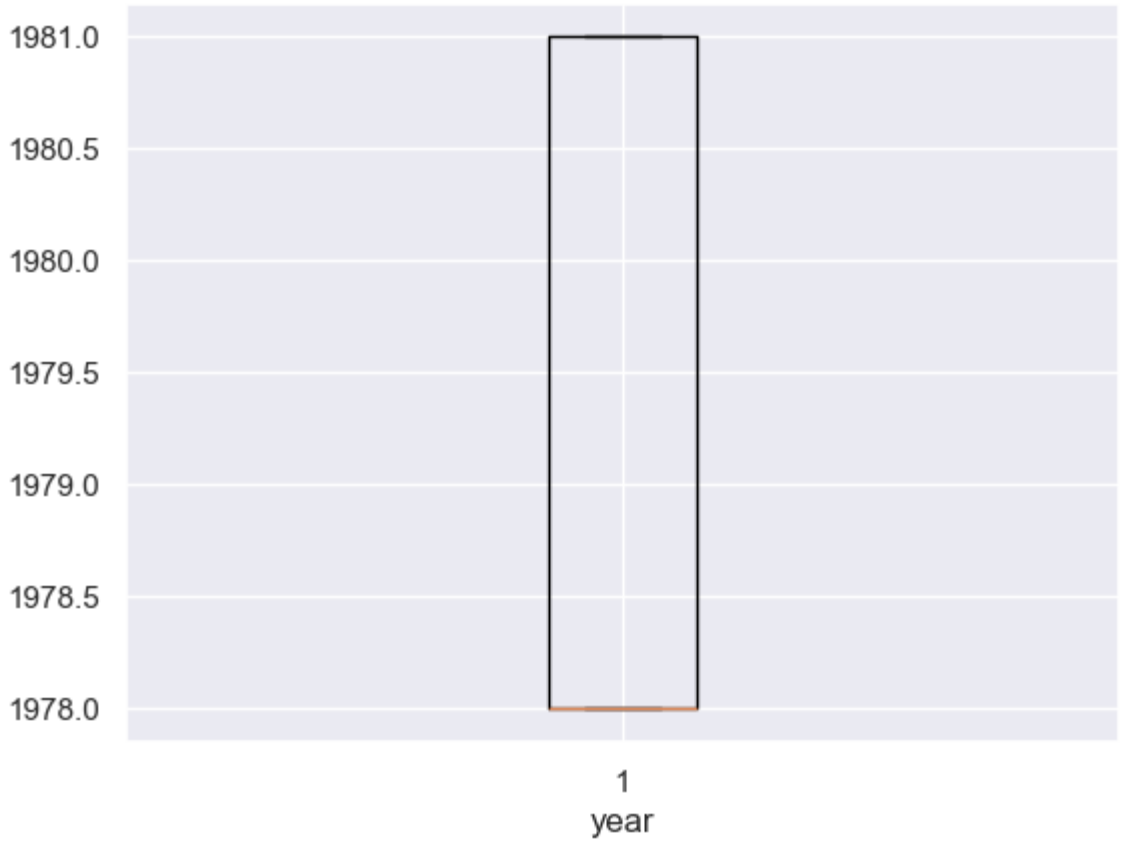
For cbd, intst, price, area, land, dist, the data follows the theoretical distribution.

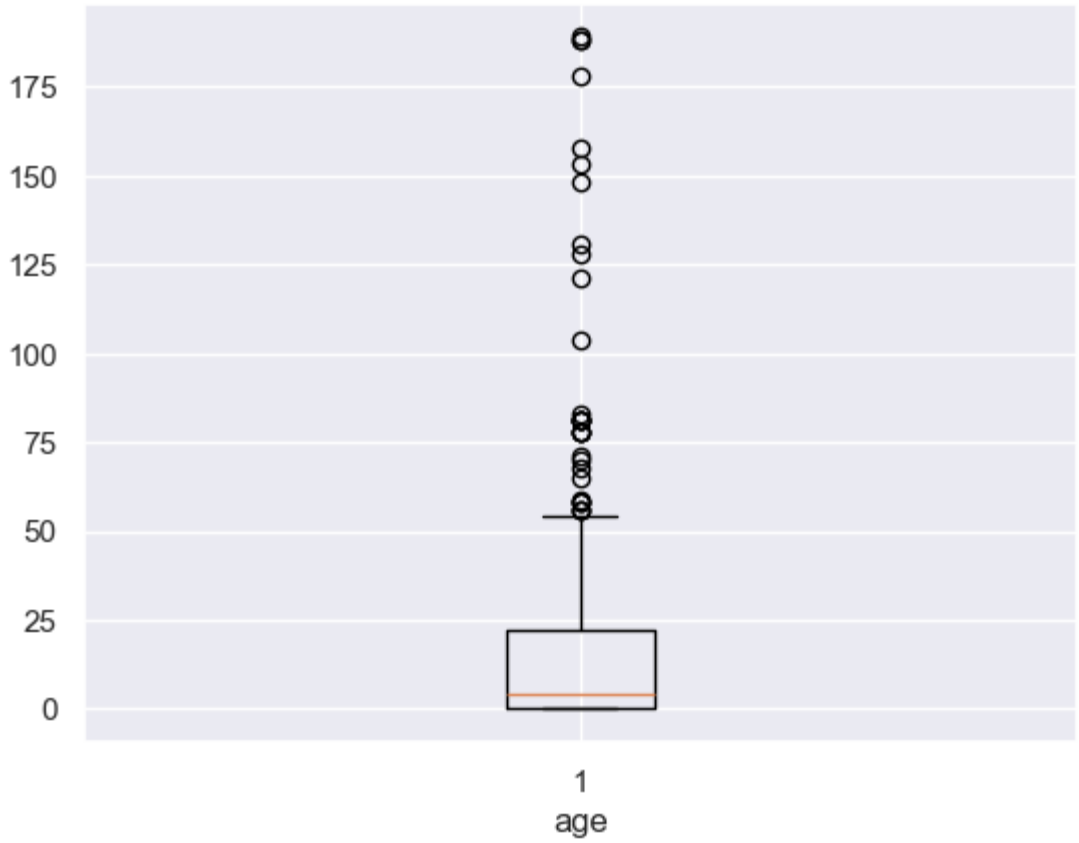
For other variables, it indicates non-normality.

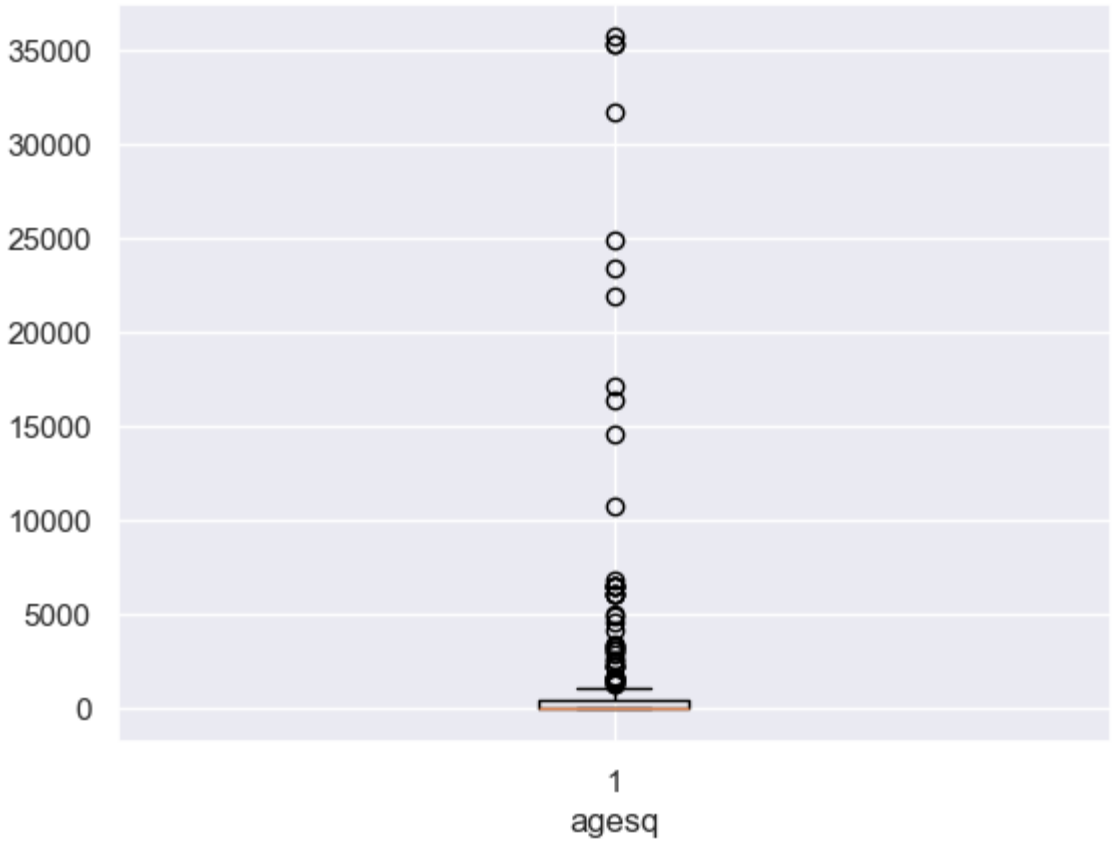
INTERPRETATION:

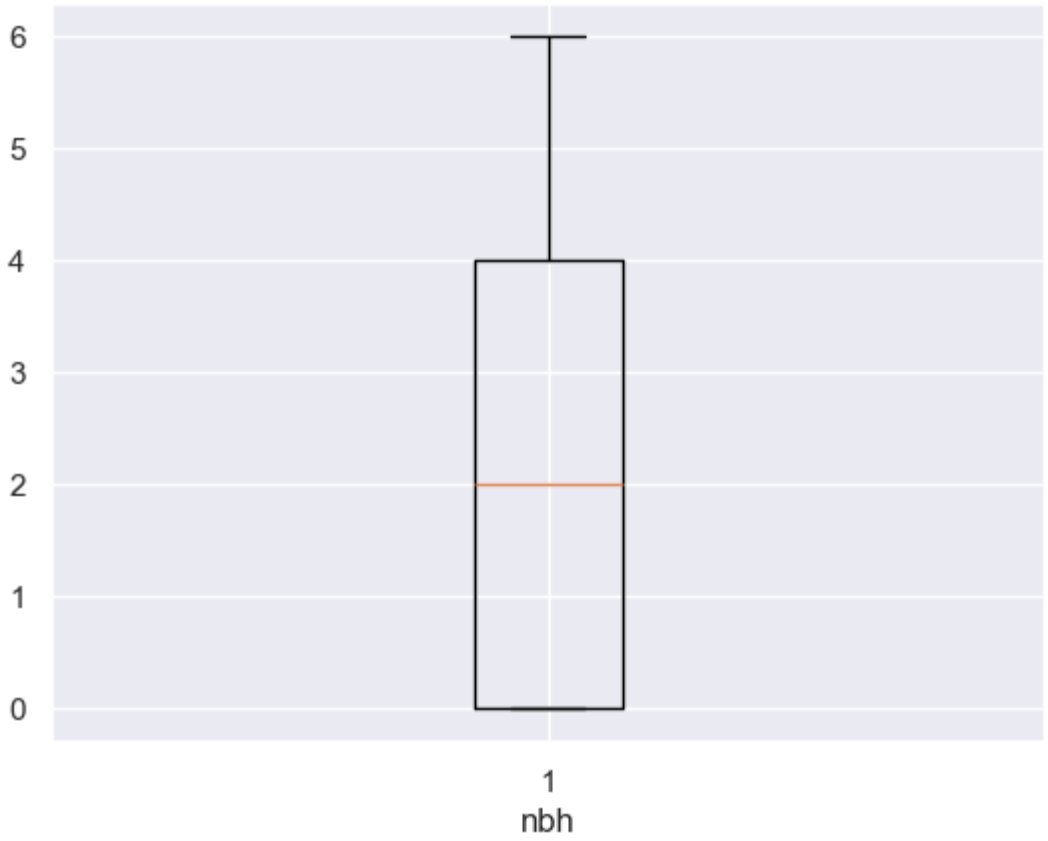
A probability plot shows how well your data is modelled by a particular distribution, typically the normal distribution. -If the points on the QQ plot fall along or near a straight line, it suggests that the data follows the theoretical distribution. -If they deviate from a straight line, it indicates a departure from the theoretical distribution.

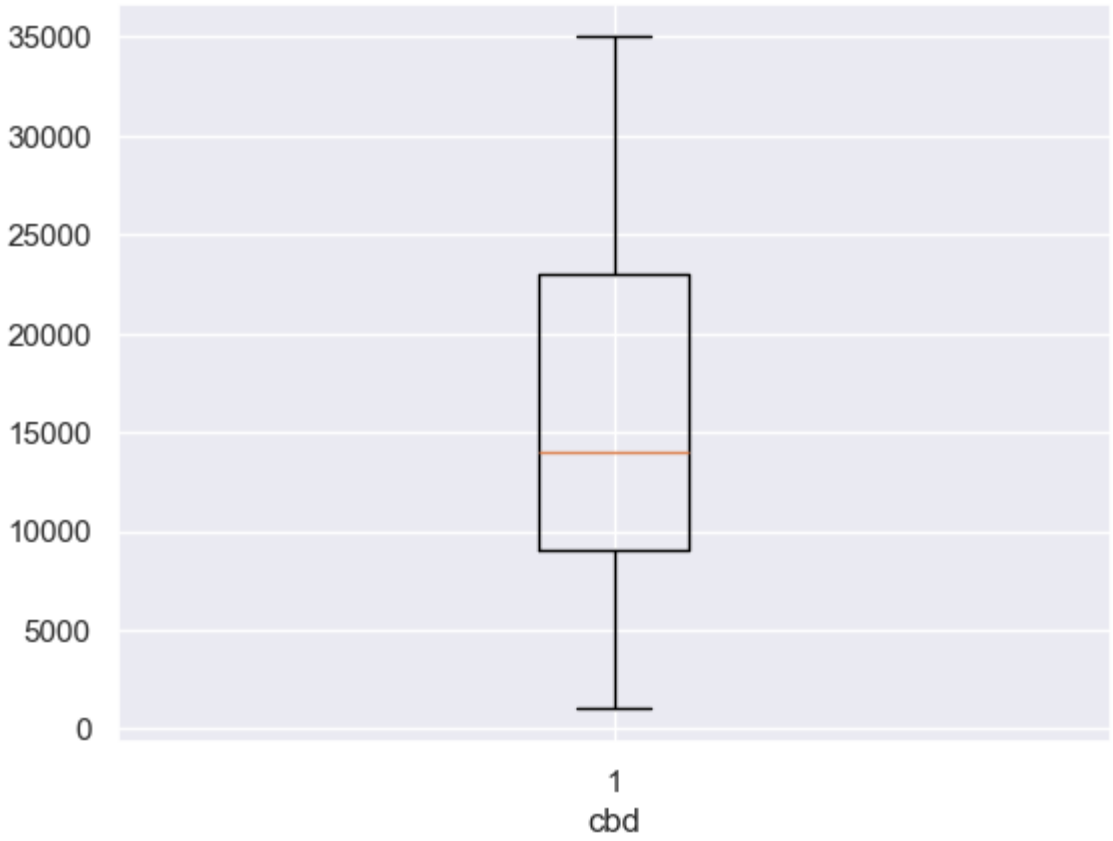
In [29]:
```
#Part4: Box Plots
selected_columns1 = df_0[["year","age","agesq","nbh","cbd","intst","price","rooms","area","land","baths","
                         
for a in selected_columns:
  plt.boxplot(selected_columns[a])
  plt.xlabel(a)
  plt.show()
```

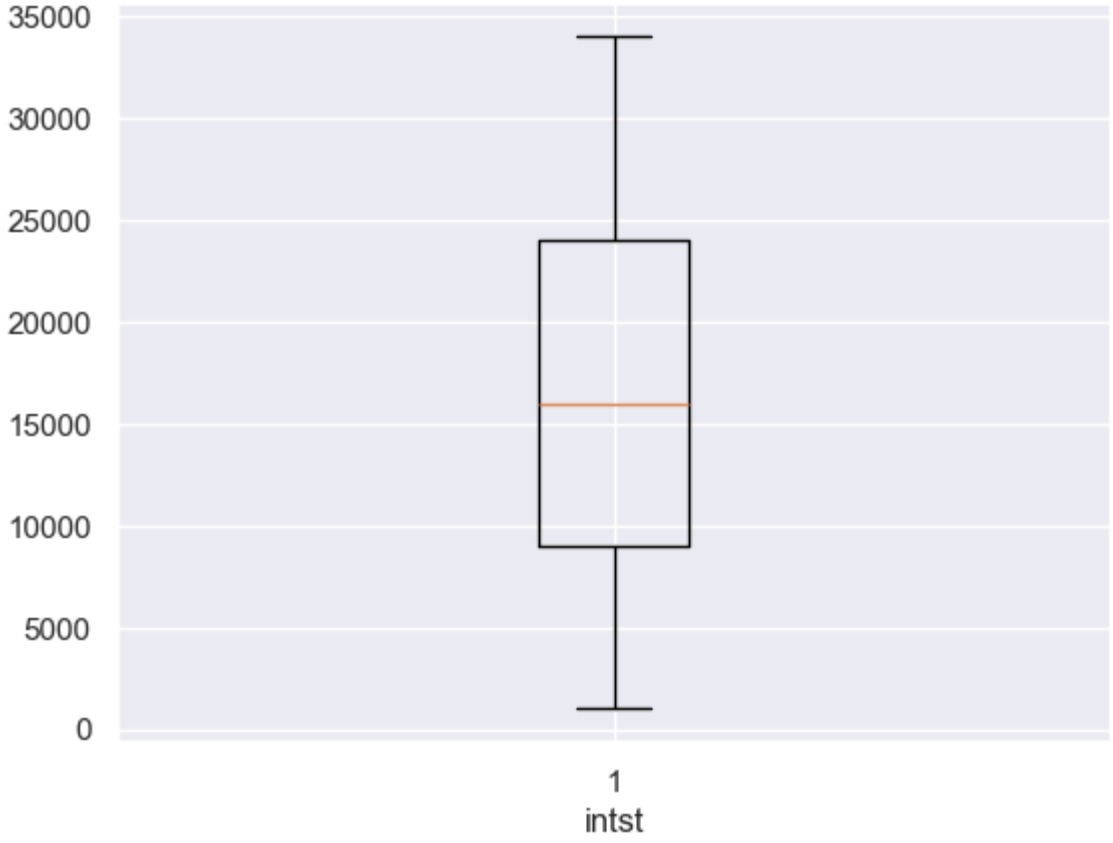

1981.0
1980.5
1980.0
1979.5
1979.0
1978.5
1978.0
1
year

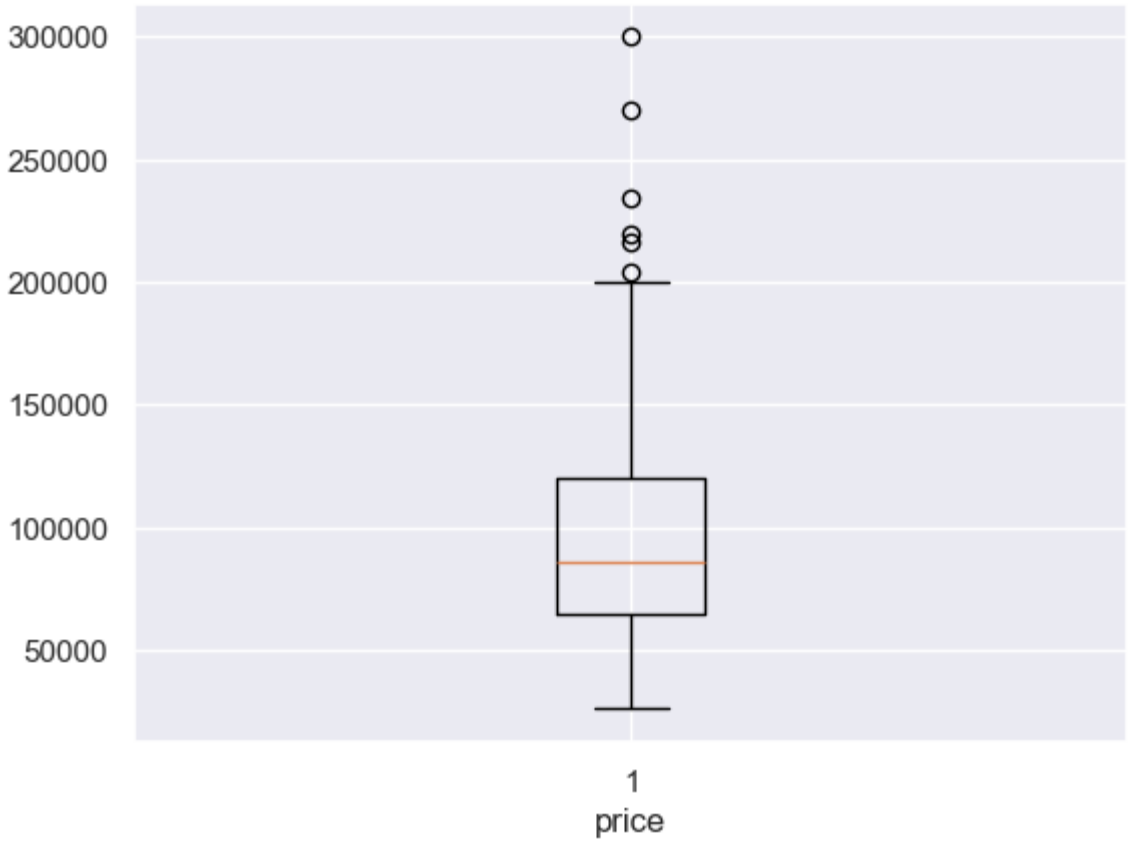

175
150
125
100
75
50
25
0
1
age

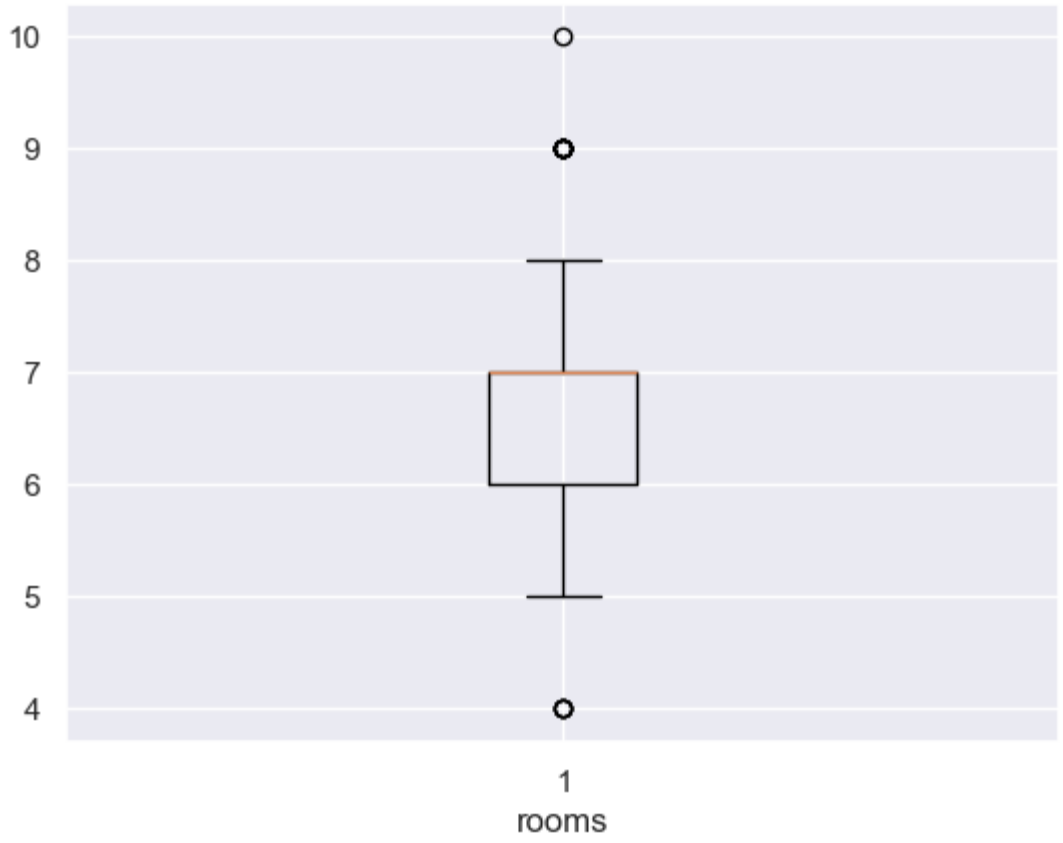

35000
30000
25000
20000
15000
10000
5000
0
1
agesq

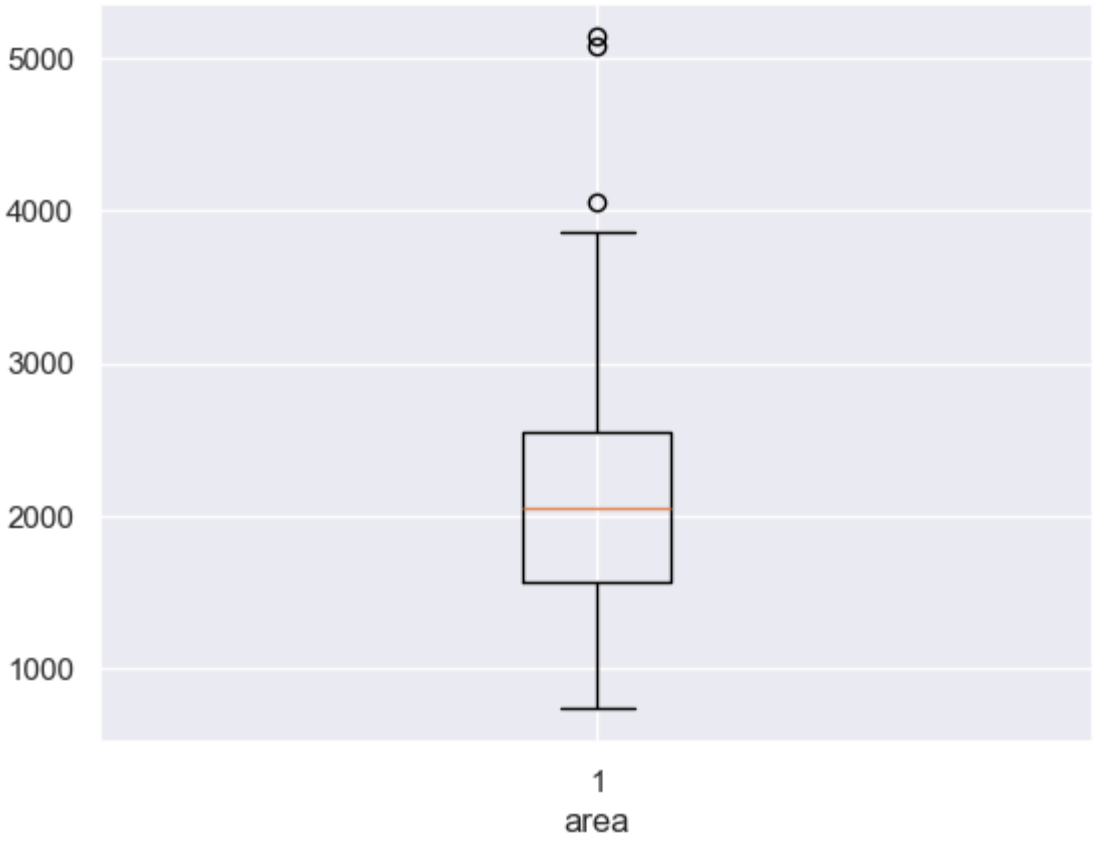

6
5
4
3
2
1
0
1
nbh

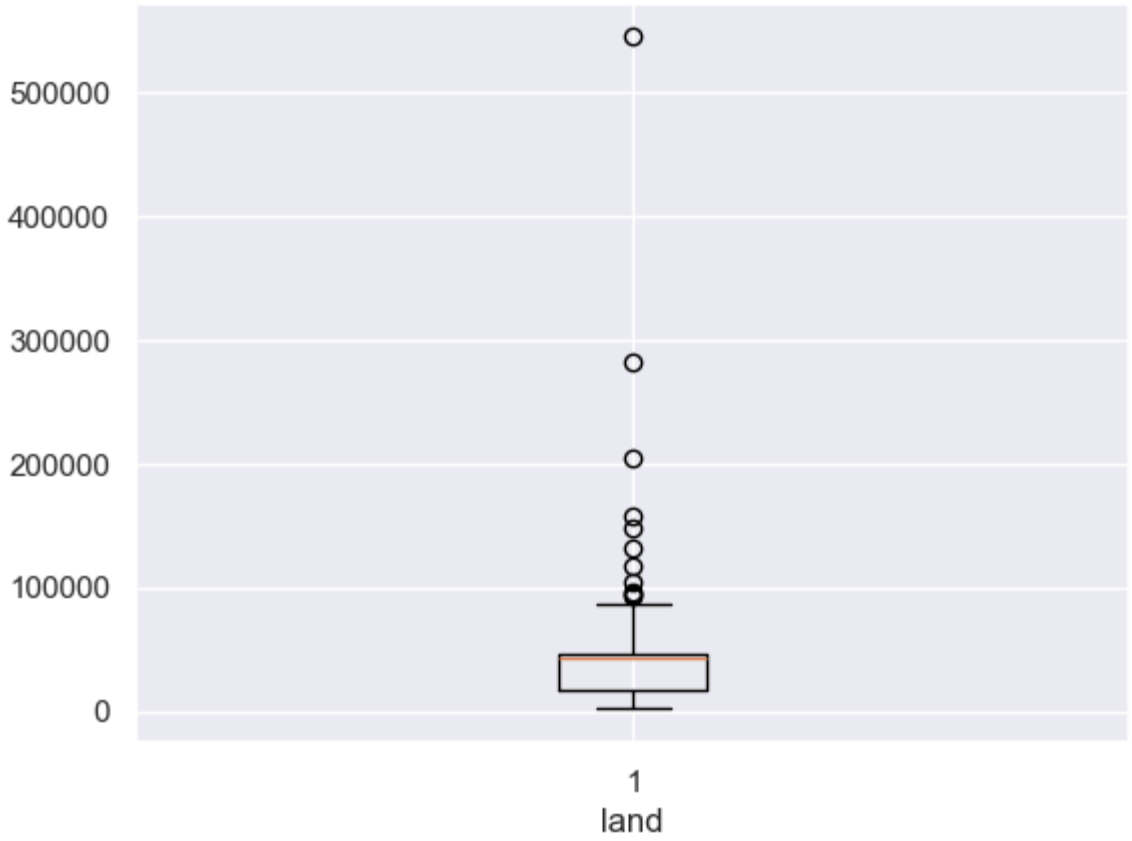

35000
30000
25000
20000
15000
10000
5000
0
1
cbd

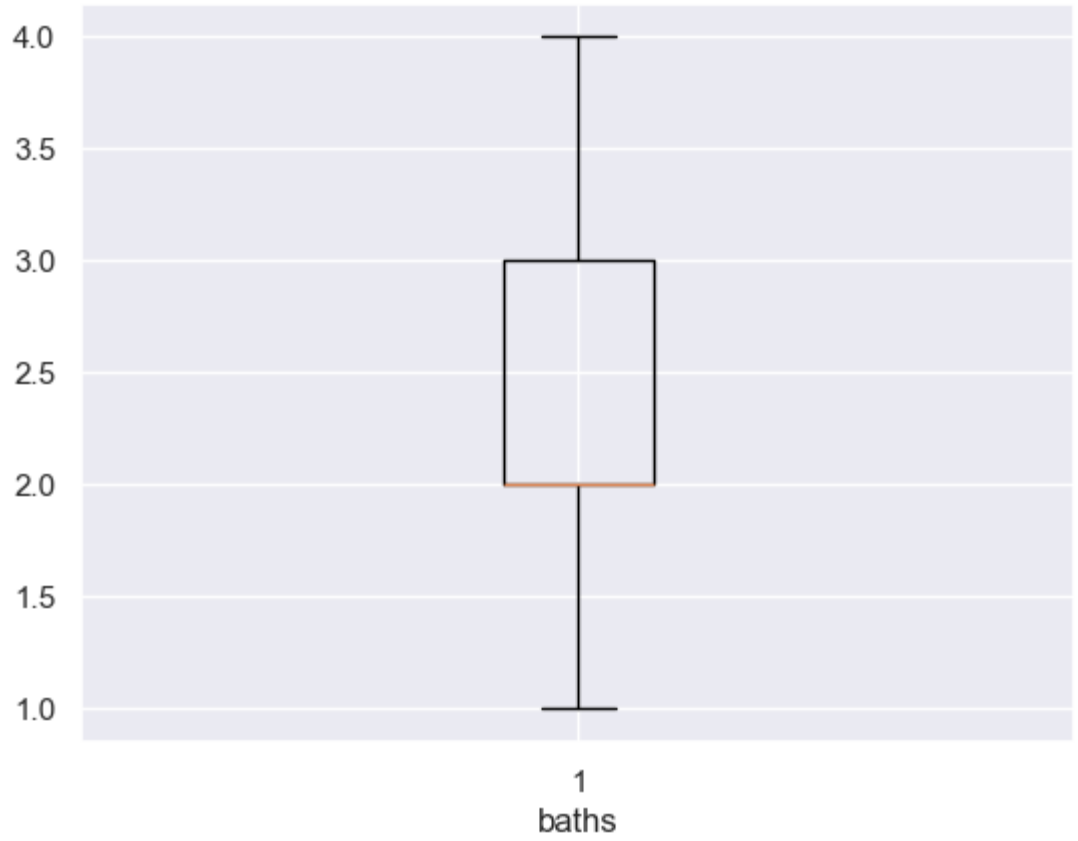

35000
30000
25000
20000
15000
10000
5000
0
1
intst

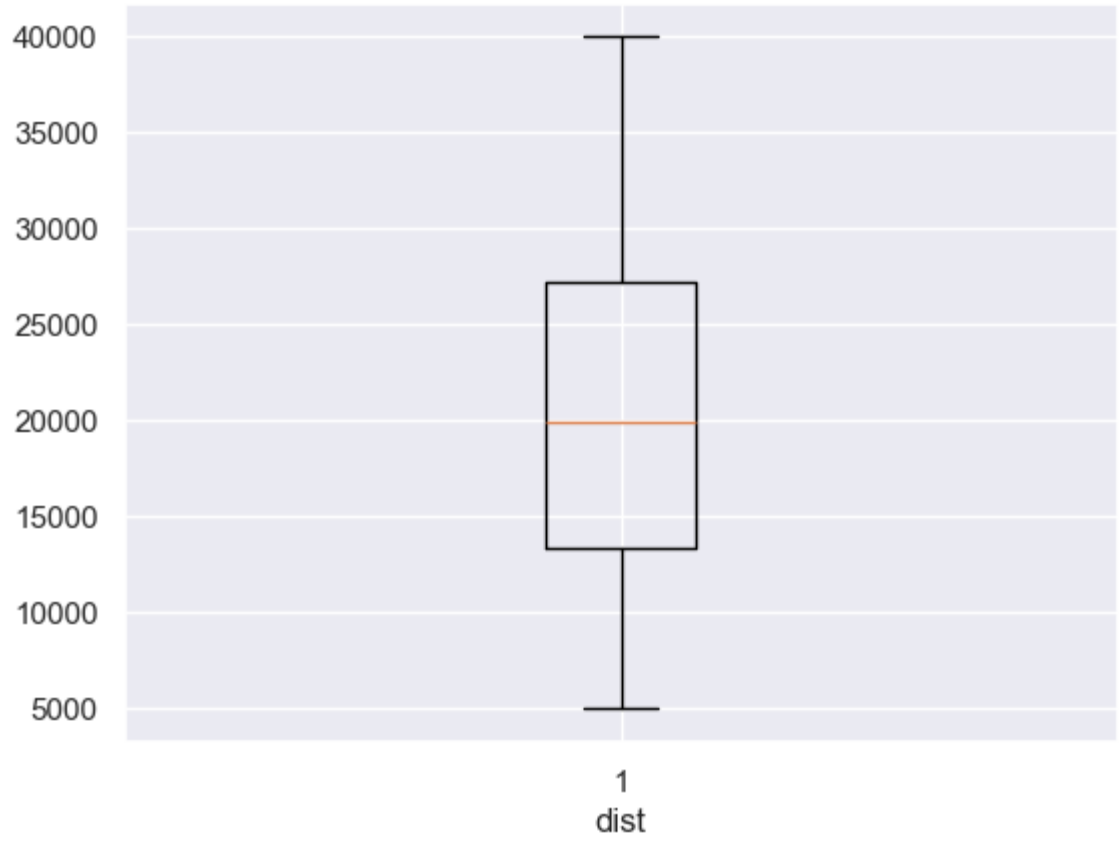

300000
250000
200000
150000
100000
50000
1
price

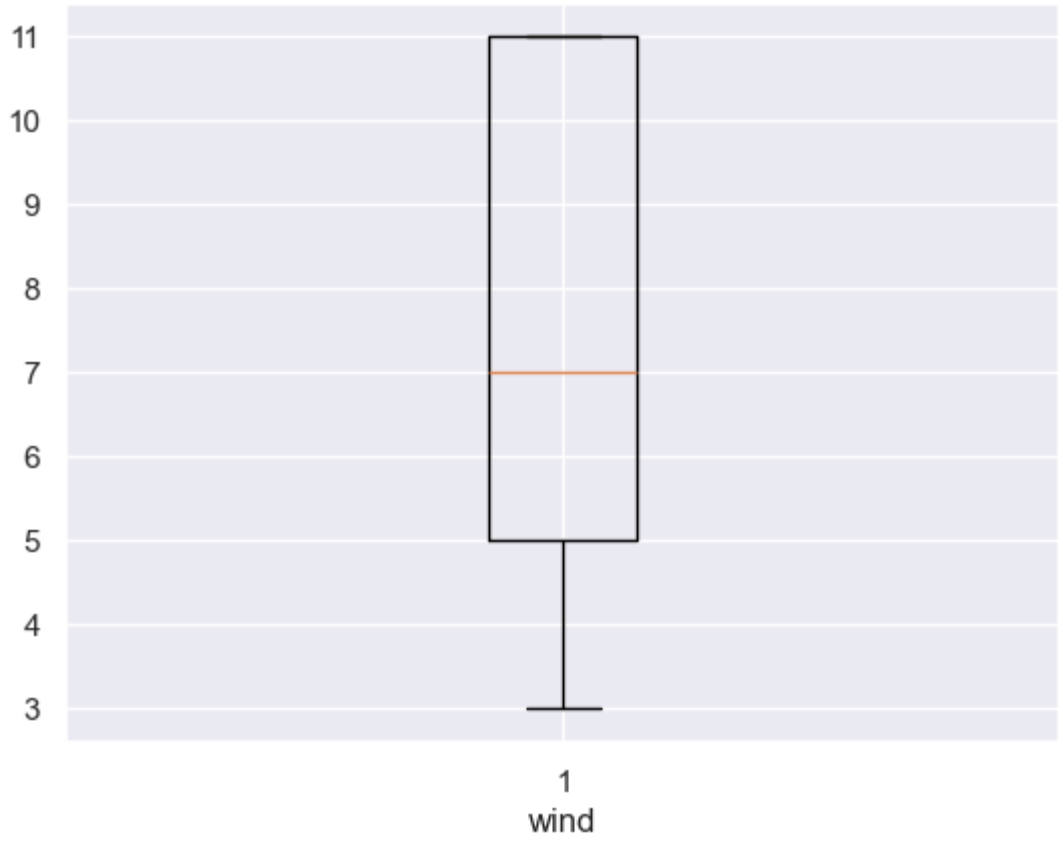

10
9
8
7
6
5
4
1
rooms

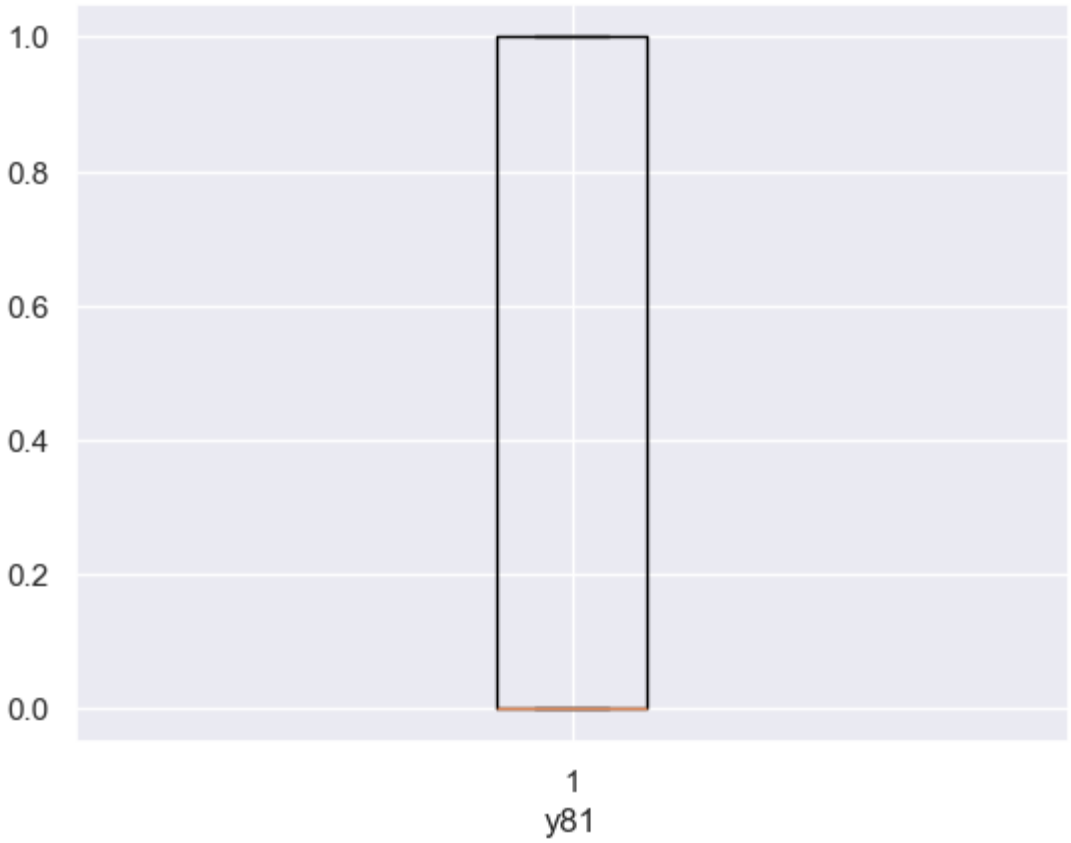

5000
4000
3000
2000
1000
1
area


500000
400000
300000
200000
100000
0
1
land

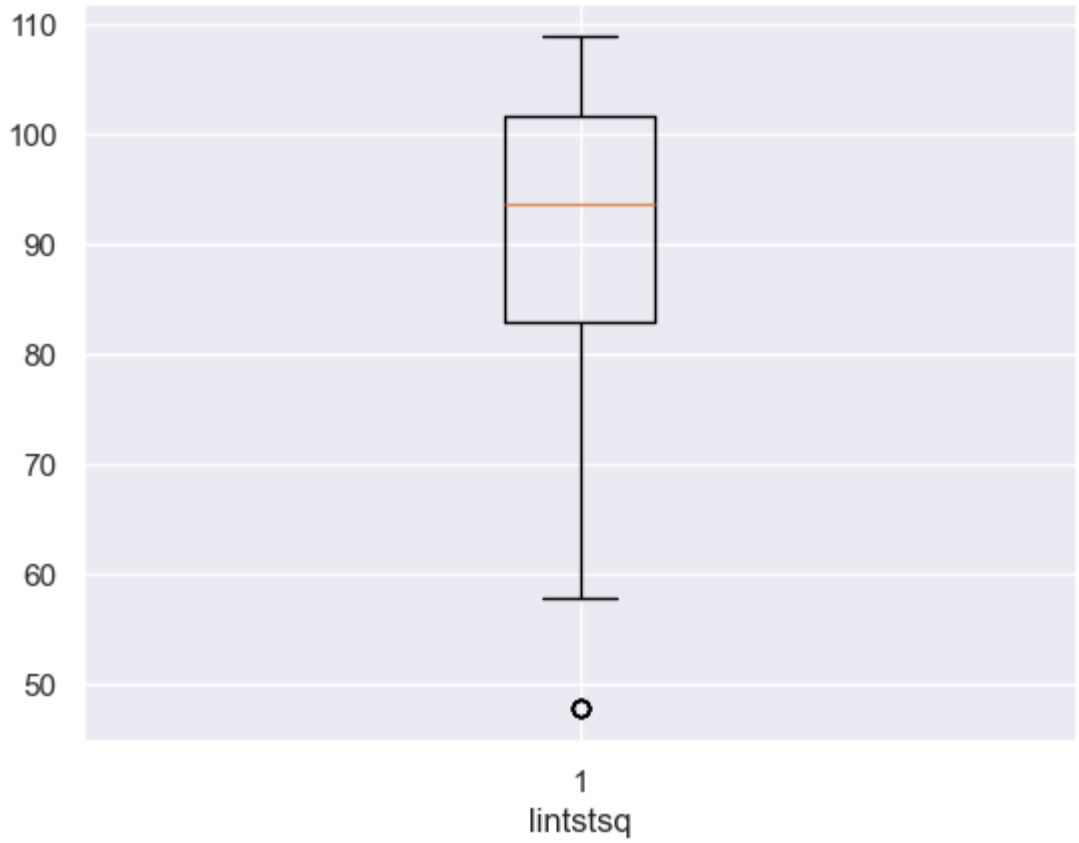

4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
baths

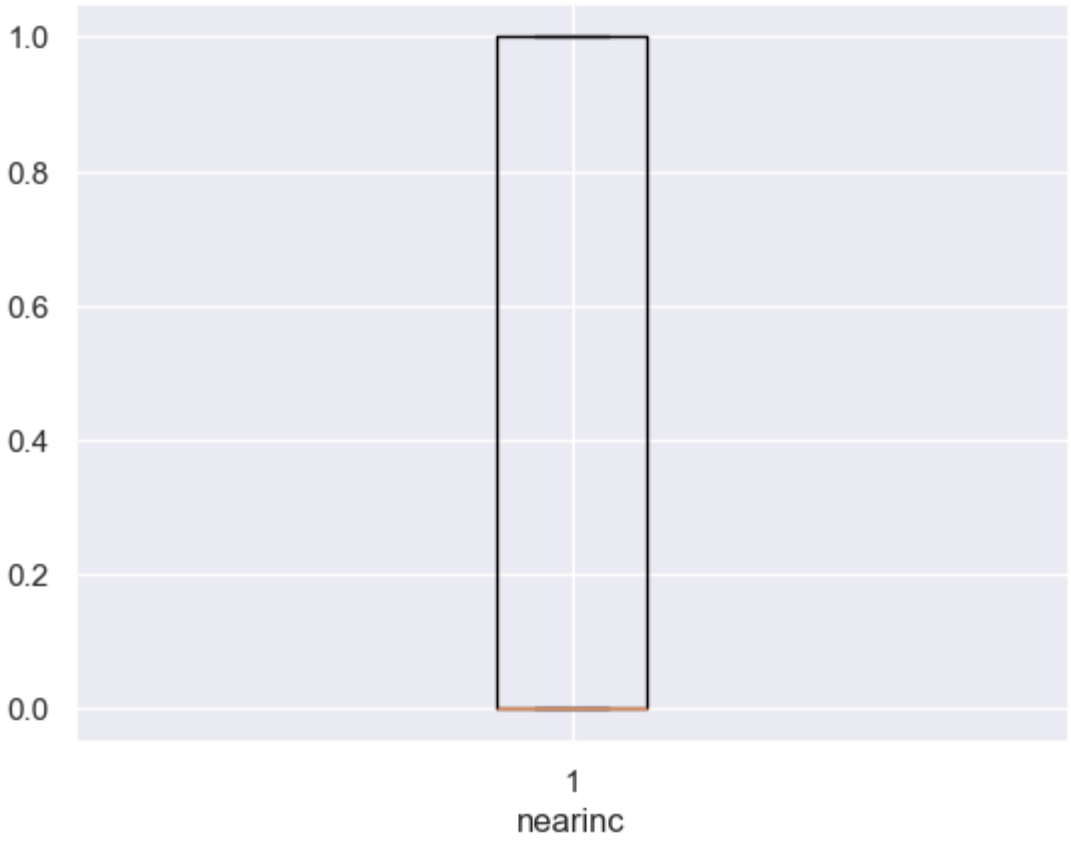

40000
35000
30000
25000
20000
15000
10000
5000
1
dist

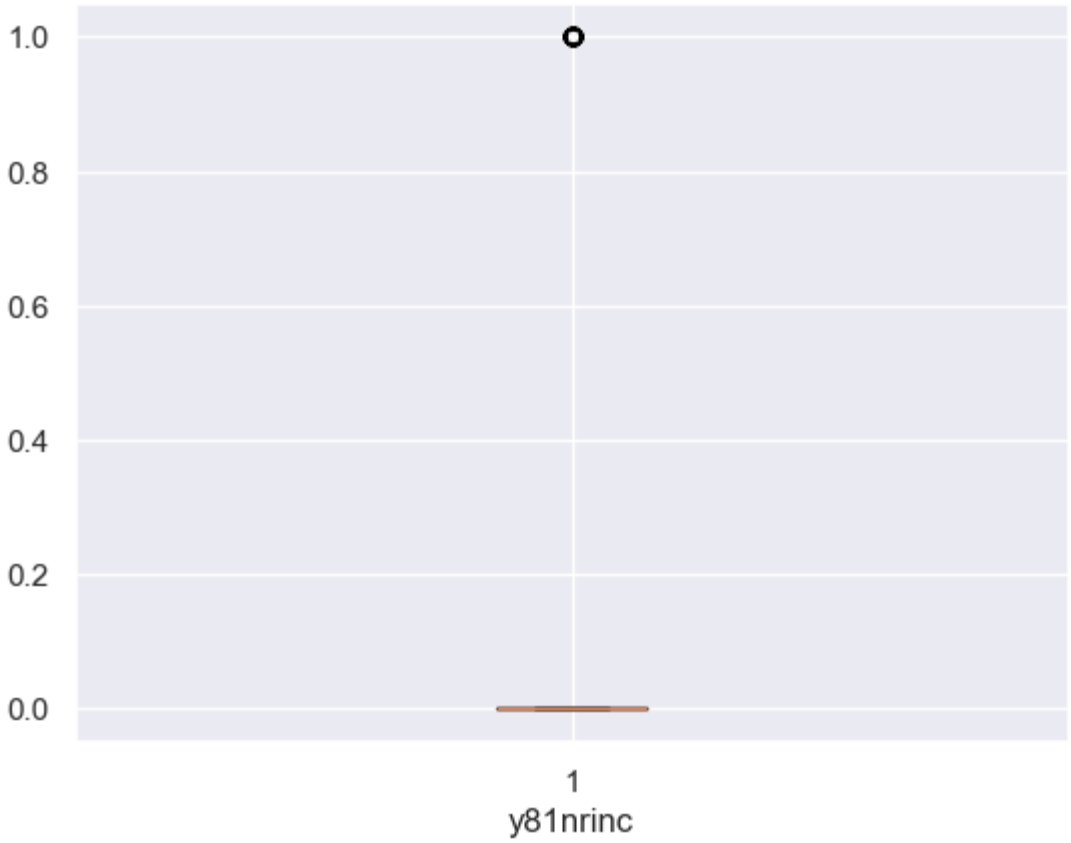

11
10
9
8
7
6
5
4
3
1
wind

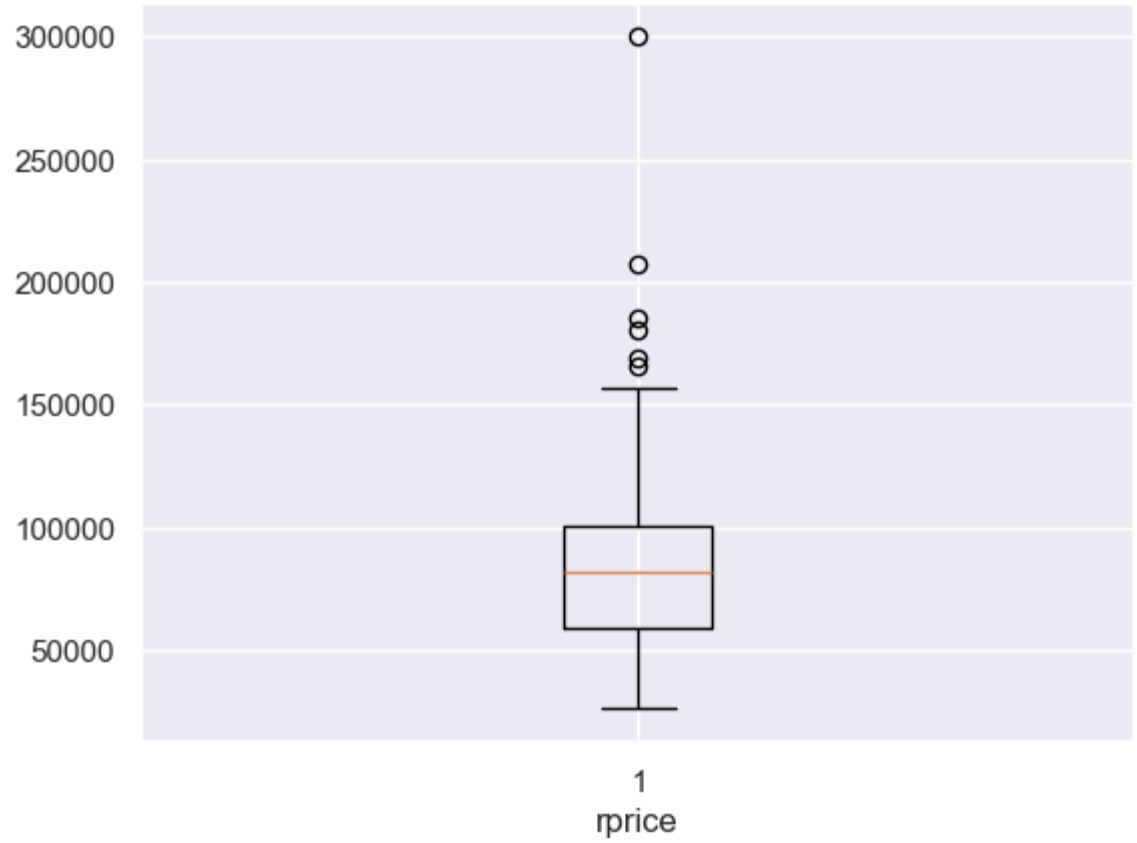

1.0
0.8
0.6
0.4
0.2
0.0
1
y81


10
8
6
4
2
0
1
y81ldist


110
100
90
80
70
60
50
1
lintstsq


1.0
0.8
0.6
0.4
0.2
0.0
1
nearinc


1.0
0.8
0.6
0.4
0.2
0.0
1
y81nrinc



Analysis of box plots:

As we can see there seems to be huge outliers which are affecting the boxplots. Hence we will have to map the variables with large numbers separately (can identify using mean value) Eg. area, land, rprice

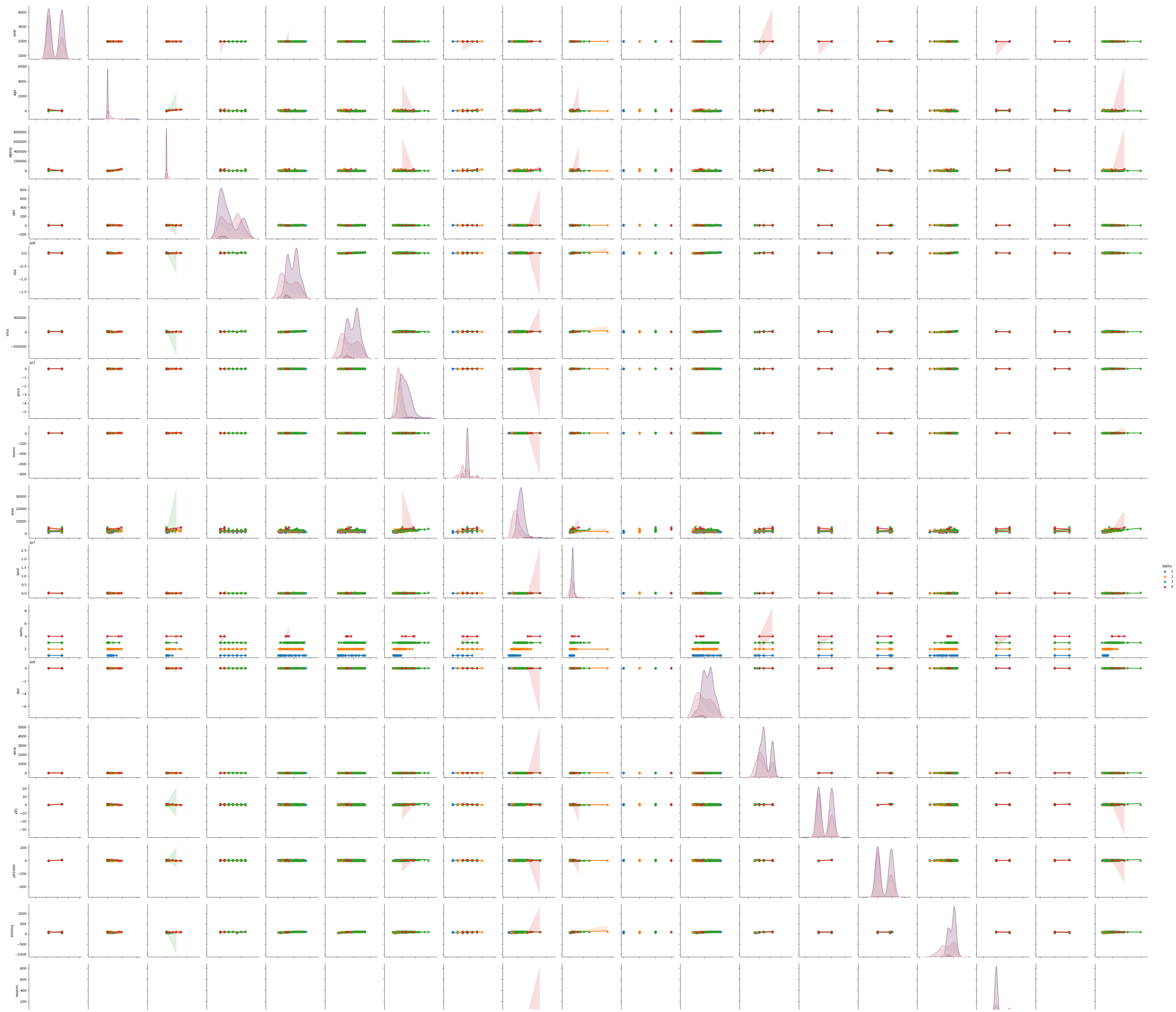
In [24]:
```
#Part 5: Scatterplot Matrix

sns.pairplot(df_0 ,vars=["year","age","agesq","nbh","cbd","intst","price","rooms","area","land","baths","d
```

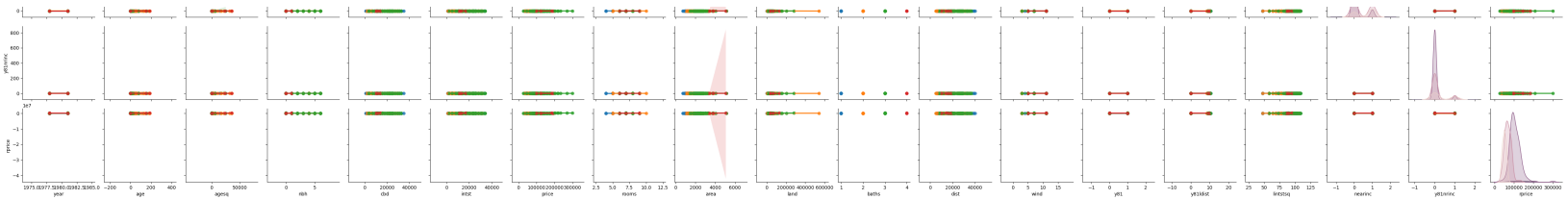
```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[24]: <seaborn.axisgrid.PairGrid at 0x177eece50>

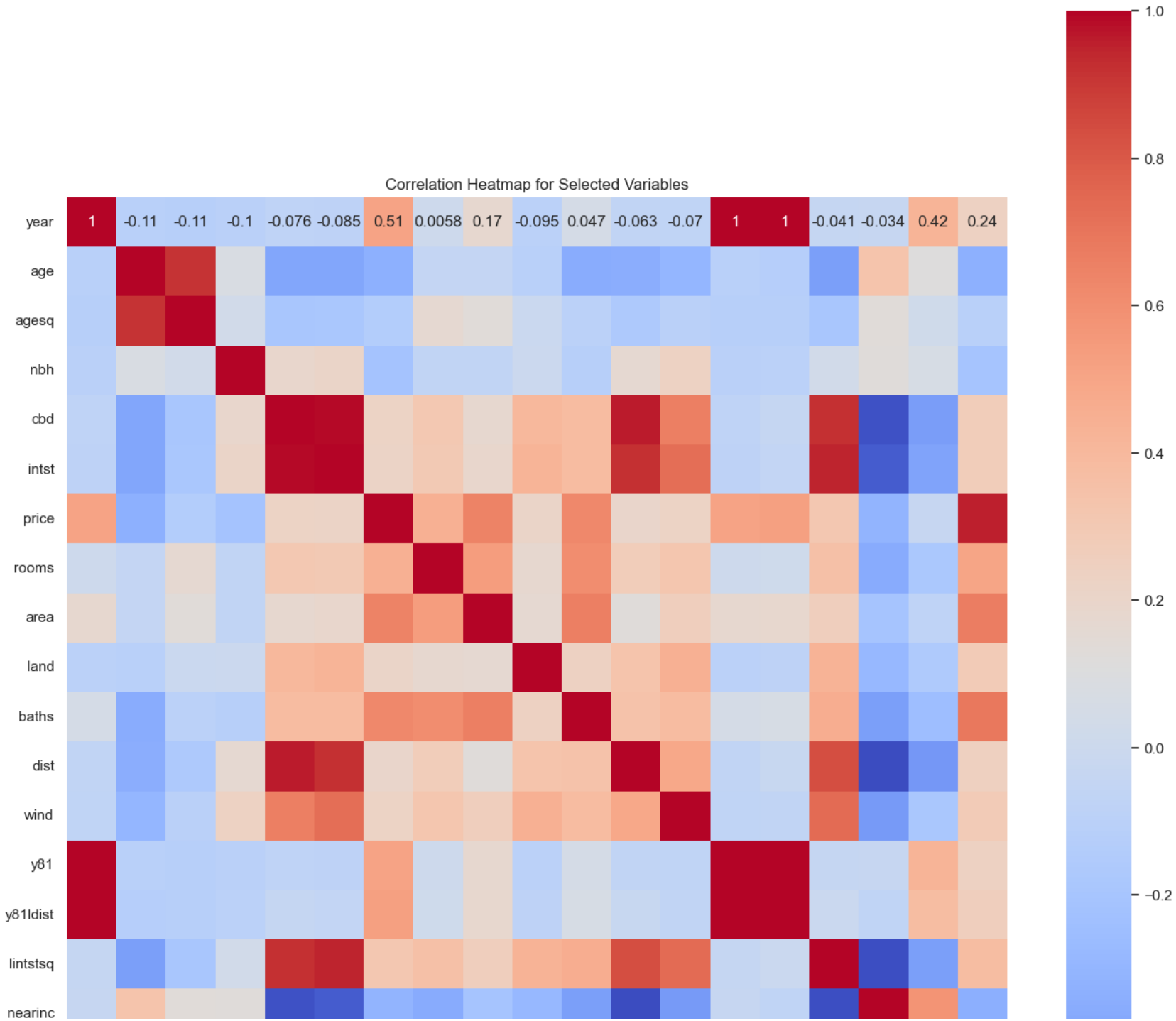


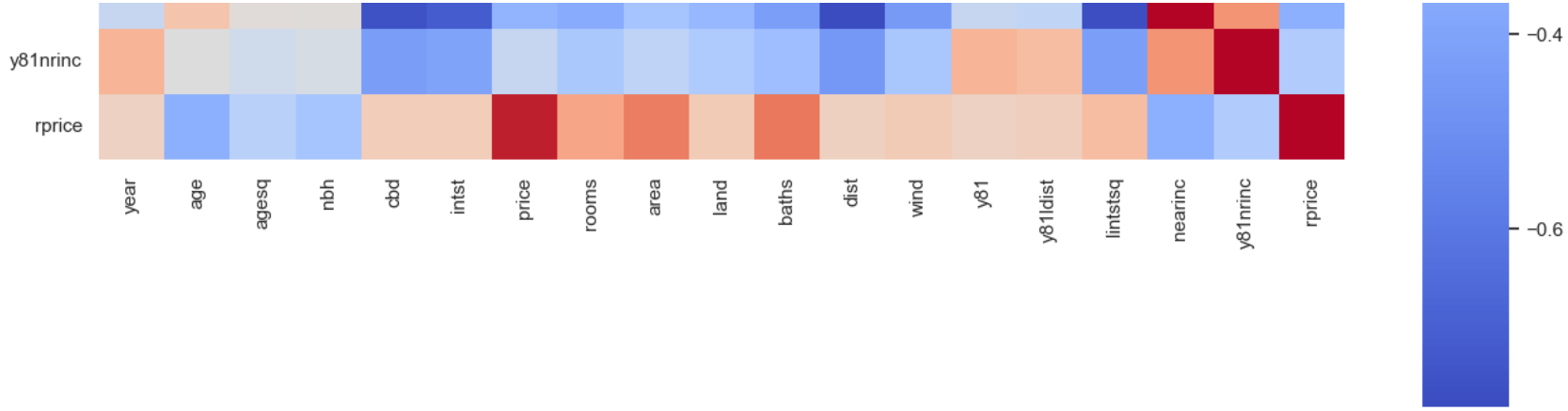


A Scatterplot Matrix is a grid of scatterplots where each variable is plotted against every other variable in the dataset. Each cell in the matrix represents the scatterplot between two variables, and the diagonal cells usually display histograms or kernel density plots for individual variables.

```
Diagonal Plots:
    The diagonal cells display univariate distributions for each variable. They provide
insights into the distribution of individual variables.

Scatterplots:
    Off-diagonal cells show scatterplots between pairs of variables. Each point on a
scatterplot represents a data point in the dataset, with one variable on the x-axis and
another on the y-axis.
    The pattern of points in a scatterplot reveals the relationship between the two
variables. Common patterns include linear relationships, non-linear relationships,
clusters, or no apparent relationship.

Correlation:
    The scatterplot matrix helps visualize the pairwise relationships between variables
and can provide an initial indication of correlation

Outliers:
    Outliers or unusual data points can be identified in scatterplots as points that
deviate significantly from the general pattern.
```

In [30]:
```
# Part 6: Using the heatmap function to create a correlation heatmap

correlation_matrix = selected_columns1.corr()

# Create a heatmap
sns.set(font_scale=1)
plt.figure(figsize=(16, 18))

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", cbar=True, square=True)

plt.title("Correlation Heatmap for Selected Variables")
plt.show()
```


Correlation Heatmap for Selected Variables
year 1 -0.11 -0.11 -0.1 -0.076 -0.085 0.51 0.0058 0.17 -0.095 0.047 -0.063 -0.07 1 1 -0.041 -0.034 0.42 0.24
age
agesq
nbh
cbd
intst
price
rooms
area
land
baths
dist
wind
y81
y81ldist
lintstsq
nearinc
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

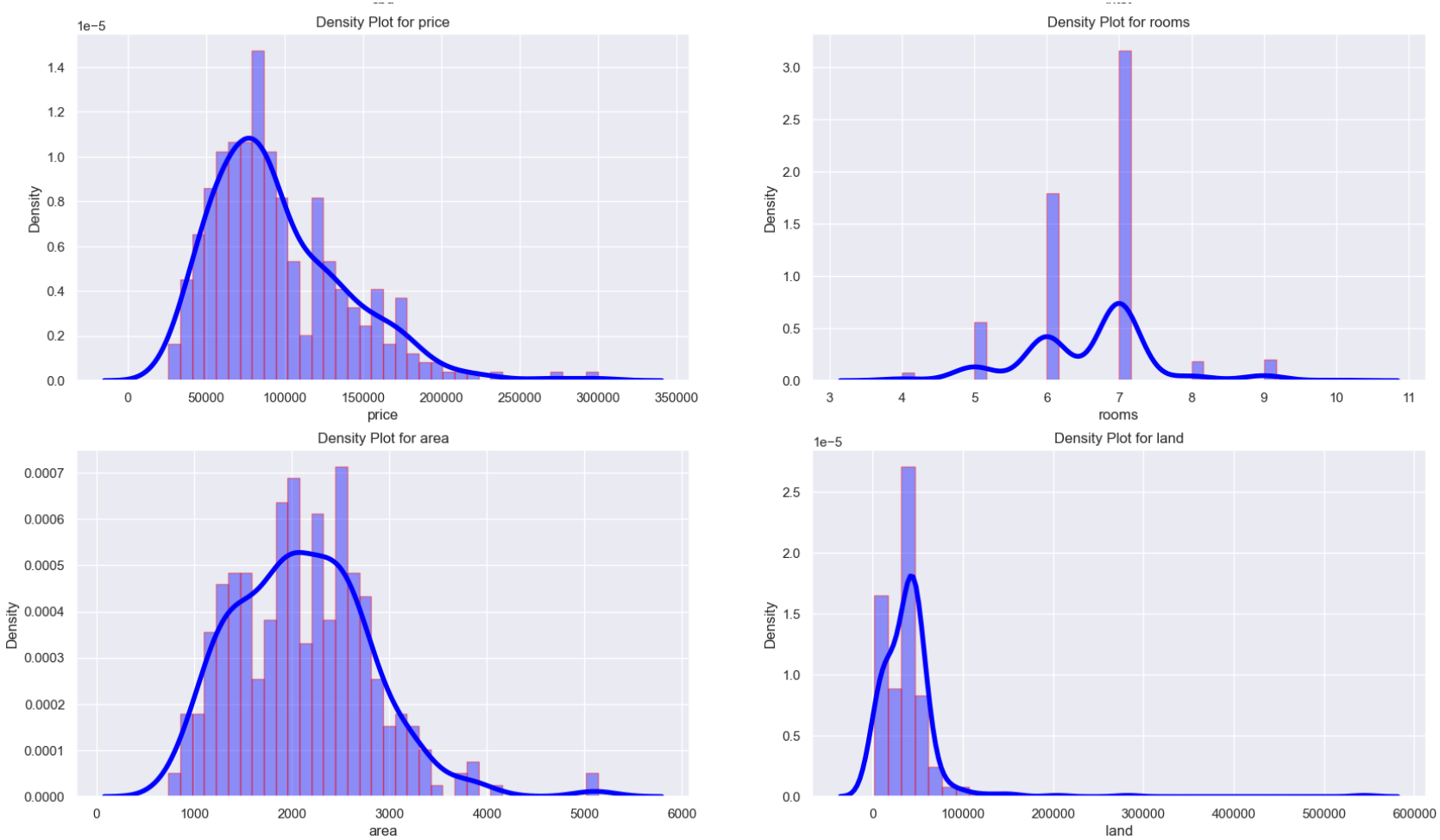


Analysis of Correlation heat map:

A correlation heatmap is a visual representation of the pairwise correlations between variables in a dataset. Analyzing a correlation heatmap can provide valuable insights into the relationships between variables.

1.Positive correlation: year tends to have a higher correlation with variable price and y81nrinc. The highest correlation is observed between varaibles 'year' and 'y81'& 'y81ldist', thus, these two variables should not be taken together in one model.

2.Negative correlation: There is a negative correlation between a few variables. However these negative correlations is not that high and thus, might not impact the model.



Analysis of Density plot:

Density plots are data visualization tools used to understand the distribution of a dataset. The following observations are made after analysing density estimation plots:

1.Data Distribution Shape: We can observe skewness in variables.

2.Spread of the density plots: There are some variables that do not seem to be a strong variable for the analysis as it is highly skewed with a very narrow distribution indicating lower variability.

3.Normality Assessment: A proper and symmetric density plot can be an indication that the data follows a normal distribution

3.Normality Assessment: A proper and symmetric density plot can be an indication that the data follows a normal distribution.

In [32]:
```
import math
```

In [34]:
```
selected_variables1= ["year","age","agesq","nbh","cbd","intst","price","rooms","area","land","baths","dist
```

In [35]:
```
from fitter import Fitter
f = Fitter(df_0[selected_variables1])
f.fit()
```

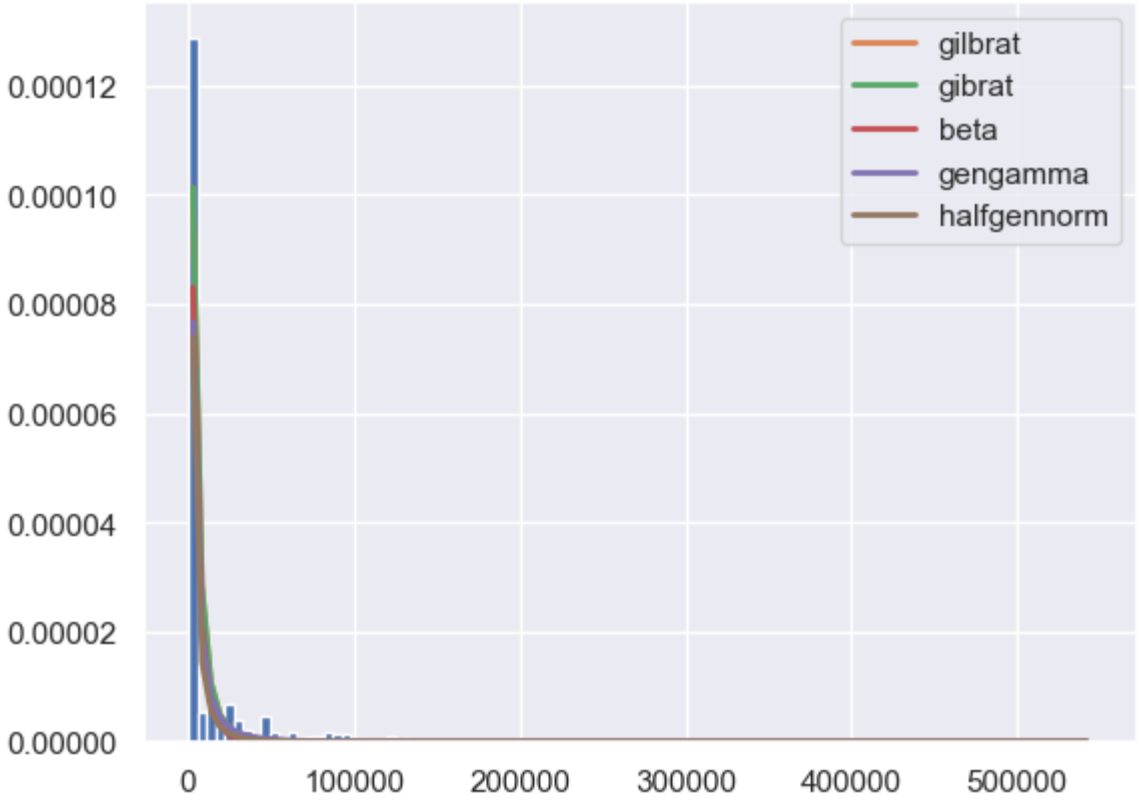
```
SKIPPED _fit distribution (taking more than 30 seconds)
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/stats/_continuous_distns.py:3485: Integrat
ionWarning: The algorithm does not converge.  Roundoff error is detected
  in the extrapolation table.  It is assumed that the requested tolerance
  cannot be achieved, and that the returned result (if full_output = 1) is
  the best which can be obtained.
  t1 = integrate.quad(llc, -np.inf, x)[0]
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/stats/_continuous_distns.py:3485: Integrat
ionWarning: The integral is probably divergent, or slowly convergent.
  t1 = integrate.quad(llc, -np.inf, x)[0]
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/stats/_continuous_distns.py:4355: Integrat
ionWarning: The integral is probably divergent, or slowly convergent.
  return integrate.quad(llc, _a, x)[0]
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED loguniform distribution (taking more than 30 seconds)
SKIPPED lomax distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED powerlaw distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED rdist distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225: Integratio
nWarning: The maximum number of subdivisions (50) has been achieved.
  If increasing the limit yields no improvement it is advised to analyze
  the integrand in order to determine the difficulties.  If the position of a
  local difficulty can be determined (singularity, discontinuity) one will
  probably gain from splitting up the interval and calling the integrator
```

```
  on the subranges.  Perhaps a special-purpose integrator should be used.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED reciprocal distribution (taking more than 30 seconds)
SKIPPED rice distribution (taking more than 30 seconds)
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225: Integratio
nWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
SKIPPED skewcauchy distribution (taking more than 30 seconds)
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED trapezoid distribution (taking more than 30 seconds)
SKIPPED trapz distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncexpon distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

In [36]:
```
f.summary()
```

Out[36]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
gilbrat	1.365306e-09	4299.963629	4313.395390	inf	0.413548	0.0
gibrat	1.365306e-09	4299.963629	4313.395390	inf	0.413548	0.0
beta	2.428441e-09	10231.519699	10258.383220	inf	0.420613	0.0
gengamma	2.951666e-09	4042.803206	4069.666727	inf	0.454645	0.0
halfgennorm	3.170352e-09	4300.841477	4320.989118	inf	0.455629	0.0


gilbrat
gibrat
beta
gengamma
halfgennorm
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
100000
200000
300000
400000
500000

Let's interpret each of these metrics:

```
Sum of Squared Error:
    This measures the discrepancy between observed and predicted values in your model.
Smaller values indicate a better fit.

AIC (Akaike Information Criterion):
    AIC is a measure of the goodness of fit of a statistical model. It penalizes models
with more parameters, helping to avoid overfitting. Lower AIC values suggest a better-
fitting model.

BIC (Bayesian Information Criterion):
    Similar to AIC, BIC is another criterion for model selection. It penalizes models
with more parameters but generally more severely than AIC. Like AIC, lower BIC values
indicate a better fit.

KL Divergence (Kullback-Leibler Divergence):
    KL divergence measures the difference between two probability distributions. An
infinite value (inf) may suggest that the true and predicted distributions are very
different.

KS Statistic (Kolmogorov-Smirnov Statistic):
    KS statistic measures the maximum vertical distance between the cumulative
distribution functions of the observed and predicted distributions. A value close to 0
suggests a good fit.

KS P-value (Kolmogorov-Smirnov P-value):
    The p-value associated with the KS statistic. A low p-value (typically below a
significance level like 0.05) suggests that the observed and predicted distributions are
significantly different.
```

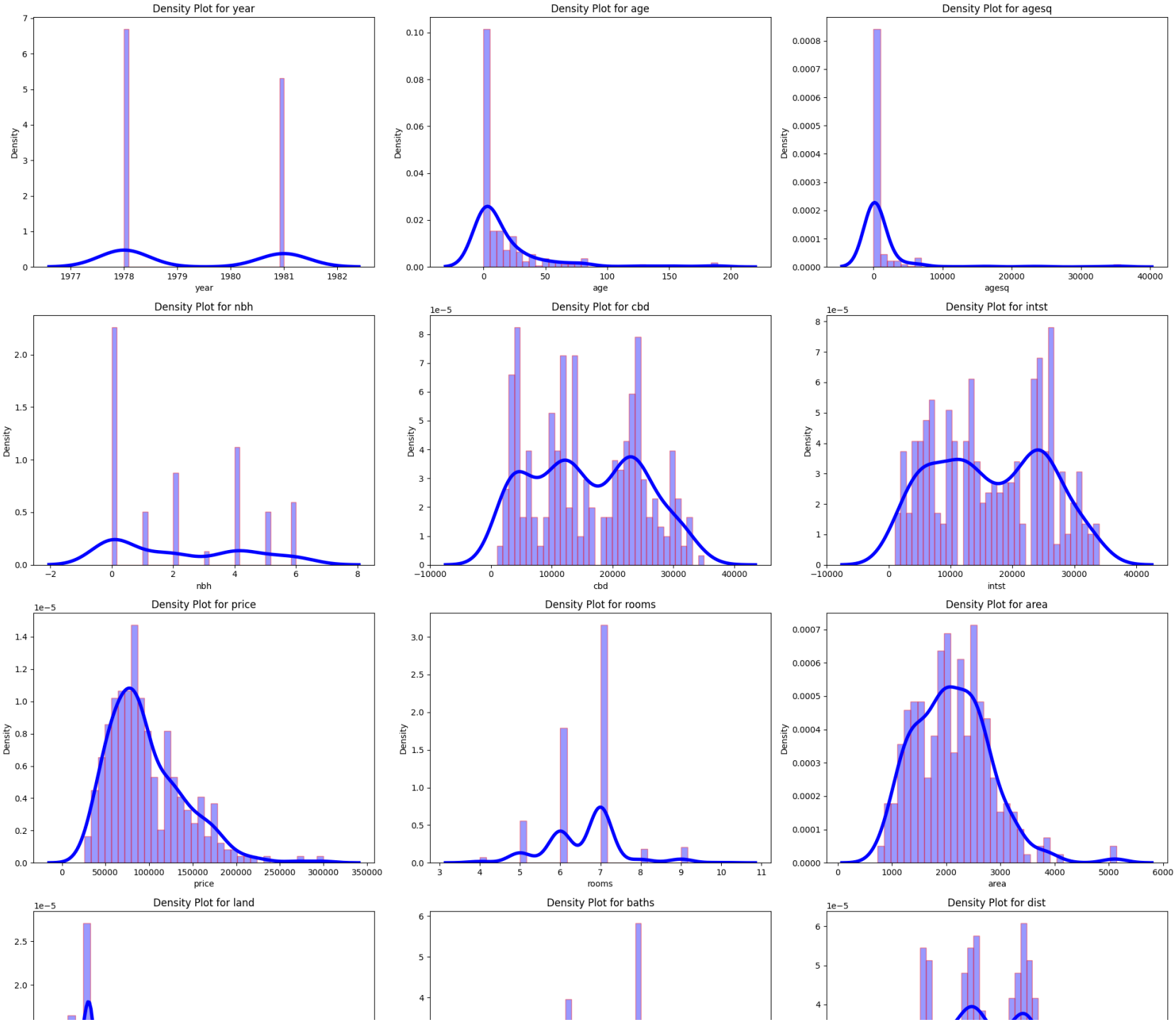
Interpretation:

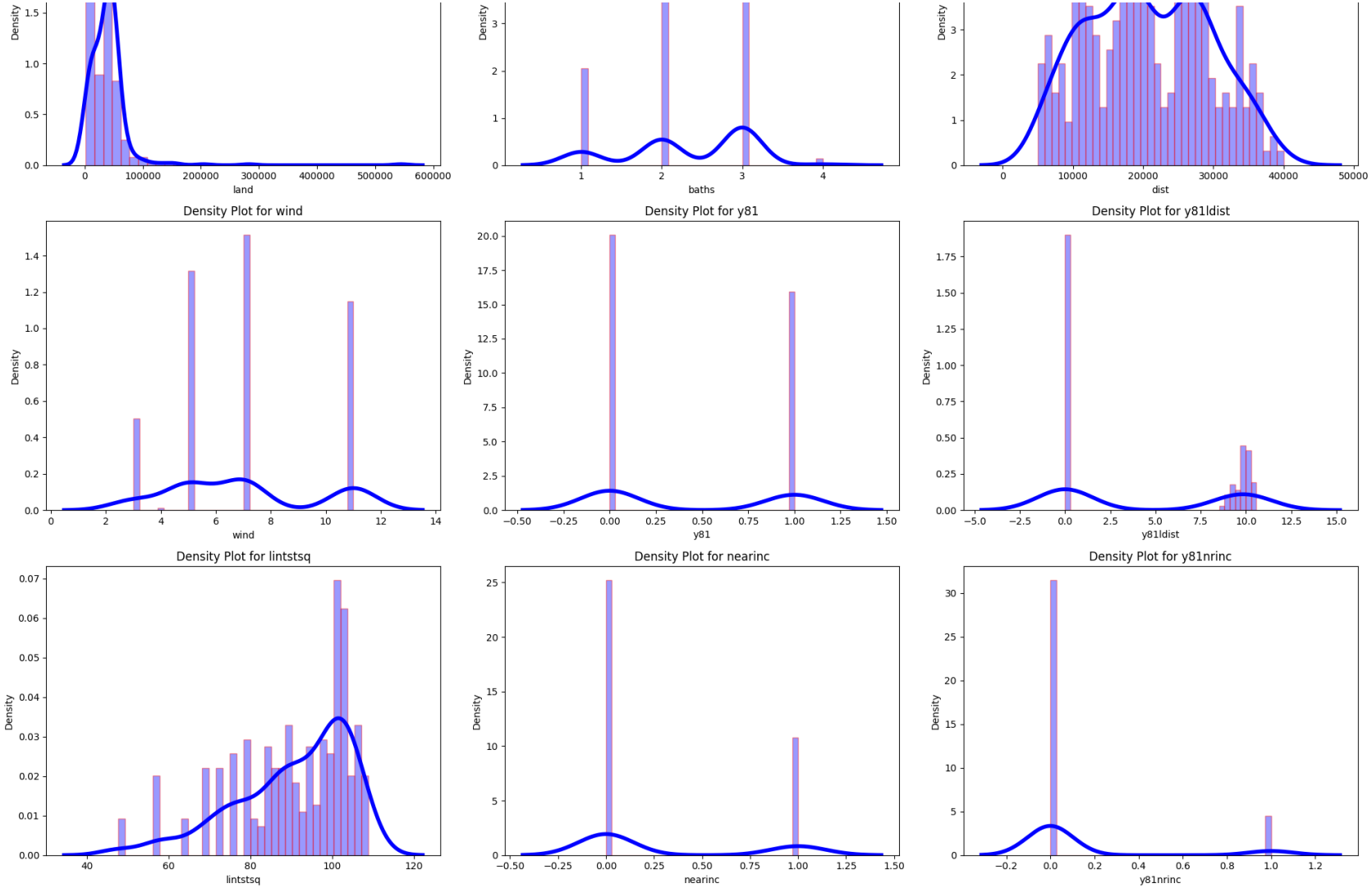
```
For all models (gilbrat, gibrat, beta, gengamma, halfgennorm), the sum of squared errors
is very close to zero, indicating a good fit to the data.
Lower AIC and BIC values suggest better-fitting models. In this case, gilbrat and gibrat
have the lowest AIC and BIC, making them potentially preferable based on these criteria
```

have the lowest AIC and BIC, making them potentially preferable based on these criteria.
KL Divergence being infinite might indicate a substantial difference between the true and predicted distributions.
High KS statistic values close to 0.45 for all models suggest good agreement between observed and predicted distributions.
Very low KS p-values (0.0) indicate that the observed and predicted distributions are significantly different.

In [ ]:

In [7]:
```
# Density Plots
import matplotlib.pyplot as plt
import seaborn as sns
# Create a figure with subplots for Density estimates
fig, axes = plt.subplots(6, 3, figsize=(20, 30))
axes = axes.ravel()
# Creating density plots for each variable in the selected_columns list using a for loop:
for a, b in enumerate(selected_columns1):
    ax = axes[a]
    data = selected_columns1[b]

# Create a density plot for the variable
    sns.distplot(data, ax=ax, hist=True, kde=True,
                 bins=int(180/5), color='blue',
                 hist_kws={'edgecolor': 'red'},
                 kde_kws={'linewidth': 4})
    ax.set_title(f'Density Plot for {b}')
    ax.set_xlabel(b)
    ax.set_ylabel('Density')

plt.tight_layout()
plt.show()
```


Density Plot for year
Density
year
Density Plot for age
Density
age
Density Plot for agesq
Density
agesq
Density Plot for nbh
Density
nbh
Density Plot for cbd
1e−5
Density
cbd
Density Plot for intst
1e−5
Density
intst
Density Plot for price
1e−5
Density
price
Density Plot for rooms
Density
rooms
Density Plot for area
Density
area
Density Plot for land
1e−5
Density Plot for baths
Density Plot for dist
1e−5



In [8]:
```
df_0['price'] = df_0['price'].astype(int)
```

In [9]:
```
df_0 = df_0.set_index(['nbh','year'],drop=False)
```

Based on the dataset and variables, a possible economic model could be a price model. The model is commonly used in real estate economics to estimate the relationship between the characteristics of a house and its market price.

In [2]:
```
#Importing relevant functions
import statsmodels.api as sm
import statsmodels as sms
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import os
%matplotlib inline
```

II. Qualitative Dependent Variable Models

In [3]:
```
data1 = pd.read_csv(r"/Users/bharatsingh/Downloads/diabetes_prediction_dataset.csv")
```

In [4]:
```
data1
```

Out[4]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

In [5]:
```
data1['gender'] = data1['gender'].map({'Male': 0, 'Female': 1})
```

In [6]:
```
data1
```

Out[6]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1.0	80.0	0	1	never	25.19	6.6	140	0
1	1.0	54.0	0	0	No Info	27.32	6.6	80	0
2	0.0	28.0	0	0	never	27.32	5.7	158	0
3	1.0	36.0	0	0	current	23.45	5.0	155	0
4	0.0	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	1.0	80.0	0	0	No Info	27.32	6.2	90	0
99996	1.0	2.0	0	0	No Info	17.37	6.5	100	0
99997	0.0	66.0	0	0	former	27.83	5.7	155	0
99998	1.0	24.0	0	0	never	35.42	4.0	100	0
99999	1.0	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

In [7]:
```
data= data1.drop(['smoking_history'], axis=1)
data.head()
```

Out[7]:

	gender	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1.0	80.0	0	1	25.19	6.6	140	0
1	1.0	54.0	0	0	27.32	6.6	80	0
2	0.0	28.0	0	0	27.32	5.7	158	0
3	1.0	36.0	0	0	23.45	5.0	155	0
4	0.0	76.0	1	1	20.14	4.8	155	0

In [8]:
```
columns = data.columns

binary_variables = []

# Iterate through each column and check if it has only two unique values
for column in columns:
    unique_values = data[column].unique()
    if len(unique_values) == 2 and set(unique_values) == {0, 1}:
        binary_variables.append(column)

# Print the binary variables and their count
print("Binary Variables:", binary_variables)
print("Number of Binary Variables:", len(binary_variables))
```

```
Binary Variables: ['hypertension', 'heart_disease', 'diabetes']
Number of Binary Variables: 3
```

Question 1

Let's first take a brief overview of the provided data:

Gender: Categorical variable (0.0 for female, 1.0 for male)

Age: Continuous variable representing the age of individuals

Hypertension: Binary variable (0 for no hypertension, 1 for hypertension)

Heart Disease: Binary variable (0 for no heart disease, 1 for heart disease)

BMI (Body Mass Index): Continuous variable representing the BMI of individuals

HbA1c Level: Continuous variable representing the HbA1c level of individuals

Blood Glucose Level: Continuous variable representing the blood glucose level of individuals

Diabetes: Binary variable (0 for no diabetes, 1 for diabetes)

Objective:

Based on our data, we are interested in predicting the likelihood of diabetes in individuals based on various health-related features. The target variable for our predictive model is "Diabetes," which is a binary variable indicating whether an individual has diabetes or not.

Potential Business Questions:

```
Risk Assessment: Can we predict the risk of diabetes in individuals based on their health
attributes? This could be valuable for preventive healthcare programs or insurance
companies looking to assess health risks.

Healthcare Planning: Understanding the factors contributing to diabetes risk can help in
planning healthcare resources and interventions, especially for populations with higher
risk.

Personalized Health Recommendations: Could the model be used to provide personalized
health recommendations to individuals based on their health profile, such as lifestyle
changes or monitoring?
```

changes of monitoring.

Insurance Underwriting: For insurance companies, predicting the likelihood of diabetes could influence underwriting decisions and pricing of health insurance policies.

Public Health Interventions: If specific factors are identified as significant predictors of diabetes, public health interventions and campaigns could be designed to target those risk factors in the population.

Question 2

Descriptive analysis

In [9]:
```
#Checking if there is any Null Value

print(data.isnull().values.any())

#Count the number of many missing obs per variable (if any)
print(data.isnull().sum())
```

```
True
gender                18
age                    0
hypertension           0
heart_disease          0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

In [10]:
```
#Dropping columns with Null Value
columns_to_dropna = ['gender']
df = data.dropna(subset=columns_to_dropna)

print("\nDataFrame after dropping null values:")
print(df)
```

```
DataFrame after dropping null values:
       gender   age  hypertension  heart_disease    bmi  HbA1c_level  \
0         1.0  80.0             0              1  25.19          6.6
1         1.0  54.0             0              0  27.32          6.6
2         0.0  28.0             0              0  27.32          5.7
3         1.0  36.0             0              0  23.45          5.0
4         0.0  76.0             1              1  20.14          4.8
...       ...   ...           ...            ...    ...          ...
99995     1.0  80.0             0              0  27.32          6.2
99996     1.0   2.0             0              0  17.37          6.5
99997     0.0  66.0             0              0  27.83          5.7
99998     1.0  24.0             0              0  35.42          4.0
99999     1.0  57.0             0              0  22.43          6.6

       blood_glucose_level  diabetes
0                      140         0
1                       80         0
2                      158         0
3                      155         0
4                      155         0
...                    ...       ...
99995                   90         0
99996                  100         0
99997                  155         0
99998                  100         0
99999                   90         0

[99982 rows x 8 columns]
```

In [11]:
```
#Checking if there is any Null Value

print(df.isnull().values.any())

#Count the number of many missing obs per variable (if any)
print(df.isnull().sum())
```

```
False
gender                0
age                   0
hypertension          0
heart_disease         0
bmi                   0
HbA1c_level           0
blood_glucose_level   0
diabetes              0
dtype: int64
```

There are no null values in the dataset.

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 99982 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   gender               99982 non-null  float64
 1   age                  99982 non-null  float64
 2   hypertension         99982 non-null  int64  
 3   heart_disease        99982 non-null  int64  
 4   bmi                  99982 non-null  float64
 5   HbA1c_level          99982 non-null  float64
 6   blood_glucose_level  99982 non-null  int64  
 7   diabetes             99982 non-null  int64  
dtypes: float64(4), int64(4)
memory usage: 6.9 MB
```

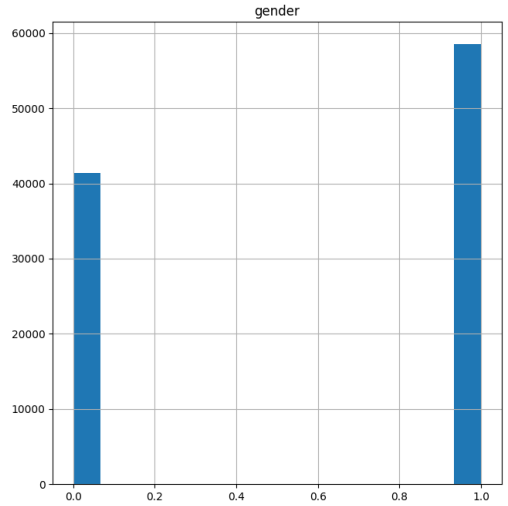
In [13]:
```
#Part 1: Summary Statistics

df.describe()
```

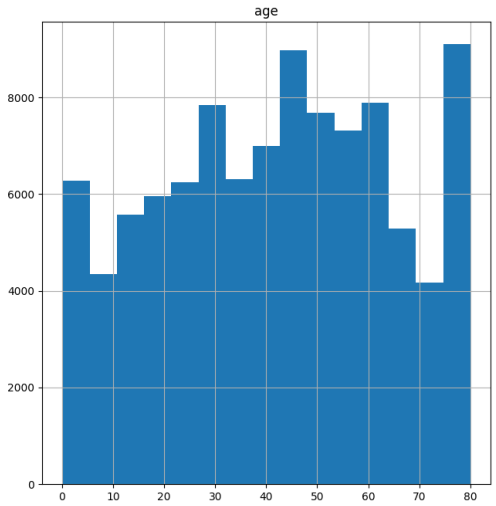
Out[13]:

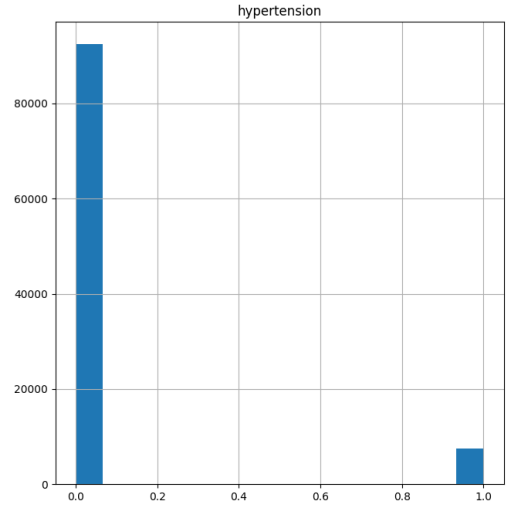
	gender	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diab
count	99982.000000	99982.000000	99982.000000	99982.000000	99982.000000	99982.000000	99982.000000	99982.000
mean	0.585625	41.888076	0.074863	0.039427	27.320757	5.527529	138.057810	0.085
std	0.492616	22.517206	0.263172	0.194610	6.636853	1.070665	40.709469	0.278
min	0.000000	0.080000	0.000000	0.000000	10.010000	3.500000	80.000000	0.000
25%	0.000000	24.000000	0.000000	0.000000	23.630000	4.800000	100.000000	0.000
50%	1.000000	43.000000	0.000000	0.000000	27.320000	5.800000	140.000000	0.000
75%	1.000000	60.000000	0.000000	0.000000	29.580000	6.200000	159.000000	0.000
max	1.000000	80.000000	1.000000	1.000000	95.690000	9.000000	300.000000	1.000

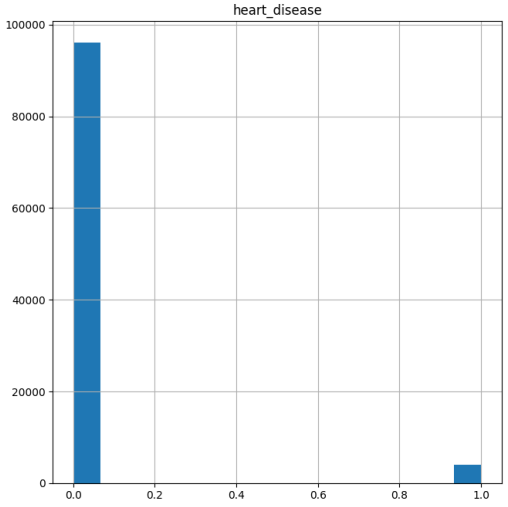
In [14]:
```
#Part 2: Plotting Histograms

df.hist(bins=15, figsize=(27, 28))
```

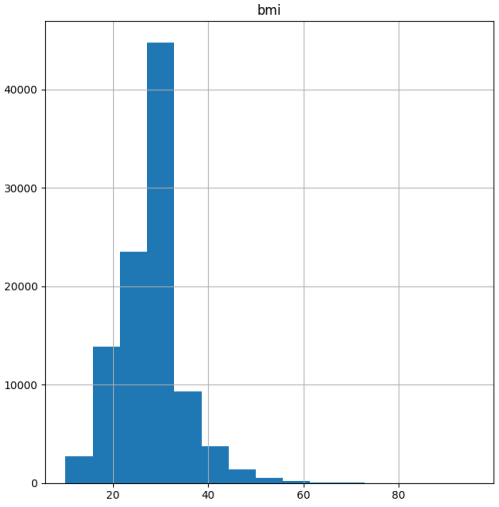
Out[14]:
```
array([[<Axes: title={'center': 'gender'}>,
        <Axes: title={'center': 'age'}>,
        <Axes: title={'center': 'hypertension'}>],
       [<Axes: title={'center': 'heart_disease'}>,
        <Axes: title={'center': 'bmi'}>,
        <Axes: title={'center': 'HbA1c_level'}>],
       [<Axes: title={'center': 'blood_glucose_level'}>,
        <Axes: title={'center': 'diabetes'}>, <Axes: >]], dtype=object)
```

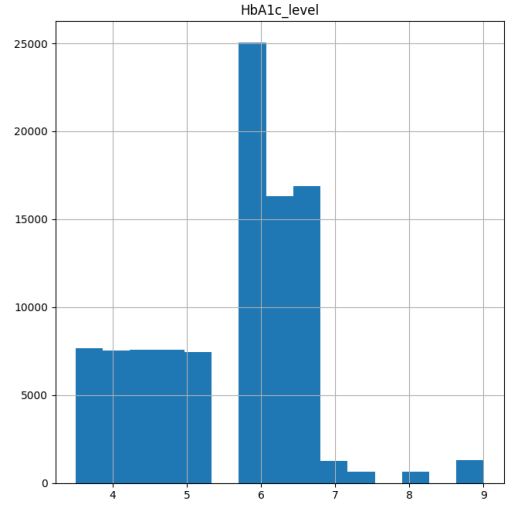

gender
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

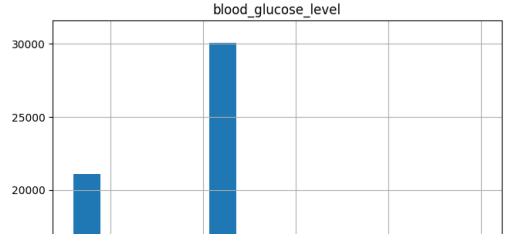

age
8000
6000
4000
2000
0
0
10
20
30
40
50
60
70
80

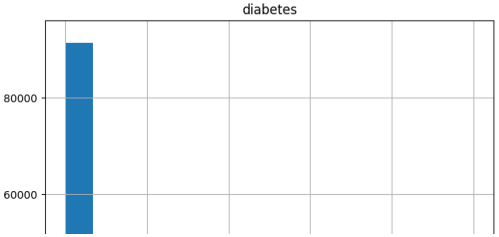

hypertension
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

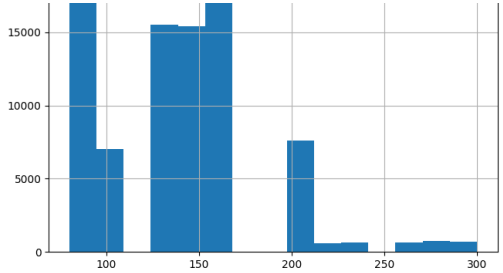

heart_disease
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

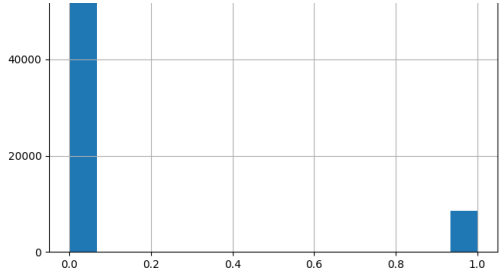

bmi
40000
30000
20000
10000
0
20
40
60
80


HbA1c_level
25000
20000
15000
10000
5000
0
4
5
6
7
8
9


blood_glucose_level
30000
25000
20000


diabetes
80000
60000





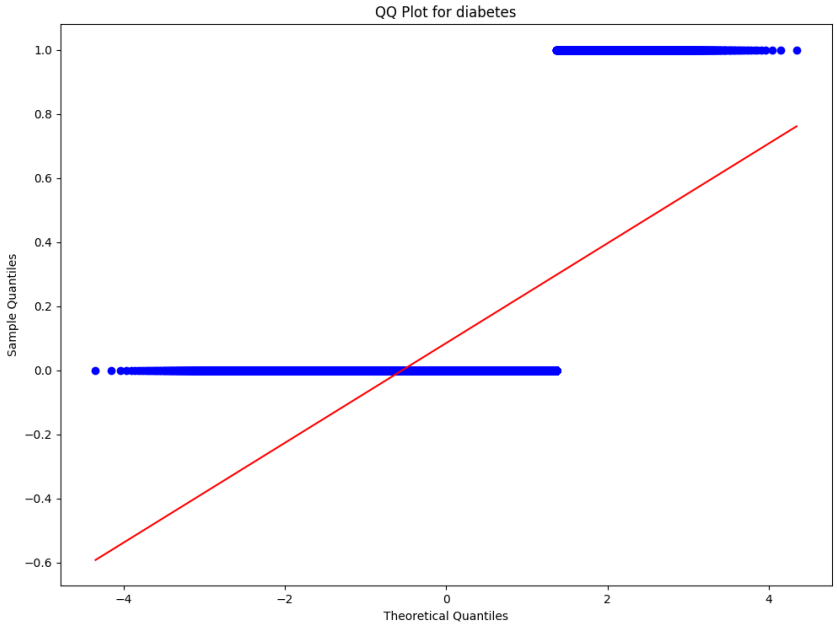
Analysis of the data through Histograms:

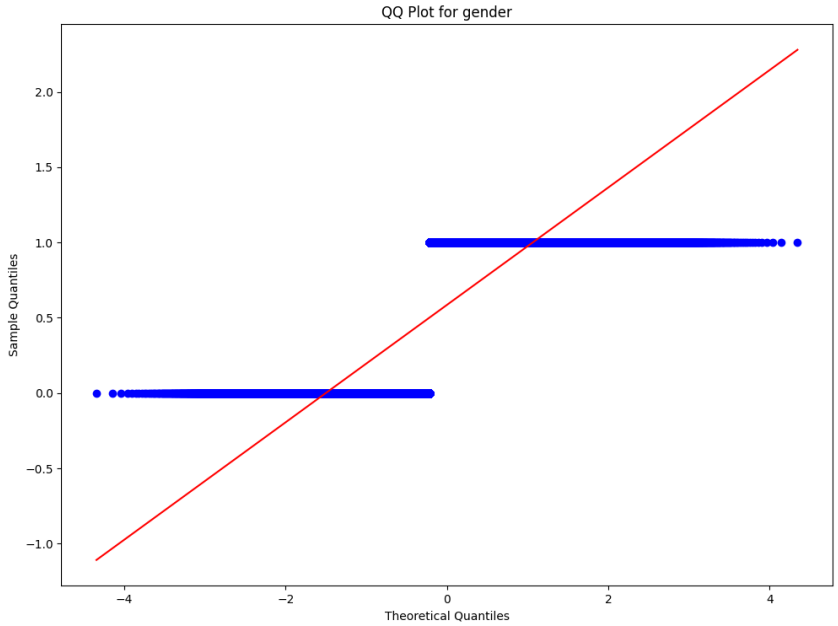
The histograms provide insights into the distribution of each variable, helping us understand the central tendency, spread, and shape of the data. They can also highlight potential outliers or skewness in the data.

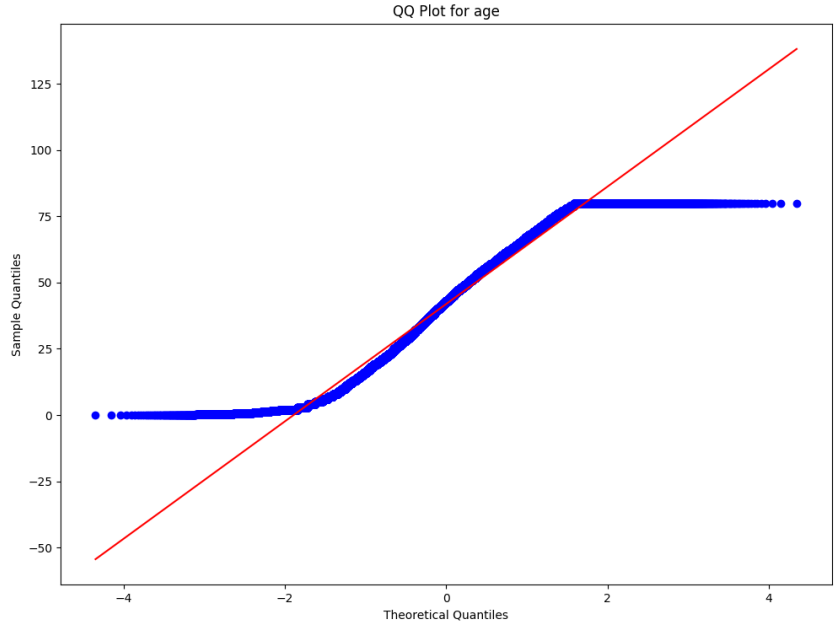
1. Most of the variables show skewness.
2. Variable that show Positively skewness are: hypertension, blood_glucose_level, diabetes, bmi and heart_disease.
3. Outliers can also be observed in various variables.

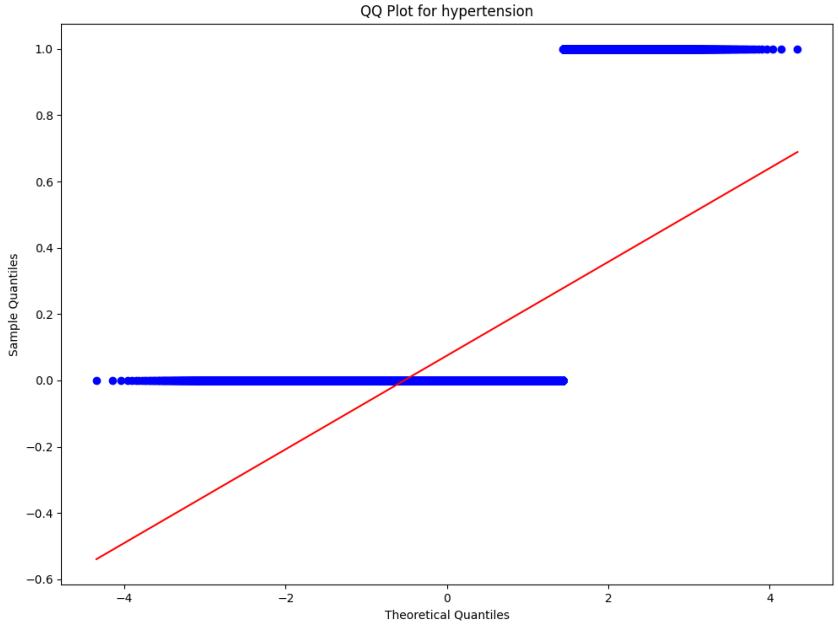
In [15]:
```
import scipy.stats as stats
```

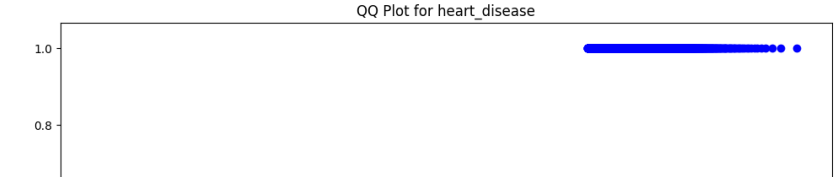
In [16]:
```
#Part 3: Plotting QQ Plots for each variable

selected_columns = df[["diabetes","gender","age","hypertension","heart_disease","bmi","HbA1c_level","blood

# Create a figure with subplots for QQ plots
qqplot = len(selected_columns)
fig, axes = plt.subplots(4 ,2 , figsize=(20,30))
axes = axes.ravel()


#Step 1: Creating qq plot for each variable in the selected_columns list by using for function:
for x, y in enumerate(selected_columns):
    ax = axes[x]
    data = selected_columns[y]

    # Create a QQ plot for the variable
    stats.probplot(data, dist="norm", plot=ax)
    ax.set_title(f'QQ Plot for {y}')
    ax.set_xlabel('Theoretical Quantiles')
    ax.set_ylabel('Sample Quantiles')

#Plot qq plots for each variable
plt.tight_layout()
plt.show()
```

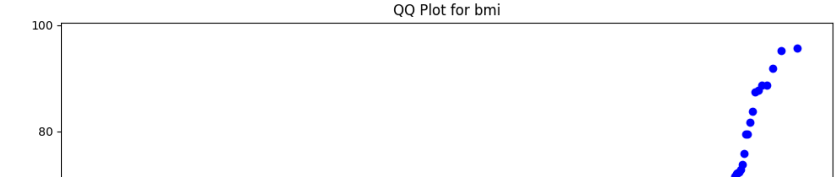

QQ Plot for diabetes
Sample Quantiles
Theoretical Quantiles

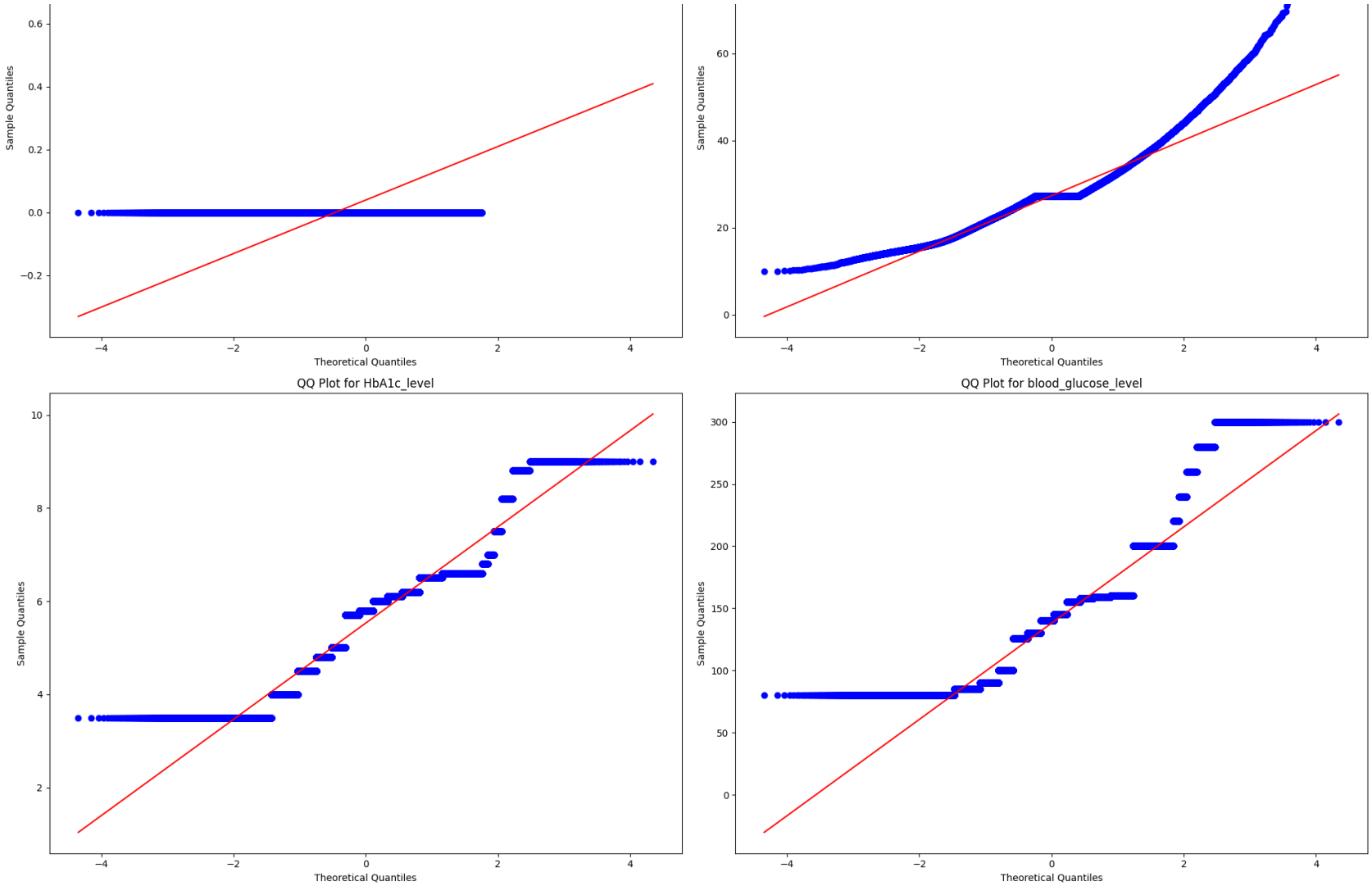

QQ Plot for gender
Sample Quantiles
Theoretical Quantiles


QQ Plot for age
Sample Quantiles
Theoretical Quantiles


QQ Plot for hypertension
Sample Quantiles
Theoretical Quantiles


QQ Plot for heart_disease


QQ Plot for bmi

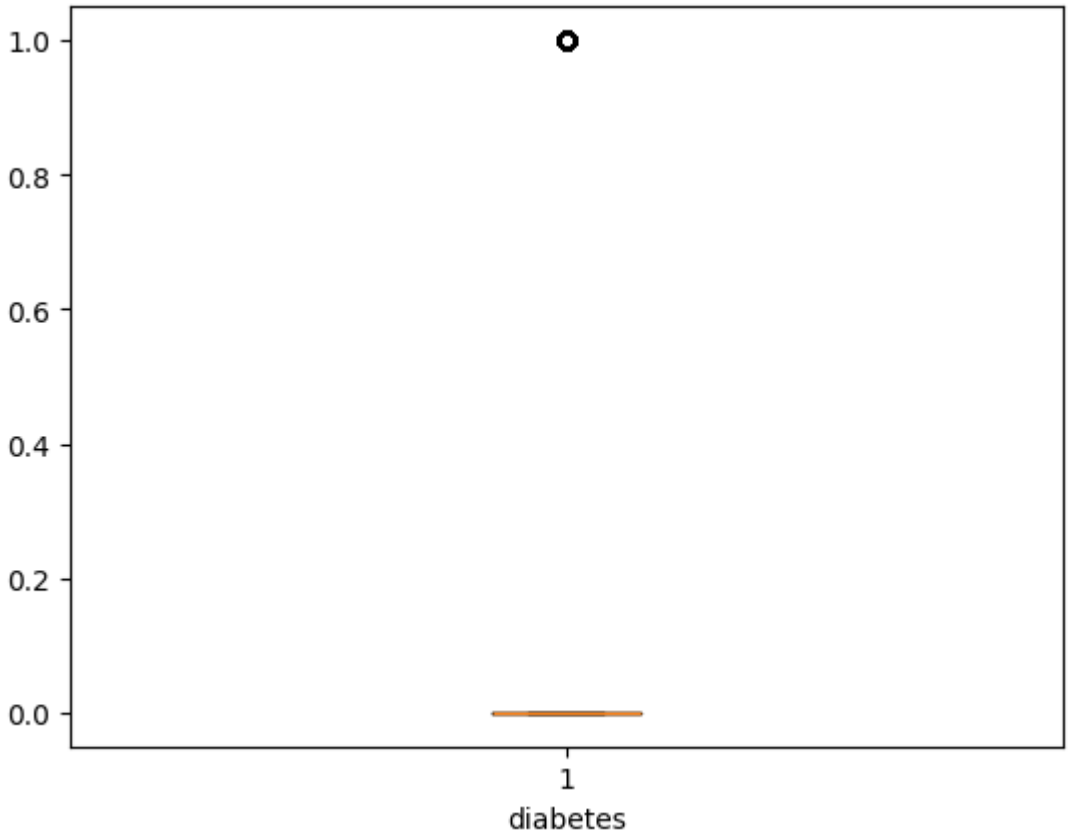


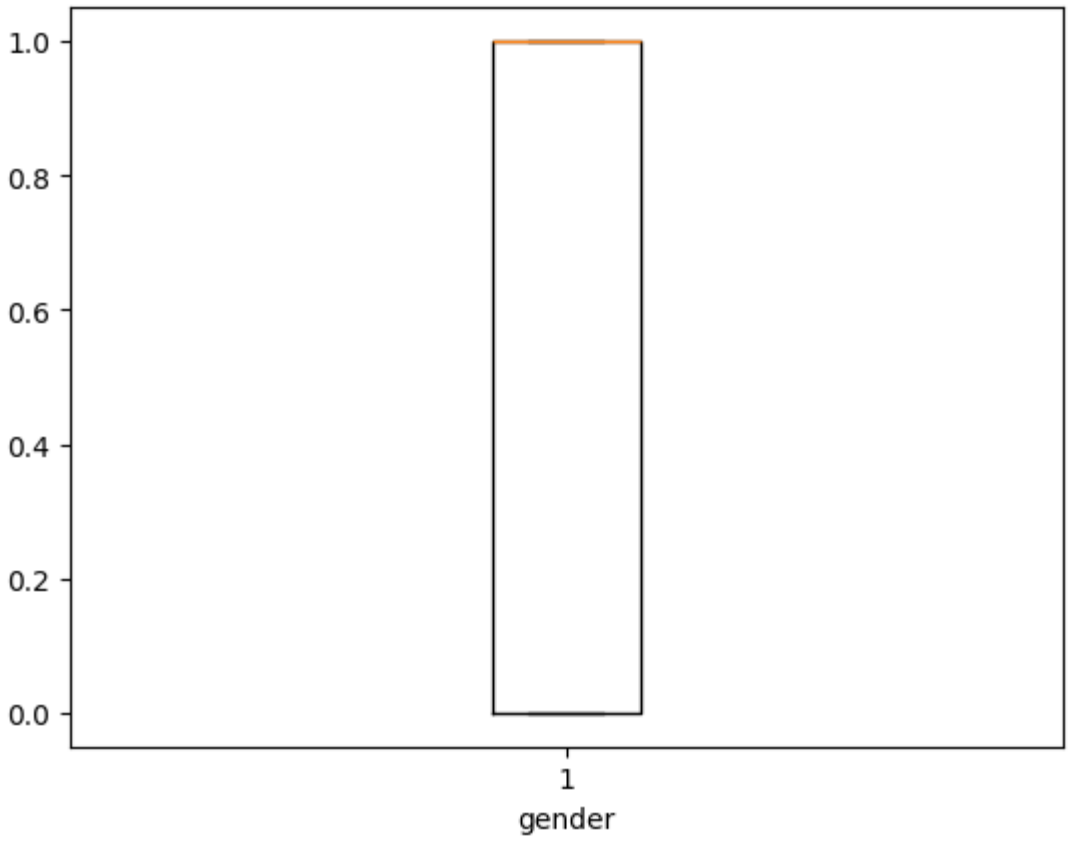
Analysing QQ plots for each variable in the data set:

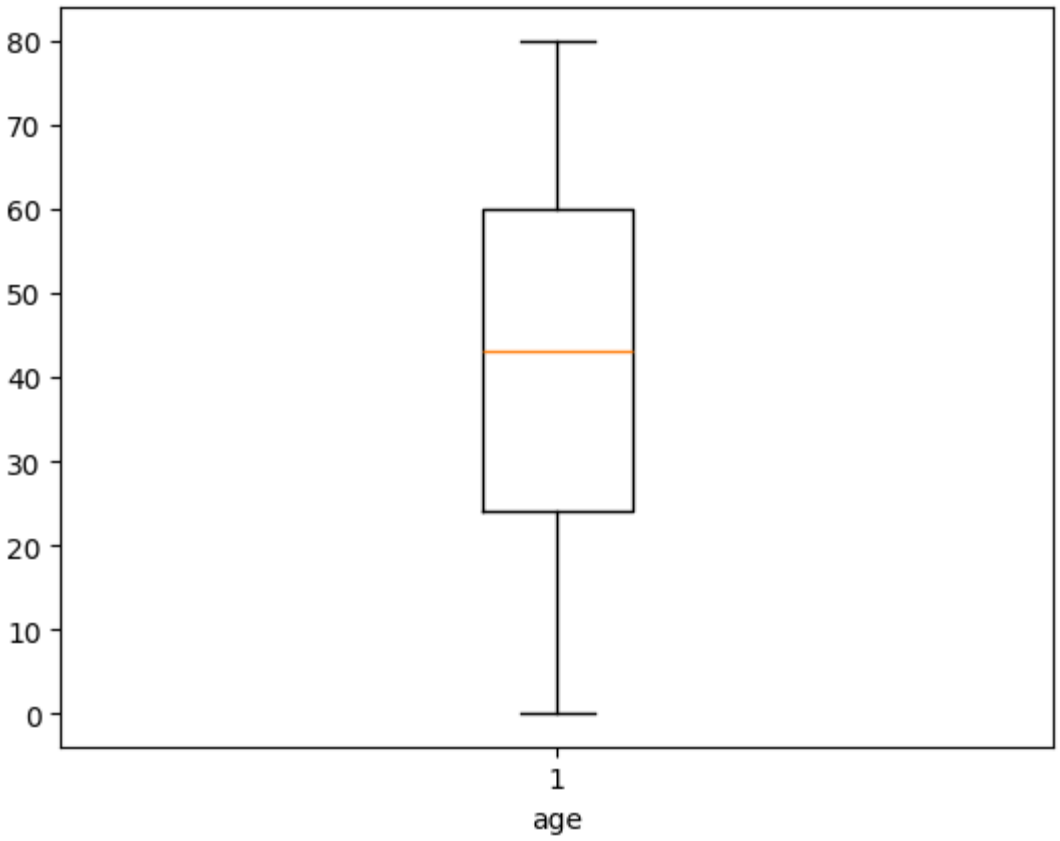
We have used this graphical tool used to assess whether the dataset follows a specific theoretical distribution, typically the normal distribution. If the points on the QQ plot fall along or near a straight line, it suggests that the data follows the theoretical distribution. If they deviate from a straight line, it indicates a departure from the theoretical distribution.

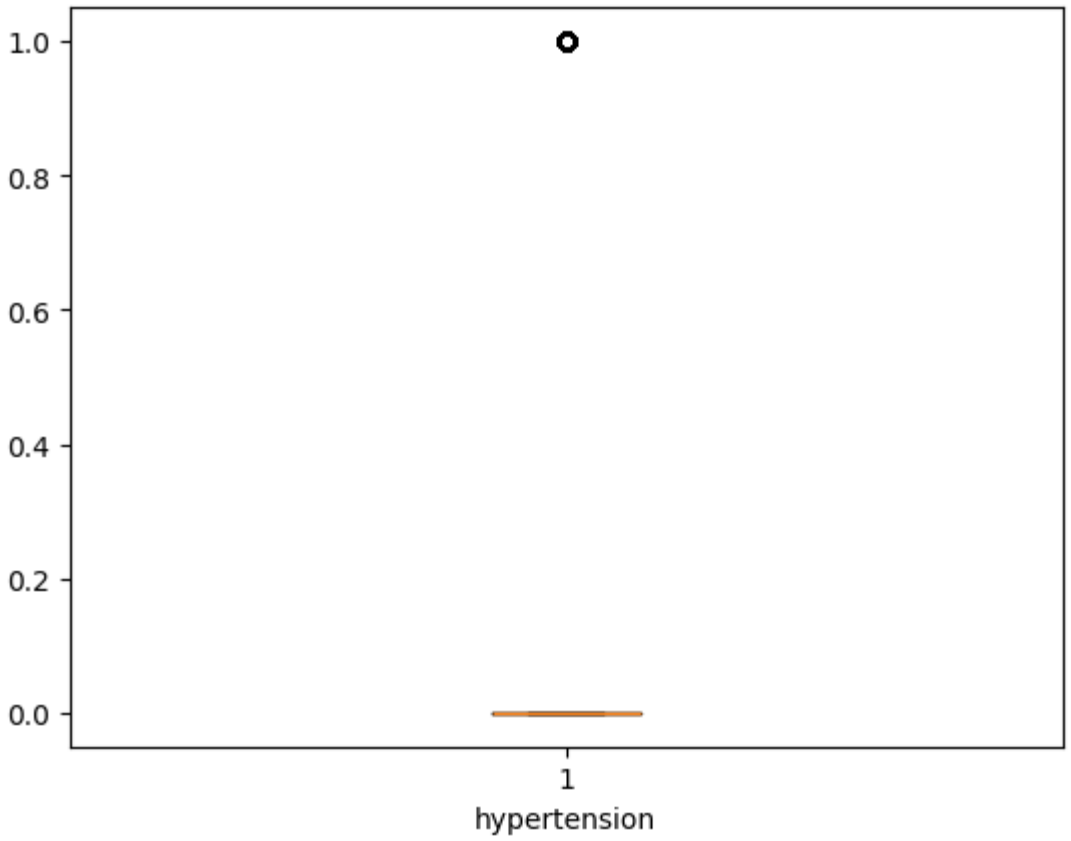
Variables like bmi, HbA1c_level, blood_glucose_level, it indicates non-normality.

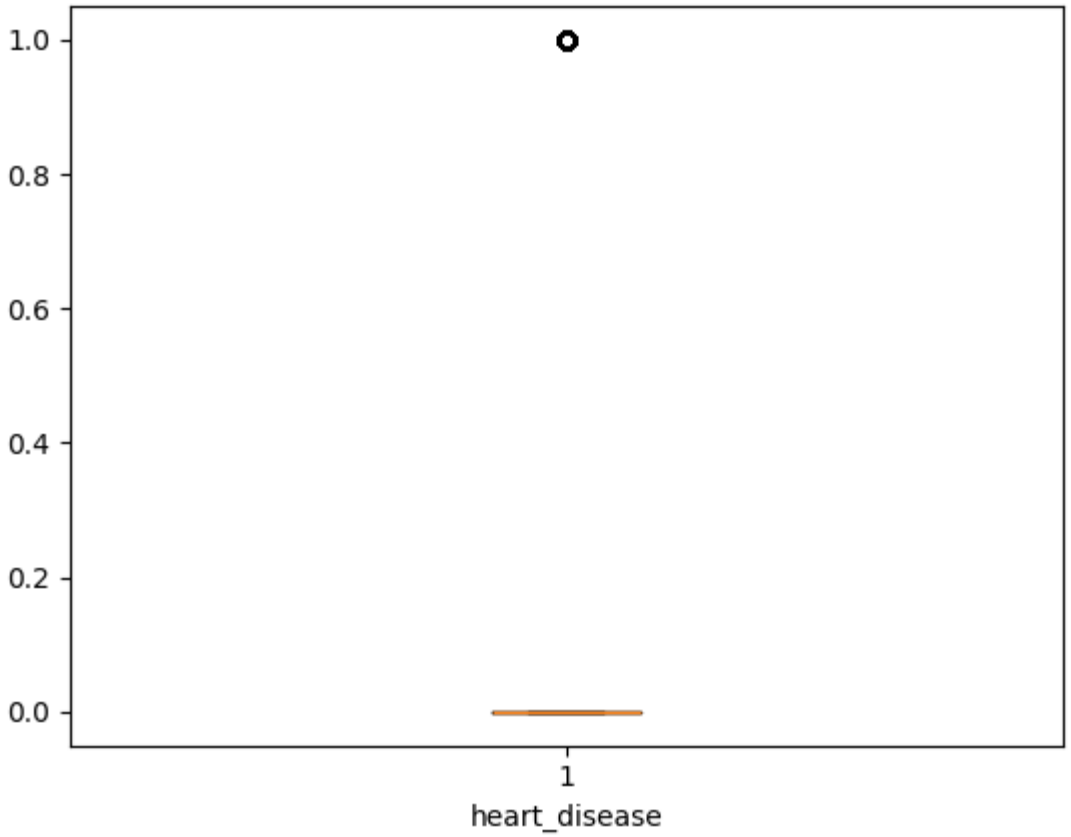
In [16]:
```
#Part4: Box Plots
selected_columns = df[["diabetes","gender","age","hypertension","heart_disease","bmi","HbA1c_level","blood_

for a in selected_columns:
  plt.boxplot(selected_columns[a])
  plt.xlabel(a)
  plt.show()
```

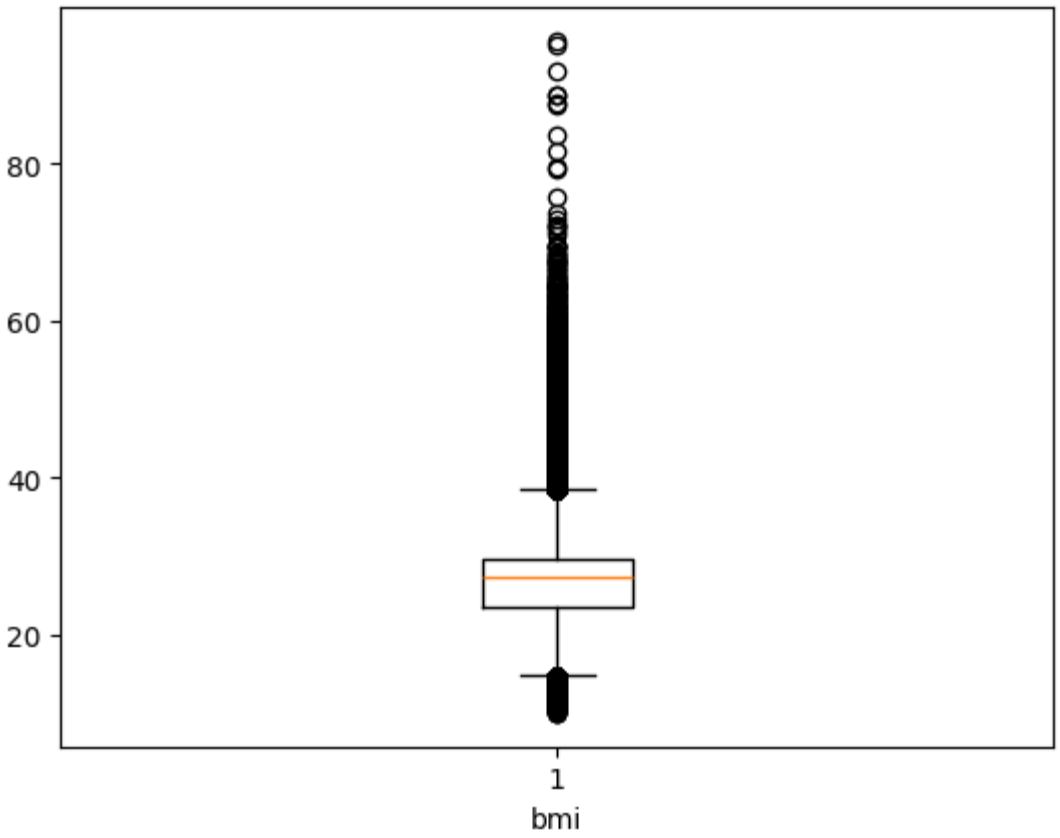


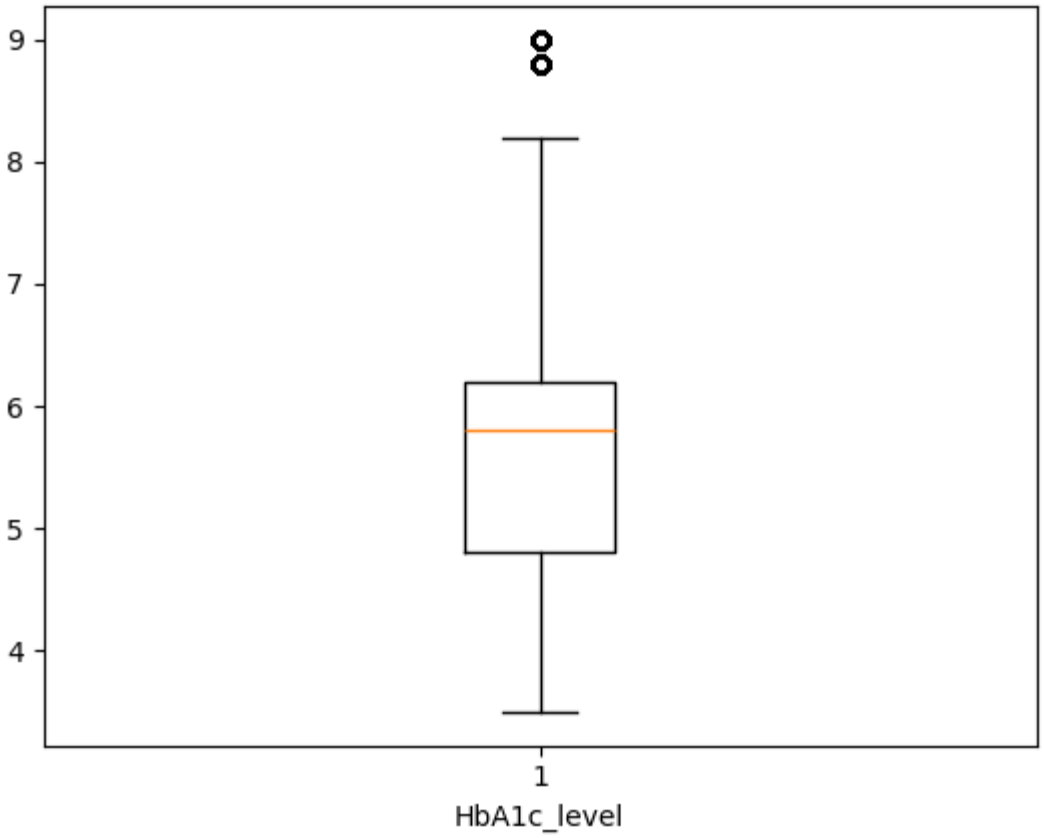

1.0
0.8
0.6
0.4
0.2
0.0
1
gender

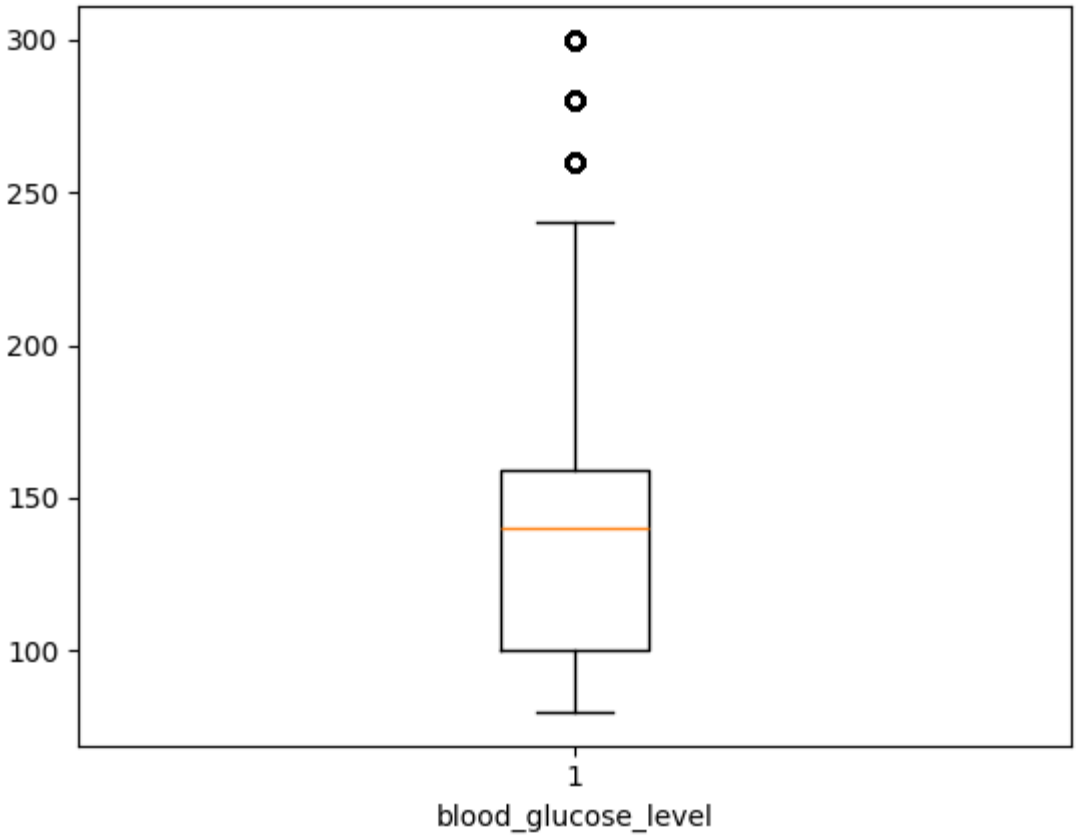

80
70
60
50
40
30
20
10
0
1
age


1.0
0.8
0.6
0.4
0.2
0.0
1
hypertension


1.0
0.8
0.6
0.4
0.2
0.0
1
heart_disease


80
60
40
20
1
bmi


9
8
7
6
5
4
1
HbA1c_level


300
250
200
150
100
1
blood_glucose_level

Analysis of box plots:

Box plots are useful for visualizing the distribution and spread of numerical variables, identifying potential outliers, and comparing the central tendencies of different groups. Key observations from box plots:

Median (line inside the box): Represents the central tendency.

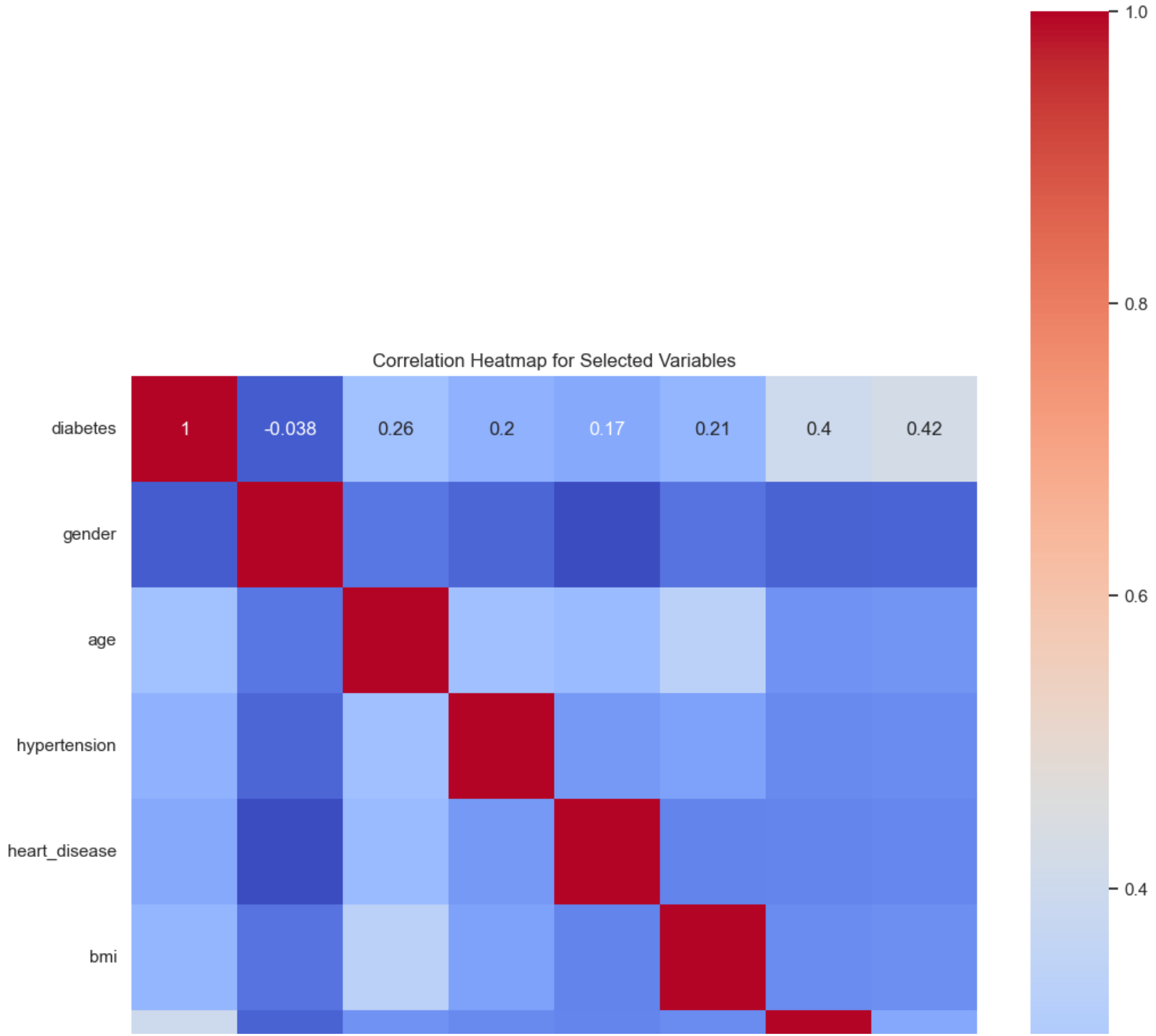
Interquartile Range (IQR): The range between the first quartile (Q1) and third quartile (Q3), providing a measure of data spread.

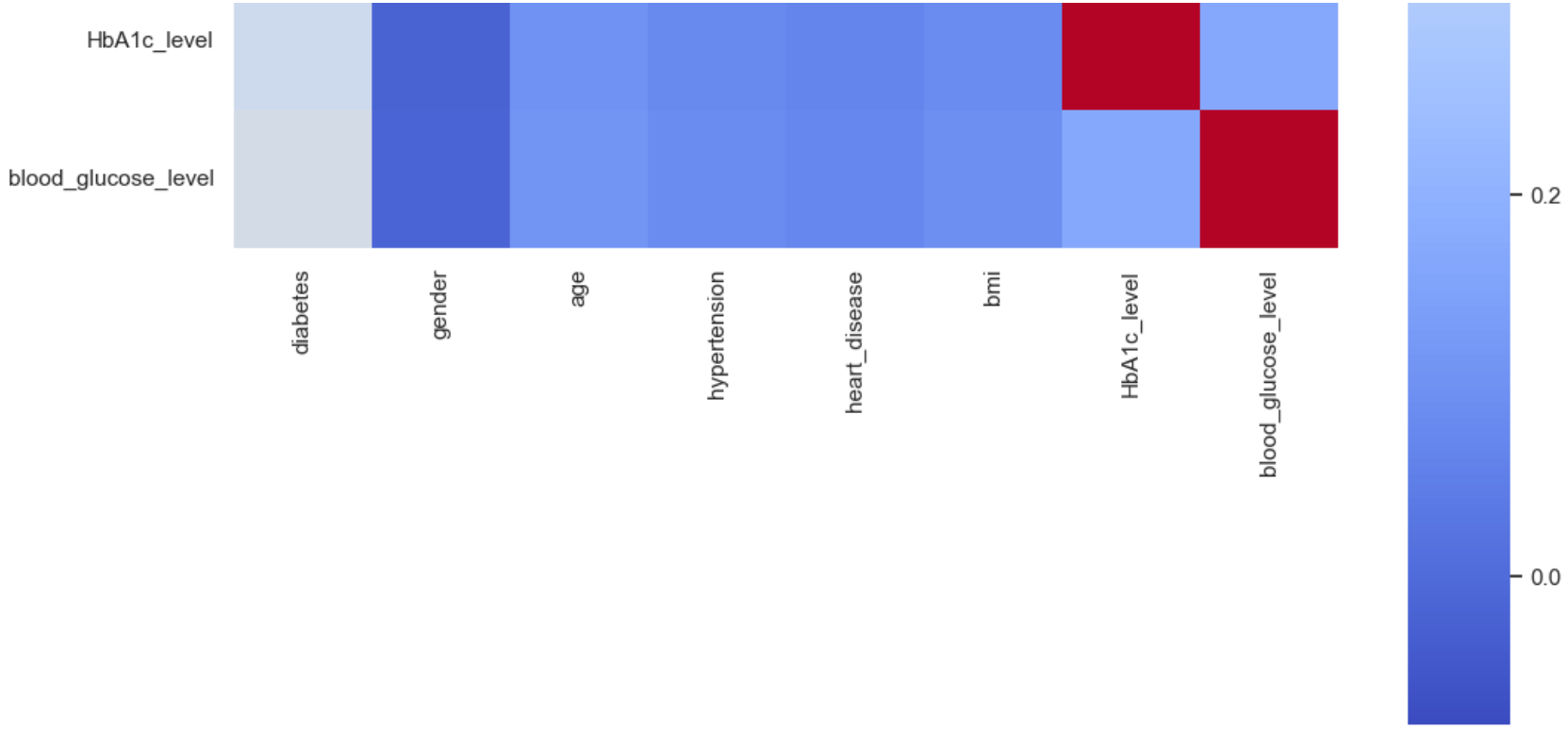
Whiskers: Lines extending from the box indicate the range of data, excluding outliers.

Outliers: Points beyond the whiskers, potentially indicating extreme values.

These box plots help us visually compare the distribution of numerical variables for individuals with and without diabetes. As we can see even after separating diabetes there seems to be outliers which are affecting the other boxplots. Hence we will have to map the variables with large numbers separately (can identify using mean value).

In [18]:
```
# Part 6: Using the heatmap function to create a correlation heatmap

correlation_matrix = selected_columns.corr()

# Create a heatmap
sns.set(font_scale=1)
plt.figure(figsize=(12, 18))

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", cbar=True, square=True)

plt.title("Correlation Heatmap for Selected Variables")
plt.show()
```


Correlation Heatmap for Selected Variables
diabetes
gender
age
hypertension
heart_disease
bmi
1
-0.038
0.26
0.2
0.17
0.21
0.4
0.42
1.0
0.8
0.6
0.4



Analysis of Correlation heat map:

A correlation heatmap is a visual representation of the pairwise correlations between variables in a dataset. Analyzing a correlation heatmap can provide valuable insights into the relationships between variables.

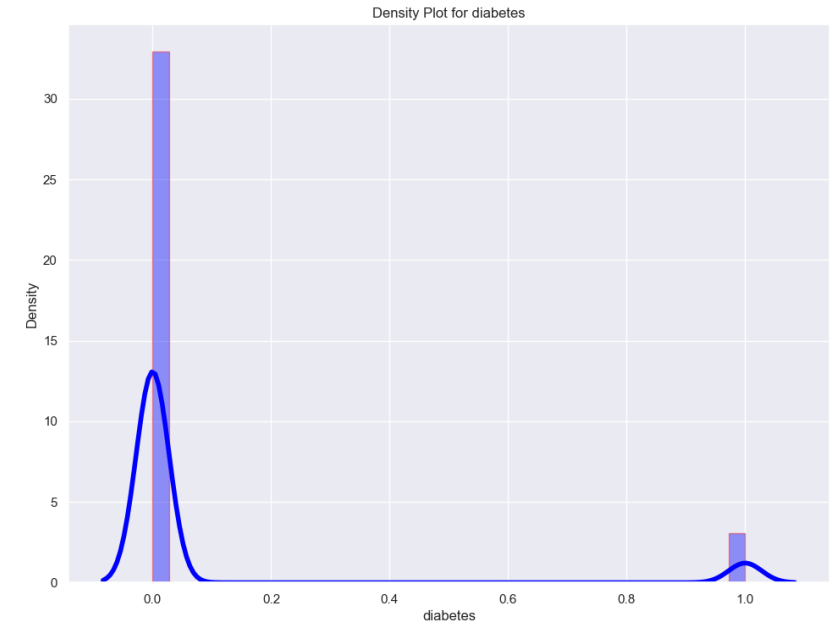
1.Positive correlation: diabetes tends to have a slightly high correlation with variable blood_glucose_level.

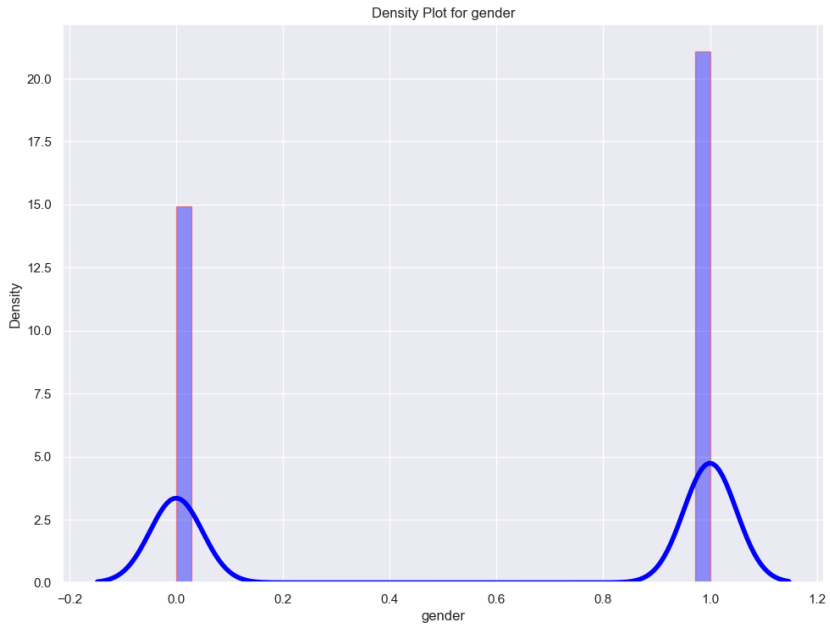
2.Negative correlation: There is a negative correlation between diabetes and gender. However this negative correlation is not that high and thus, might not impact the model.

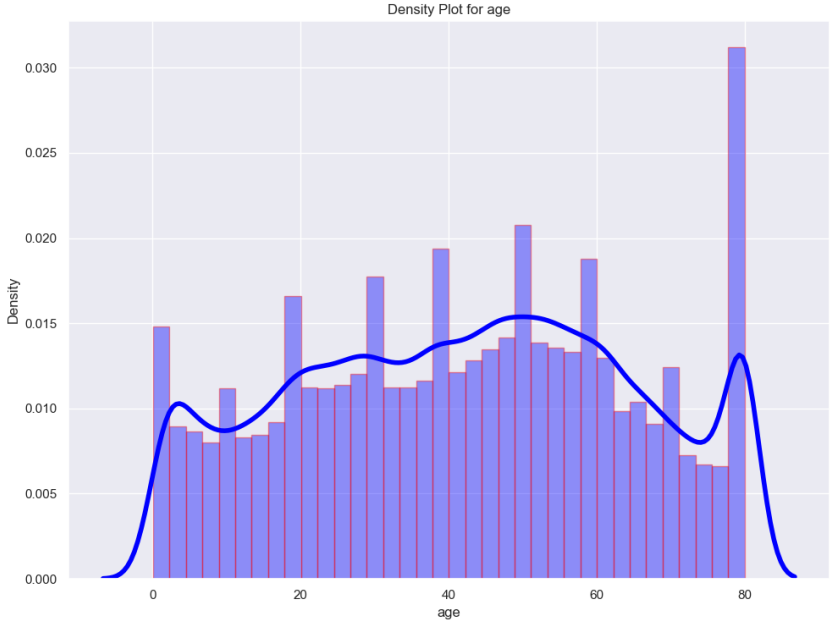
In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a figure with subplots for Density estimates
fig, axes = plt.subplots(4, 2, figsize=(20, 30))
axes = axes.ravel()

# Creating density plots for each variable in the selected_columns list using a for loop:
for a, b in enumerate(selected_columns):
    ax = axes[a]
    data = selected_columns[b]

    # Create a density plot for the variable
    sns.distplot(data, ax=ax, hist=True, kde=True,
                 bins=int(180/5), color='blue',
                 hist_kws={'edgecolor': 'red'},
                 kde_kws={'linewidth': 4})
    ax.set_title(f'Density Plot for {b}')
    ax.set_xlabel(b)
    ax.set_ylabel('Density')

plt.tight_layout()
plt.show()
```

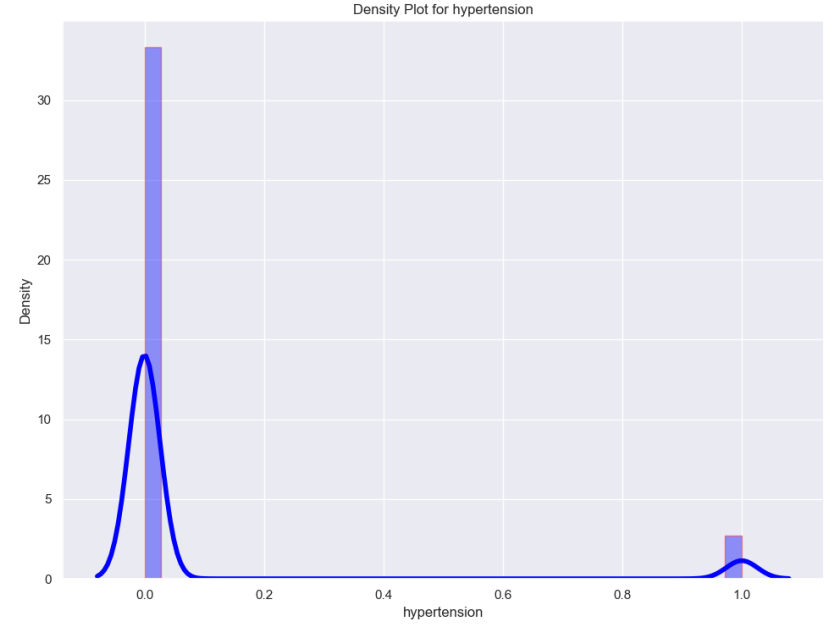
```
/var/folders/5c/3157mbn910b2gp0sphv4k8xm0000gn/T/ipykernel_94476/2118635490.py:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data, ax=ax, hist=True, kde=True,
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/var/folders/5c/3157mbn910b2gp0sphv4k8xm0000gn/T/ipykernel_94476/2118635490.py:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data, ax=ax, hist=True, kde=True,
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/var/folders/5c/3157mbn910b2gp0sphv4k8xm0000gn/T/ipykernel_94476/2118635490.py:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

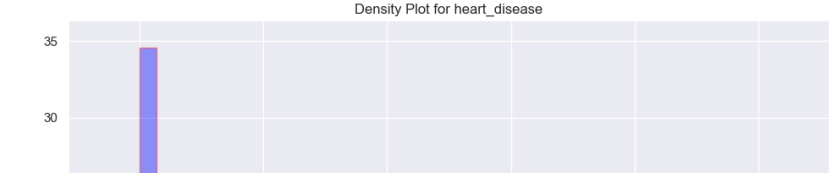
```
  with pd.option_context('mode.use_inf_as_na', True):
/var/folders/5c/3157mbn910b2gp0sphv4k8xm0000gn/T/ipykernel_94476/2118635490.py:14: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data, ax=ax, hist=True, kde=True,
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

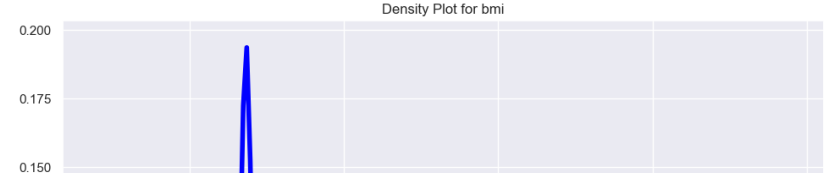

Density Plot for diabetes
Density
diabetes

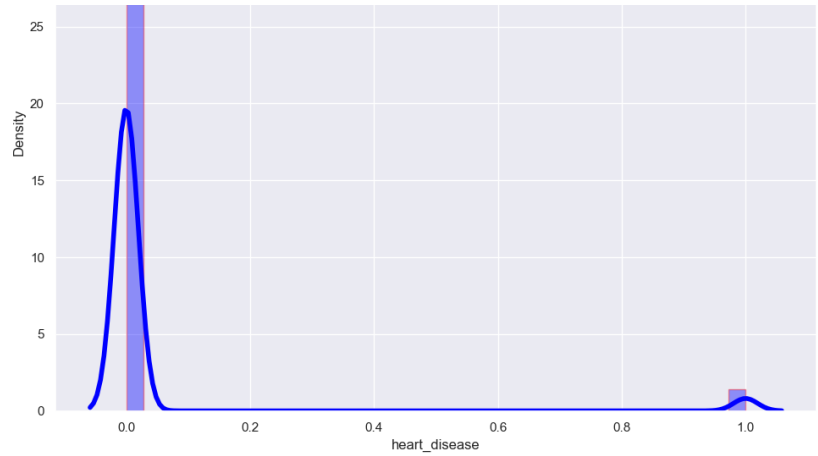

Density Plot for gender
Density
gender

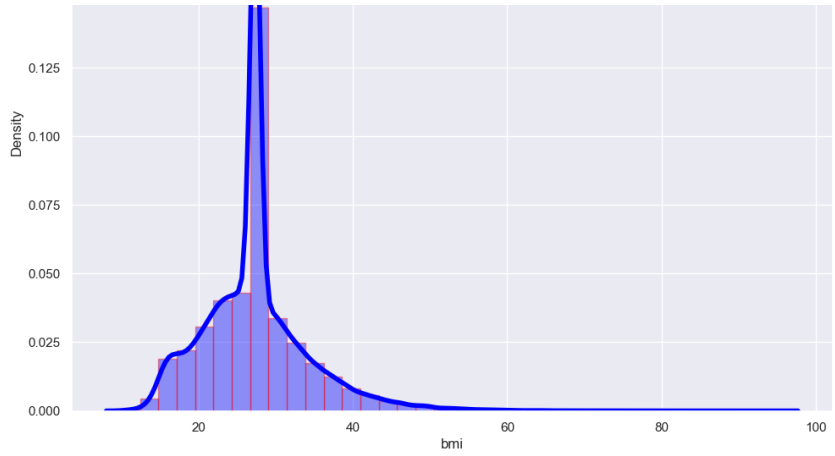

Density Plot for age
Density
age

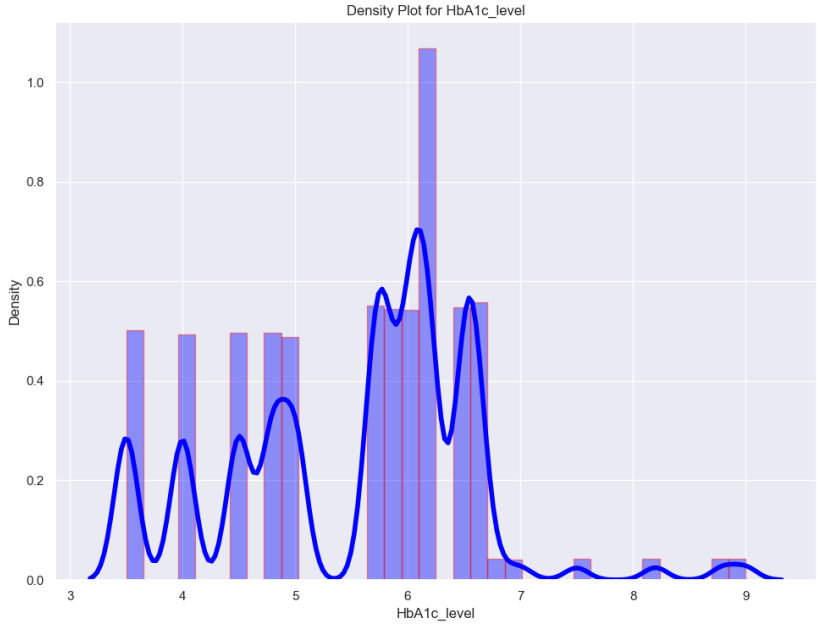

Density Plot for hypertension
Density
hypertension


Density Plot for heart_disease


Density Plot for bmi







Analysis of Density plot:

Density plots are data visualization tools used to understand the distribution of a dataset. The following observations are made after analysing density estimation plots:

Key observations from density plots:

Peak: Indicates the mode or central tendency of the distribution.

In [20]:
```
import math
```

In [21]:
```
selected_variables= ["diabetes","gender","age","hypertension","heart_disease","bmi","HbA1c_level","blood_g
```

In [22]:
```
from fitter import Fitter
f = Fitter(df[selected_variables])
f.fit()
```

2.Normality Assessment: A proper and symmetric density plot can be an indication that the data follows a normal distribution. The variable 'age' is the closest to normal distribution.

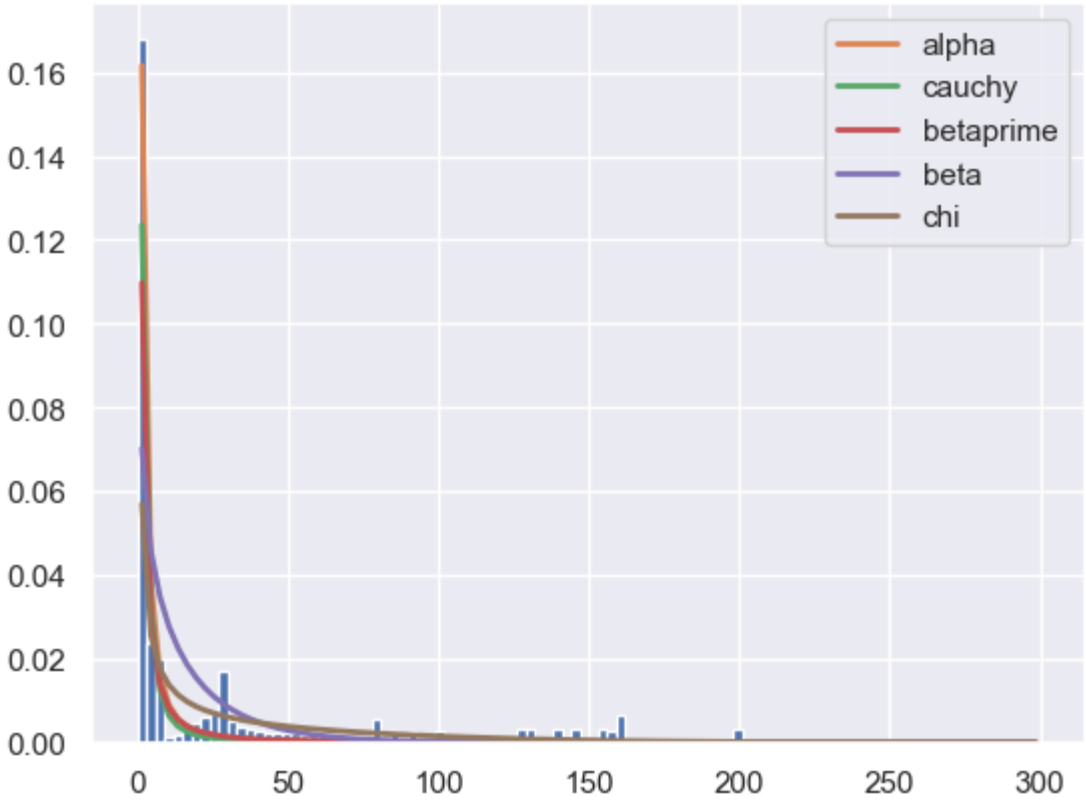
```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED burr distribution (taking more than 30 seconds)
SKIPPED burr12 distribution (taking more than 30 seconds)
SKIPPED crystalball distribution (taking more than 30 seconds)
SKIPPED exponpow distribution (taking more than 30 seconds)
SKIPPED exponweib distribution (taking more than 30 seconds)
SKIPPED f distribution (taking more than 30 seconds)
SKIPPED fatiguelife distribution (taking more than 30 seconds)
SKIPPED fisk distribution (taking more than 30 seconds)
SKIPPED foldcauchy distribution (taking more than 30 seconds)
SKIPPED gamma distribution (taking more than 30 seconds)
SKIPPED gausshyper distribution (taking more than 30 seconds)
SKIPPED genexpon distribution (taking more than 30 seconds)
SKIPPED genextreme distribution (taking more than 30 seconds)
SKIPPED gengamma distribution (taking more than 30 seconds)
SKIPPED genhalflogistic distribution (taking more than 30 seconds)
SKIPPED genhyperbolic distribution (taking more than 30 seconds)
SKIPPED geninvgauss distribution (taking more than 30 seconds)
SKIPPED genlogistic distribution (taking more than 30 seconds)
SKIPPED gennorm distribution (taking more than 30 seconds)
SKIPPED genpareto distribution (taking more than 30 seconds)
SKIPPED gibrat distribution (taking more than 30 seconds)
SKIPPED gilbrat distribution (taking more than 30 seconds)
SKIPPED gompertz distribution (taking more than 30 seconds)
SKIPPED halfcauchy distribution (taking more than 30 seconds)
SKIPPED halfgennorm distribution (taking more than 30 seconds)
SKIPPED halflogistic distribution (taking more than 30 seconds)
SKIPPED halfnorm distribution (taking more than 30 seconds)
SKIPPED invgamma distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED invgauss distribution (taking more than 30 seconds)
SKIPPED invweibull distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED kappa3 distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED laplace_asymmetric distribution (taking more than 30 seconds)
SKIPPED levy distribution (taking more than 30 seconds)
SKIPPED levy_l distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED loglaplace distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
```

```
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/scipy/integrate/_quadpack_py.py:1225: Integratio
nWarning: The integral is probably divergent, or slowly convergent.
  quad_r = quad(f, low, high, args=args, full_output=self.full_output,
```

In [23]:
```
f.summary()
```

Out[23]:

	sumsquare_error	aic	bic	kl_div	ks_statistic	ks_pvalue
alpha	0.000852	1861.044045	1895.820606	inf	0.281524	0.0
cauchy	0.002709	1967.274122	1990.458496	inf	0.425675	0.0
betaprime	0.004020	1793.761376	1840.130123	inf	0.401884	0.0
beta	0.012057	2183.828999	2230.197747	inf	0.401884	0.0
chi	0.012931	1539.447980	1574.224541	inf	0.401884	0.0



We have our table with various statistical metrics for different probability distributions (e.g., alpha, cauchy, betaprime, beta, chi). Each row in the table corresponds to a different distribution, and the columns represent different statistical measures.

Let's interpret the columns:

sumsquare_error: This could be a measure of the sum of squared errors between observed and expected values. Smaller values indicate a better fit.

aic (Akaike Information Criterion): A metric used for model selection, balancing goodness of fit with model complexity. Lower AIC values suggest better models.

bic (Bayesian Information Criterion): Similar to AIC but penalizes model complexity more strongly. Lower BIC values indicate better models.

kl_div (Kullback-Leibler Divergence): A measure of how one probability distribution diverges from a second. "inf" suggests an undefined or infinite value, which might happen if the two distributions don't have overlapping support.

ks_statistic (Kolmogorov-Smirnov statistic): A measure of the maximum discrepancy between the empirical distribution function of the sample and the cumulative distribution function of the theoretical distribution. Values closer to 0 indicate a better fit.

ks_pvalue (Kolmogorov-Smirnov p-value): The p-value associated with the Kolmogorov-Smirnov test. A low p-value suggests that the sample distribution differs from the theoretical distribution.

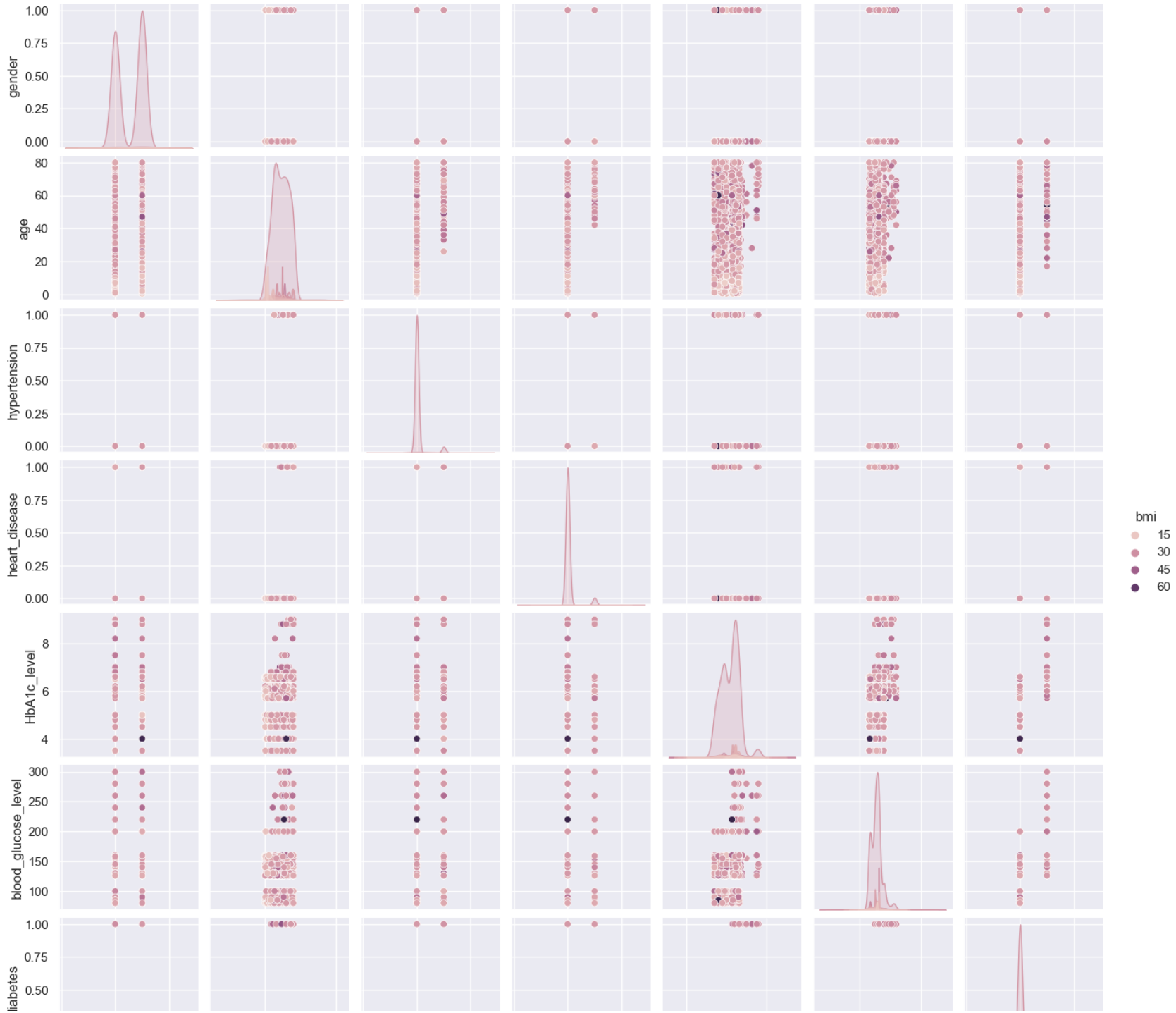
In [17]:
```
#Part 5: scatterplot
# Randomly select 1000 rows
sample_df = df.sample(n=1000, random_state=42)

# Create scatterplot
sns.set()
sns.pairplot(sample_df, size=2.0, hue='bmi')
plt.show()
```

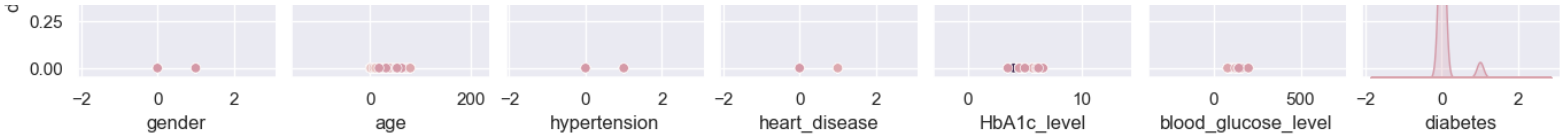
```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:2095: UserWarning: The `size
` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
```

```
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
```

```
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
```

```
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
```


gender
age
hypertension
heart_disease
HbA1c_level
blood_glucose_level
liabetes
bmi
15
30
45
60



We have a pairplot for a randomly selected sample of 1000 rows from our DataFrame. The pairplot display scatterplots for different pairs of variables, with points colored based on the 'bmi' column.

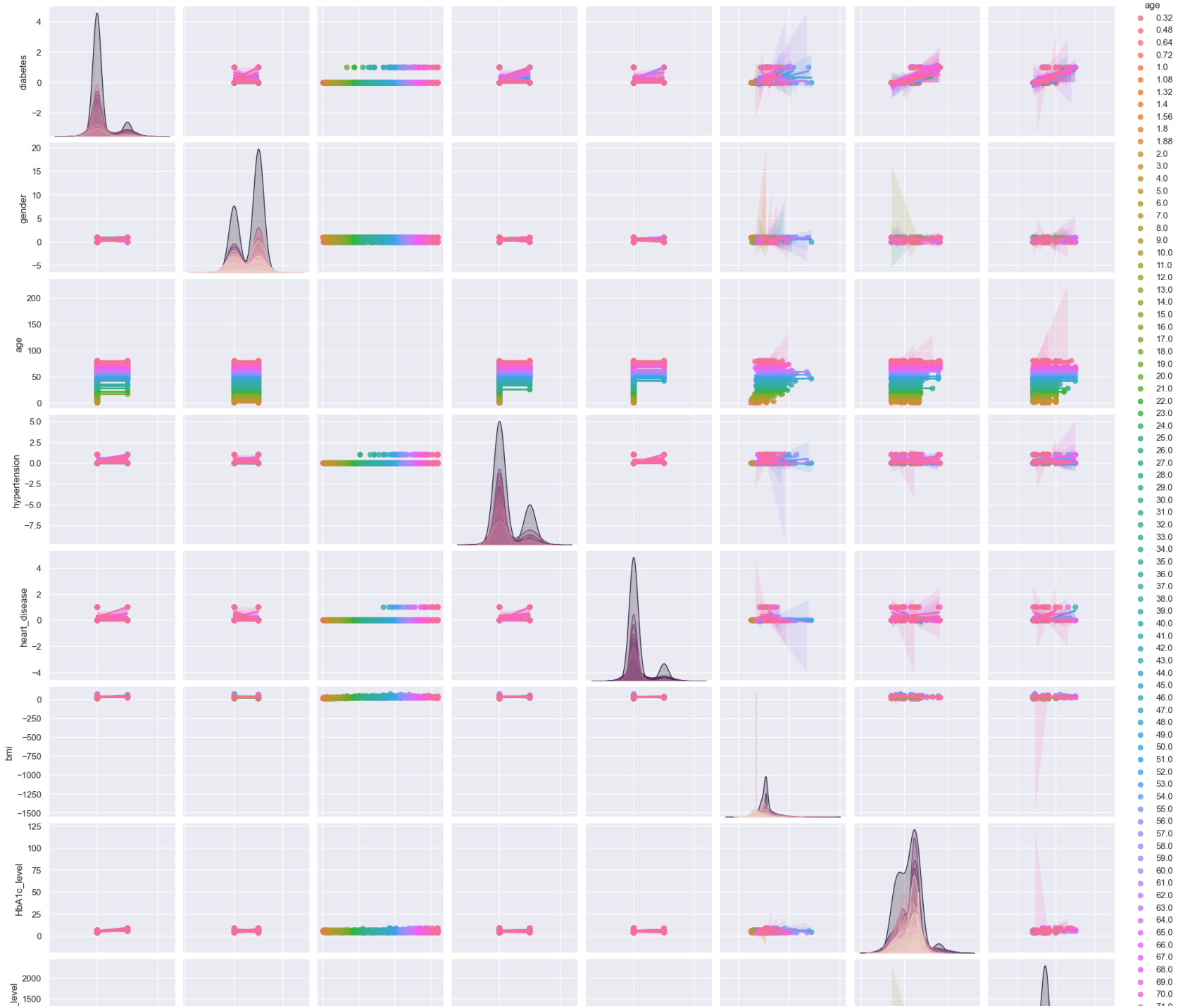
The scatter plot displays individual data points on a two-dimensional graph. The main purpose of a scatter plot is to visually assess the relationship between two continuous variables. Each point on the plot represents a unique combination of values for the two variables being compared.

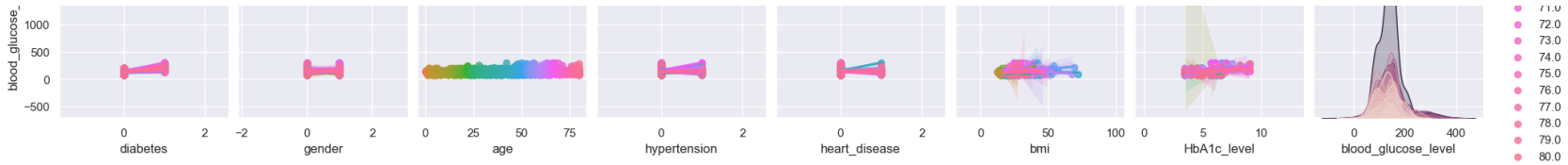
In [18]:
```
#Part 6: Scatterplot Matrix

sns.pairplot(sample_df ,vars=["diabetes","gender","age","hypertension","heart_disease","bmi","HbA1c_level"
```

```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
```

```
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_cate
gorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtyp
e) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/bharatsingh/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf
_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before oper
ating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x15edbdb90>


diabetes
gender
age
hypertension
heart_disease
bmi
HbA1c_level
_level
age
0.32
0.48
0.64
0.72
1.0
1.08
1.32
1.4
1.56
1.8
1.88
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0
24.0
25.0
26.0
27.0
28.0
29.0
30.0
31.0
32.0
33.0
34.0
35.0
36.0
37.0
38.0
39.0
40.0
41.0
42.0
43.0
44.0
45.0
46.0
47.0
48.0
49.0
50.0
51.0
52.0
53.0
54.0
55.0
56.0
57.0
58.0
59.0
60.0
61.0
62.0
63.0
64.0
65.0
66.0
67.0
68.0
69.0
70.0



The scatterplot Matrix displays pairwise relationships between multiple variables. Each cell in the matrix represents the scatterplot between two specific variables, and the diagonal cells typically show the distribution of each individual variable.

By examining the scatterplots, we can identify patterns, trends, or clusters in the data. It is helpful for assessing the correlation between variables. Correlation can be visually assessed by the direction and strength of the linear relationships between points in the scatterplots. Outliers, or data points that deviate significantly from the general pattern, can be easily identified here.

In [ ]:

In [2]:
```
#Importing relevant functions
import statsmodels.api as sm
import statsmodels as sms
import seaborn as sns
import statsmodels.formula.api as smf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import os
%matplotlib inline
```

II. Qualitative Dependent Variable Models

In [3]:
```
data1 = pd.read_csv(r"/Users/bharatsingh/Downloads/diabetes_prediction_dataset.csv")
```

In [4]:
```
data1
```

Out[4]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

In [5]:
```
data1 = data1.dropna()
```

In [6]:
```
data1.head()
```

Out[6]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

In [7]:
```
data1 = data1[data1['gender']!='Other']
```

In [8]:
```
# Assuming 'Male' is represented by 0 and 'Female' is represented by 1
data1['gender'] = data1['gender'].map({'Male': 0, 'Female': 1})

# If you want to switch the values (0 for 'Female' and 1 for 'Male'), you can use:
# data1['gender'] = 1 - data1['gender']

# Print the modified DataFrame
print(data1)
```

```
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0           1  80.0             0              1           never  25.19
1           1  54.0             0              0         No Info  27.32
2           0  28.0             0              0           never  27.32
3           1  36.0             0              0         current  23.45
4           0  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995       1  80.0             0              0         No Info  27.32
99996       1   2.0             0              0         No Info  17.37
99997       0  66.0             0              0          former  27.83
99998       1  24.0             0              0           never  35.42
99999       1  57.0             0              0         current  22.43

       HbA1c_level  blood_glucose_level  diabetes
0              6.6                  140         0
1              6.6                   80         0
2              5.7                  158         0
3              5.0                  155         0
4              4.8                  155         0
...            ...                  ...       ...
99995          6.2                   90         0
99996          6.5                  100         0
99997          5.7                  155         0
99998          4.0                  100         0
99999          6.6                   90         0

[99982 rows x 9 columns]
```

In [9]:
```
data1['gender'].isnull().sum()
```

Out[9]:
```
0
```

In [10]:
```
data1
```

Out[10]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1	80.0	0	1	never	25.19	6.6	140	0
1	1	54.0	0	0	No Info	27.32	6.6	80	0
2	0	28.0	0	0	never	27.32	5.7	158	0
3	1	36.0	0	0	current	23.45	5.0	155	0
4	0	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	1	80.0	0	0	No Info	27.32	6.2	90	0
99996	1	2.0	0	0	No Info	17.37	6.5	100	0
99997	0	66.0	0	0	former	27.83	5.7	155	0
99998	1	24.0	0	0	never	35.42	4.0	100	0
99999	1	57.0	0	0	current	22.43	6.6	90	0

99982 rows × 9 columns

In [11]:
```
df= data1.drop(['smoking_history'], axis=1)
df.head()
```

Out[11]:

	gender	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1	80.0	0	1	25.19	6.6	140	0
1	1	54.0	0	0	27.32	6.6	80	0
2	0	28.0	0	0	27.32	5.7	158	0
3	1	36.0	0	0	23.45	5.0	155	0
4	0	76.0	1	1	20.14	4.8	155	0

In [12]:
```
# Dont run this for now
#Dropping columns with Null Value
columns_to_dropna = ['gender']
df = data1.dropna(subset=columns_to_dropna)

print("\nDataFrame after dropping null values:")
print(df)
```

```
DataFrame after dropping null values:
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0           1  80.0             0              1           never  25.19
1           1  54.0             0              0         No Info  27.32
2           0  28.0             0              0           never  27.32
3           1  36.0             0              0         current  23.45
4           0  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995       1  80.0             0              0         No Info  27.32
99996       1   2.0             0              0         No Info  17.37
99997       0  66.0             0              0          former  27.83
99998       1  24.0             0              0           never  35.42
99999       1  57.0             0              0         current  22.43

       HbA1c_level  blood_glucose_level  diabetes
0              6.6                  140         0
1              6.6                   80         0
2              5.7                  158         0
3              5.0                  155         0
4              4.8                  155         0
...            ...                  ...       ...
99995          6.2                   90         0
99996          6.5                  100         0
99997          5.7                  155         0
99998          4.0                  100         0
99999          6.6                   90         0

[99982 rows x 9 columns]
```

Question 3