Problem statement

data description

1. Age: Age of the insured individual (Numerical)
2. Gender: Gender of the insured individual (Categorical: Male, Female)
3. Annual Income: Annual income of the insured individual (Numerical, skewed)
4. Marital Status: Marital status of the insured individual (Categorical: Single, Married, Divorced)
5. Number of Dependents: Number of dependents (Numerical, with missing values)
6. Education Level: Highest education level attained (Categorical: High School, Bachelor's, Master's, PhD)
7. Occupation: Occupation of the insured individual (Categorical: Employed, Self-Employed, Unemployed)
8. Health Score: A score representing the health status (Numerical, skewed)
9. Location: Type of location (Categorical: Urban, Suburban, Rural)
10. Policy Type: Type of insurance policy (Categorical: Basic, Comprehensive, Premium)
11. Previous Claims: Number of previous claims made (Numerical, with outliers)
12. Vehicle Age: Age of the vehicle insured (Numerical)
13. Credit Score: Credit score of the insured individual (Numerical, with missing values)
14. Insurance Duration: Duration of the insurance policy (Numerical, in years)
15. Premium Amount: Target variable representing the insurance premium amount (Numerical, skewed)
16. Policy Start Date: Start date of the insurance policy (Text, improperly formatted)
17. Customer Feedback: Short feedback comments from customers (Text)
18. Smoking Status: Smoking status of the insured individual (Categorical: Yes, No)
19. Exercise Frequency: Frequency of exercise (Categorical: Daily, Weekly, Monthly, Rarely)
20. Property Type: Type of property owned (Categorical: House, Apartment, Condo)

Quick overview of EDA results:

- As our data is synthetic, all the missing values comes under MCAR.
- 5% of population's annual income is lower than premium amount paid
- Insurance duration feature has inconsistency
- Annual income and premium amount feature are right skewed
- In Categorical features, all the categories are evenly distributed

Let's explore these results step by step with Exploratory Data Analysis

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
#additional config
pd.set_option('display.max_columns',None)
```

In [3]:

```
data = pd.read_csv('/kaggle/input/playground-series-s4e12/train.cs
v')
data.head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
```

Out[3]:

	id	Age	Gender	Annual Income	Marital Status	Number of Dependents	Education Level	Occupation	Health Score	Lo
0	0	19.0	Female	10049.0	Married	1.0	Bachelor's	Self-Employed	22.598761	Ur
1	1	39.0	Female	31678.0	Divorced	3.0	Master's	NaN	15.569731	Ru
2	2	23.0	Male	25602.0	Divorced	3.0	High School	Self-Employed	47.177549	Su
3	3	21.0	Male	141855.0	Married	2.0	Bachelor's	NaN	10.938144	Ru
4	4	21.0	Male	39651.0	Single	1.0	Bachelor's	Self-Employed	20.376094	Ru

In [4]:

```
data.shape
```

Out[4]:

```
(1200000, 21)
```

In [5]:

```
data.dtypes
```

Out[5]:

```
id                         int64
Age                      float64
Gender                    object
Annual Income            float64
Marital Status            object
Number of Dependents     float64
Education Level           object
Occupation                object
Health Score             float64
Location                  object
Policy Type               object
Previous Claims          float64
Vehicle Age              float64
Credit Score             float64
Insurance Duration       float64
Policy Start Date         object
Customer Feedback         object
Smoking Status            object
Exercise Frequency        object
Property Type             object
Premium Amount           float64
dtype: object
```

In [6]:

```
features = data.columns
numerical_features = data.select_dtypes(exclude='object').columns
categorical_features = data.select_dtypes(include='object').columns
print(f'numerical features: {len(numerical_features)} \ncategorical features: {len(categorical_features)}')
```

```
numerical features: 10
categorical features: 11
```

checking missing values

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
id                         0
Age                    18705
Gender                     0
Annual Income          44949
Marital Status         18529
Number of Dependents  109672
Education Level            0
Occupation            358075
Health Score           74076
Location                   0
Policy Type                0
Previous Claims       364029
Vehicle Age                6
Credit Score          137882
Insurance Duration         1
Policy Start Date          0
Customer Feedback      77824
Smoking Status             0
Exercise Frequency         0
Property Type              0
Premium Amount             0
dtype: int64
```

In [8]:

```
data['id'].nunique()
```

Out[8]:

```
1200000
```

Assumption 1: as our id is unique, we assume that no two claims are overlap by an individual (i.e), we have 12 lakh indivial people who are insured in this data.

In [9]:

```
na_features = []
for col in features:
    if data[col].isnull().any():
        na_features.append(col)
        print(f'{col:<22} has {np.round(data[col].isnull().mean() *
100,2)} % of NA')
print(f'\nNo of NA features: {len(na_features)}')
print(na_features)
```

```
Age                    has 1.56 % of NA
Annual Income          has 3.75 % of NA
Marital Status         has 1.54 % of NA
Number of Dependents   has 9.14 % of NA
Occupation             has 29.84 % of NA
Health Score           has 6.17 % of NA
Previous Claims        has 30.34 % of NA
Vehicle Age            has 0.0 % of NA
Credit Score           has 11.49 % of NA
Insurance Duration     has 0.0 % of NA
Customer Feedback      has 6.49 % of NA

No of NA features: 11
['Age', 'Annual Income', 'Marital Status', 'Number of Dependents', '
Occupation', 'Health Score', 'Previous Claims', 'Vehicle Age', 'Cred
it Score', 'Insurance Duration', 'Customer Feedback']
```

Result: 11 features have missing values

Exploratory data analysis

Analysis for feature with missing values

Age feature analysis

In [10]:

```
data['Age'].nunique()
```

Out[10]:

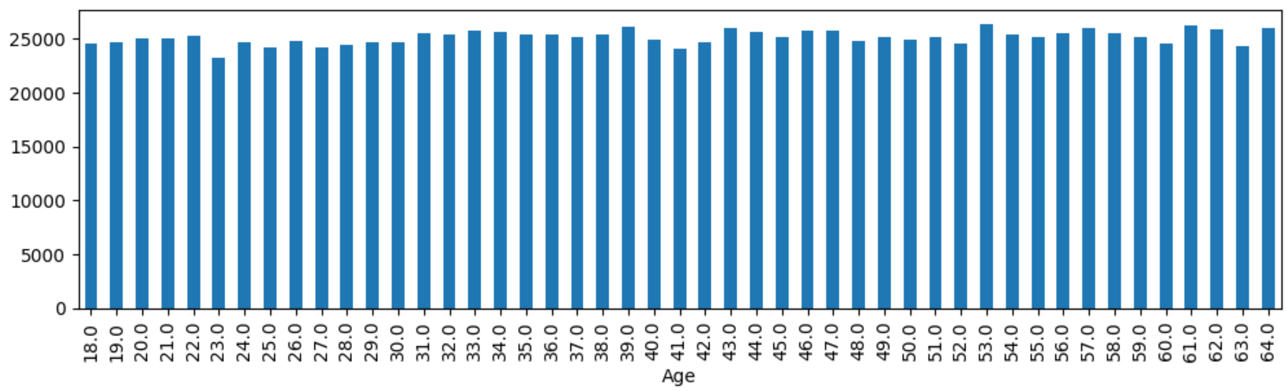
```
47
```

In [11]:

```
data['Age'].value_counts().sort_index().plot(kind='bar',figsize=(12,
3))
```

Out[11]:

```
<Axes: xlabel='Age'>
```



In [12]:

```
data[data['Age'].isna()]['Gender'].value_counts()
```

Out[12]:

```
Gender
Male      9381
Female    9324
Name: count, dtype: int64
```

Result: Age missing is not dependend on Gender

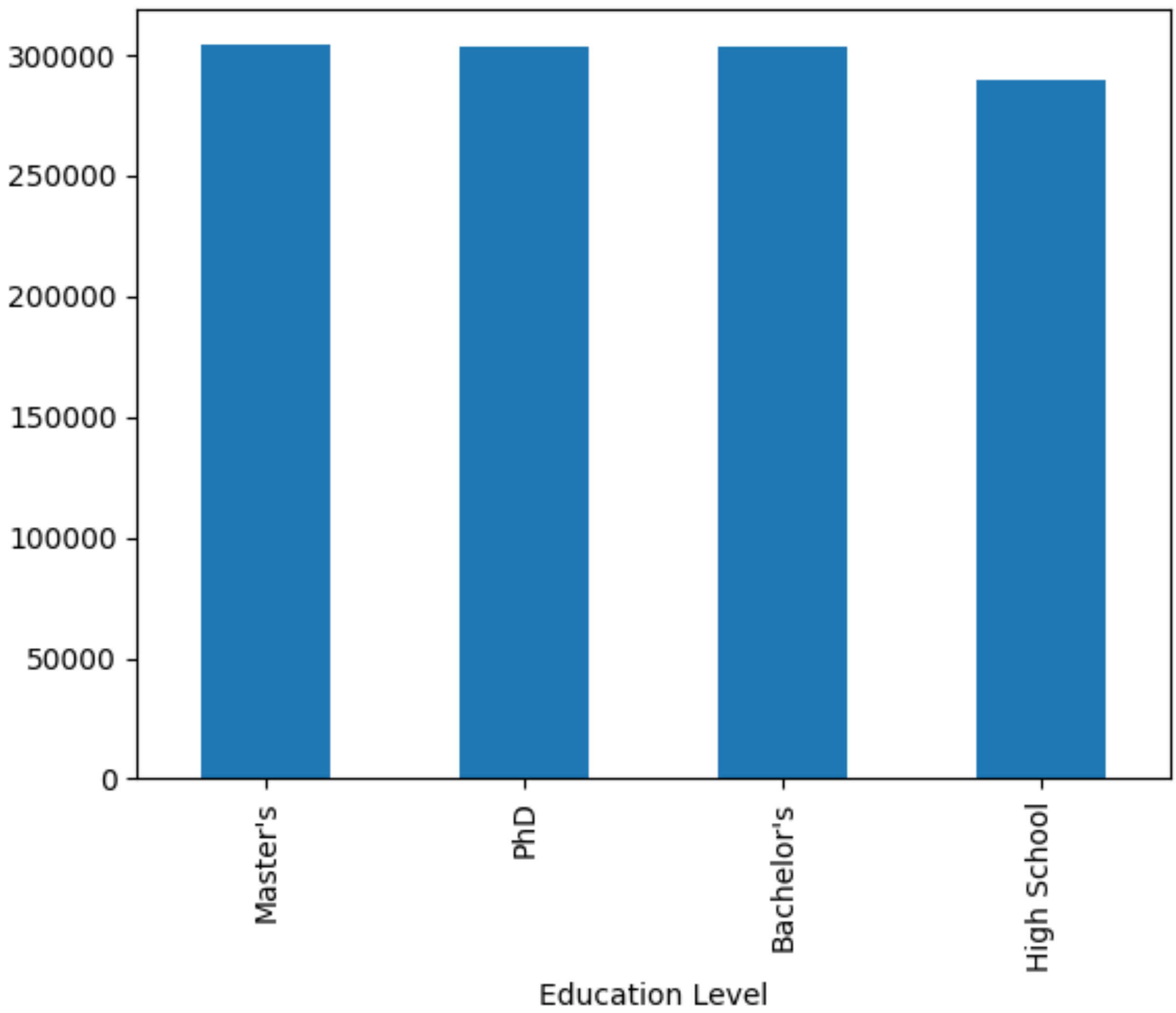
Analysing Occupation feature

In [13]:

```
data['Education Level'].value_counts().plot.bar()
```

Out[13]:

```
<Axes: xlabel='Education Level'>
```

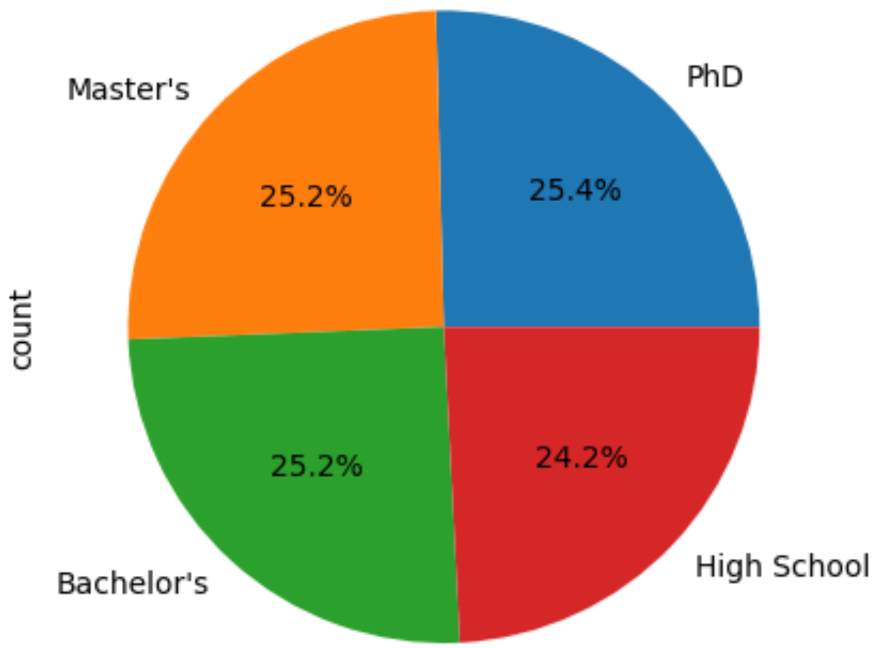

300000
250000
200000
150000
100000
50000
0
Master's
PhD
Bachelor's
High School
Education Level

In [14]:

```
#checking whether educational level has any relation
data[data['Occupation'].isna()]['Education Level'].value_counts().pl
ot(kind='pie',autopct='%1.1f%%')
```

Out[14]:

```
<Axes: ylabel='count'>
```



Result: Missing values of occupation is almost equally distributed among education level. **type**: MCAR

Analysing annual income feature

In [15]:

```
data['Occupation'].value_counts()
```

Out[15]:

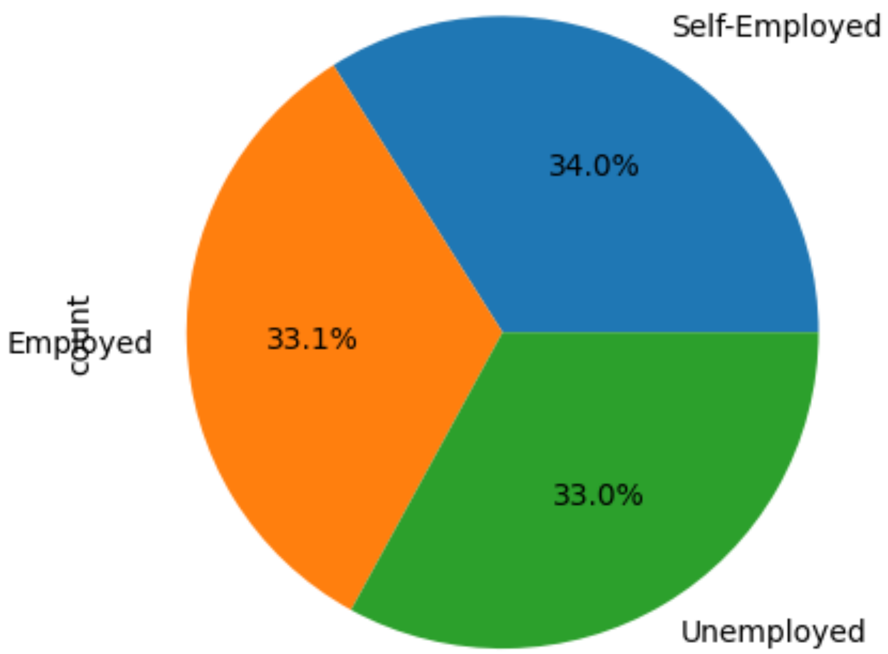
```
Occupation
Employed         282750
Self-Employed    282645
Unemployed       276530
Name: count, dtype: int64
```

In [16]:

```
data[data['Annual Income'].isna()]['Occupation'].value_counts().plo
t(kind='pie',autopct="%1.1f%%")
```

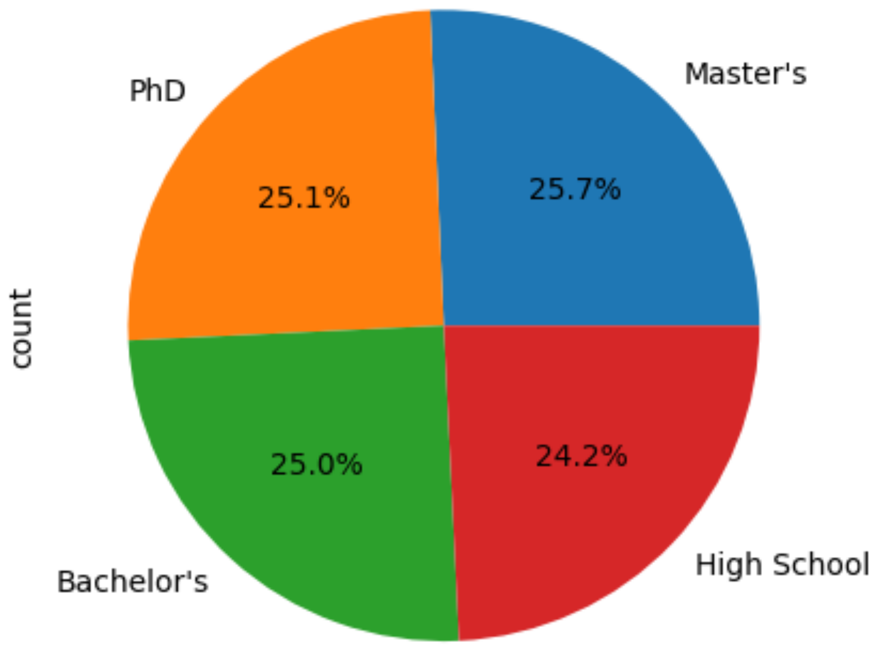
Out[16]:

```
<Axes: ylabel='count'>
```



In [17]:
```
data[data['Annual Income'].isna()]['Education Level'].value_count
s().plot(kind='pie',autopct="%1.1f%%")
```

Out[17]:
```
<Axes: ylabel='count'>
```



Result: Missing values in annual income is equally distrubed among education level and occupation

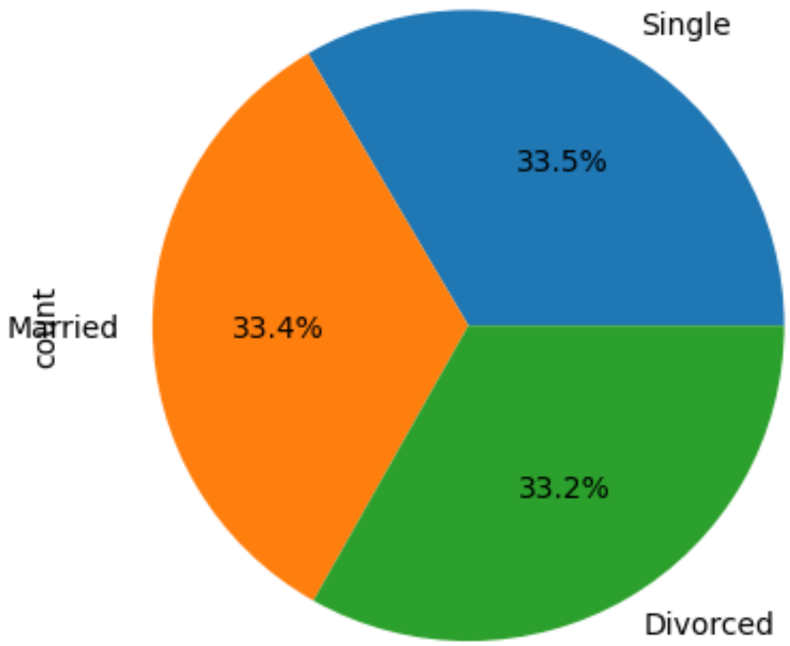
Analyzing maritial status

In [18]:

```
data['Marital Status'].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

Out[18]:

```
<Axes: ylabel='count'>
```

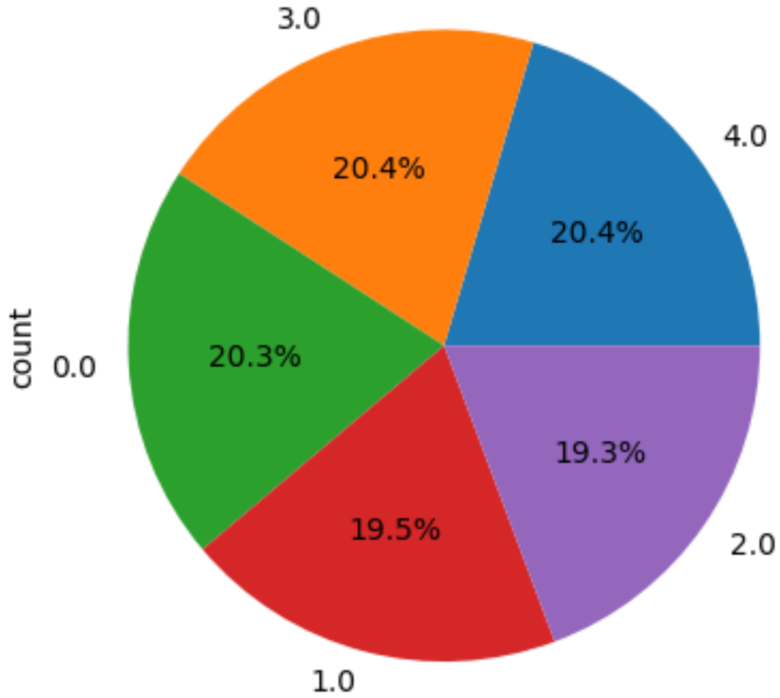


In [19]:

```
data[data['Marital Status'].isna()]['Number of Dependents'].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

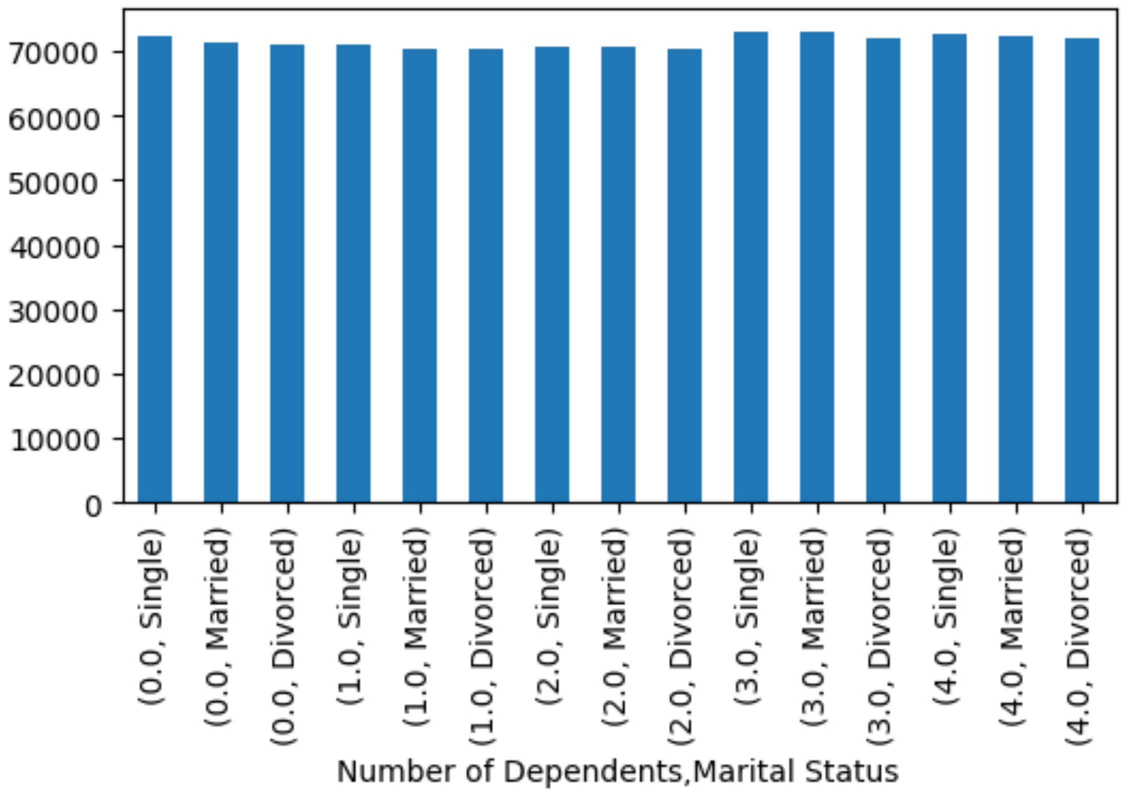
Out[19]:

```
<Axes: ylabel='count'>
```



In [20]:
```
data.groupby('Number of Dependents')['Marital Status'].value_counts().plot.bar(figsize=(6,3))
```

Out[20]:
```
<Axes: xlabel='Number of Dependents,Marital Status'>
```



Result:

- No of dependents has no relationship with marital status
- No of dependents column is float type i.e need to change to int

Analyzing health score feature

In [21]:
```
data.columns
```

Out[21]:
```
Index(['id', 'Age', 'Gender', 'Annual Income', 'Marital Status',
       'Number of Dependents', 'Education Level', 'Occupation', 'Hea
lth Score',
       'Location', 'Policy Type', 'Previous Claims', 'Vehicle Age',
       'Credit Score', 'Insurance Duration', 'Policy Start Date',
       'Customer Feedback', 'Smoking Status', 'Exercise Frequency',
       'Property Type', 'Premium Amount'],
      dtype='object')
```

In [22]:
```
data['Smoking Status'].value_counts()
```

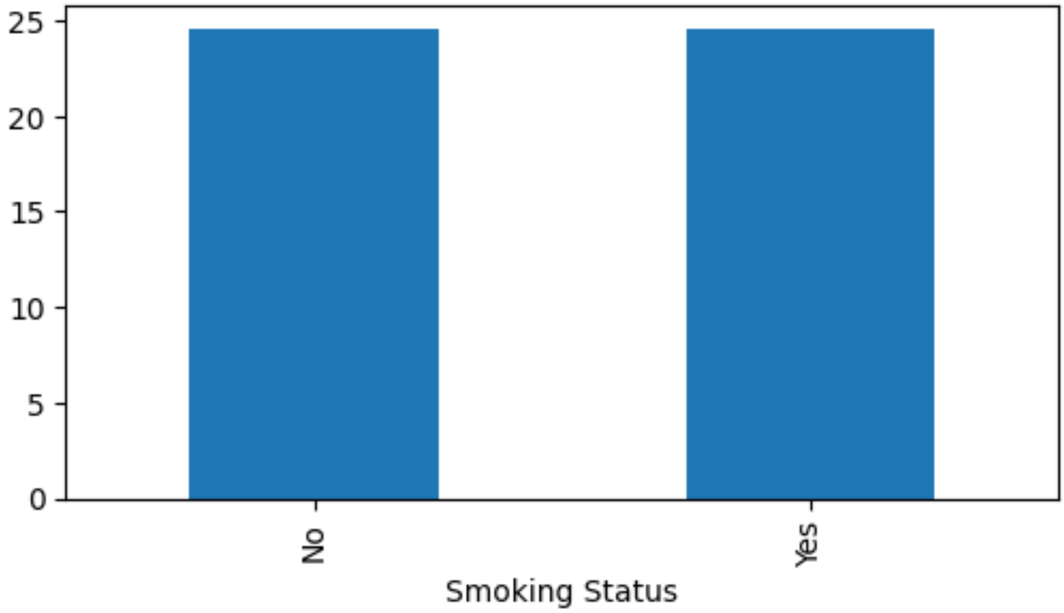
Out[22]:
```
Smoking Status
Yes    601873
No     598127
Name: count, dtype: int64
```

In [23]:

```
data.groupby('Smoking Status')['Health Score'].median().plot.bar(fig
size=(6,3))
```

Out[23]:

```
<Axes: xlabel='Smoking Status'>
```



In [24]:

```
#Analysing health score by excercise habit
data['Exercise Frequency'].value_counts()
data[data['Health Score'].isna()]['Exercise Frequency'].value_count
s()
```

Out[24]:

```
Exercise Frequency
Weekly     18715
Daily      18680
Rarely     18521
Monthly    18160
Name: count, dtype: int64
```

In [25]:
```
data.groupby('Exercise Frequency')['Health Score'].mean()
```

Out[25]:
```
Exercise Frequency
Daily      25.650123
Monthly    25.575288
Rarely     25.629956
Weekly     25.601310
Name: Health Score, dtype: float64
```

Result: In this dataset, smoking status has no affect on health score. Also, Excercising doesn't have any health improvements on avg

Analysing for insurance duration

In [26]:
```
data[data['Insurance Duration'].isna()]
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
```

Out[26]:

	id	Age	Gender	Annual Income	Marital Status	Number of Dependents	Education Level	Occupation	Health Score
711358	711358	64.0	Male	30206.0	Married	3.0	Master's	Employed	49.551(

In [27]:

```
#checking insurance duration for similar policy start data
data[data['Policy Start Date'].str.contains('2022-04-06')].head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1458: RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/format.py:
1459: RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals
> 0)).any()
```

Out[27]:

	id	Age	Gender	Annual Income	Marital Status	Number of Dependents	Education Level	Occupation	Health Score
2613	2613	36.0	Female	76683.0	Single	2.0	High School	NaN	5.594031
3018	3018	31.0	Male	38323.0	Single	2.0	PhD	Employed	29.99333
3277	3277	19.0	Female	38962.0	Single	4.0	Master's	NaN	13.84928
5172	5172	27.0	Male	NaN	Single	4.0	PhD	Self-Employed	29.63938
5645	5645	18.0	Male	107964.0	Divorced	3.0	PhD	NaN	26.76617

Result: For same policy start date, there are different insurance duration (inconsistency).

Further analysis

In [28]:

```
data[data['Premium Amount'] > data['Annual Income']].shape
```

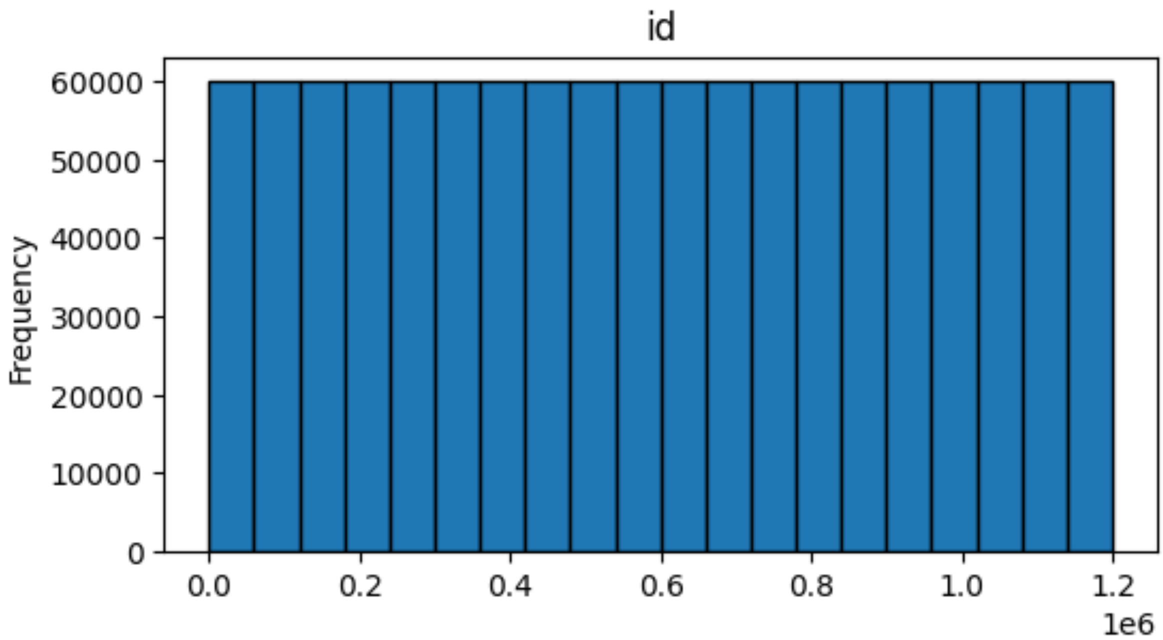
Out[28]:

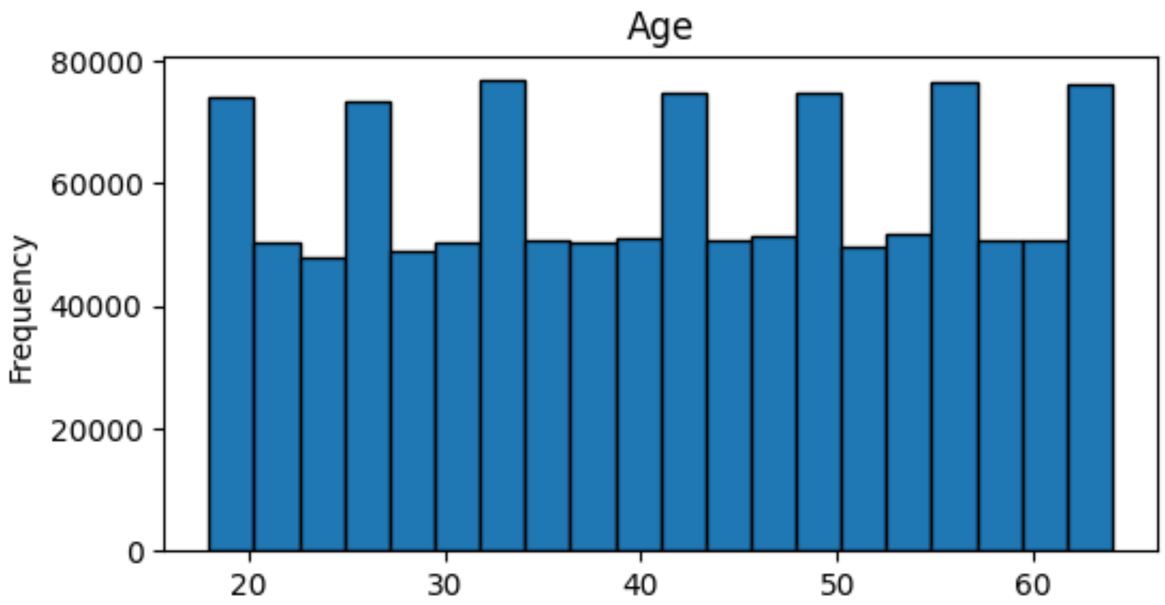
```
(51717, 21)
```

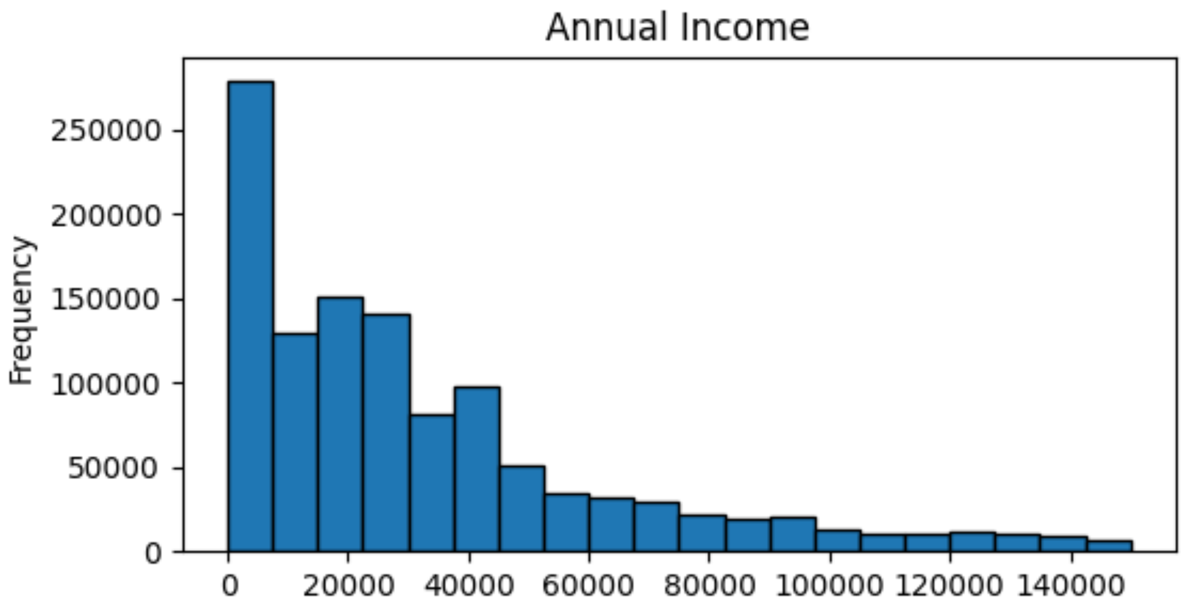
result: around 4% people's annual income is lower than premium amount they paid

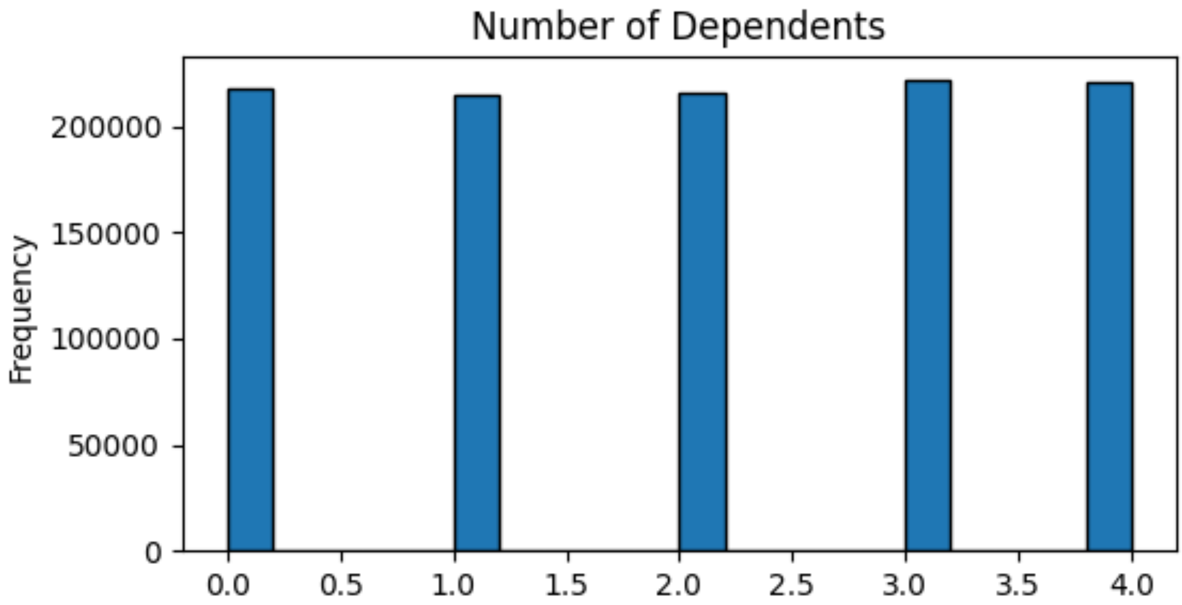
Distribution of numerical features

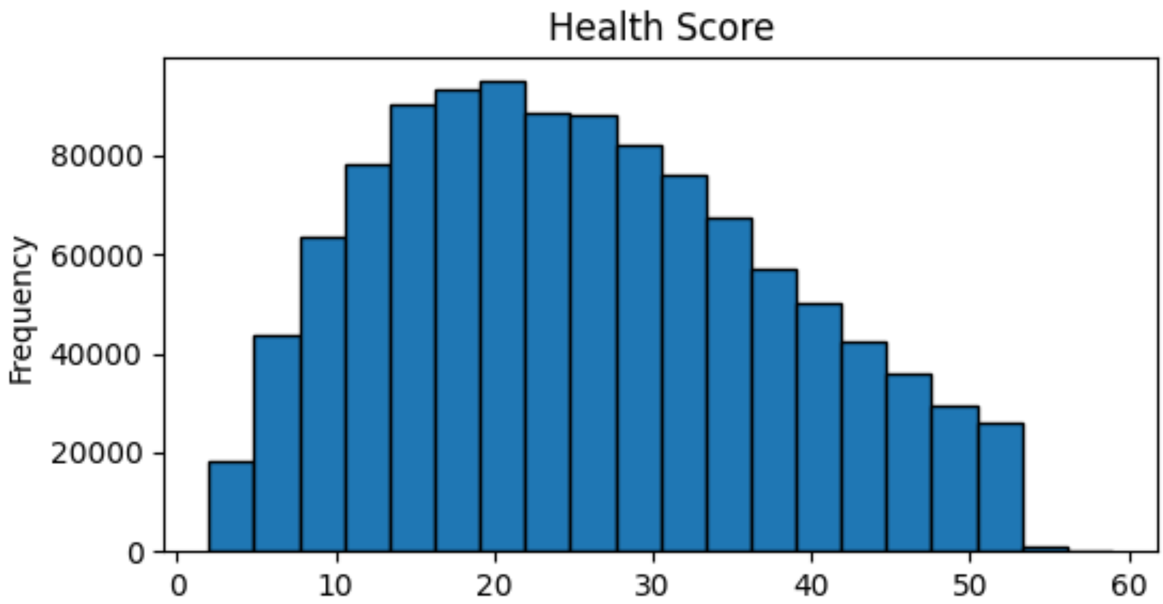
In [29]:
```
for col in numerical_features:
    data[col].plot(kind='hist',bins=20,figsize=(6,3),edgecolor='black')
    plt.title(col)
    plt.show()
```

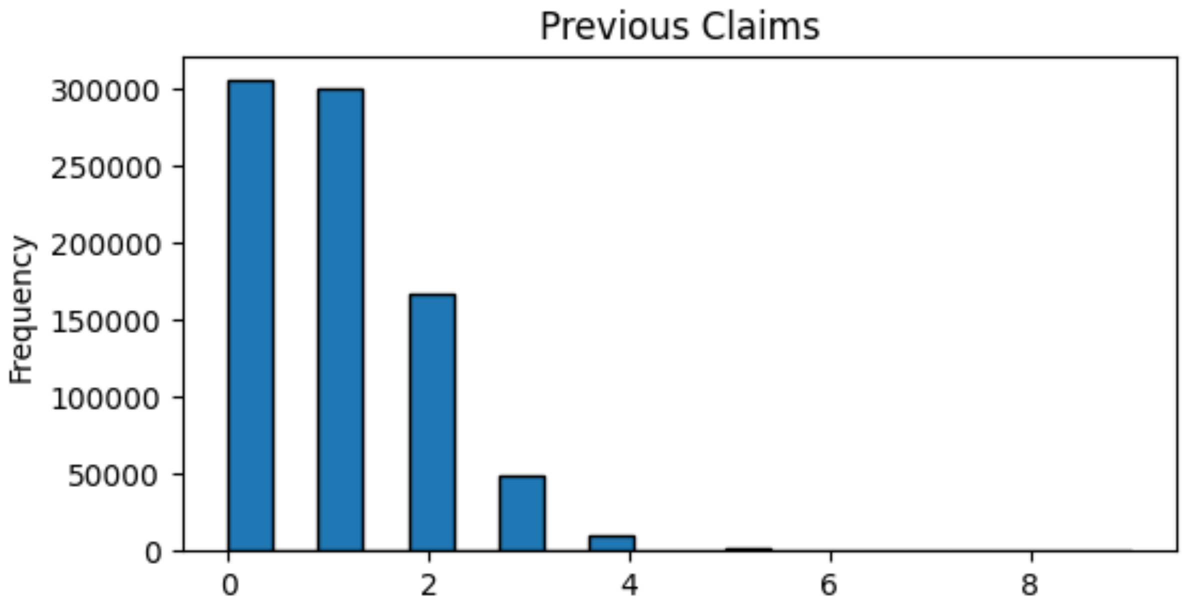


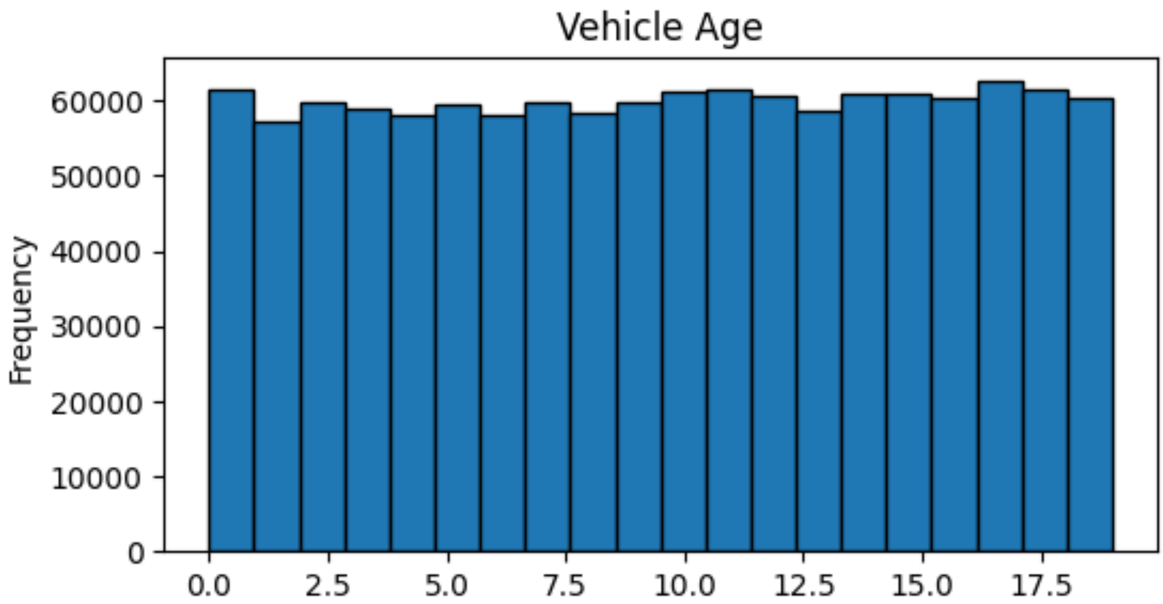

Age
Frequency
80000
60000
40000
20000
0
20
30
40
50
60

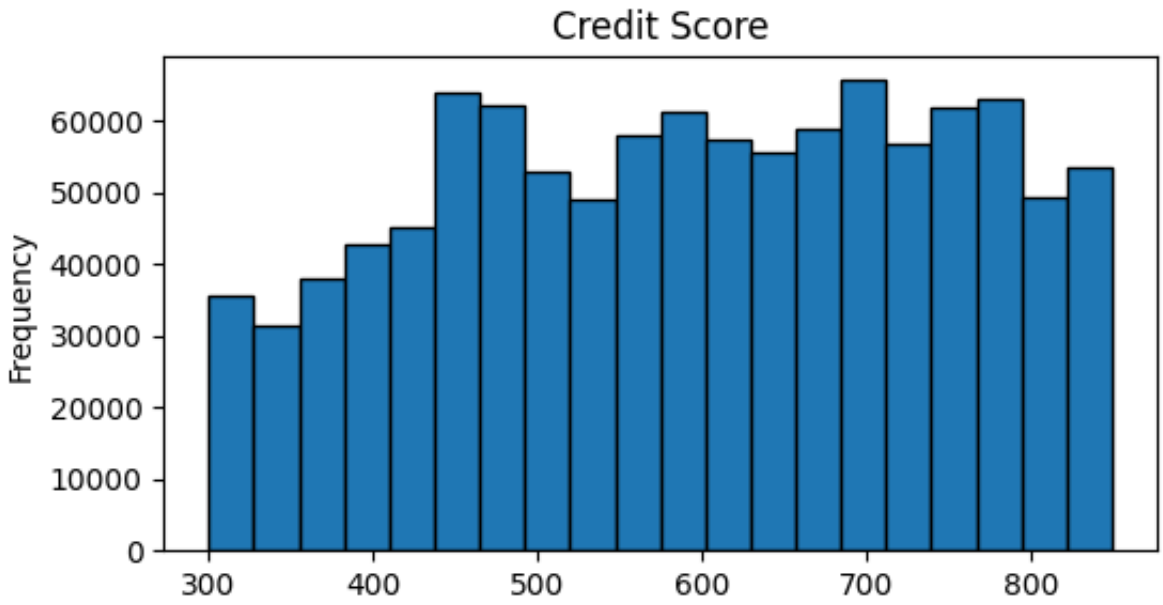

Annual Income
Frequency
250000
200000
150000
100000
50000
0
0
20000
40000
60000
80000
100000
120000
140000

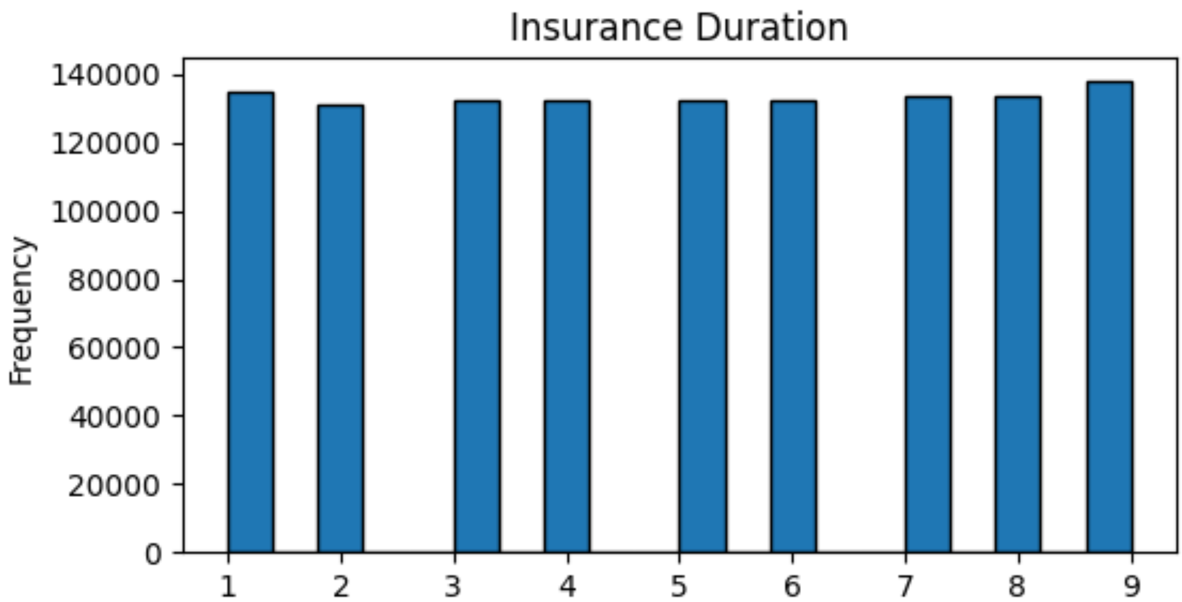

Number of Dependents
Frequency
0
50000
100000
150000
200000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

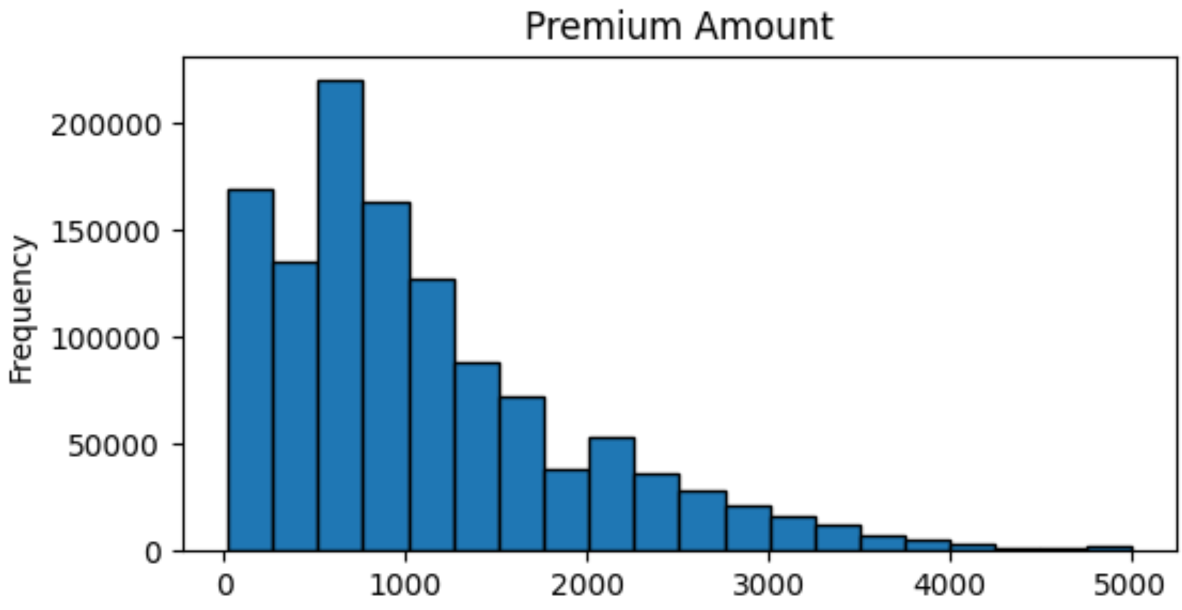

Health Score
Frequency
0
20000
40000
60000
80000
0
10
20
30
40
50
60


Previous Claims
Frequency
300000
250000
200000
150000
100000
50000
0
0
2
4
6
8


Vehicle Age
Frequency
60000
50000
40000
30000
20000
10000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Credit Score
Frequency
0
10000
20000
30000
40000
50000
60000
300
400
500
600
700
800


Insurance Duration
Frequency
0
20000
40000
60000
80000
100000
120000
140000
1
2
3
4
5
6
7
8
9

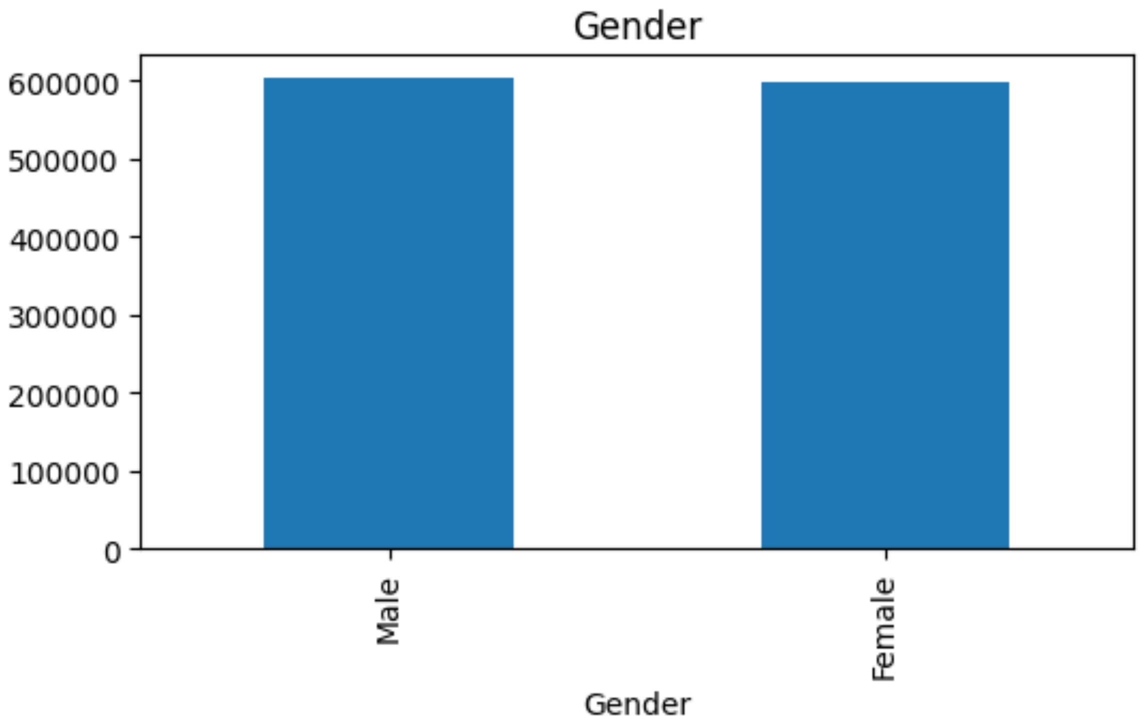


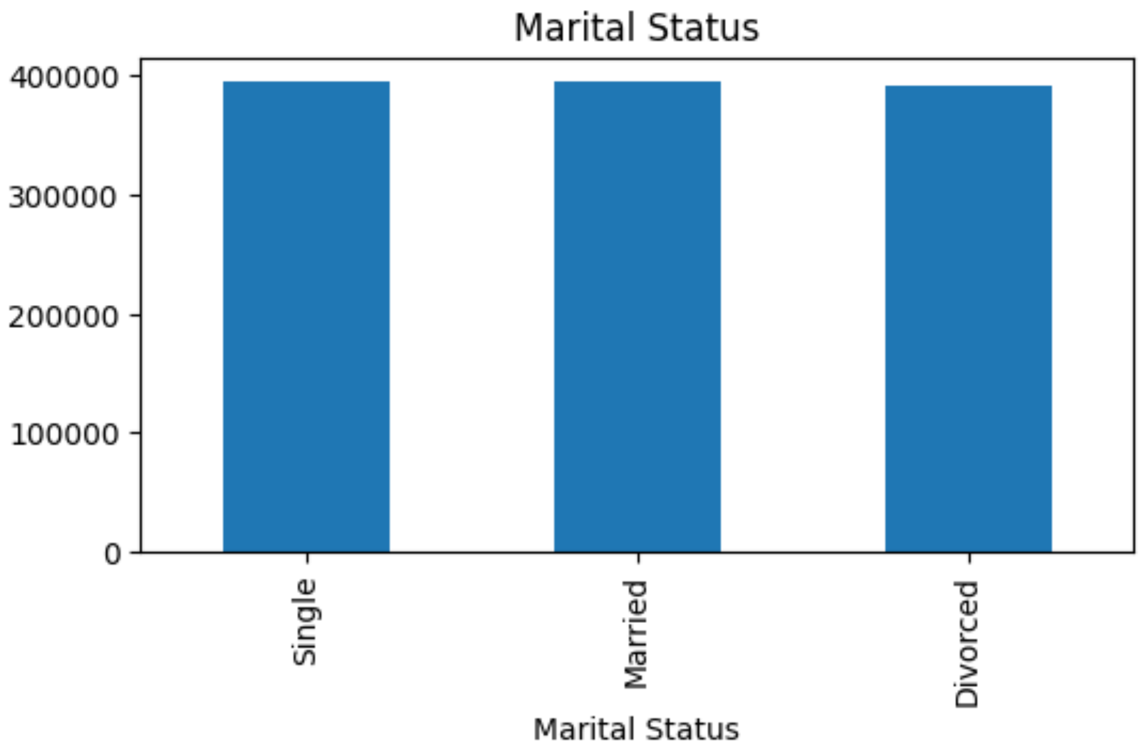
Insights:

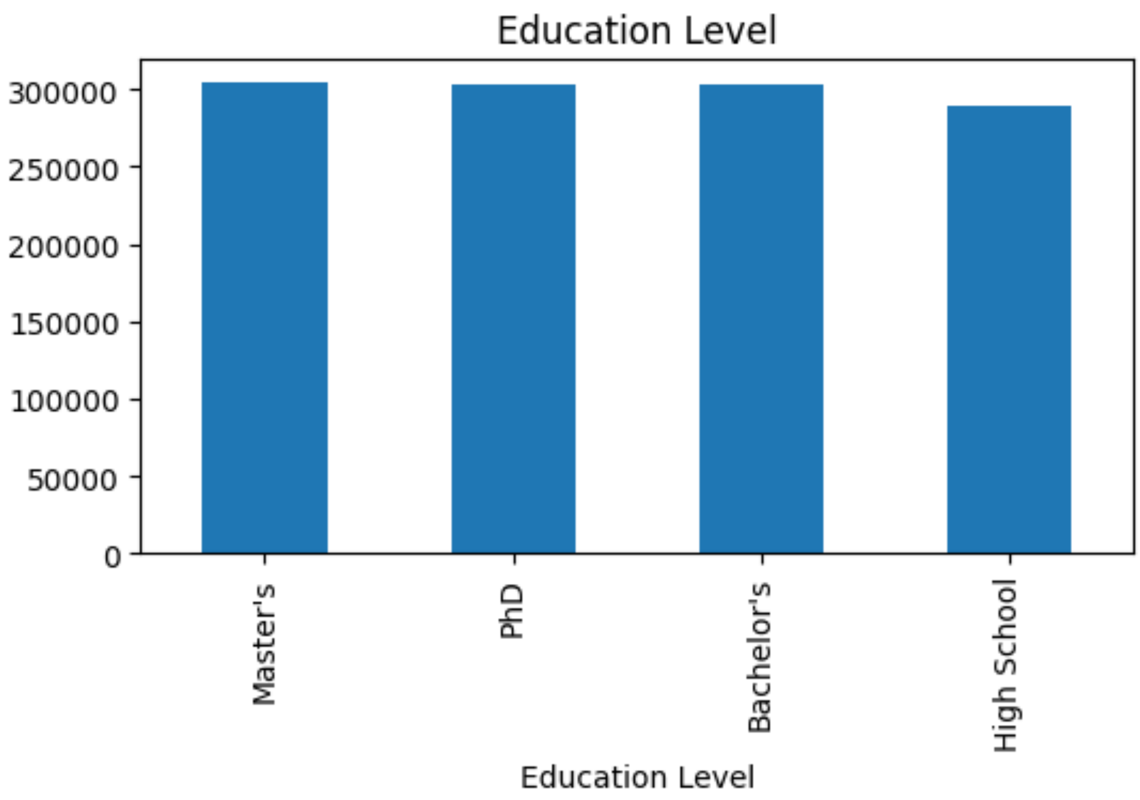
- Annual income and premium amount are right skewed

Distribution of categorical features:

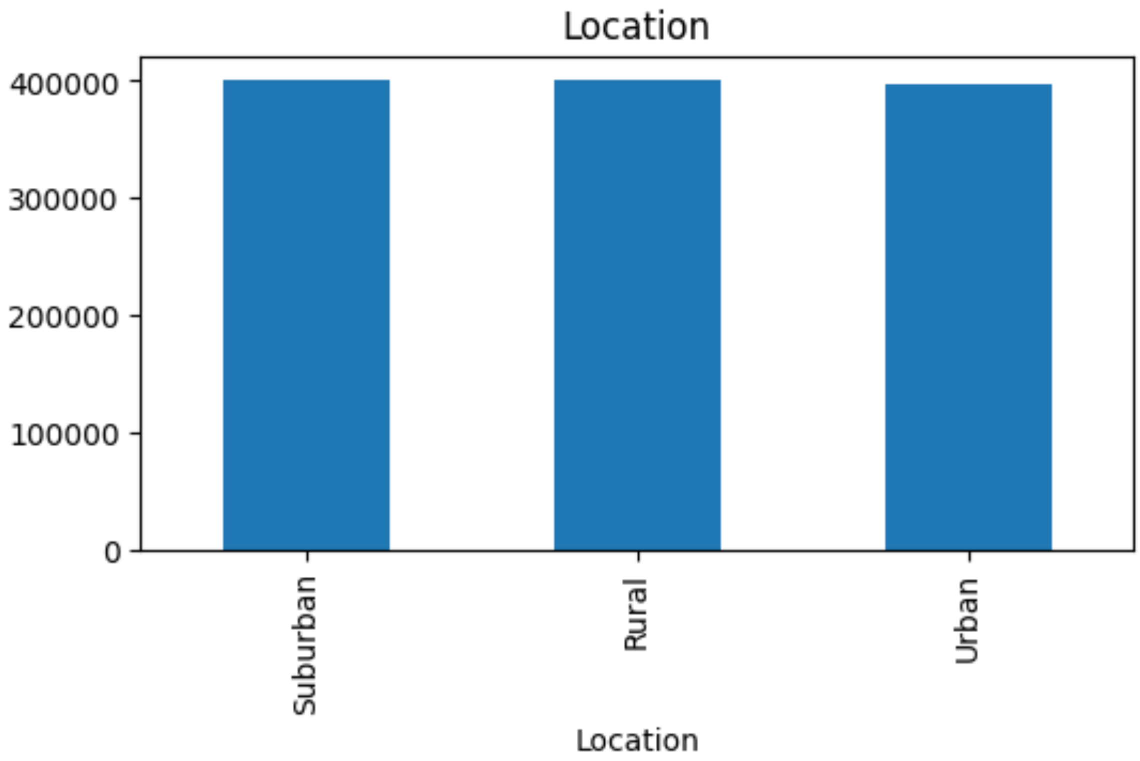
In [30]:
```
for col in categorical_features:
    if col != 'Policy Start Date':
        data[col].value_counts().plot(kind='bar',figsize=(6,3))
        plt.title(col)
        plt.show()
```

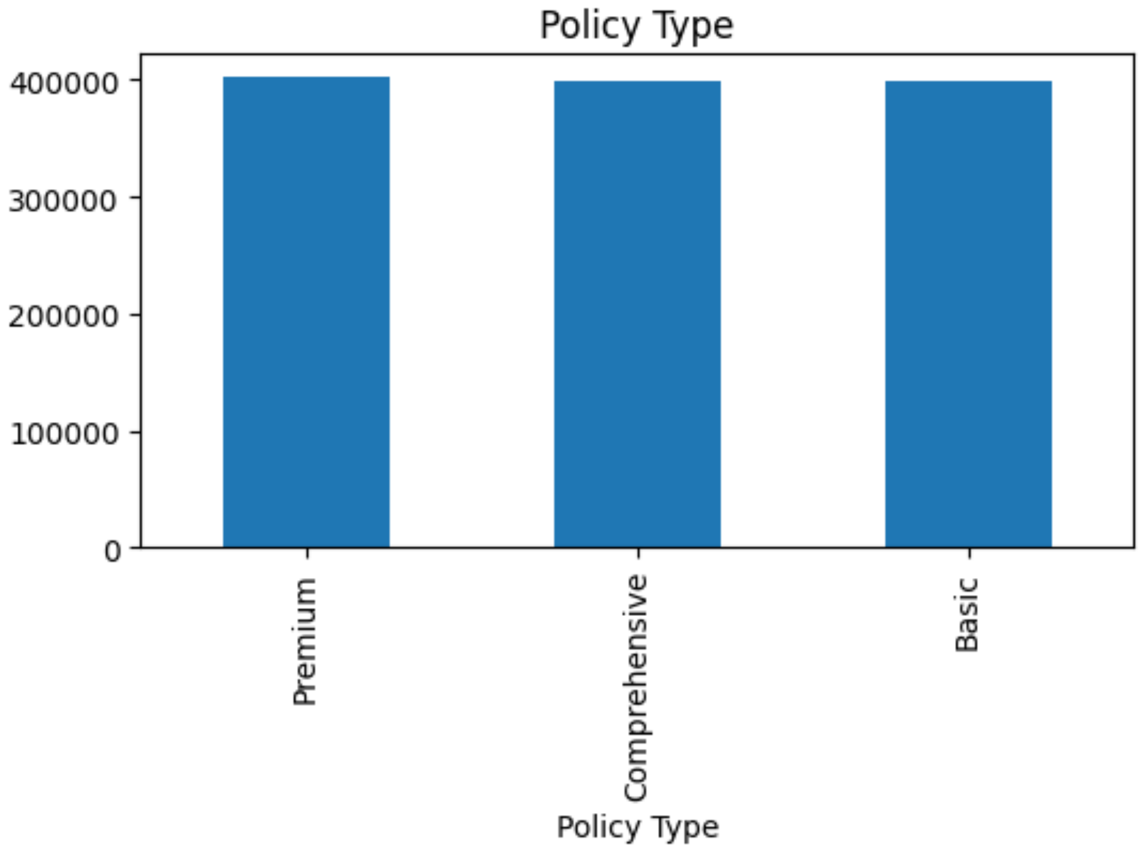

Gender
600000
500000
400000
300000
200000
100000
0
Male
Female
Gender


Marital Status
400000
300000
200000
100000
0
Single
Married
Divorced
Marital Status

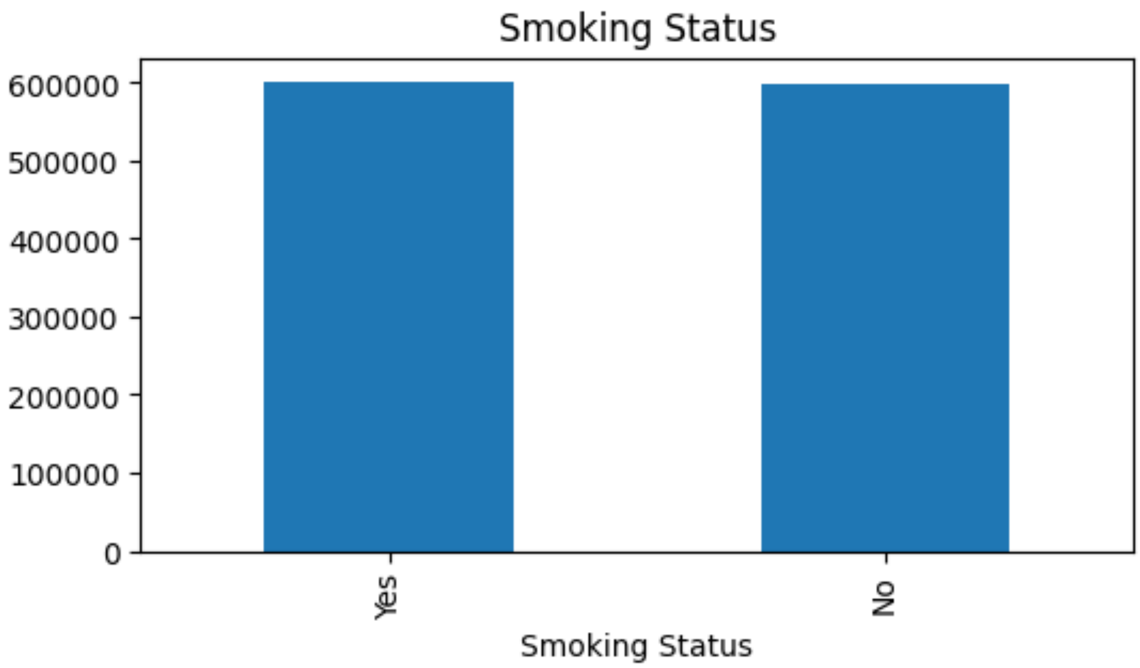

Education Level
300000
250000
200000
150000
100000
50000
0
Master's
PhD
Bachelor's
High School
Education Level

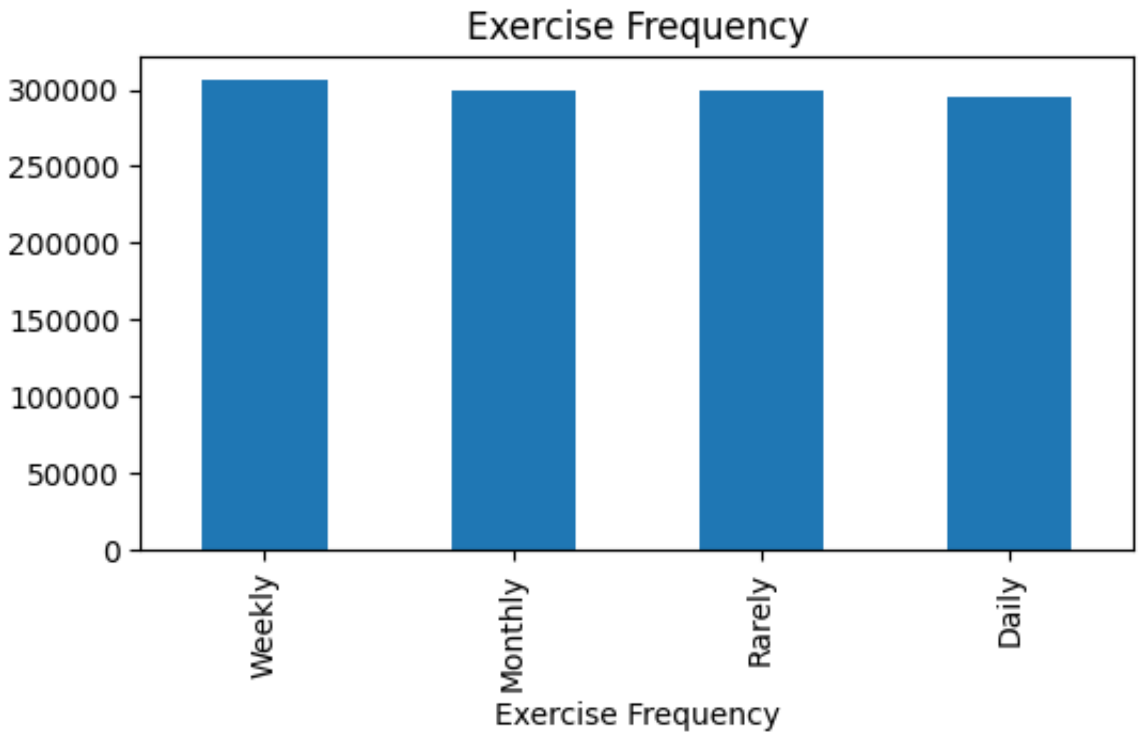

Occupation
250000
200000
150000
100000
50000
0
Employed
Self-Employed
Unemployed
Occupation

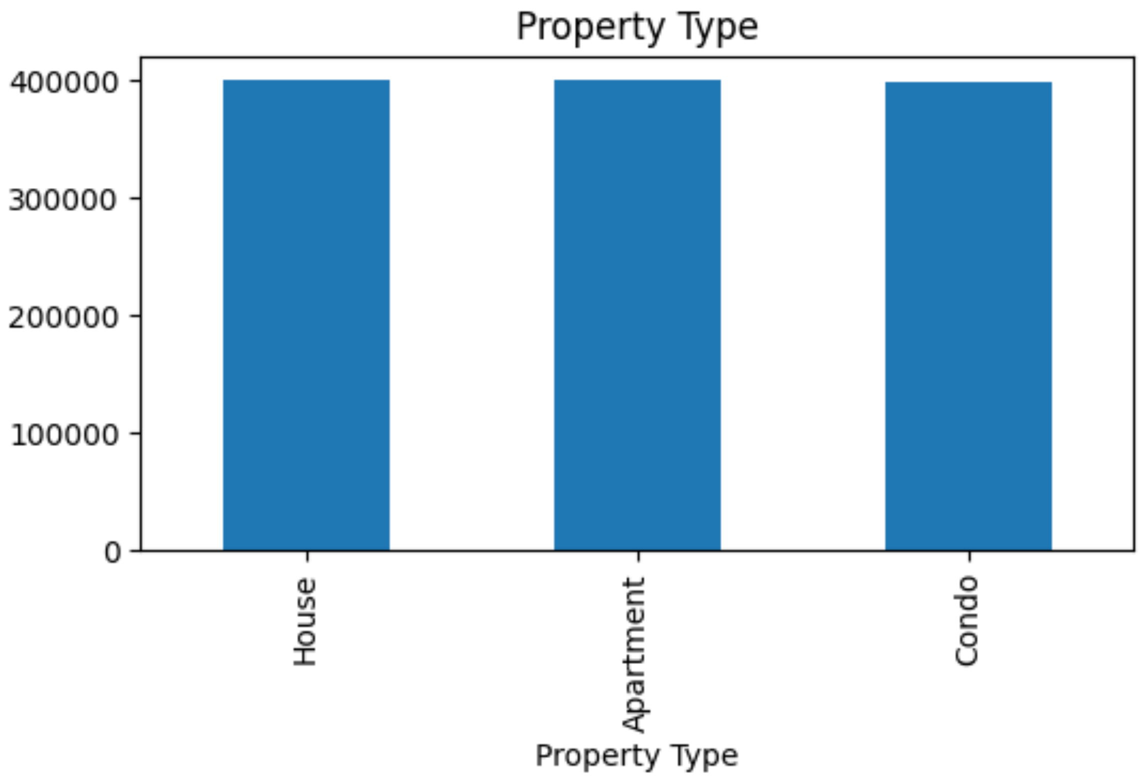

Location
400000
300000
200000
100000
0
Suburban
Rural
Urban
Location


Policy Type
400000
300000
200000
100000
0
Premium
Comprehensive
Basic
Policy Type


Customer Feedback
350000
300000
250000
200000
150000
100000
50000
0
Average
Poor
Good
Customer Feedback


Smoking Status
600000
500000
400000
300000
200000
100000
0
Yes
No
Smoking Status


Exercise Frequency
300000
250000
200000
150000
100000
50000
0
Weekly
Monthly
Rarely
Daily
Exercise Frequency


Property Type
400000
300000
200000
100000
0
House
Apartment
Condo
Property Type

No Insights

Outliers Detection Analysis

Starting with numerical features

In [31]:

```
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [32]:

```
for col in numerical_features:
    if data[col].skew() > 0.5 or data[col].skew() < -0.5:
        print(f'{col} {np.round(data[col].skew(),2)}')
```

```
Annual Income 1.47
Previous Claims 0.91
Premium Amount 1.24
```

So, features with outliers:

- Annual Income
- Previous claims
- Premium amount

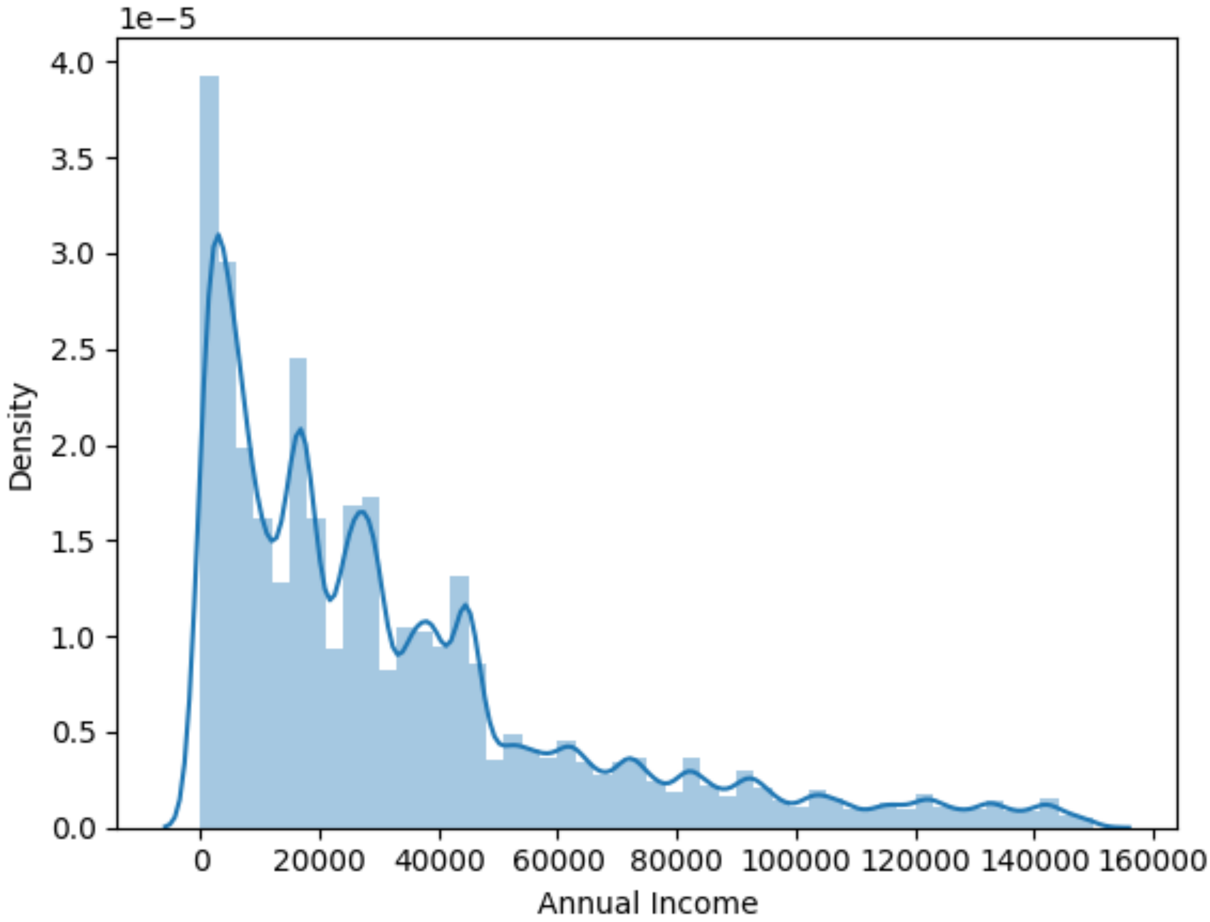
Analysing Annual income:

In [33]:

```
sns.distplot(data['Annual Income'])
```

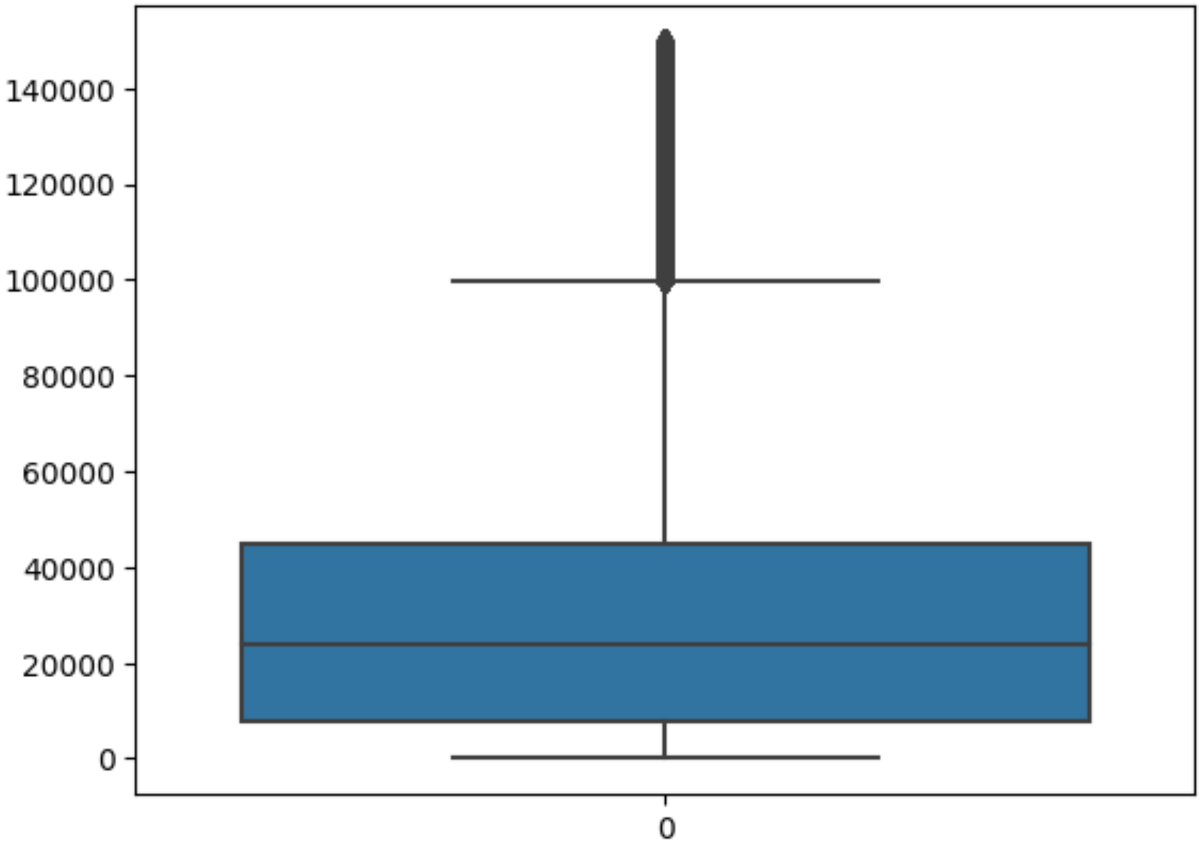
Out[33]:

```
<Axes: xlabel='Annual Income', ylabel='Density'>
```



In [34]:
```
sns.boxplot(data['Annual Income'])
```

Out[34]:
```
<Axes: >
```



Lets go with IQR to find extreme outliers and treat them

In [35]:
```
IQR = data['Annual Income'].quantile(0.75) - data['Annual Income'].q
uantile(0.25)
lower_bound = (data['Annual Income'].quantile(0.25)) - (IQR * 1.5)
upper_bound = (data['Annual Income'].quantile(0.75)) + (IQR * 1.5)
print(lower_bound,upper_bound)
```

```
-46948.5 99583.5
```

In [36]:

```
data[data['Annual Income'] > 99583].shape
```

Out[36]:

```
(67132, 21)
```

In [37]:

```
#for extreme outliers
lower_bound = (data['Annual Income'].quantile(0.25)) - (IQR * 3)
upper_bound = (data['Annual Income'].quantile(0.75)) + (IQR * 3)
print(lower_bound,upper_bound)
```

```
-101898.0 154533.0
```

In [38]:

```
data[data['Annual Income'] > 99583].shape
```

Out[38]:

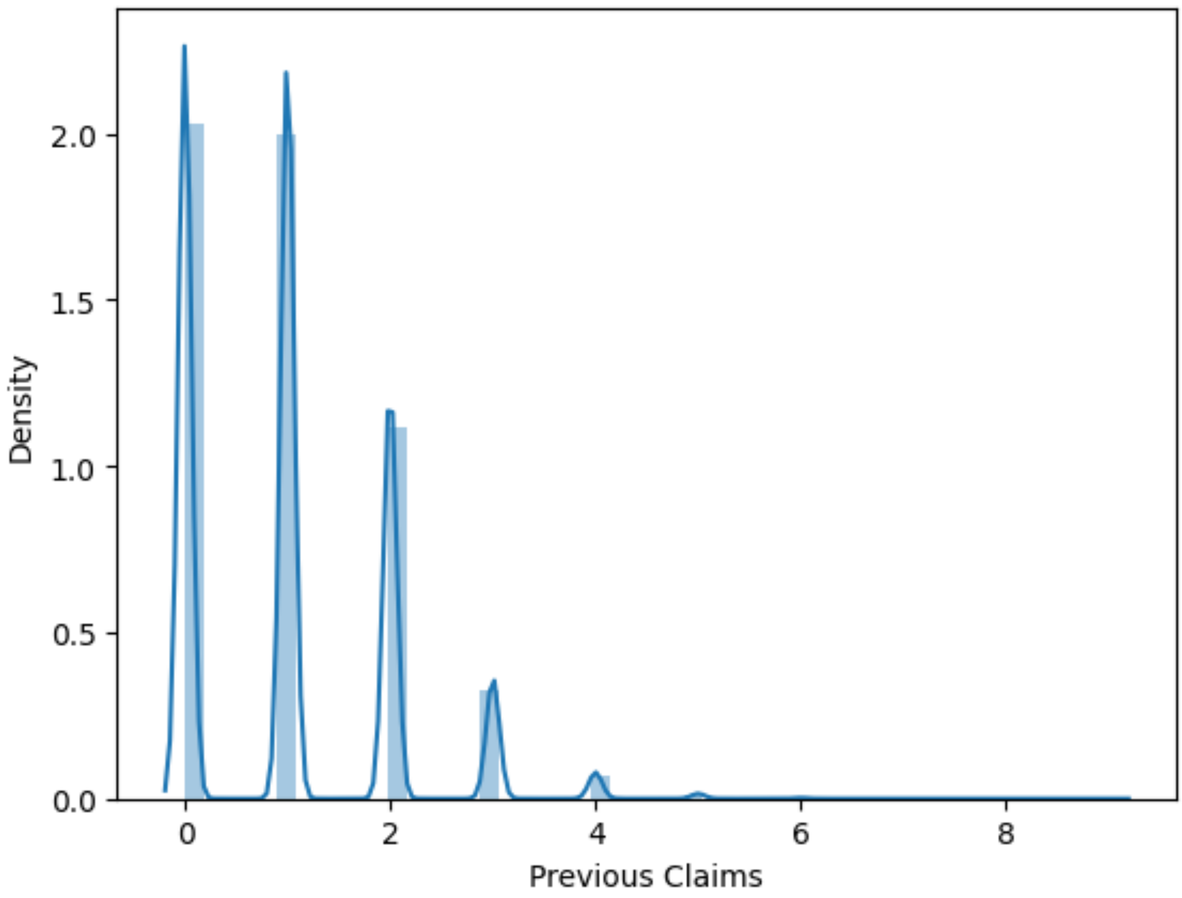
```
(67132, 21)
```

Result: Annual Income has 67132 outliers

Analying previous claims

In [39]:
```
sns.distplot(data['Previous Claims'])
```

Out[39]:
```
<Axes: xlabel='Previous Claims', ylabel='Density'>
```

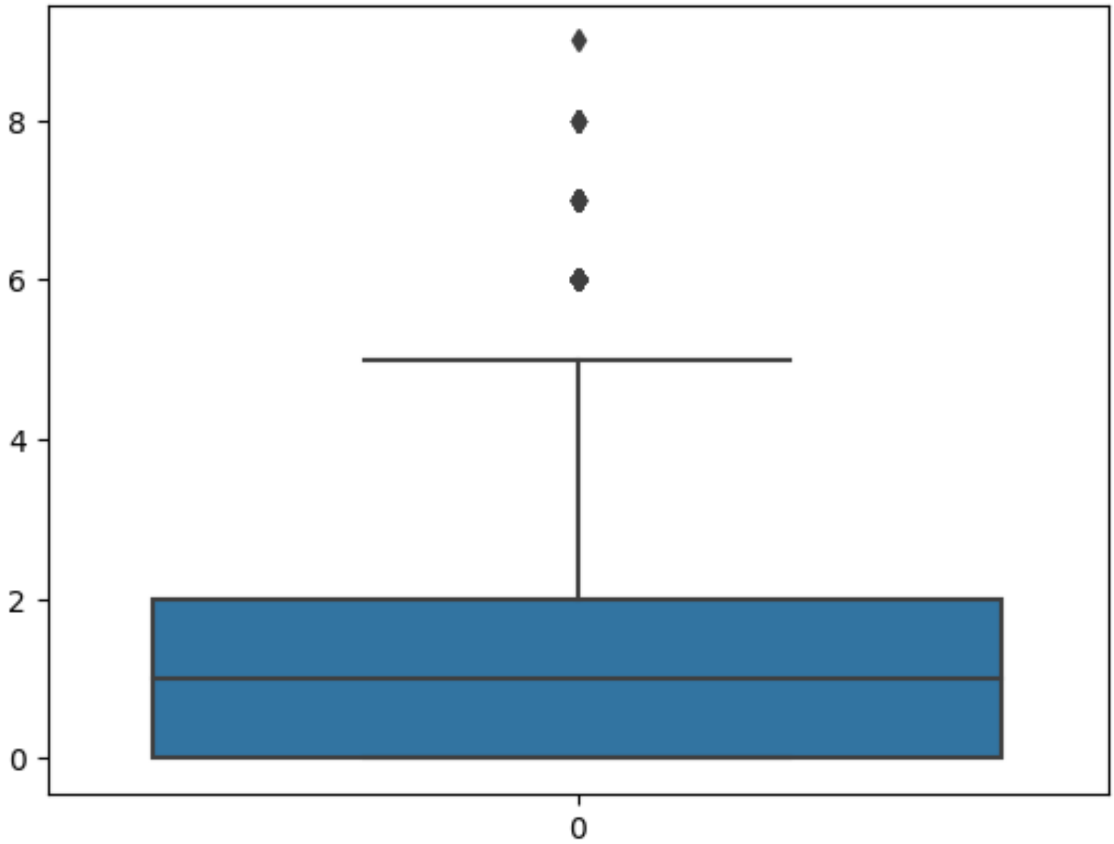


In [40]:

```
sns.boxplot(data['Previous Claims'])
```

Out[40]:

```
<Axes: >
```



In [41]:

```
IQR = data['Previous Claims'].quantile(0.75) - data['Previous Claims'].quantile(0.25)
lower_bound = (data['Previous Claims'].quantile(0.25)) - (IQR * 1.5)
upper_bound = (data['Previous Claims'].quantile(0.75)) + (IQR * 1.5)
print(lower_bound,upper_bound)
```

```
-3.0 5.0
```

In [42]:

```
data[data['Previous Claims'] > 5].shape
```

Out[42]:

```
(369, 21)
```

Result: Previous claims has 369 outliers

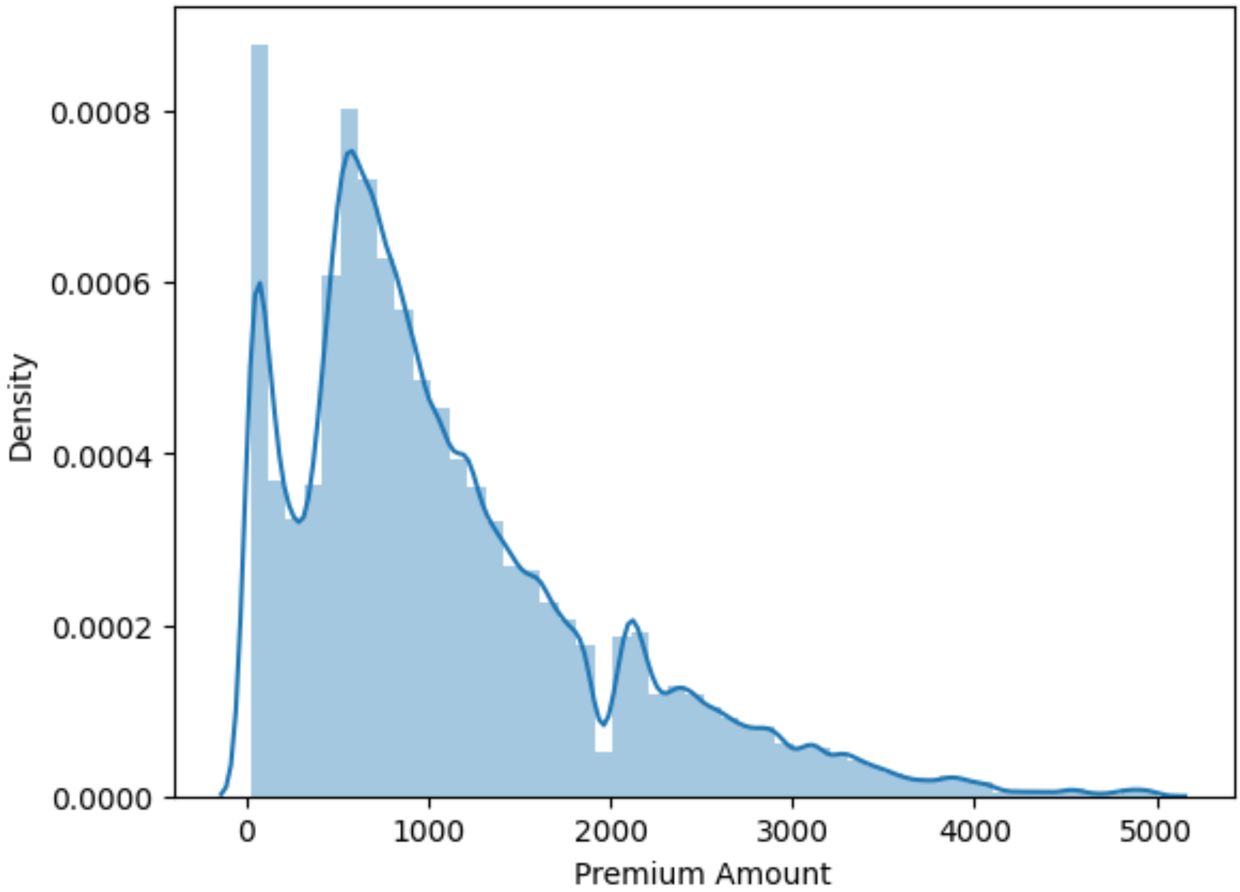
Analyzing Premium amount

In [43]:

```
sns.distplot(data['Premium Amount'])
```

Out[43]:

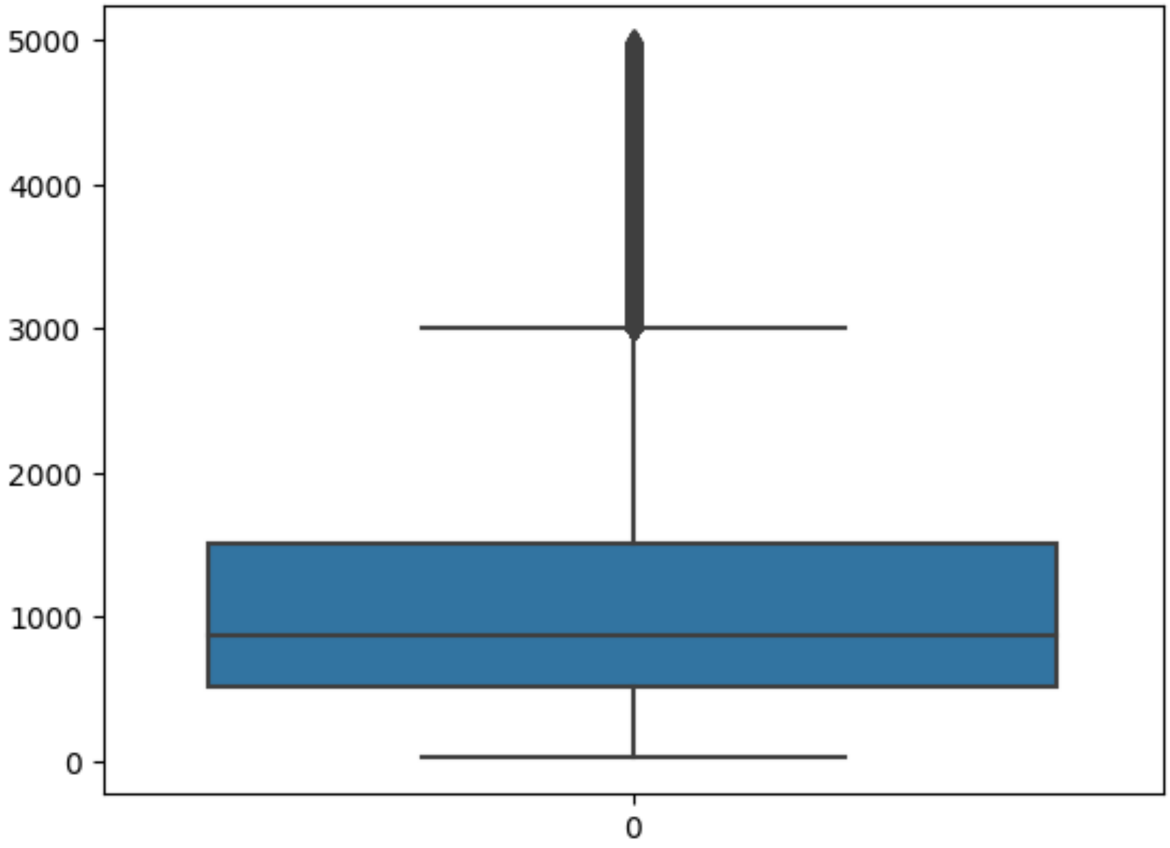
```
<Axes: xlabel='Premium Amount', ylabel='Density'>
```



In [44]:
```
sns.boxplot(data['Premium Amount'])
```

Out[44]:

<Axes: >



In [45]:
```
IQR = data['Premium Amount'].quantile(0.75) - data['Premium Amoun
t'].quantile(0.25)
lower_bound = (data['Premium Amount'].quantile(0.25)) - (IQR * 1.5)
upper_bound = (data['Premium Amount'].quantile(0.75)) + (IQR * 1.5)
print(lower_bound,upper_bound)
```

```
-978.5 3001.5
```

In [46]:
```
data[data['Premium Amount'] > 3001.5].shape
```

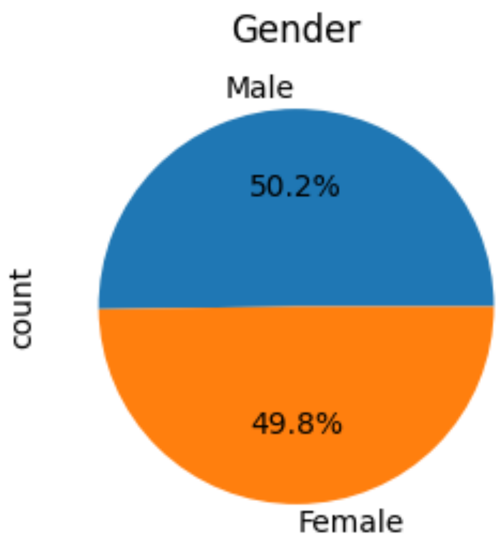
Out[46]:
```
(49320, 21)
```

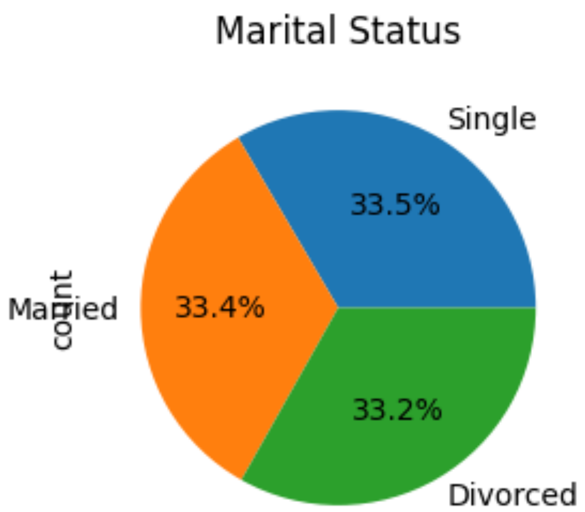
Result: Premium amount has 49320 outliers

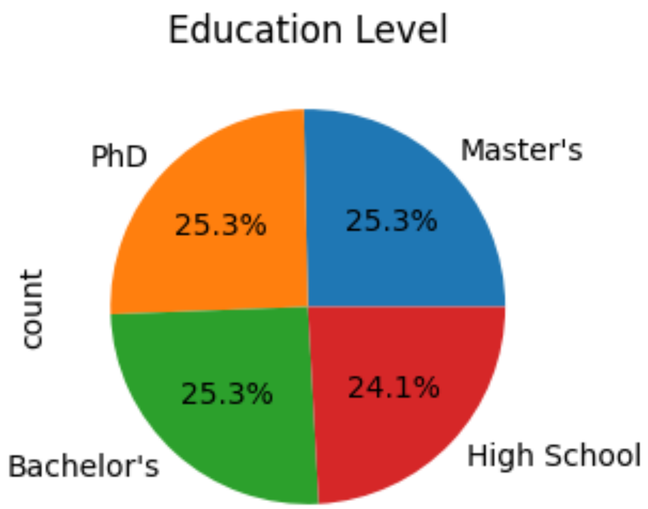
for Categorical features

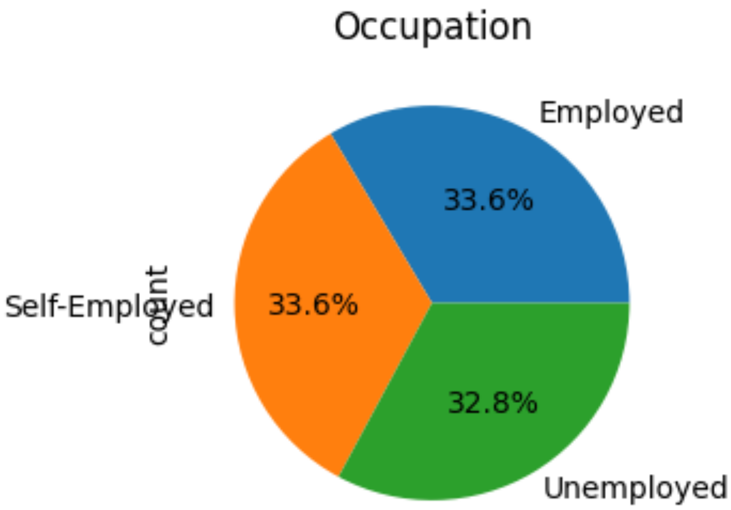
we check for rare categories and mark it as others

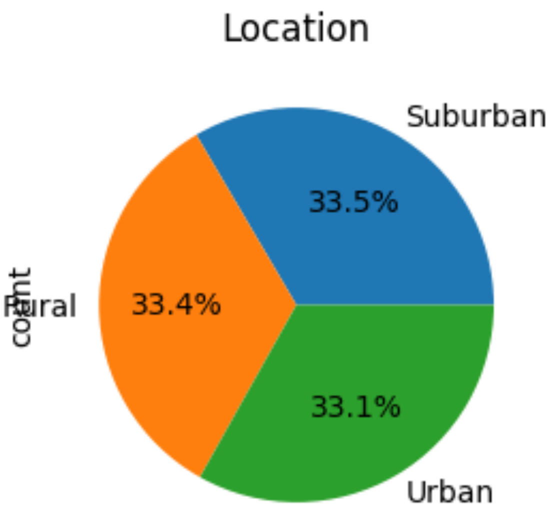
In [47]:
```
for col in categorical_features:
    if col != 'Policy Start Date':
        data[col].value_counts().plot(kind='pie',autopct="%1.1f%%",
figsize=(3,3))
        plt.title(col)
        plt.show()
```

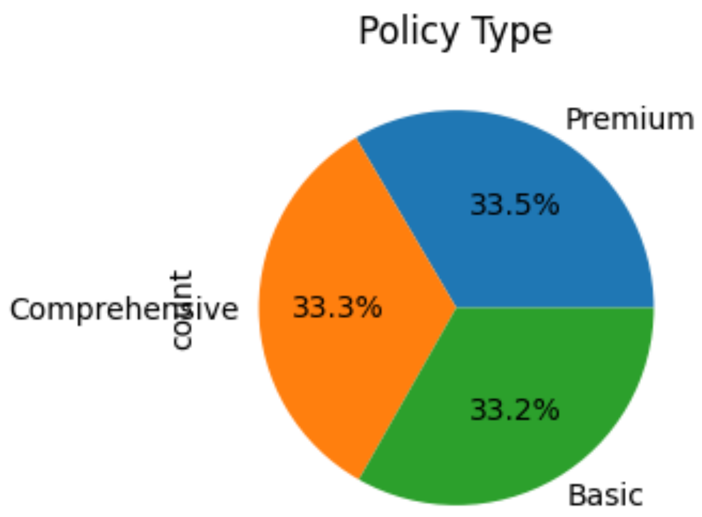


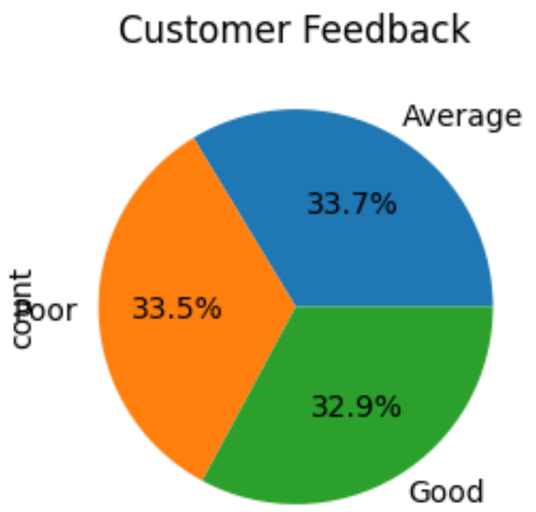

Marital Status
Single
33.5%
Married
33.4%
count
33.2%
Divorced

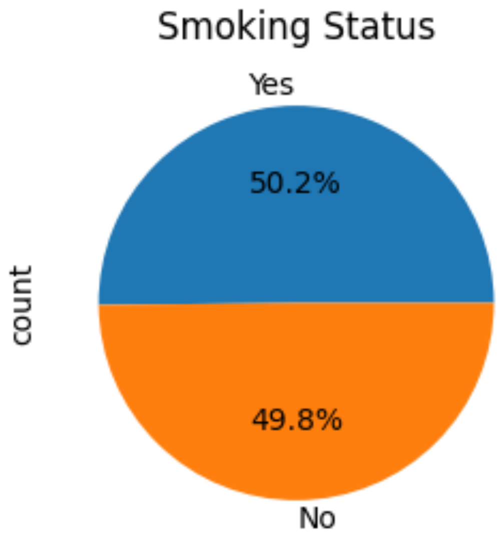

Education Level
PhD
Master's
25.3%
25.3%
count
25.3%
24.1%
Bachelor's
High School

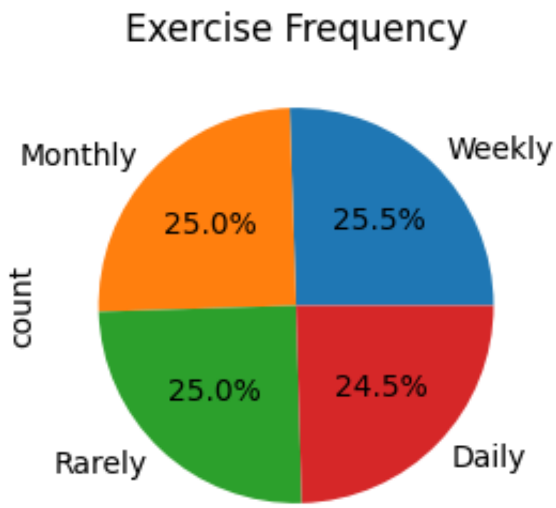

Occupation
Employed
33.6%
Self-Employed
count
33.6%
32.8%
Unemployed

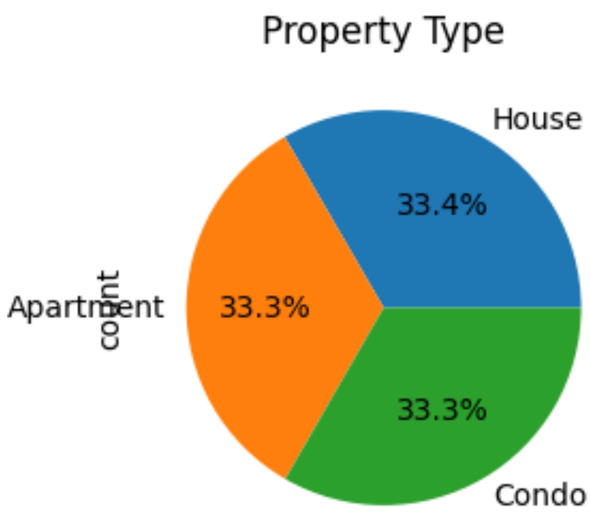

Location
Suburban
33.5%
Rural
count
33.4%
33.1%
Urban


Policy Type
Premium
33.5%
Comprehensive
count
33.3%
33.2%
Basic


Customer Feedback
Average
33.7%
count
Poor
33.5%
32.9%
Good


Smoking Status
Yes
50.2%
count
49.8%
No


Exercise Frequency
Monthly
Weekly
25.0%
25.5%
count
25.0%
24.5%
Rarely
Daily



Result: In our categorical features, all the categories are evenly distributed. So let's leave it