In [1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
df = pd.read_json("D:\Power BI\Hyderabad Assigment\data\loan_approval_dataset.json")
df.head(10)
```

Out[2]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	
1	2	7574516	40	10	single	rented	no	Software_Developer	
2	3	3991815	66	4	married	rented	no	Technical_writer	
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bh
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchi
5	6	6915937	64	0	single	rented	no	Civil_servant	
6	7	3954973	58	14	married	rented	no	Librarian	
7	8	1706172	33	2	single	rented	no	Economist	
8	9	7566849	24	17	single	rented	yes	Flight_attendant	
9	10	8964846	23	12	single	rented	no	Architect	k

In [56]:
```python
df.shape
```

Out[56]:
```
(252000, 13)
```

In [3]:
```python
df.isnull().sum()
```

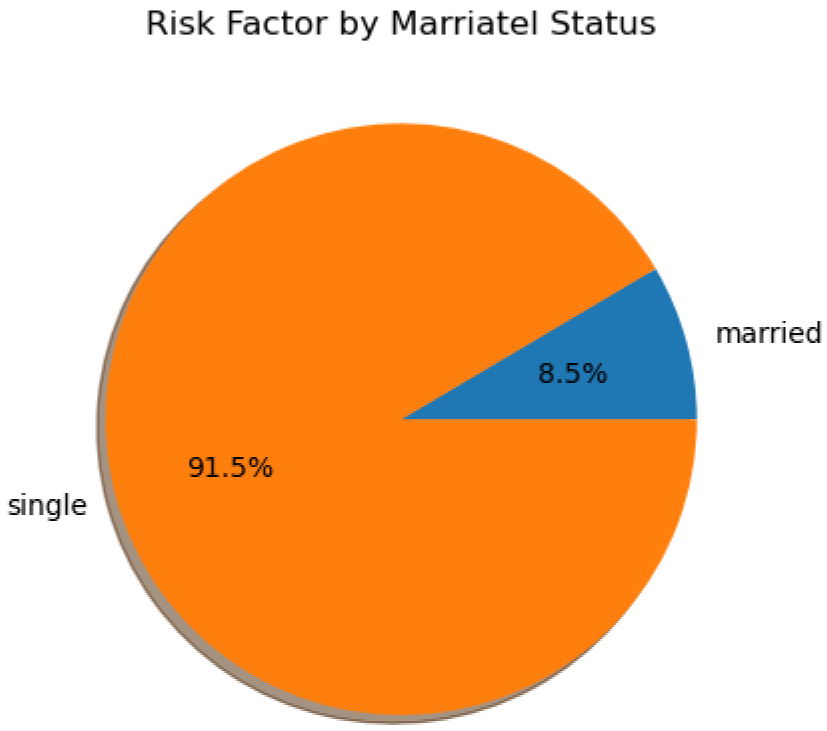
Out[3]:
```
Id                   0
Income               0
Age                  0
Experience           0
Married/Single       0
House_Ownership      0
Car_Ownership        0
Profession           0
CITY                 0
STATE                0
CURRENT_JOB_YRS      0
CURRENT_HOUSE_YRS    0
Risk_Flag            0
dtype: int64
```

In [4]:
```python
df.duplicated().sum()
```

Out[4]:
```
0
```

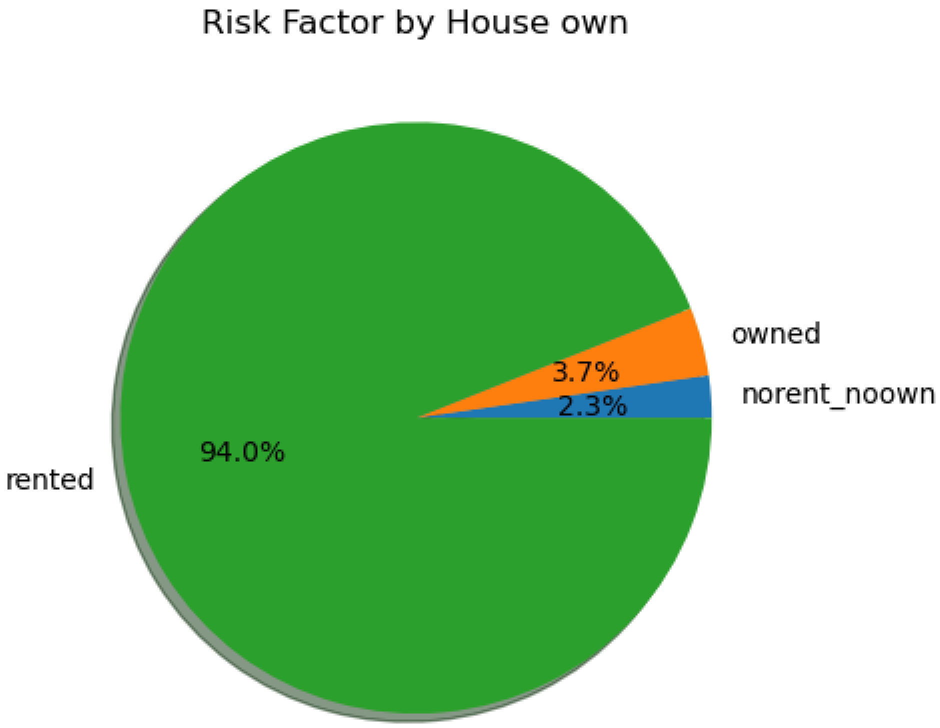
In [52]:
```python
Married = df.groupby(['Married/Single'])['Risk_Flag'].sum()

Married.plot(kind='pie',autopct='%1.1f%%',title='Risk Factor by Marriatel Status',ylabel
plt.show()
```

Loading [MathJax]/extensions/Safe.js



In [57]:

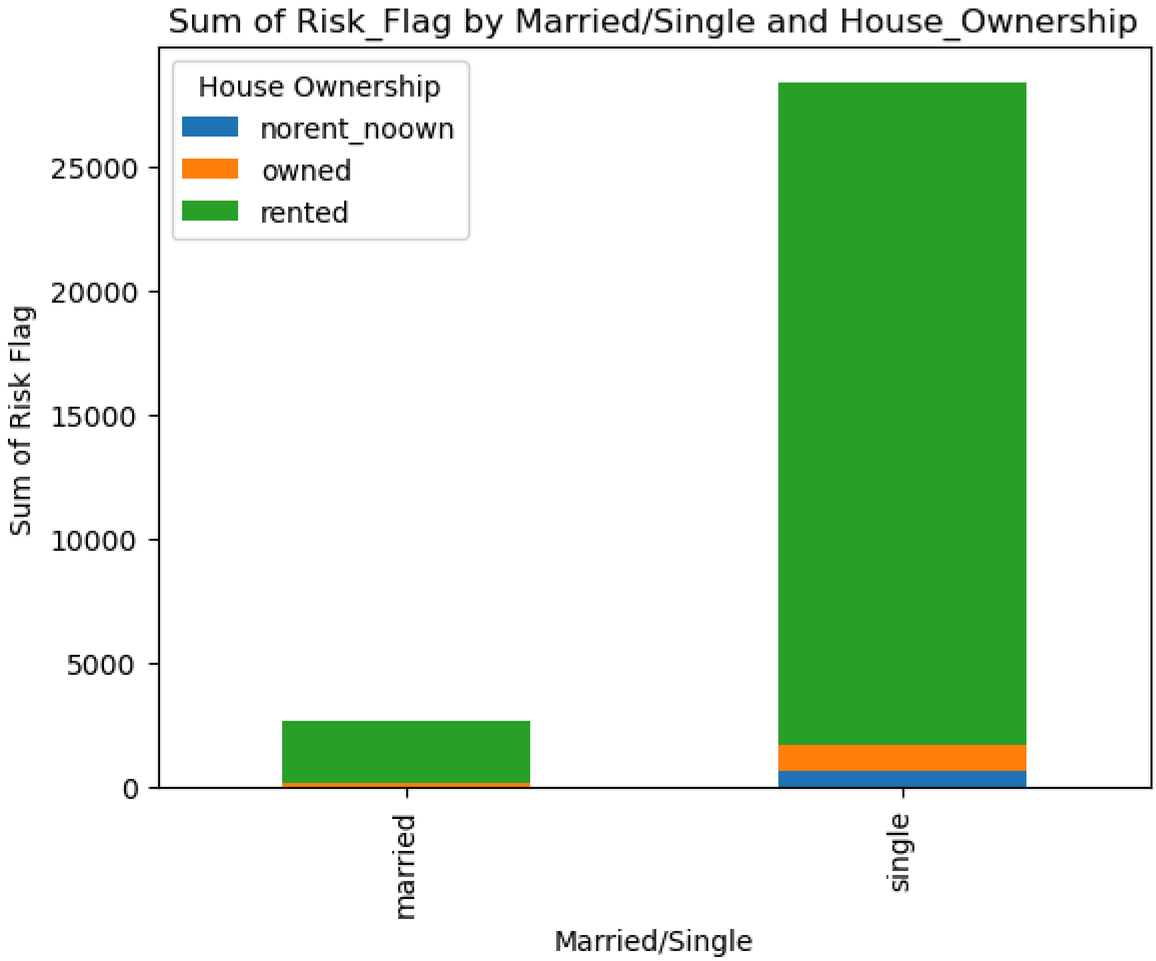
```
House = df.groupby(['House_Ownership'])['Risk_Flag'].sum()
House.plot(kind='pie',autopct='%1.1f%%',title='Risk Factor by House own',ylabel='',shado
plt.show()
```



In [61]:

```
plt.figure(figsize=(15,8))
Married_House = df.groupby(['Married/Single','House_Ownership'])['Risk_Flag'].sum().unst
Married_House.plot(kind='bar',stacked=True)
plt.xlabel('Married/Single')
plt.ylabel('Sum of Risk Flag')
plt.title('Sum of Risk_Flag by Married/Single and House_Ownership')
plt.legend(title='House Ownership')
```

```
<Figure size 1500x800 with 0 Axes>
```



Top 5 High Risk Profession

In [62]:

```
profession_risk = df.groupby('Profession')['Risk_Flag'].mean().reset_index()
profession_high_risk = profession_risk.sort_values(by='Risk_Flag', ascending=False)

profession_high_risk.head(5)
```

Out[62]:

	Profession	Risk_Flag
38	Police_officer	0.164052
7	Chartered_Accountant	0.153572
3	Army_officer	0.152113
46	Surveyor	0.151464
43	Software_Developer	0.148427

Top 5 Low Risk Profession

In [63]:

```
profession_low_risk = profession_risk.sort_values(by='Risk_Flag',ascending=True)
profession_low_risk.head(5)
```

Out[63]:

	Profession	Risk_Flag
49	Technology_specialist	0.081486
36	Petroleum_Engineer	0.085102
29	Industrial_Engineer	0.098667
20	Economist	0.099278
23	Financial_Analyst	0.103155

In [64]:

```
df.drop(columns=['CITY','STATE'],inplace= True)
df
```

Out[64]:

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Professi
0	1	1303834	23	3	single	rented	no	Mechanical_engine
1	2	7574516	40	10	single	rented	no	Software_Develop
2	3	3991815	66	4	married	rented	no	Technical_writ
3	4	6256451	41	2	single	rented	yes	Software_Develop
4	5	5768871	47	11	single	rented	no	Civil_serva
...	...	...	...	...	...	...	...	
251995	251996	8154883	43	13	single	rented	no	Surge
251996	251997	2843572	26	10	single	rented	no	Army_offic
251997	251998	4522448	46	7	single	rented	no	Design_Engine
251998	251999	6507128	45	0	single	rented	no	Graphic_Design
251999	252000	9070230	70	17	single	rented	no	Statistici

252000 rows × 11 columns

In [65]:

```
df_encoded = pd.get_dummies(df, columns=['Married/Single', 'House_Ownership', 'Car_Owner
df_encoded
```

Out[65]:

	Id	Income	Age	Experience	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag	Married/Si
0	1	1303834	23	3	3	13	0	
1	2	7574516	40	10	9	13	0	
2	3	3991815	66	4	4	10	0	
3	4	6256451	41	2	2	12	1	
4	5	5768871	47	11	3	14	1	
...	...	...	...	...	...	...	...	
251995	251996	8154883	43	13	6	11	0	
251996	251997	2843572	26	10	6	11	0	
251997	251998	4522448	46	7	7	12	0	
251998	251999	6507128	45	0	0	10	0	
251999	252000	9070230	70	17	7	11	0	

252000 rows × 65 columns

In [66]:

```
X = df_encoded.drop('Risk_Flag',axis=1)
Y = df['Risk_Flag']
```

In [67]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [68]:

```
RFC = RandomForestClassifier(n_estimators=100,criterion='gini',random_state=42,min_sampl
```

In [69]:

```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2)
```

In [70]:

```
RFC.fit(X_train,y_train)
```

Out[70]:

```
▼                     RandomForestClassifier
RandomForestClassifier(min_samples_split=15, random_state=42)
```

In [71]:

```
y_pred = RFC.predict(X_test)
```

In [72]:

```
from sklearn.metrics import accuracy_score
```

In [73]:

```
accuracy_score(y_test,y_pred)
```

Out[73]:

```
0.9089285714285714
```

In [ ]:

Loading [MathJax]/extensions/Safe.js