In [1]:
```
import pandas as pd

# Dataset con dati mancanti rappresentati da None o NaN
dataset = [
    {"età": 25, "punteggio": 90, "ammesso": 1},
    {"età": None, "punteggio": 85, "ammesso": 0},
    {"età": 28, "punteggio": None, "ammesso": 1},
    {"età": None, "punteggio": 75, "ammesso": 1},
    {"età": 23, "punteggio": None, "ammesso": None},
    {"età": 23, "punteggio": 77, "ammesso": None},
]
df = pd.DataFrame(dataset)
df
```

Out[1]:

	età	punteggio	ammesso
0	25.0	90.0	1.0
1	NaN	85.0	0.0
2	28.0	NaN	1.0
3	NaN	75.0	1.0
4	23.0	NaN	NaN
5	23.0	77.0	NaN

In [2]:
```
df["punteggio"]
```

Out[2]:
```
0    90.0
1    85.0
2     NaN
3    75.0
4     NaN
5    77.0
Name: punteggio, dtype: float64
```

In [3]:
```
# Identificazione delle righe con dati mancanti
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

Out[3]:

	età	punteggio	ammesso
1	NaN	85.0	0.0
2	28.0	NaN	1.0
3	NaN	75.0	1.0
4	23.0	NaN	NaN
5	23.0	77.0	NaN

In [4]:
```
#Conta quante righe con dati mancanti ci sono in totale
totale_dati_mancanti = righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

Out[4]:
```
5
```

In [5]:
```
print("Righe con dati mancanti:")
print(righe_con_dati_mancanti)
print("Totale dati mancanti:", totale_dati_mancanti)
```

```
Righe con dati mancanti:
    età  punteggio  ammesso
1   NaN       85.0      0.0
2  28.0        NaN      1.0
3   NaN       75.0      1.0
4  23.0        NaN      NaN
5  23.0       77.0      NaN
Totale dati mancanti: 5
```

In [6]:
```python
import pandas as pd

# Dataset con dati mancanti rappresentati da None o NaN
dataset = [
    {"nome": "Alice", "età": 25, "punteggio": 90, "email": "alice@email.com"},
    {"nome": "Bob", "età": 22, "punteggio": None, "email": None},
    {"nome": "Charlie", "età": 28, "punteggio": 75, "email": "charlie@email.com"},
]

# Converti il dataset in un DataFrame
df = pd.DataFrame(dataset)
df
```

Out[6]:

	nome	età	punteggio	email
0	Alice	25	90.0	alice@email.com
1	Bob	22	NaN	None
2	Charlie	28	75.0	charlie@email.com

In [7]:
```python
# Rimuovi le righe con dati mancanti
df1=df.dropna(inplace=False)
df1
```

Out[7]:

	nome	età	punteggio	email
0	Alice	25	90.0	alice@email.com
2	Charlie	28	75.0	charlie@email.com

In [8]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

# Genera dati di esempio
data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]
}
# Crea un DataFrame
df = pd.DataFrame(data)
df1=pd.DataFrame()
df
```

Out[8]:

	Variable1	Variable2	Missing_Column
0	1	1.0	A
1	2	2.0	B
2	3	NaN	A
3	4	4.0	C
4	5	NaN	NaN

In [9]:
```
# Trattamento dei missing values nelle variabili numeriche
numeric_cols = df.select_dtypes(include=['number'])
numeric_cols.columns
```

Out[9]:
```
Index(['Variable1', 'Variable2'], dtype='object')
```

In [10]:
```
df1[numeric_cols.columns] = df[numeric_cols.columns].fillna(df[numeric_cols.columns].mea
df1
```

Out[10]:

	Variable1	Variable2
0	1	1.000000
1	2	2.000000
2	3	2.333333
3	4	4.000000
4	5	2.333333

In [11]:
```
# Trattamento dei missing values nelle variabili categoriche
categorial_cols = df.select_dtypes(exclude=['number'])
categorial_cols.columns
```

Out[11]:
```
Index(['Missing_Column'], dtype='object')
```

In [12]:
```
df1[categorial_cols.columns] = df[categorial_cols.columns].fillna(df[categorial_cols.col
df1
```

Out[12]:

	Variable1	Variable2	Missing_Column
0	1	1.000000	A
1	2	2.000000	B
2	3	2.333333	A
3	4	4.000000	C
4	5	2.333333	A

In [13]:
```
print(f"il primo con i valori mancanti \n{df} \ne il secondo con i missing values sostit
```

```
il primo con i valori mancanti
   Variable1  Variable2 Missing_Column
0          1        1.0              A
1          2        2.0              B
2          3        NaN              A
3          4        4.0              C
4          5        NaN            NaN
e il secondo con i missing values sostituiti
   Variable1  Variable2 Missing_Column
0          1   1.000000              A
1          2   2.000000              B
2          3   2.333333              A
3          4   4.000000              C
4          5   2.333333              A
```

In [14]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}
# Crea un DataFrame
df = pd.DataFrame(data)
df
```

Out[14]:

	Feature1	Feature2	Feature3
0	1.0	NaN	1.0
1	2.0	2.0	NaN
2	NaN	3.0	3.0
3	4.0	4.0	4.0
4	5.0	NaN	5.0

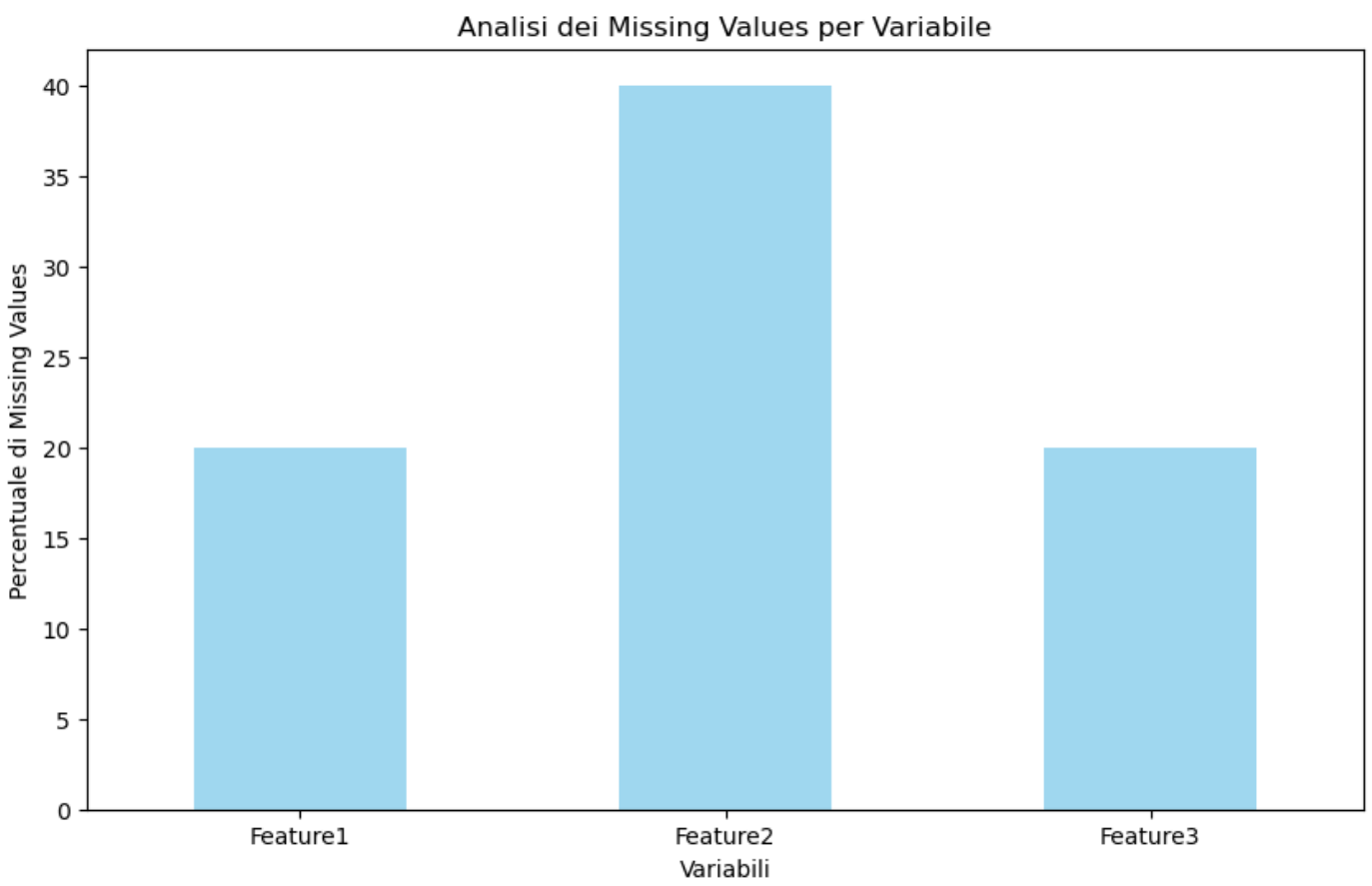
In [15]:
```python
df.isnull().sum()
```

Out[15]:
```
Feature1    1
Feature2    2
Feature3    1
dtype: int64
```

In [16]:
```python
missing_percent = (df.isnull().sum() / len(df)) *100
missing_percent
```

Out[16]:
```
Feature1    20.0
Feature2    40.0
Feature3    20.0
dtype: float64
```

In [17]:
```python
missing_percent = (df.isnull().sum() / len(df)) *100

# Crea il grafico a barre
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue',alpha=0.8)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=0)
```



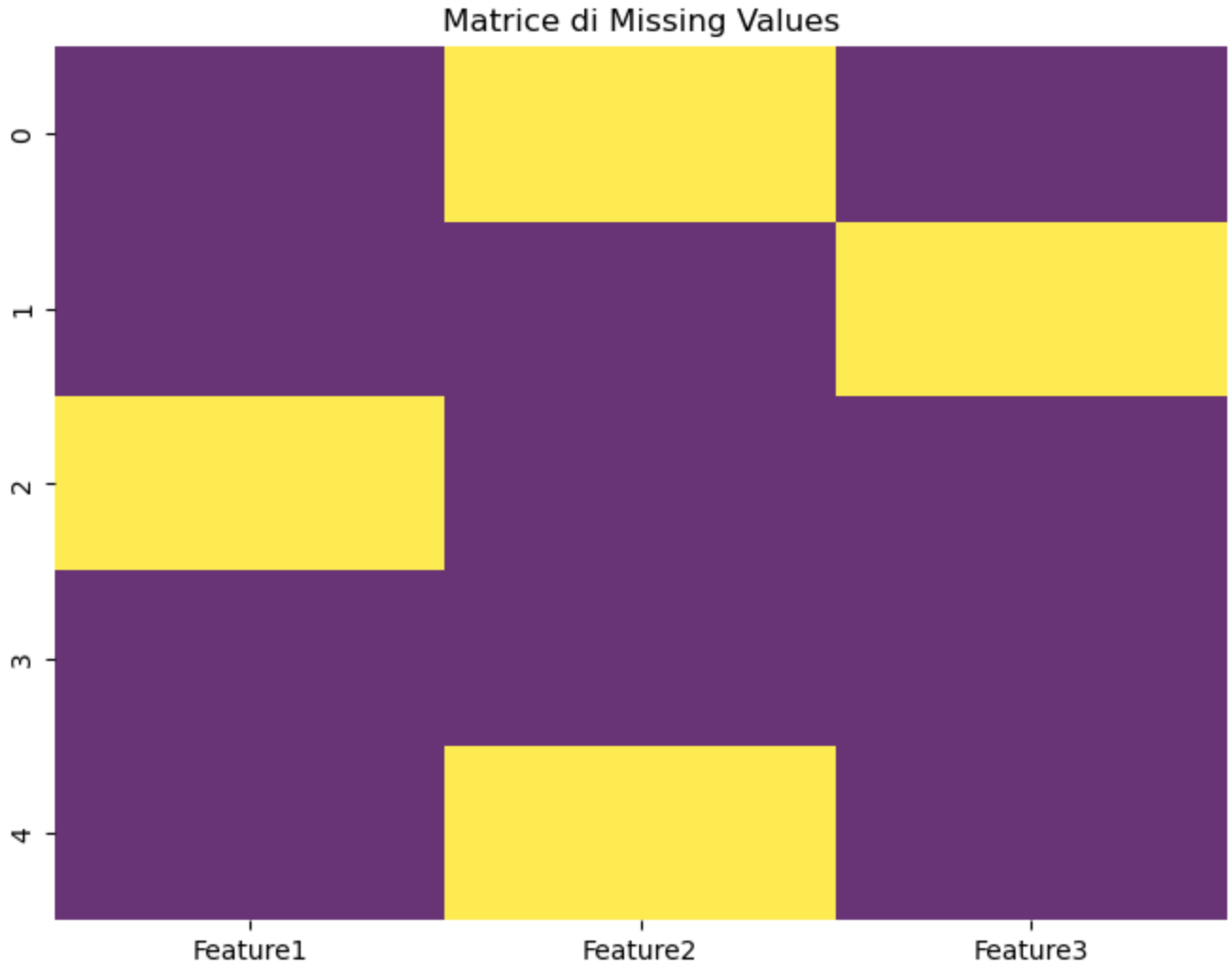
In [18]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Genera dati di esempio
data = {
    'Feature1': [1, 2, np.nan, 4, 5],
    'Feature2': [np.nan, 2, 3, 4, np.nan],
    'Feature3': [1, np.nan, 3, 4, 5]
}

# Crea un DataFrame
df = pd.DataFrame(data)

# Calcola la matrice di missing values
missing_matrix = df.isnull()
missing_matrix
```

Out[18]:

	Feature1	Feature2	Feature3
0	False	True	False
1	False	False	True
2	True	False	False
3	False	False	False
4	False	True	False

In [19]:
```
# Crea una heatmap colorata
plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.show
```

Out[19]: <function matplotlib.pyplot.show(close=None, block=None)>



In [20]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Età': np.random.randint(18, 70, size=1000),
    'Genere': np.random.choice(['Maschio', 'Femmina'], size=1000),
    'Punteggio': np.random.uniform(0, 100, size=1000),
    'Reddito': np.random.normal(50000, 15000, size=1000)
}

df = pd.DataFrame(data)

# Visualizza le prime righe del dataset
print(df.head())
```

```
   Età   Genere  Punteggio       Reddito
0   56  Maschio  85.120691  52915.764524
1   69  Maschio  49.514653  44702.505608
2   46  Maschio  48.058658  55077.257652
3   32  Femmina  59.240778  45568.978848
4   60  Maschio  82.468097  52526.914644
```

In [21]:
```
print(df.info())
# Statistiche descrittive
```

```
print (df.describe())
```

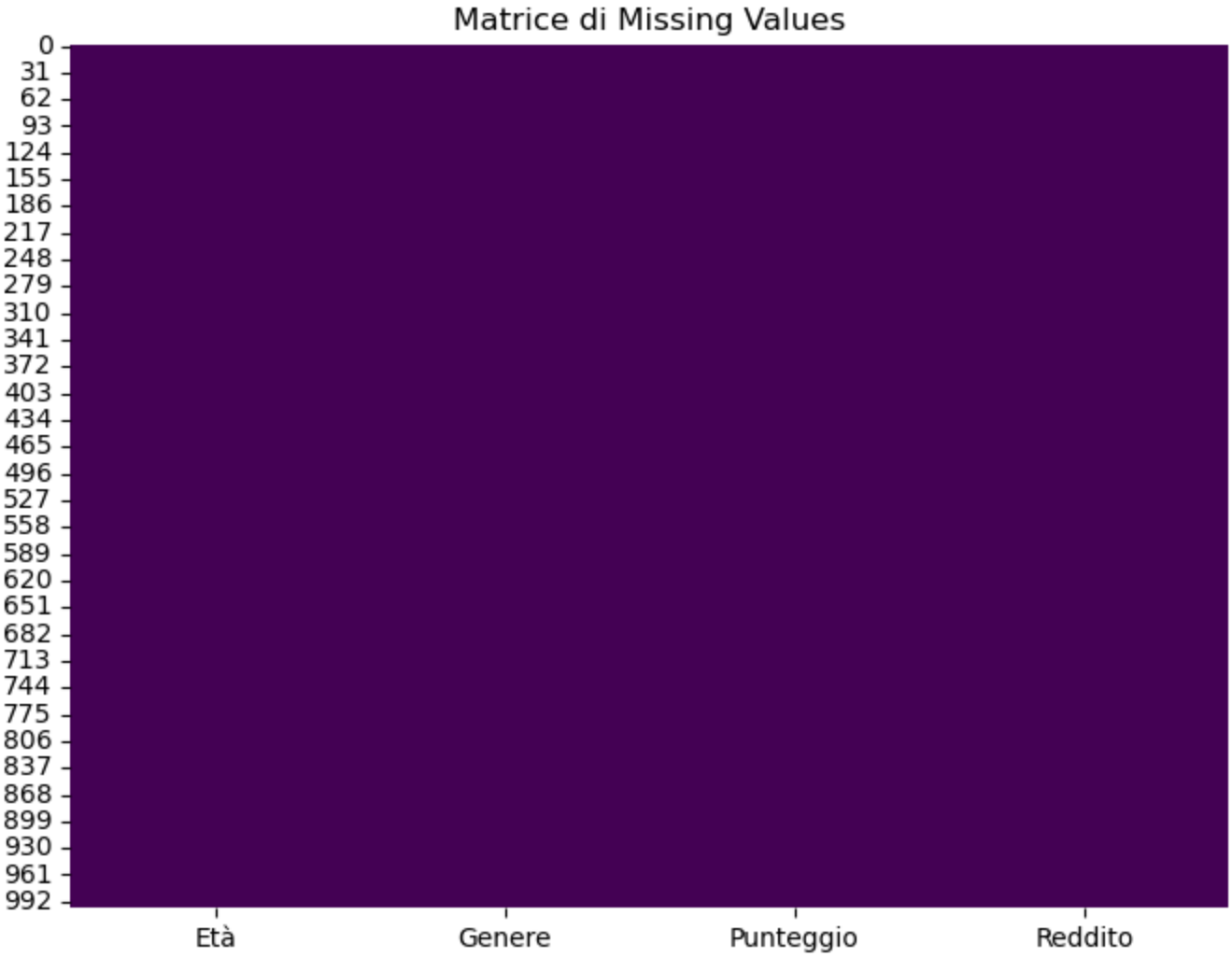
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Età        1000 non-null   int32  
 1   Genere     1000 non-null   object 
 2   Punteggio  1000 non-null   float64
 3   Reddito    1000 non-null   float64
dtypes: float64(2), int32(1), object(1)
memory usage: 27.5+ KB
None
                Età    Punteggio       Reddito
count  1000.00000  1000.000000   1000.000000
mean     43.81900    50.471078  50241.607607
std      14.99103    29.014970  14573.000585
min      18.00000     0.321826   4707.317663
25%      31.00000    24.690382  40538.177863
50%      44.00000    51.789520  50099.165858
75%      56.00000    75.549365  60089.683773
max      69.00000    99.941373  97066.228005
```

In [22]:
```
# Gestione dei valori mancanti
missing_data = df.isnull().sum()
print("Valori mancanti per ciascuna colonna")
print(missing_data)
```

```
Valori mancanti per ciascuna colonna
Età          0
Genere       0
Punteggio    0
Reddito      0
dtype: int64
```

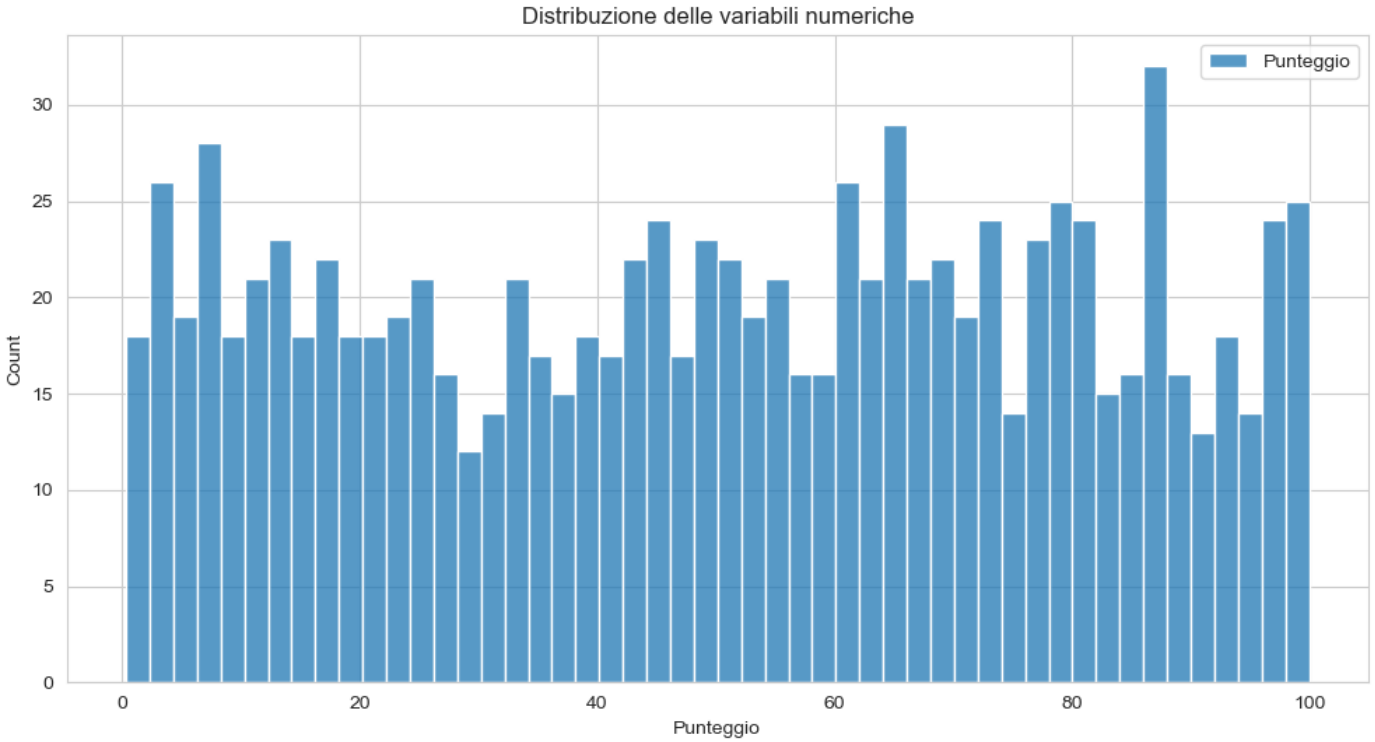
In [23]:
```
# Visualizza una heatmap dei valori mancanti
plt.figure(figsize=(8, 6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False,)
plt.title('Matrice di Missing Values')
plt.show
```

Out[23]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



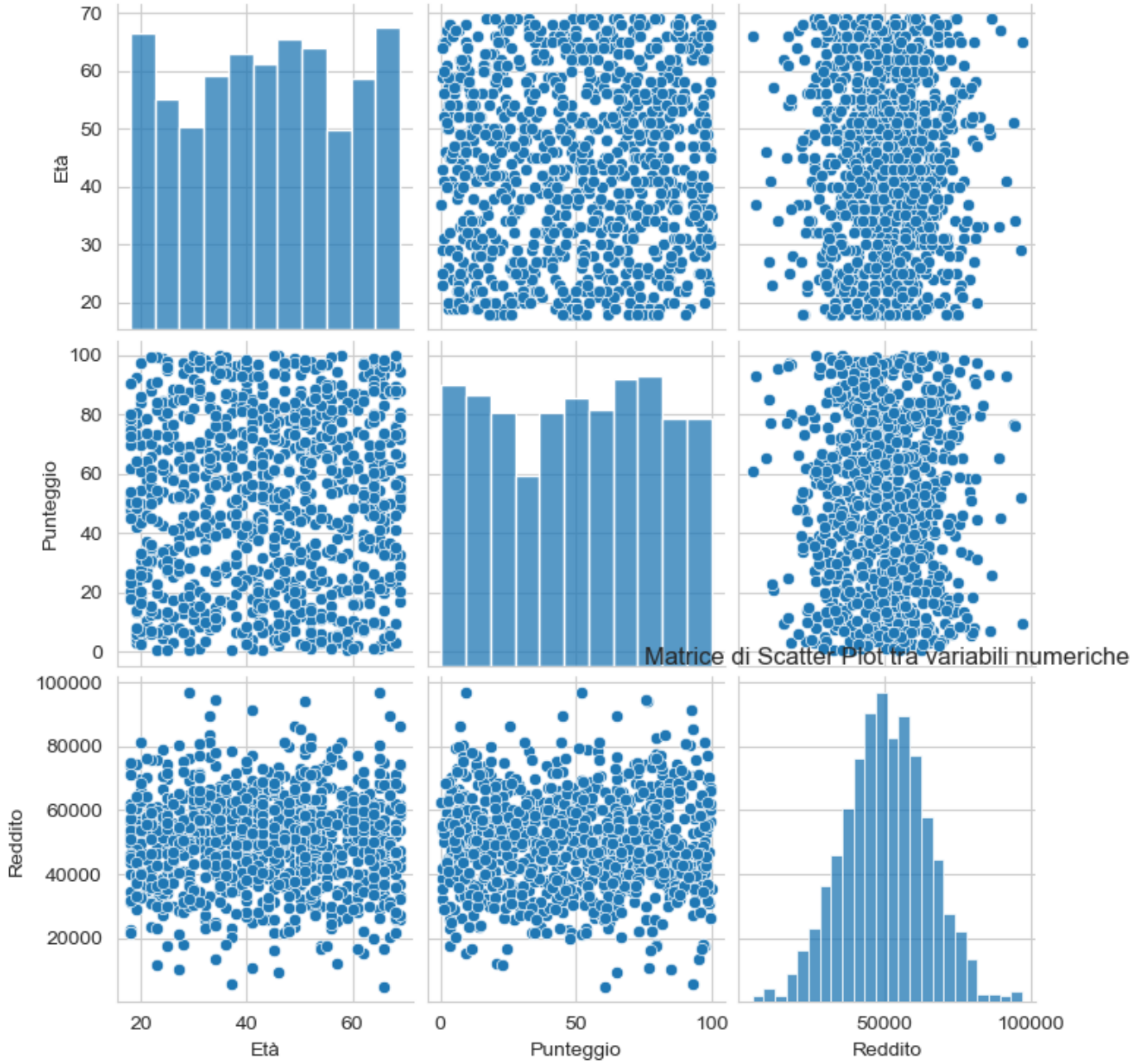
In [24]:
```
# Visualizza la distribuzione delle variabili numeriche
plt.figure(figsize=(12, 6))
sns.set_style("whitegrid")
sns.histplot(df["Punteggio"], kde=False, bins=50, label="Punteggio")
plt.legend()
plt.title('Distribuzione delle variabili numeriche')
plt.show
```

Out[24]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

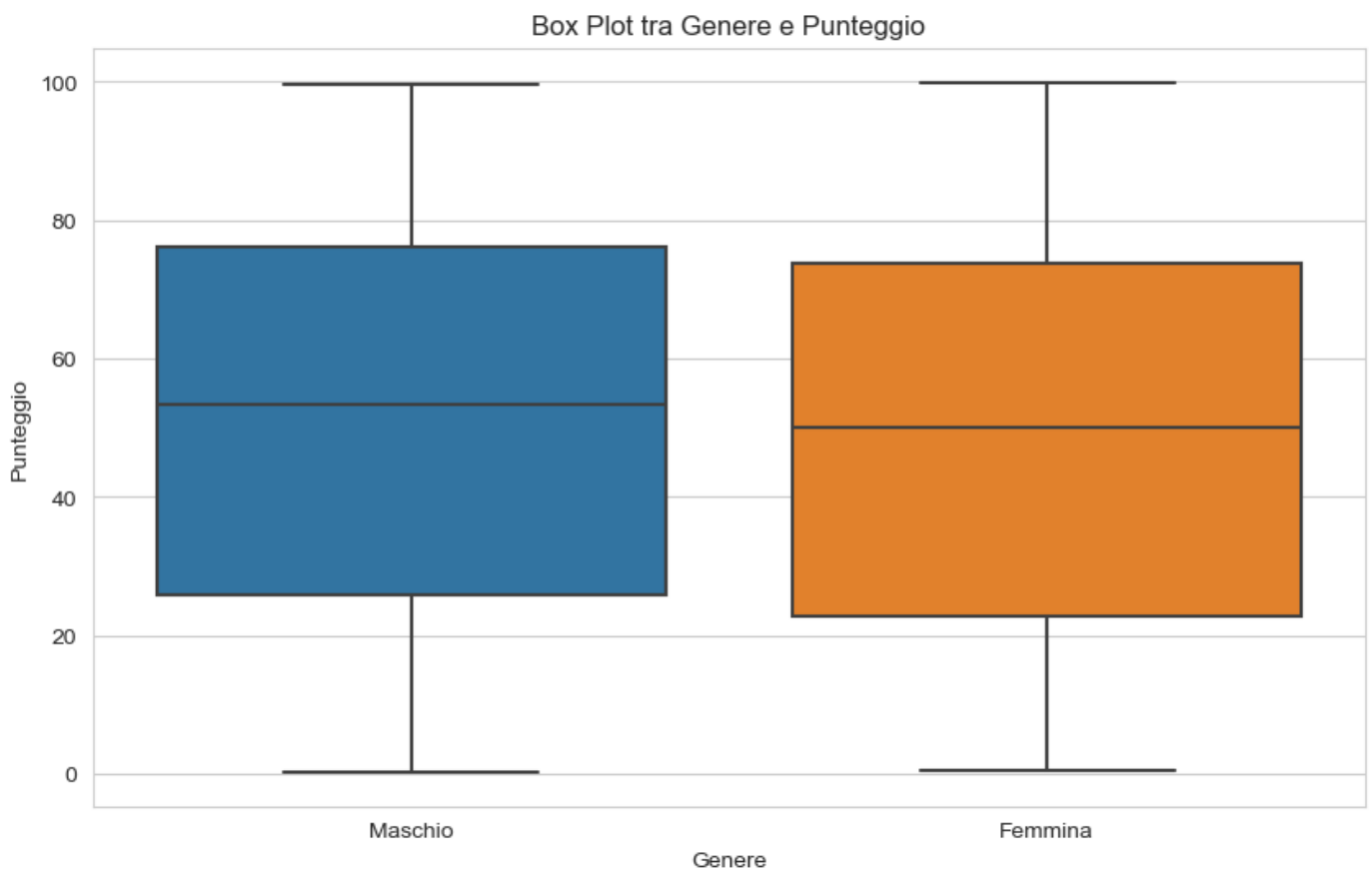


In [25]:
```
numeric_features = df.select_dtypes(include=[np.number])
sns.pairplot(df[numeric_features.columns])
plt.title('Matrice di Scatter Plot tra variabili numeriche')
plt.show()
```

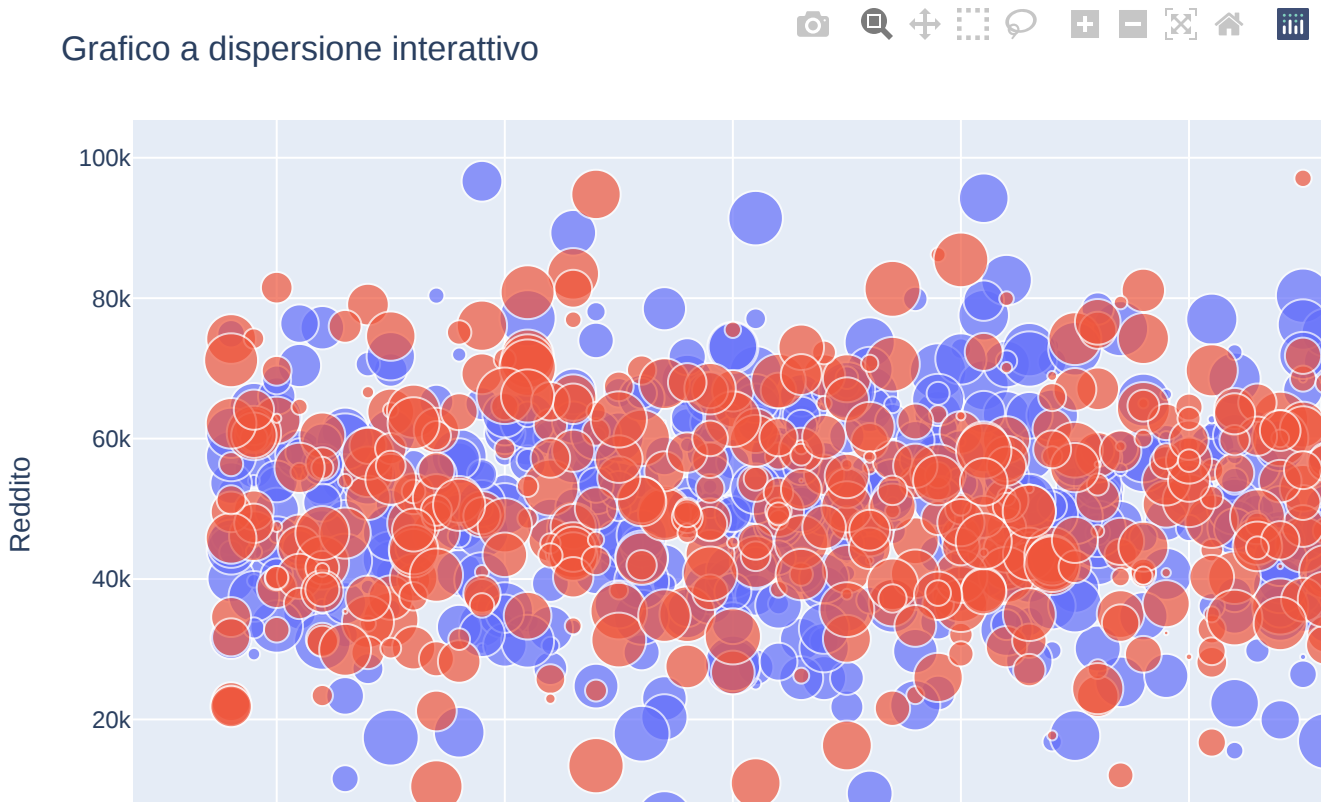
```
C:\Users\zetam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [26]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Genere', y='Punteggio', data=df)
plt.title('Box Plot tra Genere e Punteggio')
plt.show()
```



In [27]:
```
import plotly.express as px

fig = px.scatter(df, x='Età', y='Reddito', color='Genere', size='Punteggio')
fig.update_layout(title='Grafico a dispersione interattivo')
fig.show()
```

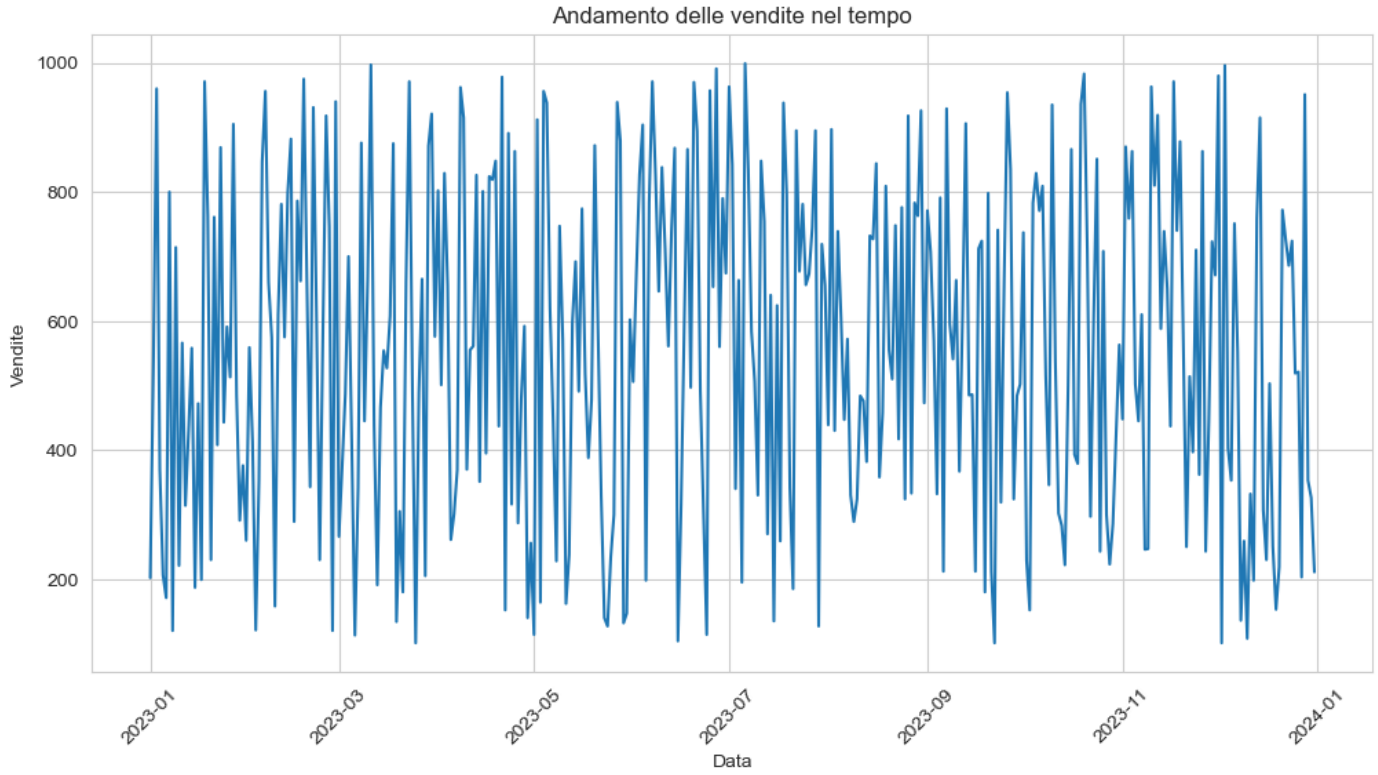


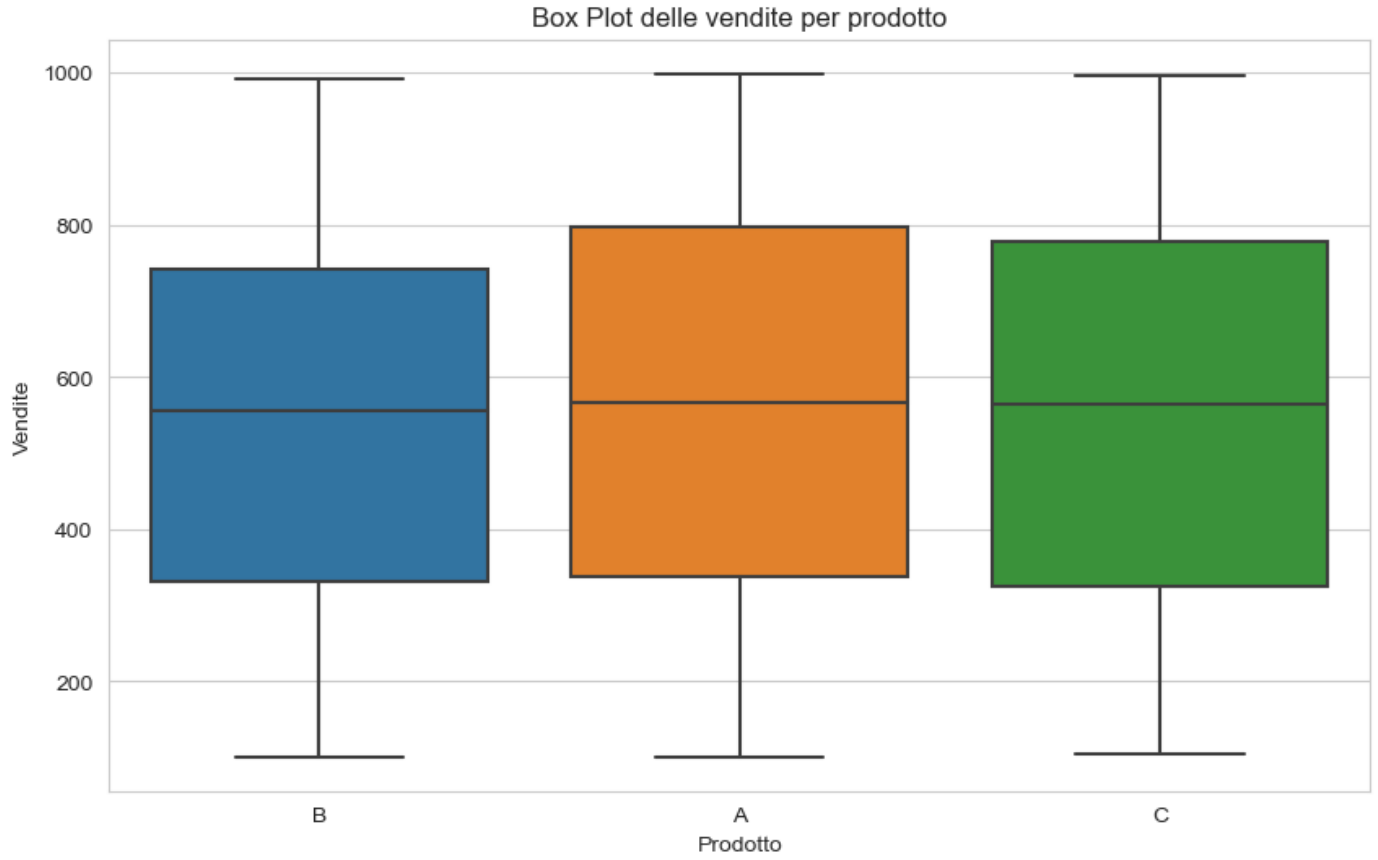
In [28]:

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

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31', freq='D'),
    'Vendite': np.random.randint(100, 1000, size=365),
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365)
}

df = pd.DataFrame(data)

# Visualizza le prime righe del dataset
print(df.head())

# Visualizza un grafico delle vendite nel tempo
plt.figure(figsize=(12, 6))
sns.lineplot(x='Data', y='Vendite', data=df)
plt.title('Andamento delle vendite nel tempo')
plt.xlabel('Data')
plt.ylabel('Vendite')
plt.xticks(rotation=45)
plt.show()

# Visualizza una box plot delle vendite per prodotto
plt.figure(figsize=(10, 6))
sns.boxplot(x='Prodotto', y='Vendite', data=df)
plt.ox Plot delle vendite per prodotto')
```

```
plt.xlabel('Prodotto')
plt.ylabel('Vendite')
plt.show()
```

```
        Data  Vendite Prodotto
0 2023-01-01      202        B
1 2023-01-02      535        A
2 2023-01-03      960        C
3 2023-01-04      370        A
4 2023-01-05      206        A
```





In [29]:
```
import pandas as pd
matplotlib.pyplot as plt
```

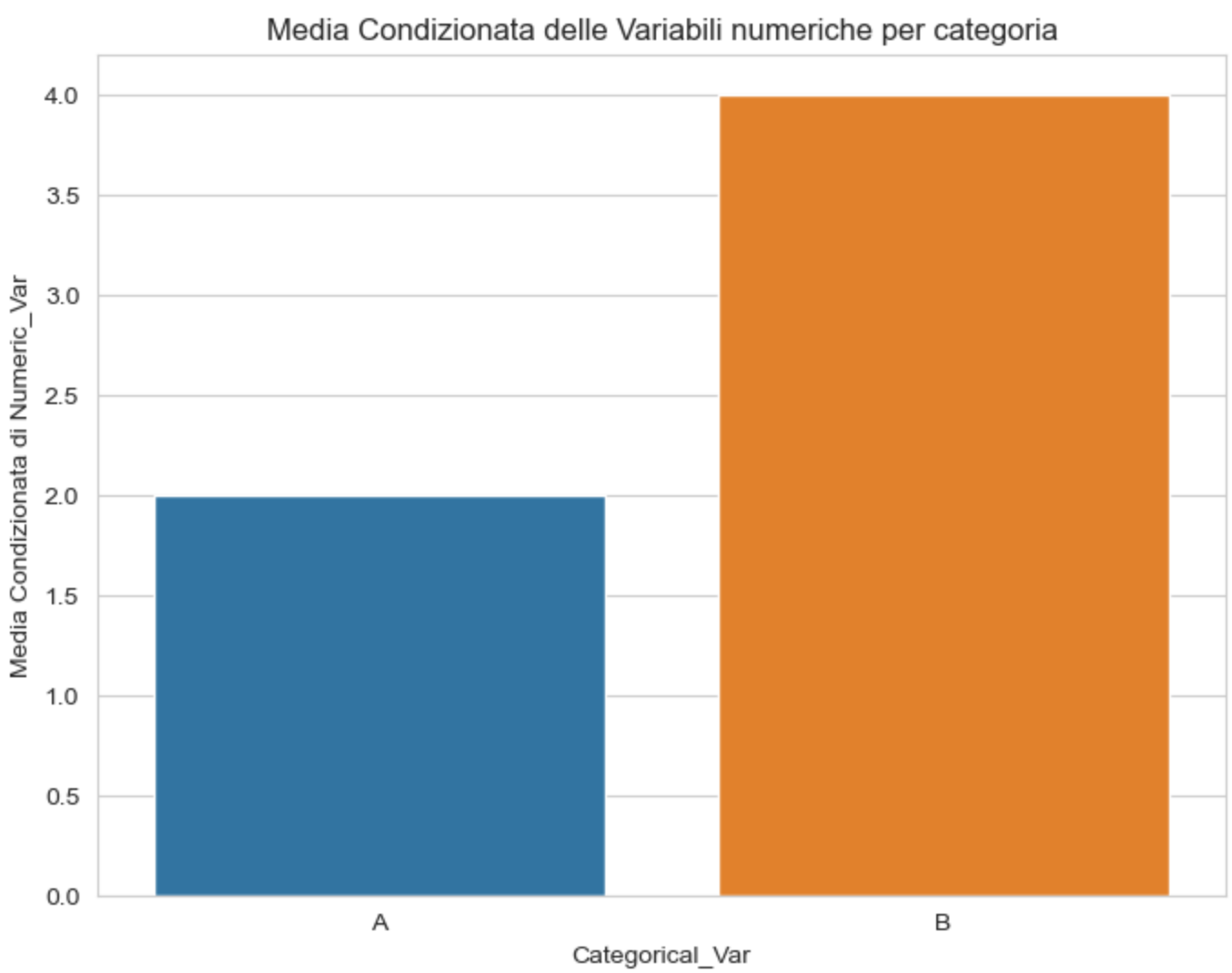
```
import numpy as np
import seaborn as sns

# Genera dati di esempio
data = {
    'Numeric_Var': [1, 2, 3, 4, np.nan, 6],
    'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']
}

# Crea un DataFrame
df = pd.DataFrame(data)
print(df)
```

```
   Numeric_Var Categorical_Var
0          1.0              A
1          2.0              B
2          3.0              A
3          4.0              B
4          NaN              A
5          6.0              B
```

In [30]:

```
#calcola la media condizionata
conditional_means = df['Numeric_Var'].fillna(df.groupby('Categorical_Var')['Numeric_Var'

#aggiorna la colonna numeric_var con la media condizionata
df['Numeric_Var'] = conditional_means
print(df)

#crea un graico a barre per mostrare la sedia condizionata per ogni categoria
plt.figure(figsize=(8,6))
sns.barplot(data=df, x='Categorical_Var', y='Numeric_Var', errorbar=None)
plt.xlabel('Categorical_Var')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili numeriche per categoria')
plt.show()
```

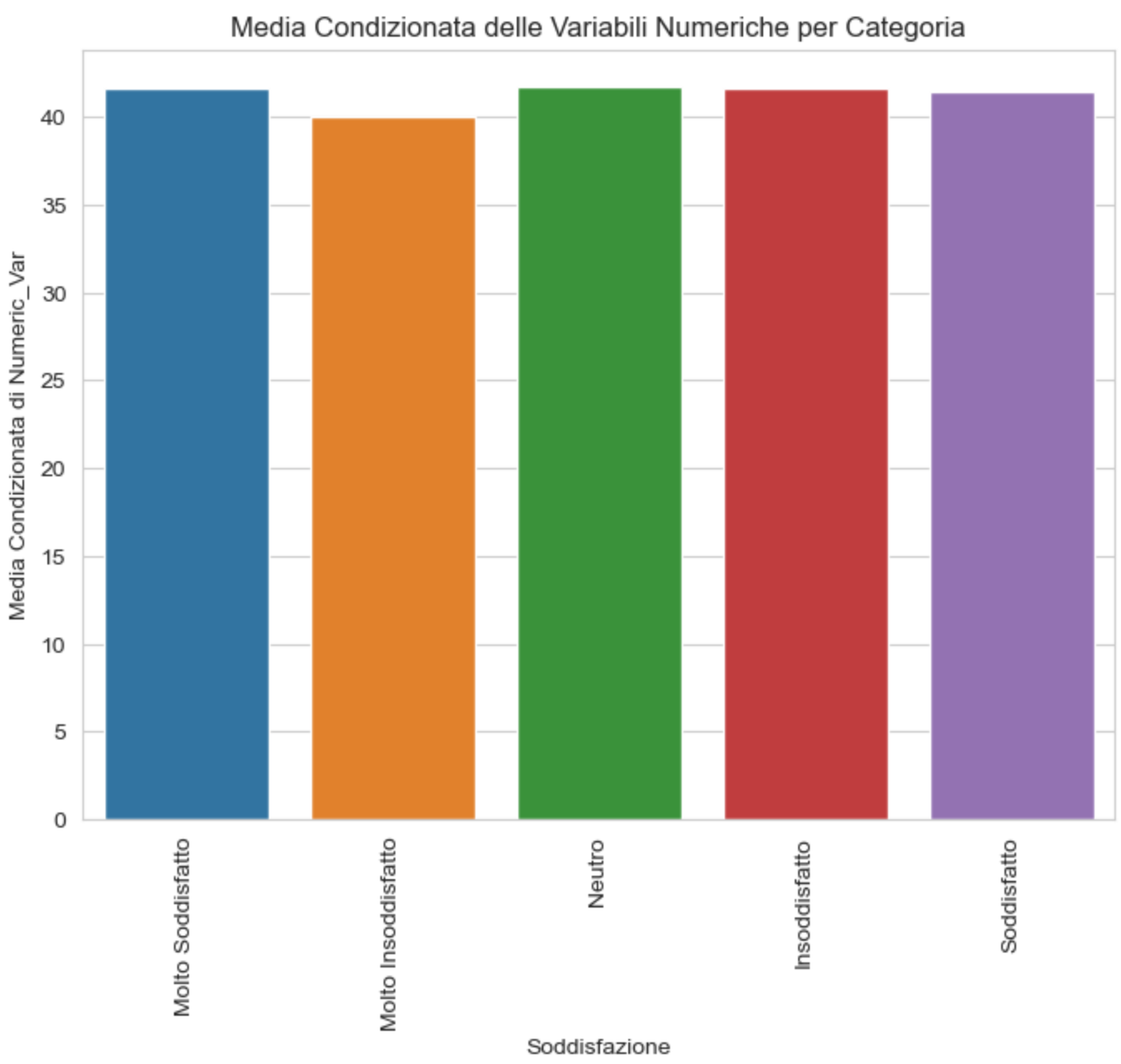
```
   Numeric_Var Categorical_Var
0          1.0              A
1          2.0              B
2          3.0              A
3          4.0              B
4          2.0              A
5          6.0              B
```



In [31]:

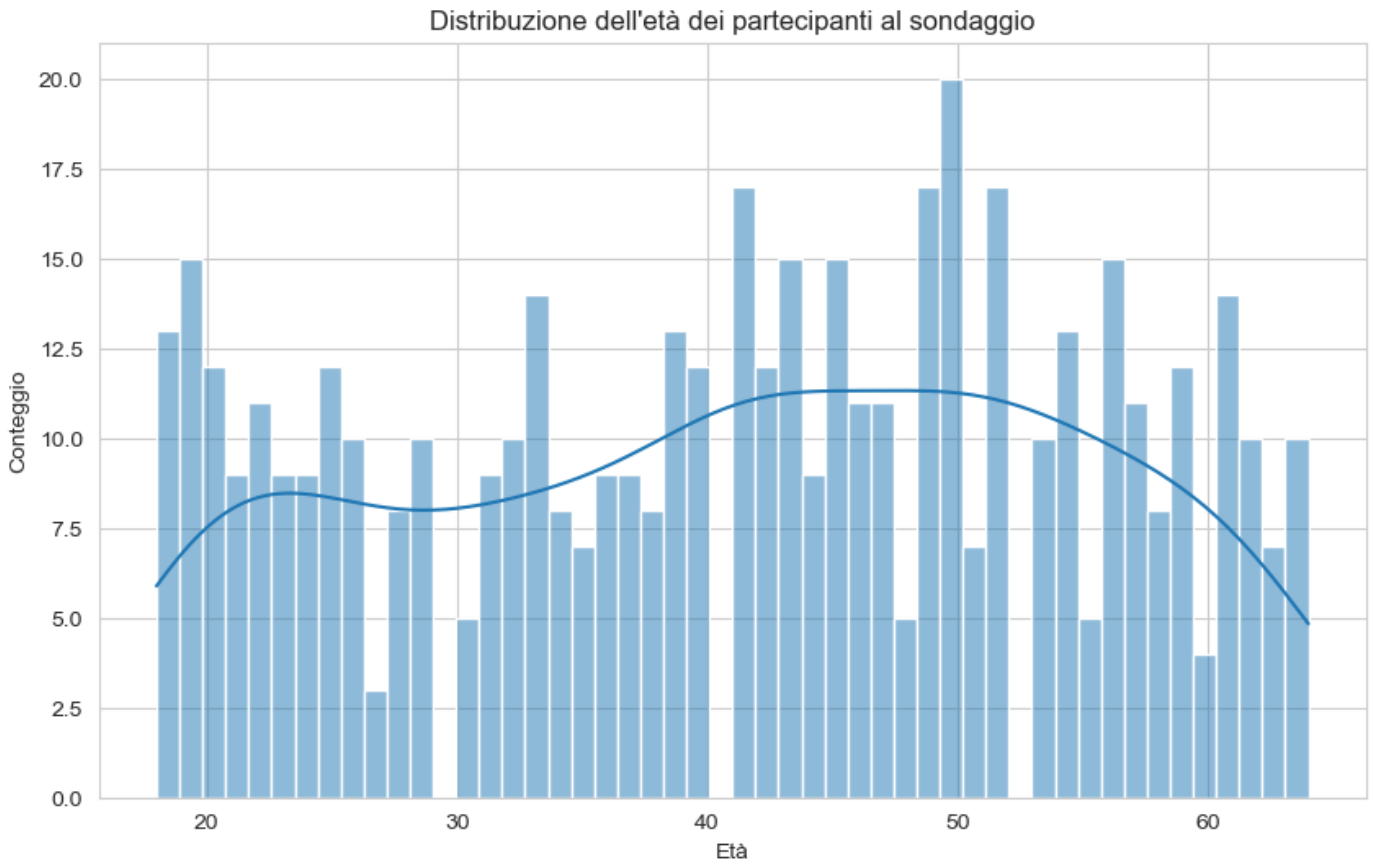
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Genera dati casuali per l'esplorazione
np.random.seed(42)
data = {
    'Età': np.random.randint(18, 65, size=500),
    'Soddisfazione': np.random.choice(['Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'In
}

df = pd.DataFrame(data)
print(df)
conditional_means = df.groupby('Soddisfazione')['Età'].transform('mean')

df['Numeric_Var'] = conditional_means
print(df)

# Crea un grafico a barre per mostrare la media condizionata per ogni categoria
plt.figure(figsize=(8, 6))
sns.barplot(data=df, x='Soddisfazione', y='Numeric_Var', ci=None)
plt.xlabel('Soddisfazione')
plt.ylabel('Media Condizionata di Numeric_Var')
plt.title('Media Condizionata delle Variabili Numeriche per Categoria')
plt.xticks(rotation=90)

plt.show()
```

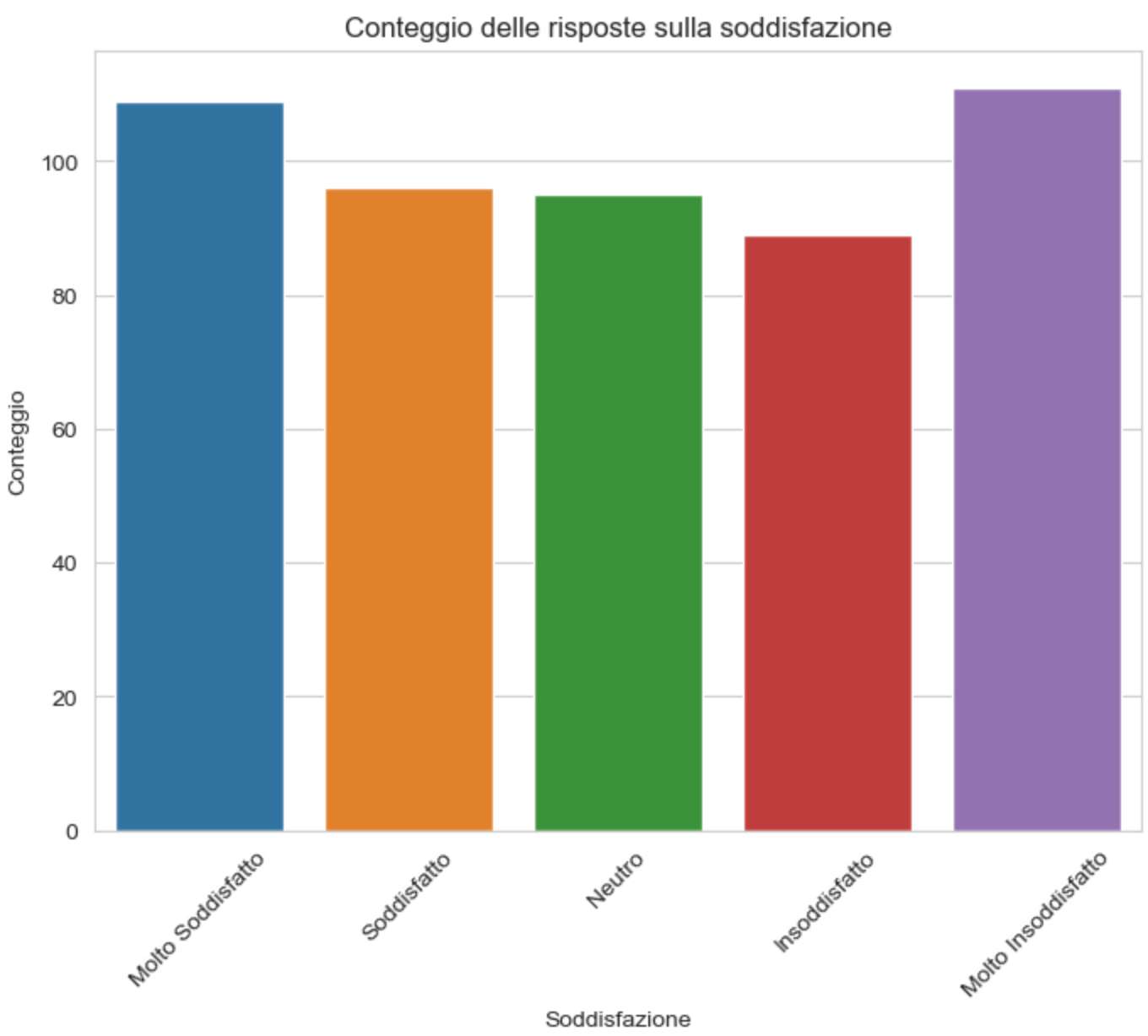
```
     Età        Soddisfazione
0     56    Molto Soddisfatto
1     46  Molto Insoddisfatto
2     32              Neutro
3     60              Neutro
4     25  Molto Insoddisfatto
..   ...                 ...
495   37    Molto Soddisfatto
496   41    Molto Soddisfatto
497   29    Molto Soddisfatto
498   52    Molto Soddisfatto
499   50    Molto Soddisfatto

[500 rows x 2 columns]
     Età        Soddisfazione  Numeric_Var
0     56    Molto Soddisfatto    41.651376
1     46  Molto Insoddisfatto    40.054054
2     32              Neutro    41.747368
3     60              Neutro    41.747368
4     25  Molto Insoddisfatto    40.054054
..   ...                 ...          ...
495   37    Molto Soddisfatto    41.651376
496   41    Molto Soddisfatto    41.651376
497   29    Molto Soddisfatto    41.651376
498   52    Molto Soddisfatto    41.651376
499   50    Molto Soddisfatto    41.651376

[500 rows x 3 columns]
C:\Users\zetam\AppData\Local\Temp\ipykernel_10996\3910047455.py:22: FutureWarning:


The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.
```



In [32]:
```
# Visualizza le prime righe del dataset
print(df.head())

# Visualizza una distribuzione dell'età
plt.figure(figsize=(10, 6))
sns.histplot(df['Età'], bins=50, kde=True)
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')
plt.xlabel('Età')
plt.ylabel('Conteggio')
plt.show()

# Visualizza un conteggio delle risposte sulla soddisfazione
plt.figure(figsize=(8, 6))
sns.countplot(x='Soddisfazione', data=df, order=['Molto Soddisfatto', 'Soddisfatto', 'Ne
plt.title('Conteggio delle risposte sulla soddisfazione')
plt.xlabel('Soddisfazione')
plt.ylabel('Conteggio')
plt.xticks(rotation=45)
plt.show()
```

```
   Età        Soddisfazione  Numeric_Var
0   56    Molto Soddisfatto    41.651376
1   46  Molto Insoddisfatto    40.054054
2   32              Neutro    41.747368
3   60              Neutro    41.747368
4   25  Molto Insoddisfatto    40.054054
```





In [33]:

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Genera un dataset di esempio con variabili numeriche
np.random.seed(42)
data = pd.DataFrame(np.random.rand(100, 5), columns=['Var1', 'Var2', 'Var3', 'Var4', 'Va

# Aggiungi alcune variabili categoriche generate casualmente
data['Categoria1'] = np.random.choice(['A', 'B', 'C'], size=100)
data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)

# Calcola la matrice di correlazione tra tutte le variabili numeriche
correlation_matrix = data.corr()

# Visualizza la matrice di correlazione come heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",alpha=0.7)
plt.title("Matrice di Correlazione")
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[33], line 14
     11 data['Categoria2'] = np.random.choice(['X', 'Y'], size=100)
     13 # Calcola la matrice di correlazione tra tutte le variabili numeriche
---> 14 correlation_matrix = data.corr()
     16 # Visualizza la matrice di correlazione come heatmap
     17 plt.figure(figsize=(10, 8))

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:10054, in DataFrame.corr(self, method, min_periods, numeric_only)
  10052 cols = data.columns
  10053 idx = cols.copy()
> 10054 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10056 if method == "pearson":
  10057     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1838, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1732, in BlockManager.as_array(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1794, in BlockManager._interleave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():

ValueError: could not convert string to float: 'B'
```

In [34]:

```python
import pandas as pd
import numpy as np

# Impostare il seed per rendere i risultati riproducibili
np.random.seed(41)

# Creare un dataframe vuoto
df = pd.DataFrame()

# Generare dati casuali
n_rows = 10000
df['CatCol1'] = np.random.choice(['A', 'B', 'C'], size=n_rows)
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)
df['NumCol1'] = np.random.randn(n_rows)
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)

# Calcolare il numero totale di missing values desiderati
total_missing_values = int(0.03 * n_rows * len(df.columns))
```

```
# Introdurre missing values casuali
for column in df.columns:
    num_missing_values = np.random.randint(0, total_missing_values + 1)
    missing_indices = np.random.choice(n_rows, size=num_missing_values, replace=False)
    df.loc[missing_indices, column] = np.nan
df
```

Out[34]:

	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
1	A	Y	1.945879	28.0	0.936825
2	C	X	0.988834	42.0	0.751516
3	A	Y	-0.181978	73.0	0.950696
4	B	X	2.080615	74.0	0.903045
...	...	...	...	...	...
9995	C	Y	1.352114	61.0	0.728445
9996	C	Y	1.143642	67.0	0.605930
9997	A	X	-0.665794	54.0	0.071041
9998	C	Y	0.004278	NaN	NaN
9999	A	X	0.622473	95.0	0.751384

10000 rows × 5 columns

In [35]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
len(righe_con_dati_mancanti)
```

Out[35]:
```
3648
```

In [36]:
```
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

Out[36]:

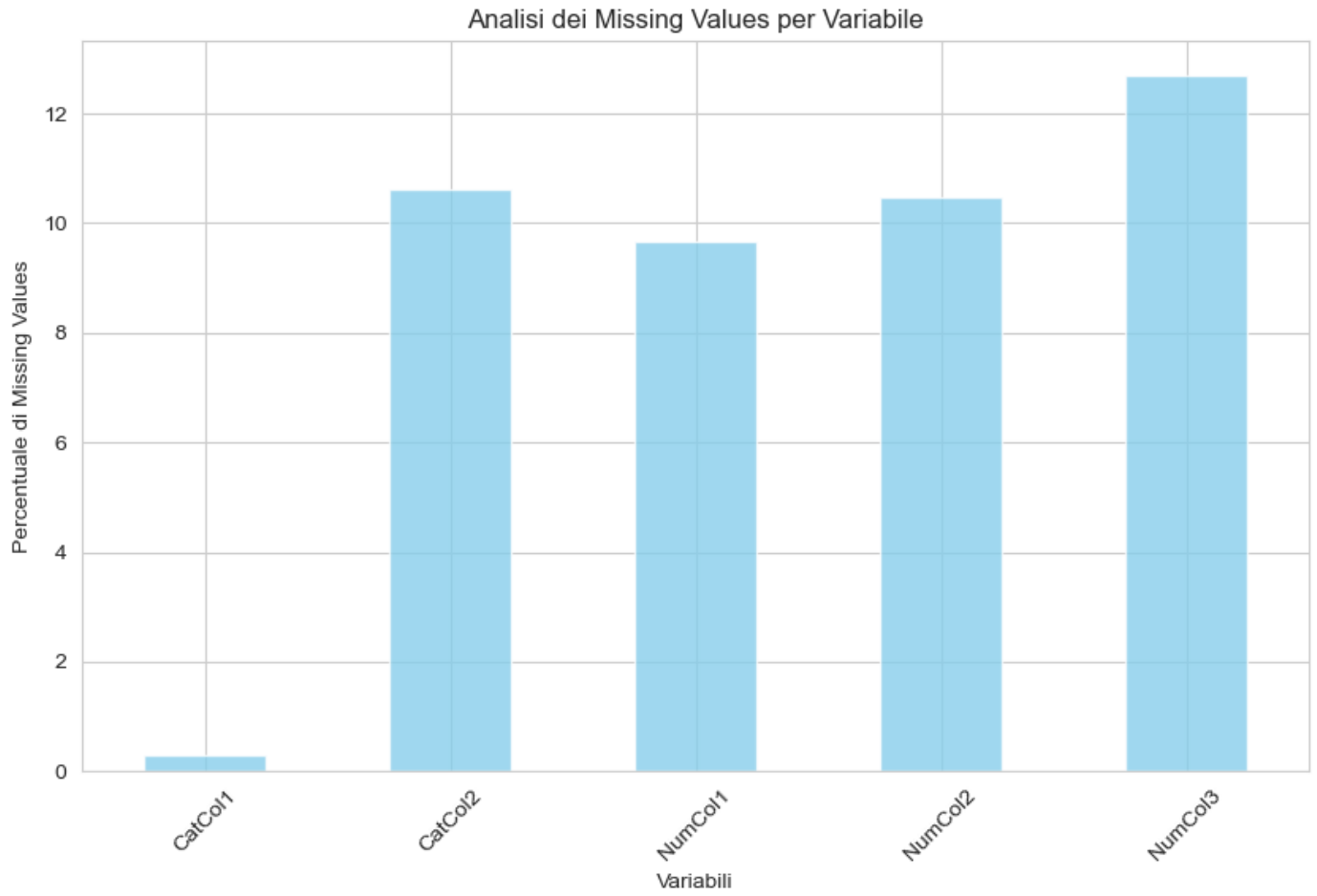
	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
5	B	NaN	NaN	71.0	0.752397
6	B	X	0.080686	31.0	NaN
8	B	Y	-1.291483	NaN	0.868791
12	C	Y	-1.193705	8.0	NaN
...	...	...	...	...	...
9986	C	X	-0.909994	NaN	0.767918
9988	A	Y	NaN	35.0	0.149513
9989	A	NaN	-0.148047	NaN	0.326089
9992	A	Y	-0.048300	58.0	NaN
9998	C	Y	0.004278	NaN	NaN

3648 rows × 5 columns

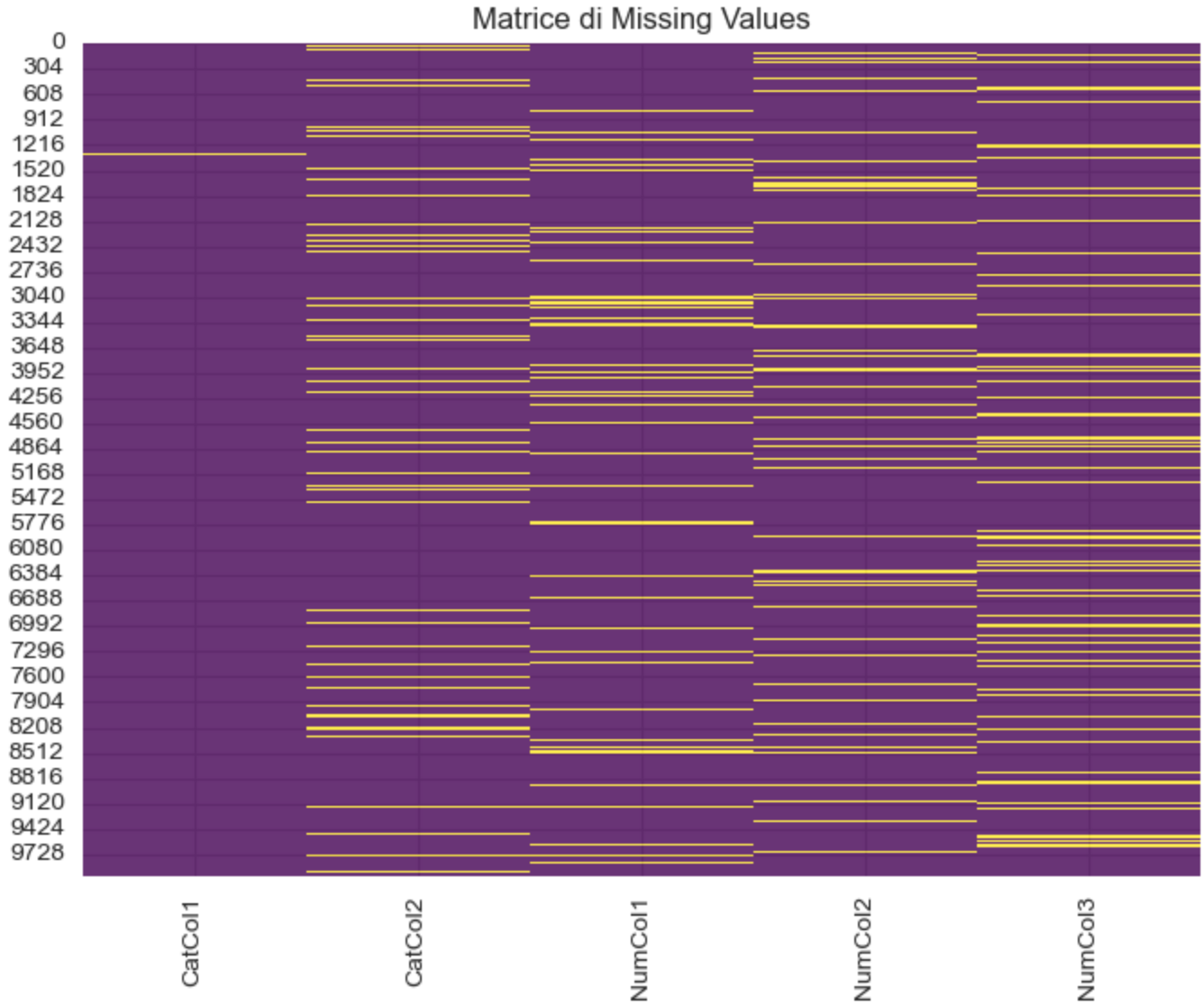
In [37]:
```
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_percent
```

Out[37]:
```
CatCol1     0.29
CatCol2    10.63
NumCol1     9.67
NumCol2    10.48
NumCol3    12.69
dtype: float64
```

In [38]:
```
# Crea il grafico a barre
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue',alpha=0.8)
plt.xlabel('Variabili')
plt.ylabel('Percentuale di Missing Values')
plt.title('Analisi dei Missing Values per Variabile')
plt.xticks(rotation=45)
plt.show()
```

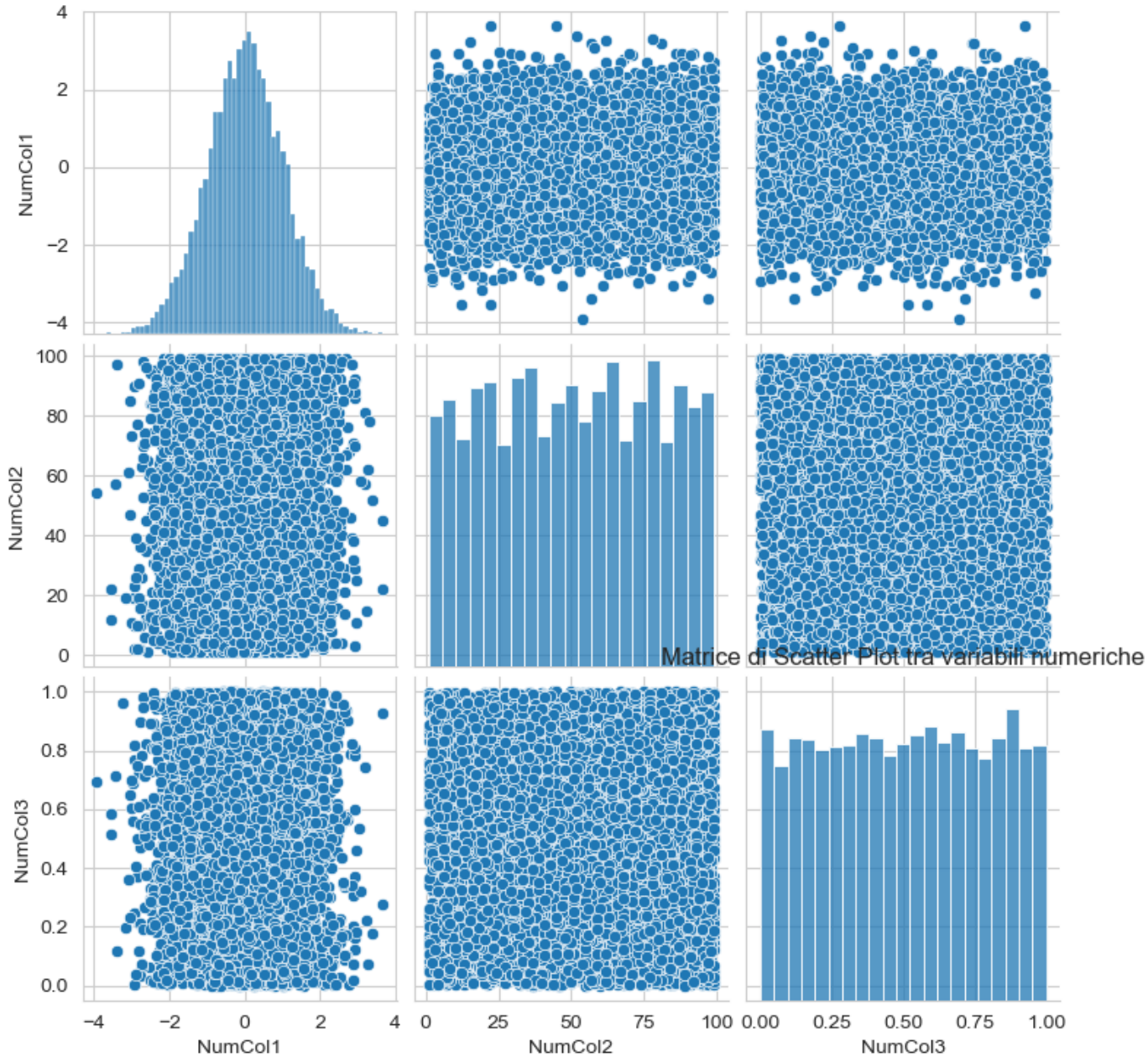


In [39]:
```
missing_matrix = df.isnull()

plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False,alpha=0.8)
plt.title('Matrice di Missing Values')
plt.xticks(rotation=90)

plt.show()
```



In [40]:
```
numeric_features = df.select_dtypes(include=[np.number])
sns.pairplot(df[numeric_features.columns])
plt.title('Matrice di Scatter Plot tra variabili numeriche')
plt.show()
```

```
C:\Users\zetam\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```



In [41]:
```
df = df.dropna(subset=["CatCol1", "CatCol2"], how='all')
df
```

Out[41]:

	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
1	A	Y	1.945879	28.0	0.936825
2	C	X	0.988834	42.0	0.751516
3	A	Y	-0.181978	73.0	0.950696
4	B	X	2.080615	74.0	0.903045
...	...	...	...	...	...
9995	C	Y	1.352114	61.0	0.728445
9996	C	Y	1.143642	67.0	0.605930
9997	A	X	-0.665794	54.0	0.071041
9998	C	Y	0.004278	NaN	NaN
9999	A	X	0.622473	95.0	0.751384

9995 rows × 5 columns

In [42]:
```
df = df.dropna(subset=["NumCol1", "NumCol2", "NumCol3"], how='all')
df
```

Out[42]:

	CatCol1	CatCol2	NumCol1	NumCol2	NumCol3
0	A	NaN	0.440877	49.0	0.246007
1	A	Y	1.945879	28.0	0.936825
2	C	X	0.988834	42.0	0.751516
3	A	Y	-0.181978	73.0	0.950696
4	B	X	2.080615	74.0	0.903045
...	...	...	...	...	...
9995	C	Y	1.352114	61.0	0.728445
9996	C	Y	1.143642	67.0	0.605930
9997	A	X	-0.665794	54.0	0.071041
9998	C	Y	0.004278	NaN	NaN
9999	A	X	0.622473	95.0	0.751384

9975 rows × 5 columns

In [43]:
```
numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])
df.loc[:, categorical_cols.columns] = df[categorical_cols.columns].fillna(df[categorical
conditional_means = df[categorical_cols.columns].fillna(df.groupby('CatCol1')[numeric_co
df.loc[:, numeric_cols.columns] = conditional_means
print(df)
```

```
     CatCol1 CatCol2  NumCol1  NumCol2  NumCol3
0          A       Y      NaN      NaN      NaN
1          A       Y      NaN      NaN      NaN
2          C       X      NaN      NaN      NaN
3          A       Y      NaN      NaN      NaN
4          B       X      NaN      NaN      NaN
...      ...     ...      ...      ...      ...
9995       C       Y      NaN      NaN      NaN
9996       C       Y      NaN      NaN      NaN
9997       A       X      NaN      NaN      NaN
9998       C       Y      NaN      NaN      NaN
9999       A       X      NaN      NaN      NaN

[9975 rows x 5 columns]
```

In [ ]: