MISSING VALUES + IMPORTO DATI

ESERCIZIO 1 (DATA FRAME)

Ho usato la libreria matplotlib(serve per i grafici) e pandas(libreria per ordinare)

In [1]:
```
# Importazione della libreria pandas con l'alias 'pd'
import pandas as pd

# Definizione di un dataset in formato di lista di dizionari
# per il dataframe si usano le parentesi graffe
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},
]

# Creazione di un DataFrame utilizzando il dataset
df = pd.DataFrame(dataset)

# Il DataFrame 'df' è ora pronto per l'analisi e la manipolazione dei dati
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

LI FACCIO SCRVERE IL DATAFRAME DEL PUNTEGGIO

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
```

LI FACCIO SCRIVERE IL DATAFREME DELL'ETà

In [3]:
```
df["età"]
```

Out[3]:
```
0    25.0
1     NaN
2    28.0
3     NaN
4    23.0
5    23.0
Name: età, dtype: float64
```

AXIS ELIMINA I DATI MANCANTI

In [4]:
```
#axis 1 si usa per le righe e axis 0 per le colonne
righe_con_dati_mancanti =df[df.isnull().any(axis=1)]
righe_con_dati_mancanti
```

Out[4]:

	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

DIMMI QUELE COLONNE HANNO I DATI MANCANTI

In [5]:
```
#shape=forma shape è una funzione che ti dice quanti colonne e righe ci sono, shape[0]è per le colonne invece shape[1]
#sono per le righe
totale_dati_mancanti=righe_con_dati_mancanti.shape[0]
totale_dati_mancanti
```

Out[5]:5

PRINTAMI IL TITOLO E IL NUMERO DELLE COLONNE CON NUMERI MANCANTI

In [6]:
```
# metto il titolo
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
```

ESERCIZIO 2 (DATAFRAME CON L'EMAIL)

In [7]:
```
# Importazione della libreria pandas con l'alias 'pd'
import pandas as pd

# Definizione del dataset come lista di dizionari
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"},
]

# Creazione di un DataFrame utilizzando il dataset
df = pd.DataFrame(dataset)

# Il DataFrame 'df' è ora pronto per l'analisi e la manipolazione dei dati
df
```

Out[7]:

	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

RIMUOVI IL DATASET CON IL DATO MANCANTE

In [8]:
```
#rimuovi le righe con dati mancanti
#[inplace False] crei una nuova tabella con le righe con i codici completi,
#[inplace True] è quando rimuovi la vecchia tabella è togli le righe con i codici incomplete
df1=df.dropna(inplace=False)
df1
```

Out[8]:

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

ESERCIZIO 3 (CON VARIABILI)

In [9]:
```
# Importazione della libreria pandas con l'alias 'pd'
import pandas as pd
# Libreria Pandas è utilizzata per manipolare e analizzare dati in formato tabellare (DataFrames)
# Importazione della libreria seaborn con l'alias 'sns'
import seaborn as sns
# Libreria Seaborn offre funzioni per la visualizzazione statistica più attraente rispetto a Matplotlib
# Importazione della libreria numpy con l'alias 'np'
import numpy as np
# Libreria NumPy è utilizzata per operazioni matematiche avanzate, gestione di array e matrici
# Importazione della libreria matplotlib.pyplot con l'alias 'plt'
import matplotlib.pyplot as plt
# Libreria Matplotlib è utilizzata per la visualizzazione dei dati, grafici e plotting


# Creazione di un dizionario con dati di esempio
data = {
    'Variable1': [1, 2, 3, 4, 5],
    'Variable2': [1, 2, np.nan, 4, np.nan],  # Utilizzo di np.nan per rappresentare valori mancanti
```

```
    'Missing_Column': ['A', 'B', 'A', 'C', np.nan]  # Alcuni valori mancanti rappresentati con np.nan
}

# Creazione di un DataFrame utilizzando il dizionario
df = pd.DataFrame(data)

# Creazione di un secondo DataFrame vuoto
df1 = pd.DataFrame()

# Il DataFrame 'df' contiene ora i dati forniti nel dizionario
# Il DataFrame 'df1' è attualmente vuoto
df
```

Out[9]:

	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 [10]:
```
#mi da solo la tebella vecchia con i numeri
numeric_cols = df.select_dtypes(include=['number'])
#mi da solo il titolo con la parte dei numeri
numeric_cols.columns
```

Out[10]:Index(['Variable1', 'Variable2'], dtype='object')

CI DICE QUALE DEI DATASET NON HA IL NUMERO

In [11]:
```
#mi da solo parti numerici
#fillana riempi i parti mancanti con la media delle colonne
df1[numeric_cols.columns]=df[numeric_cols.columns].fillna(df[numeric_cols.columns].mean)
df1
```

Out[11]:

	Variable1	Variable2
0	1	1.0
1	2	2.0
2	3	<bound method NDFrame._add_numeric_operations....
3	4	4.0
4	5	<bound method NDFrame._add_numeric_operations....

AGGIUNGIAMO DEI MISSING COLUMS

In [12]:
```
categorical_cols = df.select_dtypes(exclude=['number'])
categorical_cols.columns
```

Out[12]:Index(['Missing_Column'], dtype='object')

AGGIUNGI I COLUMS NEL DATASET

In [13]:
```
#iloc lavora sui indici
df1[categorical_cols.columns]=df[categorical_cols.columns].fillna(df[categorical_cols.columns].mode().iloc[0])
df1
```

Out[13]:

	Variable1	Variable2	Missing_Column
0	1	1.0	A
1	2	2.0	B
2	3	<bound method NDFrame._add_numeric_operations....	A
3	4	4.0	C
4	5	<bound method NDFrame._add_numeric_operations....	A

CI CREA DEI DATI CON LE FEATURE

In [14]:
```
# Importa le librerie necessarie
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 utilizzando il dizionario 'data'
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

CI DICE SE IL NEMERO ESISTO O NEN C'è CON FALSE (HA I DATI) TRUE (NON C'è L'HA)

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

Out[15]:

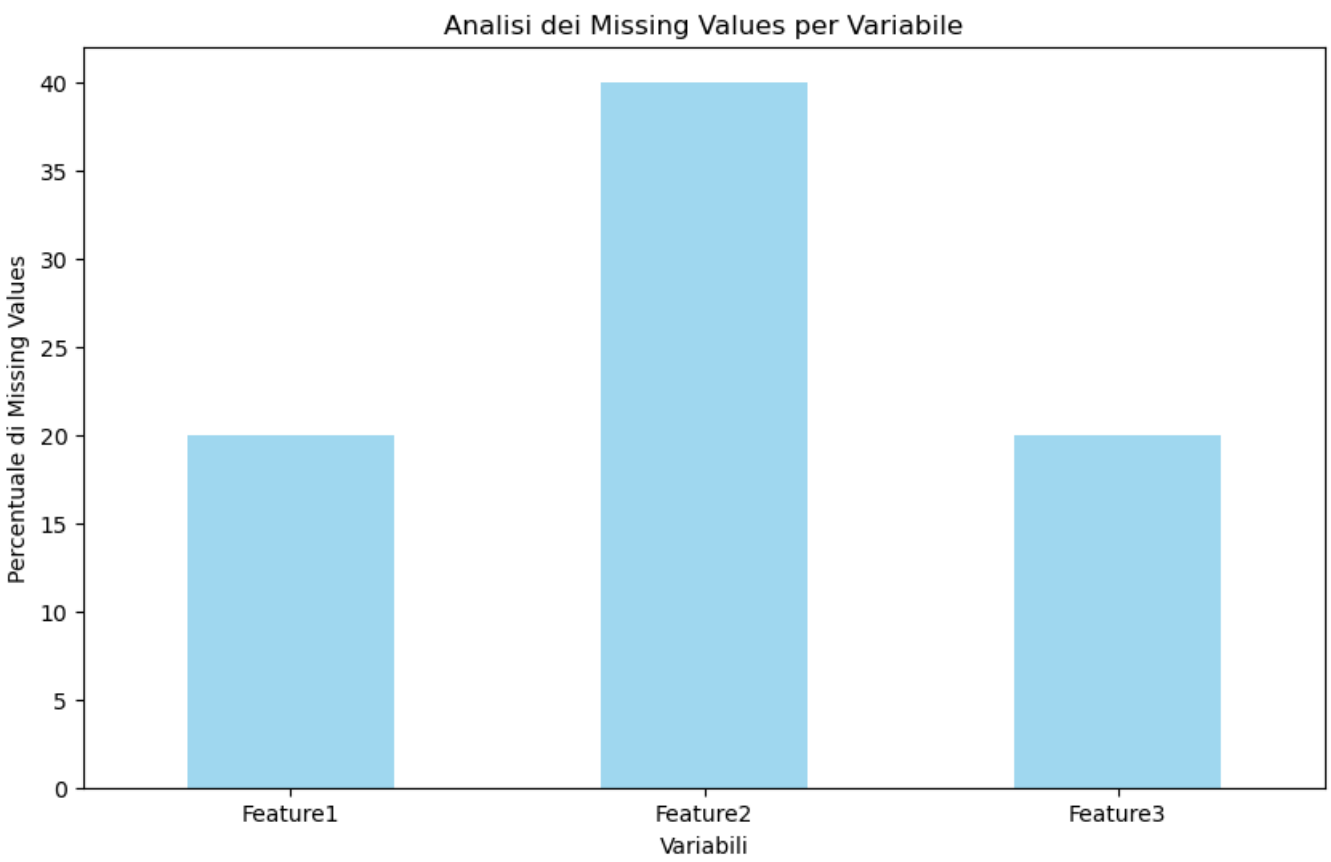
	Feature1	Feature2	Feature3
0	False	True	False
1	False	False	True
2	True	False	False
3	False	False	False
4	False	True	False

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

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

CREA UN GRAFICO

In [17]:
```
# Calcola la percentuale di valori mancanti per ciascuna variabile
missing_percent = (df.isnull().sum() / len(df)) * 100

# Crea il grafico a barre
plt.figure(figsize=(10, 6))  # Imposta le dimensioni della figura
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)  # Crea il grafico a barre con colore e trasparenza specificati
plt.xlabel('Variabili')  # Etichetta sull'asse x
plt.ylabel('Percentuale di Missing Values')  # Etichetta sull'asse y
plt.title('Analisi dei Missing Values per Variabile')  # Titolo del grafico
plt.xticks(rotation=0)  # Imposta la rotazione degli etichette sull'asse x
plt.show()  # Mostra il grafico
```



ESERCIZIO 4 (GENERIAMO UN NUOVO DATASET)

In [18]:
```
# metti la libreria
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()
df
```

Out[18]:

	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

IL GRAFICO QUI SOTTO CI DICE IN QUALI POSIZIONE C'è LA DATA E IN QUALE NON C'è LA DATA(PARTE IN GIALLO)

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

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

ESERCIZIO 5 (GENERIAMO UN NUOVO DATASET)

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),
     #random normal esce il numero con la media più alta, invece quella più bassa ha poca possibilità di uscire
    '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)
#Questo stampa informazioni informative sulla struttura del DataFrame df. Include il numero
#totale di righe, il tipo di dati di ciascuna colonna, e il conteggio dei valori non nulli per
#ogni colonna. Questa è utile per avere una
#panoramica rapida dei dati, compresa la presenza di valori nulli.
print(df.describe())
#Questo stampa le statistiche descrittive del DataFrame df. Per colonne numeriche, restituisce la conta, la media,
#la deviazione standard, il valore minimo, i quartili e il valore massimo.
#Queste statistiche offrono un'idea della distribuzione dei dati numerici nel DataFrame.
```

```
<bound method DataFrame.info of      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
..   ...      ...        ...           ...
995   60  Femmina   1.260752  28916.494712
996   64  Maschio  55.975568  50234.275009
997   62  Femmina  52.739962  63534.157607
998   35  Maschio  71.935362  36383.657298
999   55  Maschio  89.025805  72789.258855

[1000 rows x 4 columns]>
              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
```

CI DA QUALI DEI VALORI è MANCANTE

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

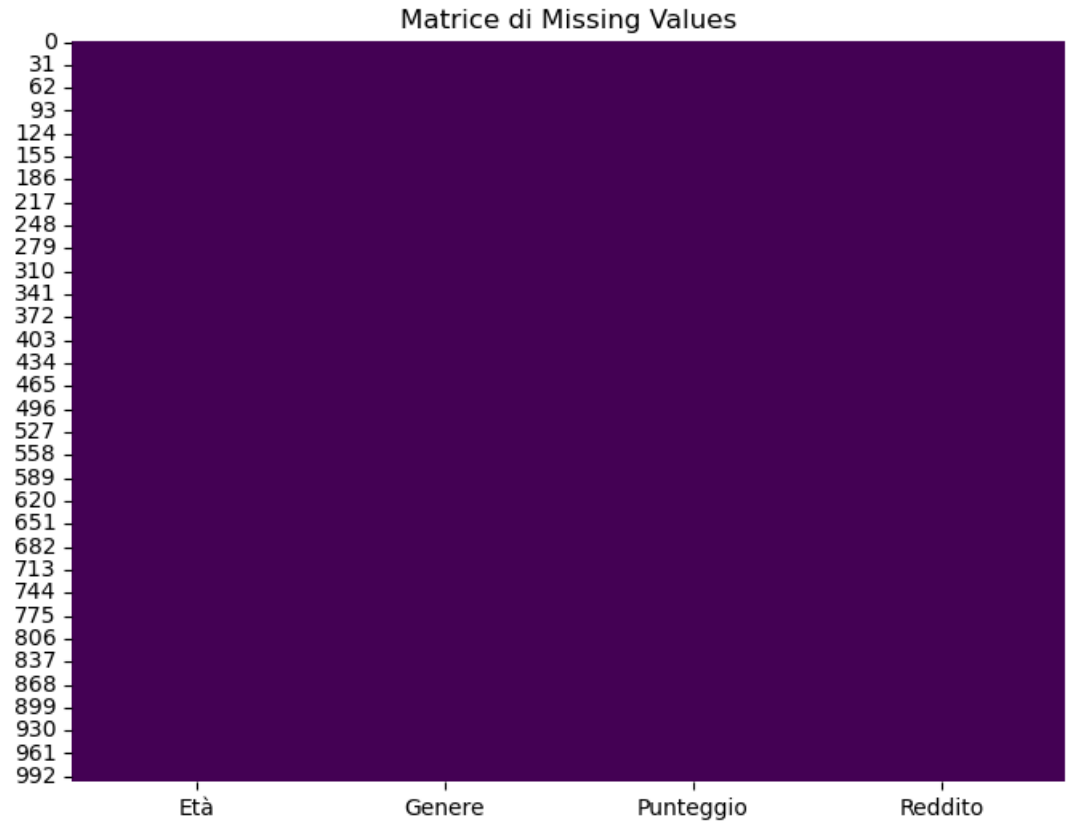
```
Valori mancanti per ciascuna colonna
Età          0
Genere       0
Punteggio    0
Reddito      0
dtype: int64
```

COMINCIAMO CON I DATASET CON IL GRAFICO

CI DA UN CODICE VUOTO PERCHè NEL CODICE PERCEDENTE è TUTTO 0

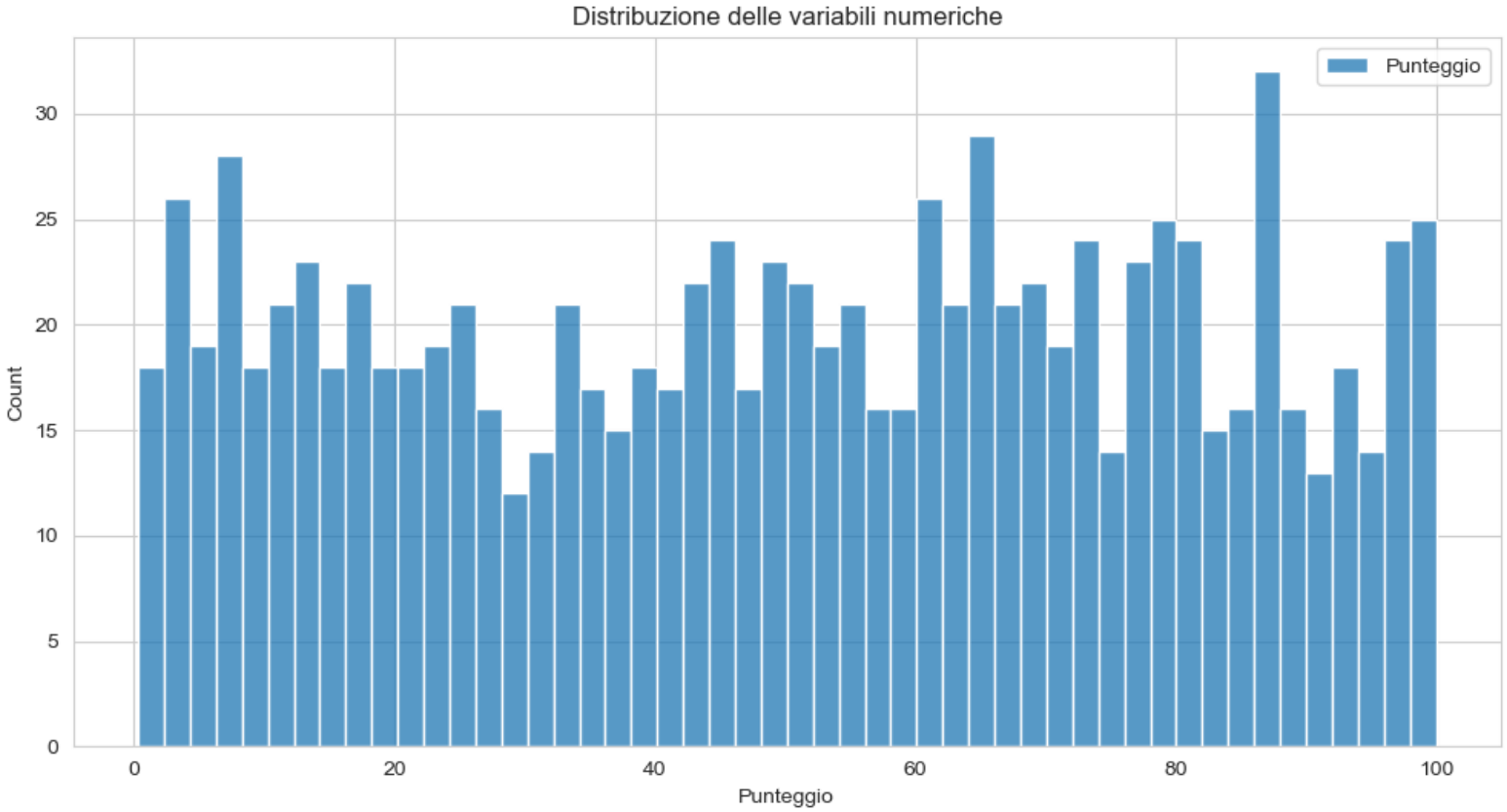
In [23]:
```
# Visualizza una heatmap dei valori mancanti
plt.figure(figsize=(8, 6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False,)
plt.title('Matrice di Missing Values')
plt.show
```

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



CREIAMO UN NUOVO DATASET CON IL GRAFICO

In [24]:
```
# Importa le librerie necessarie
import matplotlib.pyplot as plt  # Per la creazione di grafici
import seaborn as sns  # Per migliorare l'estetica dei grafici

# Crea una nuova figura di dimensioni 12x6 pollici
plt.figure(figsize=(12, 6))

# Imposta lo stile dello sfondo del grafico a "whitegrid" per una migliore leggibilità
sns.set_style("whitegrid")

# Crea un istogramma della colonna "Punteggio" nel DataFrame "df"
# kde=False indica che non si desidera visualizzare la stima della densità di probabilità
# bins=50 specifica il numero di intervalli o "barre" nell'istogramma
# label="Punteggio" fornisce una legenda per il grafico
sns.histplot(df["Punteggio"], kde=False, bins=50, label="Punteggio")

# Aggiunge la legenda al grafico
plt.legend()

# Aggiunge un titolo al grafico
plt.title('Distribuzione delle variabili numeriche')

# Mostra il grafico
plt.show()
```



CON IL CODICE PRECEDENTE CREIAMO ALTRI 9 GRAFICI

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
```

Out[25]:<function matplotlib.pyplot.show(close=None, block=None)>

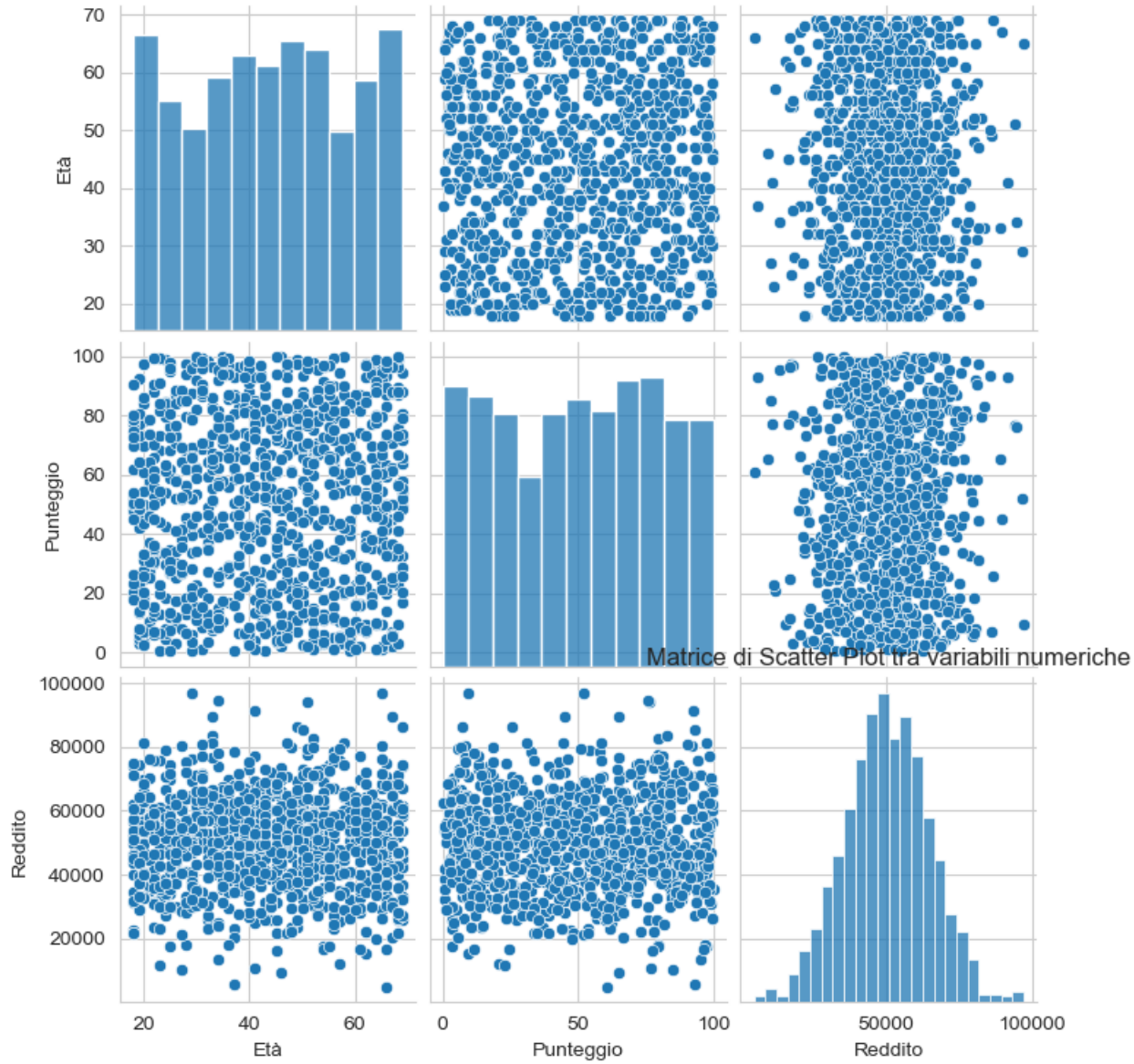
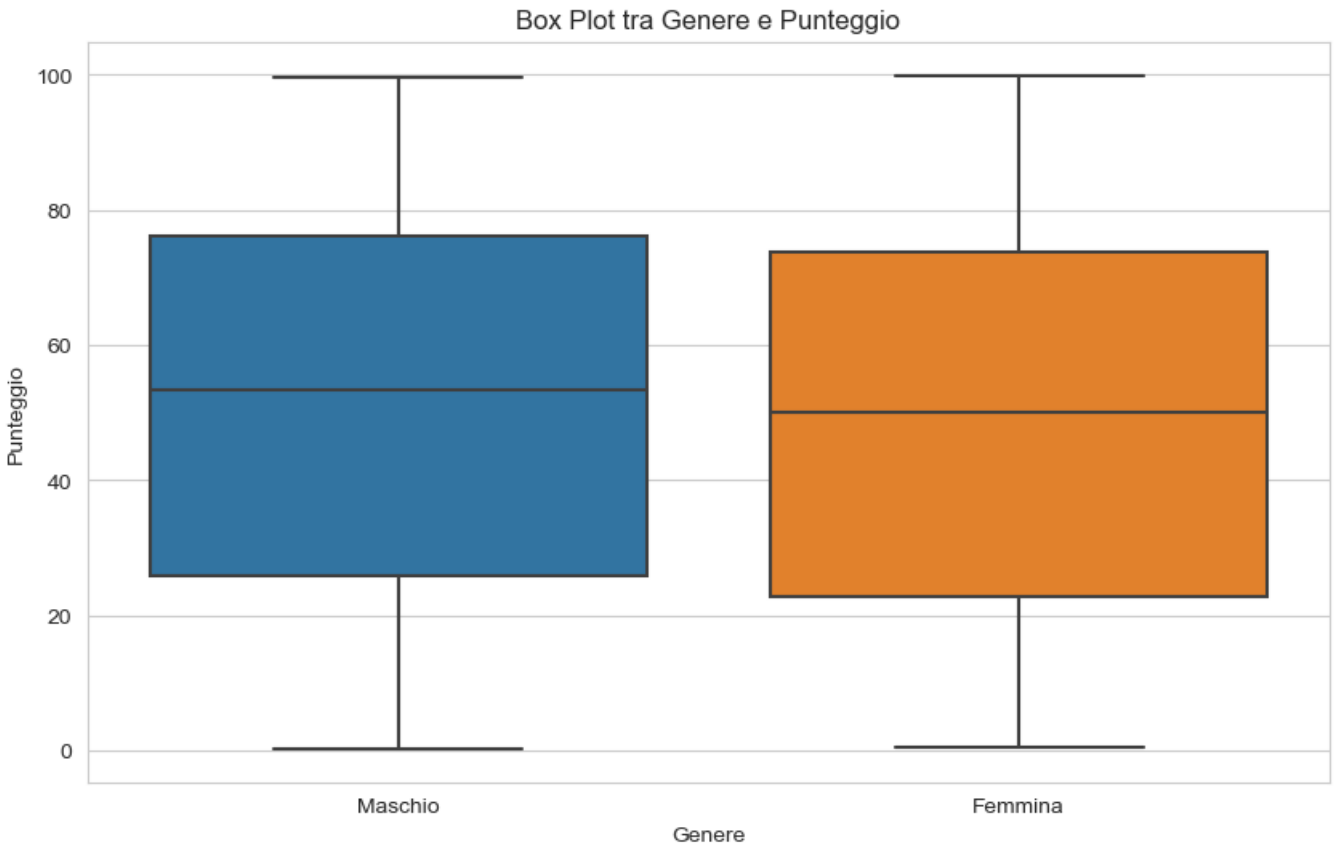


GRAFICO CON IL GENERE E IL PUNTEGGIO

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()
```



CODICI CON TANTI INFORMAZIONI QUINDI LAGGA

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()
```

CODICE CON DEI DATI A CASO

In [28]:
```
# Importa le librerie necessarie
import pandas as pd  # Per la manipolazione dei dati attraverso DataFrame
import numpy as np   # Per la generazione di dati casuali
import matplotlib.pyplot as plt  # Per la creazione di grafici
import seaborn as sns  # Per migliorare l'estetica dei grafici

# Imposta un seed per garantire la riproducibilità dei dati casuali
np.random.seed(42)
```

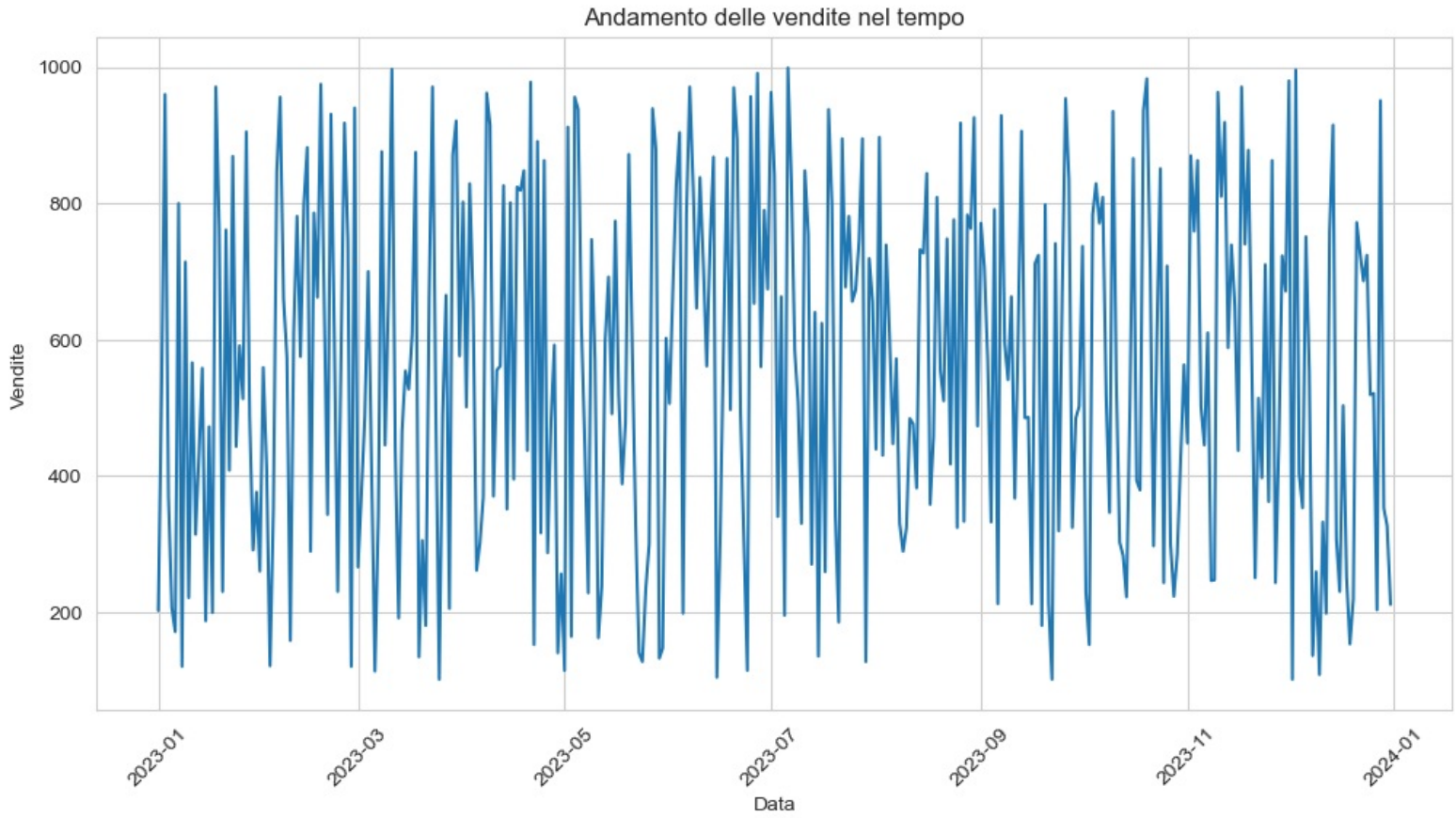
```
# Crea un dizionario con dati casuali
data = {
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31', freq='D'),  # Creazione di date giornaliere
    'Vendite': np.random.randint(100, 1000, size=365),  # Numeri casuali per le vendite
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365)  # Scelta casuale tra i prodotti A, B, C
}

# Crea un DataFrame utilizzando il dizionario di dati
df = pd.DataFrame(data)

# Stampa le prime righe del DataFrame per visualizzare i dati iniziali
print(df.head())
```

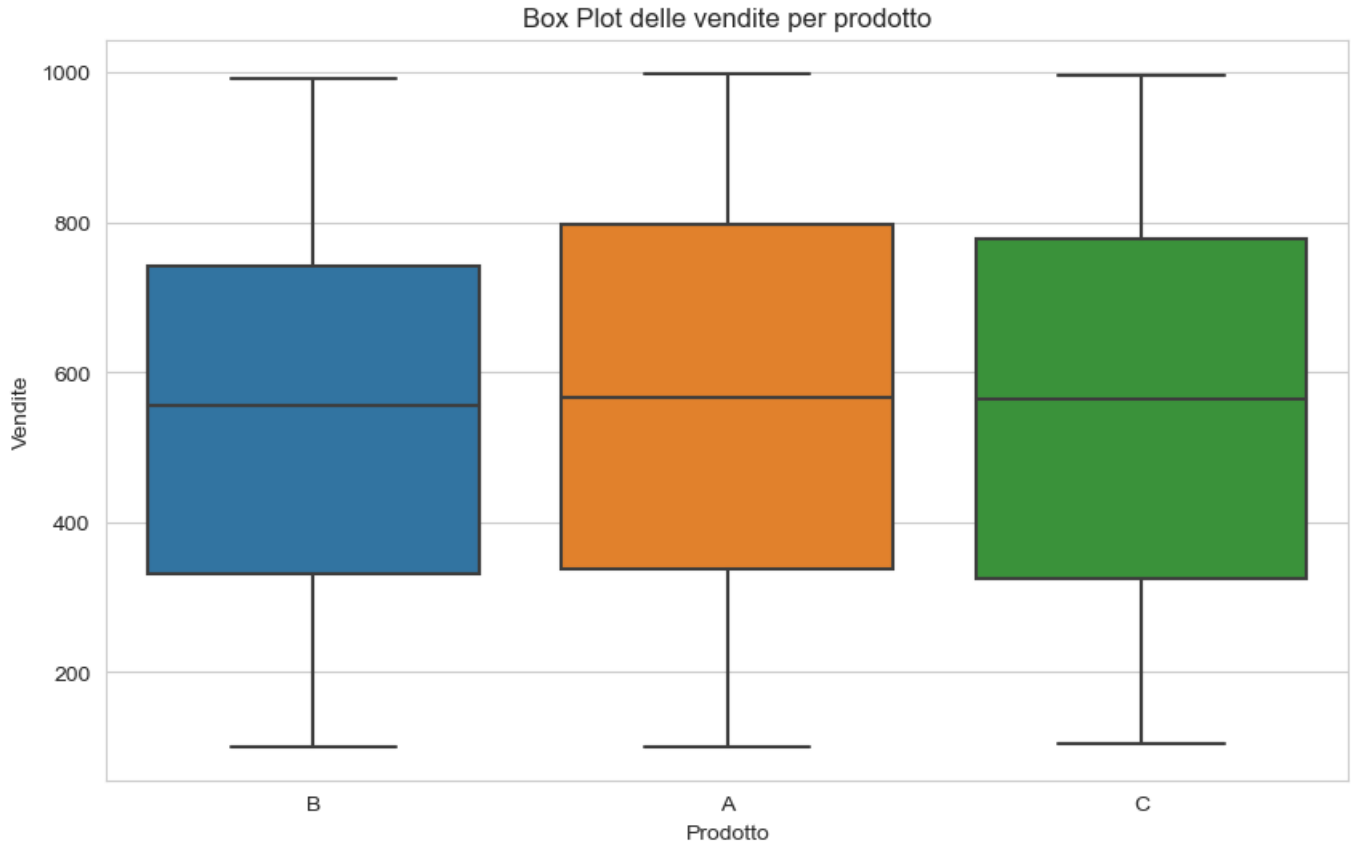
```
        Data  Vendite Prodotto
0 2023-01-01      202        B
1 2023-01-02      535        A
2 2023-01-03      960        C
3 2023-01-04      370        A
4 2023-01-05      206        A
```

CREA DI NUOVO DEI DATI NUOVI MA QUESTA VOLTA CON UN GRAFICO A LINEPLOT E BOXPLOT

In [29]:
```
# Importa le librerie necessarie
import pandas as pd  # Per la manipolazione dei dati attraverso DataFrame
import numpy as np  # Per la generazione di dati casuali
import matplotlib.pyplot as plt  # Per la creazione di grafici
import seaborn as sns  # Per migliorare l'estetica dei grafici

# Imposta un seed per garantire la riproducibilità dei dati casuali
np.random.seed(42)

# Crea un dizionario con dati casuali
data = {
    'Data': pd.date_range(start='2023-01-01', end='2023-12-31', freq='D'),  # Creazione di date giornaliere
    'Vendite': np.random.randint(100, 1000, size=365),  # Numeri casuali per le vendite
    'Prodotto': np.random.choice(['A', 'B', 'C'], size=365)  # Scelta casuale tra i prodotti A, B, C
}

# Crea un DataFrame utilizzando il dizionario di dati
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)  # Crea un grafico a linee delle vendite nel tempo
plt.title('Andamento delle vendite nel tempo')
plt.xlabel('Data')
plt.ylabel('Vendite')
plt.xticks(rotation=45)  # Ruota le etichette sull'asse x per una migliore leggibilità
plt.show()

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





ESERCIZIO 6

In [30]:
```
# Importa le librerie necessarie
import pandas as pd  # Per la manipolazione dei dati attraverso DataFrame
import matplotlib.pyplot as plt  # Per la creazione di grafici
import numpy as np  # Per la gestione di dati numerici, inclusi NaN
import seaborn as sns  # Per migliorare l'estetica dei grafici

# Genera dati di esempio
```

```
data = {
   'Numeric_Var': [1, 2, 3, 4, np.nan, 6],  # Variabile numerica con un valore mancante (NaN)
   'Categorical_Var': ['A', 'B', 'A', 'B', 'A', 'B']  # Variabile categorica
}

# Crea un DataFrame utilizzando il dizionario di dati
df = pd.DataFrame(data)

# Stampa il DataFrame per visualizzare i dati di esempio
print(df)

# Ora il DataFrame 'df' contiene due colonne: 'Numeric_Var' e 'Categorical_Var'.
# 'Numeric_Var' contiene valori numerici, incluso un valore mancante (NaN).
# 'Categorical_Var' contiene valori categorici ('A' o 'B').

# Il blocco di stampa consente di esaminare rapidamente il DataFrame risultante.
```

```
  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
```

"Creazione di un DataFrame con Dati di Esempio"

In [31]:
```
import pandas as pd
import 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)
```

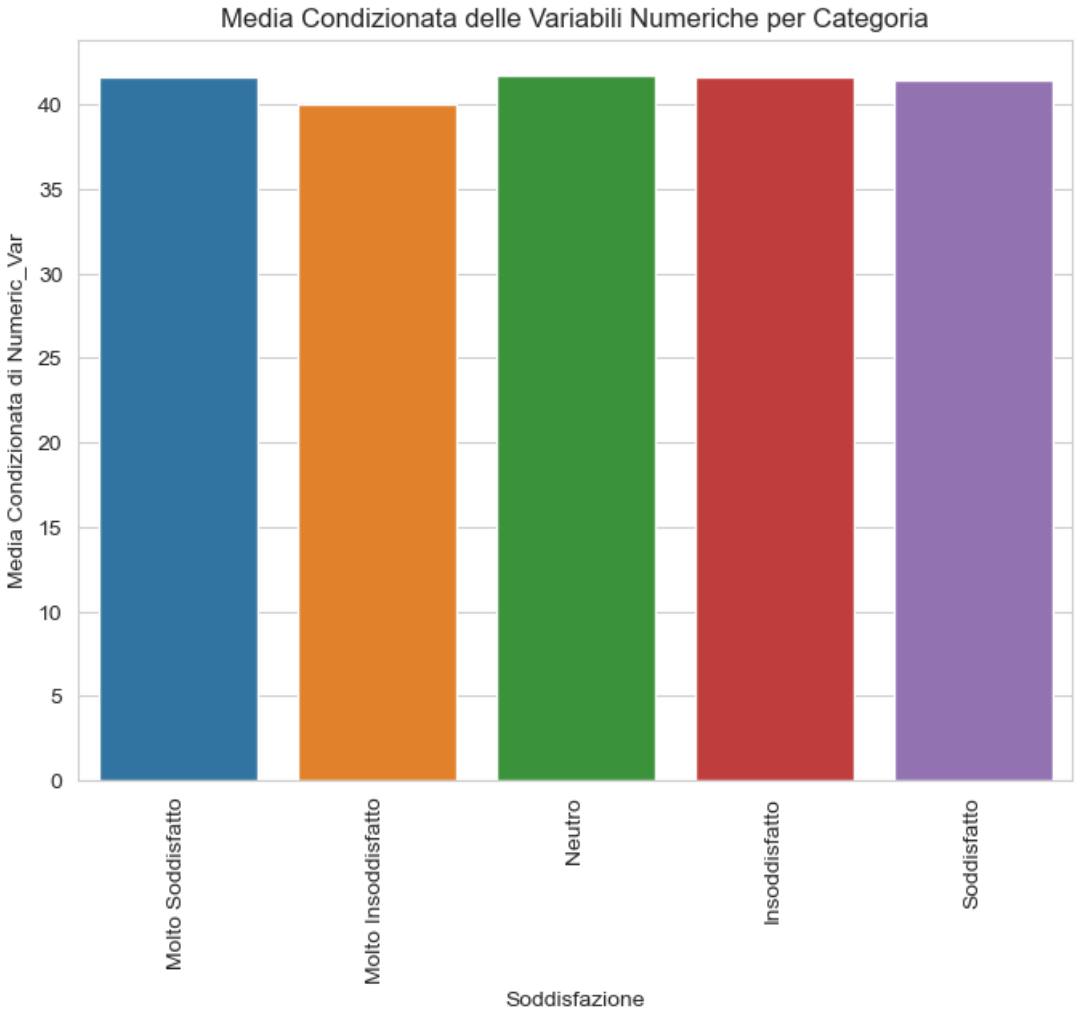
```
  Numeric_Var Categorical_Var
0         1.0              A
1         2.0              B
2         3.0              A
3         4.0              B
4         NaN              A
5         6.0              B
```

"Analisi della Soddisfazione e Media Condizionata delle Variabili Numeriche"

In [32]:
```
# Importa le librerie necessarie
import pandas as pd  # Per la manipolazione dei dati attraverso DataFrame
import numpy as np  # Per la generazione di dati casuali
import matplotlib.pyplot as plt  # Per la creazione di grafici
import seaborn as sns  # Per migliorare l'estetica dei grafici

# Imposta un seed per garantire la riproducibilità dei dati casuali
np.random.seed(42)

# Genera dati casuali per l'esplorazione
data = {
   'Età': np.random.randint(18, 65, size=500),  # Numeri casuali rappresentanti l'età
   'Soddisfazione': np.random.choice(['Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'], size=500)  # Livelli casuali di soc
}

# Crea un DataFrame utilizzando il dizionario di dati
df = pd.DataFrame(data)

# Stampa il DataFrame per visualizzare i dati iniziali
print("DataFrame iniziale:")
print(df)

# Calcola le medie condizionate sulla variabile 'Soddisfazione'
conditional_means = df.groupby('Soddisfazione')['Età'].transform('mean')

# Aggiunge una nuova colonna 'Numeric_Var' contenente le medie condizionate
df['Numeric_Var'] = conditional_means

# Stampa il DataFrame con la nuova colonna 'Numeric_Var'
```

```
print("\nDataFrame con Numeric_Var:")
print(df)

# Crea un grafico a barre per mostrare la media condizionata per ogni categoria di 'Soddisfazione'
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)  # Ruota le etichette sull'asse x per una migliore leggibilità

# Mostra il grafico
plt.show()
```

```
DataFrame iniziale:
    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]

DataFrame con Numeric_Var:
    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\enric\AppData\Local\Temp\ipykernel_5880\885402667.py:35: FutureWarning:


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



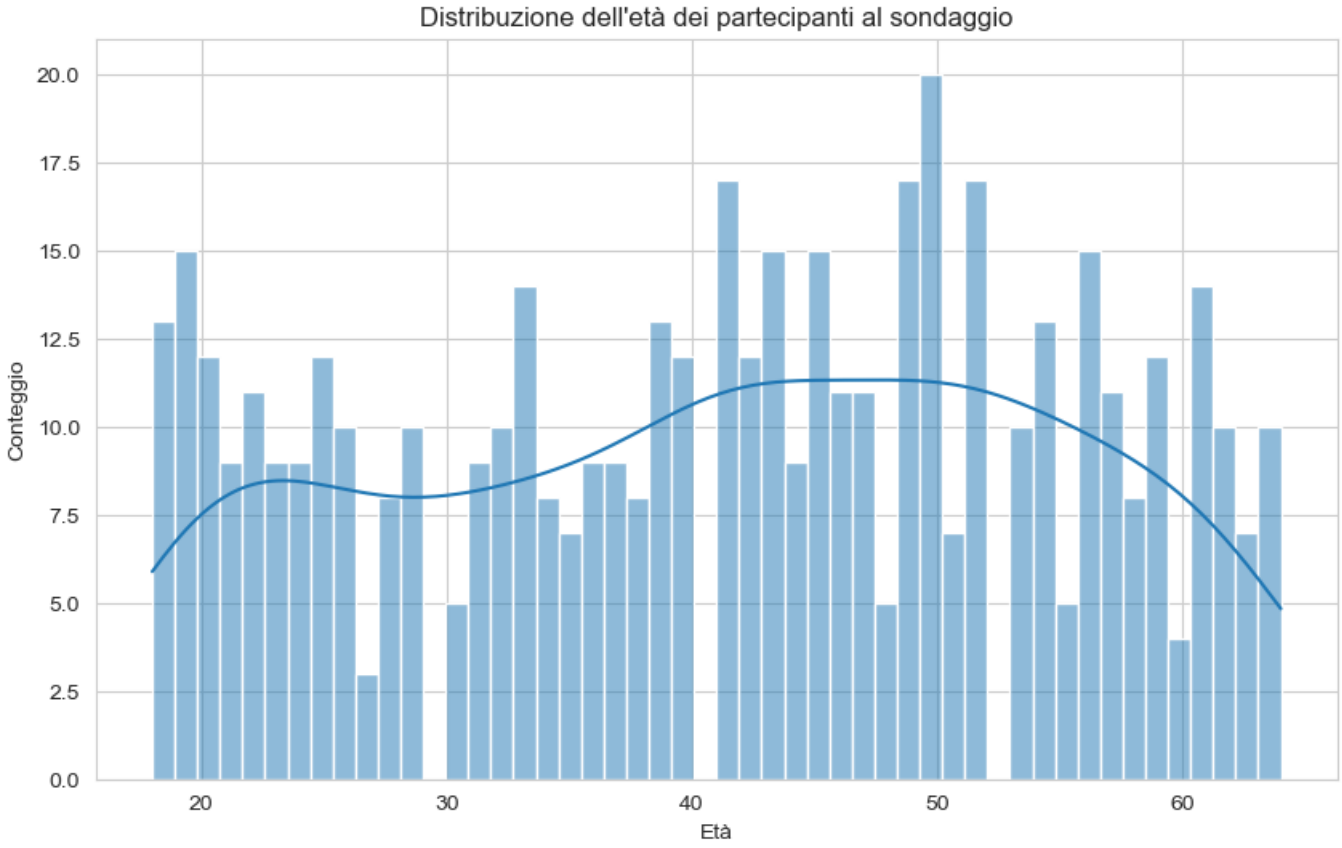
"Analisi Esplorativa delle Variabili nel DataFrame"

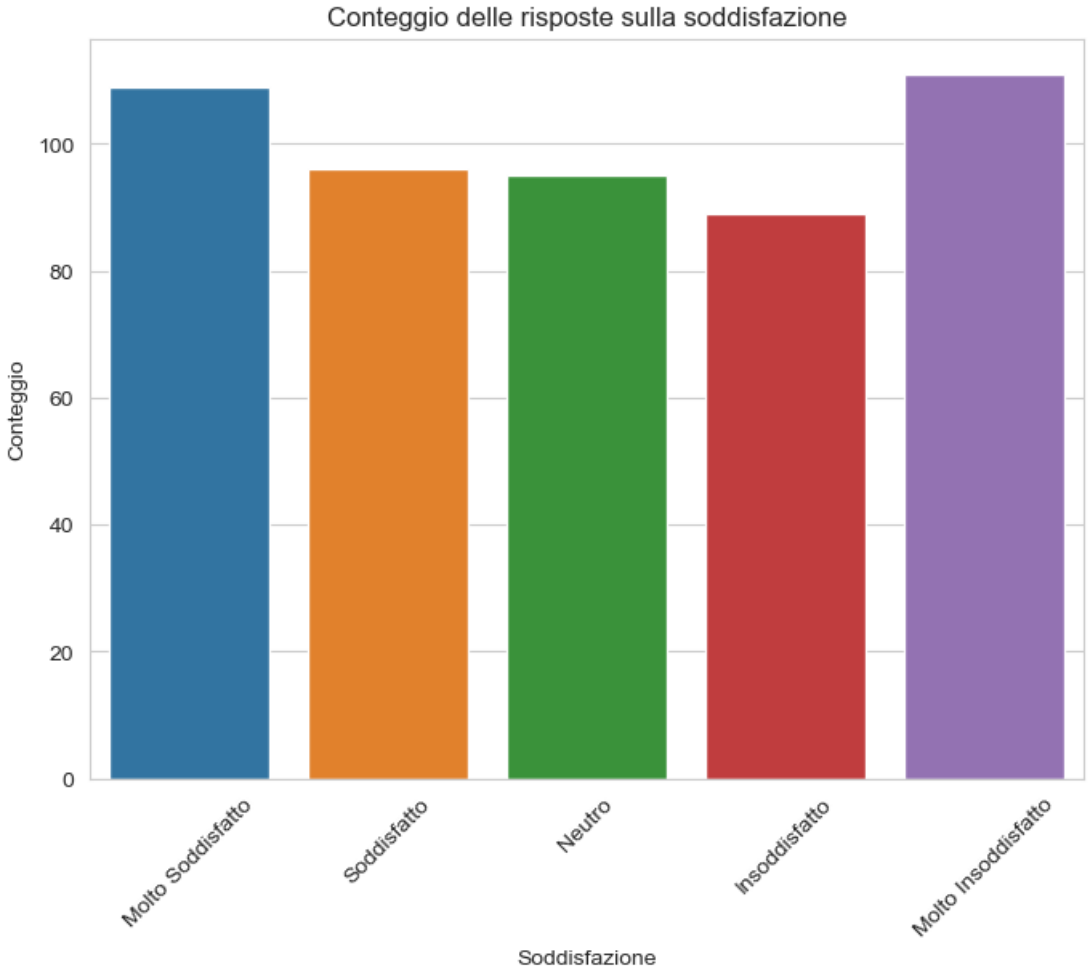
In [33]:
```
# Stampa le prime righe del DataFrame per visualizzare i dati
print("Prime righe del DataFrame:")
```

```
print(df.head())

# Visualizza una distribuzione dell'età attraverso un istogramma
plt.figure(figsize=(10, 6))
sns.histplot(df['Età'], bins=50, kde=True)  # Utilizza Seaborn per creare un istogramma con KDE sovrapposto
plt.title('Distribuzione dell\'età dei partecipanti al sondaggio')  # Aggiunge un titolo al grafico
plt.xlabel('Età')  # Etichetta l'asse x con 'Età'
plt.ylabel('Conteggio')  # Etichetta l'asse y con 'Conteggio'
plt.show()

# Visualizza un conteggio delle risposte sulla soddisfazione attraverso un countplot
plt.figure(figsize=(8, 6))
sns.countplot(x='Soddisfazione', data=df, order=['Molto Soddisfatto', 'Soddisfatto', 'Neutro', 'Insoddisfatto', 'Molto Insoddisfatto'])
# Utilizza Seaborn per creare un countplot con ordine specificato delle categorie
plt.title('Conteggio delle risposte sulla soddisfazione')  # Aggiunge un titolo al grafico
plt.xlabel('Soddisfazione')  # Etichetta l'asse x con 'Soddisfazione'
plt.ylabel('Conteggio')  # Etichetta l'asse y con 'Conteggio'
plt.xticks(rotation=45)  # Ruota le etichette sull'asse x per una migliore leggibilità
plt.show()
```

Prime righe del DataFrame:

	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



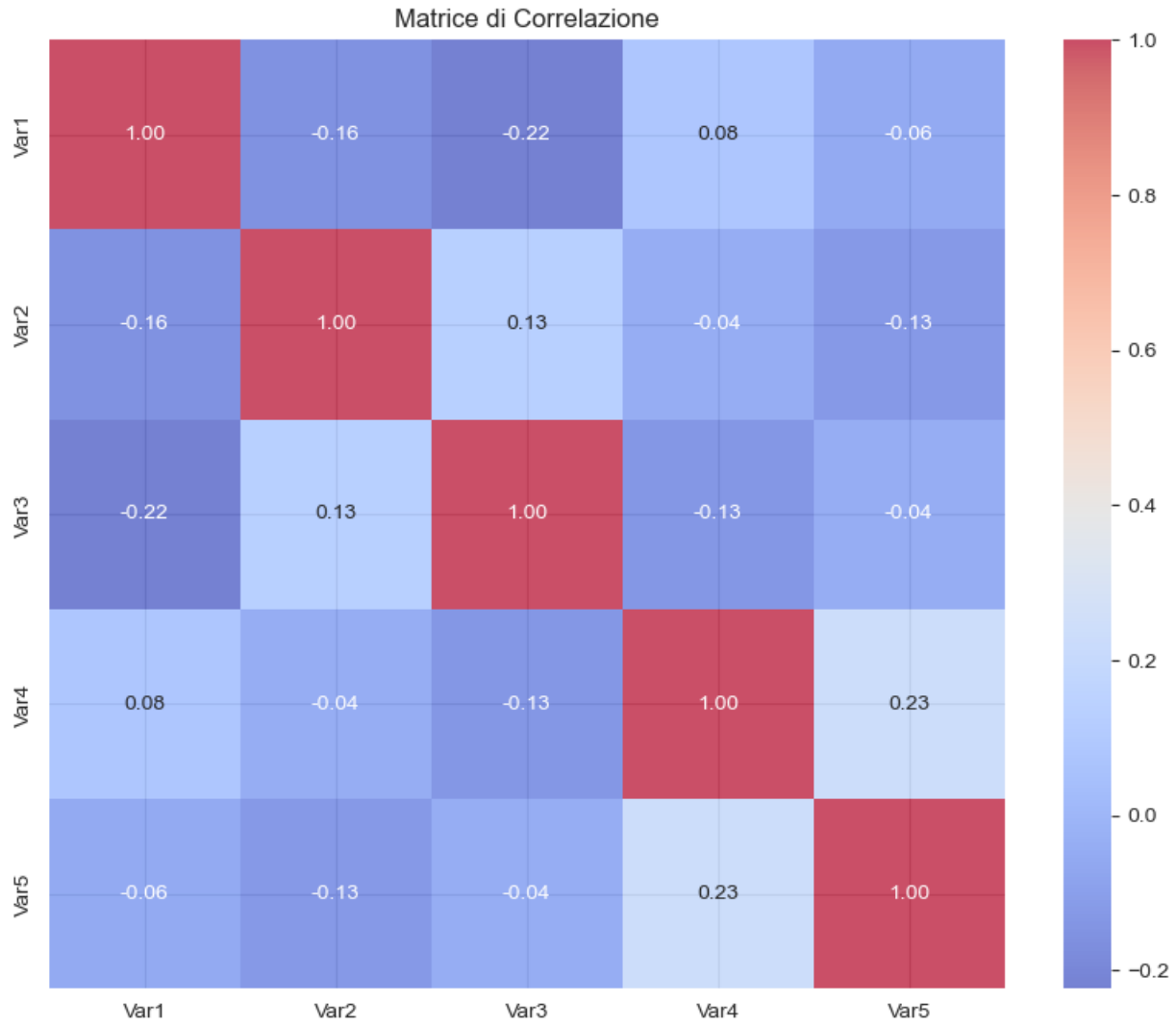


"Visualizzazione della Matrice di Correlazione tra Variabili Numeriche"

In [34]:
```
# Importa le librerie necessarie
import numpy as np  # Per la generazione di dati casuali
import seaborn as sns  # Per migliorare l'estetica dei grafici
```

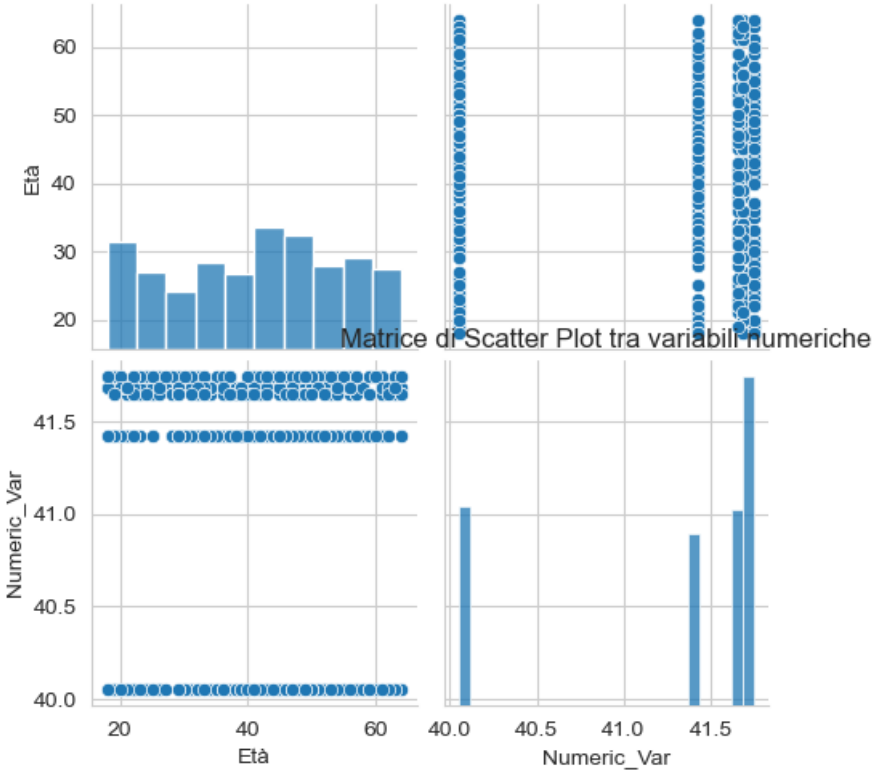
```
import matplotlib.pyplot as plt  # Per la creazione di grafici

# Imposta il seed per garantire la riproducibilità dei dati casuali
np.random.seed(42)

# Genera un dataset di esempio con variabili numeriche
data = pd.DataFrame(np.random.rand(100, 5), columns=['Var1', 'Var2', 'Var3', 'Var4', 'Var5'])

# 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)
# 'annot=True' aggiunge i valori numerici nella heatmap, 'cmap' specifica la mappa di colori
# 'fmt=".2f"' formatta i valori come float con due decimali, 'alpha=0.7' regola la trasparenza della heatmap
plt.title("Matrice di Correlazione")  # Aggiunge un titolo al grafico
plt.show()
```

```
C:\Users\enric\AppData\Local\Temp\ipykernel_5880\1696377528.py:17: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
```



"Analisi degli Scatter Plot tra Variabili Numeriche"

In [35]:
```
# Seleziona solo le colonne numeriche dal DataFrame
numeric_features = df.select_dtypes(include=[np.number])

# Crea un pair plot tra tutte le variabili numeriche
sns.pairplot(df[numeric_features.columns])

# Aggiunge un titolo al pair plot
plt.title('Matrice di Scatter Plot tra variabili numeriche')

# Mostra il pair plot
plt.show()
```



"Generazione di un DataFrame con Dati Casuali e Introduzione di Missing Values"

In [36]:
```
# Importa le librerie necessarie
import pandas as pd  # Per la manipolazione dei dati attraverso DataFrame
import numpy as np   # Per la generazione di dati casuali

# Imposta 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)  # Variabile categorica
df['CatCol2'] = np.random.choice(['X', 'Y'], size=n_rows)  # Altra variabile categorica
df['NumCol1'] = np.random.randn(n_rows)  # Variabile numerica con distribuzione normale
df['NumCol2'] = np.random.randint(1, 100, size=n_rows)  # Variabile numerica intera
df['NumCol3'] = np.random.uniform(0, 1, size=n_rows)  # Variabile numerica uniforme tra 0 e 1

# 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

# Il DataFrame risultante contiene dati casuali con un numero casuale di valori mancanti
print(df)
```

```
     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 x 5 columns]
```

CI DA QUNTI SONO I DATI MANCANTI

In [37]:
```
# Seleziona le righe con almeno un dato mancante
righe_con_dati_mancanti = df[df.isnull().any(axis=1)]
```

```
# Calcola la lunghezza del DataFrame contenente solo le righe con dati mancanti
numero_di_righe_con_dati_mancanti = len(righe_con_dati_mancanti)

# Stampa il numero di righe con dati mancanti
print("Numero di righe con dati mancanti:", numero_di_righe_con_dati_mancanti)
```

Numero di righe con dati mancanti: 3648

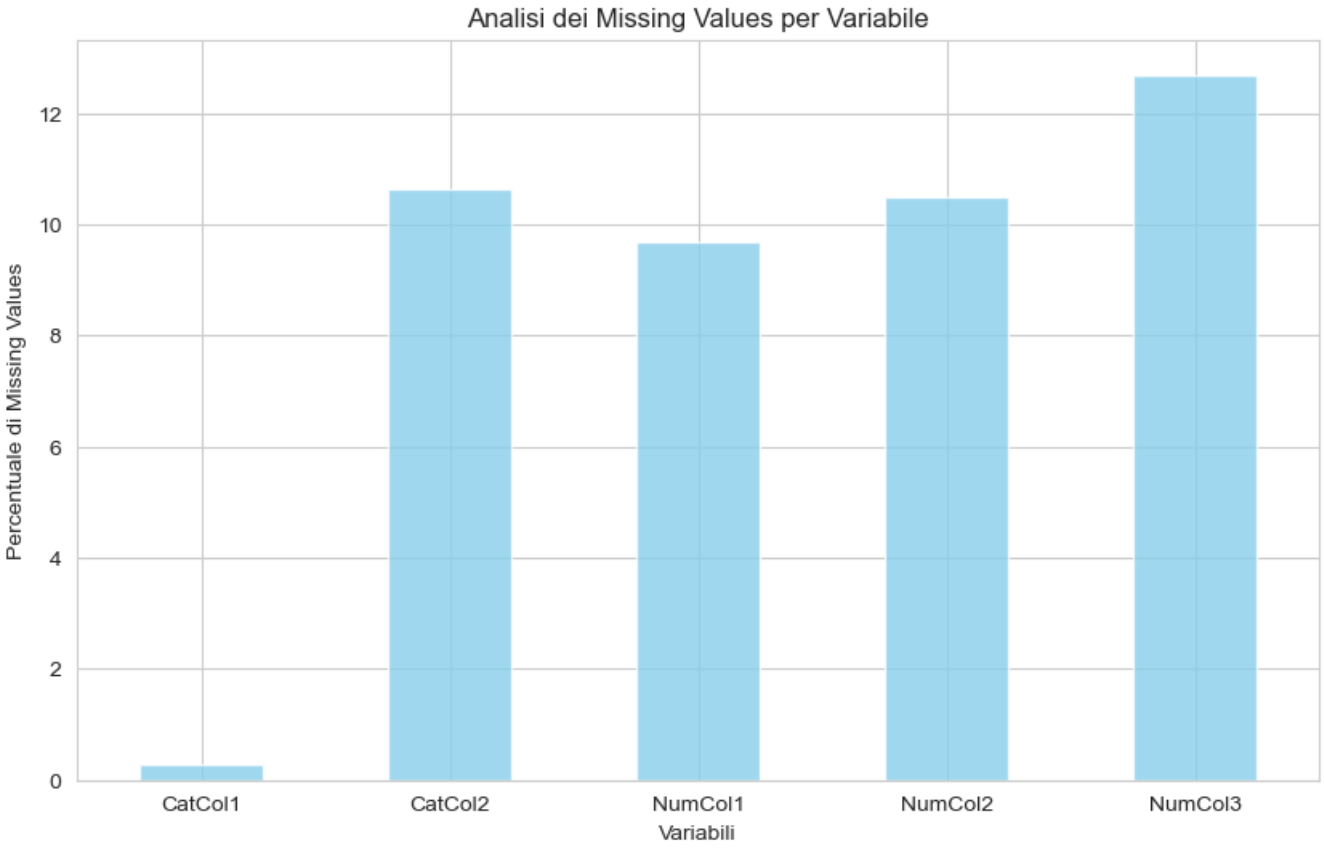
CALCOLI DI QUALCOSA

In [38]:
```
# Calcola la percentuale di valori mancanti per ciascuna colonna
missing_percent = (df.isnull().sum() / len(df)) * 100

# Stampa la percentuale di valori mancanti per ciascuna colonna
print("Percentuale di valori mancanti per ciascuna colonna:")
print(missing_percent)
```

Percentuale di valori mancanti per ciascuna colonna:
CatCol1 0.29
CatCol2 10.63
NumCol1 9.67
NumCol2 10.48
NumCol3 12.69
dtype: float64

"Analisi dei Missing Values per Variabile"

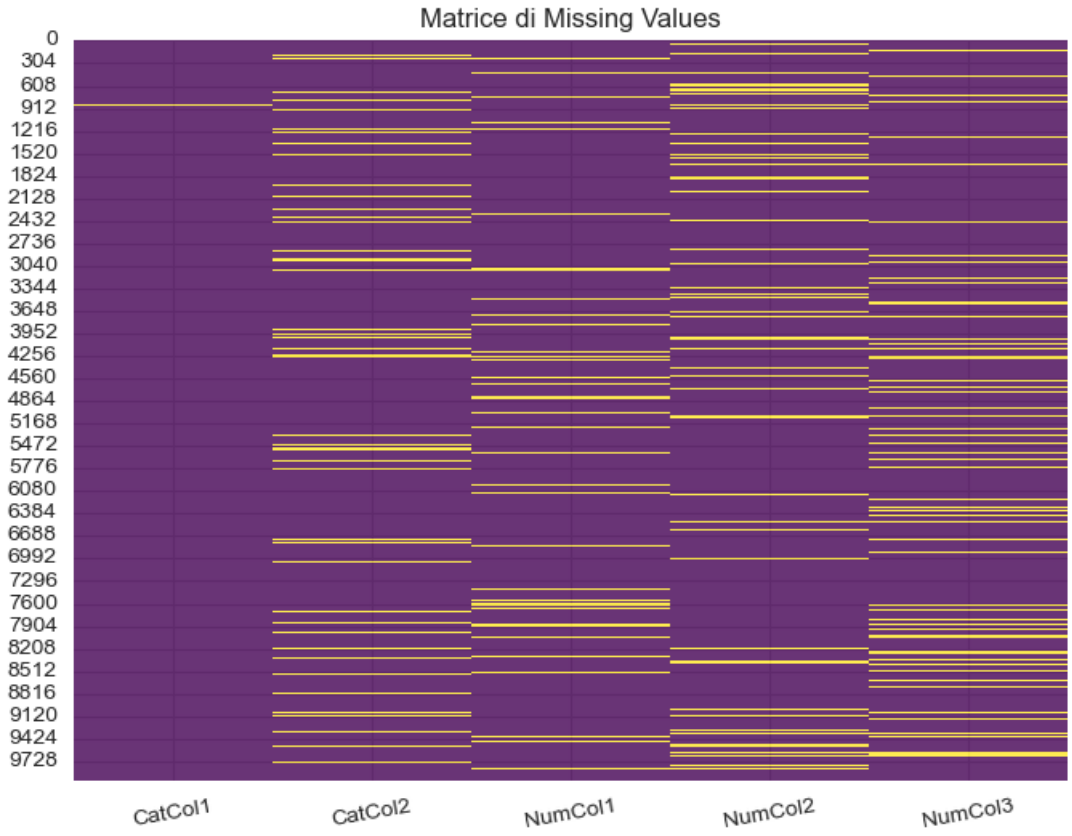
In [39]:
```
# Calcola la percentuale di valori mancanti per ciascuna colonna
missing_percent = (df.isnull().sum() / len(df)) * 100

# Crea il grafico a barre
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)  # Crea un grafico a barre con colorazione e trasparenza specificate
plt.xlabel('Variabili')  # Etichetta l'asse x con 'Variabili'
plt.ylabel('Percentuale di Missing Values')  # Etichetta l'asse y con 'Percentuale di Missing Values'
plt.title('Analisi dei Missing Values per Variabile')  # Aggiunge un titolo al grafico
plt.xticks(rotation=0)  # Imposta l'orientamento delle etichette sull'asse x
plt.show()
```



"Matrice di Missing Values"

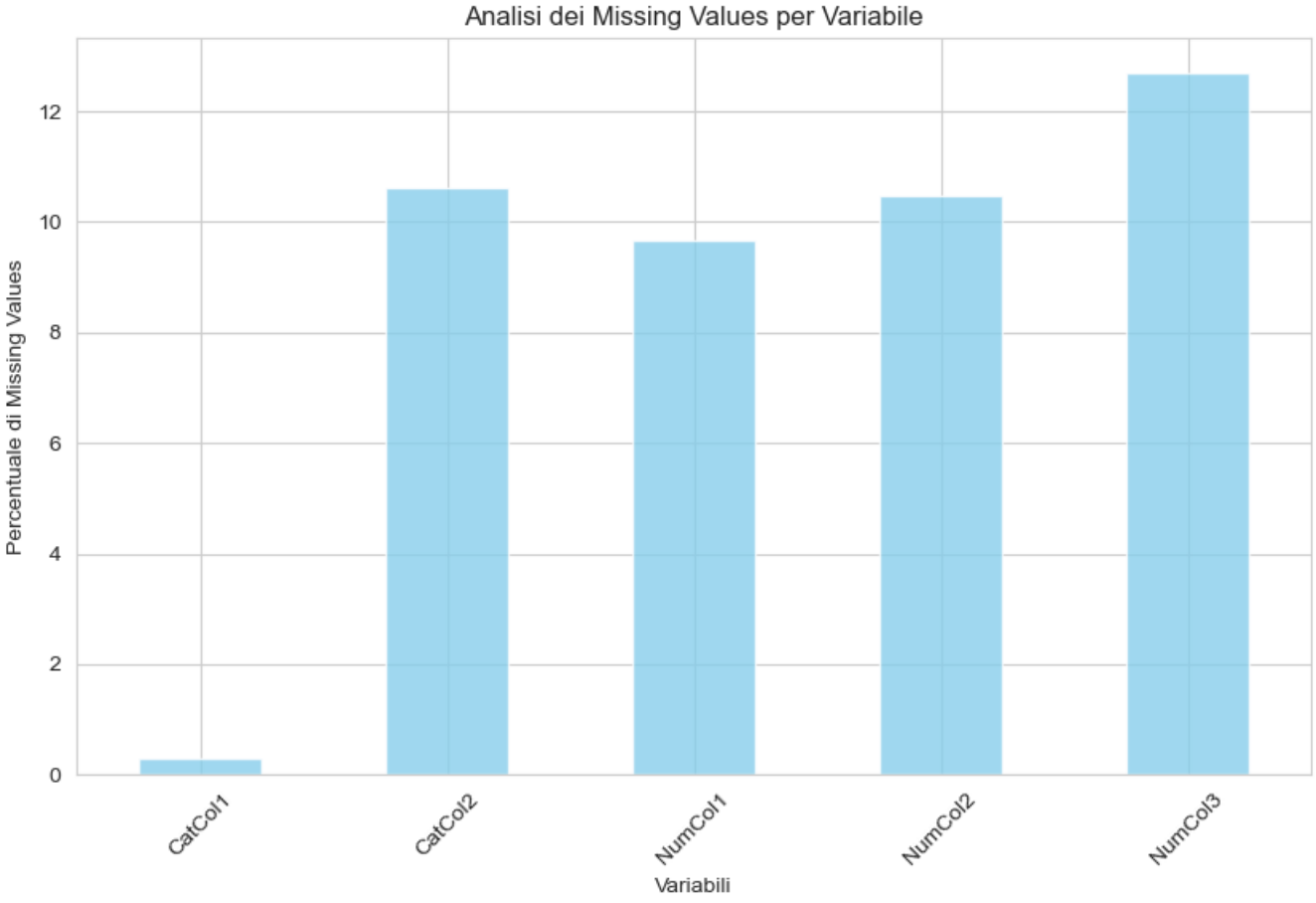
In [40]:
```
# Crea una matrice booleana indicante la presenza di missing values
missing_matrix = df.isnull()

# Crea il grafico a matrice di missing values (heatmap)
plt.figure(figsize=(8, 6))
sns.heatmap(missing_matrix, cmap='viridis', cbar=False, alpha=0.8)  # Crea una heatmap con colorazione specificata e senza barra laterale
plt.title('Matrice di Missing Values')  # Aggiunge un titolo al grafico
plt.xticks(rotation=10)  # Ruota le etichette sull'asse x per una migliore leggibilità
```

```
plt.show()
```



"Analisi dei Missing Values per Variabile"

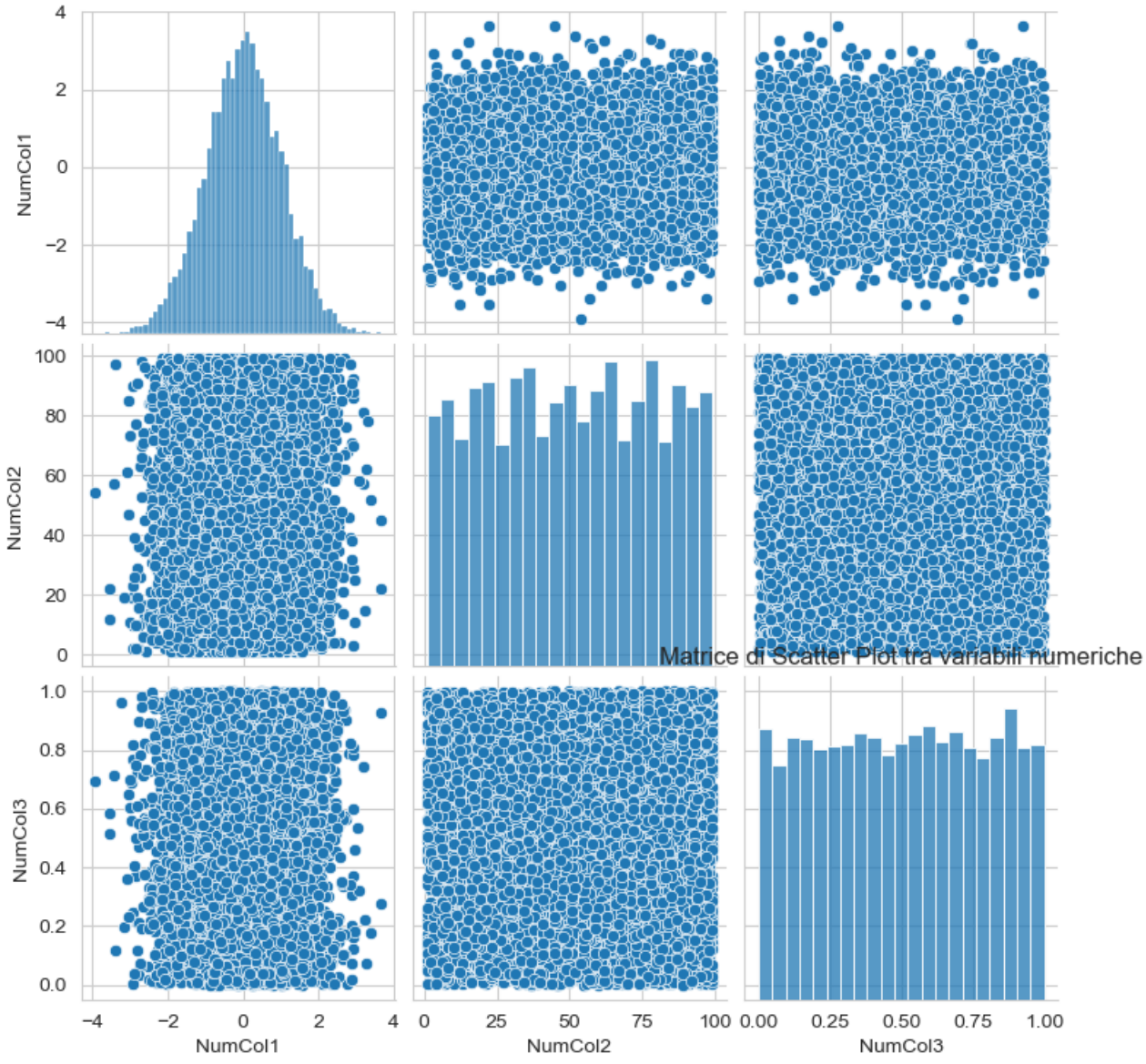
In [41]:
```
# Crea il grafico a barre
plt.figure(figsize=(10, 6))
missing_percent.plot(kind='bar', color='skyblue', alpha=0.8)  # Crea un grafico a barre con colorazione e trasparenza specificate
plt.xlabel('Variabili')  # Etichetta l'asse x con 'Variabili'
plt.ylabel('Percentuale di Missing Values')  # Etichetta l'asse y con 'Percentuale di Missing Values'
plt.title('Analisi dei Missing Values per Variabile')  # Aggiunge un titolo al grafico
plt.xticks(rotation=45)  # Ruota le etichette sull'asse x di 45 gradi per una migliore leggibilità
plt.show()
```



"Scatter Plot Matrix delle Variabili Numeriche"

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
percorso_file_csv = "C"
df = pd.read_csv(percorso_file_csv)
print (df.head())
```

In [42]:
```
# Seleziona solo le colonne numeriche dal DataFrame
numeric_features = df.select_dtypes(include=[np.number])

# Crea un pair plot tra tutte le variabili numeriche
sns.pairplot(df[numeric_features.columns])

# Aggiunge un titolo al pair plot
plt.title('Matrice di Scatter Plot tra variabili numeriche')

# Mostra il pair plot
plt.show()
```



"Rimozione delle Righe con Valori Mancanti in CatCol1 e CatCol2"

In [43]:
```
# Rimuovi le righe in cui entrambe le colonne "CatCol1" e "CatCol2" contengono valori mancanti
df = df.dropna(subset=["CatCol1", 'CatCol2'], how='all')
df
```

Out[43]:

	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

"Rimozione delle Righe con Valori Mancanti nelle Colonne Numeriche"

In [44]:
```
# Rimuovi le righe in cui tutte le colonne numeriche ("NumCol1", "NumCol2", "NumCol3") contengono valori mancanti
df = df.dropna(subset=["NumCol1", "NumCol2", "NumCol3"], how='all')
df
```

Out[44]:

	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

"Imputazione dei Valori Mancanti nelle Colonne Categoriche e Numeriche"

In [45]:
```
# Seleziona le colonne numeriche e categoriche dal DataFrame
numeric_cols = df.select_dtypes(include=['number'])
categorical_cols = df.select_dtypes(exclude=['number'])

# Imputa i valori mancanti nelle colonne categoriche con la moda
df.loc[:, categorical_cols.columns] = df[categorical_cols.columns].fillna(df[categorical_cols.columns].mode().iloc[0])

# Calcola le medie condizionate e imputa i valori mancanti nelle colonne numeriche
conditional_means = df[numeric_cols.columns].fillna(df.groupby('CatCol1')[numeric_cols.columns].transform('mean'))
df.loc[:, numeric_cols.columns] = conditional_means

# Stampa il DataFrame risultante dopo l'imputazione dei valori mancanti
print(df)
```

```
   CatCol1 CatCol2   NumCol1    NumCol2   NumCol3
0        A       Y  0.440877  49.000000  0.246007
1        A       Y  1.945879  28.000000  0.936825
2        C       X  0.988834  42.000000  0.751516
3        A       Y -0.181978  73.000000  0.950696
4        B       X  2.080615  74.000000  0.903045
...    ...     ...       ...        ...       ...
9995     C       Y  1.352114  61.000000  0.728445
9996     C       Y  1.143642  67.000000  0.605930
9997     A       X -0.665794  54.000000  0.071041
9998     C       Y  0.004278  49.845018  0.489352
9999     A       X  0.622473  95.000000  0.751384

[9975 rows x 5 columns]
C:\Users\enric\AppData\Local\Temp\ipykernel_5880\3070183700.py:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\enric\AppData\Local\Temp\ipykernel_5880\3070183700.py:10: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

import pandas as pd

percorso_file_csv = "C:\Users\enric\Desktop\pokemon 1\pokemons.csv" df = pd.read_csv(percorso_file_csv) df

In [62]:
```
# Import delle librerie necessarie
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Definizione del percorso del file CSV
percorso_file_csv = "C:\\Users\\enric\\Desktop\\pokemon 1\\pokemons.csv"

# Caricamento del file CSV in un DataFrame usando pandas
df = pd.read_csv(percorso_file_csv)

# Stampa le prime righe del DataFrame per esaminare i dati iniziali
print(df.head())

# Restituisce il DataFrame (questa riga può essere omessa poiché non ha effetto sulla visualizzazione nell'output)
df
```

```
  id       name      rank    generation evolves_from  type1   type2  hp  \
0  1  bulbasaur  ordinary  generation-i     nothing  grass  poison  45
1  2    ivysaur  ordinary  generation-i   bulbasaur  grass  poison  60
2  3   venusaur  ordinary  generation-i     ivysaur  grass  poison  80
3  4 charmander  ordinary  generation-i     nothing   fire    None  39
4  5 charmeleon  ordinary  generation-i  charmander   fire    None  58

  atk  def  spatk  spdef  speed  total  height  weight  \
0  49   49     65     65     45    318       7      69
1  62   63     80     80     60    405      10     130
2  82   83    100    100     80    525      20    1000
3  52   43     60     50     65    309       6      85
4  64   58     80     65     80    405      11     190

          abilities                                          desc
0  overgrow chlorophyll  A strange seed was planted on its back at birt...
1  overgrow chlorophyll  When the bulb on its back grows large, it appe...
2  overgrow chlorophyll  The plant blooms when it is absorbing solar en...
3     blaze solar-power  Obviously prefers hot places. When it rains, s...
4     blaze solar-power  When it swings its burning tail, it elevates t...
```

Out[62]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	spdef	speed	total	height	weight	abilities	
0	1	bulbasaur	ordinary	generation-i	nothing	grass	poison	45	49	49	65	65	45	318	7	69	overgrow chlorophyll	A se
1	2	ivysaur	ordinary	generation-i	bulbasaur	grass	poison	60	62	63	80	80	60	405	10	130	overgrow chlorophyll	W
2	3	venusaur	ordinary	generation-i	ivysaur	grass	poison	80	82	83	100	100	80	525	20	1000	overgrow chlorophyll	T w ab sc
3	4	charmander	ordinary	generation-i	nothing	fire	None	39	52	43	60	50	65	309	6	85	blaze solar-power	Ol ra
4	5	charmeleon	ordinary	generation-i	charmander	fire	None	58	64	58	80	65	80	405	11	190	blaze solar-power	sv e
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1012	1013	sinistcha	ordinary	generation-ix	poltchageist	grass	ghost	71	60	106	121	80	70	508	2	22	hospitality heatproof	A
1013	1014	okidogi	legendary	generation-ix	nothing	poison	fighting	88	128	115	58	86	80	555	18	922	toxic-chain zero-to-hero	A
1014	1015	munkidori	legendary	generation-ix	nothing	poison	psychic	88	75	66	130	90	106	555	10	122	toxic-chain frisk	A
1015	1016	fezandipiti	legendary	generation-ix	nothing	poison	fairy	88	91	82	70	125	99	555	14	301	toxic-chain technician	A
1016	1017	ogerpon	legendary	generation-ix	nothing	grass	None	80	120	84	60	96	110	550	12	398	defiant	A

1017 rows × 18 columns

In [66]:
```
# Import delle librerie necessarie
import os
import pandas as pd

# Definizione del percorso della cartella contenente i file CSV
percorso_cartella = 'C://Users//enric//Desktop//serieAnuovo/'

# Lista che conterrà i DataFrame creati da ciascun file CSV
lista_dataframes = []

# Ottieni la lista dei file nella cartella specificata
elenco_file = os.listdir(percorso_cartella)

# Itera attraverso ogni file nella cartella
```

```
for nome_file in elenco_file:
    # Costruisci il percorso completo del file CSV
    percorso_file_csv = os.path.join(percorso_cartella, nome_file)

    # Leggi il file CSV e aggiungi il DataFrame alla lista
    df = pd.read_csv(percorso_file_csv)
    lista_dataframes.append(df)

# Restituisce l'ultimo DataFrame letto (questa riga può essere omessa poiché non ha effetto sulla visualizzazione nell'output)
df
```

Out[66]:

	Div	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	...	BbAv<2.5	BbAH	BbAHh	BbMxAHH	BbAvAHH	BbMxAHA
0	I1	20/08/16	Juventus	Fiorentina	2	1	H	1.0	0.0	H	...	1.78	36	-1.00	1.85	1.82	2.11
1	I1	20/08/16	Roma	Udinese	4	0	H	0.0	0.0	D	...	2.04	32	-1.50	2.45	2.31	1.72
2	I1	21/08/16	Atalanta	Lazio	3	4	A	0.0	3.0	A	...	1.63	31	0.25	1.85	1.80	2.15
3	I1	21/08/16	Bologna	Crotone	1	0	H	0.0	0.0	D	...	1.53	31	-0.50	1.95	1.87	2.06
4	I1	21/08/16	Chievo	Inter	2	0	H	0.0	0.0	D	...	1.60	31	0.25	2.16	2.09	1.84
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
375	I1	28/05/17	Inter	Udinese	5	2	H	3.0	0.0	H	...	3.39	19	-1.50	1.94	1.91	1.99
376	I1	28/05/17	Palermo	Empoli	2	1	H	0.0	0.0	D	...	2.17	19	1.00	1.90	1.85	2.07
377	I1	28/05/17	Roma	Genoa	3	2	H	1.0	1.0	D	...	4.67	15	-3.00	1.90	1.84	2.10
378	I1	28/05/17	Sampdoria	Napoli	2	4	A	0.0	2.0	A	...	4.32	15	2.25	1.99	1.95	1.94
379	I1	28/05/17	Torino	Sassuolo	5	3	H	3.0	2.0	H	...	3.62	19	-1.00	2.03	1.97	1.93

380 rows × 64 columns