```
import pandas as pd
import warnings

warnings.filterwarnings('ignore')
arquivo = "dados.csv"


try:
    df = pd.read_csv(arquivo, encoding='latin-1', error_bad_lines=False, delimiter=';')
except pd.errors.ParserError as e:
    print(f"Erro na leitura do arquivo CSV: {e}")


if 'df' in locals():
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 132 entries, 0 to 131
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID                              115 non-null    float64
 1   Nome                            115 non-null    object
 2   Idade                           115 non-null    float64
 3   Estado                          115 non-null    object
 4   Escolaridade                    115 non-null    object
 5   Tamanho_Propriedade (hectares)  115 non-null    float64
 6   Producao_Anual (toneladas)      115 non-null    float64
 7   Faturamento_Anual (BRL)         115 non-null    float64
 8   Margem_Lucros (%)               115 non-null    object
 9   Cultura                         115 non-null    object
 10  Score_Credito                   115 non-null    float64
 11  Maquinario                      115 non-null    float64
 12  Funcionarios                    115 non-null    float64
 13  Caixa                           115 non-null    float64
 14  Tipo_credito                    115 non-null    object
dtypes: float64(9), object(6)
memory usage: 15.6+ KB
```

```
df.head()
```

	ID	Nome	Idade	Estado	Escolaridade	Tamanho_Propriedade (hectares)	Producao_Anual (toneladas)	Fatura
0	1.0	João Silva	35.0	São Paulo	Ensino Médio	100.0	500.0	
1	2.0	Maria Santos	42.0	Minas Gerais	Ensino Superior	200.0	800.0	
2	3.0	José Oliveira	28.0	Bahia	Ensino Médio	150.0	300.0	
3	4.0	Ana Pereira	55.0	Rio Grande do Sul	Ensino Fundamental	80.0	200.0	
4	5.0	Ricardo Almeida	48.0	Paraná	Ensino Superior	300.0	1200.0	

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
# Definindo os atributos
atributos = ['Faturamento_Anual (BRL)', 'Margem_Lucros (%)', 'Score_Credito', 'Caixa', 'Funcionarios']

# Remover o símbolo "%" da coluna "Margem_Lucros (%)"
df['Margem_Lucros (%)'] = df['Margem_Lucros (%)'].replace({'%': ''}, regex=True)
df['Margem_Lucros (%)'] = df['Margem_Lucros (%)'].astype(float)

# Normalização Min-Max para colocar as variáveis na faixa [0, 1]
scaler = MinMaxScaler()
df_normalized = df.copy()
df_normalized[atributos] = scaler.fit_transform(df[atributos])

# Padronização (Z-score) para deixar as variáveis com média 0 e desvio padrão 1
```

```
standard_scaler = StandardScaler()
df_standardized = df.copy()
df_standardized[atributos] = standard_scaler.fit_transform(df[atributos])

# Calcular o Score de Crédito Agro usando uma fórmula mais complexa
df['Score_Credito_Agro_Normalized'] = (
    df_normalized['Faturamento_Anual (BRL)'] * 1.2 +
    df_normalized['Margem_Lucros (%)'] * 1.15 +
    df_normalized['Score_Credito'] * 1.3 +
    df_normalized['Caixa'] * 1.1 -
    df_normalized['Funcionarios'] * 1.05
)

df['Score_Credito_Agro_Standardized'] = (
    df_standardized['Faturamento_Anual (BRL)'] * 1.2 +
    df_standardized['Margem_Lucros (%)'] * 1.15 +
    df_standardized['Score_Credito'] * 1.3 +
    df_standardized['Caixa'] * 1.1 -
    df_standardized['Funcionarios'] * 1.05
)

# Exibir o DataFrame com os Scores de Crédito Agro calculados
print(df[['ID', 'Nome', 'Score_Credito_Agro_Normalized', 'Score_Credito_Agro_Standardized']])
```

```
       ID            Nome  Score_Credito_Agro_Normalized  \
0     1.0       João Silva                       1.838241
1     2.0     Maria Santos                       2.290284
2     3.0    José Oliveira                       0.790539
3     4.0      Ana Pereira                       0.581413
4     5.0  Ricardo Almeida                       1.628826
..    ...              ...                            ...
127   NaN              NaN                            NaN
128   NaN              NaN                            NaN
129   NaN              NaN                            NaN
130   NaN              NaN                            NaN
131   NaN              NaN                            NaN

     Score_Credito_Agro_Standardized
0                          -0.131782
1                           1.806415
2                          -4.100685
3                          -5.293176
4                          -0.968917
..                               ...
127                              NaN
128                              NaN
129                              NaN
130                              NaN
131                              NaN

[132 rows x 4 columns]
```

```
df.head()
```

	ID	Nome	Idade	Estado	Escolaridade	Tamanho_Propried (hectar
0	1.0	João Silva	35.0	São Paulo	Ensino Médio	10
1	2.0	Maria Santos	42.0	Minas Gerais	Ensino Superior	20
2	3.0	José Oliveira	28.0	Bahia	Ensino Médio	15
3	4.0	Ana Pereira	55.0	Rio Grande do Sul	Ensino Fundamental	8

```
# Exportar o DataFrame para um arquivo CSV
caminho_arquivo = r'C:\Users\luish\OneDrive\Área de Trabalho\FIAP Projetos\FASE 5\MINSAIT]\dados_score_vf.csv'

# Escrever o DataFrame no arquivo CSV
df.to_csv(caminho_arquivo, index=False)
```

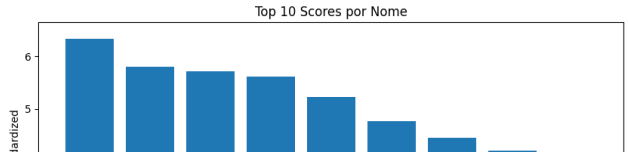
```
print('DataFrame exportado para CSV com sucesso.')
```

```
DataFrame exportado para CSV com sucesso.
```

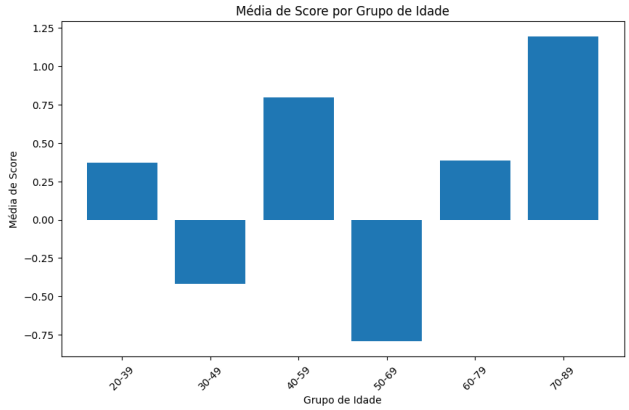
```
# Ordenar o DataFrame por uma coluna
df = df.sort_values(by='Score_Credito_Agro_Standardized', ascending=False)

df.head()
```

	ID	Nome	Idade	Estado	Escolaridade	Tamanho_Prop (he
8	9.0	Fernando Sousa	45.0	Mato Grosso	Ensino Superior	
38	39.0	Lucas Pereira	31.0	Goiás	Ensino Médio	
18	19.0	José Oliveira	56.0	Pará	Ensino Superior	
57	58.0	Bárbara	41.0	Mato Grosso	Ensino Médio	

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

df = pd.DataFrame(df)

# Ordenar o DataFrame por Score em ordem decrescente
df_ordenado = df.sort_values(by='Score_Credito_Agro_Standardized', ascending=False)

# Selecionar os 10 primeiros registros
top_10 = df_ordenado.head(10)

# Criar o gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(top_10['Nome'], top_10['Score_Credito_Agro_Standardized'])
plt.xlabel('Nome')
plt.ylabel('Score_Credito_Agro_Standardized')
plt.title('Top 10 Scores por Nome')
plt.xticks(rotation=45)  # Rotacionar os nomes no eixo x para melhor legibilidade
plt.show()
```




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

df = pd.DataFrame(df)

# Definir os intervalos de idades
intervalos = [20, 30, 40, 50, 60, 70, 80]

# Criar rótulos para os intervalos
rotulos = [f'{i}-{i+19}' for i in intervalos[:-1]]  # Remove o último intervalo

# Criar uma coluna 'Grupo_Idade' com base nos intervalos
df['Grupo_Idade'] = pd.cut(df['Idade'], bins=intervalos, labels=rotulos, right=False)

# Calcular a média de score por grupo de idade
media_scores = df.groupby('Grupo_Idade')['Score_Credito_Agro_Standardized'].mean().reset_index()

# Criar o gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(media_scores['Grupo_Idade'], media_scores['Score_Credito_Agro_Standardized'])
plt.xlabel('Grupo de Idade')
plt.ylabel('Média de Score')
plt.title('Média de Score por Grupo de Idade')
plt.xticks(rotation=45)
plt.show()
```



O código fornecido realiza o cálculo do "Score_Credito_Agro" com base em uma combinação linear dos atributos fornecidos. O código seleciona os atributos relevantes que serão usados para calcular o "Score_Credito_Agro". Esses atributos são: faturamento anual, margem de lucros, score de crédito, caixa e número de funcionários.

Cada atributo é multiplicado por um peso específico e, em seguida, esses valores ponderados são somados para obter o score final.

O faturamento anual contribui com 20% para o score. A margem de lucros contribui com 15%. O score de crédito contribui com 30%. O caixa contribui com 10%. O número de funcionários é subtraído com um peso de -5%. Esses pesos são valores fictícios e podem ser ajustados

conforme necessário, dependendo da lógica de cálculo real do score de crédito agro. A ideia geral é que atributos positivos (como faturamento, margem e score) aumentem o score, enquanto atributos negativos o diminuam.

```
import pandas as pd

df = pd.DataFrame(df)

# Calcule alguns dados estatísticos
media_idade = df['Idade'].mean()
media_faturamento = df['Faturamento_Anual (BRL)'].mean()
maior_score = df['Score_Credito'].max()
menor_score = df['Score_Credito'].min()

# Exiba os resultados
print(f'Média de Idade: {media_idade}')
print(f'Média de Faturamento Anual: {media_faturamento}')
print(f'Maior Score de Crédito: {maior_score}')
print(f'Menor Score de Crédito: {menor_score}')
```

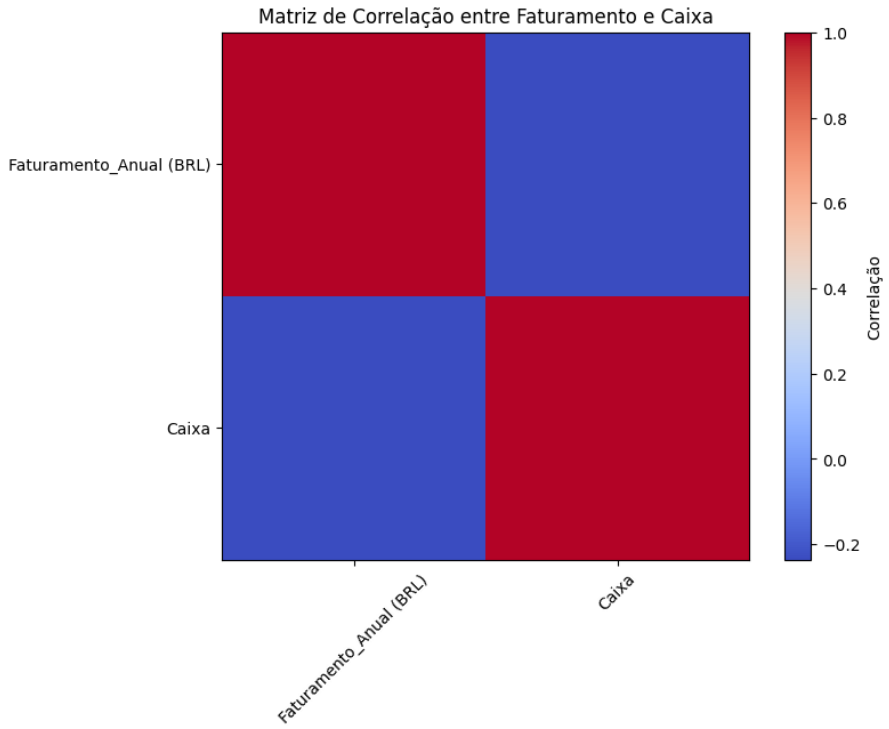
```
Média de Idade: 44.6
Média de Faturamento Anual: 2760777.3739130436
Maior Score de Crédito: 960.0
Menor Score de Crédito: 500.0
```

```
import pandas as pd

df = pd.DataFrame(df)

# 1. Resumo Estatístico Geral
resumo_geral = df.describe()

# 2. Correlação entre Variáveis
correlacao = df.corr()

# 3. Histogramas e Gráficos de Boxplot
import matplotlib.pyplot as plt

#Histograma do Score de Crédito
plt.figure(figsize=(8, 6))
plt.hist(df['Score_Credito'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Score de Crédito')
plt.ylabel('Frequência')
plt.title('Distribuição do Score de Crédito')
plt.show()
```

```
import pandas as pd

# Criar a matriz de correlação entre "Faturamento_Anual (BRL)" e "Caixa"
matriz_correlacao = df[['Faturamento_Anual (BRL)', 'Caixa']].corr()

# Exibir a matriz de correlação
print(matriz_correlacao)
```

```
                        Faturamento_Anual (BRL)     Caixa
Faturamento_Anual (BRL)                1.000000 -0.236321
Caixa                                 -0.236321  1.000000
```

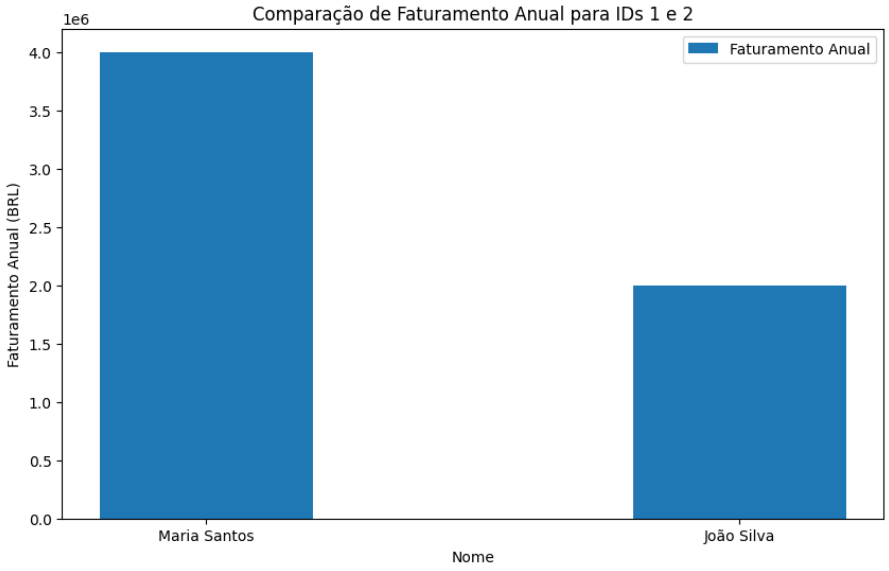
```
import pandas as pd
import matplotlib.pyplot as plt

# Criar a matriz de correlação entre "Faturamento_Anual (BRL)" e "Caixa"
matriz_correlacao = df[['Faturamento_Anual (BRL)', 'Caixa']].corr()

# Criar o gráfico de matriz de correlação
plt.figure(figsize=(8, 6))
plt.imshow(matriz_correlacao, cmap='coolwarm', interpolation='nearest')
plt.colorbar(label='Correlação')
plt.xticks(range(len(matriz_correlacao.columns)), matriz_correlacao.columns, rotation=45)
plt.yticks(range(len(matriz_correlacao.columns)), matriz_correlacao.columns)
plt.title('Matriz de Correlação entre Faturamento e Caixa')
plt.show()
```



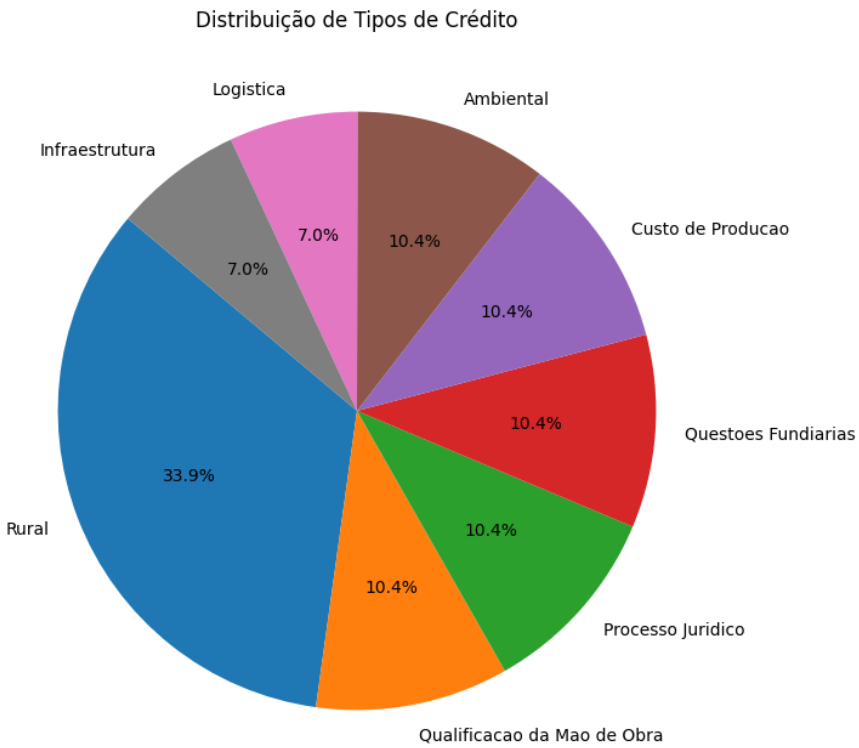

```
import pandas as pd

id_desejado = 2
dados_do_id = df[df['ID'] == id_desejado]


media_faturamento_id = dados_do_id['Faturamento_Anual (BRL)']
print(f'Média do Faturamento Anual para o ID {id_desejado}: {media_faturamento_id}')
```

```
Média do Faturamento Anual para o ID 2: 1    4000000.0
Name: Faturamento_Anual (BRL), dtype: float64
```

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

# Selecione os IDs que deseja comparar (por exemplo, IDs 1 e 2)
ids_desejados = [1, 2]
dados_dos_ids = df[df['ID'].isin(ids_desejados)]

# Realize análises específicas com base nos dados dos IDs selecionados
# Por exemplo, você pode criar um gráfico de barras para comparar o faturamento anual dos dois IDs com os nomes correspondentes:
nomes_dos_ids = dados_dos_ids['Nome'].tolist()
faturamento_anual = dados_dos_ids['Faturamento_Anual (BRL)'].tolist()

plt.figure(figsize=(10, 6))
plt.bar(nomes_dos_ids, faturamento_anual, width=0.4, label='Faturamento Anual')
plt.xlabel('Nome')
plt.ylabel('Faturamento Anual (BRL)')
plt.title('Comparação de Faturamento Anual para IDs 1 e 2')
plt.legend()
plt.show()

# Você pode realizar outras análises e visualizações com base nos dados dos IDs selecionados.
```




```
import matplotlib.pyplot as plt

df = pd.DataFrame(df)

# Contar o número de ocorrências de cada tipo de crédito
contagem_tipos_credito = df['Tipo_credito '].value_counts()

# Criar um gráfico de pizza
plt.figure(figsize=(8, 8))
plt.pie(contagem_tipos_credito, labels=contagem_tipos_credito.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribuição de Tipos de Crédito')
plt.show()
```


Distribuição de Tipos de Crédito
Logistica
Ambiental
Infraestrutura
Custo de Producao
Questoes Fundiarias
Rural
Processo Juridico
Qualificacao da Mao de Obra
7.0%
10.4%
7.0%
10.4%
10.4%
33.9%
10.4%
10.4%