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

In [2]:
```
np.random.seed(42)
```

In [3]:
```
data = {
    "Unemployment_Rate (%)": np.random.normal(loc=5.5, scale=1.5, size=594).clip
    "Flood_Drought_Risk_Score": np.random.randint(0, 11, 594),
    "Crime_Rate (per 1000)": np.random.normal(loc=4, scale=2, size=594).clip(1,
    "Agri_Workforce_%": np.random.normal(loc=45, scale=20, size=594).clip(5, 80)
    "Slum_Population_%": np.random.normal(loc=15, scale=10, size=594).clip(0, 40
    "Credit_Penetration_%": np.random.normal(loc=60, scale=10, size=594).clip(30
    "Net_Migration_Rate (%)": np.random.normal(loc=0.5, scale=2.5, size=594).cli
    "Political_Unrest_Score": np.random.randint(0, 11, 594),
    "Economic_Vulnerability_Score": np.random.randint(0, 11, 594)
}
```

In [4]:
```
df = pd.DataFrame(data)
```

In [5]:
```
geo = pd.read_csv(r"C:\Users\Admin\Desktop\Ind_adm2_Points.csv", encoding='ISO-8
```

In [6]:
```
geo
```

Out[6]:

	ÿCountry	District	Ind_adm2_ID	point_order	State	sub_polygon_id	Lati
0	India	Andaman Islands	0	326	Andaman and Nicobar	28	11.
1	India	Andaman Islands	0	327	Andaman and Nicobar	28	11.
2	India	Andaman Islands	0	328	Andaman and Nicobar	28	11.
3	India	Andaman Islands	0	329	Andaman and Nicobar	28	11.
4	India	Andaman Islands	0	330	Andaman and Nicobar	28	11.
...	...	...	...	...	...	...	
784557	India	North 24 Parganas	589	62	West Bengal	45	22.
784558	India	North 24 Parganas	589	63	West Bengal	45	22.
784559	India	North 24 Parganas	589	64	West Bengal	45	22.
784560	India	North 24 Parganas	589	65	West Bengal	45	22.
784561	India	North 24 Parganas	589	66	West Bengal	45	22.

784562 rows × 9 columns

In [7]:
```
geo_df = geo[['District', 'State', 'Latitude', 'Longitude']]
```

In [8]:
```
geo_clean = geo_df.groupby(['District', 'State']).agg({
    'Latitude': 'mean',
    'Longitude': 'mean'
}).reset_index()
```

In [9]:
```
print(geo_clean.head(), len(geo_clean))
```

```
     District           State   Latitude  Longitude
0    Adilabad  Andhra Pradesh  19.284514  78.813212
1        Agra   Uttar Pradesh  26.995130  78.052783
2   Ahmadabad         Gujarat  22.605426  72.238981
3  Ahmednagar     Maharashtra  19.217599  74.692203
4      Aizawl         Mizoram  23.907813  92.853595 594
```

In [10]:
```
# Adding synthetic target: Historical NPA Rate (%)
df["Historical_NPA_Rate (%)"] = (
    0.3 * df["Unemployment_Rate (%)"] +
    0.2 * df["Flood_Drought_Risk_Score"] +
    0.2 * df["Crime_Rate (per 1000)"] +
    0.1 * df["Agri_Workforce_%"]/10 +
    0.1 * df["Slum_Population_%"]/10 +
    0.1 * df["Economic_Vulnerability_Score"] +
    np.random.normal(loc=0, scale=1.5, size=594)
).round(2).clip(2, 15)
```

In [11]:
```
final_df = pd.concat([geo_clean, df], axis=1)
```

In [12]:
```
final_df
```

Out[12]:

_Rate (%)	Flood_Drought_Risk_Score	Crime_Rate (per 1000)	Agri_Workforce_%	Slum_Population_%	Cre
45071	0	4.980839	59.685218	12.451002	
92604	5	2.792440	25.132436	19.436139	
71533	5	3.532019	44.600491	21.170302	
34545	3	7.693612	31.669590	18.673887	
48770	7	2.826904	77.122390	13.908915	
...	...	...	...	...	
15874	3	1.000000	41.336435	14.320853	
07349	3	4.058094	55.065693	13.091826	
33514	1	6.150628	61.261473	14.631204	
18470	3	2.924621	27.818241	12.637639	
00493	10	4.606890	68.906722	9.719662	

In [13]:
```
final_df.describe()
```

Out[13]:

	Latitude	Longitude	Unemployment_Rate (%)	Flood_Drought_Risk_Score	Crime_R (per 10
count	594.000000	594.000000	594.000000	594.000000	594.000
mean	23.215872	81.100989	5.482519	5.070707	4.207
std	5.743757	6.245789	1.444596	3.212284	1.957
min	7.835291	68.821923	2.000000	0.000000	1.000
25%	20.570902	76.414526	4.452968	2.000000	2.819
50%	24.478686	79.429121	5.517119	5.000000	4.174
75%	26.861661	85.312213	6.435704	8.000000	5.544
max	34.599235	96.596708	10.000000	10.000000	10.000

In [14]:
```
final_df.isnull().sum()
```

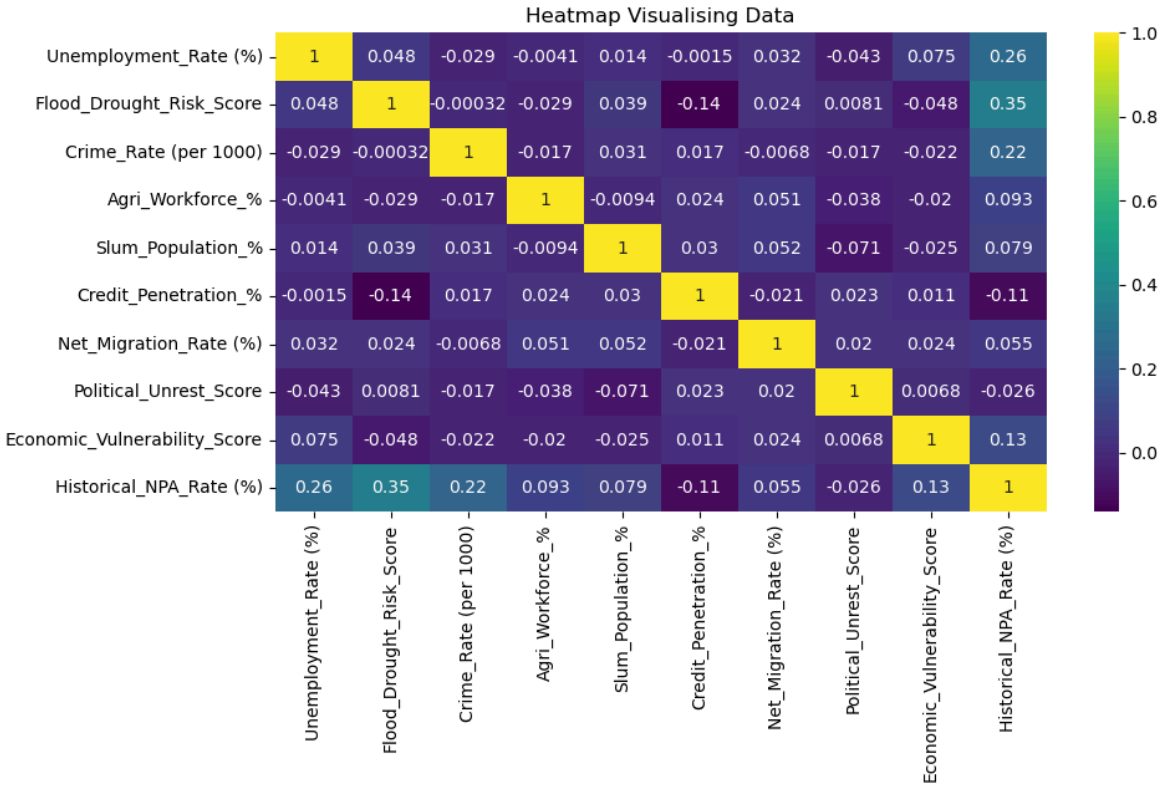
Out[14]:
```
District                    0
State                       0
Latitude                    0
Longitude                   0
Unemployment_Rate (%)       0
Flood_Drought_Risk_Score    0
Crime_Rate (per 1000)       0
Agri_Workforce_%            0
Slum_Population_%           0
Credit_Penetration_%        0
Net_Migration_Rate (%)      0
Political_Unrest_Score      0
Economic_Vulnerability_Score    0
Historical_NPA_Rate (%)     0
dtype: int64
```

In [15]:
```
final_df.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
final_df_numeric=df.drop(['District','State'],errors='ignore',axis=1)
```

In [17]:
```
plt.figure(figsize=(10,5))
sns.heatmap(data=final_df_numeric.corr(),annot=True,cmap="viridis")
plt.title("Heatmap Visualising Data")
plt.show()
```



In [18]:
```
def categorize_unemployment(val):
    if val <= 4: return 'Low Risk'
    elif val <= 7: return 'Medium Risk'
    else: return 'High Risk'

def categorize_flood_drought(val):
    if val <= 3: return 'Low Risk'
    elif val <= 6: return 'Medium Risk'
    else: return 'High Risk'

def categorize_crime(val):
    if val <= 3: return 'Low Risk'
    elif val <= 6: return 'Medium Risk'
    else: return 'High Risk'

def categorize_agri(val):
    if val <= 30: return 'Low Risk'
    elif val <= 60: return 'Medium Risk'
    else: return 'High Risk'

def categorize_slum(val):
    if val <= 10: return 'Low Risk'
    elif val <= 25: return 'Medium Risk'
    else: return 'High Risk'

def categorize_credit(val):
    if val >= 70: return 'Low Risk'
    elif val >= 50: return 'Medium Risk'
    else: return 'High Risk'

def categorize_migration(val):
    if val > 0: return 'Low Risk'
    elif val >= -2: return 'Medium Risk'
    else: return 'High Risk'
```

```
def categorize_political(val):
    if val <= 3: return 'Low Risk'
    elif val <= 6: return 'Medium Risk'
    else: return 'High Risk'

def categorize_econ(val):
    if val <= 3: return 'Low Risk'
    elif val <= 6: return 'Medium Risk'
    else: return 'High Risk'
```

In [19]:
```
final_df['Unemployment_Risk'] = final_df['Unemployment_Rate (%)'].apply(categori
final_df['Flood_Drought_Risk'] = final_df['Flood_Drought_Risk_Score'].apply(cate
final_df['Crime_Risk'] = final_df['Crime_Rate (per 1000)'].apply(categorize_crim
final_df['Agri_Risk'] = final_df['Agri_Workforce_%'].apply(categorize_agri)
final_df['Slum_Risk'] = final_df['Slum_Population_%'].apply(categorize_slum)
final_df['Credit_Risk'] = final_df['Credit_Penetration_%'].apply(categorize_cred
final_df['Migration_Risk'] = final_df['Net_Migration_Rate (%)'].apply(categorize
final_df['Political_Risk'] = final_df['Political_Unrest_Score'].apply(categorize
final_df['Econ_Risk'] = final_df['Economic_Vulnerability_Score'].apply(categoriz
```

In [20]:
```
risk_map = {'Low Risk': 1, 'Medium Risk': 2, 'High Risk': 3}
```

In [21]:
```
final_df['Unemployment_Score'] = final_df['Unemployment_Risk'].map(risk_map)
final_df['Flood_Drought_Score'] = final_df['Flood_Drought_Risk'].map(risk_map)
final_df['Crime_Score'] = final_df['Crime_Risk'].map(risk_map)
final_df['Agri_Score'] = final_df['Agri_Risk'].map(risk_map)
final_df['Slum_Score'] = final_df['Slum_Risk'].map(risk_map)
final_df['Credit_Score'] = final_df['Credit_Risk'].map(risk_map)
final_df['Migration_Score'] = final_df['Migration_Risk'].map(risk_map)
final_df['Political_Score'] = final_df['Political_Risk'].map(risk_map)
final_df['Econ_Score'] = final_df['Econ_Risk'].map(risk_map)
```

In [22]:
```
risk_score_columns = [
    'Unemployment_Score', 'Flood_Drought_Score', 'Crime_Score', 'Agri_Score',
    'Slum_Score', 'Credit_Score', 'Migration_Score', 'Political_Score', 'Econ_Sc
]

final_df['Total_Lending_Risk_Score'] = final_df[risk_score_columns].sum(axis=1)
final_df['Average_Lending_Risk_Score'] = final_df[risk_score_columns].mean(axis=
```

In [23]:
```
def overall_risk_category(avg_score):
    if avg_score <= 1.6:
        return 'Low Risk'
    elif avg_score <= 2.3:
        return 'Medium Risk'
    else:
        return 'High Risk'

final_df['Overall_Lending_Risk_Category'] = final_df['Average_Lending_Risk_Score
```

In [24]:
```
final_df['Overall_Lending_Risk_Category'].value_counts()
```

Out[24]:
```
Overall_Lending_Risk_Category
Medium Risk    492
Low Risk        55
High Risk       47
Name: count, dtype: int64
```

In [53]:
```
final_df.to_csv('CreditChitra:District-Wise_Credit')
```