```
import pandas as pd

# Load data
audit_data = pd.read_csv('/content/drive/MyDrive/audit_data.csv')
trial_data = pd.read_csv('/content/drive/MyDrive/trial.csv')

# Display the first few rows to understand the structure
audit_data.head()
```

	Sector_score	LOCATION_ID	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers	...	RiSk_E	History	Prob	Risk_F	S
0	3.89	23	4.18	0.6	2.508	2.50	0.2	0.500	6.68	5.0	...	0.4	0	0.2	0.0	
1	3.89	6	0.00	0.2	0.000	4.83	0.2	0.966	4.83	5.0	...	0.4	0	0.2	0.0	
2	3.89	6	0.51	0.2	0.102	0.23	0.2	0.046	0.74	5.0	...	0.4	0	0.2	0.0	
3	3.89	6	0.00	0.2	0.000	10.80	0.6	6.480	10.80	6.0	...	0.4	0	0.2	0.0	
4	3.89	6	0.00	0.2	0.000	0.08	0.2	0.016	0.08	5.0	...	0.4	0	0.2	0.0	

5 rows × 27 columns

```
audit_data.shape
```

```
(776, 27)
```

```
audit_data.columns
```

```
Index(['Sector_score', 'LOCATION_ID', 'PARA_A', 'Score_A', 'Risk_A', 'PARA_B',
       'Score_B', 'Risk_B', 'TOTAL', 'numbers', 'Score_B.1', 'Risk_C',
       'Money_Value', 'Score_MV', 'Risk_D', 'District_Loss', 'PROB', 'RiSk_E',
       'History', 'Prob', 'Risk_F', 'Score', 'Inherent_Risk', 'CONTROL_RISK',
       'Detection_Risk', 'Audit_Risk', 'Risk'],
      dtype='object')
```

```
audit_data.dtypes
```

```
Sector_score      float64
LOCATION_ID        object
PARA_A            float64
Score_A           float64
Risk_A            float64
PARA_B            float64
Score_B           float64
Risk_B            float64
TOTAL             float64
numbers           float64
Score_B.1         float64
Risk_C            float64
Money_Value       float64
Score_MV          float64
Risk_D            float64
District_Loss       int64
PROB              float64
RiSk_E            float64
History             int64
Prob              float64
Risk_F            float64
Score             float64
Inherent_Risk     float64
CONTROL_RISK      float64
Detection_Risk    float64
Audit_Risk        float64
Risk                int64
dtype: object
```

```
trial_data.head()
```

	Sector_score	LOCATION_ID	PARA_A	SCORE_A	PARA_B	SCORE_B	TOTAL	numbers	Marks	Money_Value	MONEY_Marks	District	Loss	LOSS
0	3.89	23	4.18	6	2.50	2	6.68	5.0	2	3.38	2	2	0	
1	3.89	6	0.00	2	4.83	2	4.83	5.0	2	0.94	2	2	0	
2	3.89	6	0.51	2	0.23	2	0.74	5.0	2	0.00	2	2	0	
3	3.89	6	0.00	2	10.80	6	10.80	6.0	6	11.75	6	2	0	
4	3.89	6	0.00	2	0.08	2	0.08	5.0	2	0.00	2	2	0	

```
trial_data.shape
```

```
(776, 18)
```

```
trial_data.columns
```

```
Index(['Sector_score', 'LOCATION_ID', 'PARA_A', 'SCORE_A', 'PARA_B', 'SCORE_B',
       'TOTAL', 'numbers', 'Marks', 'Money_Value', 'MONEY_Marks', 'District',
       'Loss', 'LOSS_SCORE', 'History', 'History_score', 'Score', 'Risk'],
      dtype='object')
```

```
trial_data.dtypes
```

```
Sector_score     float64
LOCATION_ID       object
PARA_A           float64
SCORE_A            int64
PARA_B           float64
SCORE_B            int64
TOTAL            float64
numbers          float64
Marks              int64
Money_Value      float64
MONEY_Marks        int64
District           int64
Loss               int64
LOSS_SCORE         int64
History            int64
History_score      int64
Score            float64
Risk               int64
dtype: object
```

```
trial_data = trial_data.dropna()
```

```
# Example: Summarizing sector scores by location
sector_scores = audit_data.groupby('LOCATION_ID')['Sector_score'].mean().reset_index()
print(sector_scores)
```

```
   LOCATION_ID  Sector_score
0            1     20.342727
1           11     18.074231
2           12     20.585745
3           13     25.265143
4           14     23.278500
5           15     23.140286
6           16     16.519615
7           17     55.570000
8           18     23.853125
9           19     14.153676
10           2     15.687073
11          20     13.516000
12          21     37.795000
13          22     17.662917
14          23      3.890000
15          24      3.890000
16          25     29.138333
17          27     28.987500
18          28     22.855000
19          29     26.654286
20           3      3.890000
21          30      3.002500
22          31      4.227500
23          32     19.034138
24          33      3.890000
25          34      3.410000
26          35      2.890000
27          36     29.610000
28          37     14.130000
29          38      3.097500
30          39      9.374444
31           4     26.225946
32          40      2.706667
33          41      3.410000
34          42      3.410000
35          43     28.307143
36          44     55.570000
37           5     21.449773
38           6     19.270000
39           7     17.467500
40           8     21.414079
41           9     27.762264
42      LOHARU      1.990000
43         NUH      1.990000
44     SAFIDON      1.990000
```

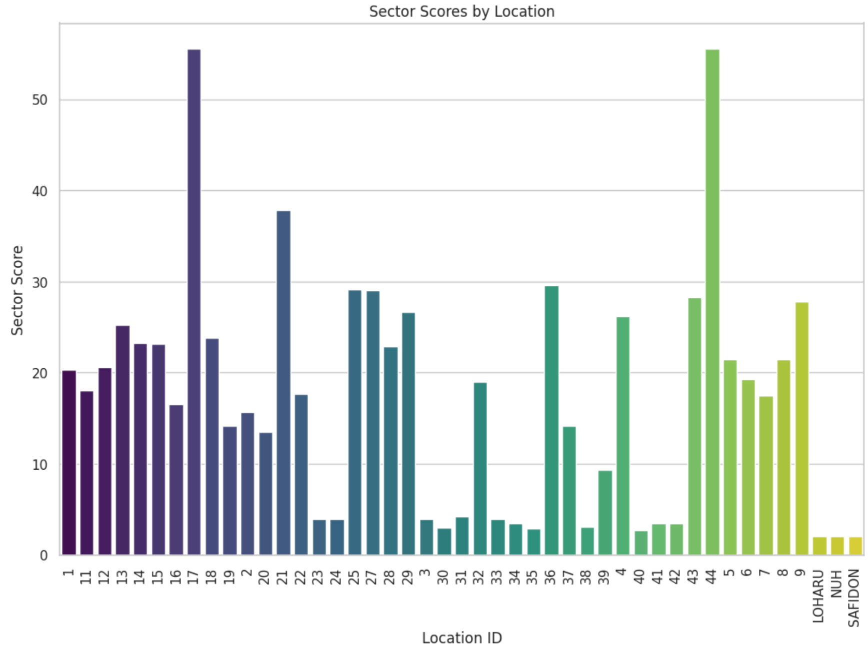
```
import matplotlib.pyplot as plt
import seaborn as sns

# Setting the style
sns.set(style="whitegrid")
# Visualize Sector Scores by Location
# Visualize Sector Scores by Location
plt.figure(figsize=(12, 8))
sns.barplot(x='LOCATION_ID', y='Sector_score', data=sector_scores, palette='viridis')
plt.title('Sector Scores by Location')
plt.xlabel('Location ID')
plt.ylabel('Sector Score')
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-11-58af00e27b2b>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x='LOCATION_ID', y='Sector_score', data=sector_scores, palette='viridis
```



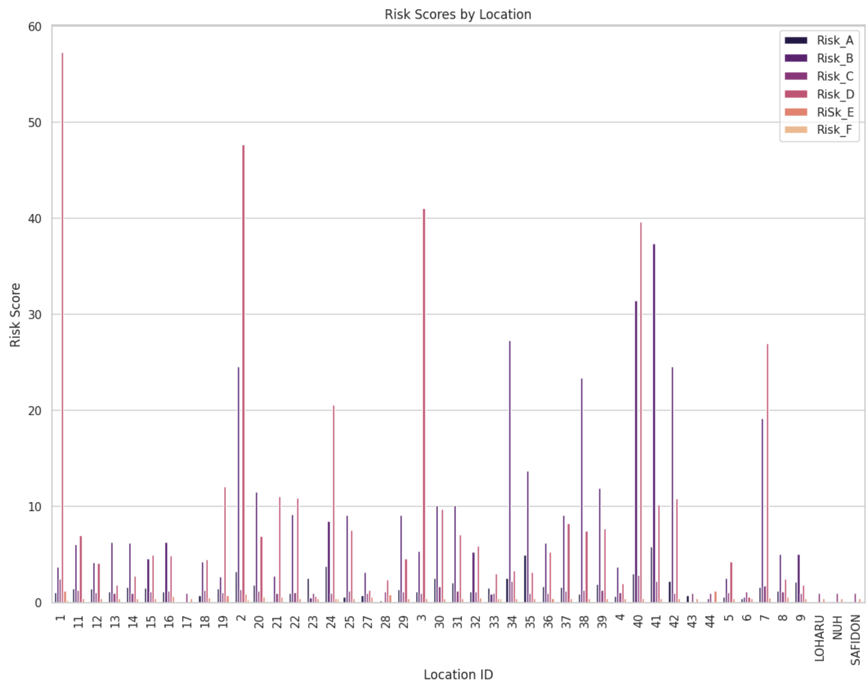

```
risk_scores = audit_data.groupby('LOCATION_ID')[['Risk_A', 'Risk_B', 'Risk_C', 'Risk_D', 'RiSk_E', 'Risk_F']].mean().reset_index()
print(risk_scores)
```

```
   LOCATION_ID    Risk_A     Risk_B    Risk_C     Risk_D    RiSk_E    Risk_F
0            1  1.047636   3.730000  2.454545  57.230545  1.200000  0.254545
1           11  1.450538   6.055628  1.296154   7.010615  0.400000  0.046154
2           12  1.424809   4.174043  1.055319   4.111787  0.451064  0.059574
3           13  1.130229   6.243447  1.000000   1.852971  0.411429  0.011429
4           14  1.583800   6.176900  1.000000   2.782600  0.400000  0.020000
5           15  1.553714   4.604186  1.108571   4.931543  0.400000  0.022857
6           16  1.157731   6.290450  1.219231   4.897885  0.638462  0.015385
7           17  0.000000   0.000000  1.000000   0.000000  0.400000  0.000000
8           18  0.755500   4.253750  1.312500   4.503500  0.475000  0.125000
9           19  1.410647   2.725206  1.017647  12.032176  0.705882  0.011765
10           2  3.275122  24.552195  1.397561  47.681756  0.878049  0.312195
11          20  1.825600  11.484000  1.240000   6.888400  0.560000  0.000000
12          21  0.081250   2.742000  1.000000  11.031500  0.550000  0.150000
13          22  0.941917   9.175500  1.050000  10.860000  0.466667  0.016667
14          23  2.508000   0.500000  1.000000   0.676000  0.400000  0.000000
15          24  3.756000   8.460000  1.000000  20.544000  0.400000  0.400000
16          25  0.592000   9.072000  1.200000   7.555667  0.400000  0.000000
17          27  0.720250   3.154500  1.000000   1.251750  0.550000  0.150000
18          28  0.162750   0.022000  1.150000   2.377750  0.800000  0.000000
19          29  1.392571   9.072390  1.114286   4.593810  0.419048  0.019048
20           3  1.126000   5.350667  1.000000  41.045333  0.400000  0.133333
21          30  2.573500  10.139500  1.650000   9.714500  0.400000  0.000000
22          31  2.096000  10.142500  1.200000   7.029500  0.466667  0.000000
23          32  1.133448   5.230016  1.165517   5.881517  0.482759  0.027586
24          33  1.542000   0.898000  1.000000   3.024000  0.400000  0.400000
25          34  2.550000  27.306000  2.200000   3.280000  0.400000  0.000000
26          35  4.923000  13.725000  1.000000   3.132000  0.400000  0.000000
27          36  1.688500   6.220500  1.000000   5.281500  0.400000  0.100000
28          37  1.614200   9.061400  1.240000   8.251200  0.400000  0.040000
```

```
29          38  0.865500  23.395500  1.300000   7.489000  0.400000  0.100000
30          39  1.875111  11.921111  1.266667   7.707556  0.400000  0.000000
31           4  0.673459   3.670432  1.070270   2.011389  0.432432  0.000000
32          40  3.022000  31.376000  2.866667  39.627333  0.400000  0.000000
33          41  5.802000  37.350000  2.200000  10.212000  0.400000  0.000000
34          42  2.262000  24.516000  1.000000  10.800000  0.400000  0.000000
35          43  0.702857   0.045714  1.000000   0.014000  0.400000  0.000000
36          44  0.000120   0.444000  1.000000   0.000000  1.200000  0.000000
37           5  0.568818   2.551000  1.065909   4.230682  0.400000  0.009091
38           6  0.427394   0.571394  1.115152   0.585030  0.412121  0.090909
39           7  1.572000  19.189500  1.725000  26.939000  0.500000  0.000000
40           8  1.170921   5.007109  1.151316   2.430308  0.547368  0.055263
41           9  2.171094   5.063102  1.000000   1.802302  0.422642  0.052830
42      LOHARU  0.060000   0.000000  1.000000   0.000000  0.400000  0.000000
43         NUH  0.110000   0.000000  1.000000   0.134000  0.400000  0.000000
44     SAFIDON  0.096000   0.000000  1.000000   0.094000  0.400000  0.000000
```

```
# Melt the dataframe to plot multiple risk columns
risk_scores_melted = risk_scores.melt(id_vars=['LOCATION_ID'], var_name='Risk_Type', value_name='Risk_Score')

plt.figure(figsize=(14, 10))
sns.barplot(x='LOCATION_ID', y='Risk_Score', hue='Risk_Type', data=risk_scores_melted, palette='magma')
plt.title('Risk Scores by Location')
plt.xlabel('Location ID')
plt.ylabel('Risk Score')
plt.xticks(rotation=90)
plt.legend(loc='upper right')
plt.show()
```




```
total_scores = audit_data.groupby('LOCATION_ID')['TOTAL'].sum().reset_index()
print(total_scores)
```

```
   LOCATION_ID      TOTAL
0            1    90.4300
1           11   339.4617
2           12   460.3600
3           13   448.4632
4           14   264.5600
5           15   373.2125
6           16   667.9570
7           17     0.0000
8           18   140.0000
9           19   499.9200
10           2  1897.3500
11          20   112.7900
12          21    39.6800
13          22   414.6400
14          23     6.6800
15          24    20.3600
16          25    98.1800
17          27    53.4300
18          28     4.7900
19          29   376.9209
20           3    35.0100
21          30    86.4300
22          31   247.4200
```

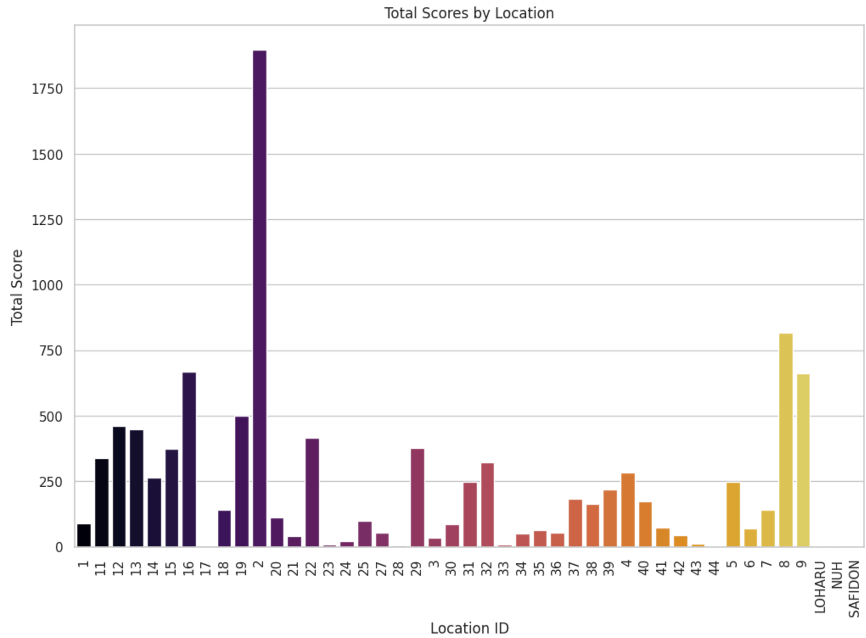
```
23          32   323.1023
24          33     7.0600
25          34    49.7600
26          35    62.8200
27          36    53.3200
28          37   183.9300
29          38   162.7300
30          39   216.7300
31           4   282.8200
32          40   172.6100
33          41    71.9200
34          42    44.6300
35          43    10.8200
36          44     1.1106
37           5   246.2600
38           6    69.6500
39           7   141.4400
40           8   815.0113
41           9   662.4421
42      LOHARU     0.3000
43         NUH     0.5500
44     SAFIDON     0.4800
```

```
# Visualize Total Scores by Location
plt.figure(figsize=(12, 8))
sns.barplot(x='LOCATION_ID', y='TOTAL', data=total_scores, palette='inferno')
plt.title('Total Scores by Location')
plt.xlabel('Location ID')
plt.ylabel('Total Score')
plt.xticks(rotation=90)
plt.show()
```

```
<ipython-input-15-d3ef92356476>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x='LOCATION_ID', y='TOTAL', data=total_scores, palette='inferno')
```



```
def generate_detailed_report(data):
    report = {}

    # Sector scores analysis
    report['sector_scores'] = data.groupby('LOCATION_ID')['Sector_score'].mean().reset_index().rename(columns={'Sector_score': 'Avg_Sect

    # Risk scores analysis
    risk_columns = ['Risk_A', 'Risk_B', 'Risk_C', 'Risk_D', 'RiSk_E', 'Risk_F']
    report['risk_scores'] = data.groupby('LOCATION_ID')[risk_columns].mean().reset_index()

    # Total scores analysis
    report['total_scores'] = data.groupby('LOCATION_ID')['TOTAL'].sum().reset_index().rename(columns={'TOTAL': 'Sum_TOTAL'})

    # Historical analysis
    report['historical_analysis'] = data.groupby('LOCATION_ID')[['History', 'District_Loss']].agg({'History': 'sum', 'District_Loss': 's

    # Detailed risk assessment
    risk_assessment_columns = ['Inherent_Risk', 'CONTROL_RISK', 'Detection_Risk', 'Audit_Risk']
    report['detailed_risk_assessment'] = data.groupby('LOCATION_ID')[risk_assessment_columns].mean().reset_index()

    # Overall summary statistics for numeric columns only
    numeric_cols = data.select_dtypes(include=[float, int]).columns
    summary_stats = data[numeric_cols].describe().transpose()
    summary_stats['variance'] = data[numeric_cols].var()
    summary_stats['skewness'] = data[numeric_cols].skew()
    summary_stats['kurtosis'] = data[numeric_cols].kurt()
    report['summary_statistics'] = summary_stats.reset_index().rename(columns={'index': 'Feature'})

    # Convert report sections to DataFrames for easy export to CSV or other formats
    report_dfs = {section: pd.DataFrame(report[section]) for section in report}

    return report_dfs

# Example usage
audit_report = generate_detailed_report(audit_data)

# Print sections of the report
for section, df in audit_report.items():
    print(f"--- {section} ---")
    print(df.head())
```

```
--- sector_scores ---
   LOCATION_ID  Avg_Sector_score
0            1         20.342727
1           11         18.074231
2           12         20.585745
3           13         25.265143
4           14         23.278500
--- risk_scores ---
   LOCATION_ID    Risk_A    Risk_B    Risk_C     Risk_D    RiSk_E    Risk_F
0            1  1.047636  3.730000  2.454545  57.230545  1.200000  0.254545
1           11  1.450538  6.055628  1.296154   7.010615  0.400000  0.046154
2           12  1.424809  4.174043  1.055319   4.111787  0.451064  0.059574
3           13  1.130229  6.243447  1.000000   1.852971  0.411429  0.011429
4           14  1.583800  6.176900  1.000000   2.782600  0.400000  0.020000
--- total_scores ---
   LOCATION_ID  Sum_TOTAL
0            1    90.4300
1           11   339.4617
2           12   460.3600
3           13   448.4632
4           14   264.5600
--- historical_analysis ---
   LOCATION_ID  History  District_Loss
0            1        5             54
1           11        2             52
2           12        6            106
3           13        1             70
4           14        1             40
--- detailed_risk_assessment ---
   LOCATION_ID  Inherent_Risk  CONTROL_RISK  Detection_Risk  Audit_Risk
0            1      65.917273      1.454545             0.5   36.574000
1           11      16.259090      0.446154             0.5    3.704218
2           12      11.276596      0.510638             0.5    5.205838
3           13      10.649504      0.422857             0.5    3.306427
4           14      11.963300      0.420000             0.5    2.969740
--- summary_statistics ---
        Feature  count       mean        std   min    25%    50%     75%  \
0  Sector_score  776.0  20.184536  24.319017  1.85  2.370  3.890  55.570
1        PARA_A  776.0   2.450194   5.678870  0.00  0.210  0.875   2.480
2       Score_A  776.0   0.351289   0.174055  0.20  0.200  0.200   0.600
3        Risk_A  776.0   1.351029   3.440447  0.00  0.042  0.175   1.488
4        PARA_B  776.0  10.799988  50.083624  0.00  0.000  0.405   4.160

      max     variance   skewness    kurtosis
```

```
    0    59.85    591.414594    0.769987    -1.335745
```

```
from transformers import pipeline

# Load the summarization pipeline
summarizer = pipeline("summarization")

def split_text(text, max_chunk_length=1000):
    """Split text into smaller chunks"""
    chunks = []
    while len(text) > max_chunk_length:
        split_point = text.rfind('.', 0, max_chunk_length) + 1
        if split_point == 0:  # No period found
            split_point = max_chunk_length
        chunks.append(text[:split_point])
        text = text[split_point:]
    chunks.append(text)
    return chunks

def summarize_text(text, max_length=150, min_length=30):
    """Summarize text with a given length constraint"""
    summaries = []
    chunks = split_text(text)
    for chunk in chunks:
        summary = summarizer(chunk, max_length=max_length, min_length=min_length, do_sample=False)
        summaries.append(summary[0]['summary_text'])
    return ' '.join(summaries)

def summarize_report(report_dfs, max_length=150, min_length=30):
    summaries = {}
    for section, df in report_dfs.items():
        report_text = df.to_string()
        summary = summarize_text(report_text, max_length=max_length, min_length=min_length)
        summaries[section] = summary
    return summaries

# Generate detailed report
audit_report = generate_detailed_report(audit_data)

# Summarize the report
audit_summary = summarize_report(audit_report)
for section, summary in audit_summary.items():
    print(f"--- {section} Summary ---")
    print(summary)
```

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (https://huggingface.co/sshleifer/distilbart
Using a pipeline without specifying a model name and revision in production is not recommended.
--- sector_scores Summary ---
    LOCATION_ID   Avg_Sector_score   is 20.342727.34 . The score is based on an average of 3.890000 . The average is 3.410000 .  LOI
--- risk_scores Summary ---
 RiSk_E    Risk_A    risk_A, Risk_B and Risk_C risk factors are presented in the RiSk . The RiSk is based on the risk profile of a lo
--- total_scores Summary ---
    LOCATION_ID  Sum_TOTAL:   Sum_Total logistics includes conversations_took and reversal targets of comprehensive research tweets
--- historical_analysis Summary ---
 The location of the site has been chosen for the first time in the U.S. has been selected as a national Geographic Geographic Geogr
--- detailed_risk_assessment Summary ---
   LOCATION_ID  Inherent_Risk  Control_RISK  Detection_Risks  Detection .Risk . Audit_risk is 1.0% . Audit risk is 0.5% . Inherent r
--- summary_statistics Summary ---
 PARA_A  776.0  20.0 .0 . 0.00  0.351289    -1.335745.0 - 1.450194   5.678870 . Para_B   10.799988  50.0000   8.505663  100.00 . Ris
```

Start coding or generate with AI.