```
import pandas as pd
import numpy as np
import os
import pprint
import logging
```

The code defines a ProjectBudgetAnalyzer class that helps manage and analyze projects based on budgets, strategic alignment, and return on investment (ROI). Here is what it does:

Configuration Setup: It sets default settings like available budget, strategic focus areas, and maximum budget per project.

Data Generation: It can generate demo project data with random but realistic values for testing.

Data Loading: It can load project data from CSV or Excel files.

Strategic Scoring: It calculates strategic alignment scores by counting how many strategic goals each project meets.

Project Ranking: It ranks projects based on their strategic score and ROI.

Top Projects Selection: It selects top ROI projects that are within a given budget limit.

Add New Projects: It allows adding new projects dynamically.

Budget Checking: It checks if the total project costs exceed the available budget.

Export Results: It exports results (original data, ranked projects, budget summary) into files like CSV.

Summary Report: It generates a quick summary report (like average ROI, strategic coverage, and budget status).

```
# Setup basic logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class ProjectBudgetAnalyzer:
    def __init__(self, budget_limit=1_000_000):
        self.projects = pd.DataFrame()
        self.budget_limit = budget_limit
        self.results_path = "./results"
        if not os.path.exists(self.results_path):
            os.makedirs(self.results_path)
        logger.info("ProjectBudgetAnalyzer initialized.")

    def generate_demo_data(self, num_projects=10):
        np.random.seed(42)
        self.projects = pd.DataFrame({
            'PROJECT NUMBER': np.arange(1, num_projects + 1),
            'PROJECT DESCRIPTION': [f"Project {i}" for i in range(1, num_projects + 1)],
            'CUSTOMER': np.random.choice(['Vodacom', 'MTN', 'Telkom'], num_projects),
            'VERTICAL': np.random.choice(['Technology', 'Retail', 'Healthcare'], num_projects),
            'WH': np.random.choice(['JHB-DC1', 'CPT-DC2', 'DBN-DC3'], num_projects),
            'SQM': np.random.randint(500, 5000, num_projects),
            'COUNTRY': 'South Africa',
            'TURNOVER': np.random.randint(10_000_000, 100_000_000, num_projects),
            'YEAR 1': np.random.randint(100_000, 1_000_000, num_projects),
            'YEAR 2': np.random.randint(100_000, 1_000_000, num_projects),
            'YEAR 3': np.random.randint(100_000, 1_000_000, num_projects),
            'TOTAL': lambda df: df['YEAR 1'] + df['YEAR 2'] + df['YEAR 3'],
            'ROI': np.round(np.random.uniform(1.0, 10.0, num_projects), 2),
            'SUSTAINABILITY': np.random.randint(0, 2, num_projects),
            'BUSINESS DEV': np.random.randint(0, 2, num_projects),
            'HSE': np.random.randint(0, 2, num_projects),
            'DIGITAL TRANSFORMATION': np.random.randint(0, 2, num_projects),
            'CSR': np.random.randint(0, 2, num_projects),
            'OPERATIONAL EXCELLENCE': np.random.randint(0, 2, num_projects)
        })
        self.projects['TOTAL'] = self.projects['YEAR 1'] + self.projects['YEAR 2'] + self.projects['YEAR 3']
        logger.info("Demo data generated.")

    def calculate_strategic_scores(self):
        strategic_cols = ['SUSTAINABILITY', 'BUSINESS DEV', 'HSE', 'DIGITAL TRANSFORMATION', 'CSR', 'OPERATIONAL EXCELLENCE']
        self.projects['STRATEGIC SCORE'] = self.projects[strategic_cols].sum(axis=1)
        logger.info("Strategic scores calculated.")

    def rank_projects(self):
        self.projects['RANK'] = self.projects.sort_values(
            ['STRATEGIC SCORE', 'ROI'], ascending=[False, False]
        ).reset_index().index + 1
        logger.info("Projects ranked.")
        return self.projects[['PROJECT NUMBER', 'PROJECT DESCRIPTION', 'RANK', 'STRATEGIC SCORE', 'ROI']]

    def get_top_roi_projects(self, top_n=5):
```

```
            top_projects = self.projects.sort_values('ROI', ascending=False).head(top_n)
            logger.info(f"Top {top_n} ROI projects selected.")
            return top_projects[['PROJECT NUMBER', 'PROJECT DESCRIPTION', 'ROI']]

        def add_project(self, project_data):
            project_df = pd.DataFrame([project_data])
            self.projects = pd.concat([self.projects, project_df], ignore_index=True)
            logger.info(f"New project added: {project_data['PROJECT DESCRIPTION']}.")

        def check_budget_constraints(self):
            total_budget = self.projects['TOTAL'].sum()
            budget_remaining = self.budget_limit - total_budget
            logger.info(f"Total budget used: {total_budget}. Remaining: {budget_remaining}.")
            return {
                'Total Budget Used': float(total_budget),
                'Budget Remaining': float(budget_remaining),
                'Within Budget': total_budget <= self.budget_limit
            }

        def generate_summary_report(self):
            summary = {
                'Total Projects': int(len(self.projects)),
                'Average ROI': float(self.projects['ROI'].mean()),
                'Top Strategic Project': self.projects.sort_values('STRATEGIC SCORE', ascending=False).iloc[0]['PROJECT DESCRIPTION'],
                'Budget Status': self.check_budget_constraints()
            }
            logger.info("Summary report generated.")
            return summary

        def export_results(self):
            ranked_path = os.path.join(self.results_path, "ranked_projects.csv")
            self.projects.to_csv(ranked_path, index=False)
            logger.info(f"Results exported to {ranked_path}.")
            return {"ranked_projects": ranked_path}


    def run_analysis():
        try:
            analyzer = ProjectBudgetAnalyzer()
            analyzer.generate_demo_data(num_projects=15)

            # Print the entire projects DataFrame here
            print("\n=== DEMO PROJECTS DATAFRAME ===")
            print(analyzer.projects)

            analyzer.calculate_strategic_scores()
            ranked_projects = analyzer.rank_projects()

            print("\n=== RANKED PROJECTS ===")
            print(ranked_projects)

            top_roi = analyzer.get_top_roi_projects()
            print("\n=== TOP ROI PROJECTS ===")
            print(top_roi)

            new_project = {
                'PROJECT NUMBER': 100,
                'PROJECT DESCRIPTION': 'New Strategic Initiative',
                'CUSTOMER': 'Vodacom SA',
                'VERTICAL': 'Technology',
                'WH': 'JHB-DC1',
                'SQM': 2000,
                'COUNTRY': 'South Africa',
                'TURNOVER': 50000000,
                'YEAR 1': 3000000,
                'YEAR 2': 4000000,
                'YEAR 3': 3000000,
                'TOTAL': 10000000,
                'ROI': 5.0,
                'SUSTAINABILITY': 1,
                'BUSINESS DEV': 0,
                'HSE': 1,
                'DIGITAL TRANSFORMATION': 1,
                'CSR': 0,
                'OPERATIONAL EXCELLENCE': 1
            }
            analyzer.add_project(new_project)

            budget_info = analyzer.check_budget_constraints()
            print("\n=== BUDGET INFORMATION ===")
            pprint.pprint(budget_info)

            summary = analyzer.generate_summary_report()
```

```
            print("\n=== SUMMARY REPORT ===")
            pprint.pprint(summary)

            export_paths = analyzer.export_results()
            print("\n=== EXPORTED FILES ===")
            pprint.pprint(export_paths)

            return analyzer

        except Exception as e:
            logger.error(f"Error in analysis: {e}")
            raise


    if __name__ == "__main__":
        run_analysis()
```

```
    OPERATIONAL EXCELLENCE
0                        1
1                        1
2                        1
3                        0
4                        0
5                        1
6                        1
7                        1
8                        1
9                        0
10                       1
11                       0
12                       1
13                       0
14                       1

=== RANKED PROJECTS ===
    PROJECT NUMBER PROJECT DESCRIPTION  RANK  STRATEGIC SCORE   ROI
0                1           Project 1     1                3  8.77
1                2           Project 2     2                3  6.61
2                3           Project 3     3                5  3.98
3                4           Project 4     4                2  1.57
4                5           Project 5     5                2  3.80
5                6           Project 6     6                4  3.93
6                7           Project 7     7                2  7.57
7                8           Project 8     8                5  6.74
8                9           Project 9     9                2  8.98
9               10          Project 10    10                0  5.25
10              11          Project 11    11                3  2.08
11              12          Project 12    12                4  7.42
12              13          Project 13    13                2  7.85
13              14          Project 14    14                2  6.05
14              15          Project 15    15                3  7.94

=== TOP ROI PROJECTS ===
    PROJECT NUMBER PROJECT DESCRIPTION   ROI
8                9           Project 9  8.98
0                1           Project 1  8.77
14              15          Project 15  7.94
12              13          Project 13  7.85
6                7           Project 7  7.57

=== BUDGET INFORMATION ===
{'Budget Remaining': -31929865.0,
 'Total Budget Used': 32929865.0,
 'Within Budget': np.False_}

=== SUMMARY REPORT ===
{'Average ROI': 5.8462499999999995,
 'Budget Status': {'Budget Remaining': -31929865.0,
                   'Total Budget Used': 32929865.0,
                   'Within Budget': np.False_},
 'Top Strategic Project': 'Project 3',
 'Total Projects': 16}

=== EXPORTED FILES ===
{'ranked_projects': './results/ranked_projects.csv'}
```