CONSULATION REPORT

TOPIC:

Personal finance and budgeting machine learning model that can help users manage their income, expenses, and savings.

OBJECTIVES:

Track income and expenses: Your code allows users to input their monthly income and expenses across various categories, which helps track their financial situation.

Set and monitor savings goals: The code includes functionality for users to input their savings goals, and it calculates the remaining balance to show whether they are on track to meet these goals.

Provide tailored recommendations: Based on the user's income, expenses, and remaining balance, the code suggests if the user needs to adjust their budget to meet savings goals.

Real-time feedback: The code provides instant results once the income, expenses, and savings goals are input, showing users their financial summary.

FEATURES:

1. Income Input:

Description: The user can input their total monthly income, which serves as the basis for financial calculations.

Benefit: It helps track monthly earnings and gives a clear understanding of available funds.

2. Expense Categorization:

Description: Users can input expenses across multiple categories (like rent, groceries, entertainment, etc.).

Benefit: Allows for detailed tracking of where money is spent, providing insights into spending habits.

3. Savings Goal Input:

Description: Users can set a savings goal for the month.

Benefit: It helps users prioritize savings and ensures that the code checks if the spending aligns with the savings target.

4. Automatic Calculation of Total Expenses:

Description: The code automatically sums all the expenses entered by the user.

Benefit: Simplifies the process of managing expenses by removing the need for manual calculations.

5. Remaining Balance Calculation:

Description: After subtracting expenses from income, the code calculates and displays the remaining balance.

Benefit: Provides a quick view of how much money is left after expenses, helping users see their financial status at a glance.

6. Savings Goal Achievement Check:

Description: The code checks if the remaining balance is sufficient to meet the user's savings goal.

Benefit: It helps users evaluate whether they need to cut down on expenses or adjust their savings target.

7. Real-Time Feedback:

Description: The program gives immediate feedback on whether the user is meeting their savings goal or if adjustments are needed.

Benefit: It encourages users to take action to meet their financial goals, offering a prompt reminder to adjust spending if necessary.

8. Customizable Expense Categories:

Description: Users can input as many expense categories as they need.

Benefit: The flexibility allows the tool to adapt to a wide range of personal budgeting needs.

9. User-Friendly Input and Output:

Description: The program uses simple input() prompts for users to enter data and presents a clean summary of income, expenses, and savings.

Benefit: It's easy to use, even for those without much technical knowledge, making it accessible to a wide range of users.

10. Potential for Future Enhancements:

Description: The code can be easily extended to include more advanced features, such as recurring expenses, data storage, or even integration with financial APIs.

Benefit: The structure allows for growth as users' financial needs evolve.

EXISTING PROBLEMS & ENHANCED SOLUTIONS:

Existing problems in Excel for personal finance budgeting, and how your Python code can solve them:

1. Manual Data Entry and Repetitive Tasks:

Problem in Excel: Excel requires users to manually input data and repeat calculations each time they want to update their budget, which can be tedious.

Solution in Code: The Python code simplifies the process by allowing users to input their income, expenses, and savings goals in an intuitive way. You can easily extend this code to automatically store or retrieve data for repeated use, reducing manual effort.

2. Lack of Automation for Savings Insights:

Problem in Excel: In Excel, users must manually calculate if they are on track to meet their savings goals, and there's no built-in alert system for adjustments.

Solution in Code: Your code automatically checks if the user's remaining balance meets the savings goal and provides feedback in real time, reducing the need for manual calculations.

3. Complex Formulas and Error-Prone:

Problem in Excel: For complex budgets, users may need to create custom formulas, which can be error-prone, especially for those without advanced Excel skills.

Solution in Code: Python allows you to handle calculations behind the scenes, without requiring users to write or troubleshoot formulas. The process becomes more user-friendly and less prone to errors.

4. Lack of Real-Time Recommendations:

Problem in Excel: Excel provides no automatic recommendations or insights based on the data input; users need to interpret the data themselves.

Solution in Code: Python can be programmed to analyze spending behavior and give actionable feedback (like suggesting expense adjustments), offering real-time recommendations that Excel doesn’t provide natively.

5. Data Handling for Multiple Categories:

Problem in Excel: Managing multiple categories of expenses can become cumbersome with Excel’s grid-based system, especially when updating and adding new categories.

Solution in Code: Python can handle dynamic input more flexibly, and new expense categories can be added seamlessly without needing to reformat spreadsheets.

6. Lack of Scalability:

Problem in Excel: Excel becomes harder to manage as datasets grow larger. Advanced users may face performance issues when working with complex budgets over time.

Solution in Code: Python can handle large datasets more efficiently, allowing scalability. If the code is expanded to include data storage in databases, it could be more scalable than Excel for large financial datasets.

7. Limited Customization for Users:

Problem in Excel: While Excel offers customizations, it can be difficult for users to tailor spreadsheets exactly to their financial needs without advanced knowledge.

Solution in Code: Python can easily be customized to fit specific user preferences (such as changing budget categories, setting notifications, or building personalized recommendations), making it more adaptable to individual users.

CODE:

```
def main():
    # Input: Getting user's income
    income = float(input("Enter your total monthly income: "))

    # Input: Getting user's expenses
    num_expenses = int(input("Enter the number of expense categories: "))
    total_expenses = 0.0
    expense_categories = []  # List to hold category names
    expenses = []  # List to hold expense amounts

    # Loop to get expenses and category names
    for i in range(1, num_expenses + 1):
        category_name = input(f"Enter name for expense category {i}: ")
        expense = float(input(f"Enter expense for {category_name}: "))
        expense_categories.append(category_name)  # Add category name to the list
        expenses.append(expense)  # Add expense to the list
        total_expenses += expense  # Add to total expenses

    # Input: Getting savings goal
    savings_goal = float(input("Enter your savings goal for the month: "))

    # Calculate remaining balance
    remaining_balance = income - total_expenses

    # Display the results
    print("\n--- Financial Summary ---")
    print(f"Total Income: ${income:.2f}")
    print(f"Total Expenses: ${total_expenses:.2f}")
    print(f"Savings Goal: ${savings_goal:.2f}")
    print(f"Remaining Balance: ${remaining_balance:.2f}")

    # Display the detailed expenses
    print("\n--- Detailed Expenses ---")
    for i in range(num_expenses):
        print(f"{expense_categories[i]}: ${expenses[i]:.2f}")
    # Check if the remaining balance meets the savings goal
    if remaining_balance >= savings_goal:
        print("Congratulations! You have enough to meet your savings goal!")
    else:
        print("You need to adjust your budget to meet your savings goal.")

        # Provide recommendations
        print("\n--- Recommendations ---")

        # Calculate the shortfall
        shortfall = savings_goal - remaining_balance

        # Provide personalized recommendations
        if total_expenses > income:
            print("Consider reducing your expenses. You are spending more than your income.")
            print("Here are some areas to review:")
            print("- Discretionary spending (e.g., dining out, entertainment)")
            print("- Subscriptions (e.g., streaming services, memberships)")
            print("- Impulse purchases")
        print(f"\nTo meet your savings goal, you need to cut back by:
${shortfall:.2f}")

        # Suggestions for savings
        print("\nSuggestions to increase savings:")
        print("- Set up an automatic transfer to your savings account.")
        print("- Look for additional sources of income (e.g., freelance
work, part-time jobs).")
        print("- Revisit your financial goals and adjust them if
necessary.")

        # Encourage regular monitoring of finances
        print("\nRegularly monitor your expenses and savings progress
to stay on track!")

if _name_ == "_main_":
    main()
```

CONSULATION:

The personal finance budgeting tool provides a simple yet effective way for users to manage their income, track expenses, and work towards their savings goals. By allowing customizable expense categories and real-time feedback on whether savings targets are being met, it encourages responsible financial planning. The flexibility of the code makes it adaptable to various user needs, and its intuitive design ensures that even those with minimal technical expertise can easily use it. This tool addresses common limitations found in traditional methods, such as manual calculations in spreadsheets, by offering automation and instant analysis. Overall, it promotes better financial awareness and decision-making, paving the way for improved personal budgeting.

P RAGA AMRUTHA RATNA – 2320030355

G SHEETAL – 2320030092