```
# Personal Finance Tracker for Google Colab

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import ipywidgets as widgets
from IPython.display import display, clear_output, HTML
import warnings
from google.colab import files
import io

warnings.filterwarnings('ignore')

# Initialize DataFrame to store transactions
transactions = pd.DataFrame(columns=['Date', 'Description', 'Amount', 'Category', 'Type'])

# Categories setup
income_categories = ['Salary', 'Freelance', 'Investments', 'Gifts', 'Other Income']
expense_categories = ['Food', 'Transportation', 'Housing', 'Entertainment',
                      'Healthcare', 'Education', 'Shopping', 'Utilities', 'Other Expenses']
all_categories = income_categories + expense_categories

# Category color map
category_colors = {
    'Salary': '#3498db', 'Freelance': '#9b59b6', 'Investments': '#2ecc71',
    'Gifts': '#e67e22', 'Other Income': '#1abc9c', 'Food': '#e74c3c',
    'Transportation': '#f39c12', 'Housing': '#34495e', 'Entertainment': '#d35400',
    'Healthcare': '#c0392b', 'Education': '#16a085', 'Shopping': '#8e44ad',
    'Utilities': '#7f8c8d', 'Other Expenses': '#95a5a6'
}
```

```
# Custom CSS
css = """
<style>
.finance-widget {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    background: linear-gradient(145deg, #ffffff, #f0f2f5);
    margin-bottom: 20px;
    width: 95%;
}
.finance-header {
    color: #2c3e50;
    font-size: 24px;
    font-weight: 600;
    margin-bottom: 15px;
    text-align: center;
}
.transaction-table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 15px;
}
.transaction-table th {
    background: #3498db;
    color: white;
    padding: 10px;
    text-align: left;
}
.transaction-table td {
    padding: 10px;
```

```
    border-bottom: 1px solid #ecf0f1;
}
.income-amount {
    color: #27ae60;
    font-weight: 500;
}
.expense-amount {
    color: #e74c3c;
    font-weight: 500;
}
</style>
"""
display(HTML(css))


def add_transaction(date, description, amount, category, transaction_type):
    global transactions
    new_transaction = pd.DataFrame([[date, description, amount, category,
transaction_type]],
                                   columns=['Date', 'Description', 'Amount', 'Category', 'Type'])
    transactions = pd.concat([transactions, new_transaction], ignore_index=True)
    transactions['Date'] = pd.to_datetime(transactions['Date'])
    transactions.sort_values('Date', inplace=True)
    display(HTML(f'<div style="color:green;padding:10px;">Transaction added: {description}
(${amount})</div>'))


def show_transactions(n=10):
    if transactions.empty:
        display(HTML('<div style="padding:20px;color:#7f8c8d;">No transactions yet</div>'))
    else:
        df = transactions.tail(n).copy()
        df['Amount'] = df.apply(lambda x: f'<span class="{"income-amount" if
x["Type"]=="Income" else "expense-amount"}">'
                                          f'{"+" if x["Type"]=="Income" else "-"}${x["Amount"]:,.2f}</span>',
axis=1)
```

```
        df['Date'] = df['Date'].dt.strftime('%Y-%m-%d')

        html = f"""
        <div class="finance-widget">
            <h3 class="finance-header">Recent Transactions</h3>
            <table class="transaction-table">
                <tr><th>Date</th><th>Description</th><th>Amount</th><th>Category</th></tr>
        """
        for _, row in df.iterrows():
            html += f"""
                <tr>
                    <td>{row['Date']}</td>
                    <td>{row['Description']}</td>
                    <td>{row['Amount']}</td>
                    <td style="color:{category_colors.get(row['Category'],
'#000')}">{row['Category']}</td>
                </tr>
            """
        html += "</table></div>"
        display(HTML(html))

def show_summary():
    if transactions.empty:
        display(HTML('<div style="padding:20px;color:#7f8c8d;">No transactions yet</div>'))
        return

    income = transactions[transactions['Type'] == 'Income']['Amount'].sum()
    expenses = transactions[transactions['Type'] == 'Expense']['Amount'].sum()
    balance = income - expenses

    html = f"""
    <div class="finance-widget">
        <h3 class="finance-header">Financial Summary</h3>
```

```
        <div style="display:flex;justify-content:space-around;">
            <div style="text-align:center;">
                <div style="font-size:16px;color:#27ae60;">Income</div>
                <div style="font-size:24px;">${income:,.2f}</div>
            </div>
            <div style="text-align:center;">
                <div style="font-size:16px;color:#e74c3c;">Expenses</div>
                <div style="font-size:24px;">${expenses:,.2f}</div>
            </div>
            <div style="text-align:center;">
                <div style="font-size:16px;color:{"#27ae60" if balance>=0 else
"#e74c3c"}">Balance</div>
                <div style="font-size:24px;">${balance:,.2f}</div>
            </div>
        </div>
    </div>
    """
    display(HTML(html))

def save_data():
    if transactions.empty:
        display(HTML('<div style="color:red;padding:10px;">No data to save!</div>'))
        return
    transactions.to_csv("transactions.csv", index=False)
    files.download("transactions.csv")

def load_data():
    global transactions
    uploaded = files.upload()
    for filename in uploaded.keys():
        try:
            transactions = pd.read_csv(io.BytesIO(uploaded[filename]))
            transactions['Date'] = pd.to_datetime(transactions['Date'])
```

```
            display(HTML(f'<div style="color:green;padding:10px;">Loaded {len(transactions)} transactions</div>'))
        except Exception as e:
            display(HTML(f'<div style="color:red;padding:10px;">Error loading file: {str(e)}</div>'))

# UI elements
date_picker = widgets.DatePicker(description='Date:', value=datetime.now())
desc_input = widgets.Text(description='Description:')
amount_input = widgets.FloatText(description='Amount:', value=0.0)
type_toggle = widgets.ToggleButtons(options=['Income', 'Expense'], description='Type:')
category_dropdown = widgets.Dropdown(options=expense_categories, description='Category:')

def update_categories(change):
    category_dropdown.options = income_categories if change['new'] == 'Income' else expense_categories
type_toggle.observe(update_categories, names='value')

add_button = widgets.Button(description='Add Transaction', button_style='success')
refresh_button = widgets.Button(description='Refresh View')
save_button = widgets.Button(description='Save Data')
load_button = widgets.Button(description='Load Data')

def on_add_click(b):
    add_transaction(
        date_picker.value,
        desc_input.value,
        amount_input.value,
        category_dropdown.value,
        type_toggle.value
    )
    desc_input.value = ''
    amount_input.value = 0.0
```

```
def on_refresh_click(b):
    clear_output(wait=True)
    display(HTML(css))
    show_main_ui()

def on_save_click(b):
    save_data()

def on_load_click(b):
    load_data()

add_button.on_click(on_add_click)
refresh_button.on_click(on_refresh_click)
save_button.on_click(on_save_click)
load_button.on_click(on_load_click)

def show_main_ui():
    display(HTML("<h1 style='color:#2c3e50;'>Personal Finance Tracker</h1>"))
    input_box = widgets.VBox([
        widgets.HBox([date_picker, type_toggle]),
        desc_input,
        widgets.HBox([amount_input, category_dropdown]),
        widgets.HBox([add_button, refresh_button, save_button, load_button])
    ])
    display(input_box)
    show_summary()
    show_transactions()

# Run the UI
show_main_ui()
```