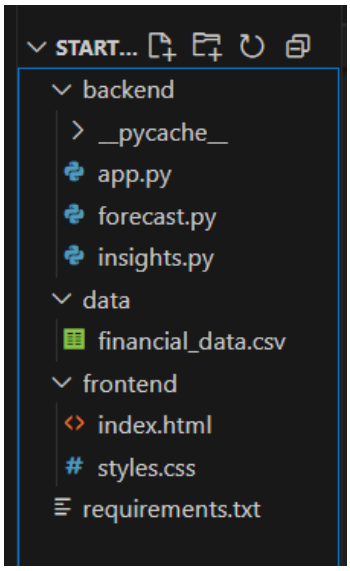
BUDGET PLANNER FOR STARTUP



App.py:

```
from flask import Flask, jsonify, request, send_from_directory

app = Flask(__name__, static_folder='../frontend')

@app.route('/')
def index():
    return send_from_directory(app.static_folder, 'index.html')

@app.route('/styles.css')
def styles():
    return send_from_directory(app.static_folder, 'styles.css')

@app.route('/api/recommendation', methods=['POST'])
def recommendation():
    data = request.json
    expenses = data.get('expenses', 0)
```

```
    try:
        expenses = float(expenses)
    except ValueError:
        return jsonify({"error": "Invalid expense value"}), 400

    if expenses <= 0:
        return jsonify({"error": "Expense must be greater than zero"}), 400

    # Strategic budget calculation: add savings margin
    recommended_budget = expenses   # 10% savings buffer

    allocations = {
        "Operations": 0.30,
        "Marketing": 0.20,
        "Salaries": 0.25,
        "Research & Development": 0.10,
        "Miscellaneous": 0.05,
        "Savings": 0.10
    }

    breakdown = {k: round(recommended_budget * v, 2) for k, v in allocations.items()}

    if expenses > recommended_budget:
        message = f"Expenses are high. Try reducing them to stay under ₹{recommended_budget:.2f} to allow for savings and better financial stability."
    else:
        message = "Your current expenses are healthy. Keep maintaining this balance for consistent growth."

    return jsonify({
        "total_budget": round(recommended_budget, 2),
        "breakdown": breakdown,
```

```
        "recommendation": message
    })

if __name__ == '__main__':
    app.run(debug=True)
```

forecast.py:

```
import pandas as pd
import matplotlib.pyplot as plt

# Reading the dataset from a CSV string (you can replace this part with reading from a file if you have one)
from io import StringIO

data = """
data,ds,marketing,salaries,operations
1990-01-01,2099,7521,3234
1990-02-01,1977,7830,2873
1990-03-01,2139,8021,3029
1990-04-01,2319,8044,2891
1990-05-01,1973,7905,2901
1990-06-01,1978,8236,2935
1990-07-01,2345,7739,3070
1990-08-01,2188,8027,2897
1990-09-01,1946,8116,3333
1990-10-01,2153,8244,2798
1990-11-01,1957,8313,2978
1990-12-01,1961,7772,2917
1991-01-01,2108,7660,3391
1991-02-01,1682,8513,3081
1991-03-01,1725,8240,3293
1991-04-01,1962,7925,2666
```

1991-05-01,1877,8625,3106

1991-06-01,2148,8205,3265

1991-07-01,1908,8534,3065

1991-08-01,1812,8210,3313

1991-09-01,2393,8818,2920

1991-10-01,2060,8737,3404

1991-11-01,2123,8145,3629

1991-12-01,1830,8522,2966

1992-01-01,2011,8434,2948

1992-02-01,2147,8661,3426

1992-03-01,1900,7971,3460

1992-04-01,2210,8476,2936

1992-05-01,2020,8598,2965

1992-06-01,2087,7763,3002

1992-07-01,2030,7945,2739

1992-08-01,2525,7699,2848

1992-09-01,2157,8239,2829

1992-10-01,1953,8546,2890

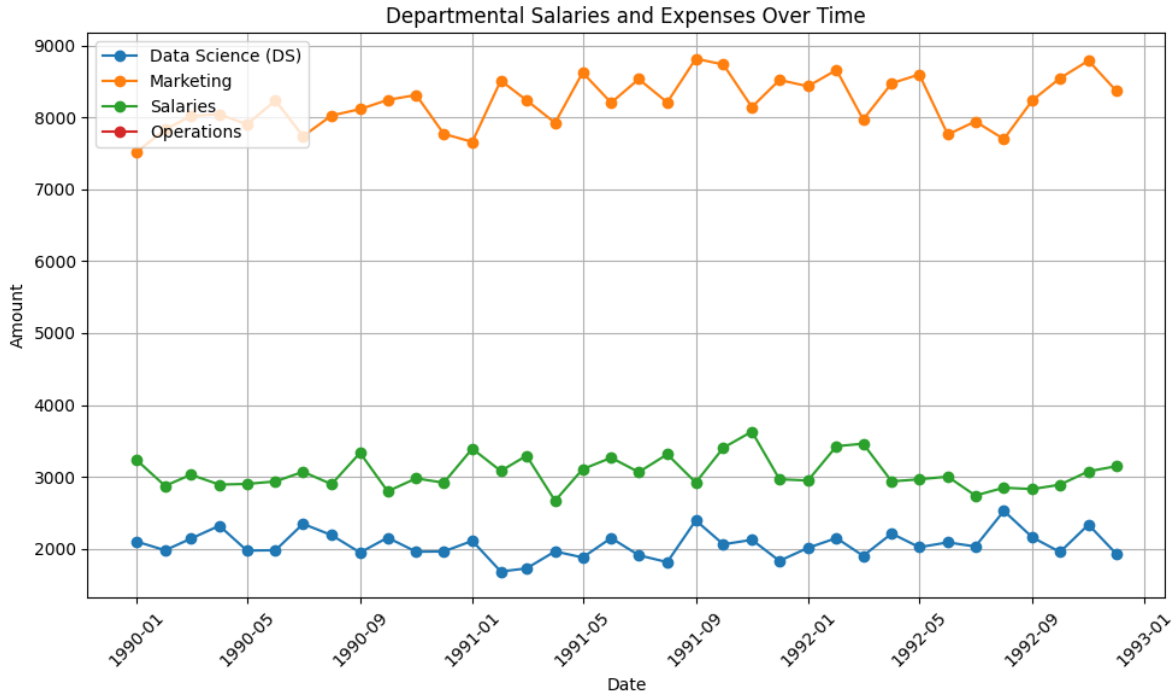
1992-11-01,2334,8791,3076

1992-12-01,1931,8373,3146

```
df = pd.read_csv(StringIO(data), parse_dates=['data'])
plt.figure(figsize=(10, 6))
plt.plot(df['data'], df['ds'], label='Data Science (DS)', marker='o')
plt.plot(df['data'], df['marketing'], label='Marketing', marker='o')
plt.plot(df['data'], df['salaries'], label='Salaries', marker='o')
plt.plot(df['data'], df['operations'], label='Operations', marker='o')
plt.title('Departmental Salaries and Expenses Over Time')
plt.xlabel('Date')
plt.ylabel('Amount')
plt.legend(loc='upper left')
```

```
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

OUTPUT:



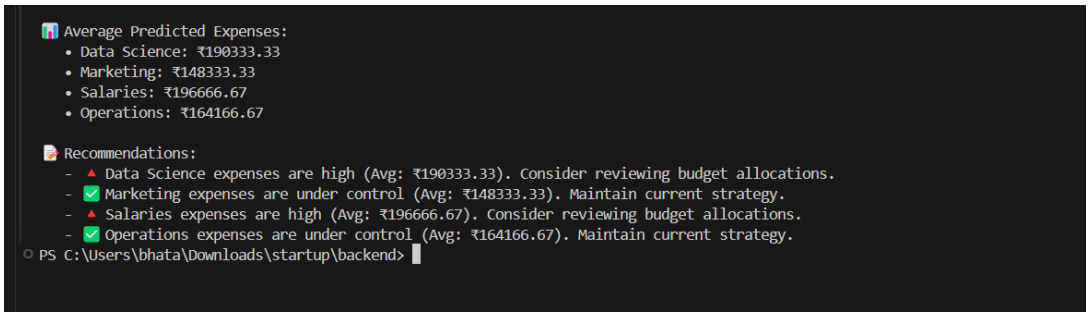
insights.py:

```
from pprint import pprint


def generate_insights(forecast_df):
    latest = forecast_df.tail(6)
    average_expenses = {}

    departments = ['Data Science', 'Marketing', 'Salaries', 'Operations']

    for department in departments:
        avg_expense = latest[department].mean()
        average_expenses[department] = round(avg_expense, 2)
```

```
    # Generate recommendations based on the average expenses for each department
    recommendations = []

    for department, avg_expense in average_expenses.items():
        if avg_expense > 180000:
            recommendations.append(f" 🔺 {department} expenses are high (Avg: ₹{avg_expense}). Consider reviewing budget allocations.")
        else:
            recommendations.append(f" ✅ {department} expenses are under control (Avg: ₹{avg_expense}). Maintain current strategy.")

    # Pretty formatted output
    print("\n📊 Average Predicted Expenses:")
    for dept, expense in average_expenses.items():
        print(f"  • {dept}: ₹{expense}")

    print("\n📝 Recommendations:")
    for rec in recommendations:
        print(f"  - {rec}")

    return {
        "average_predicted_expenses": average_expenses,
        "recommendations": recommendations
    }

import pandas as pd

data = {
    'Data Science': [190000, 195000, 200000, 180000, 185000, 192000],
    'Marketing': [150000, 145000, 155000, 140000, 152000, 148000],
    'Salaries': [200000, 210000, 190000, 180000, 195000, 205000],
```

```
    'Operations': [170000, 175000, 160000, 165000, 155000, 160000]
}

forecast_df = pd.DataFrame(data)

generate_insights(forecast_df)
```

OUTPUT:



Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Smart Financial Planner</title>
    <link rel="stylesheet" href="styles.css">
    <style>
        /* Optional styling for hover effect */
        button:hover {
            opacity: 0.9;
            transform: scale(1.02);
            transition: all 0.2s ease-in-out;
        }
    </style>
</head>
<body>
```

```
  <h1>Smart Financial Planner</h1>

  <div class="section">
    <h2>Enter Total Monthly Company Expenses (₹)</h2>
    <input type="number" id="expenses" placeholder="e.g., 500000" />
    <button onclick="submitBudget()">Get Budget Allocation</button>
  </div>

  <div id="recommendation" class="section"></div>

  <!-- Chart Controls -->
  <div class="section" id="chartControls" style="display:none;">
    <label for="chartType">Select Chart Type:</label>
    <select id="chartType" onchange="plotChart()">
      <option value="bar">Bar Chart</option>
      <option value="line">Line Chart</option>
    </select>
    <button id="plotBtn" onclick="plotChart()">Show Spending Chart</button>
  </div>

  <!-- Chart Display Section -->
  <div id="chartSection" class="section" style="display:none;">
    <h3>Spending Breakdown Chart</h3>
    <canvas id="spendingChart" width="400" height="300"></canvas>
  </div>

  <!-- Prophet Model Button -->
  <div class="section" style="text-align: center; margin-top: 30px;">
    <button onclick="window.open('https://colab.research.google.com/drive/1Rg3IPYyI-TY3HtM6x85KHHZUSeCURMzB')"
            style="background-color: #8e44ad; color: white; padding: 10px 20px; font-size: 16px; border: none; border-radius: 8px; cursor: pointer;">
```

```
        Open Prophet Model
    </button>
</div>

<!-- Chart.js CDN -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>
    let latestBreakdown = null;
    let myChart = null;

    function submitBudget() {
        const expenses = parseFloat(document.getElementById('expenses').value);

        if (isNaN(expenses) || expenses <= 0) {
            alert("Please enter a valid amount.");
            return;
        }

        fetch('/api/recommendation', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({ expenses })
        })
        .then(res => res.json())
        .then(data => {
            const breakdown = data.breakdown;
            latestBreakdown = breakdown;

            const breakdownList = Object.entries(breakdown)
                .map(([key, val]) => `<li><strong>${key}:</strong> ₹${val}</li>`)
```

```
            .join('');

        document.getElementById('recommendation').innerHTML = `
            <h3>Suggested Budget Allocation:</h3>
            <p><strong>Total Recommended Budget:</strong> ₹${data.total_budget}</p>
            <ul>${breakdownList}</ul>
            <p class="advice">${data.recommendation}</p>
        `;

        document.getElementById('chartControls').style.display = 'block';
    })
    .catch(err => {
        console.error(err);
        alert("Something went wrong while getting recommendation.");
    });
}

function plotChart() {
    const ctx = document.getElementById('spendingChart').getContext('2d');
    const breakdown = latestBreakdown;
    const selectedType = document.getElementById('chartType').value;

    if (!breakdown) {
        alert("No data to plot!");
        return;
    }

    if (myChart) {
        myChart.destroy();
    }
```

```
    const labels = Object.keys(breakdown);
    const values = Object.values(breakdown);

    myChart = new Chart(ctx, {
        type: selectedType,
        data: {
            labels: labels,
            datasets: [{
                label: 'Spending (₹)',
                data: values,
                backgroundColor: selectedType === 'bar'
                    ? ['#3498db', '#2ecc71', '#f1c40f', '#9b59b6', '#e67e22', '#e74c3c']
                    : 'rgba(75, 192, 192, 0.2)',
                borderColor: '#3498db',
                borderWidth: 2,
                tension: 0.3,
                fill: selectedType === 'line'
            }]
        },
        options: {
            responsive: true,
            scales: {
                y: {
                    beginAtZero: true
                }
            }
        }
    });

    document.getElementById('chartSection').style.display = 'block';
}
```

```
    </script>
</body>
</html>
```

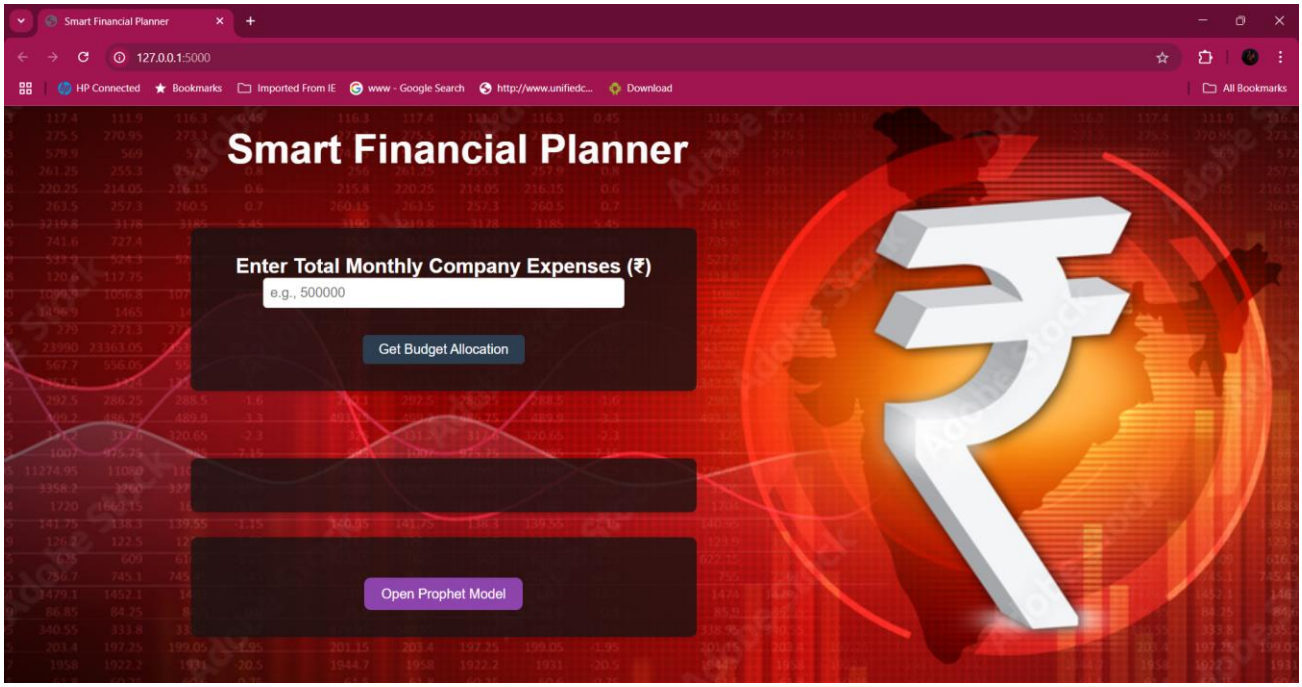
Styles.css:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    margin: 5;
    padding: 0;
    height: 100vh;
    width:150vh;
    background-image:
url('https://as1.ftcdn.net/v2/jpg/03/55/40/20/1000_F_355402060_F3JfUnBykuKlnY7sRg7n5DeiL8afdpMj.jpg');
    background-size: 115%;
    background-position: center;
    background-attachment: fixed;
    color: white;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;
}

h1 {
```

```
    color: rgb(252, 252, 252);
    font-size: 3rem;
    position: absolute;
    top: 1.5rem;
    left: 35%;
    transform: translateX(-50%);
}


.section {
    margin-top: 5rem;
    padding: 2rem;
    background-color: rgba(24, 19, 19, 0.8);
    border-radius: 8px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
    width: 105%;
    max-width: 600px;
    text-align: center;
}

input[type="number"] {
    width: 80%;
    padding: 0.5rem;
    font-size: 1rem;
    margin-bottom: 1rem;
    border: 1px solid #c5bfbf;
    border-radius: 5px;
}

button {
    padding: 0.5rem 1.2rem;
```

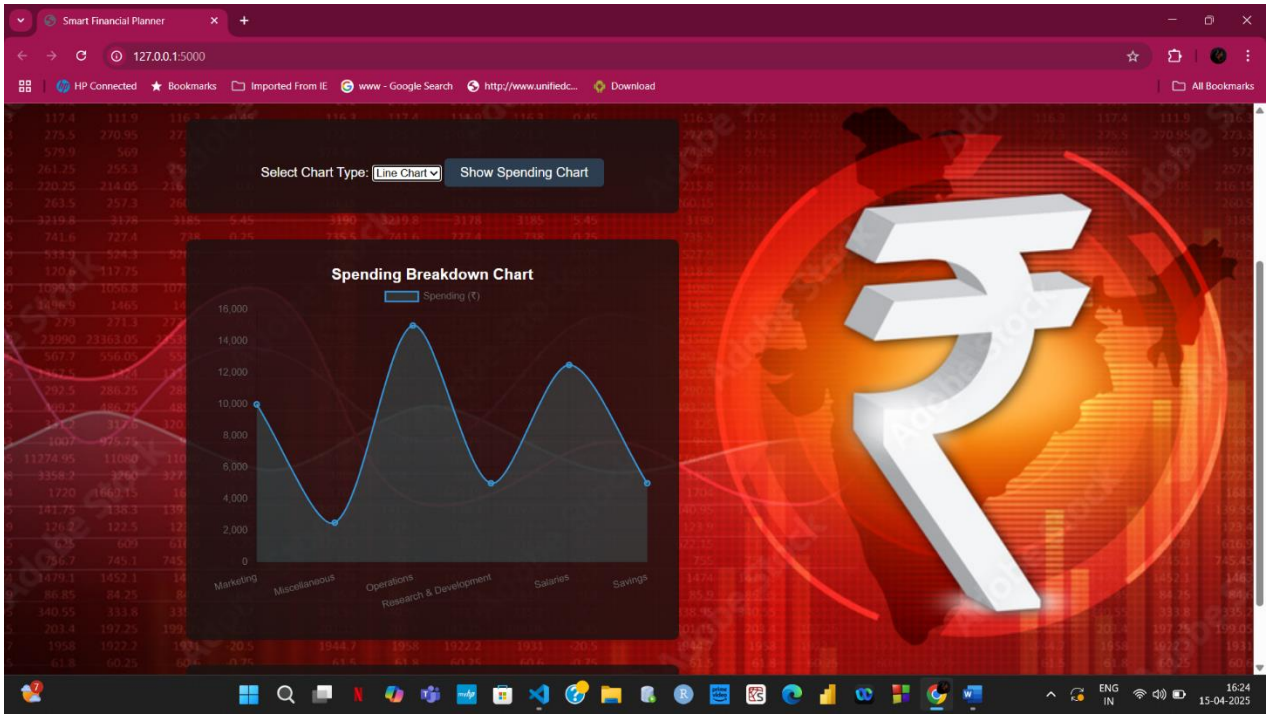
```
    font-size: 1rem;
    color: #fff;
    background-color: #2c3e50;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-top: 1rem;
}

button:hover {
    background-color: #1a252f;
}

ul {
    list-style: none;
    padding-left: 0;
}

li {
    margin: 0.5rem 0;
}

.advice {
    font-style: italic;
    margin-top: 1rem;
    color: #00796b;
}

#chartControls, #chartSection {
    margin-top: 2rem;
}
```

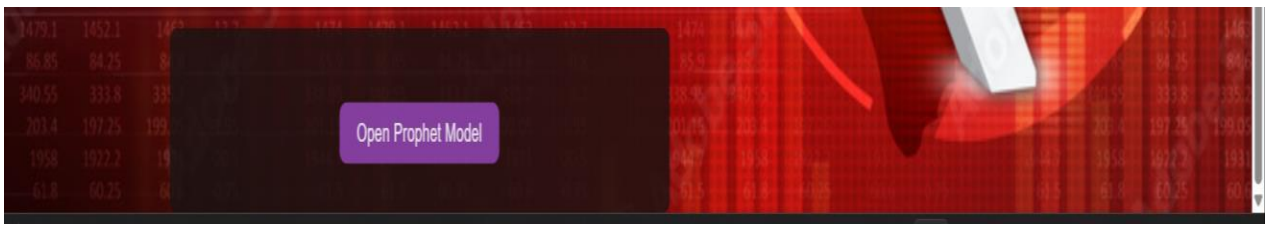
OUTPUT:


Smart Financial Planner
127.0.0.1:5000
Smart Financial Planner
Enter Total Monthly Company Expenses (₹)
e.g., 500000
Get Budget Allocation
Open Prophet Model


Smart Financial Planner
127.0.0.1:5000
50000
Smart Financial Planner
Get Budget Allocation
Suggested Budget Allocation:
Total Recommended Budget: ₹50000
Marketing: ₹10000
Miscellaneous: ₹2500
Operations: ₹15000
Research & Development: ₹5000
Salaries: ₹12500
Savings: ₹5000
Your current expenses are healthy. Keep maintaining this balance for consistent growth.
Select Chart Type: Bar Chart
Show Spending Chart


Smart Financial Planner
127.0.0.1:5000
HP Connected
Bookmarks
Imported From IE
www - Google Search
http://www.unifiedc...
Download
All Bookmarks
Select Chart Type: Bar Chart
Show Spending Chart
Spending Breakdown Chart
Spending (₹)
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
Marketing
Miscellaneous
Operations
Research & Development
Salaries
Savings


Smart Financial Planner
127.0.0.1:5000
HP Connected
Bookmarks
Imported From IE
www - Google Search
http://www.unifiedc...
Download
All Bookmarks
Select Chart Type: Line Chart
Show Spending Chart
Spending Breakdown Chart
Spending (₹)
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
Marketing
Miscellaneous
Operations
Research & Development
Salaries
Savings
ENG
IN
16:24
15-04-2025


Open Prophet Model

Install dependencies in your Colab environment:

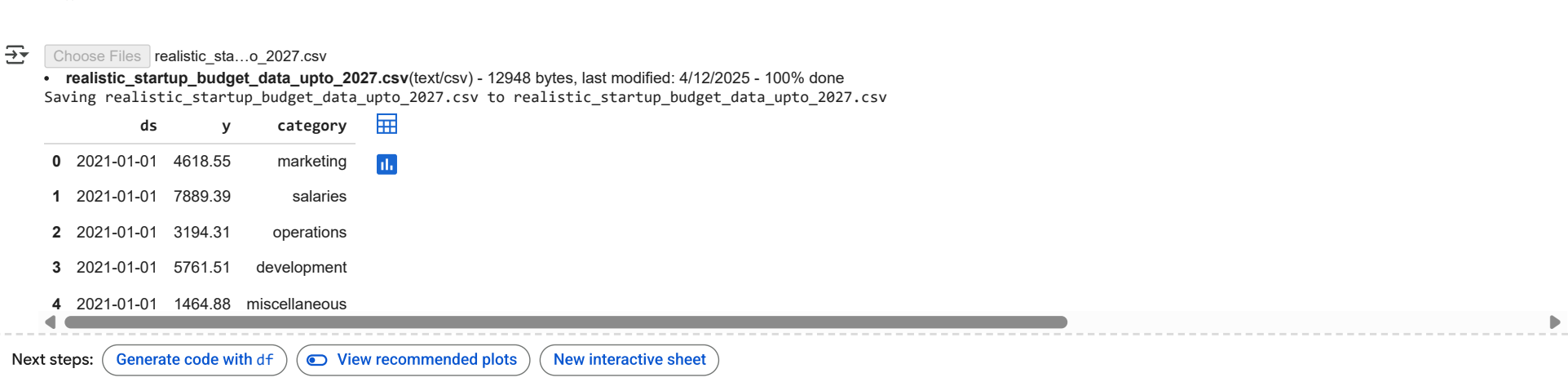
```
!pip install prophet pandas matplotlib ipywidgets
```

```
Requirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (24.0.1)
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.11/dist-packages (from ipykernel>=4.5.1->ipywidgets) (6.4.2)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (75.2.0)
Collecting jedi>=0.16 (from ipython>=4.0.0->ipywidgets)
  Downloading jedi-0.19.2-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (3.0.50)
Requirement already satisfied: pygments in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (2.18.0)
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=4.0.0->ipywidgets) (4.9.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.11/dist-packages (from widgetsnbextension~=3.6.0->ipywidgets) (6.5.7)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=4.0.0->ipywidgets) (0.8.4)
Requirement already satisfied: jupyter-core>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets) (5.7.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (3.1.6)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (23.1.0)
Requirement already satisfied: nbformat in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (5.10.4)
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (7.16.6)
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (1.8.3)
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (0.18.1)
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (0.21.1)
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.11/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (1.2.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=4.0.0->ipywidgets) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.11/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=4.0.0->ipywidgets) (0.2.13)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.11/dist-packages (from jupyter-core>=4.6.0->jupyter-client>=6.1.12->ipykernel>=4.5.1->ipywidgets)
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.11/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (4.13.3)
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->i
Requirement already satisfied: defusedxml in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (0.
Requirement already satisfied: markupsafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (3.0.2)
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (3.1.
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (0.10.2
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (1
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (2.21.
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.11/dist-packages (from nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (4.23.0)
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.11/dist-packages (from argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets) (21
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert>=5->notebook>=4.4.1->widgetsnbexten
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidge
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbext
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ip
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=2.6->nbformat->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidg
Requirement already satisfied: jupyter-server<3,>=1.8 in /usr/local/lib/python3.11/dist-packages (from notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbexte
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.6.0->ip
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywi
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3
Requirement already satisfied: pycparser in /usr/local/lib/python3.11/dist-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1->widgetsnbextension
Requirement already satisfied: anyio>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook>=4.4.1-
Requirement already satisfied: websocket-client in /usr/local/lib/python3.11/dist-packages (from jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->notebook>=4.
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->noteb
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio>=3.1.0->jupyter-server<3,>=1.8->notebook-shim>=0.2.3->nbclassic>=0.4.7->no
Downloading jedi-0.19.2-py2.py3-none-any.whl (1.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 12.4 MB/s eta 0:00:00
Installing collected packages: jedi
Successfully installed jedi-0.19.2
```

Double-click (or enter) to edit

Upload your historical budget data (must include a ds column for dates and y for amounts):

```
from google.colab import files
import pandas as pd

uploaded = files.upload()

# Read the uploaded CSV (replace with your file name if known)
for fn in uploaded.keys():
    df = pd.read_csv(fn)

# Show first few rows
df.head()
```



Choose Files realistic_sta…o_2027.csv

- **realistic_startup_budget_data_upto_2027.csv**(text/csv) - 12948 bytes, last modified: 4/12/2025 - 100% done

Saving realistic_startup_budget_data_upto_2027.csv to realistic_startup_budget_data_upto_2027.csv

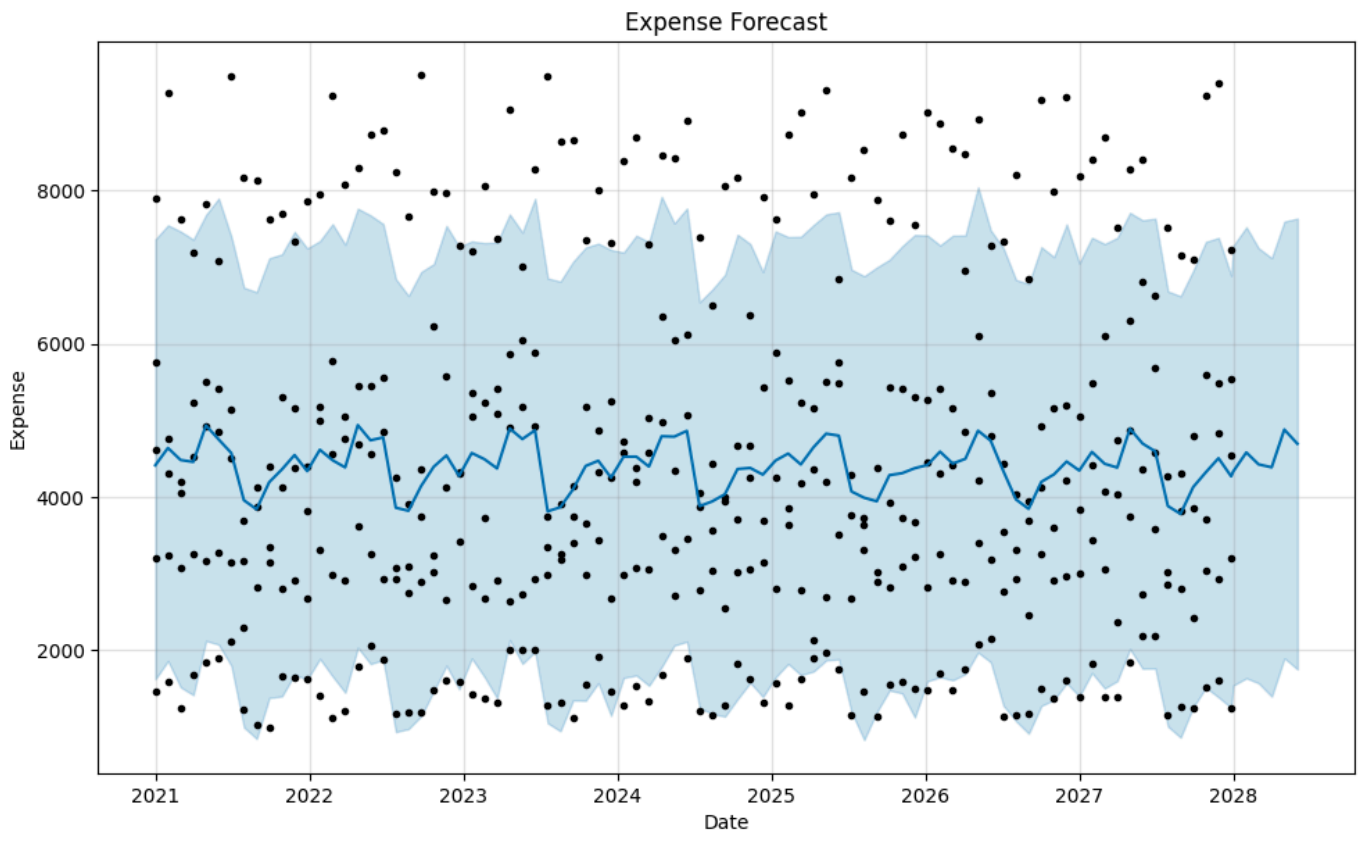
	ds	y	category
0	2021-01-01	4618.55	marketing
1	2021-01-01	7889.39	salaries
2	2021-01-01	3194.31	operations
3	2021-01-01	5761.51	development
4	2021-01-01	1464.88	miscellaneous

Next steps: Generate code with df | View recommended plots | New interactive sheet

Step 3: Prophet Forecasting Model Use Prophet to forecast future expenses:

```
from prophet import Prophet
import matplotlib.pyplot as plt

# Create and fit the model
model = Prophet()
model.fit(df[['ds', 'y']])

# Make future dataframe
future = model.make_future_dataframe(periods=6, freq='M')
forecast = model.predict(future)

# Plot the forecast
fig = model.plot(forecast)
plt.title("Expense Forecast")
plt.xlabel("Date")
plt.ylabel("Expense")
plt.show()
```

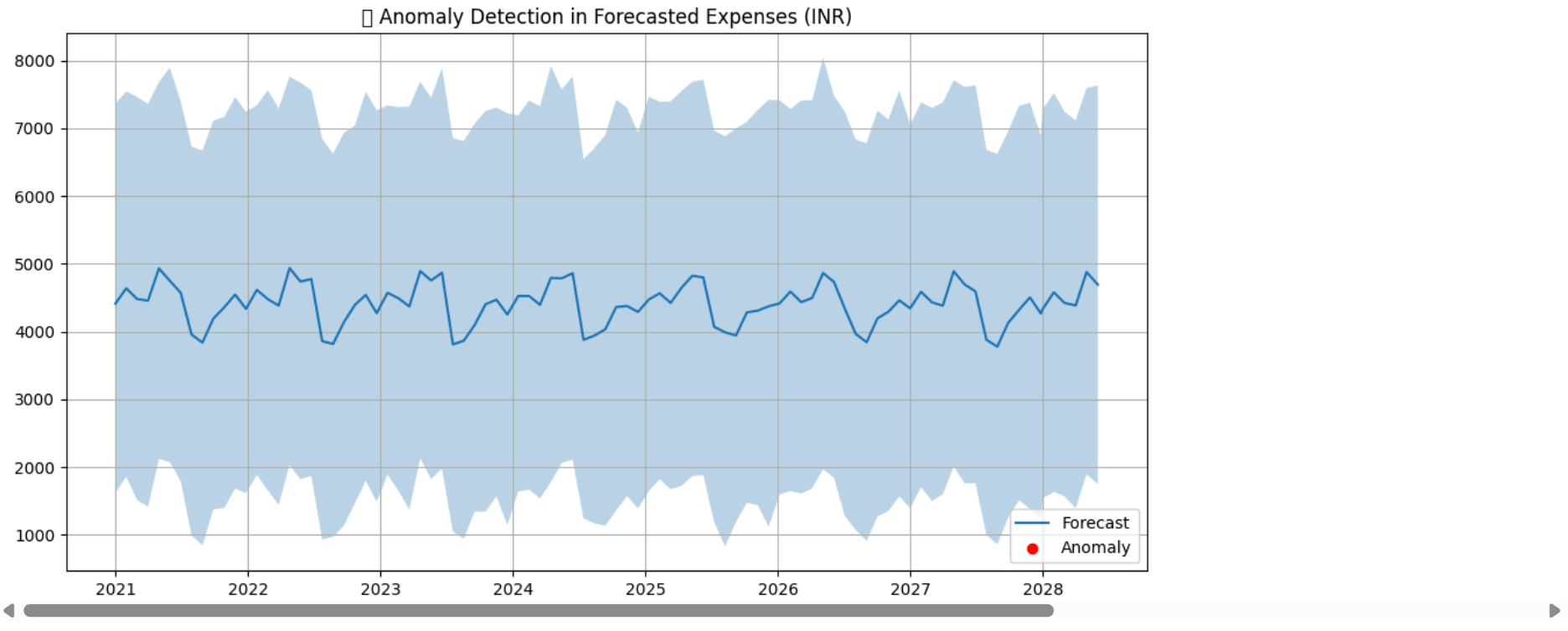
```
INFO:prophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmpui2bodne/_7cohumg.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpui2bodne/icqtnyjy.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=93891', 'data', 'file=/tmp/tmpui2bodne/_7cohumg.
15:01:49 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
15:01:49 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
/usr/local/lib/python3.11/dist-packages/prophet/forecaster.py:1854: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  dates = pd.date_range(
```



Step 4: Generate Recommendations Basic rules-based budget suggestions from forecast:

```
# Detect anomalies
threshold = forecast['yhat_upper'].std() * 0.5
forecast['anomaly'] = forecast['yhat'] > (forecast['yhat_upper'] + threshold)

# Plot anomalies
plt.figure(figsize=(12, 6))
plt.plot(forecast['ds'], forecast['yhat'], label='Forecast')
plt.fill_between(forecast['ds'], forecast['yhat_lower'], forecast['yhat_upper'], alpha=0.3)
plt.scatter(forecast[forecast['anomaly']]['ds'], forecast[forecast['anomaly']]['yhat'],
            color='red', label='Anomaly')
plt.title("🔍 Anomaly Detection in Forecasted Expenses (INR)")
plt.legend()
plt.grid()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
latest_expense = forecast.iloc[-6:][['ds', 'yhat']]
```

```
recommendations = []
for index, row in latest_expense.iterrows():
    if row['yhat'] > df['y'].mean():
        recommendations.append(f"⚠️ High forecasted expense in {row['ds'].strftime('%B %Y')}. Consider reducing discretionary spending.")
    else:
        recommendations.append(f"✅ Stable forecast for {row['ds'].strftime('%B %Y')}. Maintain current spending levels.")

for rec in recommendations:
    print(rec)
```

```
✅ Stable forecast for December 2027. Maintain current spending levels.
⚠️ High forecasted expense in January 2028. Consider reducing discretionary spending.
⚠️ High forecasted expense in February 2028. Consider reducing discretionary spending.
⚠️ High forecasted expense in March 2028. Consider reducing discretionary spending.
⚠️ High forecasted expense in April 2028. Consider reducing discretionary spending.
⚠️ High forecasted expense in May 2028. Consider reducing discretionary spending.
```

Simple Frontend Interface Using ipywidgets to get feedback and simulate a UI:

```
import ipywidgets as widgets
from IPython.display import display

print("\n💬 Budget Recommendations:")
for rec in recommendations:
    print("•", rec)

print("\n📣 Please provide your feedback:")

feedback = widgets.Textarea(
    value='',
    placeholder='Type your feedback here...',
    description='Feedback:',
    layout=widgets.Layout(width='100%', height='100px')
)
display(feedback)
```

```
submit_button = widgets.Button(description="Submit Feedback")

def handle_submit(b):
    print("\n✅ Thank you for your feedback:")
    print(feedback.value)

submit_button.on_click(handle_submit)
display(submit_button)
```

```
💬 Budget Recommendations:
• ✅ Stable forecast for December 2027. Maintain current spending levels.
• ⚠️ High forecasted expense in January 2028. Consider reducing discretionary spending.
• ⚠️ High forecasted expense in February 2028. Consider reducing discretionary spending.
• ⚠️ High forecasted expense in March 2028. Consider reducing discretionary spending.
• ⚠️ High forecasted expense in April 2028. Consider reducing discretionary spending.
• ⚠️ High forecasted expense in May 2028. Consider reducing discretionary spending.

📣 Please provide your feedback:
Feedback: Type your feedback here...
Submit Feedback
```

```
with open('user_feedback.txt', 'a') as f:
    f.write(feedback.value + "\n")
```

```
# Recommend monthly budget
latest_avg = forecast.tail(12)['yhat'].mean()
suggested = latest_avg * 0.9  # Suggest 10% buffer

# Display values in INR format
def format_inr(value):
    return f"₹{value:,.2f}"
```

```
print(f"📌 Forecasted Avg. Monthly Expense (next 12 months): {format_inr(latest_avg)}")
print(f"💡 Suggested Monthly Budget (10% lower): {format_inr(suggested)}")
```

```
📌 Forecasted Avg. Monthly Expense (next 12 months): ₹4,346.78
💡 Suggested Monthly Budget (10% lower): ₹3,912.10
```