```
# Uses the server database and a query that accepts 'timeframe'  (hours/week/
month) as input

import pyodbc
import pandas as pd
from tabulate import tabulate
from termcolor import colored
from colored import fg
from flask import Flask, render_template, request

app = Flask(__name__)
server = 'tp-dev-sql.database.windows.net'  # Replace with your server name or
IP
database = 'Staging_Web_Interactions'  # Replace with your database name
username = 'sqladmin'  # Replace with your username
password = 'TPDon#2024'  # Replace with your passworD
# server = 'localhost\SQLEXPRESS'  # Replace with your server name or IP
# server = r'localhost\SQLEXPRESS'
# username = 'sqladmin'  # Replace with your username
# password = ''  # Replace with your passworD

@app.route('/')
def display_data():
    try:
        # Create connection string
        connection_string = f"DRIVER={{ODBC Driver 17 for SQL Server}};
SERVER={server};DATABASE={database};UID={username};PWD={password
}"

        # Establish the connection
        connection = pyodbc.connect(connection_string)

        # Create a cursor to execute SQL queries
        cursor = connection.cursor()
        print(colored("Connection to SQL Server database established
successfully.", "green"))
        print("Connection to SQL Server database established successfully.")
        # timeframe = 'week'
        timeframe =  input ("Enter the timeframe (hours/week/prev_week/month
) : ") #timeframe
        query = """
            DECLARE @timeframe NVARCHAR(50) = ?;
            DECLARE @month_start DATE;
            DECLARE @month_end DATE;
            DECLARE @prev_week_start DATE;
            DECLARE @prev_week_end DATE;
```

```
            DECLARE @week_start DATE;
            DECLARE @week_end DATE;

            SET @month_start = DATEADD(MONTH, -1, GETDATE());
            SET @month_end = GETDATE();
            SET @prev_week_start = DATEADD(WEEK, -2, GETDATE());
            SET @prev_week_end = DATEADD(WEEK, -1, GETDATE());
            SET @week_start = DATEADD(WEEK, -1, GETDATE());
            SET @week_end = GETDATE();

            -- Declined Quotes Count
            WITH DeclinedQuoteCounts AS (
              SELECT
                am.AgencyName,
                COUNT(DISTINCT fvd.QuoteNumber) AS
  DeclinedQuoteCount
              FROM FlattenPageViewData fvd
              JOIN AgencyMapping am
                ON fvd.UserId = am.AgentName
              WHERE fvd.QuoteNumber IN (
                -- Subquery returns only QuoteNumbers with 'Declined' status
                SELECT DISTINCT QuoteNumber
                FROM FlattenPageViewData
                WHERE QuoteStatus = 'Declined'
              )
              AND (
                (@timeframe = 'hours' AND SaveDateTime >= DATEADD(
  HOUR, -24, GETDATE()))
                OR (@timeframe = 'prev_week' AND CONVERT(DATE,
  SaveDateTime) BETWEEN @prev_week_start AND @prev_week_end)
                OR (@timeframe = 'week' AND CONVERT(DATE,
  SaveDateTime) BETWEEN @week_start AND @week_end)
                OR (@timeframe = 'month' AND CONVERT(DATE,
  SaveDateTime) BETWEEN @month_start AND @month_end)
              )
              GROUP BY am.AgencyName
            ),
            -- UW Block Counts
            uwBlockCounts AS (
              SELECT
                am.AgencyName,
                COUNT(DISTINCT sq5.QuoteNumber) AS UWBlockCount
              FROM (
                SELECT DISTINCT UserId, QuoteNumber
                FROM FlattenPageViewData
                WHERE ErrorType = 'UW Block'
                  AND QuoteNumber IS NOT NULL
```

```
                AND (
                  (@timeframe = 'hours' AND SaveDateTime >= DATEADD(
HOUR, -24, GETDATE()))
                  OR (@timeframe = 'prev_week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @prev_week_start AND @prev_week_end)
                  OR (@timeframe = 'week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @week_start AND @week_end)
                  OR (@timeframe = 'month' AND CONVERT(DATE,
SaveDateTime) BETWEEN @month_start AND @month_end)
                )
            ) sq5
            JOIN AgencyMapping am
              ON sq5.UserId = am.AgentName
            GROUP BY am.AgencyName
          ),
          -- Non-UW Error Counts
          nonUWErrorCounts AS (
            SELECT
              am.AgencyName,
              COUNT(DISTINCT sq5.QuoteNumber) AS
NonUWErrorCount
            FROM (
              SELECT DISTINCT UserId, QuoteNumber
              FROM FlattenPageViewData
              WHERE ErrorType <> 'UW Block'
                AND ErrorType IS NOT NULL
                AND QuoteNumber IS NOT NULL
                AND (
                  (@timeframe = 'hours' AND SaveDateTime >= DATEADD(
HOUR, -24, GETDATE()))
                  OR (@timeframe = 'prev_week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @prev_week_start AND @prev_week_end)
                  OR (@timeframe = 'week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @week_start AND @week_end)
                  OR (@timeframe = 'month' AND CONVERT(DATE,
SaveDateTime) BETWEEN @month_start AND @month_end)
                )
            ) sq5
            JOIN AgencyMapping am
              ON sq5.UserId = am.AgentName
            GROUP BY am.AgencyName
          ),
          -- Successful Quote Counts
          successfulQuoteCounts AS (
            SELECT
              am.AgencyName,
              COUNT(DISTINCT f.QuoteNumber) AS
```

```
SuccessfulQuoteCount
            FROM FlattenPageViewData f
            JOIN AgencyMapping am
              ON f.UserId = am.AgentName
            WHERE f.QuoteStatus IN ('Bound')
             AND (
               (@timeframe = 'hours' AND SaveDateTime >= DATEADD(
HOUR, -24, GETDATE()))
               OR (@timeframe = 'prev_week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @prev_week_start AND @prev_week_end)
               OR (@timeframe = 'week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @week_start AND @week_end)
               OR (@timeframe = 'month' AND CONVERT(DATE,
SaveDateTime) BETWEEN @month_start AND @month_end)
             )
            GROUP BY am.AgencyName
          ),
          -- Total Submissions Count
          SubmissionsCount AS (
            SELECT
              am.AgencyName,
              COUNT(DISTINCT fvd.QuoteNumber) AS TotalSubmissions
            FROM FlattenPageViewData fvd
            JOIN AgencyMapping am
              ON fvd.UserId = am.AgentName
            WHERE fvd.QuoteNumber IS NOT NULL
             AND (
               (@timeframe = 'hours' AND SaveDateTime >= DATEADD(
HOUR, -24, GETDATE()))
               OR (@timeframe = 'prev_week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @prev_week_start AND @prev_week_end)
               OR (@timeframe = 'week' AND CONVERT(DATE,
SaveDateTime) BETWEEN @week_start AND @week_end)
               OR (@timeframe = 'month' AND CONVERT(DATE,
SaveDateTime) BETWEEN @month_start AND @month_end)
             )
            GROUP BY am.AgencyName
          )
          SELECT
            COALESCE(uw.AgencyName, nwe.AgencyName, sqc.
AgencyName, sc.AgencyName, dq.AgencyName) AS AgencyName,
            COALESCE(UWBlockCount, 0) AS UWBlockCount,
            COALESCE(NonUWErrorCount, 0) AS NonUWErrorCount,
            COALESCE(SuccessfulQuoteCount, 0) AS SuccessfulQuoteCount
,
            COALESCE(DeclinedQuoteCount, 0) AS DeclinedQuoteCount,
            COALESCE(TotalSubmissions, 0) AS SubmissionsCount,
```

```
            CASE
              WHEN COALESCE(TotalSubmissions, 0) > 0 THEN
                ROUND(COALESCE(UWBlockCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0), 2)
              ELSE 0
            END AS UWBlockPercentage,
            CASE
              WHEN COALESCE(TotalSubmissions, 0) > 0 THEN
                ROUND(COALESCE(NonUWErrorCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0), 2)
              ELSE 0
            END AS NonUWErrorPercentage,
            CASE
              WHEN COALESCE(TotalSubmissions, 0) > 0 THEN
                ROUND(COALESCE(SuccessfulQuoteCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0), 2)
              ELSE 0
            END AS SuccessfulQuotePercentage,
            CASE
              WHEN COALESCE(TotalSubmissions, 0) > 0 THEN
                ROUND(COALESCE(DeclinedQuoteCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0), 2)
              ELSE 0
            END AS DeclinedQuotePercentage,
            CASE
              WHEN COALESCE(SuccessfulQuoteCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0) <= 50 THEN 'Red'
              WHEN COALESCE(SuccessfulQuoteCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0) <= 75 THEN 'Orange'
              WHEN COALESCE(SuccessfulQuoteCount, 0) * 100.0 /
COALESCE(TotalSubmissions, 0) > 75 THEN 'Green'
              ELSE 'Unknown'
            END AS PerformanceStatus
          FROM uwBlockCounts uw
          FULL OUTER JOIN nonUWErrorCounts nwe ON uw.
AgencyName = nwe.AgencyName
          FULL OUTER JOIN successfulQuoteCounts sqc ON COALESCE(
uw.AgencyName, nwe.AgencyName) = sqc.AgencyName
          FULL OUTER JOIN SubmissionsCount sc ON COALESCE(uw.
AgencyName, nwe.AgencyName, sqc.AgencyName) = sc.AgencyName
          FULL OUTER JOIN DeclinedQuoteCounts dq ON COALESCE(uw
.AgencyName, nwe.AgencyName, sqc.AgencyName, sc.AgencyName) = dq.
AgencyName
          ORDER BY AgencyName;
    """
    cursor.execute(query, (timeframe, ))
    rows = cursor.fetchall()
```

```
        from datetime import datetime, timedelta

        # Get current date
        current_date = datetime.now()

        # Calculate date ranges based on timeframe
        if timeframe == 'hours':
            start_date = (current_date - timedelta(hours=24)).strftime('%Y-%m-%d
 %H:%M:%S')
            end_date = current_date.strftime('%Y-%m-%d %H:%M:%S')
        elif timeframe == 'week':
            start_date = (current_date - timedelta(days=7)).strftime('%Y-%m-%d')
            end_date = current_date.strftime('%Y-%m-%d')
        elif timeframe == 'prev_week':
            start_date = (current_date - timedelta(days=14)).strftime('%Y-%m-%d')
            end_date = (current_date - timedelta(days=7)).strftime('%Y-%m-%d')
        elif timeframe == 'month':
            start_date = (current_date - timedelta(days=30)).strftime('%Y-%m-%d')
            end_date = current_date.strftime('%Y-%m-%d')
        else:
            start_date = "N/A"
            end_date = "N/A"
        # Get column names
        columns = [column[0] for column in cursor.description]

        percentage_columns = ['UWBlockPercentage', 'NonUWErrorPercentage',
'SuccessfulQuotePercentage']
        # Clean the rows to remove newline characters
        cleaned_rows = [
            tuple(str(value).replace("\n", " ").strip() if isinstance(value, str) else value
for value in row)
            for row in rows
        ]

        # Create a DataFrame
        df = pd.DataFrame.from_records(cleaned_rows, columns=columns)
        df = df[df["AgencyName"] != "Agency not mapped"]

        # Replace \n in the DataFrame for clean display
        df.replace(r'\n', ' ', regex=True, inplace=True)
        percentage_columns = ['UWBlockPercentage', 'NonUWErrorPercentage',
'SuccessfulQuotePercentage', 'DeclinedQuotePercentage']

        if not df.empty:
            for col in percentage_columns:
                df[col] = df[col].apply(lambda x: f"{x:.2f}%")
```

```
        # Function to color the text based on PerformanceStatus
        def colorize_text(df, text_column):
            """
            Applies color to text based on color names in a dataframe column.
            """
            def apply_color(row):
                color = row[text_column]
                return [f"color: {color}" if pd.notna(color) else "" for _ in row]

                # Apply styling
            styled_df = df.style.apply(apply_color, axis=1)

            # Hide the column by setting display properties
            styled_df = styled_df.hide(axis="columns", subset=[text_column])

            return styled_df
        styled_df = colorize_text(df, 'PerformanceStatus')

        connection.close()
        print("Connection closed.")
        return render_template('AgencyAnalysisTable.html', tables=[styled_df.
to_html(classes='data', header="False")],
                               timeframe=timeframe, start_date=start_date, end_date=
end_date)

    except pyodbc.Error as e:
        print(colored(f"Error while connecting to SQL Server: {e}", "red"))

    # finally:
    #     # Clean up and close the connection
    #     if 'connection' in locals() and connection:
    #         connection.close()
    #         print(colored("Connection closed.", "blue"))
if __name__ == '__main__':
    app.run(debug=True)
```