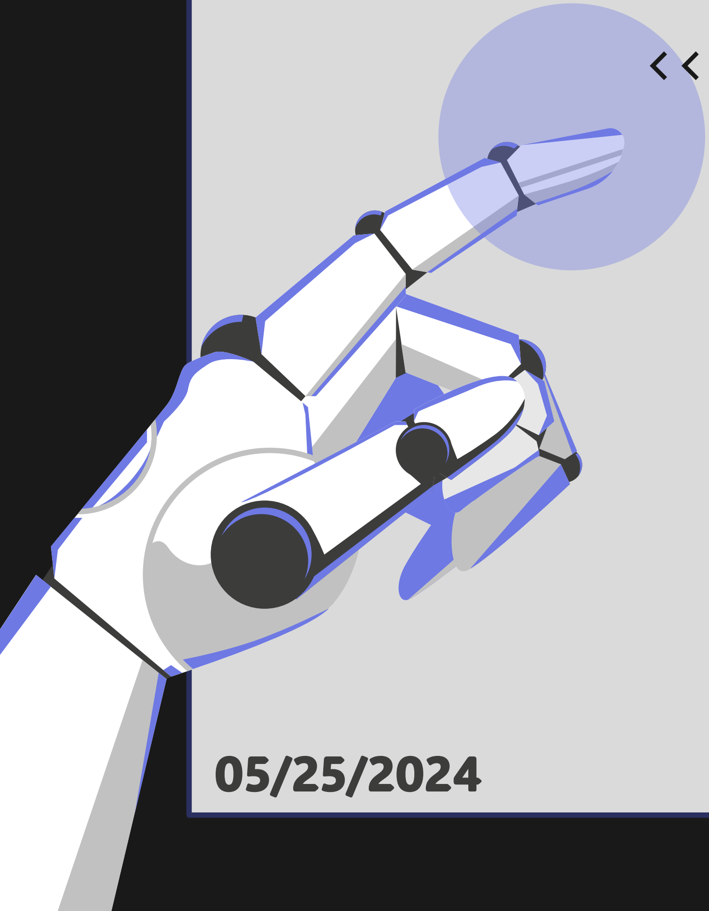


MONITORING ANALYST TEST

Case study

05/25/2024

ABOUT ME

GENERAL INFORMATION

01. CASSIANO JOSÉ SOBIERAI

02. CURRENTLY LIVING IN FARROUPILHA, BRAZIL

03. CURRENTLY WORKING AT JOST BRAZIL AS A JR DATA INTELLIGENCE ANALYST

04. GRADUATING SYSTEMS ANALYSIS AND DEVELOPMENT AT IFRS

TABLE OF CONTENTS

01.

TEST 1:
THE PROBLEM

An introduction to the first part of the test

02.

TEST 1: THE PROCESS OF ANALYSIS

Getting insights of the avaliable data

03.

TEST 2: THE MONITORING SYSTEM

An introduction to the second part of the test

04.

TEST 2: CREATING THE SYSTEM

The development process of the system

05.

TEST 2: VISUALIZING THE SYSTEM

Showing the results of the system

06.

CONCLUSION

Lessons learned

TEST 1: THE PROBLEM

Identify any anomalous behavior in a sales dataset.

The test provided two similar datasets, about transactions per hour. Both have 24 lines from 00h to 23h.

THE DATASETS

Column	Type of data	Ex:
Time	String	01h
today	Integer	9
Yesterday	Integer	11
same_day_last_week	Integer	11
avg_last_week	Float	6.42
avg_last_month	Float	4.85

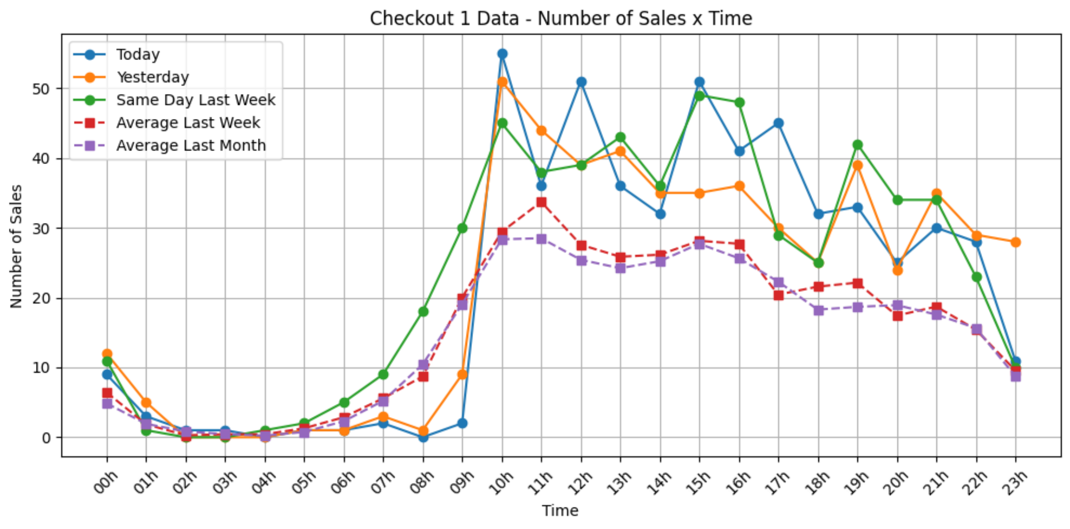
TEST 1: THE PROCESS OF ANALYSIS

CHECKOUT 1 DATA

All daily data shows spikes in sales, if compared to the averages.

Yesterday data does not finish the day like all the other data.

Averages have similar trends.



OVERVIEW OF THE DATA

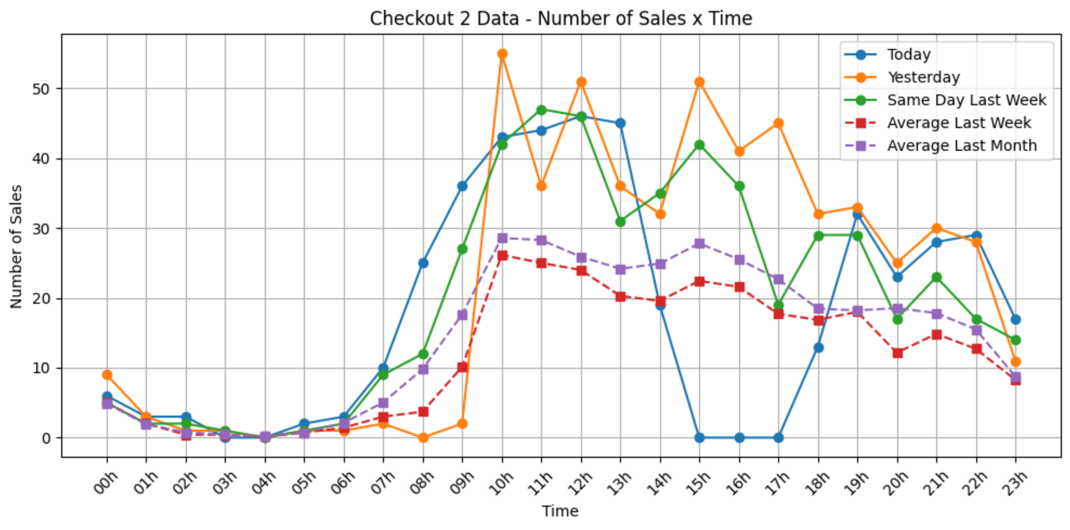
TEST 1: THE PROCESS OF ANALYSIS

CHECKOUT 2 DATA

Yesterday and Same_day_last_week data shows spikes in sales, if compared to the averages.

Today data have zero sales between 15h and 17h

Averages have similar trends.



OVERVIEW OF THE DATA

> > > >

ABOUT THE REQUIRED SQL QUERY

The limited dataset precludes robusts queries, but here is an example where SQL could be applicable.

IDENTIFY POTENTIAL ANOMALIES

```sql
SELECT
    time,
    today,
    AVG(avg_last_week) AS avg_sales,
       STDDEV(avg_last_week) AS sales_std_dev,
       CASE
           WHEN today > AVG(avg_last_week) + 2 * STDDEV(avg_last_week) THEN 'High Anomaly'
           WHEN today < AVG(avg_last_week) - 2 * STDDEV(avg_last_week) THEN 'Low Anomaly'
           ELSE 'Normal'
       END AS anomaly_status
FROM ['Table Name']
GROUP BY time, today;
```

TEST 1: THE PROCESS OF ANALYSIS

PROBLEM

The dataset given is too small to perform a robust analysis. Limited data points can lead to unreliable conclusions and missed anomalies.

SOLUTION

Communicate the issue to the team and stakeholders. Request more extensive data to conduct a thorough and accurate analysis.

TEST 2: THE MONITORING SYSTEM

ENDPOINT

Receive transactions at any time

SEND ALERTS

Notify relevant teams when transactions are above normal

LIVE DASHBOARD

Provide a real-time view of some transaction statuses.

DETECT ANOMALIES

Identify unusual transaction points using predefined criteria

TEST 2: CREATING THE SYSTEM

ACTION PLAN TO DEVELOPMENT

01. FLASK ENVIRONMENT

02. DASHBOARD FRONT-END WITH PLOTLY

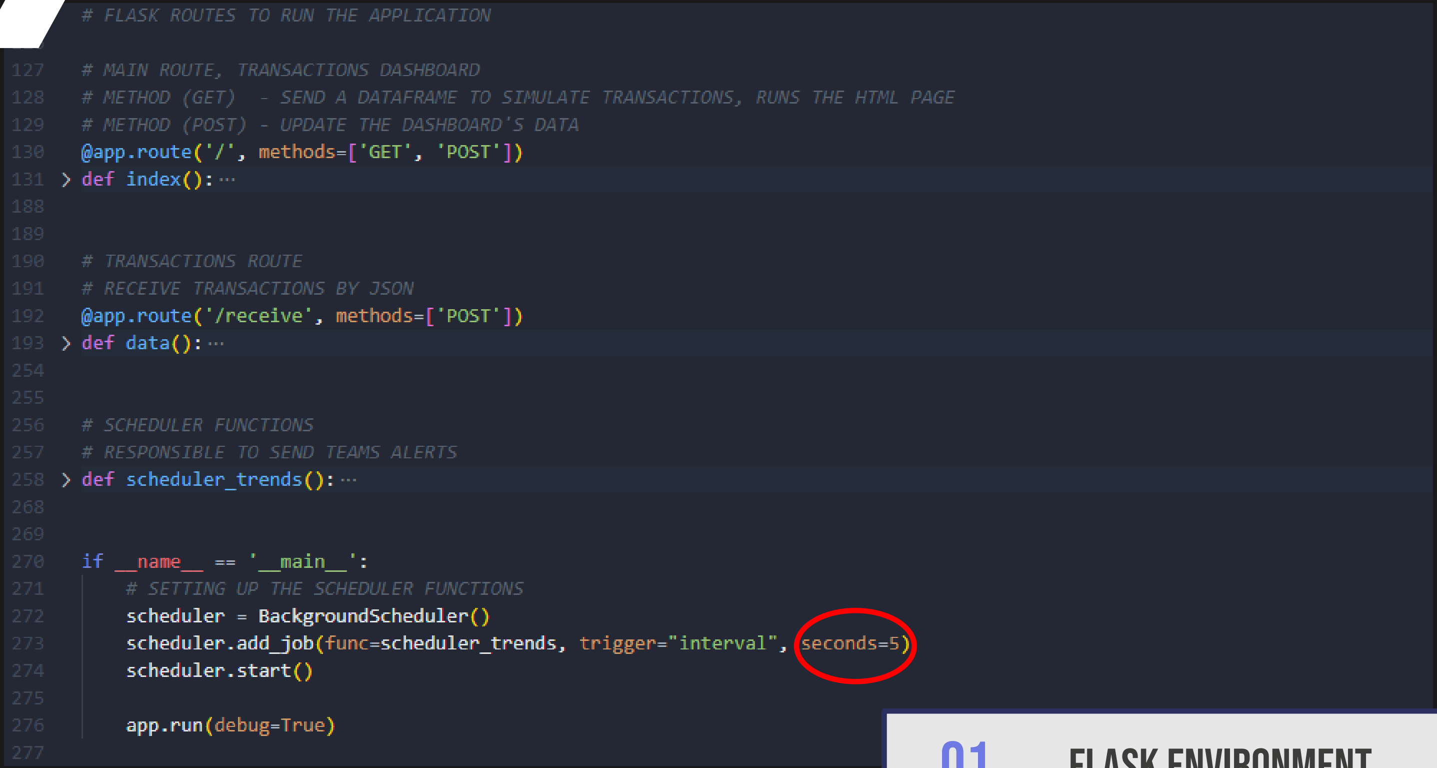
03. TRANSACTIONS ENDPOINT

04. TRENDS ALERTS IN TEAMS

05. DETECT ANOMALIES

> > > >

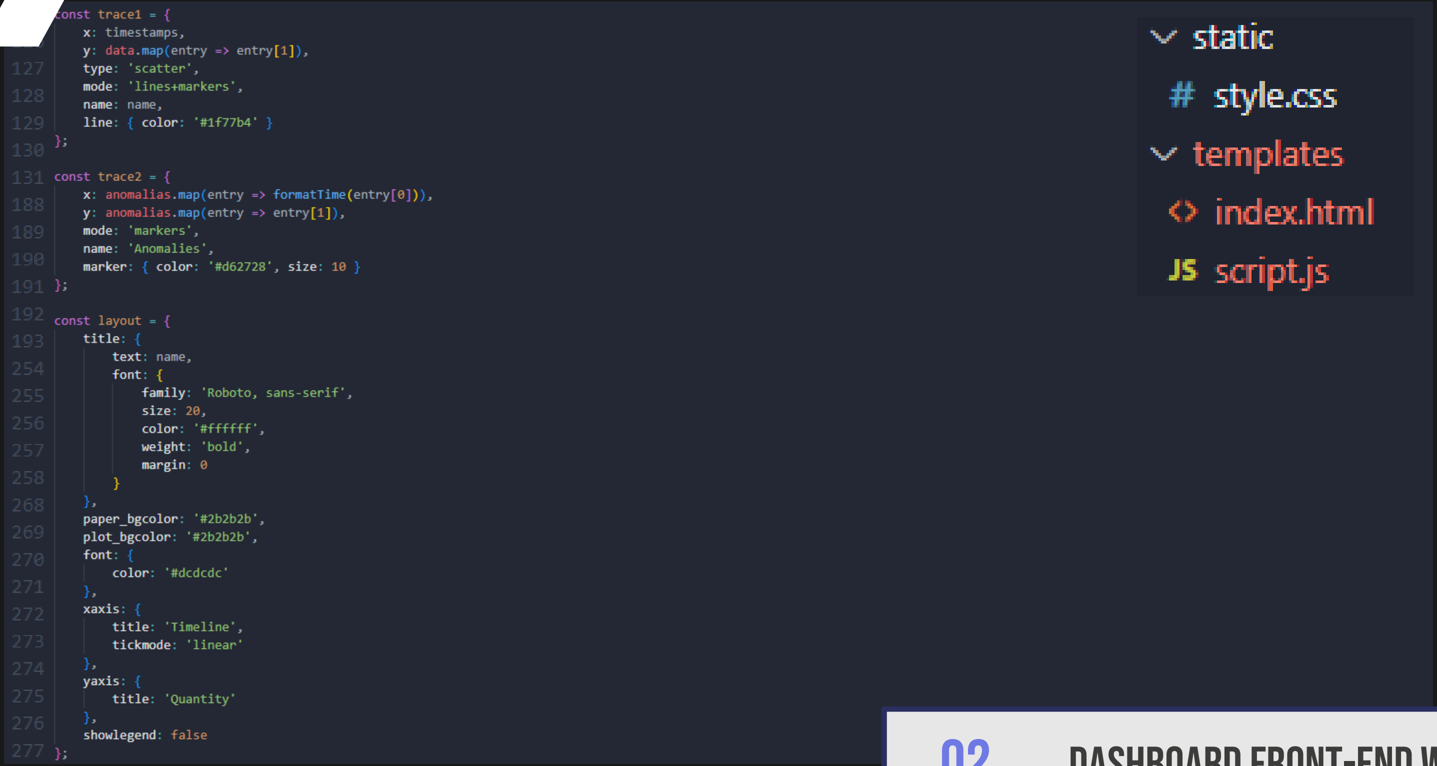
TEST 2: CREATING THE SYSTEM



```python
# FLASK ROUTES TO RUN THE APPLICATION

# MAIN ROUTE, TRANSACTIONS DASHBOARD
# METHOD (GET)  - SEND A DATAFRAME TO SIMULATE TRANSACTIONS, RUNS THE HTML PAGE
# METHOD (POST) - UPDATE THE DASHBOARD'S DATA
@app.route('/', methods=['GET', 'POST'])
def index():⋯


# TRANSACTIONS ROUTE
# RECEIVE TRANSACTIONS BY JSON
@app.route('/receive', methods=['POST'])
def data():⋯


# SCHEDULER FUNCTIONS
# RESPONSIBLE TO SEND TEAMS ALERTS
def scheduler_trends():⋯


if __name__ == '__main__':
    # SETTING UP THE SCHEDULER FUNCTIONS
    scheduler = BackgroundScheduler()
    scheduler.add_job(func=scheduler_trends, trigger="interval", seconds=5)
    scheduler.start()

    app.run(debug=True)

```

01. FLASK ENVIRONMENT

TEST 2: CREATING THE SYSTEM

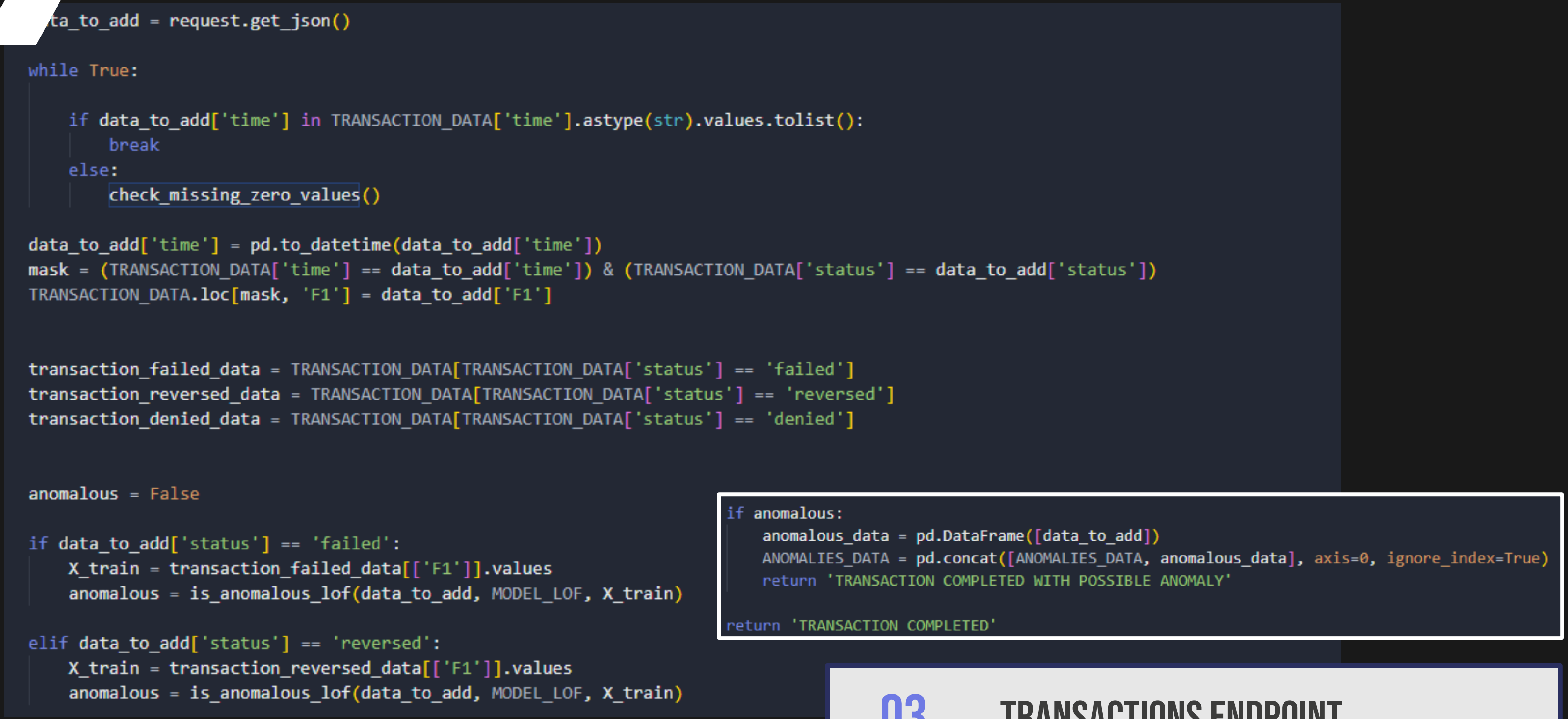


```javascript
const trace1 = {
    x: timestamps,
    y: data.map(entry => entry[1]),
    type: 'scatter',
    mode: 'lines+markers',
    name: name,
    line: { color: '#1f77b4' }
};

const trace2 = {
    x: anomalias.map(entry => formatTime(entry[0])),
    y: anomalias.map(entry => entry[1]),
    mode: 'markers',
    name: 'Anomalies',
    marker: { color: '#d62728', size: 10 }
};

const layout = {
    title: {
        text: name,
        font: {
            family: 'Roboto, sans-serif',
            size: 20,
            color: '#ffffff',
            weight: 'bold',
            margin: 0
        }
    },
    paper_bgcolor: '#2b2b2b',
    plot_bgcolor: '#2b2b2b',
    font: {
        color: '#dcdcdc'
    },
    xaxis: {
        title: 'Timeline',
        tickmode: 'linear'
    },
    yaxis: {
        title: 'Quantity'
    },
    showlegend: false
};
```

```
∨ static
    # style.css
∨ templates
    <> index.html
    JS script.js
```

02. DASHBOARD FRONT-END WITH PLOTLY

TEST 2: CREATING THE SYSTEM

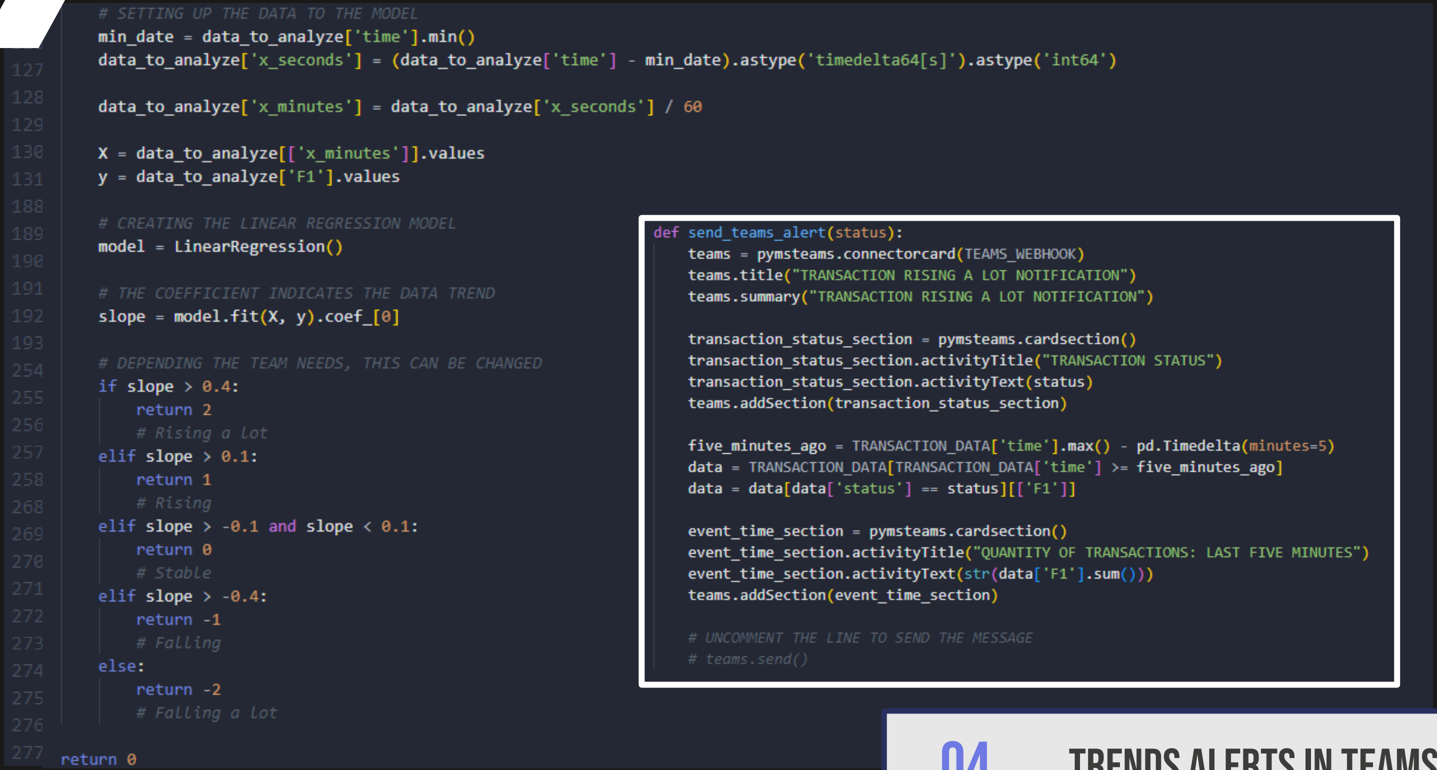


```
ta_to_add = request.get_json()

while True:

    if data_to_add['time'] in TRANSACTION_DATA['time'].astype(str).values.tolist():
        break
    else:
        check_missing_zero_values()

data_to_add['time'] = pd.to_datetime(data_to_add['time'])
mask = (TRANSACTION_DATA['time'] == data_to_add['time']) & (TRANSACTION_DATA['status'] == data_to_add['status'])
TRANSACTION_DATA.loc[mask, 'F1'] = data_to_add['F1']


transaction_failed_data = TRANSACTION_DATA[TRANSACTION_DATA['status'] == 'failed']
transaction_reversed_data = TRANSACTION_DATA[TRANSACTION_DATA['status'] == 'reversed']
transaction_denied_data = TRANSACTION_DATA[TRANSACTION_DATA['status'] == 'denied']


anomalous = False

if data_to_add['status'] == 'failed':
    X_train = transaction_failed_data[['F1']].values
    anomalous = is_anomalous_lof(data_to_add, MODEL_LOF, X_train)

elif data_to_add['status'] == 'reversed':
    X_train = transaction_reversed_data[['F1']].values
    anomalous = is_anomalous_lof(data_to_add, MODEL_LOF, X_train)
```

```
if anomalous:
    anomalous_data = pd.DataFrame([data_to_add])
    ANOMALIES_DATA = pd.concat([ANOMALIES_DATA, anomalous_data], axis=0, ignore_index=True)
    return 'TRANSACTION COMPLETED WITH POSSIBLE ANOMALY'

return 'TRANSACTION COMPLETED'
```

03. TRANSACTIONS ENDPOINT

TEST 2: CREATING THE SYSTEM

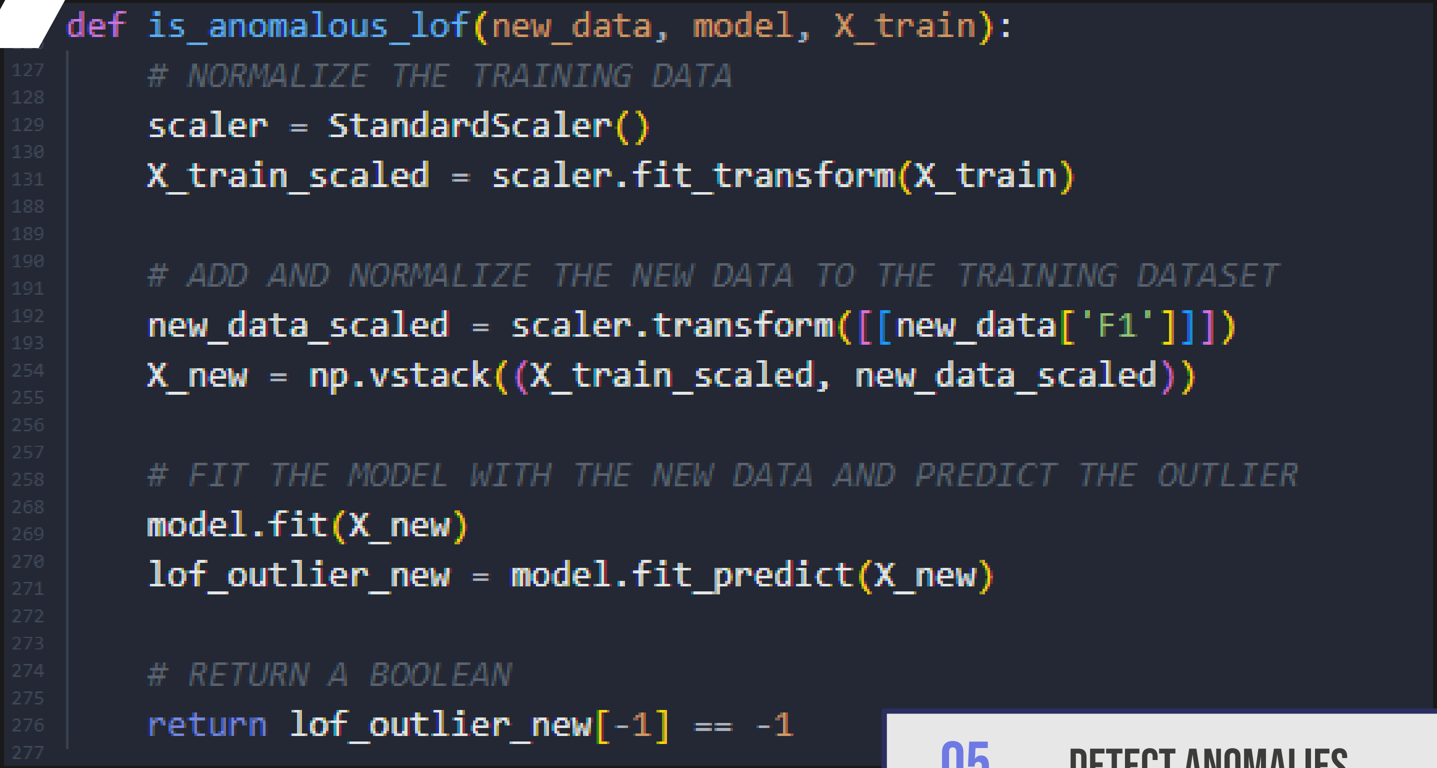


```python
    # SETTING UP THE DATA TO THE MODEL
    min_date = data_to_analyze['time'].min()
    data_to_analyze['x_seconds'] = (data_to_analyze['time'] - min_date).astype('timedelta64[s]').astype('int64')

    data_to_analyze['x_minutes'] = data_to_analyze['x_seconds'] / 60

    X = data_to_analyze[['x_minutes']].values
    y = data_to_analyze['F1'].values

    # CREATING THE LINEAR REGRESSION MODEL
    model = LinearRegression()

    # THE COEFFICIENT INDICATES THE DATA TREND
    slope = model.fit(X, y).coef_[0]

    # DEPENDING THE TEAM NEEDS, THIS CAN BE CHANGED
    if slope > 0.4:
        return 2
        # Rising a lot
    elif slope > 0.1:
        return 1
        # Rising
    elif slope > -0.1 and slope < 0.1:
        return 0
        # Stable
    elif slope > -0.4:
        return -1
        # Falling
    else:
        return -2
        # Falling a lot

return 0
```

```python
def send_teams_alert(status):
    teams = pymsteams.connectorcard(TEAMS_WEBHOOK)
    teams.title("TRANSACTION RISING A LOT NOTIFICATION")
    teams.summary("TRANSACTION RISING A LOT NOTIFICATION")

    transaction_status_section = pymsteams.cardsection()
    transaction_status_section.activityTitle("TRANSACTION STATUS")
    transaction_status_section.activityText(status)
    teams.addSection(transaction_status_section)

    five_minutes_ago = TRANSACTION_DATA['time'].max() - pd.Timedelta(minutes=5)
    data = TRANSACTION_DATA[TRANSACTION_DATA['time'] >= five_minutes_ago]
    data = data[data['status'] == status][['F1']]

    event_time_section = pymsteams.cardsection()
    event_time_section.activityTitle("QUANTITY OF TRANSACTIONS: LAST FIVE MINUTES")
    event_time_section.activityText(str(data['F1'].sum()))
    teams.addSection(event_time_section)

    # UNCOMMENT THE LINE TO SEND THE MESSAGE
    # teams.send()
```

04. TRENDS ALERTS IN TEAMS

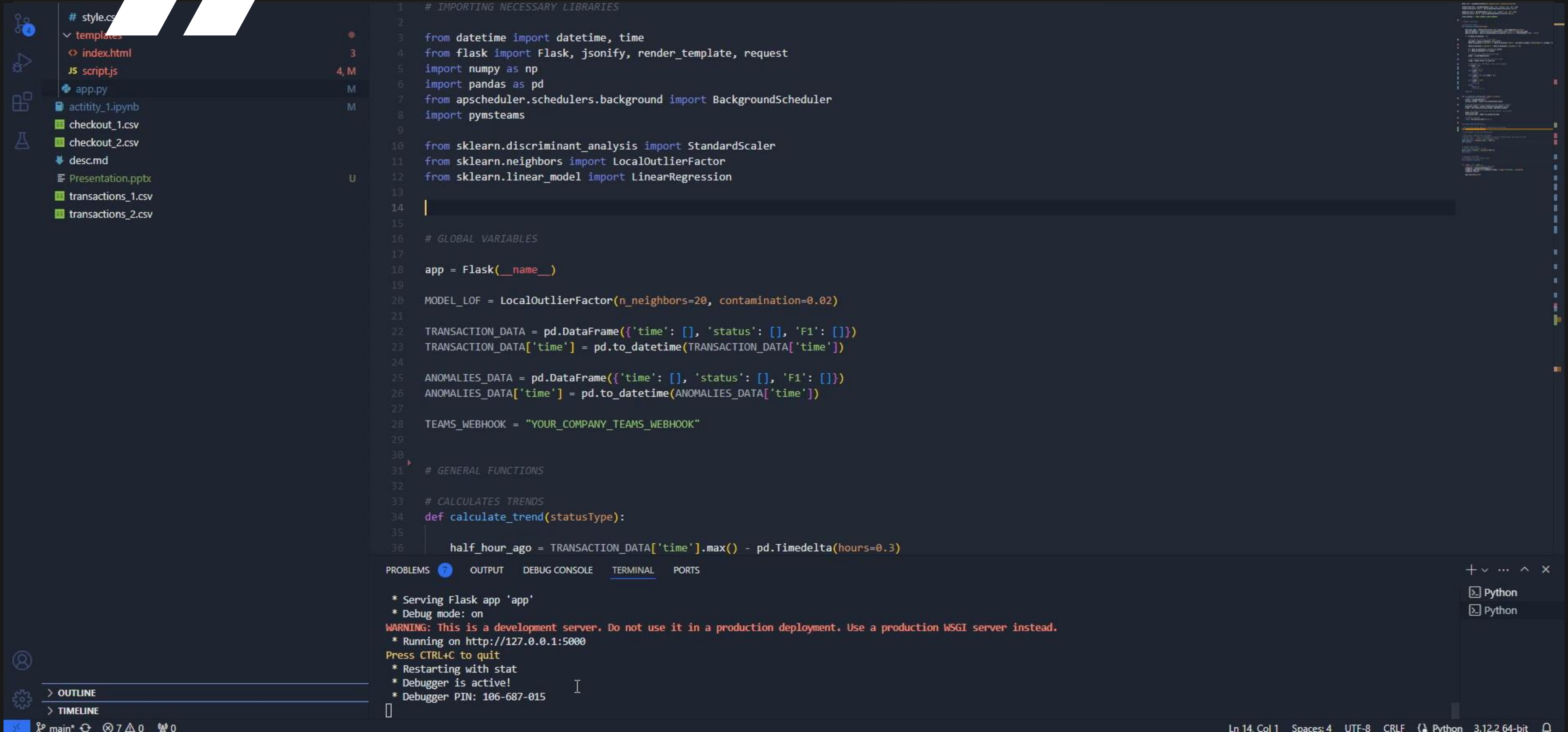
TEST 2: CREATING THE SYSTEM



```python
def is_anomalous_lof(new_data, model, X_train):
    # NORMALIZE THE TRAINING DATA
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)

    # ADD AND NORMALIZE THE NEW DATA TO THE TRAINING DATASET
    new_data_scaled = scaler.transform([[new_data['F1']]])
    X_new = np.vstack((X_train_scaled, new_data_scaled))

    # FIT THE MODEL WITH THE NEW DATA AND PREDICT THE OUTLIER
    model.fit(X_new)
    lof_outlier_new = model.fit_predict(X_new)

    # RETURN A BOOLEAN
    return lof_outlier_new[-1] == -1
```

05. DETECT ANOMALIES

TEST 2: VISUALIZING THE SYSTEM



```python
# IMPORTING NECESSARY LIBRARIES

from datetime import datetime, time
from flask import Flask, jsonify, render_template, request
import numpy as np
import pandas as pd
from apscheduler.schedulers.background import BackgroundScheduler
import pymsteams

from sklearn.discriminant_analysis import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
from sklearn.linear_model import LinearRegression



# GLOBAL VARIABLES

app = Flask(__name__)

MODEL_LOF = LocalOutlierFactor(n_neighbors=20, contamination=0.02)

TRANSACTION_DATA = pd.DataFrame({'time': [], 'status': [], 'F1': []})
TRANSACTION_DATA['time'] = pd.to_datetime(TRANSACTION_DATA['time'])

ANOMALIES_DATA = pd.DataFrame({'time': [], 'status': [], 'F1': []})
ANOMALIES_DATA['time'] = pd.to_datetime(ANOMALIES_DATA['time'])

TEAMS_WEBHOOK = "YOUR_COMPANY_TEAMS_WEBHOOK"


# GENERAL FUNCTIONS

# CALCULATES TRENDS
def calculate_trend(statusType):

    half_hour_ago = TRANSACTION_DATA['time'].max() - pd.Timedelta(hours=0.3)
```

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 106-687-015
```

CONCLUSION
LESSONS LEARNED

1. Need more practice with Local Outlier Factor
2. Linear Regression was useful for understanding trends
3. Communication is essencial when data is limited
4. Chat GPT improves our front-end skills
5. Need to explore more data analysis libraries.

ARTIFICIAL

INTE

(AI)

THANK YOU

ANY QUESTIONS?
e-mail to cassianosobieraii@gmail.com

(AI)