BROADCOM®
MAINFRAME SOFTWARE

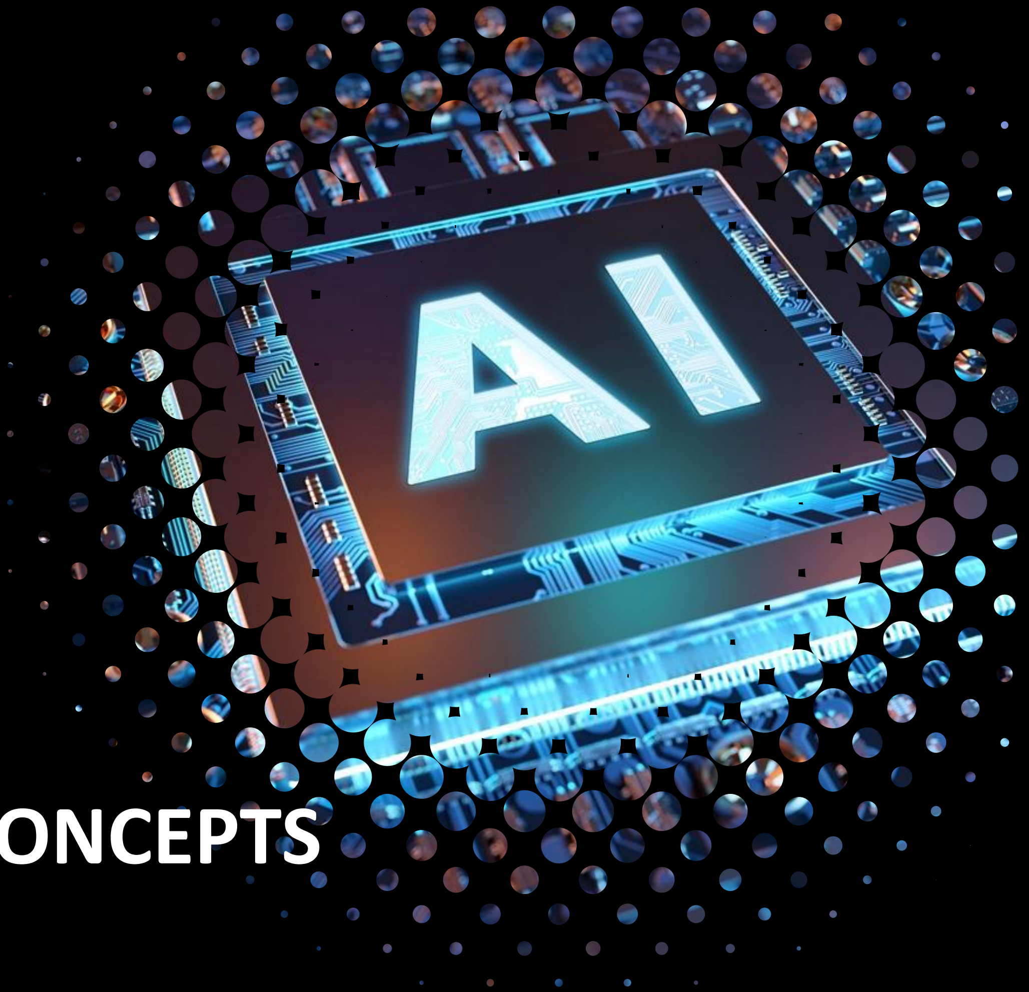
Using AI/ML to Enhance Db2 Subsystem and Application Performance

Antonio Couto

Broadcom

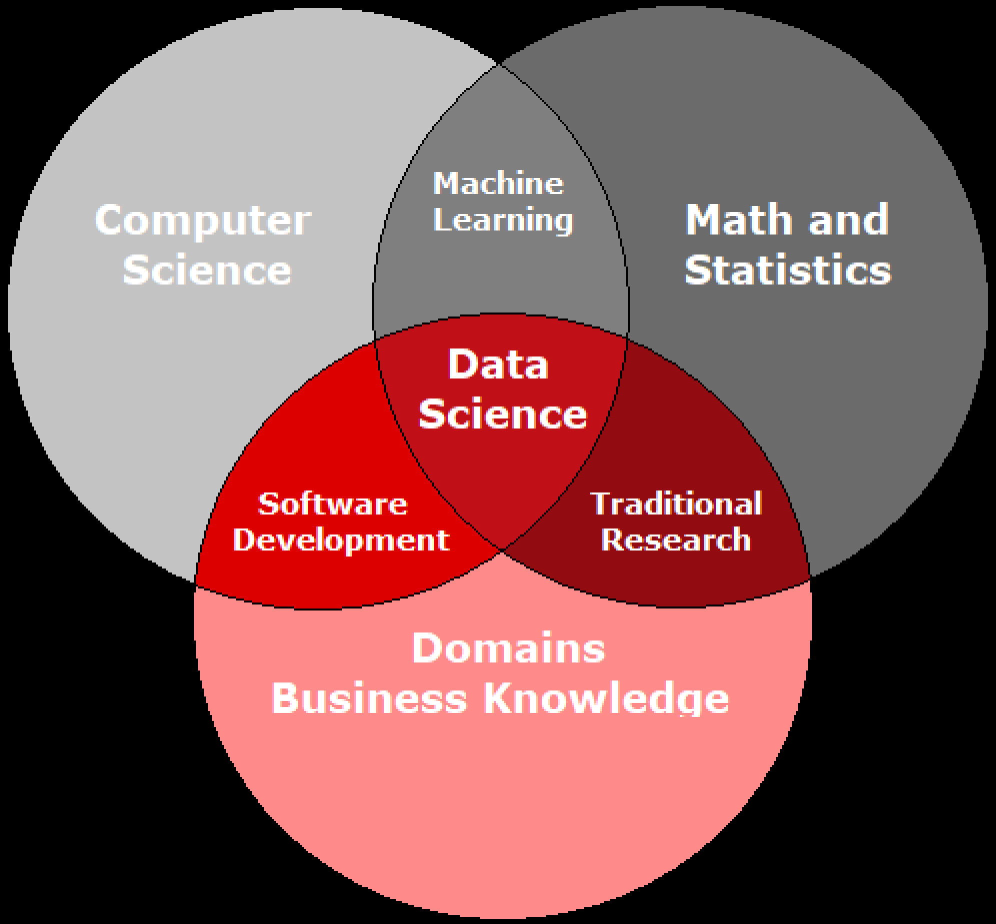
Agenda

- **Machine Learning** Concepts
- Applied Machine Learning to **enhance Db2 Performance**
- Customer **Use Cases**
  - What makes a Business Application?
  - Db2 Application Elapsed Time on CICS
  - High CPU usage on Db2 DIST Address Space
  - Db2 Package and Plan performance investigation



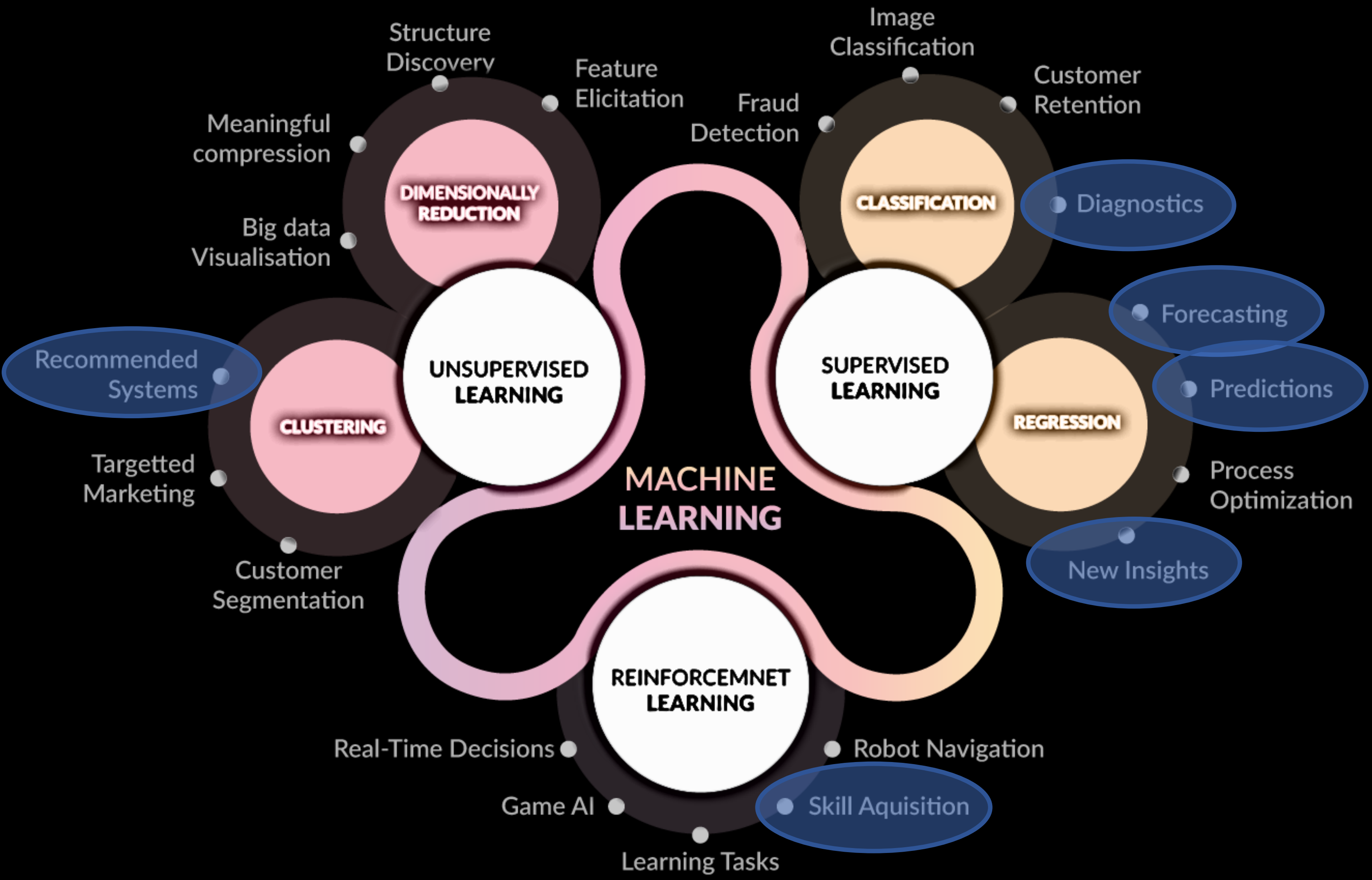
MACHINE LEARNING CONCEPTS

Machine **Learning**



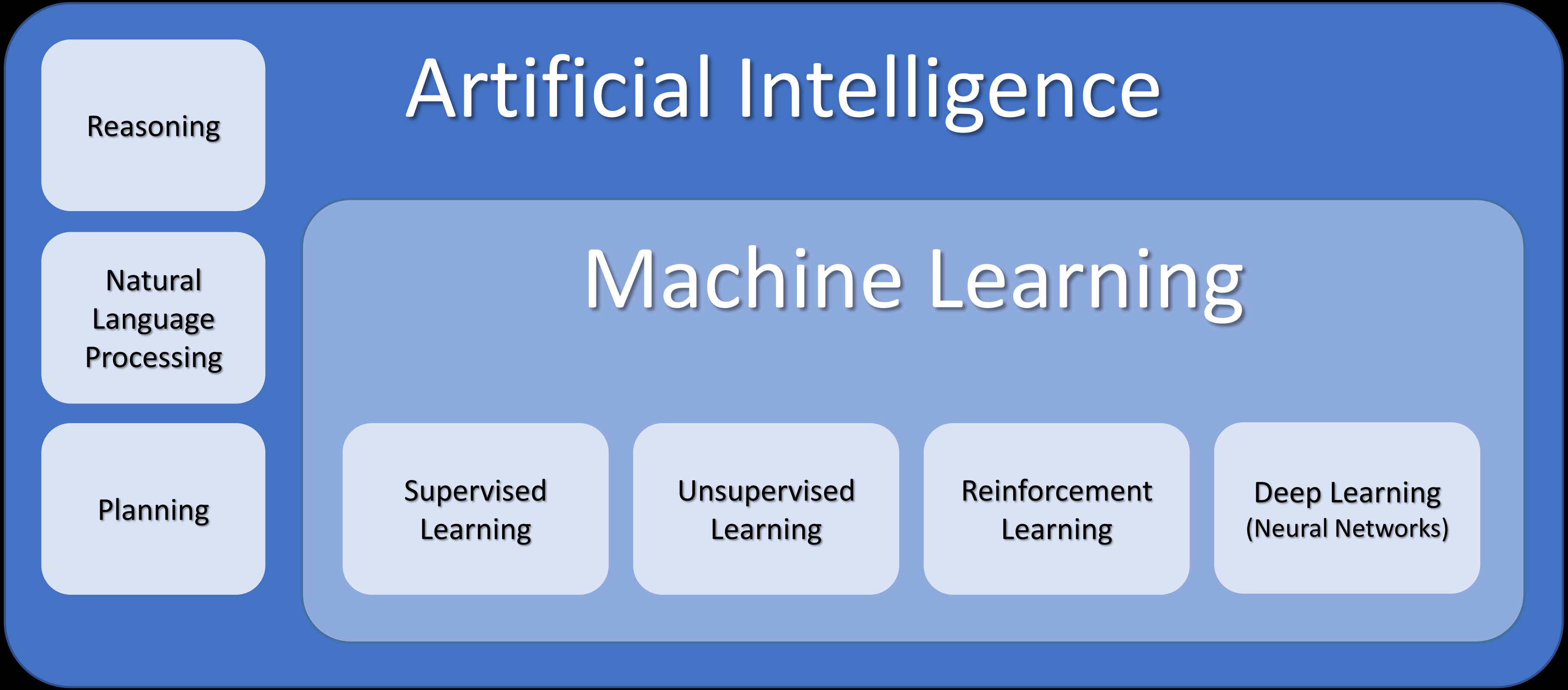
(1) https://towardsdatascience.com/introduction-to-statistics-e9d72d818745

Machine **Learning**



(1) https://towardsdatascience.com/coding-deep-learning-for-beginners-types-of-machine-learning-b9e651e1ed9d

Machine **Learning** and Artificial **Intelligence**



Artificial Intelligence

Reasoning

Natural Language Processing

Planning

Machine Learning

Supervised Learning

Unsupervised Learning

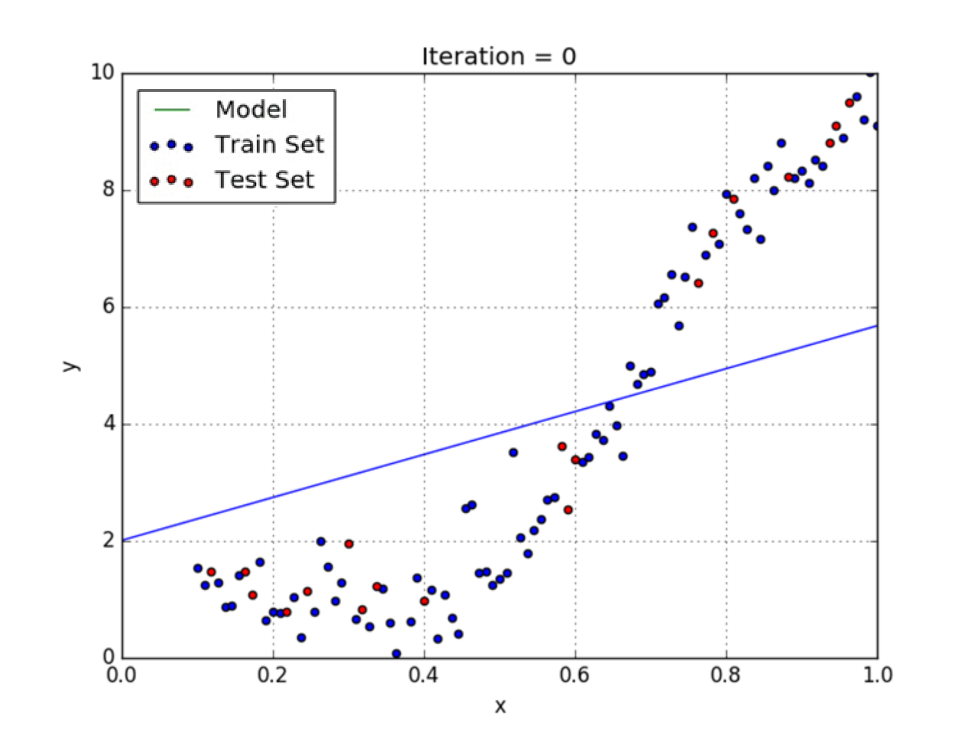
Reinforcement Learning

Deep Learning (Neural Networks)

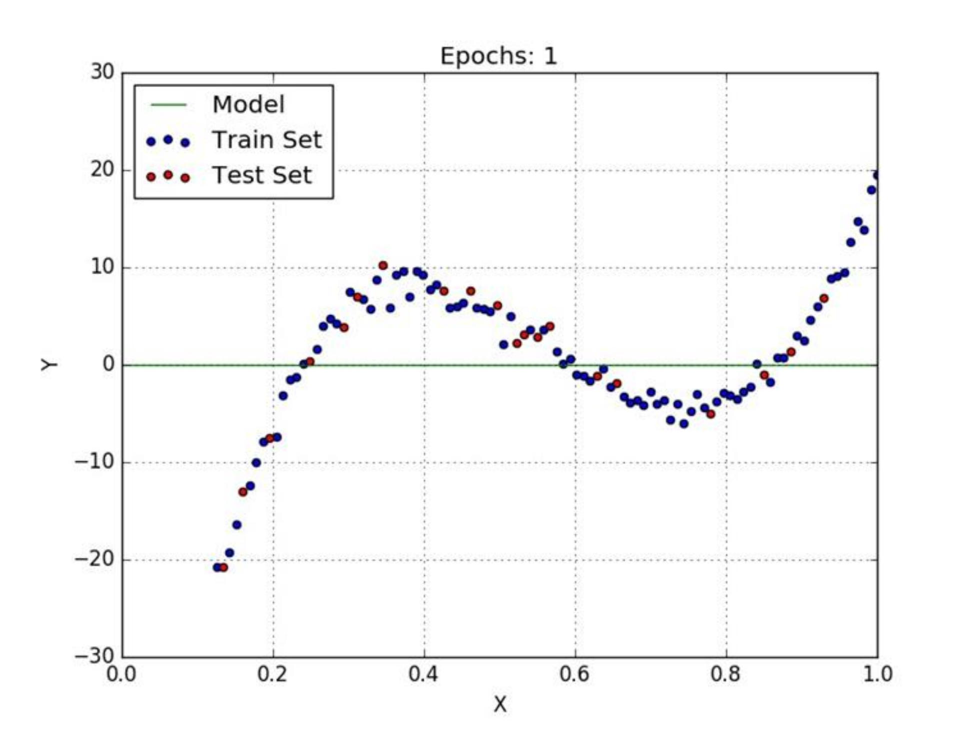
[1] https://www.ibm.com/downloads/cas/GB8ZMQZ3

Supervised Learning

Linear Regression



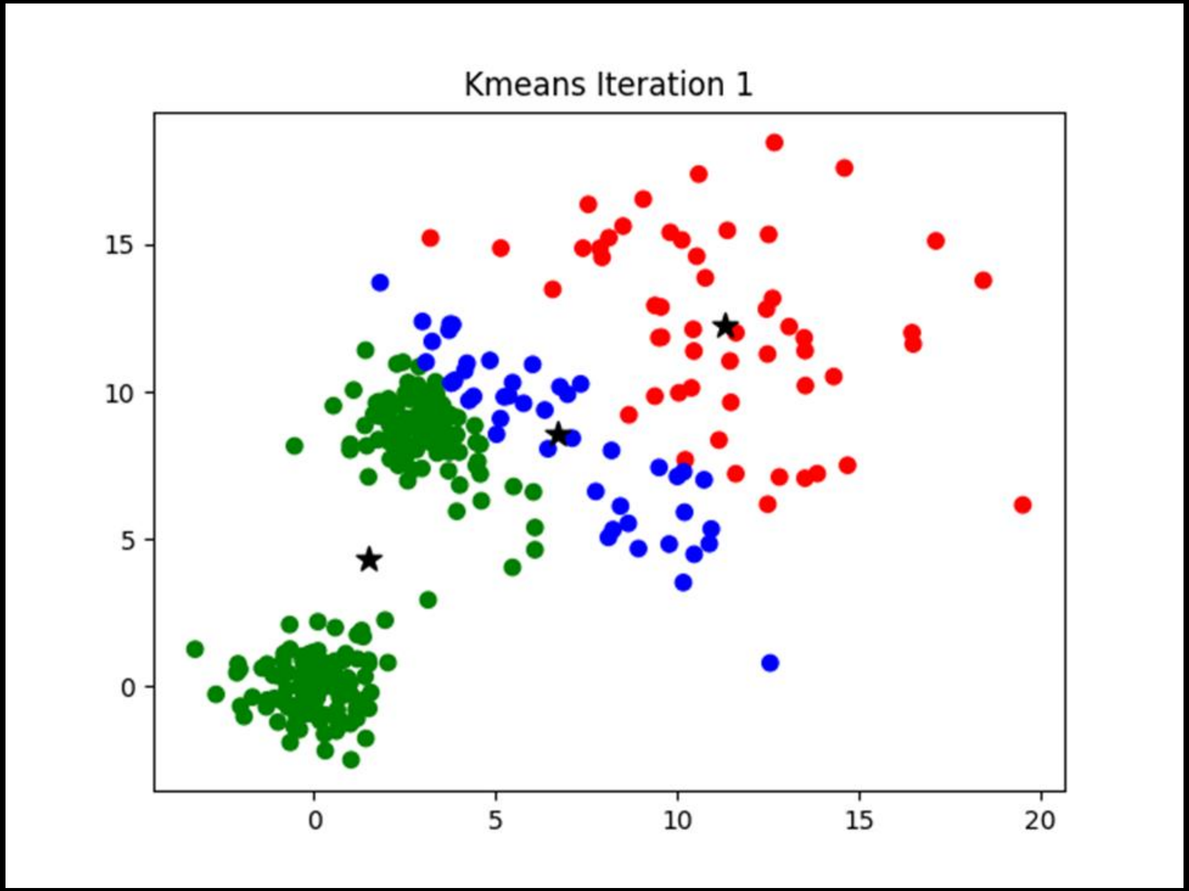
Logistic Regression



(1) https://medium.com/analytics-vidhya/simple-linear-regression-cost-function-gradient-descent-50c5ed085770
(2) https://medium.com/@vaibhavnohria36/coming-up-logistic-regression-for-you-23a3134a4d7e

Unsupervised Learning

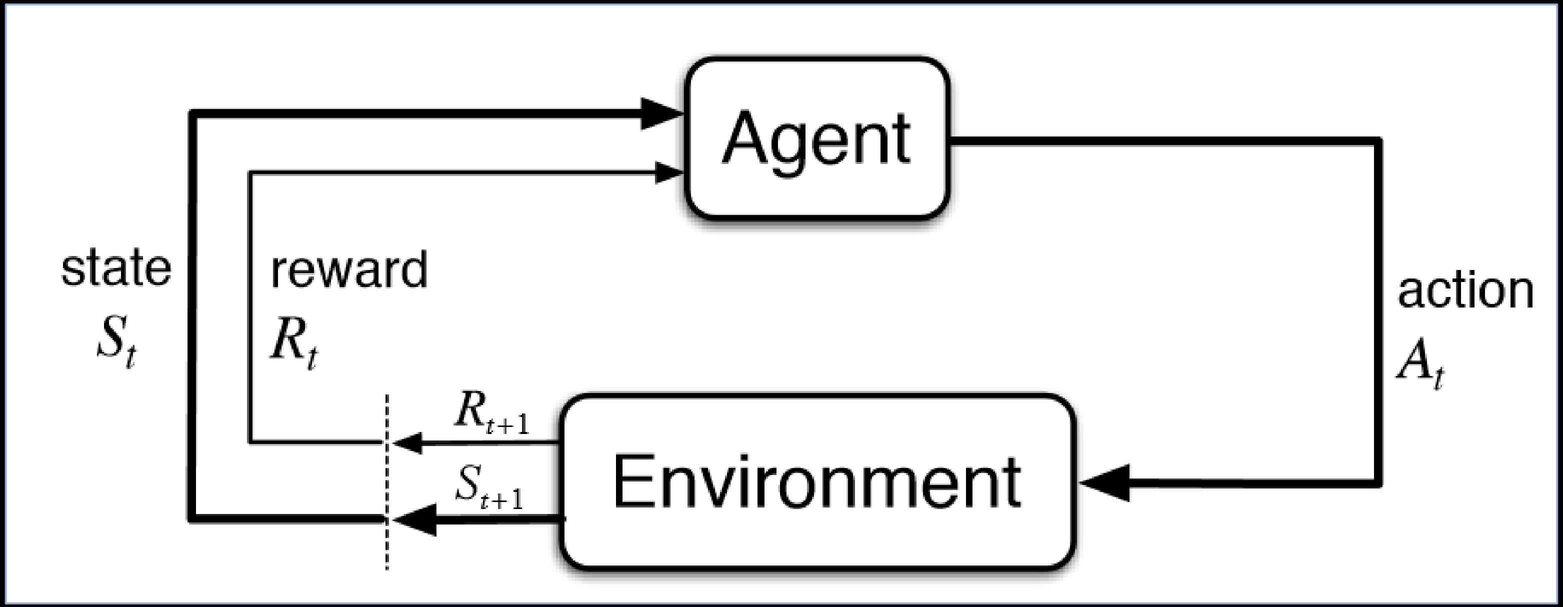
K-means Clustering



(1) https://medium.com/@imparth/k-means-clustering-algorithm-34807a7cec71

Reinforcement Learning

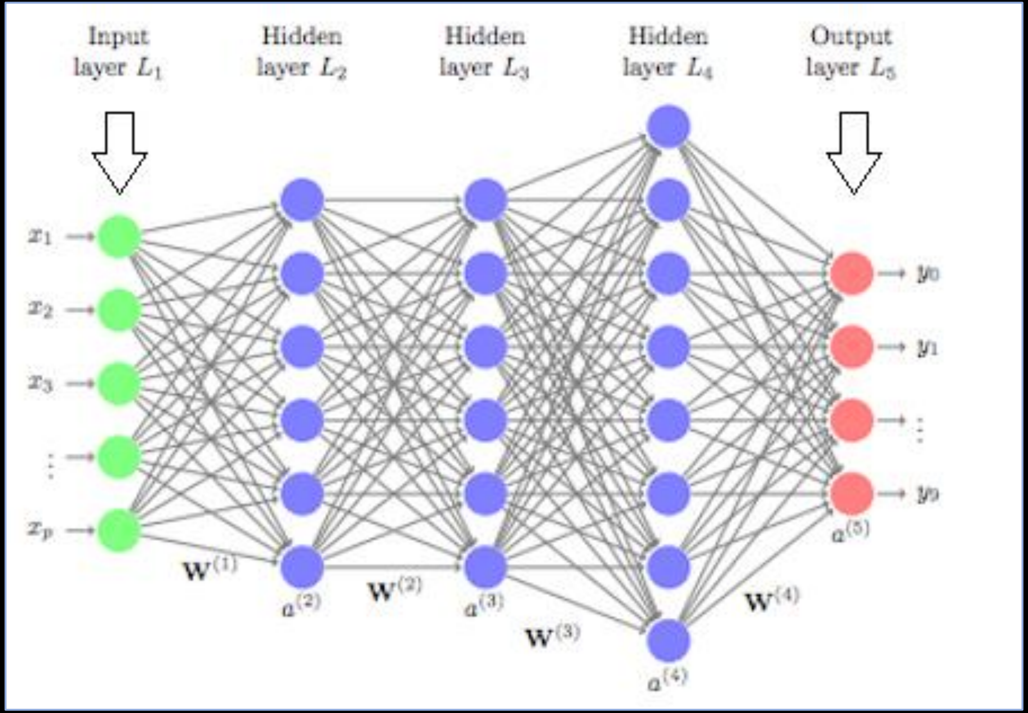
Generic Reinforcement Learning Model



(1) https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Deep Learning

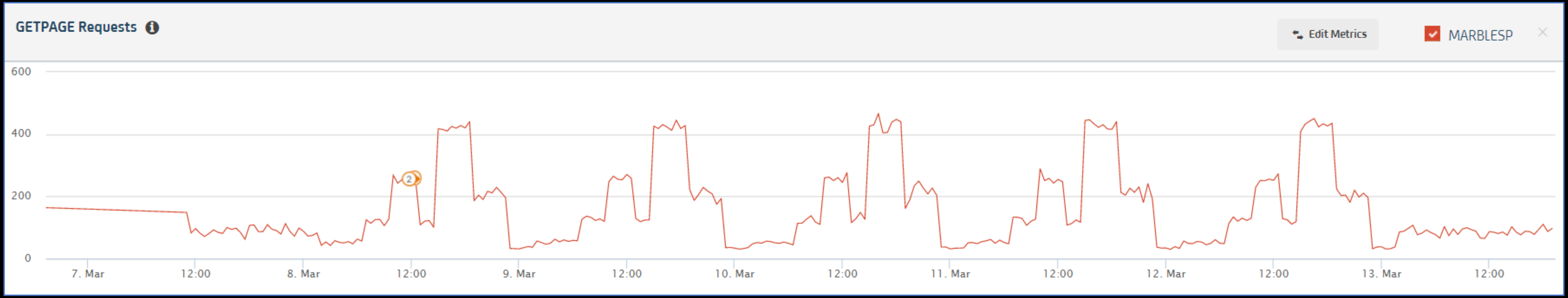
Artificial Neural Network



(1) https://vishalyadav831874.medium.com/industry-use-cases-of-neural-network-cccc4b333621

Time Series Analysis

Time Series Data

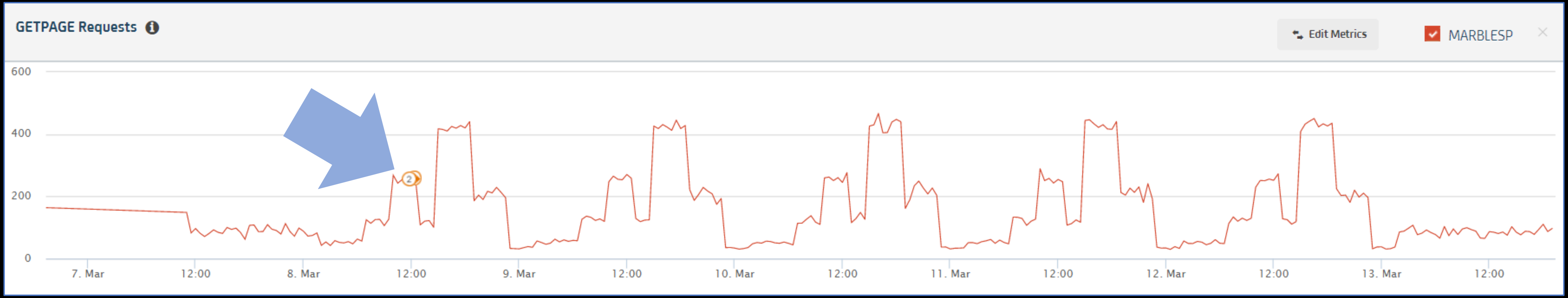


For example: Db2 performance metrics, such as QBSTGET - Total number of GETPAGE requests at every 5 seconds, for a particular application, throughout the week.

(1) https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

Anomaly Detection

Detecting an Anomaly

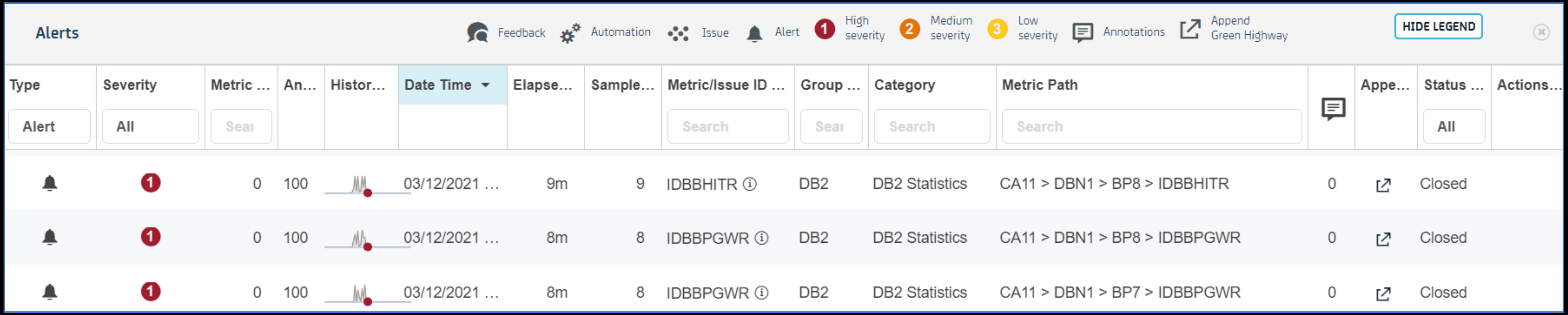


For example: Db2 performance metrics, such as QBSTGET - Total number of GETPAGE requests at every 5 seconds, for a particular application, throughout the week.

(1) https://towardsdatascience.com/real-time-time-series-anomaly-detection-981cf1e1ca13

Pattern Discovery

Discovering Patterns



Alerts

Feedback · Automation · Issue · Alert · 1 High severity · 2 Medium severity · 3 Low severity · Annotations · Append Green Highway

HIDE LEGEND

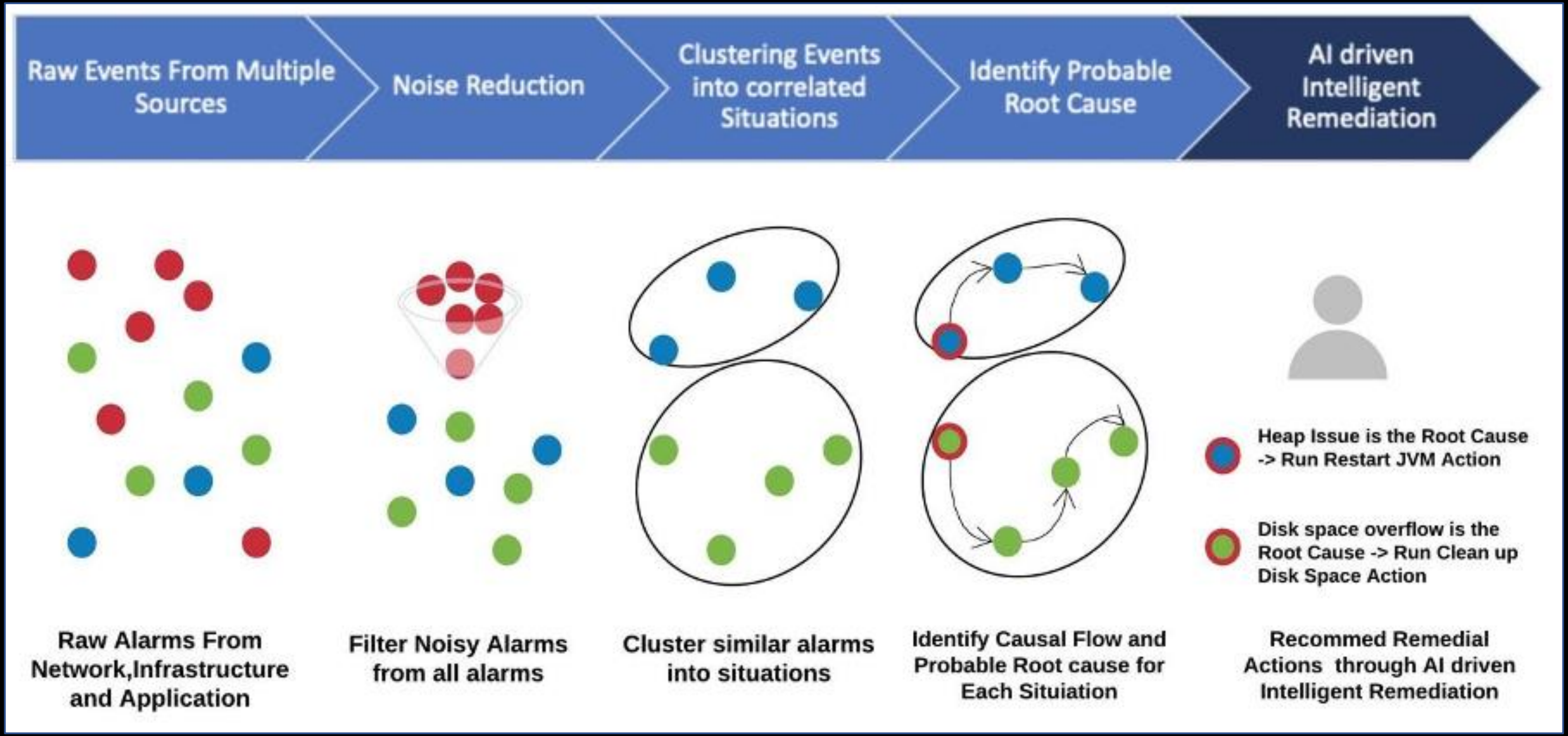
Type	Severity	Metric ...	An...	Histor...	Date Time	Elapse...	Sample...	Metric/Issue ID ...	Group ...	Category	Metric Path	☰	Appe...	Status ...	Actions...
Alert	All	Sear						Search	Sear	Search	Search			All	
🔔	1	0	100		03/12/2021 ...	9m	9	IDBBHITR ⓘ	DB2	DB2 Statistics	CA11 > DBN1 > BP8 > IDBBHITR	0	↗	Closed	
🔔	1	0	100		03/12/2021 ...	8m	8	IDBBPGWR ⓘ	DB2	DB2 Statistics	CA11 > DBN1 > BP8 > IDBBPGWR	0	↗	Closed	
🔔	1	0	100		03/12/2021 ...	8m	8	IDBBPGWR ⓘ	DB2	DB2 Statistics	CA11 > DBN1 > BP7 > IDBBPGWR	0	↗	Closed	

Pattern recognition is the process which can detect different categories and get information about particular data, predicting a label of an observation, such as: high, medium and low.

(1) https://medium.com/edureka/pattern-recognition-5e2d30ab68b9

Root Cause Analysis

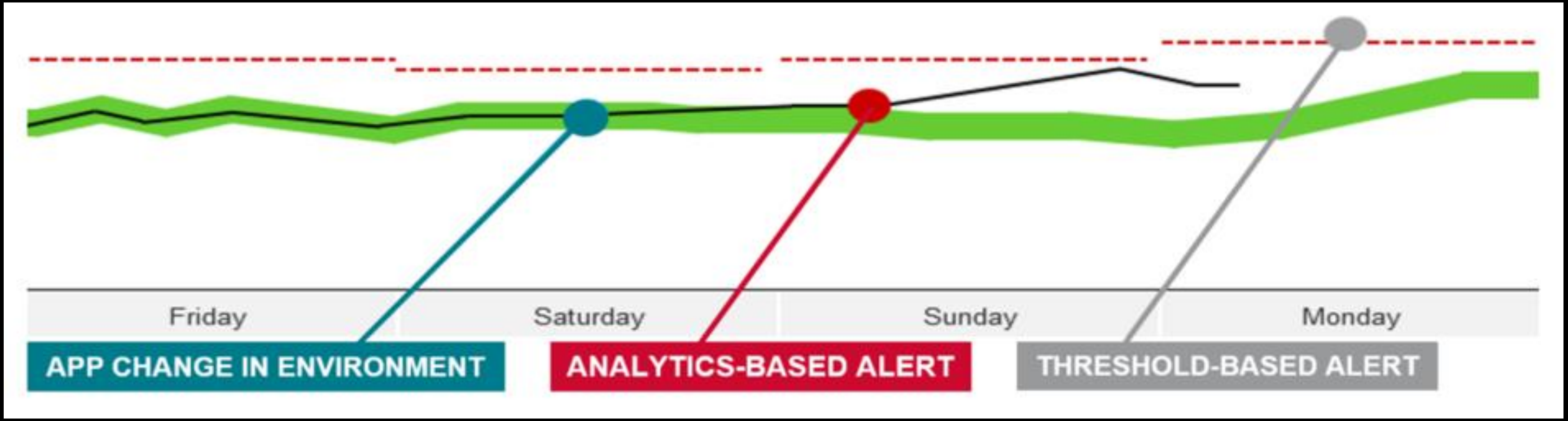
Steps to Root Cause Analysis


Raw Events From Multiple Sources
Noise Reduction
Clustering Events into correlated Situations
Identify Probable Root Cause
AI driven Intelligent Remediation
Heap Issue is the Root Cause -> Run Restart JVM Action
Disk space overflow is the Root Cause -> Run Clean up Disk Space Action
Raw Alarms From Network,Infrastructure and Application
Filter Noisy Alarms from all alarms
Cluster similar alarms into situations
Identify Causal Flow and Probable Root cause for Each Situation
Recommed Remedial Actions through AI driven Intelligent Remediation

(1) https://towardsdatascience.com/how-to-conduct-a-proper-root-cause-analysis-789b9847f84b
(2) https://www.broadcom.cn/aiops-blog/reduce-toil-with-AI-driven-intelligent-remediation

Dynamic and Adaptive Alerting (1|2)

***ssid*DIST DB2 CPU Utilization**



Analytics-Based Alerts detect signal from noise

- Subtler than “**human-observed**”
- Subtler than **static thresholds** which may be the “**last defense**”

Dynamic and Adaptive Alerting (2|2)

Green Highway

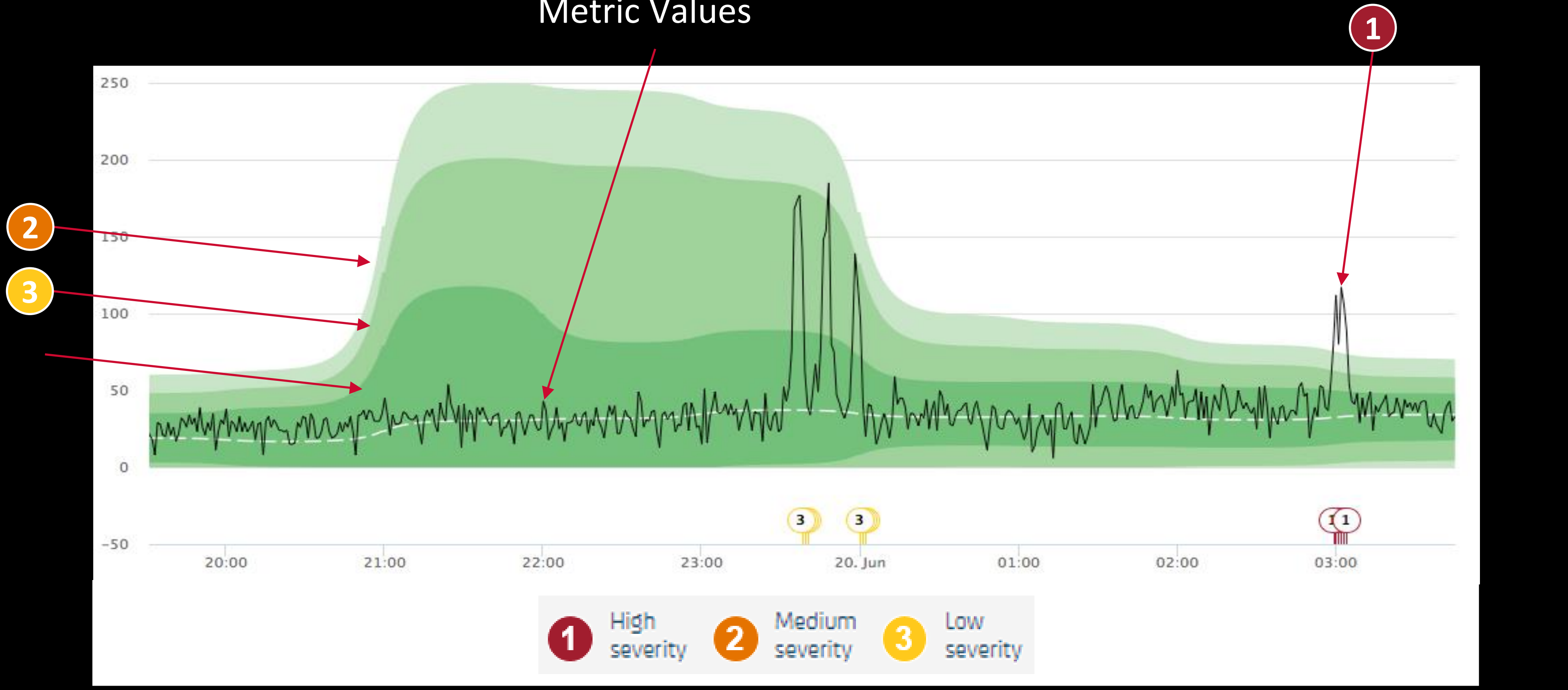
Outside the Bands =
Anomaly / Possible Alert

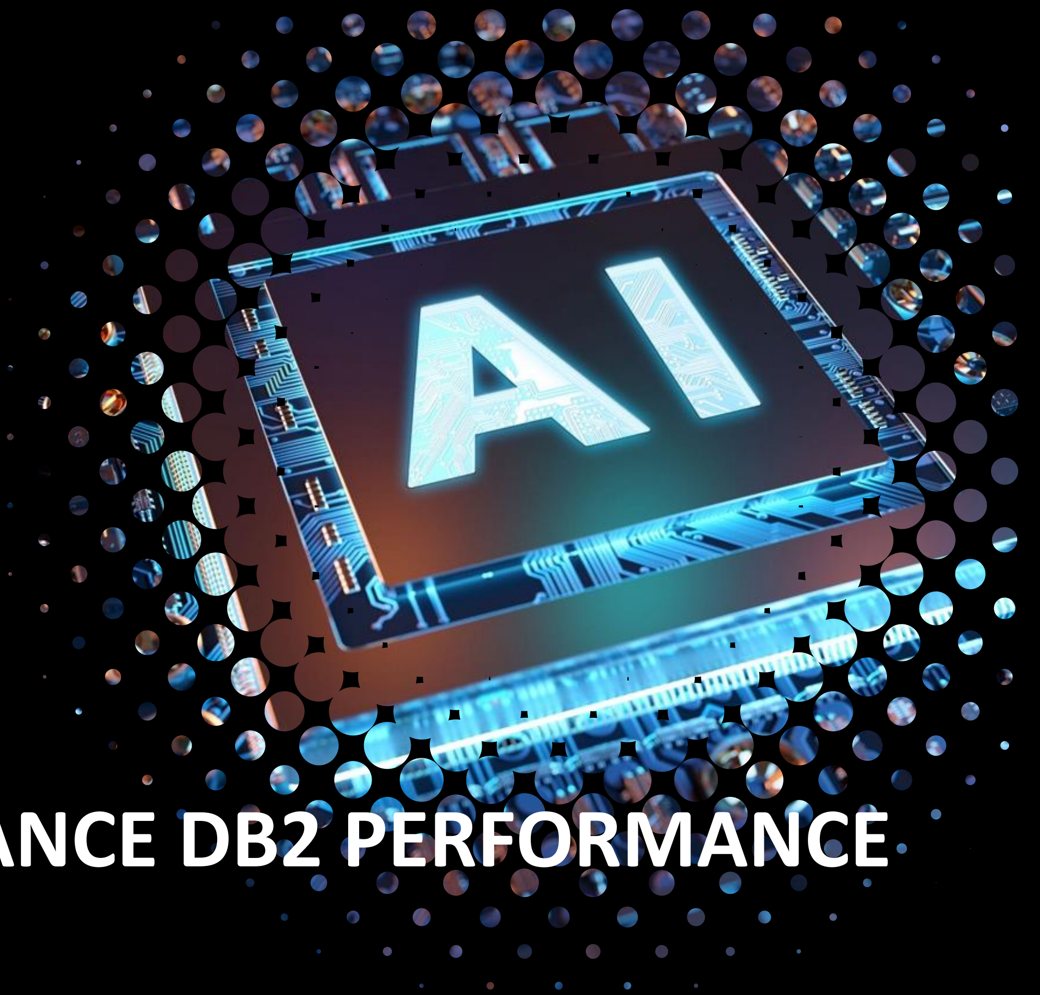
Metric Values

Rare Band

Probable Band

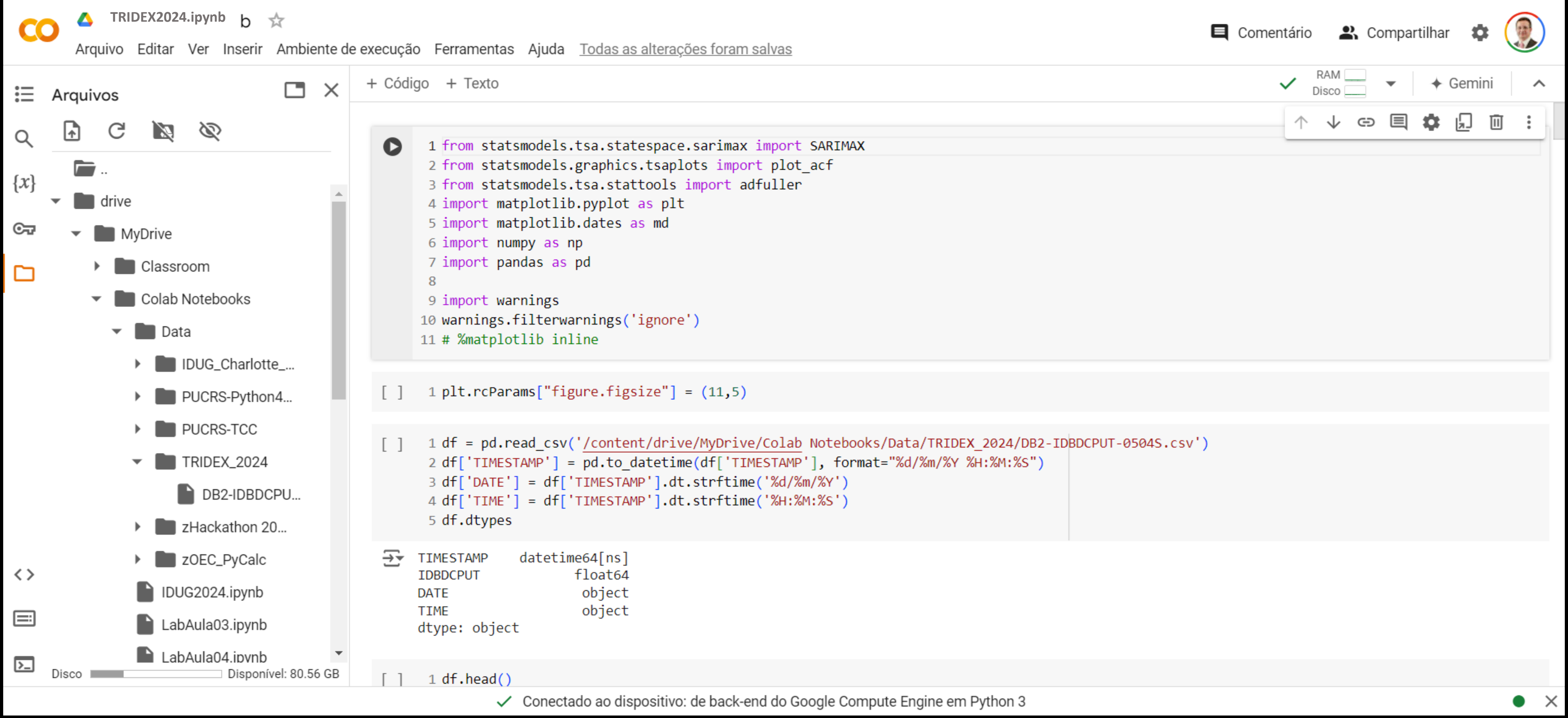
Common Band





APPLIED MACHINE LEARNING TO ENHANCE DB2 PERFORMANCE

Google **Colab** Environment



Todas as alterações foram salvas

```python
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import matplotlib.dates as md
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
# %matplotlib inline
```

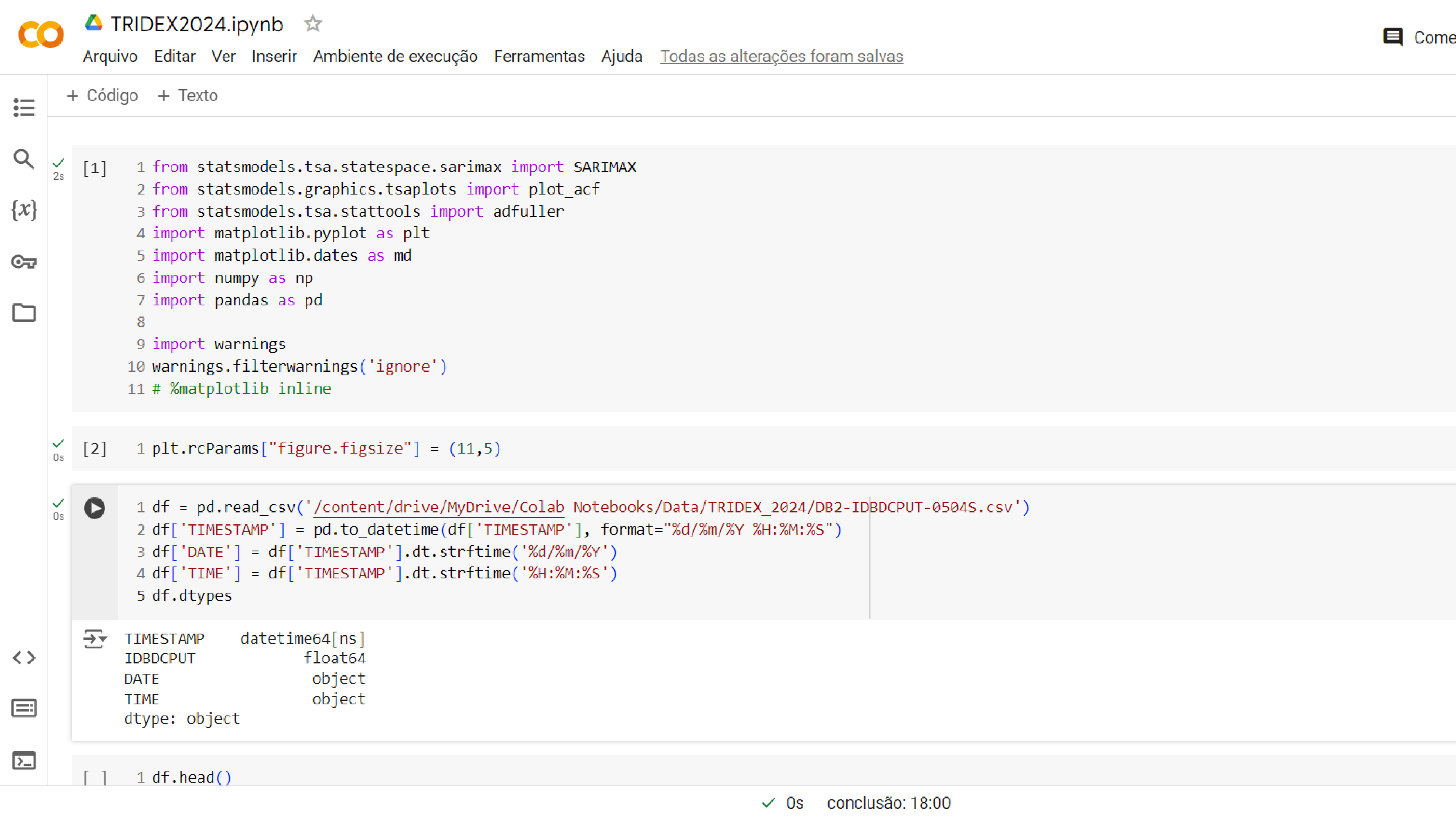
```python
plt.rcParams["figure.figsize"] = (11,5)
```

```python
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data/TRIDEX_2024/DB2-IDBDCPUT-0504S.csv')
df['TIMESTAMP'] = pd.to_datetime(df['TIMESTAMP'], format="%d/%m/%Y %H:%M:%S")
df['DATE'] = df['TIMESTAMP'].dt.strftime('%d/%m/%Y')
df['TIME'] = df['TIMESTAMP'].dt.strftime('%H:%M:%S')
df.dtypes
```

```
TIMESTAMP    datetime64[ns]
IDBDCPUT            float64
DATE                 object
TIME                 object
dtype: object
```

```python
df.head()
```

✓ Conectado ao dispositivo: de back-end do Google Compute Engine em Python 3



[1]

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import matplotlib.dates as md
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings('ignore')
# %matplotlib inline
```

[2]

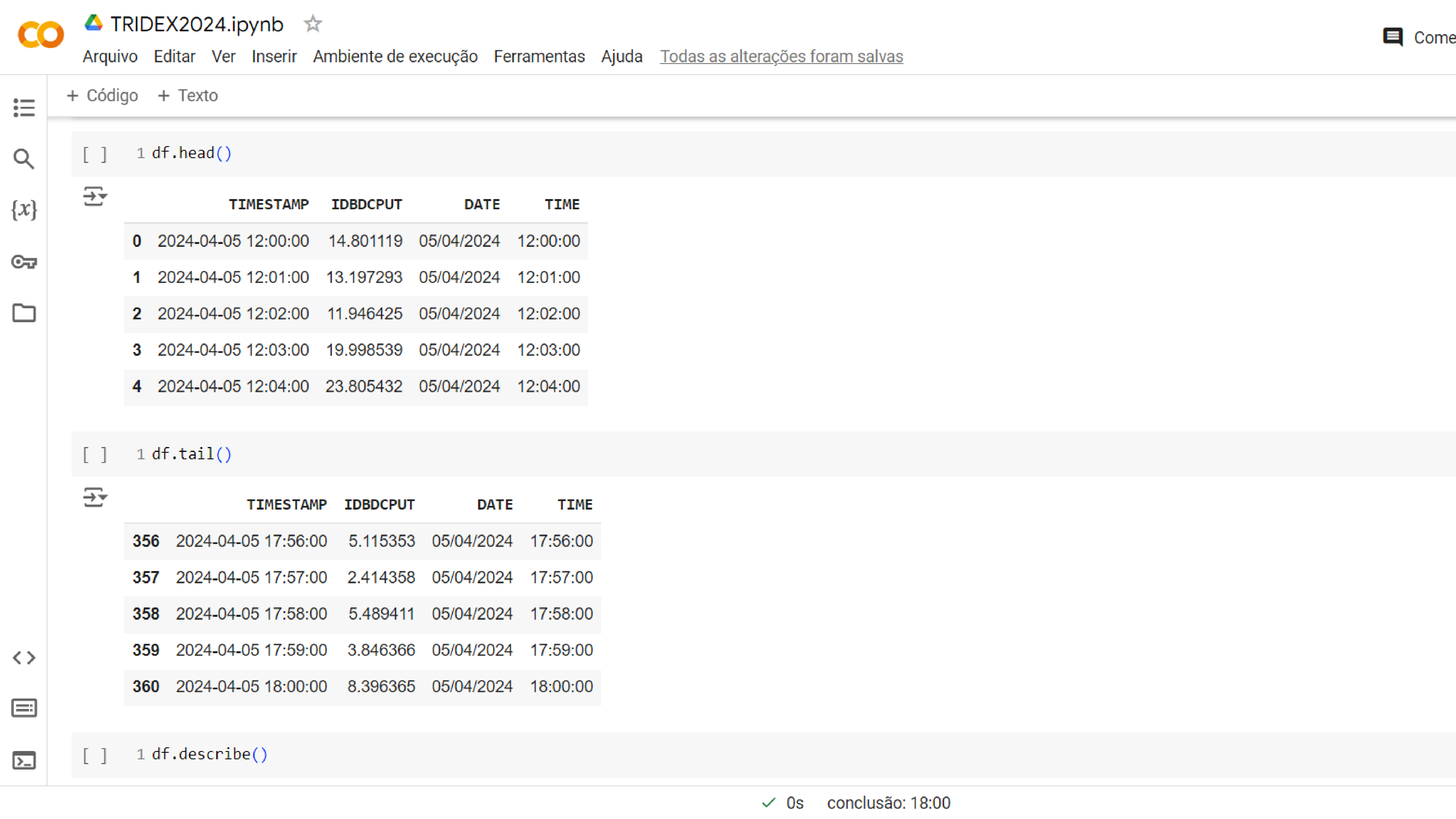
```
plt.rcParams["figure.figsize"] = (11,5)
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data/TRIDEX_2024/DB2-IDBDCPUT-0504S.csv')
df['TIMESTAMP'] = pd.to_datetime(df['TIMESTAMP'], format="%d/%m/%Y %H:%M:%S")
df['DATE'] = df['TIMESTAMP'].dt.strftime('%d/%m/%Y')
df['TIME'] = df['TIMESTAMP'].dt.strftime('%H:%M:%S')
df.dtypes
```

```
TIMESTAMP    datetime64[ns]
IDBDCPUT            float64
DATE                 object
TIME                 object
dtype: object
```

[ ]

```
df.head()
```


TRIDEX2024.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas
Come
+ Código + Texto
✓ 0s conclusão: 18:00

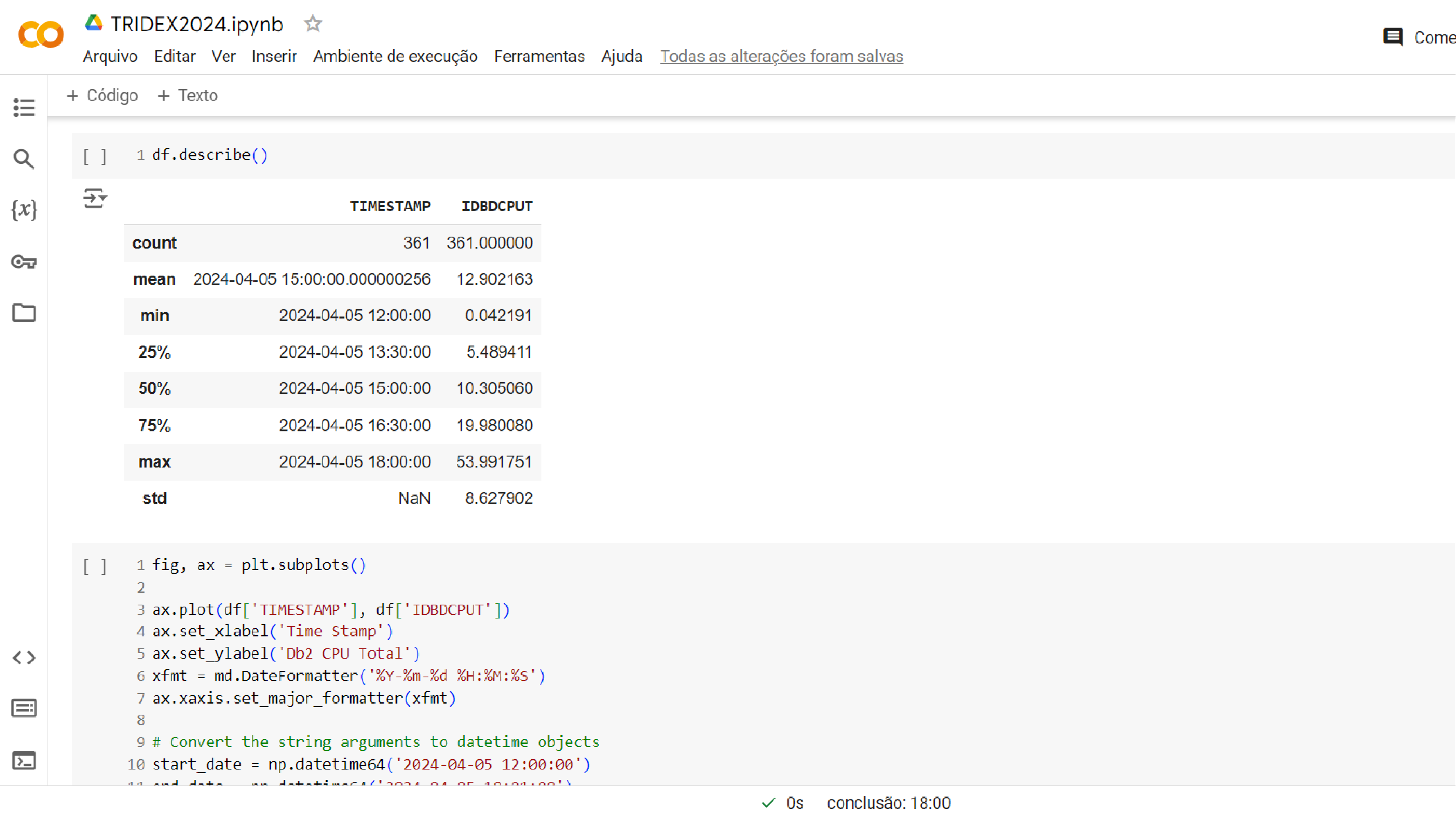
```
df.head()
```

	TIMESTAMP	IDBDCPUT	DATE	TIME
0	2024-04-05 12:00:00	14.801119	05/04/2024	12:00:00
1	2024-04-05 12:01:00	13.197293	05/04/2024	12:01:00
2	2024-04-05 12:02:00	11.946425	05/04/2024	12:02:00
3	2024-04-05 12:03:00	19.998539	05/04/2024	12:03:00
4	2024-04-05 12:04:00	23.805432	05/04/2024	12:04:00

```
df.tail()
```

	TIMESTAMP	IDBDCPUT	DATE	TIME
356	2024-04-05 17:56:00	5.115353	05/04/2024	17:56:00
357	2024-04-05 17:57:00	2.414358	05/04/2024	17:57:00
358	2024-04-05 17:58:00	5.489411	05/04/2024	17:58:00
359	2024-04-05 17:59:00	3.846366	05/04/2024	17:59:00
360	2024-04-05 18:00:00	8.396365	05/04/2024	18:00:00

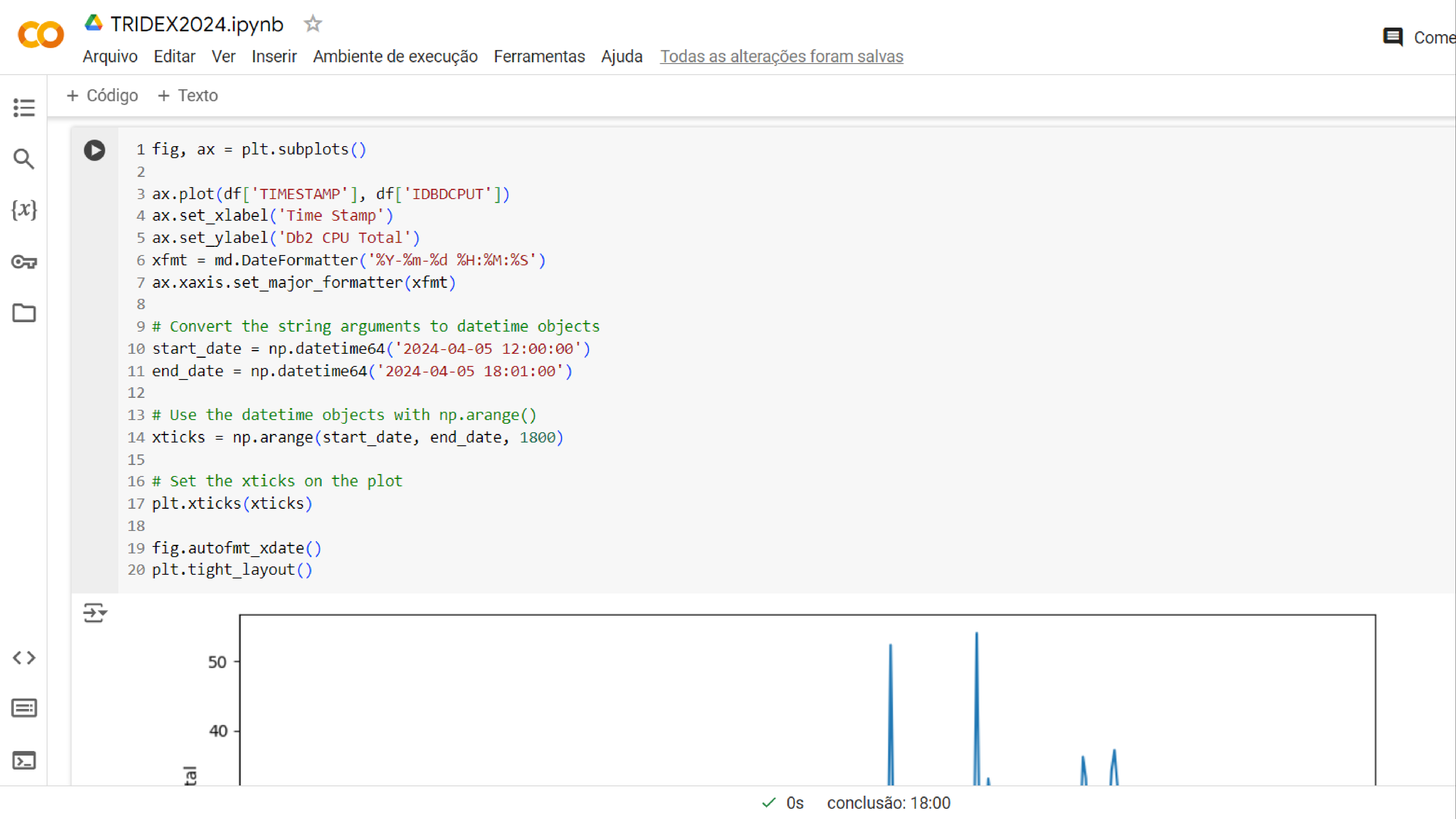
```
df.describe()
```



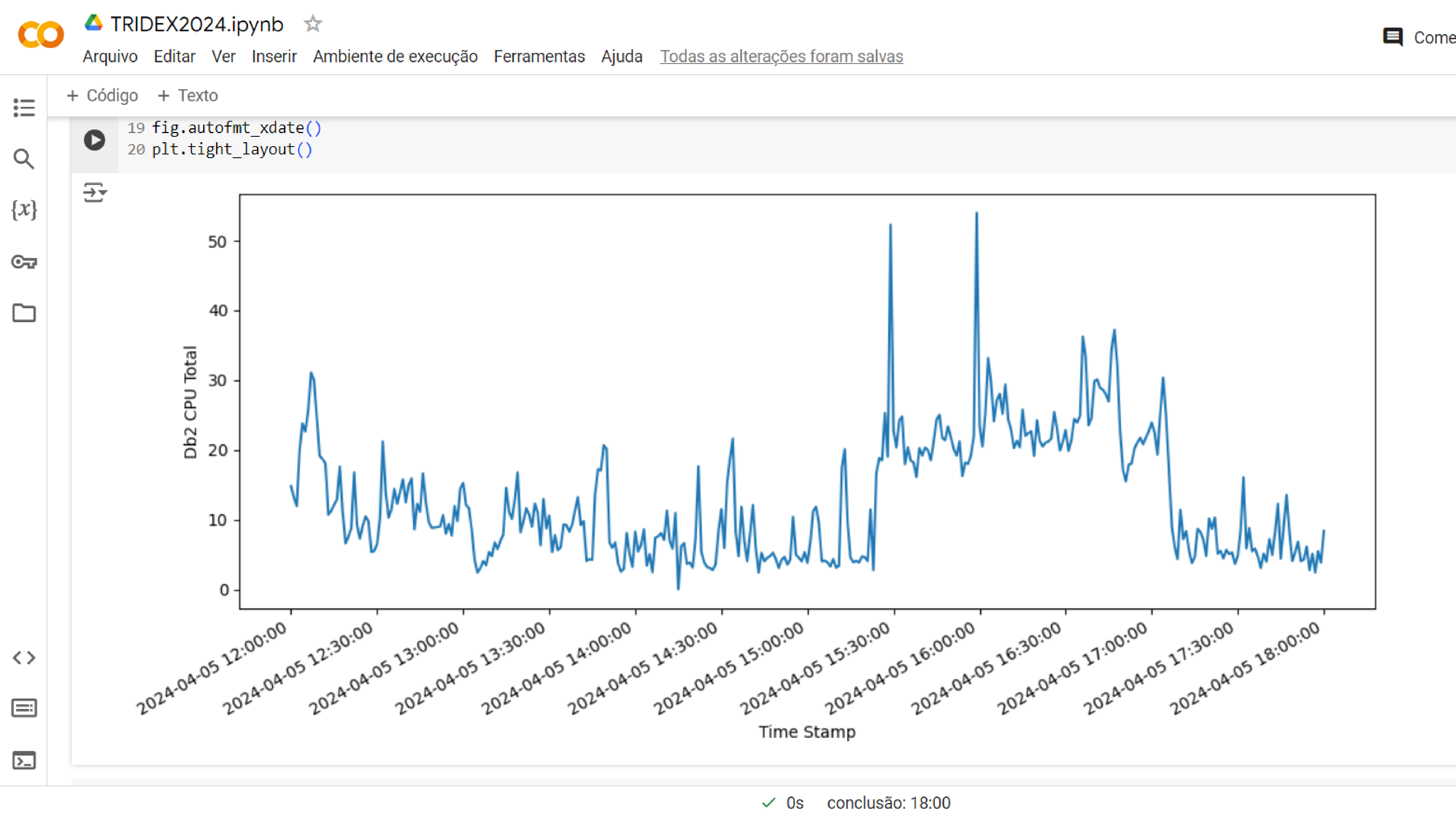
```
df.describe()
```

	TIMESTAMP	IDBDCPUT
count	361	361.000000
mean	2024-04-05 15:00:00.000000256	12.902163
min	2024-04-05 12:00:00	0.042191
25%	2024-04-05 13:30:00	5.489411
50%	2024-04-05 15:00:00	10.305060
75%	2024-04-05 16:30:00	19.980080
max	2024-04-05 18:00:00	53.991751
std	NaN	8.627902

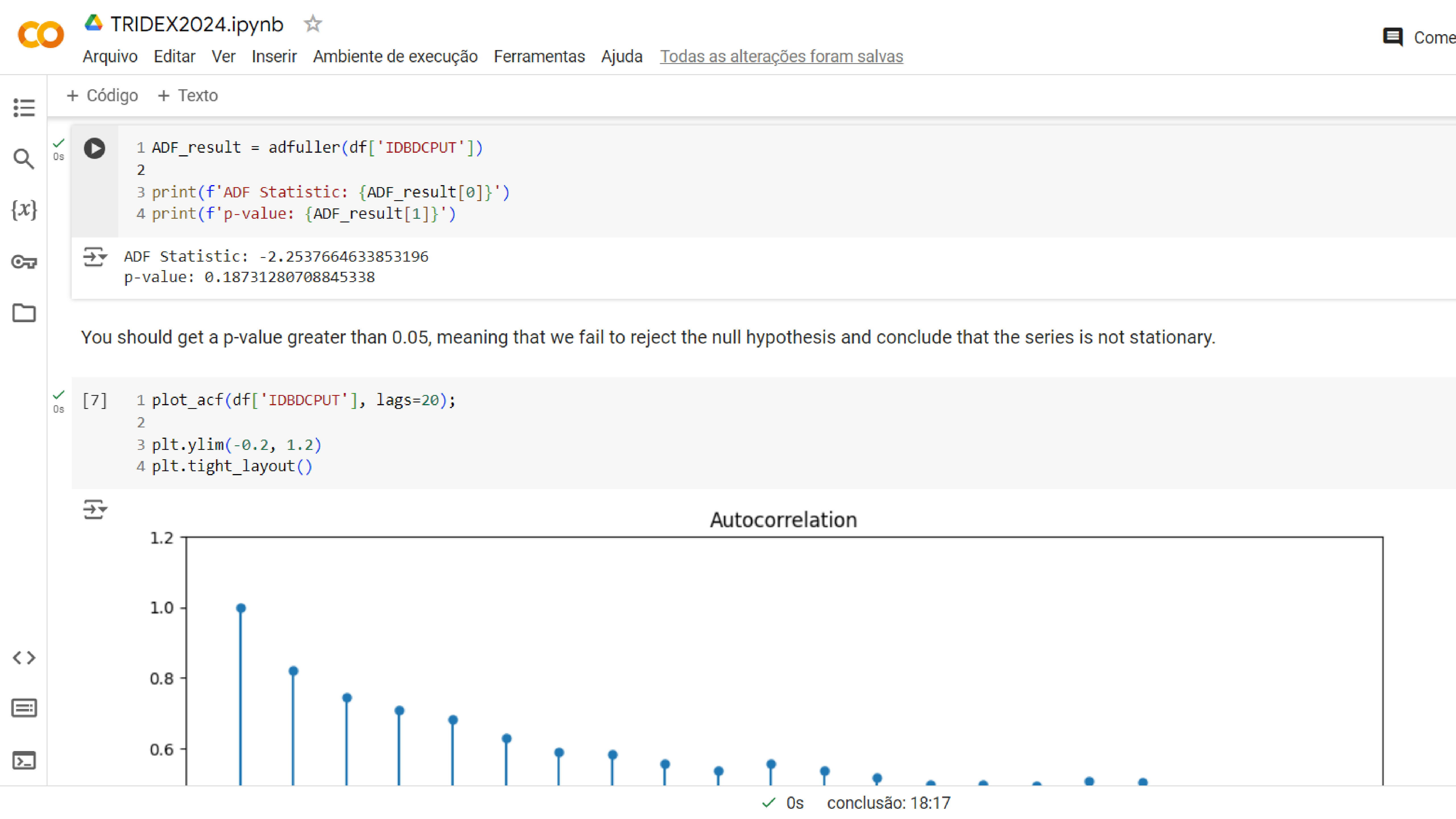
```
fig, ax = plt.subplots()

ax.plot(df['TIMESTAMP'], df['IDBDCPUT'])
ax.set_xlabel('Time Stamp')
ax.set_ylabel('Db2 CPU Total')
xfmt = md.DateFormatter('%Y-%m-%d %H:%M:%S')
ax.xaxis.set_major_formatter(xfmt)

# Convert the string arguments to datetime objects
start_date = np.datetime64('2024-04-05 12:00:00')
end_date = np.datetime64('2024-04-05 18:01:00')
```



```
fig, ax = plt.subplots()

ax.plot(df['TIMESTAMP'], df['IDBDCPUT'])
ax.set_xlabel('Time Stamp')
ax.set_ylabel('Db2 CPU Total')
xfmt = md.DateFormatter('%Y-%m-%d %H:%M:%S')
ax.xaxis.set_major_formatter(xfmt)

# Convert the string arguments to datetime objects
start_date = np.datetime64('2024-04-05 12:00:00')
end_date = np.datetime64('2024-04-05 18:01:00')

# Use the datetime objects with np.arange()
xticks = np.arange(start_date, end_date, 1800)

# Set the xticks on the plot
plt.xticks(xticks)

fig.autofmt_xdate()
plt.tight_layout()
```



```
fig.autofmt_xdate()
plt.tight_layout()
```

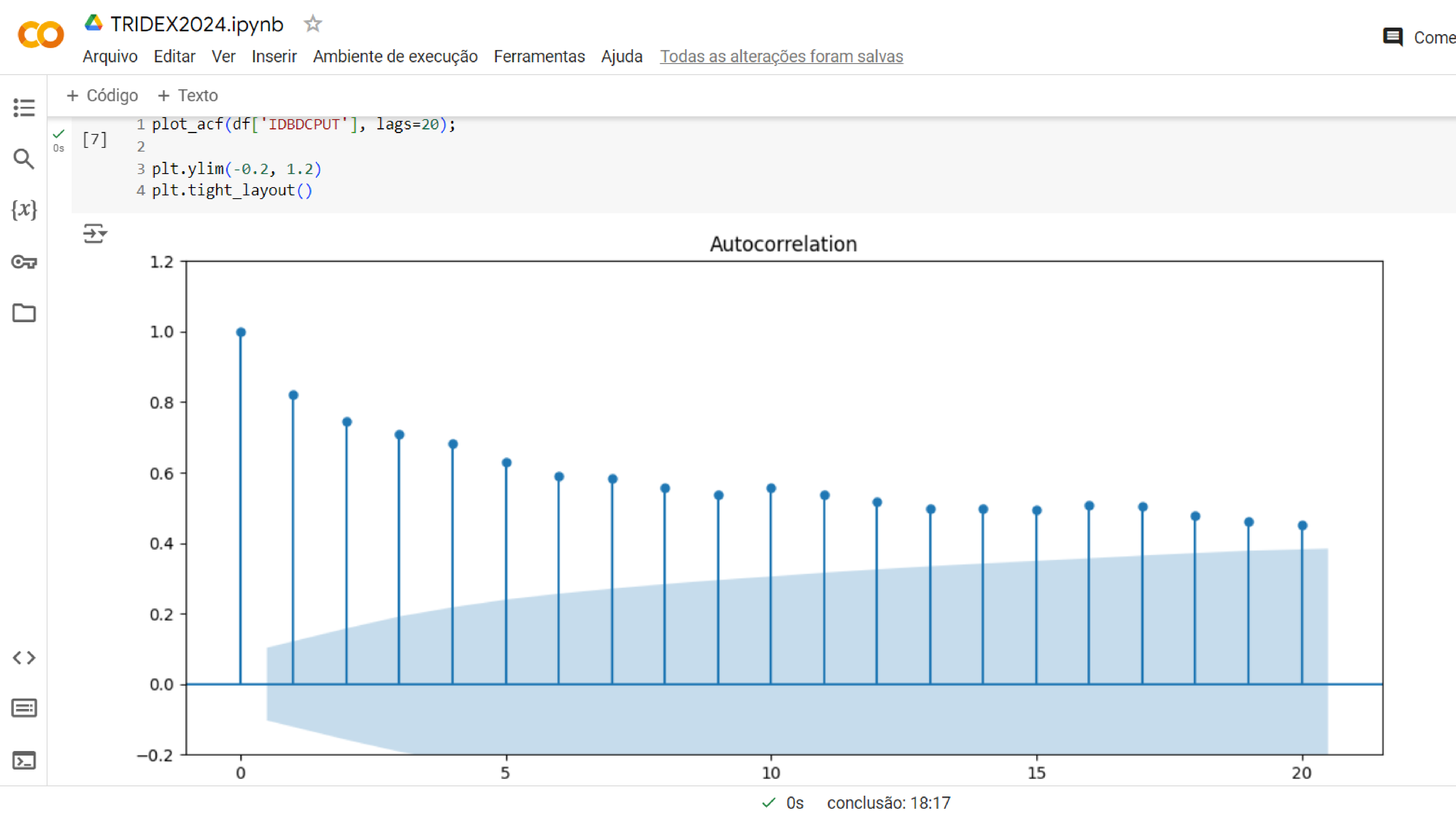


```
ADF_result = adfuller(df['IDBDCPUT'])

print(f'ADF Statistic: {ADF_result[0]}')
print(f'p-value: {ADF_result[1]}')
```

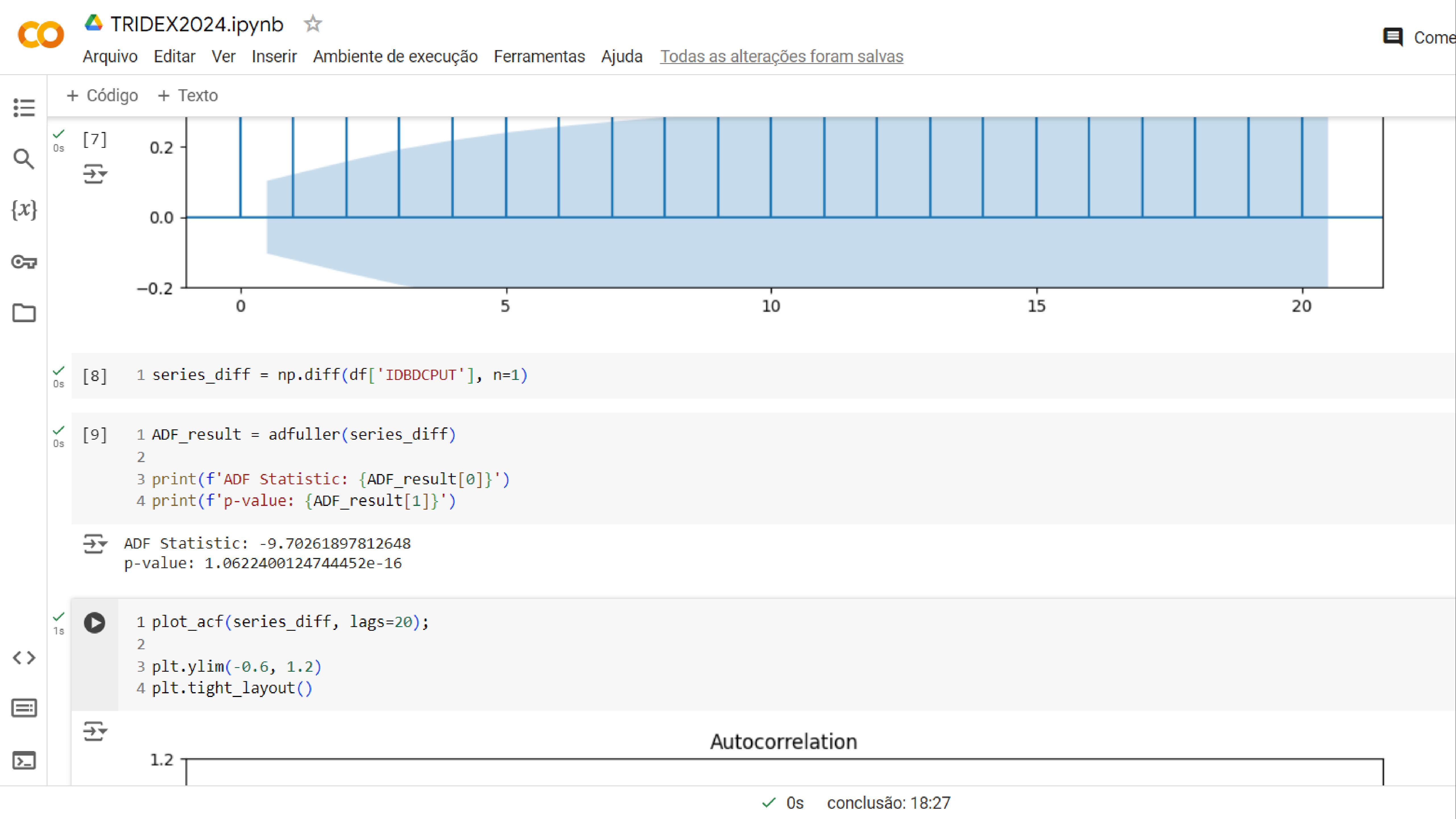
```
ADF Statistic: -2.2537664633853196
p-value: 0.18731280708845338
```

You should get a p-value greater than 0.05, meaning that we fail to reject the null hypothesis and conclude that the series is not stationary.

```
plot_acf(df['IDBDCPUT'], lags=20);

plt.ylim(-0.2, 1.2)
plt.tight_layout()
```



```
plot_acf(df['IDBDCPUT'], lags=20);

plt.ylim(-0.2, 1.2)
plt.tight_layout()
```

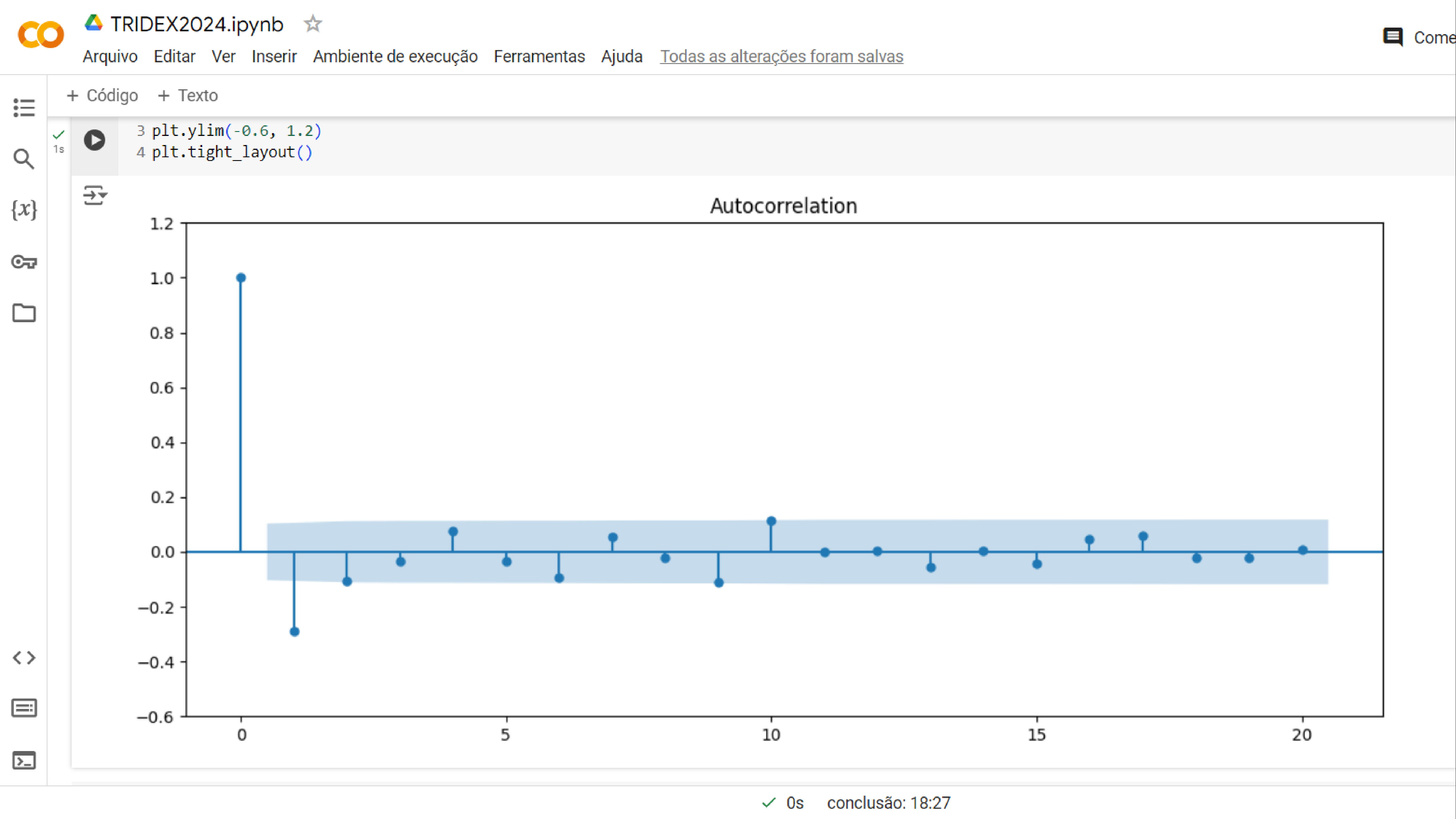


```python
series_diff = np.diff(df['IDBDCPUT'], n=1)
```

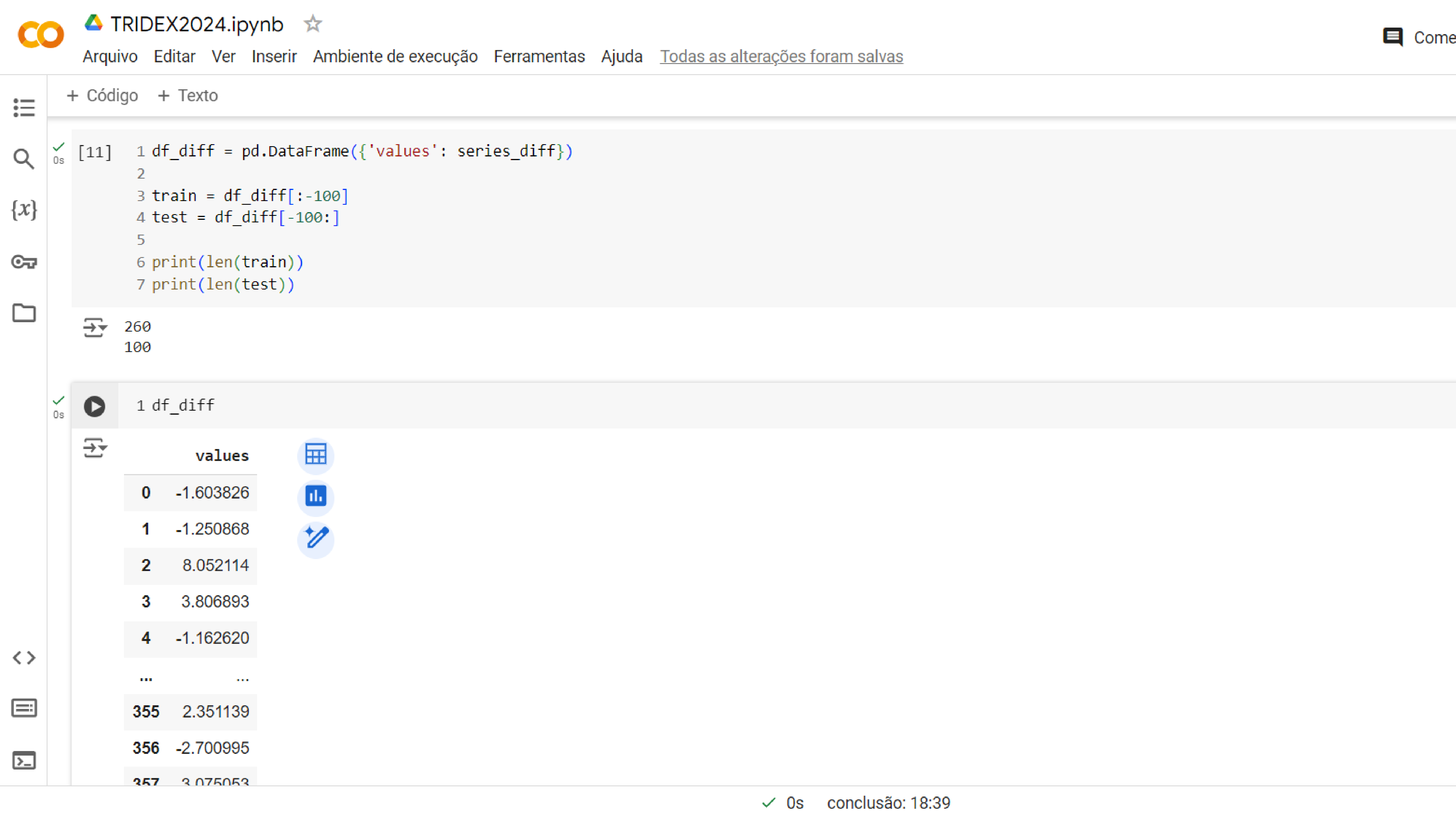
```python
ADF_result = adfuller(series_diff)

print(f'ADF Statistic: {ADF_result[0]}')
print(f'p-value: {ADF_result[1]}')
```

```
ADF Statistic: -9.70261897812648
p-value: 1.0622400124744452e-16
```

```python
plot_acf(series_diff, lags=20);

plt.ylim(-0.6, 1.2)
plt.tight_layout()
```



```
plt.ylim(-0.6, 1.2)
plt.tight_layout()
```

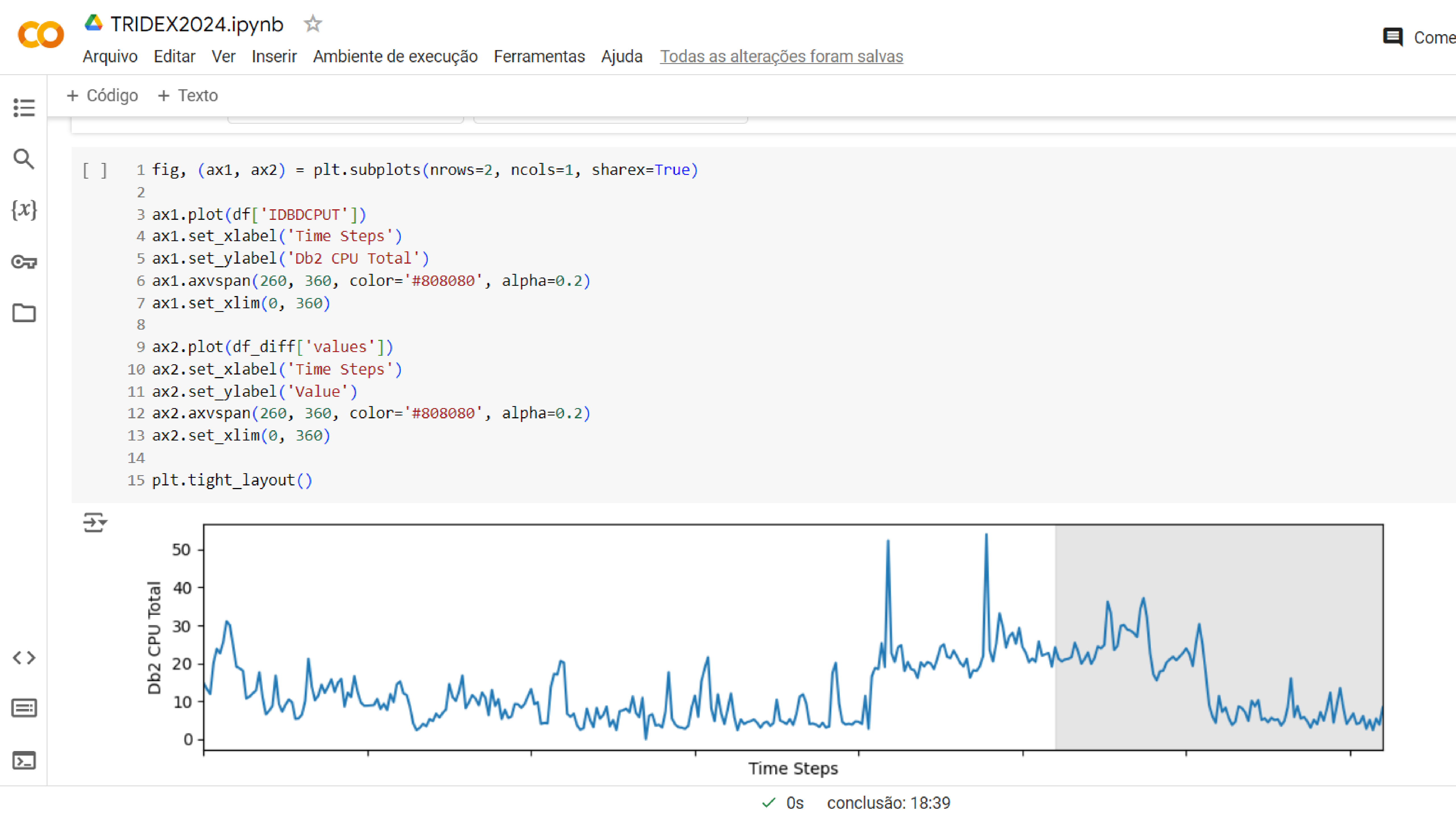


```python
df_diff = pd.DataFrame({'values': series_diff})

train = df_diff[:-100]
test = df_diff[-100:]

print(len(train))
print(len(test))
```

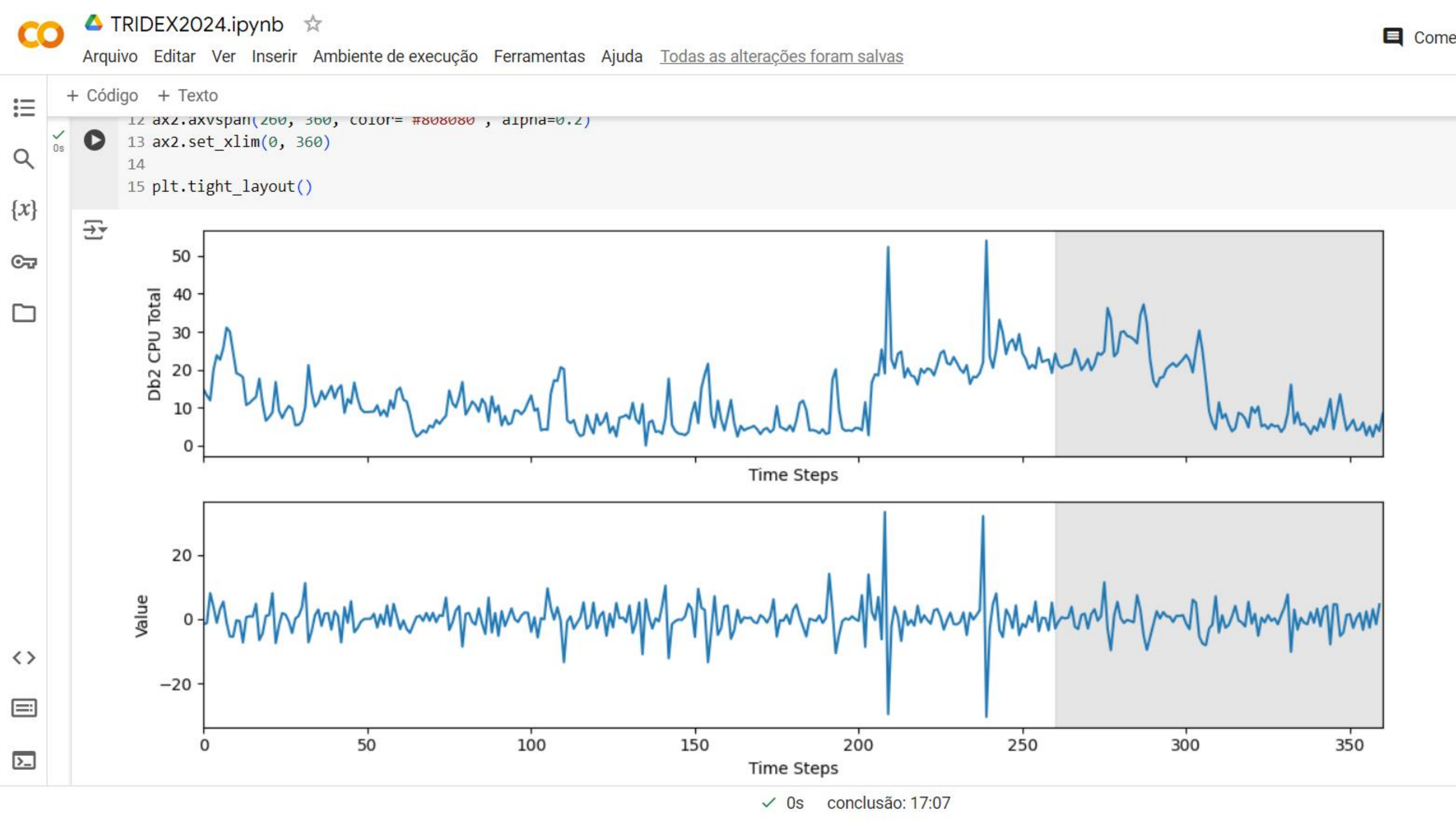
```
260
100
```

```python
df_diff
```

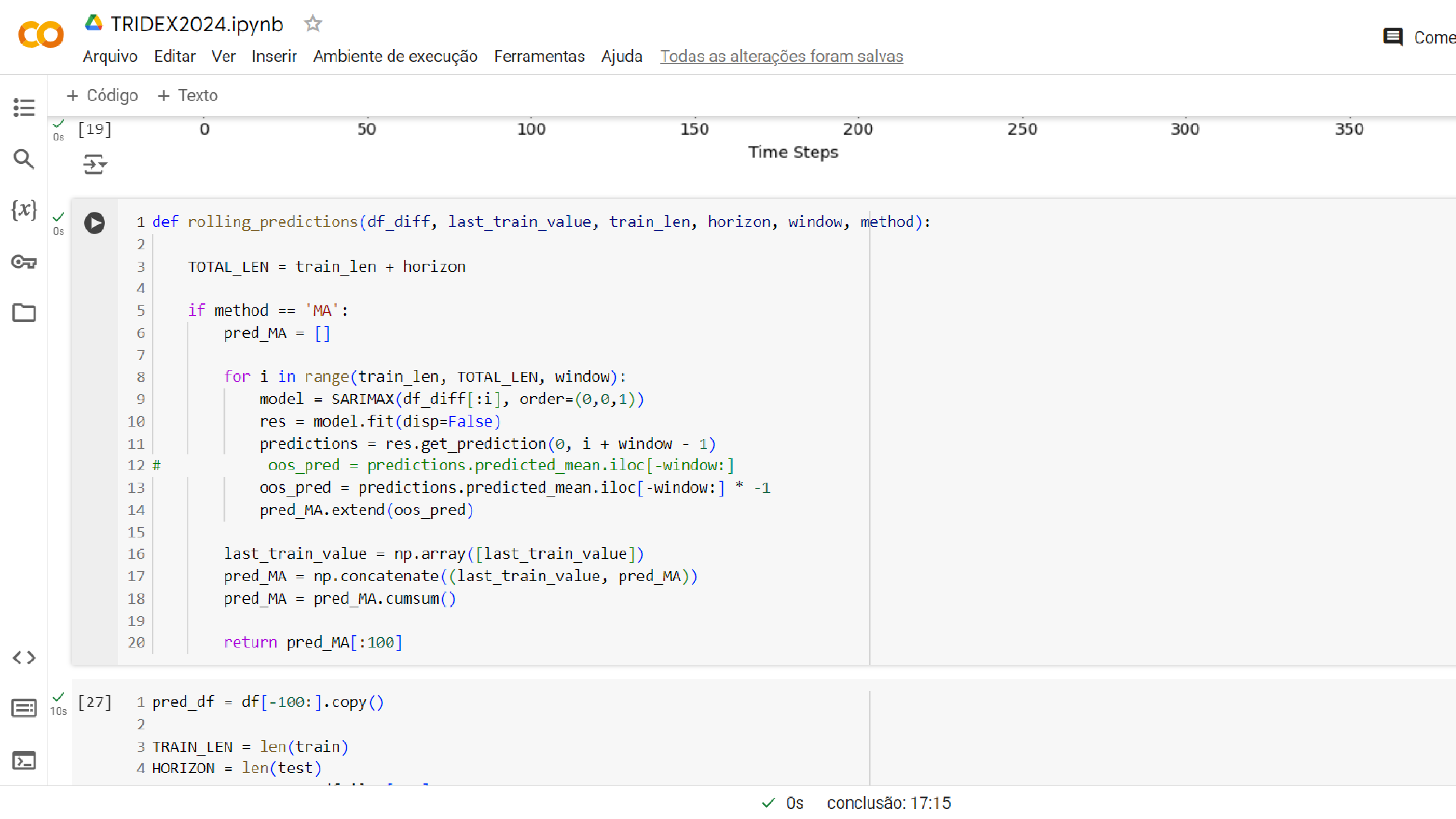
	values
0	-1.603826
1	-1.250868
2	8.052114
3	3.806893
4	-1.162620
...	...
355	2.351139
356	-2.700995
357	3.075053



```
fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, sharex=True)

ax1.plot(df['IDBDCPUT'])
ax1.set_xlabel('Time Steps')
ax1.set_ylabel('Db2 CPU Total')
ax1.axvspan(260, 360, color='#808080', alpha=0.2)
ax1.set_xlim(0, 360)

ax2.plot(df_diff['values'])
ax2.set_xlabel('Time Steps')
ax2.set_ylabel('Value')
ax2.axvspan(260, 360, color='#808080', alpha=0.2)
ax2.set_xlim(0, 360)

plt.tight_layout()
```

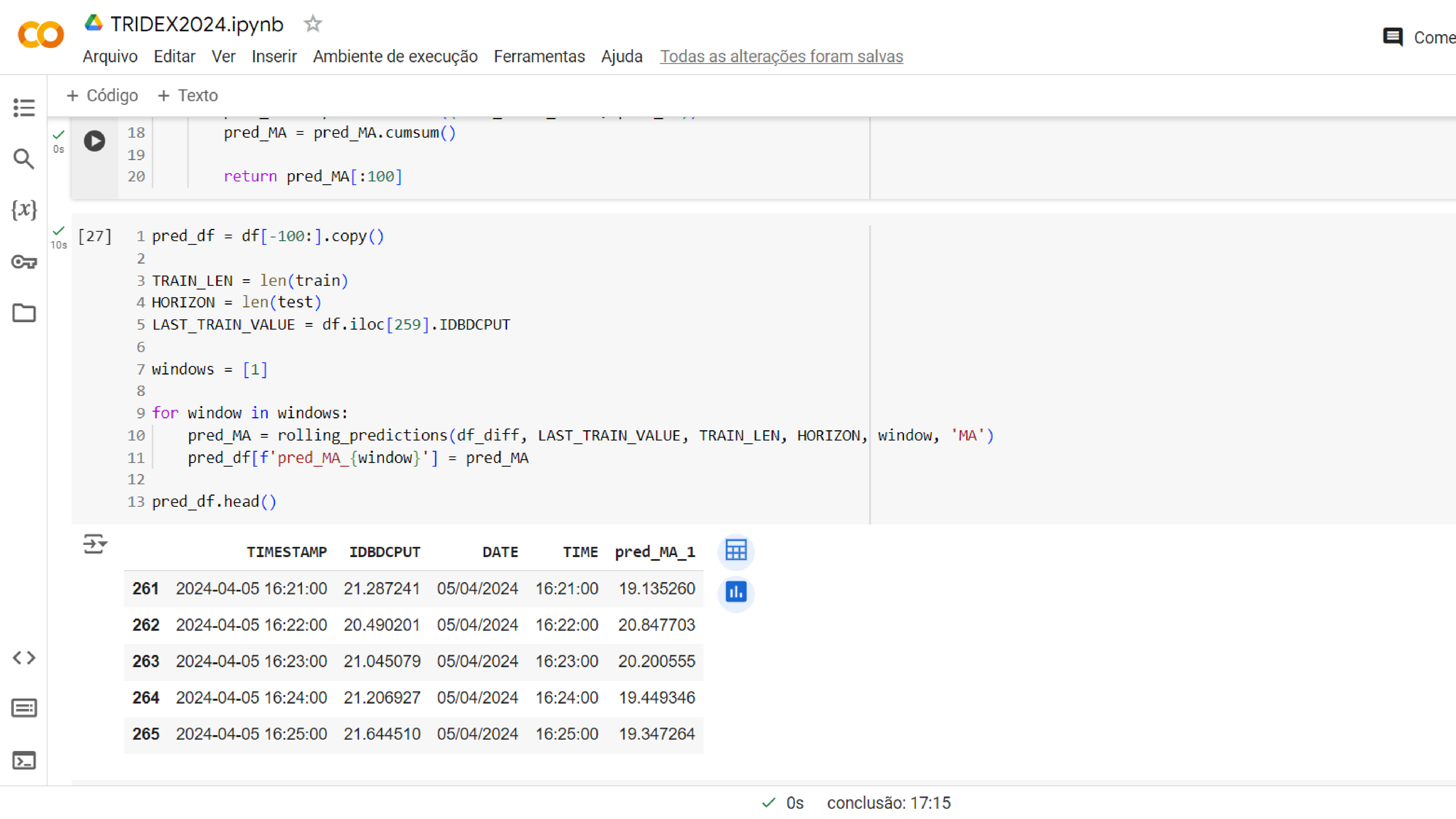


```
ax2.axvspan(260, 360, color='#808080', alpha=0.2)
ax2.set_xlim(0, 360)

plt.tight_layout()
```



```python
def rolling_predictions(df_diff, last_train_value, train_len, horizon, window, method):

    TOTAL_LEN = train_len + horizon

    if method == 'MA':
        pred_MA = []

        for i in range(train_len, TOTAL_LEN, window):
            model = SARIMAX(df_diff[:i], order=(0,0,1))
            res = model.fit(disp=False)
            predictions = res.get_prediction(0, i + window - 1)
#            oos_pred = predictions.predicted_mean.iloc[-window:]
            oos_pred = predictions.predicted_mean.iloc[-window:] * -1
            pred_MA.extend(oos_pred)

        last_train_value = np.array([last_train_value])
        pred_MA = np.concatenate((last_train_value, pred_MA))
        pred_MA = pred_MA.cumsum()

        return pred_MA[:100]
```

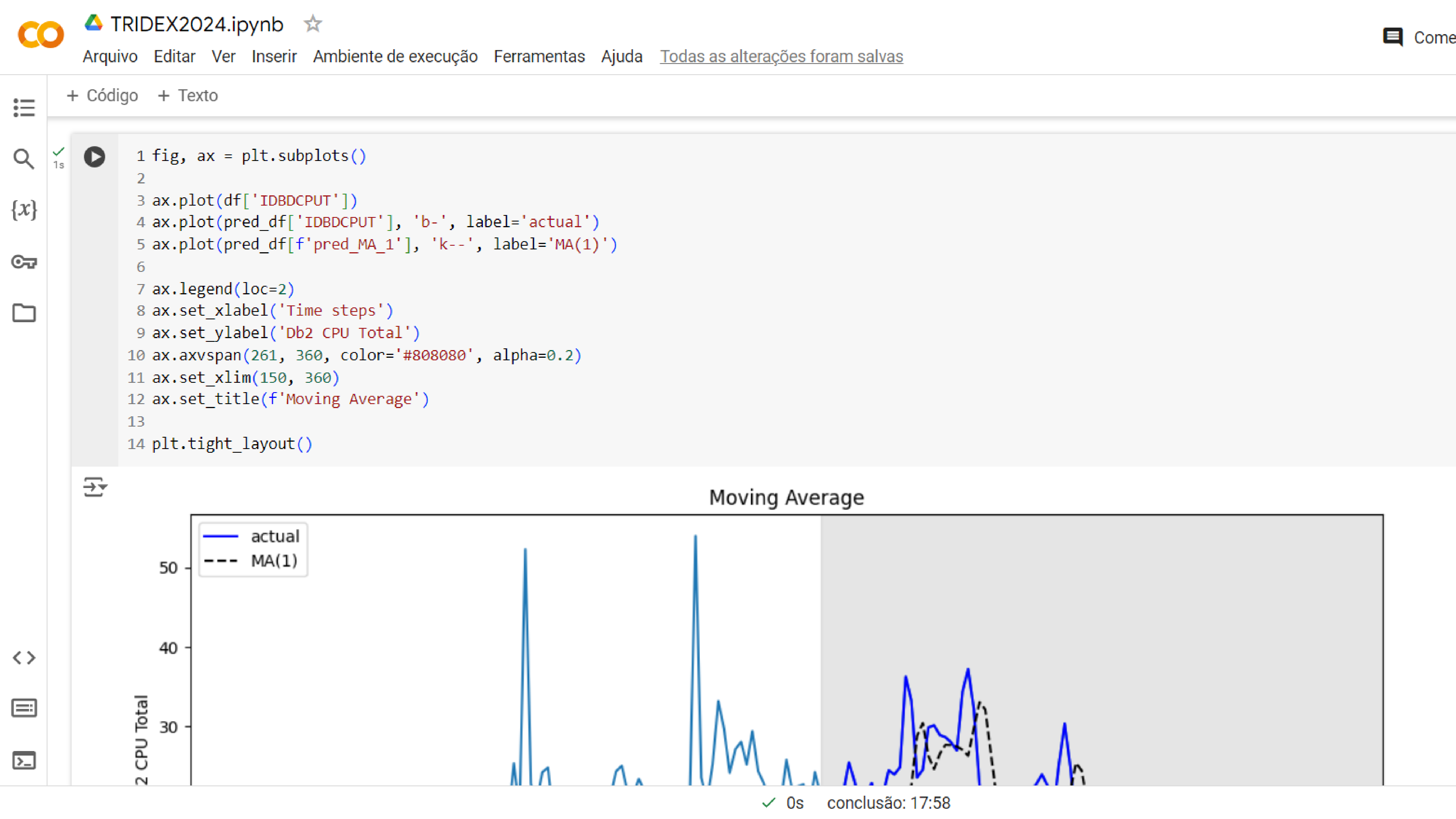
```python
pred_df = df[-100:].copy()

TRAIN_LEN = len(train)
HORIZON = len(test)
```



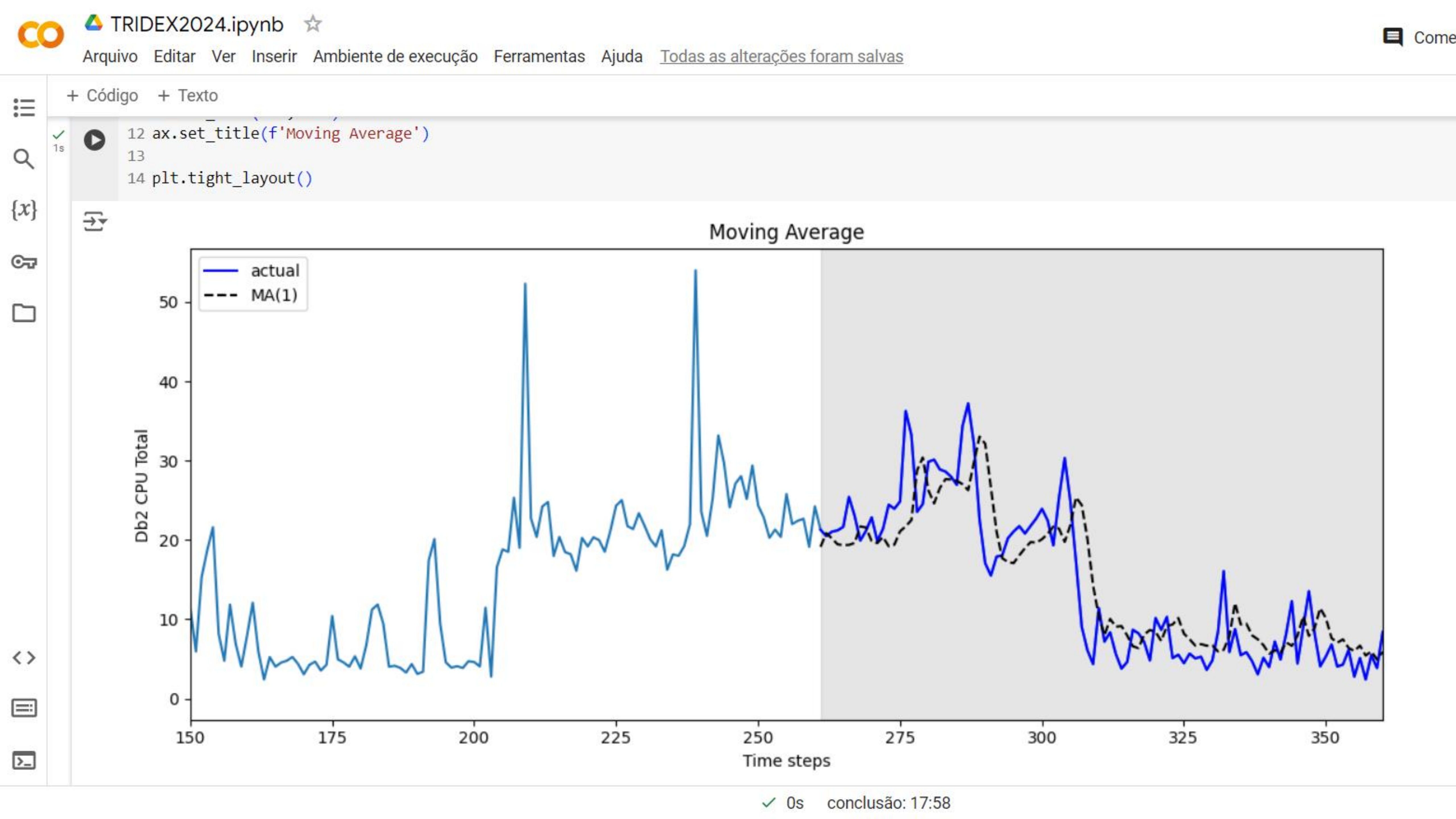
```python
        pred_MA = pred_MA.cumsum()

        return pred_MA[:100]
```

```python
pred_df = df[-100:].copy()

TRAIN_LEN = len(train)
HORIZON = len(test)
LAST_TRAIN_VALUE = df.iloc[259].IDBDCPUT

windows = [1]

for window in windows:
    pred_MA = rolling_predictions(df_diff, LAST_TRAIN_VALUE, TRAIN_LEN, HORIZON, window, 'MA')
    pred_df[f'pred_MA_{window}'] = pred_MA

pred_df.head()
```

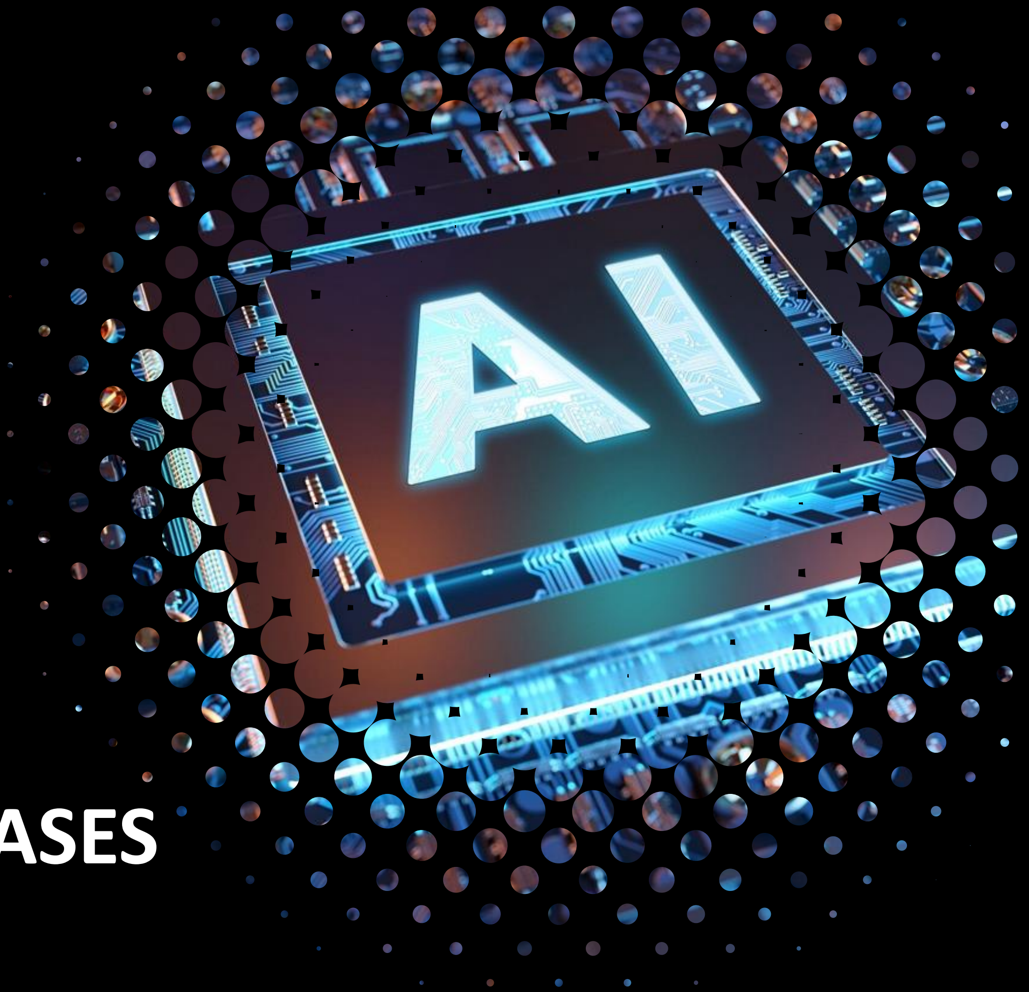
```
               TIMESTAMP   IDBDCPUT        DATE      TIME   pred_MA_1
261  2024-04-05 16:21:00  21.287241  05/04/2024  16:21:00   19.135260
262  2024-04-05 16:22:00  20.490201  05/04/2024  16:22:00   20.847703
263  2024-04-05 16:23:00  21.045079  05/04/2024  16:23:00   20.200555
264  2024-04-05 16:24:00  21.206927  05/04/2024  16:24:00   19.449346
265  2024-04-05 16:25:00  21.644510  05/04/2024  16:25:00   19.347264
```



```python
fig, ax = plt.subplots()

ax.plot(df['IDBDCPUT'])
ax.plot(pred_df['IDBDCPUT'], 'b-', label='actual')
ax.plot(pred_df[f'pred_MA_1'], 'k--', label='MA(1)')

ax.legend(loc=2)
ax.set_xlabel('Time steps')
ax.set_ylabel('Db2 CPU Total')
ax.axvspan(261, 360, color='#808080', alpha=0.2)
ax.set_xlim(150, 360)
ax.set_title(f'Moving Average')

plt.tight_layout()
```



```
ax.set_title(f'Moving Average')

plt.tight_layout()
```



CUSTOMER USE CASES

What makes a **Business Application**?

Web and Mobile apps

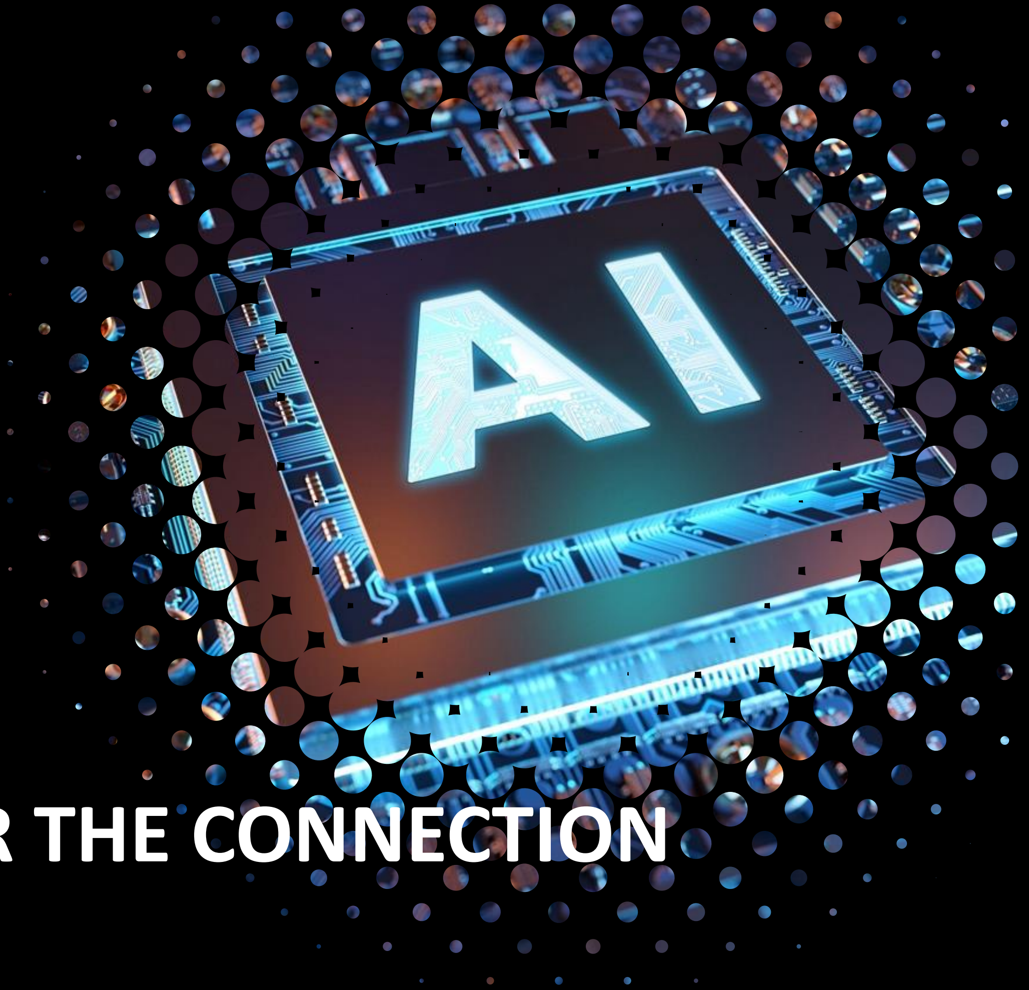
- Distributed App
- Middleware
  - MQ
  - WAS
- Package DISTSERV
- Dynamic SQL Statements
- Tables/Indexes
  - Other DB2 objects

CICS Transaction

- Transaction ID
- Program
- Plan
- Package
- SQL Statements
- Tables/Indexes
  - Other DB2 objects

Batch Job

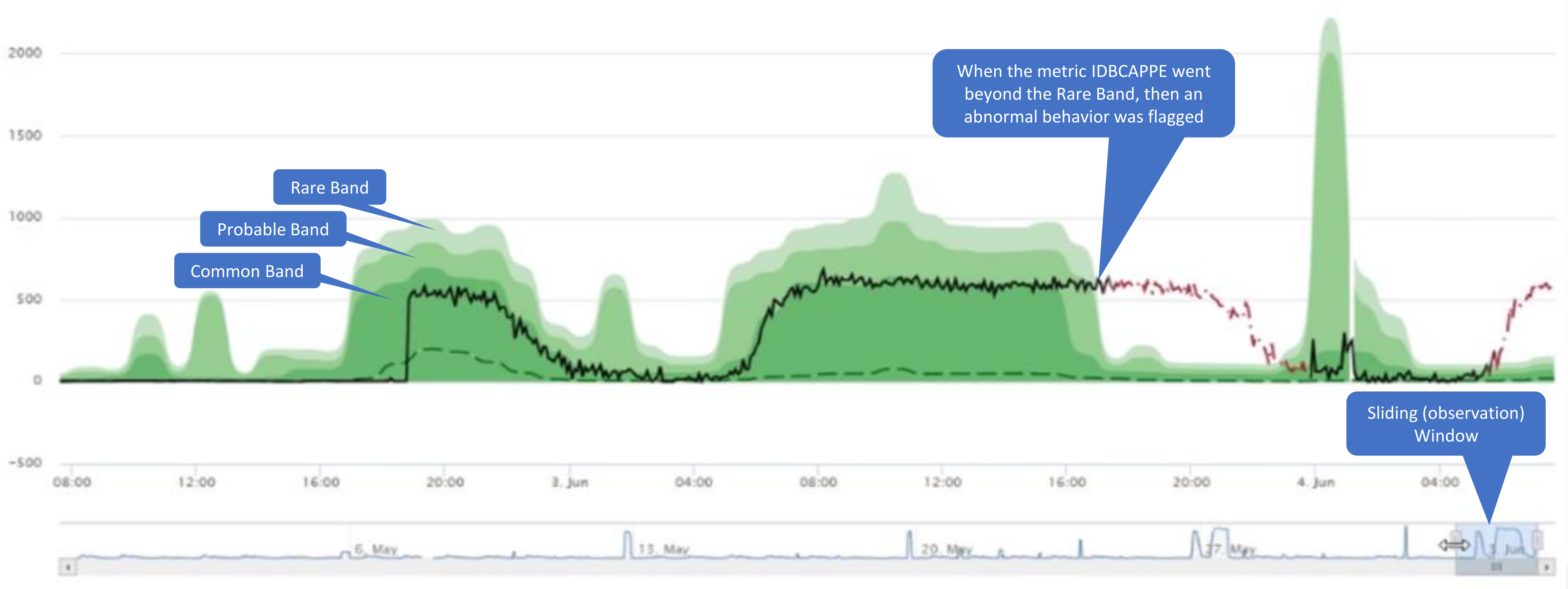
- Program
- Plan
- Package
- SQL Statements
- Tables/Indexes
  - Other DB2 objects



APPLICATION ELAPSED TIME FOR THE CONNECTION

Db2 Application Elapsed Time on CICS (1|2)

- Connection with CICS the metric **IDBCAPPE** (IFCID 369 – field QWACESC)

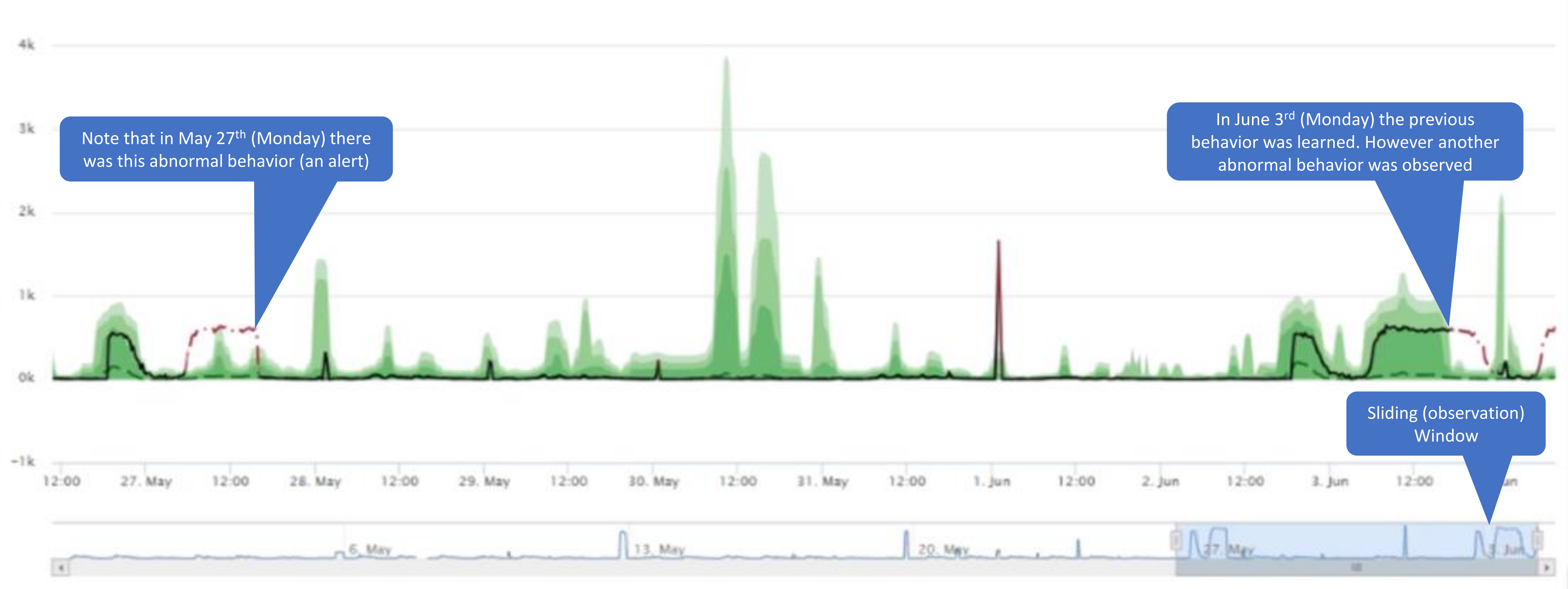


When the metric IDBCAPPE went beyond the Rare Band, then an abnormal behavior was flagged

Sliding (observation) Window

Db2 Application Elapsed Time on CICS (2|2)

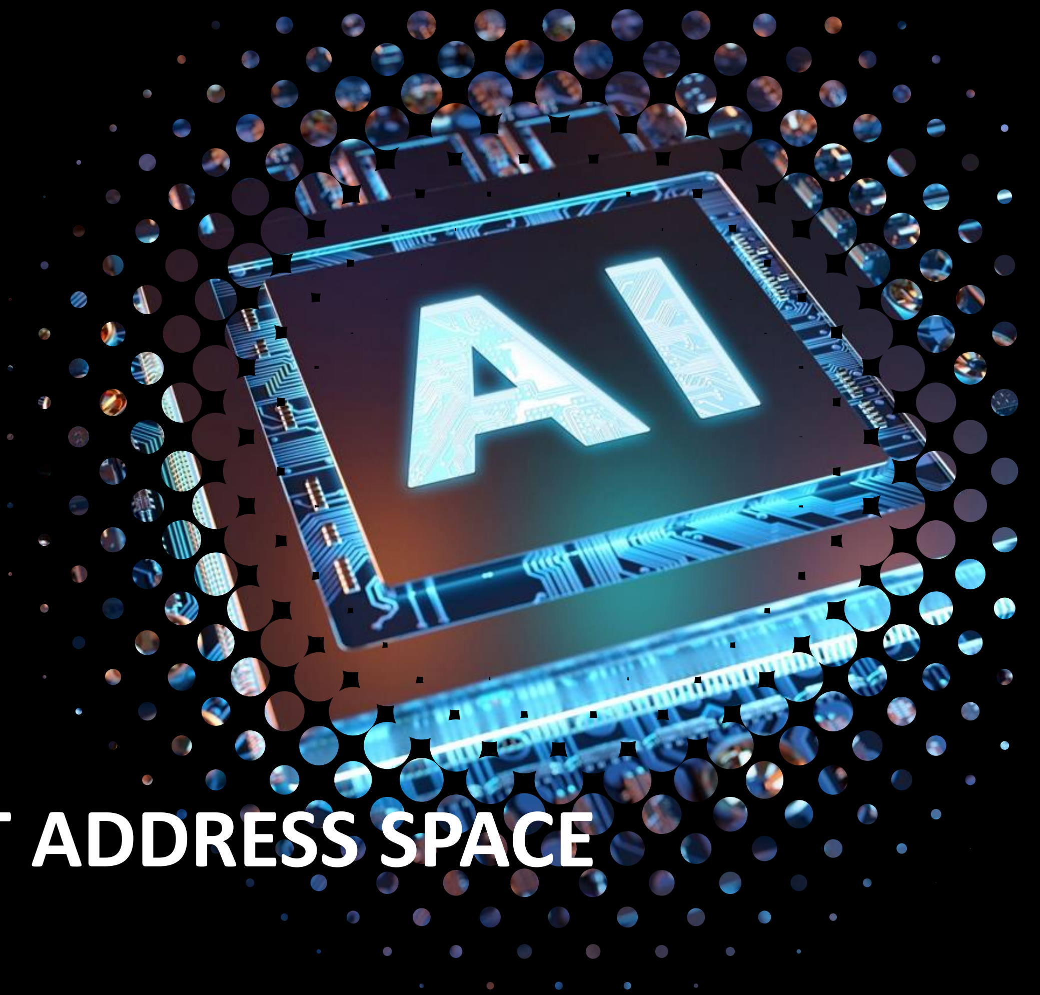
- Connection with CICS the metric **IDBCAPPE** (IFCID 369 – field QWACESC)



Note that in May 27th (Monday) there was this abnormal behavior (an alert)

In June 3rd (Monday) the previous behavior was learned. However another abnormal behavior was observed

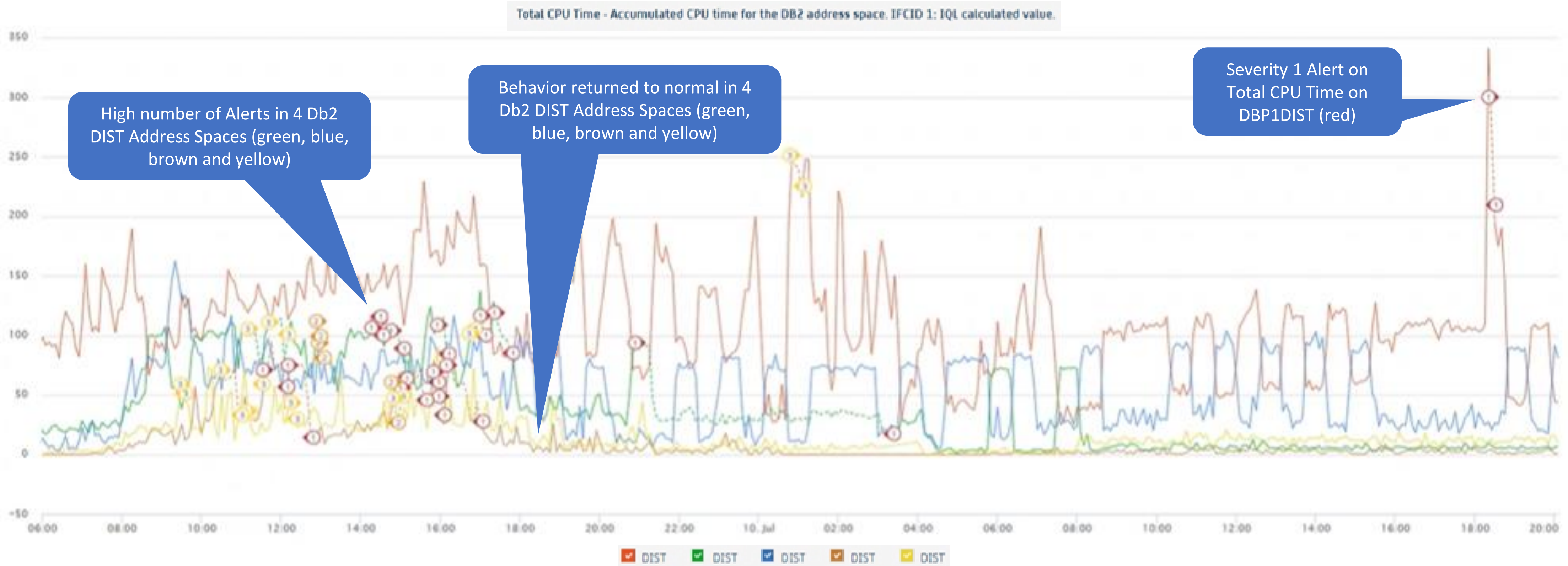
Sliding (observation) Window



HIGH CPU USAGE ON DB2 DIST ADDRESS SPACE

High CPU usage on Db2 **ssidDIST** tasks (1|5)

- 5 Production LPARs and 5 Production Db2 Subsystems
- Accumulated CPU time for the Db2 **DIST**ributed Address Space



High number of Alerts in 4 Db2 DIST Address Spaces (green, blue, brown and yellow)

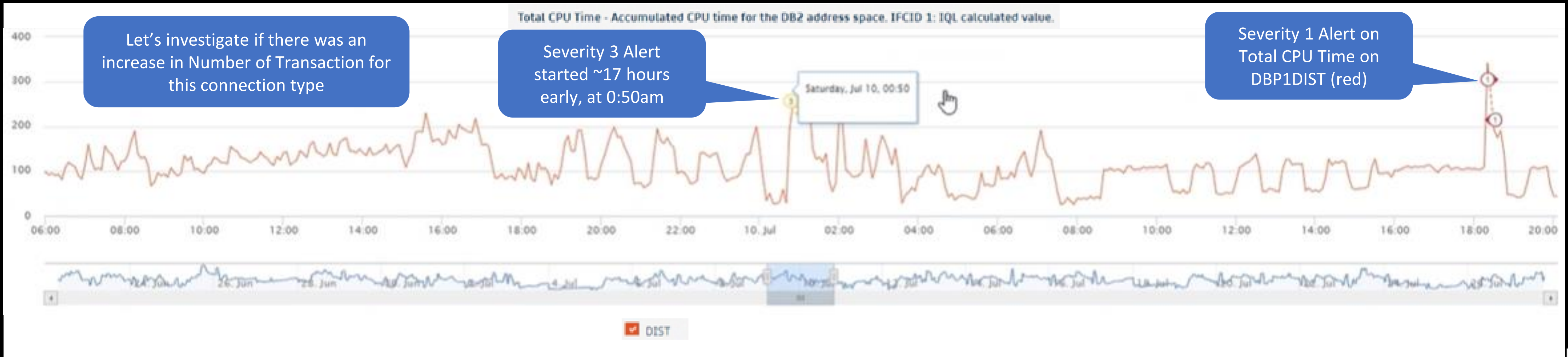
Behavior returned to normal in 4 Db2 DIST Address Spaces (green, blue, brown and yellow)

Severity 1 Alert on Total CPU Time on DBP1DIST (red)

High CPU usage on Db2 **ssidDIST** tasks (2|5)

- Zooming in a particular LPAR and Db2 Subsystem
- Accumulated CPU time for the Db2 **DIST**ributed Address Space

Severity	Metric Value	Date	Time	Elapsed Time	Samples	Metric	Metric Path	Status
1	3.078766	07/10/21	18:29	6m	6	IDBDCPUT	S1P1 > **DBP1**> DIST > IDBDCPUT	Closed
3	1.513442	07/10/21	1:06	16m	16	IDBDCPUT	S1P1 > **DBP1**> DIST > IDBDCPUT	Closed



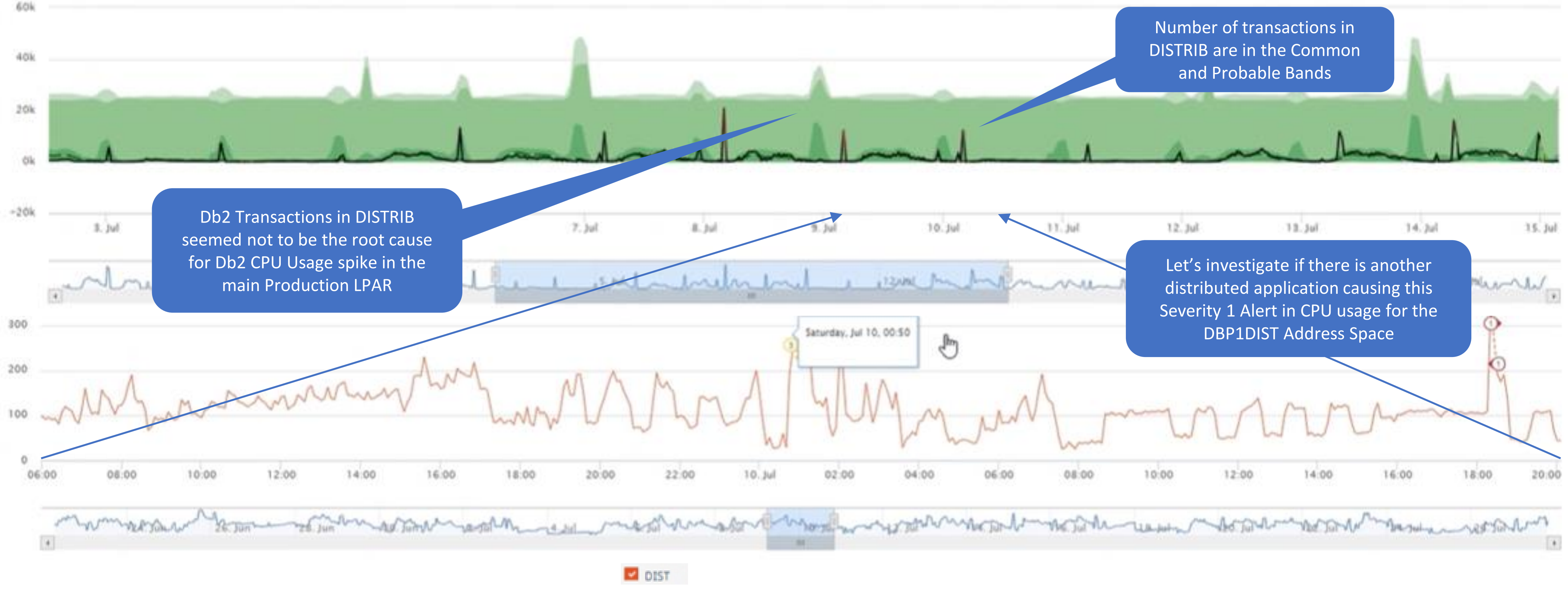
Let’s investigate if there was an increase in Number of Transaction for this connection type

Severity 3 Alert started ~17 hours early, at 0:50am

Severity 1 Alert on Total CPU Time on DBP1DIST (red)

High CPU usage on Db2 ssidDIST tasks (3|5)

- Investigate if there was an increase in Number of Transaction for this connection type



Number of transactions in DISTRIB are in the Common and Probable Bands

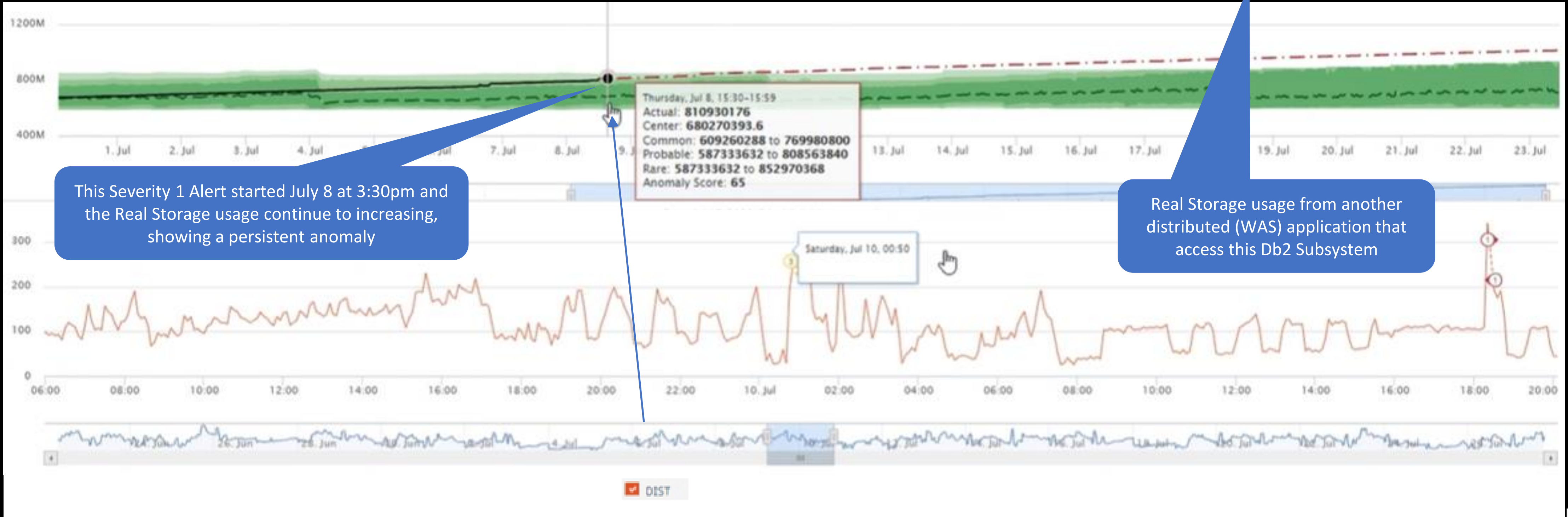
Db2 Transactions in DISTRIB seemed not to be the root cause for Db2 CPU Usage spike in the main Production LPAR

Let's investigate if there is another distributed application causing this Severity 1 Alert in CPU usage for the DBP1DIST Address Space

High CPU usage on Db2 **ssidDIST** tasks (4|5)

- Investigate if there was another distributed application causing this Severity 1 Alert

Severity	Metric Value	Date	Time	Elapsed Time	Samples	Metric	Metric Path	Status
1	13915648	07/23/21	11:02	14d 23h 0m	21541	JOBRSTG	S1P1 > **W8S161AA** > JOBRSTG	Open



Thursday, Jul 8, 15:30-15:59
Actual: **810930176**
Center: **680270393.6**
Common: **609260288** to **769980800**
Probable: **587333632** to **808563840**
Rare: **587333632** to **852970368**
Anomaly Score: **65**

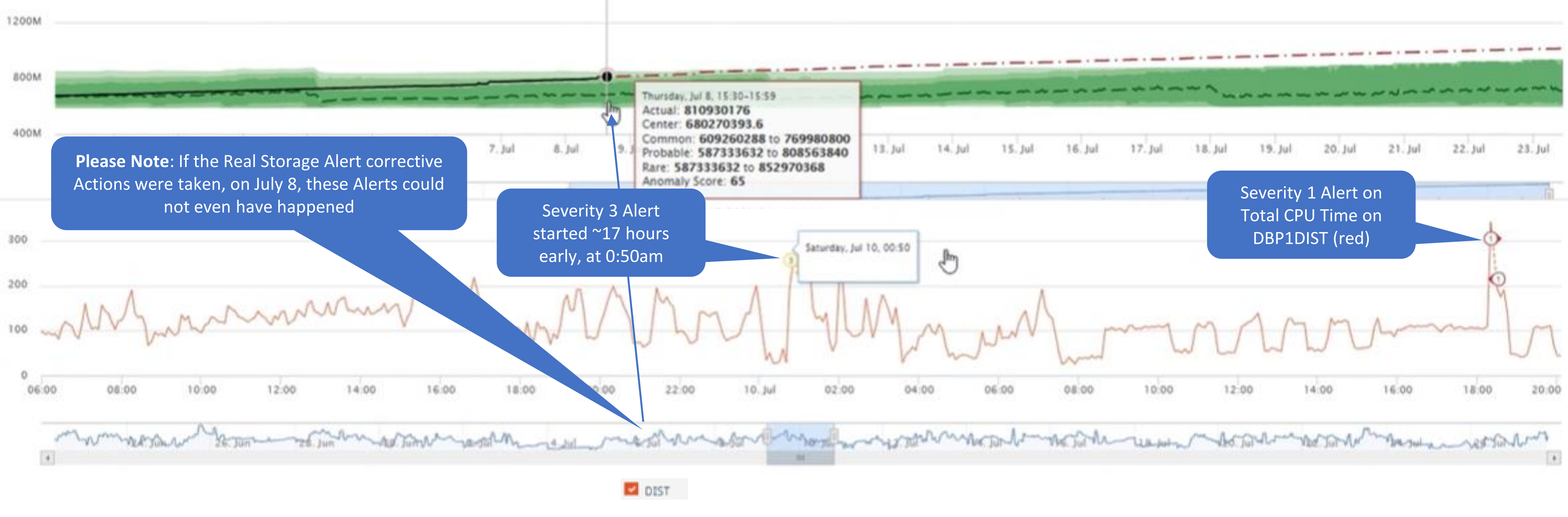
This Severity 1 Alert started July 8 at 3:30pm and the Real Storage usage continue to increasing, showing a persistent anomaly

Real Storage usage from another distributed (WAS) application that access this Db2 Subsystem

High CPU usage on Db2 **ssidDIST** tasks (5|5)

- Investigate if there was another distributed application causing this Severity 1 Alert

Severity	Metric Value	Date	Time	Elapsed Time	Samples	Metric	Metric Path	Status
1	13915648	07/23/21	11:02	14d 23h 0m	21541	JOBRSTG	S1P1 > **W8S161AA** > JOBRSTG	Open



```
Thursday, Jul 8, 15:30-15:59
Actual: 810930176
Center: 680270393.6
Common: 609260288 to 769980800
Probable: 587333632 to 808563840
Rare: 587333632 to 852970368
Anomaly Score: 65
```

```
Saturday, Jul 10, 00:50
```

Please Note: If the Real Storage Alert corrective Actions were taken, on July 8, these Alerts could not even have happened

Severity 3 Alert started ~17 hours early, at 0:50am

Severity 1 Alert on Total CPU Time on DBP1DIST (red)

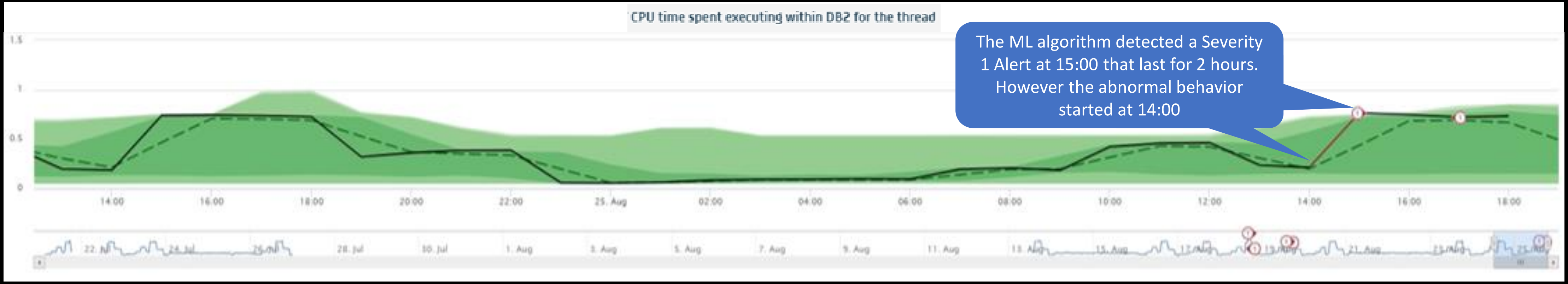
BROADCOM
MAINFRAME SOFTWARE

DB2 PACKAGE AND PLAN PERFORMANCE INVESTIGATION

Db2 Package and Plan CPU usage issue

- Analyze the Total CPU Time spent executing a particular PLAN and PACKAGE

Severity	Metric Value	Date	Time	Elapsed Time	Samples	Metric	Metric Path	Status
1	0.721046	08/25/21	17:00	2h 0m	2	DB2_CPU_PKGE	MSTRSVW > D121 > **MARBLES** > DB2_CPU_PKGE	Closed
1	0.000362	08/25/21	15:00	2h 0m	2	DB2_CPU_AVG_PKGE	MSTRSVW > D121 > **MARBLES** > DB2_CPU_AVG_PKGE	Closed


CPU time spent executing within DB2 for the thread

The ML algorithm detected a Severity 1 Alert at 15:00 that last for 2 hours. However the abnormal behavior started at 14:00

BROADCOM®
MAINFRAME SOFTWARE

Thank you

Antonio Couto

antonio.couto@broadcom.com

GitHub Repository

https://github.com/AntonioJSCouto/TRIDEX2024Q2

Antonio Couto
antonio.couto@broadcom.com
https://www.linkedin.com/in/ancouto/
@antoniojscouto