Capstone Project of Summer Analytics 2025

hosted by Consulting & Analytics Club × Pathway

This sample notebook demonstrates how to process live data streams using Pathway. The dataset used here is a subset of the one provided — specifically, it includes data for only a single parking spot. You are expected to implement your model across all parking spots.

Please note that the pricing model used in this notebook is a simple baseline. You are expected to design and implement a more advanced and effective model.

```
!pip install pathway bokeh --quiet # This cell may take a few seconds to execute.
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.4/60.4 kB 3.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.4/149.4 kB 7.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.7/69.7 MB 11.8 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.6/77.6 kB 6.7 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 777.6/777.6 kB 52.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.2/139.2 kB 12.2 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.5/26.5 MB 89.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 4.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.3/135.3 kB 13.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.6/244.6 kB 23.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 318.4/318.4 kB 28.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 985.8/985.8 kB 64.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.6/148.6 kB 14.4 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.8/139.8 kB 12.5 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 6.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.7/55.7 kB 5.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.5/118.5 kB 11.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.2/196.2 kB 19.3 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.9/434.9 kB 30.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 85.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.7/2.7 MB 77.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.3/13.3 MB 105.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.2/83.2 kB 7.4 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 95.8 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 75.8 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency con
bigframes 2.8.0 requires google-cloud-bigquery[bqstorage,pandas]>=3.31.0, but you have google-cloud-bigquery 3.29.0 which is incompatible.
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
from datetime import datetime
import pathway as pw
import bokeh.plotting
import panel as pn
```

Step 1: Importing and Preprocessing the Data

```
df = pd.read_csv('/content/dataset.csv')
df

# You can find the sample dataset here: https://drive.google.com/file/d/1D479FLjp9aO3Mg8g6Lpj9oRViWacurA6/view?usp=sharing
```

	ID	SystemCodeNumber	Capacity	Latitude	Longitude	Occupancy	VehicleType	TrafficConditionNearby	QueueLength	IsSpecialDay	LastUpdatedDate	LastUpdatedTime
0	0	BHMBCCMKT01	577	26.144536	91.736172	61	car	low	1	0	04-10-2016	07:59:00
1	1	BHMBCCMKT01	577	26.144536	91.736172	64	car	low	1	0	04-10-2016	08:25:00
2	2	BHMBCCMKT01	577	26.144536	91.736172	80	car	low	2	0	04-10-2016	08:59:00
3	3	BHMBCCMKT01	577	26.144536	91.736172	107	car	low	2	0	04-10-2016	09:32:00
4	4	BHMBCCMKT01	577	26.144536	91.736172	150	bike	low	2	0	04-10-2016	09:59:00
...	...	...	...	...	...	...	...	...	...	...	...	...
18363	18363	Shopping	1920	26.150504	91.733531	1517	truck	average	6	0	19-12-2016	14:30:00
18364	18364	Shopping	1920	26.150504	91.733531	1487	car	low	3	0	19-12-2016	15:03:00
18365	18365	Shopping	1920	26.150504	91.733531	1432	cycle	low	3	0	19-12-2016	15:29:00
18366	18366	Shopping	1920	26.150504	91.733531	1321	car	low	2	0	19-12-2016	16:03:00
18367	18367	Shopping	1920	26.150504	91.733531	1180	car	low	2	0	19-12-2016	16:30:00

18368 rows × 12 columns

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

```
# Combine the 'LastUpdatedDate' and 'LastUpdatedTime' columns into a single datetime column
df['Timestamp'] = pd.to_datetime(df['LastUpdatedDate'] + ' ' + df['LastUpdatedTime'],
                                 format='%d-%m-%Y %H:%M:%S')

# Sort the DataFrame by the new 'Timestamp' column and reset the index
df = df.sort_values('Timestamp').reset_index(drop=True)


# Save the selected columns to a CSV file for streaming or downstream processing
df[["Timestamp", "Occupancy", "Capacity"]].to_csv("parking_stream.csv", index=False)

# Note: Only three features are used here for simplicity.
# Participants are expected to incorporate additional relevant features in their models.


# Define the schema for the streaming data using Pathway
# This schema specifies the expected structure of each data row in the stream

class ParkingSchema(pw.Schema):
    Timestamp: str   # Timestamp of the observation (should ideally be in ISO format)
    Occupancy: int   # Number of occupied parking spots
    Capacity: int    # Total parking capacity at the location
```

```
# Load the data as a simulated stream using Pathway's replay_csv function
# This replays the CSV data at a controlled input rate to mimic real-time streaming
# input_rate=1000 means approximately 1000 rows per second will be ingested into the stream.

data = pw.demo.replay_csv("parking_stream.csv", schema=ParkingSchema, input_rate=1000)


# Define the datetime format to parse the 'Timestamp' column
fmt = "%Y-%m-%d %H:%M:%S"

# Add new columns to the data stream:
# - 't' contains the parsed full datetime
# - 'day' extracts the date part and resets the time to midnight (useful for day-level aggregations)
data_with_time = data.with_columns(
    t = data.Timestamp.dt.strptime(fmt),
    day = data.Timestamp.dt.strptime(fmt).dt.strftime("%Y-%m-%dT00:00:00")
)
```

Step 2: Making a simple pricing function

```
# Define a daily tumbling window over the data stream using Pathway
# This block performs temporal aggregation and computes a dynamic price for each day
import datetime

delta_window = (
    data_with_time.windowby(
        pw.this.t,  # Event time column to use for windowing (parsed datetime)
        instance=pw.this.day,  # Logical partitioning key: one instance per calendar day
        window=pw.temporal.tumbling(datetime.timedelta(days=1)),  # Fixed-size daily window
        behavior=pw.temporal.exactly_once_behavior()  # Guarantees exactly-once processing semantics
    )
    .reduce(
        t=pw.this._pw_window_end,                       # Assign the end timestamp of each window
        occ_max=pw.reducers.max(pw.this.Occupancy),     # Highest occupancy observed in the window
        occ_min=pw.reducers.min(pw.this.Occupancy),     # Lowest occupancy observed in the window
        cap=pw.reducers.max(pw.this.Capacity),          # Maximum capacity observed (typically constant per spot)
    )
    .with_columns(
        price=10 + (pw.this.occ_max - pw.this.occ_min) / pw.this.cap
    )
)
```

Step 3: Visualizing Daily Price Fluctuations with a Bokeh Plot

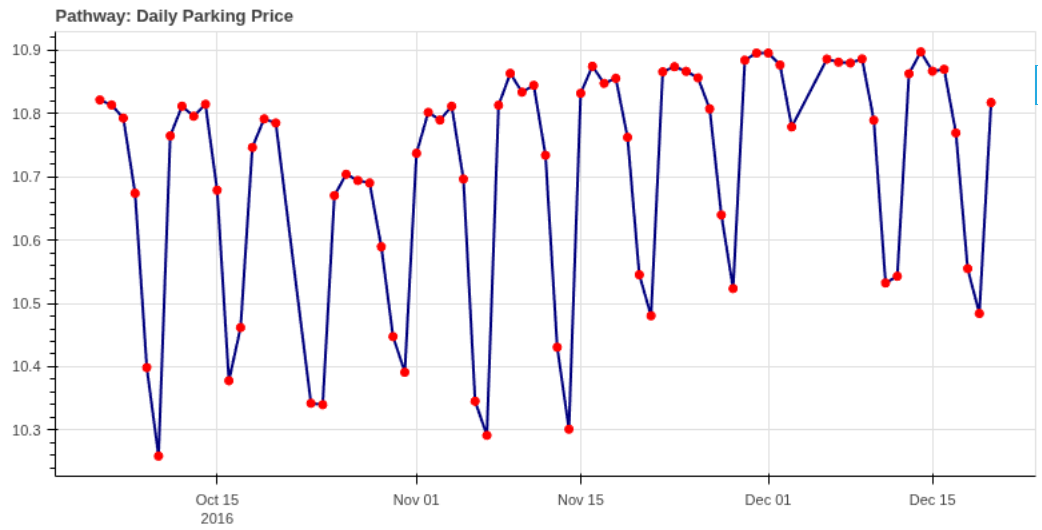
Note: The Bokeh plot in the next cell will only be generated after you run the `pw.run()` cell (i.e., the final cell).

```
# Activate the Panel extension to enable interactive visualizations
pn.extension()

# Define a custom Bokeh plotting function that takes a data source (from Pathway) and returns a figure
```

```
    def price_plotter(source):
        # Create a Bokeh figure with datetime x-axis
        fig = bokeh.plotting.figure(
            height=400,
            width=800,
            title="Pathway: Daily Parking Price",
            x_axis_type="datetime",  # Ensure time-based data is properly formatted on the x-axis
        )
        # Plot a line graph showing how the price evolves over time
        fig.line("t", "price", source=source, line_width=2, color="navy")

        # Overlay red circles at each data point for better visibility
        fig.circle("t", "price", source=source, size=6, color="red")

        return fig

    # Use Pathway's built-in .plot() method to bind the data stream (delta_window) to the Bokeh plot
    # - 'price_plotter' is the rendering function
    # - 'sorting_col="t"' ensures the data is plotted in time order
    viz = delta_window.plot(price_plotter, sorting_col="t")

    # Create a Panel layout and make it servable as a web app
    # This line enables the interactive plot to be displayed when the app is served
    pn.Column(viz).servable()
```

```
BokehDeprecationWarning: 'circle() method with size value' was deprecated in Bokeh 3.4.0 and will be removed, use 'scatter(size=...) instead' instead.
```

Streaming mode




```
    # Start the Pathway pipeline execution in the background
    # - This triggers the real-time data stream processing defined above
    # - %%capture --no-display suppresses output in the notebook interface

    %%capture --no-display
    pw.run()
```

```
WARNING:pathway_engine.connectors.monitoring:PythonReader: Closing the data source
```