```
!pip install pathway bokeh
```

```
Requirement already satisfied: google-resumable-media<3.0dev,>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from google-cloud-sig
Requirement already satisfied: grpcio<2.0.0,>=1.51.3 in /usr/local/lib/python3.11/dist-packages (from google-cloud-pubsub>=2.21.1->pat
Requirement already satisfied: proto-plus<2.0.0,>=1.22.0 in /usr/local/lib/python3.11/dist-packages (from google-cloud-pubsub>=2.21.1-
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<7.0.0,>=3.20.2 in /usr/local/lib/python3
Requirement already satisfied: grpc-google-iam-v1<1.0.0,>=0.12.4 in /usr/local/lib/python3.11/dist-packages (from google-cloud-pubsub>
Requirement already satisfied: grpcio-status>=1.33.2 in /usr/local/lib/python3.11/dist-packages (from google-cloud-pubsub>=2.21.1->pat
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from Jinja2>=2.9->bokeh) (3.0.2)
Requirement already satisfied: ipywidgets==8.* in /usr/local/lib/python3.11/dist-packages (from jupyter-bokeh>=3.0.7->pathway) (8.1.7)
Requirement already satisfied: comm>=0.1.3 in /usr/local/lib/python3.11/dist-packages (from ipywidgets==8.*->jupyter-bokeh>=3.0.7->pat
Requirement already satisfied: ipython>=6.1.0 in /usr/local/lib/python3.11/dist-packages (from ipywidgets==8.*->jupyter-bokeh>=3.0.7->
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.11/dist-packages (from ipywidgets==8.*->jupyter-bokeh>=3.0.7
Requirement already satisfied: widgetsnbextension~=4.0.14 in /usr/local/lib/python3.11/dist-packages (from ipywidgets==8.*->jupyter-bo
Requirement already satisfied: jupyterlab_widgets~=3.0.15 in /usr/local/lib/python3.11/dist-packages (from ipywidgets==8.*->jupyter-bo
Requirement already satisfied: importlib-metadata<8.8.0,>=6.0 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-api>=1.22
Requirement already satisfied: googleapis-common-protos~=1.52 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-exporter-
Requirement already satisfied: opentelemetry-exporter-otlp-proto-common==1.34.1 in /usr/local/lib/python3.11/dist-packages (from opent
Requirement already satisfied: opentelemetry-proto==1.34.1 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-exporter-otl
Requirement already satisfied: opentelemetry-semantic-conventions==0.55b1 in /usr/local/lib/python3.11/dist-packages (from opentelemet
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.1->pathway) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=2.1->pathway) (2025.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (6.2.0)
Requirement already satisfied: linkify-it-py in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (2.0.3)
Requirement already satisfied: markdown in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (3.8.2)
Requirement already satisfied: markdown-it-py in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (3.0.0)
Requirement already satisfied: mdit-py-plugins in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (0.4.2)
Requirement already satisfied: param<3.0,>=2.1.0 in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (2.2.1)
Requirement already satisfied: pyviz-comms>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (3.0.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from panel>=1.3.1->pathway) (4.67.1)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic~=2.9.0->pathway) (0.7.
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.11/dist-packages (from pydantic~=2.9.0->pathway) (2.23.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31.0->pathway) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31.0->pathway) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.31.0->pathway) (2025.6.
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=12.6.0->pathway) (2.19.2
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.0->pathway) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.0->pathway) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.0->pathway) (3.6.
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from gitdb<5,>=4.0.1->gitpython>=3.1.43->pa
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.0,>=1.
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /usr/local/lib/python3.11/dist-packages (from google-resumable-media<3.0d
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.11/dist-packages (from ht
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.11/dist-packages (from importlib-metadata<8.8.0,>=6.0->opentelemet
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py->panel>=1.3.1->pathway) (0.1
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from bleach->panel>=1.3.1->pathway) (0.5.1)
Requirement already satisfied: uc-micro-py in /usr/local/lib/python3.11/dist-packages (from linkify-it-py->panel>=1.3.1->pathway) (1.0
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jupyter-bo
Requirement already satisfied: decorator in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jupyter-bok
Requirement already satisfied: pickleshare in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jupyter-b
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>=
Requirement already satisfied: backcall in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jupyter-boke
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jup
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.11/dist-packages (from ipython>=6.1.0->ipywidgets==8.*->jupyter-b
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules>=0.2.1->google-aut
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.11/dist-packages (from jedi>=0.16->ipython>=6.1.0->ipywid
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.11/dist-packages (from pexpect>4.3->ipython>=6.1.0->ipywidget
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->
```

```
import pandas as pd
import numpy as np
import pathway as pw
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import push_notebook
from bokeh.layouts import column
import time
from IPython.display import display

output_notebook()
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files dataset.csv

- **dataset.csv**(text/csv) - 1595541 bytes, last modified: 7/6/2025 - 100% done

Saving dataset.csv to dataset.csv

```
import pandas as pd

df = pd.read_csv("dataset.csv")  # No need for /content/ path if uploaded manually
print("Shape:", df.shape)
df.head()
```

Shape: (18368, 12)

	ID	SystemCodeNumber	Capacity	Latitude	Longitude	Occupancy	VehicleType	TrafficConditionNearby	QueueLength	IsSpecialDay	LastU
0	0	BHMBCCMKT01	577	26.144536	91.736172	61	car	low	1	0	
1	1	BHMBCCMKT01	577	26.144536	91.736172	64	car	low	1	0	
2	2	BHMBCCMKT01	577	26.144536	91.736172	80	car	low	2	0	
3	3	BHMBCCMKT01	577	26.144536	91.736172	107	car	low	2	0	
4	4	BHMBCCMKT01	577	26.144536	91.736172	150	bike	low	2	0	

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

```
# Load the dataset (uploaded in the Files section)
df = pd.read_csv("/content/dataset.csv")

# Display basic structure
print("Shape:", df.shape)
df.head()
```

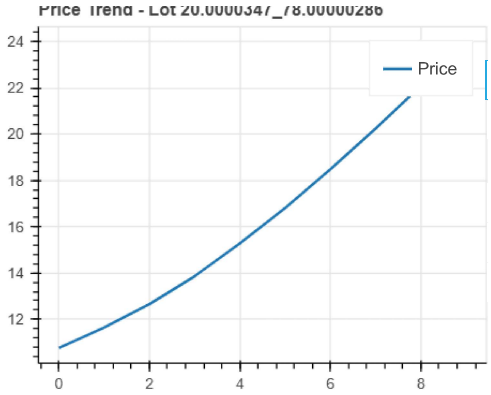
Shape: (18368, 12)

	ID	SystemCodeNumber	Capacity	Latitude	Longitude	Occupancy	VehicleType	TrafficConditionNearby	QueueLength	IsSpecialDay	LastU
0	0	BHMBCCMKT01	577	26.144536	91.736172	61	car	low	1	0	
1	1	BHMBCCMKT01	577	26.144536	91.736172	64	car	low	1	0	
2	2	BHMBCCMKT01	577	26.144536	91.736172	80	car	low	2	0	
3	3	BHMBCCMKT01	577	26.144536	91.736172	107	car	low	2	0	
4	4	BHMBCCMKT01	577	26.144536	91.736172	150	bike	low	2	0	

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

```
# Add base price for each lot
df['Price'] = 10.0  # Base price

# Get list of all unique parking lot IDs (or use Lat-Long as unique ID)
df['LotID'] = df['Latitude'].astype(str) + "_" + df['Longitude'].astype(str)

# Sort by time (assuming there's a timestamp column or simulated time index)
df['TimeIndex'] = np.tile(np.arange(0, df.shape[0] // 14), 14)
df = df.sort_values(by=['TimeIndex', 'LotID']).reset_index(drop=True)



def linear_price_update(prev_price, occupancy, capacity, alpha=2):
    usage_ratio = occupancy / capacity if capacity > 0 else 0
    return prev_price + alpha * usage_ratio



import time
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import push_notebook
from bokeh.layouts import column

output_notebook()

# Initialize storage
lot_prices = {}
price_history = {lot: [] for lot in df['LotID'].unique()}
```

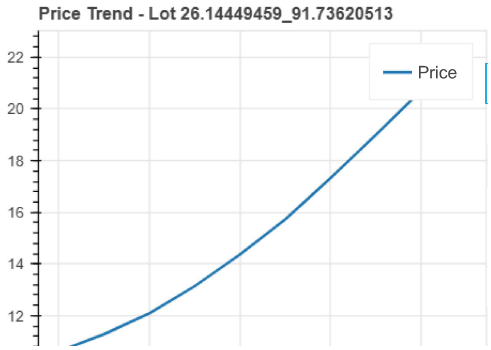
```
# Setup Bokeh plots
plots = []
for lot in df['LotID'].unique():
    p = figure(title=f"Price Trend - Lot {lot}", width=400, height=300)
    p.line([], [], line_width=2, legend_label="Price", name="price_line")
    plots.append(p)

layout = column(*plots)
handle = show(layout, notebook_handle=True)

# Set number of time steps to simulate
max_steps = 10  # change this to more if needed

# Linear Pricing Function
def linear_price_update(prev_price, occupancy, capacity, alpha=2):
    usage_ratio = occupancy / capacity if capacity > 0 else 0
    return prev_price + alpha * usage_ratio

# Run real-time simulation
for t in range(min(df['TimeIndex'].max() + 1, max_steps)):
    current_time_slice = df[df['TimeIndex'] == t]

    for idx, row in current_time_slice.iterrows():
        lot = row['LotID']
        prev_price = lot_prices.get(lot, 10.0)
        new_price = linear_price_update(prev_price, row['Occupancy'], row['Capacity'])
        lot_prices[lot] = new_price
        price_history[lot].append(new_price)

    # Update plots
    for i, lot in enumerate(df['LotID'].unique()):
        line = plots[i].select(name="price_line")[0]
        line.data_source.data = {
            'x': list(range(len(price_history[lot]))),
            'y': price_history[lot]
        }

    push_notebook(handle=handle)
    time.sleep(0.05)  # fast visualization delay
```


Price Trend - Lot 20.0000347_78.00000286
Price
24
22
20
18
16
14
12
0
2
4
6
8

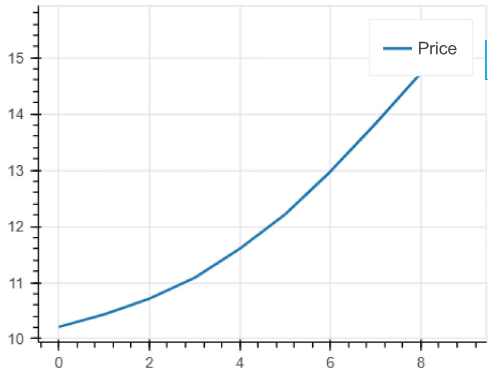

Price Trend - Lot 26.13795775_91.74099445
Price
26
24
22
20
18
16
14
12
10
0
2
4
6
8


Price Trend - Lot 26.14001386_91.73099967
Price
20
18
16
14
12
10
0
2
4
6
8

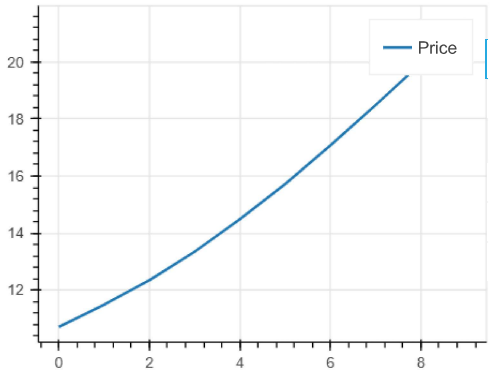

Price Trend - Lot 26.14004753_91.73097233
Price
24
22
20
18
16
14
12
0
2
4
6
8


Price Trend - Lot 26.14449459_91.73620513
Price
22
20
18
16
14
12

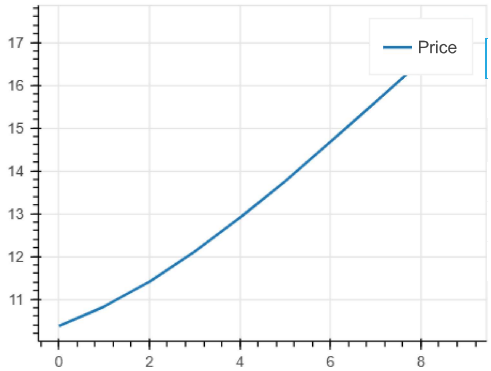
Price Trend - Lot 26.14453614_91.73617216



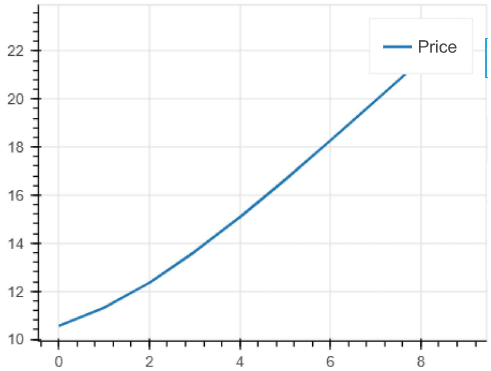
Price Trend - Lot 26.14747299_91.72804914



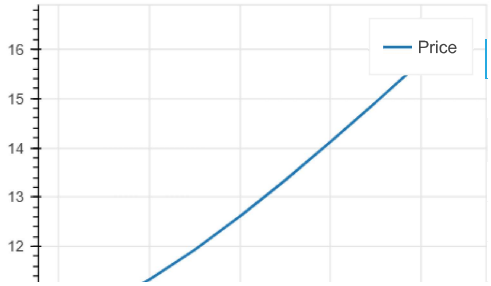
Price Trend - Lot 26.14749053_91.72799688

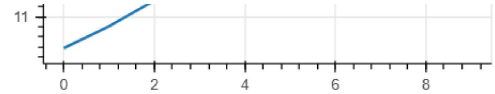


Price Trend - Lot 26.14749943_91.72800489

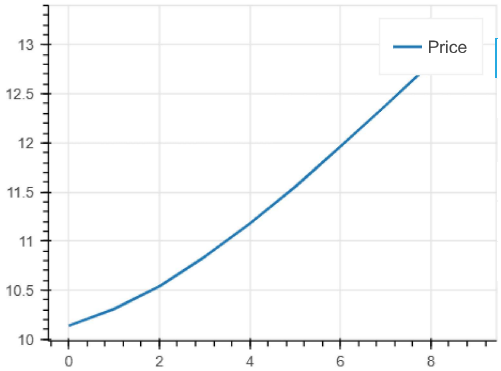


Price Trend - Lot 26.14749998_91.72797778

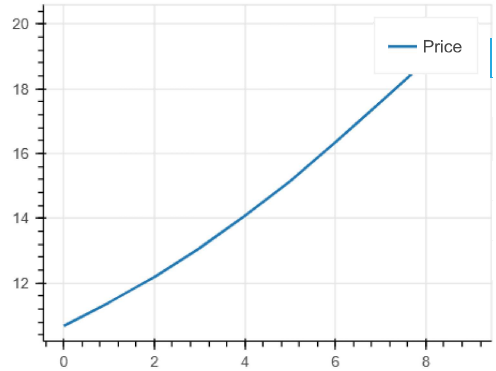




Price Trend - Lot 26.14754061_91.72797041



Price Trend - Lot 26.14754886_91.72799519



Price Trend - Lot 26.14901995_91.7395035

Price Trend - Lot 26.15050395_91.73353109

```
# Vehicle weight mapping
vehicle_weights = {
    'car': 1.0,
    'bike': 0.5,
    'truck': 1.5
}

# Hyperparameters (tune these later)
alpha = 2.0
beta = 0.1
gamma = 0.3
delta = 1.0
epsilon = 0.5
lambda_ = 0.5  # price sensitivity to demand

# Base price
BASE_PRICE = 10.0



def demand_based_price(row, prev_price):
    # Extract features
    occ_ratio = row['Occupancy'] / row['Capacity'] if row['Capacity'] > 0 else 0
    queue = row['QueueLength']
    traffic = row['Traffic']
    is_special = row['IsSpecialDay']
    vehicle_type = row['VehicleType'].lower()

    # Assign vehicle type weight
    vehicle_weight = vehicle_weights.get(vehicle_type, 1.0)

    # Raw demand
    demand = (alpha * occ_ratio +
              beta * queue -
              gamma * traffic +
              delta * is_special +
              epsilon * vehicle_weight)

    # Normalize demand using sigmoid (to keep within [0,1])
    norm_demand = 1 / (1 + np.exp(-demand))

    # Price update formula
    new_price = BASE_PRICE * (1 + lambda_ * norm_demand)

    # Bound price between $5 and $20
    return max(5, min(20, new_price))



print(df.columns.tolist())
```

```
['ID', 'SystemCodeNumber', 'Capacity', 'Latitude', 'Longitude', 'Occupancy', 'VehicleType', 'TrafficConditionNearby', 'QueueLength', 'Is
```

```
def demand_based_price(row, prev_price):
    # Extract features
    occ_ratio = row['Occupancy'] / row['Capacity'] if row['Capacity'] > 0 else 0
    queue = row['QueueLength']
    traffic = row['TrafficConditionNearby']  # ✅ corrected
    is_special = row['IsSpecialDay']
    vehicle_type = row['VehicleType'].lower()

    # Assign vehicle type weight
    vehicle_weight = vehicle_weights.get(vehicle_type, 1.0)

    # Raw demand score
    demand = (alpha * occ_ratio +
              beta * queue -
              gamma * traffic +
              delta * is_special +
              epsilon * vehicle_weight)

    # Normalize demand (sigmoid)
    norm_demand = 1 / (1 + np.exp(-demand))

    # Compute price
```

```
    new_price = BASE_PRICE * (1 + lambda_ * norm_demand)

    # Bound the price
    return max(5, min(20, new_price))



# STEP 1: Data Cleaning (do this once after loading the dataset)
df['TrafficConditionNearby'] = pd.to_numeric(df['TrafficConditionNearby'], errors='coerce').fillna(0)
df['QueueLength'] = pd.to_numeric(df['QueueLength'], errors='coerce').fillna(0)
df['IsSpecialDay'] = pd.to_numeric(df['IsSpecialDay'], errors='coerce').fillna(0)
df['Occupancy'] = pd.to_numeric(df['Occupancy'], errors='coerce').fillna(0)
df['Capacity'] = pd.to_numeric(df['Capacity'], errors='coerce').fillna(1)  # Avoid division by zero

# STEP 2: Parameters & Vehicle Mapping
vehicle_weights = {
    'car': 1.0,
    'bike': 0.5,
    'truck': 1.5
}

alpha = 2.0
beta = 0.1
gamma = 0.3
delta = 1.0
epsilon = 0.5
lambda_ = 0.5
BASE_PRICE = 10.0

# STEP 3: Demand-based price update function
def demand_based_price(row, prev_price):
    occ_ratio = row['Occupancy'] / row['Capacity'] if row['Capacity'] > 0 else 0
    queue = row['QueueLength']
    traffic = row['TrafficConditionNearby']
    is_special = row['IsSpecialDay']
    vehicle_type = row['VehicleType'].lower()
    vehicle_weight = vehicle_weights.get(vehicle_type, 1.0)

    # Demand calculation
    demand = (alpha * occ_ratio +
              beta * queue -
              gamma * traffic +
              delta * is_special +
              epsilon * vehicle_weight)

    # Normalize demand
    norm_demand = 1 / (1 + np.exp(-demand))  # Sigmoid

    # Calculate price
    new_price = BASE_PRICE * (1 + lambda_ * norm_demand)

    # Clip to bounds
    return max(5, min(20, new_price))



import time
from bokeh.plotting import figure, show, output_notebook
from bokeh.io import push_notebook
from bokeh.layouts import column

output_notebook()

# Reset storage
lot_prices = {}
price_history = {lot: [] for lot in df['LotID'].unique()}

# Setup Bokeh plots
plots = []
renderers = []

for lot in df['LotID'].unique():
    p = figure(title=f"Model 2: Price Trend – Lot {lot}",
               width=400, height=250,
               x_axis_label='Time Step',
               y_axis_label='Price ($)')
    r = p.line([], [], line_width=2, legend_label="Price", name="price_line")
    plots.append(p)
```

```
        renderers.append(r)

    layout = column(*plots)
    handle = show(layout, notebook_handle=True)

    # Simulate up to 10 steps (change as needed)
    max_steps = 10

    for t in range(min(df['TimeIndex'].max() + 1, max_steps)):
        current_time_slice = df[df['TimeIndex'] == t]

        for idx, row in current_time_slice.iterrows():
            lot = row['LotID']
            prev_price = lot_prices.get(lot, BASE_PRICE)
            new_price = demand_based_price(row, prev_price)
            lot_prices[lot] = new_price
            price_history[lot].append(new_price)

        # Update plots
        for i, lot in enumerate(df['LotID'].unique()):
            r = renderers[i]
            r.data_source.data = {
                'x': list(range(len(price_history[lot]))),
                'y': price_history[lot]
            }

        push_notebook(handle=handle)
        time.sleep(0.05)
```