Capstone Project Final

```
!pip install pathway panel bokeh pandas numpy --quiet
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.4/60.4 kB 3.8 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 149.4/149.4 kB 6.8 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69.7/69.7 MB 9.9 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.6/77.6 kB 5.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 777.6/777.6 kB 39.9 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.2/139.2 kB 9.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.5/26.5 MB 24.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 2.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.3/135.3 kB 9.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244.6/244.6 kB 15.1 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 319.1/319.1 kB 18.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 985.8/985.8 kB 38.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 148.6/148.6 kB 9.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.8/139.8 kB 8.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 4.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.7/55.7 kB 3.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.5/118.5 kB 7.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.2/196.2 kB 11.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.9/434.9 kB 25.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 58.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.7/2.7 MB 28.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.3/13.3 MB 51.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.2/83.2 kB 4.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.2/2.2 MB 53.7 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 53.5 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 sou
bigframes 2.8.0 requires google-cloud-bigquery[bqstorage,pandas]>=3.31.0, but you have google-cloud-bigquery 3.29.0 which is incompa
```

```
import numpy as np
import pandas as pd
from datetime import datetime
import pathway as pw
import panel as pn
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource
pn.extension()
```

```
from google.colab import files
df = pd.read_csv("dataset.csv")
df['Timestamp'] = pd.to_datetime(df['LastUpdatedDate'] + ' ' + df['LastUpdatedTime'],
                                 format='%d-%m-%Y %H:%M:%S').astype(str)

# Combining and convert timestamp
df['Timestamp'] = pd.to_datetime(df['LastUpdatedDate'] + ' ' + df['LastUpdatedTime'], errors='coerce')

# Drop rows with invalid timestamps
df = df[df['Timestamp'].notna()]

# Convert to string for Pathway
df['Timestamp'] = df['Timestamp'].astype(str)

# Map categorical fields
traffic_map = {"low": 0, "medium": 1, "high": 2}
vehicle_weight_map = {"car": 1.0, "bike": 0.5, "truck": 1.5}
df['TrafficLevel'] = df['TrafficConditionNearby'].map(traffic_map)
df['VehicleWeight'] = df['VehicleType'].map(vehicle_weight_map)

# Clean all numeric fields and cast to string (Pathway-safe)
numeric_columns = ['Capacity', 'Occupancy', 'QueueLength', 'IsSpecialDay', 'TrafficLevel']
for col in numeric_columns:
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)

df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce').fillna(0).astype(float).astype(str)
df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce').fillna(0).astype(float).astype(str)
df['VehicleWeight'] = df['VehicleWeight'].fillna(1.0).astype(float).astype(str)

# Avoid divide-by-zero in Capacity
df['Capacity'] = df['Capacity'].replace("0", "1")

# Compute OccupancyRate
df['OccupancyRate'] = (
    df['Occupancy'].astype(float) / df['Capacity'].astype(float)
```

```
).round(3).astype(str)

# Final selection
df = df[['Timestamp', 'SystemCodeNumber', 'Latitude', 'Longitude', 'Capacity', 'Occupancy',
         'QueueLength', 'TrafficLevel', 'VehicleWeight', 'IsSpecialDay', 'OccupancyRate']]

# Replace any remaining NaNs with "0"
df = df.fillna("0")

# Save cleaned CSV
df.to_csv("parking_stream_final.csv", index=False)

# Confirm saved
print("✅ Saved cleaned file as parking_stream_final.csv")
df.head()
```

```
/tmp/ipython-input-5-2733263730.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['Timestamp'] = df['Timestamp'].astype(str)
/tmp/ipython-input-5-2733263730.py:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['TrafficLevel'] = df['TrafficConditionNearby'].map(traffic_map)
/tmp/ipython-input-5-2733263730.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['VehicleWeight'] = df['VehicleType'].map(vehicle_weight_map)
/tmp/ipython-input-5-2733263730.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)
/tmp/ipython-input-5-2733263730.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)
/tmp/ipython-input-5-2733263730.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)
/tmp/ipython-input-5-2733263730.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)
/tmp/ipython-input-5-2733263730.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0).astype(int).astype(str)
✅ Saved cleaned file as parking_stream_final.csv
```

	Timestamp	SystemCodeNumber	Latitude	Longitude	Capacity	Occupancy	QueueLength	TrafficLevel	VehicleWeight	IsSpecialDay
0	2016-04-10 07:59:00	BHMBCCMKT01	26.14453614	91.73617216	577	61	1	0	1.0	0
1	2016-04-10 08:25:00	BHMBCCMKT01	26.14453614	91.73617216	577	64	1	0	1.0	0
	2016-04-									

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

```
print(df.dtypes)
```

```
Timestamp           object
SystemCodeNumber    object
Latitude            object
Longitude           object
Capacity            object
```

```
Occupancy           object
QueueLength         object
TrafficLevel        object
VehicleWeight       object
IsSpecialDay        object
OccupancyRate       object
dtype: object
```

```
class RawSchema(pw.Schema):
    Timestamp: str
    SystemCodeNumber: str
    Latitude: str
    Longitude: str
    Capacity: str
    Occupancy: str
    QueueLength: str
    TrafficLevel: str
    VehicleWeight: str
    IsSpecialDay: str
    OccupancyRate: str
```

```
from datetime import datetime
import pathway as pw

@pw.udf
def parse_timestamp(ts: str) -> str:
    try:
        return datetime.fromisoformat(ts).isoformat()
    except:
        return "1970-01-01T00:00:00"
```

```
raw = pw.demo.replay_csv("parking_stream_final.csv", schema=RawSchema, input_rate=500)

data = raw.with_columns(
    Timestamp=parse_timestamp(pw.this.Timestamp),
    Latitude=pw.apply(float, pw.this.Latitude),
    Longitude=pw.apply(float, pw.this.Longitude),
    Capacity=pw.apply(float, pw.this.Capacity),
    Occupancy=pw.apply(float, pw.this.Occupancy),
    QueueLength=pw.apply(float, pw.this.QueueLength),
    TrafficLevel=pw.apply(float, pw.this.TrafficLevel),
    VehicleWeight=pw.apply(float, pw.this.VehicleWeight),
    IsSpecialDay=pw.apply(float, pw.this.IsSpecialDay),
    OccupancyRate=pw.apply(float, pw.this.OccupancyRate)
)
```

```
@pw.udf
def model_1(prev_price: float, occ_rate: float) -> float:
    return round(prev_price + 0.05 * occ_rate, 2)

@pw.udf
def model_2(base_price: float, occ_rate: float, queue: float, traffic: float, special: float, weight: float) -> float:
    demand = (0.5 * occ_rate + 0.3 * queue - 0.2 * traffic + 0.4 * special + 1.0 * weight)
    norm_demand = min(max(demand / 10, -1), 1)
    return round(min(max(base_price * (1 + 0.2 * norm_demand), 5), 20), 2)

@pw.udf
def model_3(base_price: float, price: float, occ: float, cap: float, lat: float, lon: float) -> float:
    if occ >= cap:
        return max(price - 1, 5)
    else:
        return min(price + 1, 20)
```

```
base_price = 10.0

table = data.with_columns(
    Model1=model_1(base_price, pw.this.OccupancyRate),
    Model2=model_2(base_price, pw.this.OccupancyRate, pw.this.QueueLength, pw.this.TrafficLevel,
                   pw.this.IsSpecialDay, pw.this.VehicleWeight),
).with_columns(
    Model3=model_3(base_price, pw.this.Model2, pw.this.Occupancy, pw.this.Capacity,
                   pw.this.Latitude, pw.this.Longitude)
)
```

```
pw.io.csv.write(
    table.select(
        Timestamp=pw.this.Timestamp,
        Lot=pw.this.SystemCodeNumber,
        Model1=pw.this.Model1,
        Model2=pw.this.Model2,
        Model3=pw.this.Model3
    ),
    filename="output_stream.csv"
)
```

```
/usr/local/lib/python3.11/dist-packages/beartype/_util/hint/pep/utilpeptest.py:311: BeartypeDecorHintPep585DeprecationWarning: PEP 4
    https://beartype.readthedocs.io/en/latest/api_roar/#pep-585-deprecations
  warn(
```

```
# Run the Pathway pipeline so it processes and writes the stream
pw.run()
```

```
WARNING:pathway.engine.connectors.monitoring:PythonReader: Closing the data source
```

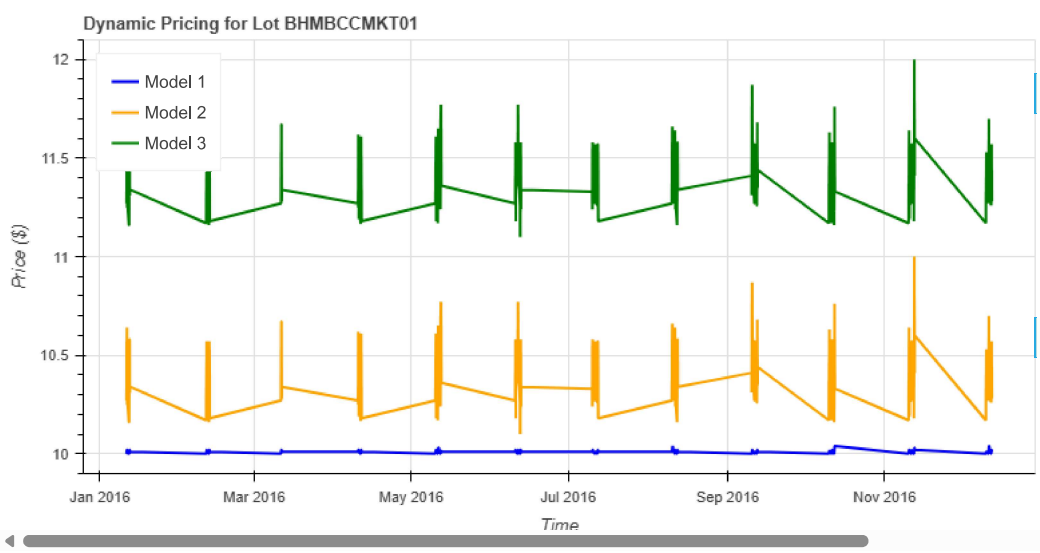
```
import time
time.sleep(5)  # Wait for a few seconds for the file to populate

# Read the CSV output stream into pandas
output_df = pd.read_csv("output_stream.csv")
output_df['Timestamp'] = pd.to_datetime(output_df['Timestamp'])
output_df.head()
```

	Timestamp	Lot	Model1	Model2	Model3	time	diff
0	2016-04-10 08:25:00	BHMBCCMKT01	10.01	10.27	11.27	1752083145698	1
1	2016-04-10 09:32:00	BHMBCCMKT01	10.01	10.34	11.34	1752083145698	1
2	2016-04-10 10:26:00	BHMBCCMKT01	10.02	10.41	11.41	1752083145698	1
3	2016-04-10 08:59:00	BHMBCCMKT01	10.01	10.33	11.33	1752083145698	1
4	2016-04-10 07:59:00	BHMBCCMKT01	10.01	10.27	11.27	1752083145698	1

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

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
import pandas as pd

output_notebook()

# Choose a parking lot
selected_lot = output_df['Lot'].iloc[0]  # For  specific ID

# Filter and sort by time
lot_df = output_df[output_df['Lot'] == selected_lot].sort_values('Timestamp')
lot_df['Timestamp'] = pd.to_datetime(lot_df['Timestamp'])

# Setup source
source = ColumnDataSource(lot_df)

# Create figure
p = figure(title=f"Dynamic Pricing for Lot {selected_lot}",
           x_axis_type="datetime", width=800, height=400)

# Plot Model 1, 2, 3
p.line(x='Timestamp', y='Model1', source=source, line_width=2, color='blue', legend_label='Model 1')
p.line(x='Timestamp', y='Model2', source=source, line_width=2, color='orange', legend_label='Model 2')
p.line(x='Timestamp', y='Model3', source=source, line_width=2, color='green', legend_label='Model 3')

# Hover tool
hover = HoverTool(tooltips=[
    ("Time", "@Timestamp{%F %T}"),
    ("Model 1", "@Model1"),
    ("Model 2", "@Model2"),
    ("Model 3", "@Model3")
], formatters={'@Timestamp': 'datetime'})
p.add_tools(hover)
```

```
# Styling
p.xaxis.axis_label = "Time"
p.yaxis.axis_label = "Price ($)"
p.legend.location = "top_left"

show(p)
```



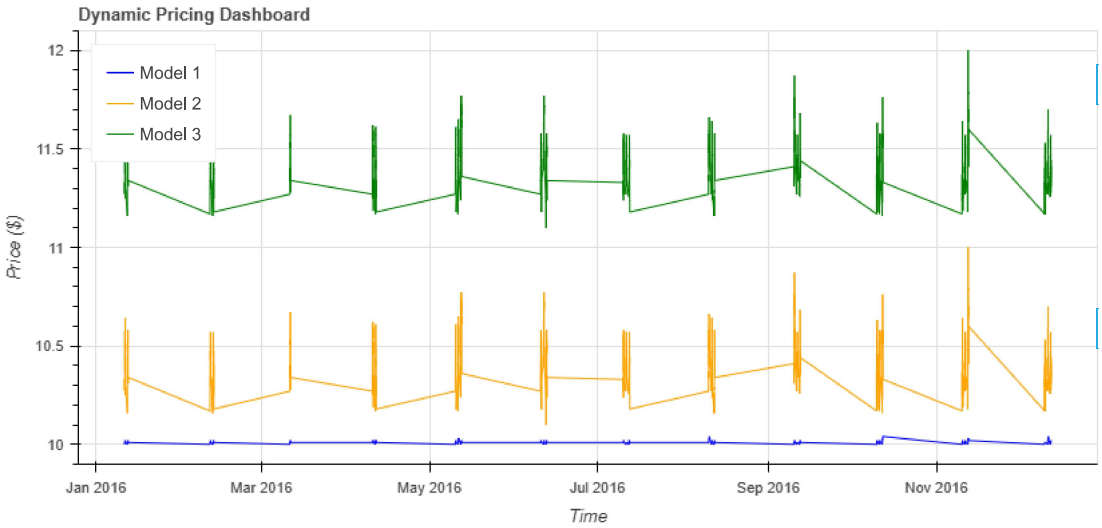

```
import panel as pn
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource, HoverTool

pn.extension('bokeh')

# Create dropdown with all Lot IDs
lot_selector = pn.widgets.Select(name='Parking Lot', options=list(output_df['Lot'].unique()))

# Setup empty Bokeh plot and data source
plot = figure(title="Dynamic Pricing Dashboard", x_axis_type="datetime", width=850, height=400)

source = ColumnDataSource(data=dict(Timestamp=[], Model1=[], Model2=[], Model3=[]))

# Add all 3 model lines
plot.line(x='Timestamp', y='Model1', source=source, color='blue', legend_label='Model 1')
plot.line(x='Timestamp', y='Model2', source=source, color='orange', legend_label='Model 2')
plot.line(x='Timestamp', y='Model3', source=source, color='green', legend_label='Model 3')

# Hover tool
hover = HoverTool(tooltips=[
    ("Time", "@Timestamp{%F %T}"),
    ("Model 1", "@Model1"),
    ("Model 2", "@Model2"),
    ("Model 3", "@Model3")
], formatters={'@Timestamp': 'datetime'})
plot.add_tools(hover)

plot.xaxis.axis_label = "Time"
plot.yaxis.axis_label = "Price ($)"
plot.legend.location = "top_left"

# Update plot based on selected lot
@pn.depends(lot_selector)
def update_selected_lot(lot):
    df = output_df[output_df['Lot'] == lot].sort_values("Timestamp").copy()
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Update the Bokeh data source
    source.data = {
        'Timestamp': df['Timestamp'],
        'Model1': df['Model1'],
        'Model2': df['Model2'],
        'Model3': df['Model3'],
    }
    return plot

# Combine into Panel layout
dashboard = pn.Column("### Dynamic Pricing Dashboard (Model 1, 2, 3)", lot_selector, update_selected_lot)

# Show in notebook (or use .show() if in Colab)
dashboard.servable()
```

```
WARNING:param.panel_extension: bokeh extension not recognized and will be skipped.
WARNING:param.panel_extension:bokeh extension not recognized and will be skipped.
```

Dynamic Pricing Dashboard (Model 1, 2, 3)

Parking Lot

BHMBCCMKT01




```
WARNING:param.Column00128: Comm received message that could not be deserialized.
WARNING:param.Column00128:Comm received message that could not be deserialized.
WARNING:param.Column00128: Comm received message that could not be deserialized.
WARNING:param.Column00128:Comm received message that could not be deserialized.
WARNING:param.Column00128: Comm received message that could not be deserialized.
WARNING:param.Column00128:Comm received message that could not be deserialized.
WARNING:param.Column00128: Comm received message that could not be deserialized.
WARNING:param.Column00128:Comm received message that could not be deserialized.
WARNING:param.Column00128: Comm received message that could not be deserialized.
WARNING:param.Column00128:Comm received message that could not be deserialized.
```