```
Start coding or generate with AI.
```

```
import pandas as pd
from google.colab import files
import io

uploaded = files.upload()
file_name = list(uploaded.keys())[0]
df = pd.read_csv(io.BytesIO(uploaded[file_name]))

df.rename(columns={'SystemCodeNumber': 'Location_ID', 'VehicleType': 'Vehicle_Type',
                   'TrafficConditionsNearby': 'Traffic', 'LastUpdatedDate': 'Date',
                   'LastUpdatedTime': 'Time'}, inplace=True)

print("Columns:", df.columns.tolist())
print("Unique Parking Lots:", df['Location_ID'].nunique())
print("Vehicle Types:", df['Vehicle_Type'].unique())
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving dataset.csv to dataset (3).csv
Columns: ['ID', 'Location_ID', 'Capacity', 'Latitude', 'Longitude', 'Occupancy', 'Vehicle_Type', 'TrafficConditionNearby', 'QueueLer
Unique Parking Lots: 14
```

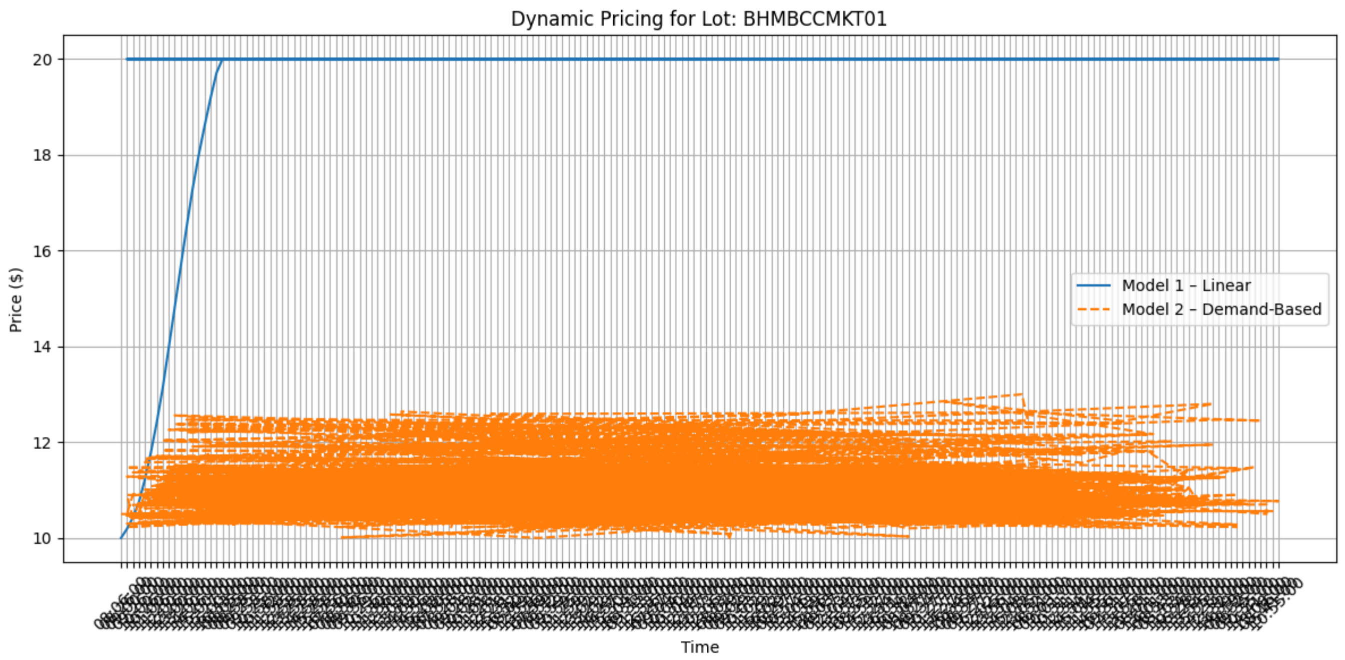
```
base_price = 10
alpha = 2

linear_prices = [base_price]

for i in range(1, len(lot_df)):
    occ = lot_df.iloc[i]['Occupancy']
    cap = lot_df.iloc[i]['Capacity']
    prev_price = linear_prices[-1]
    new_price = prev_price + alpha * (occ / cap)
    new_price = max(5, min(20, new_price))
    linear_prices.append(round(new_price, 2))

lot_df['Linear_Price'] = linear_prices
```

```
alpha = 1
beta = 0.5
gamma = 0.7
delta = 2
epsilon = {'car': 1, 'bike': 0.5, 'truck': 1.5, 'cycle': 0.75}
lamda = 0.3

lot_df['Vehicle_Weight'] = lot_df['Vehicle_Type'].map(epsilon)

traffic_map = {'low': 0, 'medium': 1, 'high': 2, 'average': 1}
lot_df['Traffic_Level'] = lot_df['TrafficConditionNearby'].map(traffic_map)

lot_df['Raw_Demand'] = (
    alpha * (lot_df['Occupancy'] / lot_df['Capacity']) +
    beta * lot_df['QueueLength'] -
    gamma * lot_df['Traffic_Level'] +
    delta * lot_df['IsSpecialDay'] +
    lot_df['Vehicle_Weight']
)

min_d = lot_df['Raw_Demand'].min()
max_d = lot_df['Raw_Demand'].max()
lot_df['Norm_Demand'] = (lot_df['Raw_Demand'] - min_d) / (max_d - min_d)

lot_df['Demand_Price'] = base_price * (1 + lamda * lot_df['Norm_Demand'])
lot_df['Demand_Price'] = lot_df['Demand_Price'].round(2)
```

```
min_d = lot_df['Raw_Demand'].min()
max_d = lot_df['Raw_Demand'].max()
lot_df['Norm_Demand'] = (lot_df['Raw_Demand'] - min_d) / (max_d - min_d)

lot_df['Demand_Price'] = base_price * (1 + lamda * lot_df['Norm_Demand'])
lot_df['Demand_Price'] = lot_df['Demand_Price'].clip(lower=5, upper=20).round(2)
```

```
import matplotlib.pyplot as plt

# Define lot_id
```

```
lot_id = lot_df['Location_ID'].iloc[0]

plt.figure(figsize=(12, 6))
plt.plot(lot_df['Time'], lot_df['Linear_Price'], label='Model 1 – Linear')
plt.plot(lot_df['Time'], lot_df['Demand_Price'], label='Model 2 – Demand-Based', linestyle='--')
plt.xticks(rotation=45)
plt.xlabel("Time")
plt.ylabel("Price ($)")
plt.title(f"Dynamic Pricing for Lot: {lot_id}")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



Start coding or generate with AI.

```
!pip install bokeh
```

```
Requirement already satisfied: bokeh in /usr/local/lib/python3.11/dist-packages (3.7.3)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.11/dist-packages (from bokeh) (3.1.6)
Requirement already satisfied: contourpy>=1.2 in /usr/local/lib/python3.11/dist-packages (from bokeh) (1.3.2)
Requirement already satisfied: narwhals>=1.13 in /usr/local/lib/python3.11/dist-packages (from bokeh) (1.45.0)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.11/dist-packages (from bokeh) (2.0.2)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.11/dist-packages (from bokeh) (24.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from bokeh) (2.2.2)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.11/dist-packages (from bokeh) (11.2.1)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.11/dist-packages (from bokeh) (6.0.2)
Requirement already satisfied: tornado>=6.2 in /usr/local/lib/python3.11/dist-packages (from bokeh) (6.4.2)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.11/dist-packages (from bokeh) (2025.4.0)
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: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->bokeh) (2.9.0.pc
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->bokeh) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->bokeh) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.2->bokeh
```

```
import pandas as pd
from google.colab import files
import io

uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded['dataset.csv']))
df.head()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving dataset.csv to dataset.csv
```

	ID	SystemCodeNumber	Capacity	Latitude	Longitude	Occupancy	VehicleType	TrafficConditionNearby	QueueLength	IsSpecialDay	L
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	

```
df.rename(columns={
    'SystemCodeNumber': 'Location_ID',
    'VehicleType': 'Vehicle_Type',
    'TrafficConditionsNearby': 'Traffic',
    'LastUpdatedDate': 'Date',
    'LastUpdatedTime': 'Time'
}, inplace=True)
```

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource
from bokeh.io import push_notebook
import time

output_notebook()
```

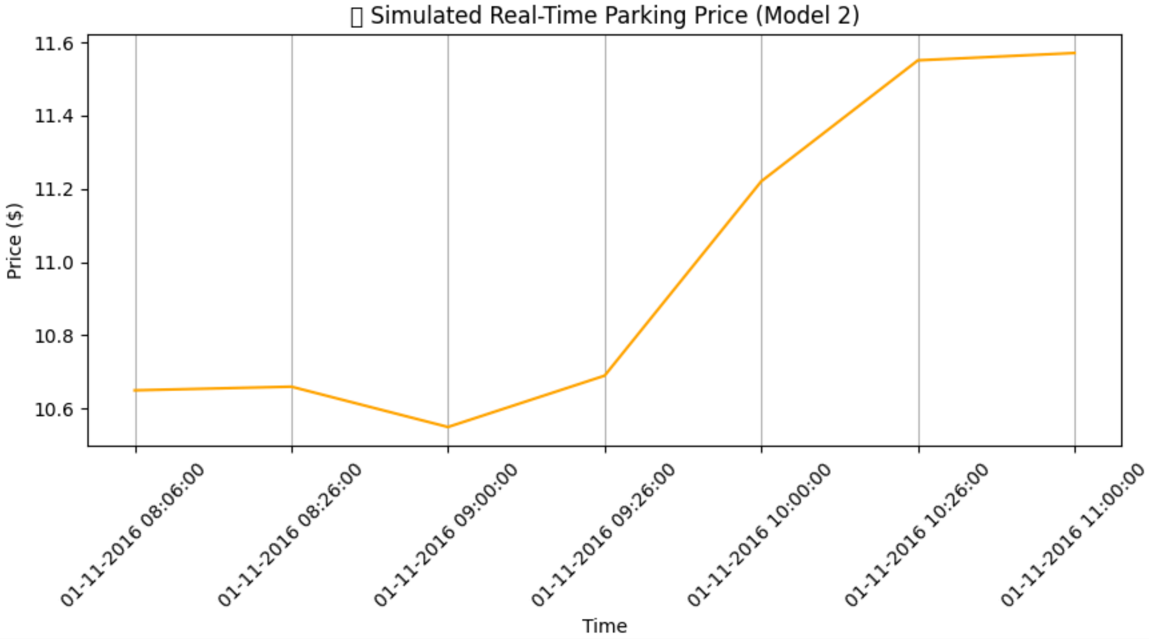
```
lot_df = df[df['Location_ID'] == df['Location_ID'].unique()[0]].copy()
lot_df = lot_df.sort_values(by=['Date', 'Time']).reset_index(drop=True)

from bokeh.layouts import column

source = ColumnDataSource(data=dict(x=[], y=[]))

p = figure(title="⏱️ Real-Time Parking Price", x_axis_label='Time', y_axis_label='Price ($)',
           width=800, height=400, x_range=[])

p.line(x='x', y='y', source=source, line_width=2, legend_label='Demand-Based Price', color='orange')
p.legend.location = 'top_left'
```

```
def get_price(row, base_price=10):
    occ = row['Occupancy']
    cap = row['Capacity']
    queue = row['QueueLength']
    traffic = {'low': 0, 'medium': 1, 'high': 2}.get(row['TrafficConditionNearby'], 1)
    special = row['IsSpecialDay']
    vehicle = {'car': 1, 'bike': 0.5, 'truck': 1.5}.get(row['Vehicle_Type'], 1)

    demand = (1 * (occ / cap)) + (0.5 * queue) - (0.7 * traffic) + (2 * special) + vehicle
    norm_demand = (demand - 0.5) / (5.5 - 0.5)  # Normalize between 0–1
    price = base_price * (1 + 0.3 * norm_demand)
    return round(min(max(price, 5), 20), 2)
```

```
Start coding or generate with AI.
```

```
from IPython.display import clear_output
import matplotlib.pyplot as plt

timestamps = []
prices = []

for i in range(len(lot_df)):
    row = lot_df.iloc[i]
    timestamp = f"{row['Date']} {row['Time']}"
    price = get_price(row)

    timestamps.append(timestamp)
    prices.append(price)

    clear_output(wait=True)
    plt.figure(figsize=(10, 4))
    plt.plot(timestamps, prices, color='orange', label='Demand-Based Price')
    plt.title("⏱️ Simulated Real-Time Parking Price (Model 2)")
    plt.xlabel("Time")
    plt.ylabel("Price ($)")
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

    time.sleep(0.5)  # 50ms delay to simulate speed
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-17-906930772.py in <cell line: 0>()
     21     plt.ylabel("Price ($)")
     22     plt.xticks(rotation=45)
---> 23     plt.grid(True)
     24     plt.legend()
     25     plt.tight_layout()

                                    5 frames
/usr/local/lib/python3.11/dist-packages/matplotlib/artist.py in <lambda>(self, **kwargs)
    144             return
    145 
--> 146         cls.set = lambda self, **kwargs: Artist.set(self, **kwargs)
    147         cls.set.__name__ = "set"
    148         cls.set.__qualname__ = f"{cls.__qualname__}.set"

KeyboardInterrupt: 

/usr/local/lib/python3.11/dist-packages/IPython/core/events.py:89: UserWarning: Glyph 9201 (\N{STOPWATCH}) missing from font(s) Deja
  func(*args, **kwargs)
```


□ Simulated Real-Time Parking Price (Model 2)
Price ($)
01-11-2016 08:06:00
01-11-2016 08:26:00
01-11-2016 09:00:00
01-11-2016 09:26:00
01-11-2016 10:00:00
01-11-2016 10:26:00
01-11-2016 11:00:00
Time