In [1]:
```
# Import the required libraries and dependencies
import pandas as pd
import hvplot.pandas as hvplot
from pathlib import Path
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
# Read in the CSV file as a Pandas DataFrame
CO2_df = pd.read_csv(
    Path("./Resources/CO2 Emissions_Canada.csv")
)

# Review the DataFrame
CO2_df.head()
```

Out[2]:

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244

In [3]:
```
CO2_df.dtypes
```

Out[3]:
```
Make                              object
Model                             object
Vehicle Class                     object
Engine Size(L)                   float64
Cylinders                          int64
Transmission                      object
Fuel Type                         object
Fuel Consumption City (L/100 km)  float64
Fuel Consumption Hwy (L/100 km)   float64
Fuel Consumption Comb (L/100 km)  float64
Fuel Consumption Comb (mpg)         int64
CO2 Emissions(g/km)                 int64
dtype: object
```

In [4]:
```
# Scale price data, return, and variance values
CO2_scaled = StandardScaler().fit_transform(
    CO2_df[["Engine Size(L)",
            "Cylinders",
            "Fuel Consumption City (L/100 km)",
            "Fuel Consumption Hwy (L/100 km)",
            "CO2 Emissions(g/km)"]]
)
```

In [5]:
```
CO2_scaled
```

Out[5]:
```
array([[-0.85672099, -0.88340757, -0.75900153, -1.05278069, -0.93293275],
       [-0.5613172 , -0.88340757, -0.3875769 , -0.60320221, -0.50564599],
       [-1.22597573, -0.88340757, -1.87327544, -1.45740132, -1.95842095],
       ...,
       [-0.85672099, -0.88340757, -0.24472127, -0.19858158, -0.18090806],
       [-0.85672099, -0.88340757, -0.3875769 , -0.33345513, -0.31763982],
       [-0.85672099, -0.88340757, -0.10186564, -0.15362374, -0.0441763 ]])
```

In [6]:
```
# Create a DataFrame with the scaled data
CO2_scaled_df = pd.DataFrame(
    CO2_scaled,
    columns=["Engine Size(L)",
            "Cylinders",
             "Fuel Consumption City (L/100 km)",
            "Fuel Consumption Hwy (L/100 km)",
            "CO2 Emissions(g/km)"]
)
CO2_scaled_df.head()
```

Out[6]:

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	CO2 Emissions(g/km)
0	-0.856721	-0.883408	-0.759002	-1.052781	-0.932933
1	-0.561317	-0.883408	-0.387577	-0.603202	-0.505646
2	-1.225976	-0.883408	-1.873275	-1.457401	-1.958421
3	0.251043	0.210575	0.040990	0.026208	0.075464
4	0.251043	0.210575	-0.130437	-0.153624	-0.112542

In [7]:
```
fuel_type_dummies = pd.get_dummies(CO2_df[["Vehicle Class","Transmission","Fuel Type"]])
fuel_type_dummies.head()
```

Out[7]:

	Vehicle Class_COMPACT	Vehicle Class_FULL-SIZE	Vehicle Class_MID-SIZE	Vehicle Class_MINICOMPACT	Vehicle Class_MINIVAN	Vehicle Class_PICKUP TRUCK - SMALL	Vehicle Class_PICKUP TRUCK - STANDARD	Vehicle Class_SPECIAL PURPOSE VEHICLE	Vehicle Class_STATION WAGON - MID-SIZE	Vehicle Class_STATION WAGON - SMALL
0	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0

5 rows × 48 columns

In [8]:
```
# Concatenate the "FuelType" variables with the scaled data DataFrame.
CO2_scaled_df = pd.concat([CO2_scaled_df, fuel_type_dummies], axis=1)

# Display the sample data
CO2_scaled_df.head()
```

Out[8]:

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	CO2 Emissions(g/km)	Vehicle Class_COMPACT	Vehicle Class_FULL-SIZE	Vehicle Class_MID-SIZE	Vehicle Class_MINICOMPACT	Vehicle Class_MINIVAN	...	Tr
0	-0.856721	-0.883408	-0.759002	-1.052781	-0.932933	1	0	0	0	0	...	
1	-0.561317	-0.883408	-0.387577	-0.603202	-0.505646	1	0	0	0	0	...	
2	-1.225976	-0.883408	-1.873275	-1.457401	-1.958421	1	0	0	0	0	...	
3	0.251043	0.210575	0.040990	0.026208	0.075464	0	0	0	0	0	...	
4	0.251043	0.210575	-0.130437	-0.153624	-0.112542	0	0	0	0	0	...	

5 rows × 53 columns

In [9]:
```
# Create a a list to store inertia values and the values of k
inertia = []
k = list(range(1, 11))
```

In [10]:
```
# Create a for-loop where each value of k is evaluated using the K-means algorithm
# Fit the model using the service_ratings DataFrame
# Append the value of the computed inertia from the `inertia_` attribute of the KMeans model instance
for i in k:
    k_model = KMeans(n_clusters=i, random_state=0)
    k_model.fit(CO2_scaled_df)
    inertia.append(k_model.inertia_)
```

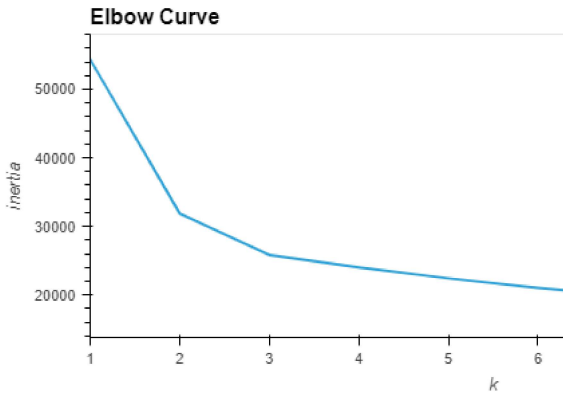
In [11]:
```
# Define a DataFrame to hold the values for k and the corresponding inertia
elbow_data = {"k": k, "inertia": inertia}

# Create the DataFrame from the elbow data
df_elbow = pd.DataFrame(elbow_data)

# Review the DataFrame
df_elbow.head(10)
```

Out[11]:

	k	inertia
0	1	54336.612187
1	2	31865.213784
2	3	25828.568960
3	4	23994.765517
4	5	22410.129357
5	6	21010.493916
6	7	19828.526963
7	8	18706.751577
8	9	18020.666990
9	10	17457.270982

In [12]:
```
# Plot the DataFrame
df_elbow.hvplot.line(
    x="k",
    y="inertia",
    title="Elbow Curve",
    xticks=k
)
```

Out[12]:



In [13]:
```
# Initialize the K-Means model with n_clusters=4
model = KMeans(n_clusters=3)
```

In [14]:
```
# Fit the model for the df_stocks_scaled DataFrame
model.fit(CO2_scaled_df)
```

Out[14]:
```
KMeans(n_clusters=3)
```

In [15]:
```
# Predict the model segments (clusters)
CO2_clusters = model.predict(CO2_scaled_df)

# View the stock segments
print(CO2_clusters)
```

```
[0 0 0 ... 0 0 2]
```

In [16]:
```
# Create a new column in the DataFrame with the predicted clusters
CO2_scaled_df["CO2_Clusters"] = CO2_clusters

# Review the DataFrame
CO2_scaled_df
```

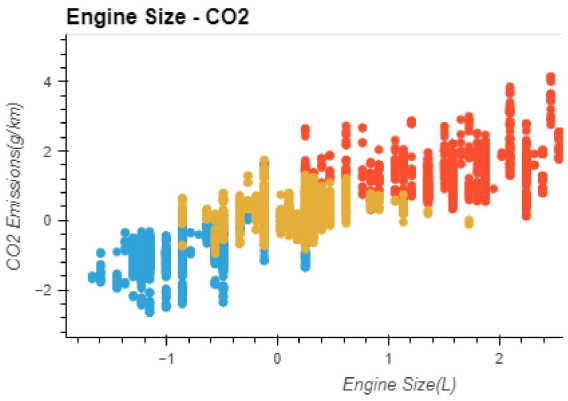
Out[16]:

	Engine Size(L)	Cylinders	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	CO2 Emissions(g/km)	Vehicle Class_COMPACT	Vehicle Class_FULL-SIZE	Vehicle Class_MID-SIZE	Vehicle Class_MINICOMPACT	Vehicle Class_MINIVAN	...
0	-0.856721	-0.883408	-0.759002	-1.052781	-0.932933	1	0	0	0	0	...
1	-0.561317	-0.883408	-0.387577	-0.603202	-0.505646	1	0	0	0	0	...
2	-1.225976	-0.883408	-1.873275	-1.457401	-1.958421	1	0	0	0	0	...
3	0.251043	0.210575	0.040990	0.026208	0.075464	0	0	0	0	0	...
4	0.251043	0.210575	-0.130437	-0.153624	-0.112542	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
7380	-0.856721	-0.883408	-0.530433	-0.603202	-0.539829	0	0	0	0	0	...
7381	-0.856721	-0.883408	-0.387577	-0.333455	-0.317640	0	0	0	0	0	...
7382	-0.856721	-0.883408	-0.244721	-0.198582	-0.180908	0	0	0	0	0	...
7383	-0.856721	-0.883408	-0.387577	-0.333455	-0.317640	0	0	0	0	0	...
7384	-0.856721	-0.883408	-0.101866	-0.153624	-0.044176	0	0	0	0	0	...

7385 rows × 54 columns

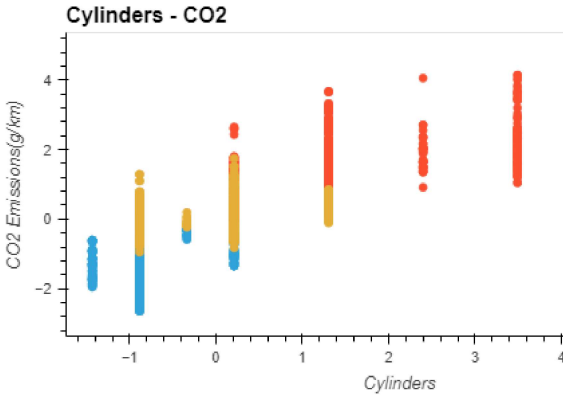
In [30]:
```
# Create a scatter plot with x=,  y=
CO2_scaled_df.hvplot.scatter(
    x="Engine Size(L)",
    y="CO2 Emissions(g/km)",
    by="CO2_Clusters",
    title = "Engine Size - CO2"
)
```

Out[30]:



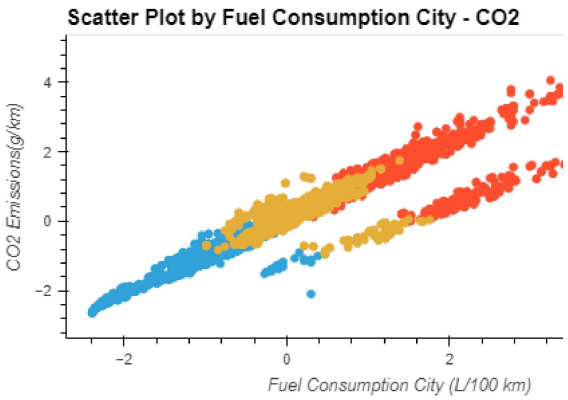
In [31]:
```
# Create a scatter plot with x=,  y=
CO2_scaled_df.hvplot.scatter(
    x="Cylinders",
    y="CO2 Emissions(g/km)",
    by="CO2_Clusters",
    title = "Cylinders - CO2"
)
```

Out[31]:



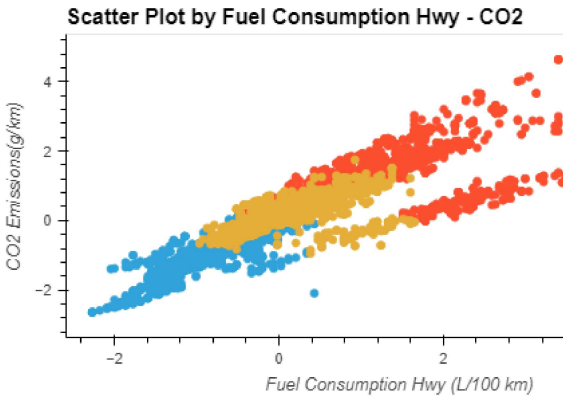
In [33]:
```
# Create a scatter plot with x=,  y=
CO2_scaled_df.hvplot.scatter(
    x="Fuel Consumption City (L/100 km)",
    y="CO2 Emissions(g/km)",
    by="CO2_Clusters",
    title = "Scatter Plot by Fuel Consumption City - CO2"
)
```

Out[33]:



In [34]:
```
# Create a scatter plot with x=,  y=
CO2_scaled_df.hvplot.scatter(
    x="Fuel Consumption Hwy (L/100 km)",
    y="CO2 Emissions(g/km)",
    by="CO2_Clusters",
    title = "Scatter Plot by Fuel Consumption Hwy - CO2"
)
```

Out[34]:


Scatter Plot by Fuel Consumption Hwy - CO2
CO2 Emissions(g/km)
Fuel Consumption Hwy (L/100 km)
4
2
0
−2
−2
0
2

In [ ]: