JAISE GEORGE

23122016

3MScDs A

VISUALIZATION OF INDIAN AUTOMOBILE DATASET USING JAVA PROGRAMMING

INTRODUCTION

The project aims to explore and analyse an Indian automobile dataset through comprehensive visualizations and data analysis techniques using Java programming language. By leveraging this dataset, the project seeks to uncover and illustrate the relationships between different features of vehicles, such as capacity, power, fuel efficiency, and more. The goal is to provide valuable insights into the Indian automobile market, helping stakeholders make informed decisions based on data-driven analysis

DATASET DESCRIPTION

The data taken into consideration is taken from Kaggle website which hosts a variety of datasets from all over the world. The dataset contains 5975 rows and 14 columns, cars with their variants. The data concerns pricing of vehicles in rupees, to be predicted in terms of 5 multivalued discrete attributes - manufacturer, location, fuel type, transmission, ownership, ownership and 6 continuous attributes - year, km driven, engine, seats, horsepower.

PROBLEM STATEMENT

Develop a Java-based visualization tool for an extensive automobile dataset that provides interactive, real-time, and user-friendly insights into vehicle specifications, market trends, and performance metrics. The tool should address the complexity of the data by offering various visualization options, facilitate easy exploration and comparison through interactive features, and ensure efficient performance even with large datasets.

STEPS FOR CREATING MAVEN PROJECT

Creating a Maven project in Visual Studio Code involves a few steps:

1. Install Java Development Kit (JDK)

2. Install Maven Extension for Visual Studio Code: Open Visual Studio Code, go to the Extensions view by clicking on the square icon in the sidebar or pressing Ctrl+Shift+X. Search for "Maven for Java" and install it.

3. Open the Command Palette by pressing Ctrl+Shift+P.

4. Create a New Maven Project:

5. Choose a location for your project and press Enter.

6. Open the Project in Visual Studio Code: Once the project is created, open it in Visual Studio Code by selecting "Open Folder" from the File menu and navigating to the location of your Maven project

7. Edit pom.xml: Open the pom.xml file in Visual Studio Code. This file contains project configuration including dependencies. You can add dependencies here.

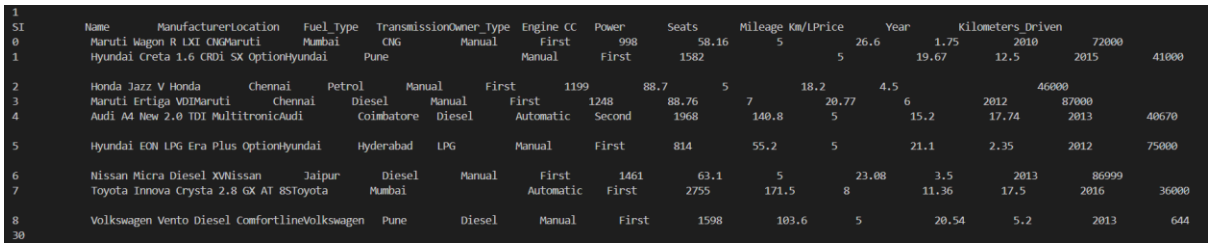
DATA LOADING

CSVReader.java

```
package com.example;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class CSVReader {

public static void mian(String[] arg){

CSVReader csvReader=new CSVReader();

csvReader.reader();

}

public void reader() {

String csvFile = "indian-auto-mpg (1).csv"; //READING CSV FILE

String line = "";

String cvsSplitBy = ",";

int maxRows = 10;

try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {

int rowNum = 0;

while ((line = br.readLine()) != null && rowNum < maxRows) {

if(rowNum == 0) {

// Print column names
```

```
String[] columnNames = line.split(cvsSplitBy);

for (String columnName : columnNames) {

System.out.printf("%-12s", columnName);

}

System.out.println(); // Move to the next line after printing column names

} else {

String[] data = line.split(cvsSplitBy);

for (String value : data) {

System.out.printf("%-13s", value);

}

System.out.println(); // Move to the next line for the next row

}

rowNum++;

}

} catch (IOException e) {

e.printStackTrace();

}

}

}
```

OUTPUT



INTERPRETATION

The main function of this code is to pass our csv file to CSV reader class and print the first seven rows of the data set.

DATA CLEANING

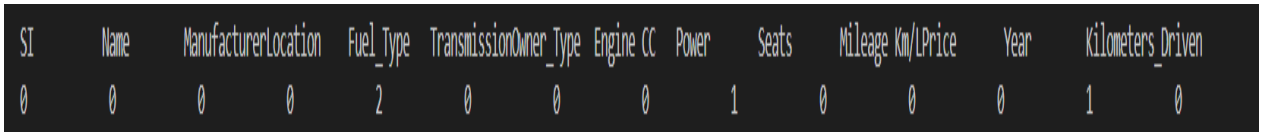
NULLReader.java:

```
package com.example;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class NULLReader {

public void NulREAD() {

String csvFile = "indian-auto-mpg (1).csv";

String line = "";

String cvsSplitBy = ",";

int maxRows = 10; // Maximum rows to process

try (BufferedReader br = new BufferedReader(new FileReader(csvFile))) {

int rowNum = 0;

int numColumns = 0;

int[] nullCountByColumn=new int[300];

String[] columnNames = null;

while ((line = br.readLine()) != null && rowNum < maxRows) {

String[] data = line.split(cvsSplitBy);

if (rowNum == 0) {

numColumns = data.length;
```

```
columnNames = data;
nullCountByColumn = new int[numColumns];
} else {
for (int i = 0; i < numColumns && i < data.length; i++) {
if (data[i].isEmpty() || data[i].equalsIgnoreCase("null")) {
nullCountByColumn[i]++;
}
}
}
rowNum++;
}
// Print column names
for (String columnName : columnNames) {
System.out.printf("%-12s", columnName); // Adjust width as needed
}
System.out.println(); // Move to the next line after printing column names
// Print sum of null values for each column
for (int i = 0; i < numColumns; i++) {
System.out.printf("%-13d", nullCountByColumn[i]);
}
System.out.println(); // Move to the next line
} catch (IOException e) {
e.printStackTrace();
}
}
```

}

OUTPUT



SI	Name	ManufacturerLocation	Fuel_Type	TransmissionOwner_Type	Engine CC	Power	Seats	Mileage Km/LPrice	Year	Kilometers_Driven
0	0	0	0	2	0	0	0	1	0	0	0	1	0

INTERPRETATION

In code we will pass our data set into the code and it will find the null values from the data set. After finding the null values it will print the count of null value in each coloumns.

CSVProcessor.java:

```
package com.example;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class CSVProcessor {

public void removeNullRows(String inputCsvFile, String outputCsvFile) {

String line = "";

String cvsSplitBy = ",";

List<List<String>> data = new ArrayList<>();

try (BufferedReader br = new BufferedReader(new FileReader(inputCsvFile))) {

// Read CSV into 2D List

while ((line = br.readLine()) != null) {

String[] row = line.split(cvsSplitBy);
```

```
List<String> rowData = new ArrayList<>();
for (String value : row) {
rowData.add(value.trim());
}
data.add(rowData);
}
// Remove rows containing null values
List<List<String>> newData = new ArrayList<>();
for (List<String> row : data) {
boolean containsNull = false;
for (String value : row) {
if (value.isEmpty()) {
containsNull = true;
break;
}
}
if (!containsNull) {
newData.add(row);
}
}
// Write modified data to output CSV
try (BufferedWriter bw = new BufferedWriter(new FileWriter(outputCsvFile))) {
for (List<String> row : newData) {
bw.write(String.join(",", row));
bw.newLine();
```

```
}

}

System.out.println("Rows containing null values removed successfully.");

} catch (IOException e) {

e.printStackTrace();

}

}

}
```

OUTPUT:

```
Rows containing null values removed successfully.
```

INTERPRETATION:

This code will take the automobile data set and clean the data set. It will find the null values from the data set and remove this null values and it will create a new data set which has a name output.java which will contain the cleaned data of the input data. And it will return a message that rows containing null values removed successfully.

DATA VISUALIAZATION

PieChart,java:

```
package com.example;

import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVRecord;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.ChartUtilities;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PiePlot;

import org.jfree.data.general.DefaultPieDataset;
```

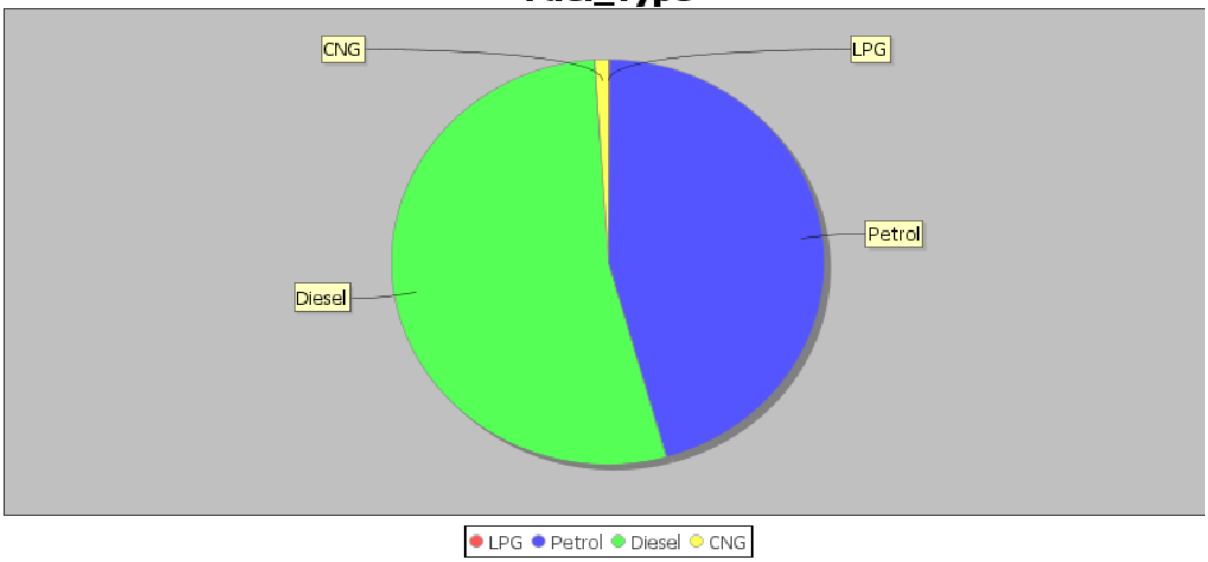
```
import javax.swing.*;

import java.awt.*;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.io.Reader;

import java.util.HashMap;

import java.util.Map;

public class PieChart extends JFrame {

public PieChart(String title) {

super(title);

// Load the data and count occurrences

Map<String, Integer> Manufacturer = loadDataAndCount("output.csv",
"Manufacturer");

Map<String, Integer> Fuel_Type = loadDataAndCount("output.csv", "Fuel_Type");

Map<String, Integer> Owner_Type = loadDataAndCount("output.csv",
"Owner_Type");

Map<String, Integer> seatsData = loadDataAndCount("output.csv", "Seats");

// Create the charts

JFreeChart Manufacturer1 = createChart(Manufacturer, "Year");

JFreeChart Fuel_TypeChart = createChart(Fuel_Type, "Fuel_Type");

JFreeChart Owner_TypeChart = createChart(Owner_Type, "Owner_Type");

JFreeChart seatsChart = createChart(seatsData, "Seats");

File chart3=new File("chart1.png");

try{

ChartUtilities.saveChartAsPNG(chart3,seatsChart,620, 480);
```

```
}catch(IOException e){

}

// Create a panel and add charts

JPanel panel = new JPanel(new GridLayout(2, 3));

panel.add(new ChartPanel(Manufacturer1));

panel.add(new ChartPanel(Fuel_TypeChart));

panel.add(new ChartPanel(Owner_TypeChart));

panel.add(new ChartPanel(seatsChart));

add(panel, BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

pack();

setVisible(true);

}

private static Map<String, Integer> loadDataAndCount(String csvFile, String field) {

Map<String, Integer> dataCount = new HashMap<>();

try (Reader reader = new FileReader(csvFile)) {

Iterable<CSVRecord> records =
CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(reader);

for (CSVRecord record : records) {

String key = record.get(field);

dataCount.put(key, dataCount.getOrDefault(key, 0) + 1);

}

} catch (IOException e) {

e.printStackTrace();

}

return dataCount;
```

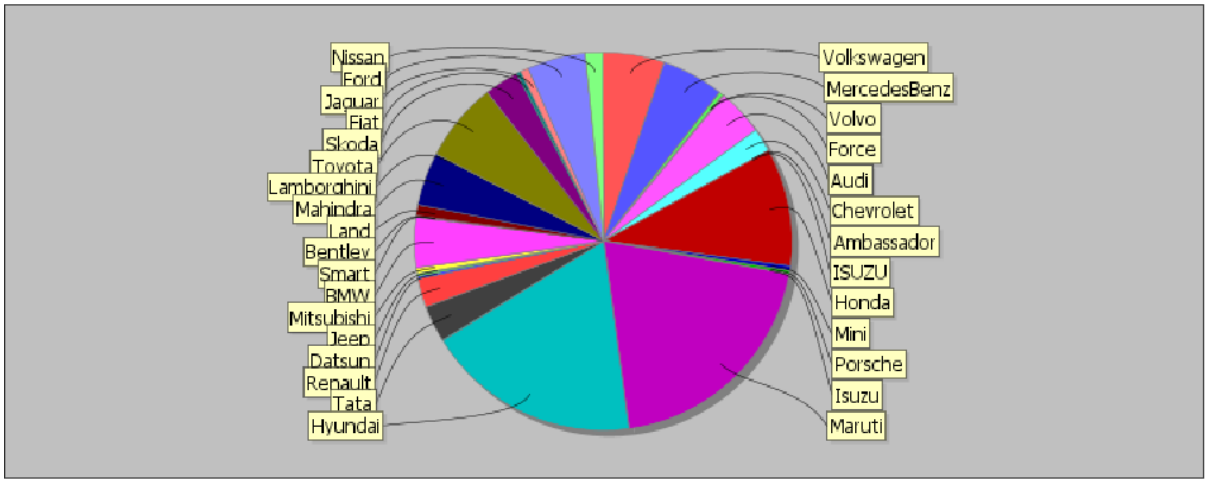
```
}
private static JFreeChart createChart(Map<String, Integer> data, String title) {
DefaultPieDataset dataset = new DefaultPieDataset();
for (Map.Entry<String, Integer> entry : data.entrySet()) {
dataset.setValue(entry.getKey(), entry.getValue());
}
JFreeChart chart = ChartFactory.createPieChart(title, dataset,true,false,false);
PiePlot plot = (PiePlot) chart.getPlot();
return chart;
}
public static void main(String[] args) {
SwingUtilities.invokeLater(() -> {
PieChart example = new PieChart("Pie Chart Example");
example.setSize(800, 600);
example.setLocationRelativeTo(null);
example.setVisible(true);
});
}
}
```

OUTPUT:

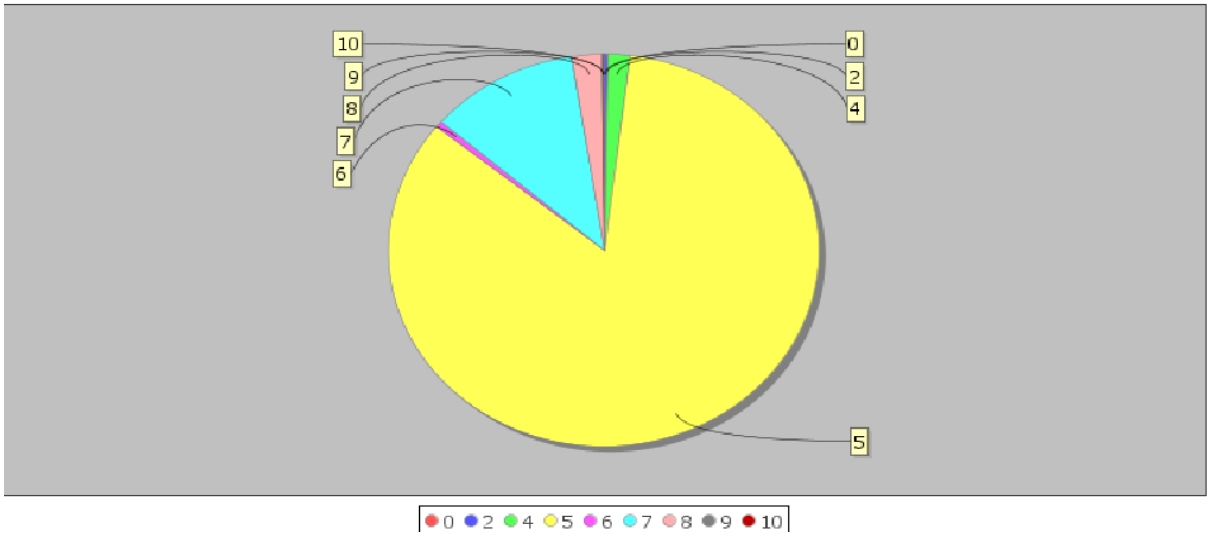
Fuel_Type

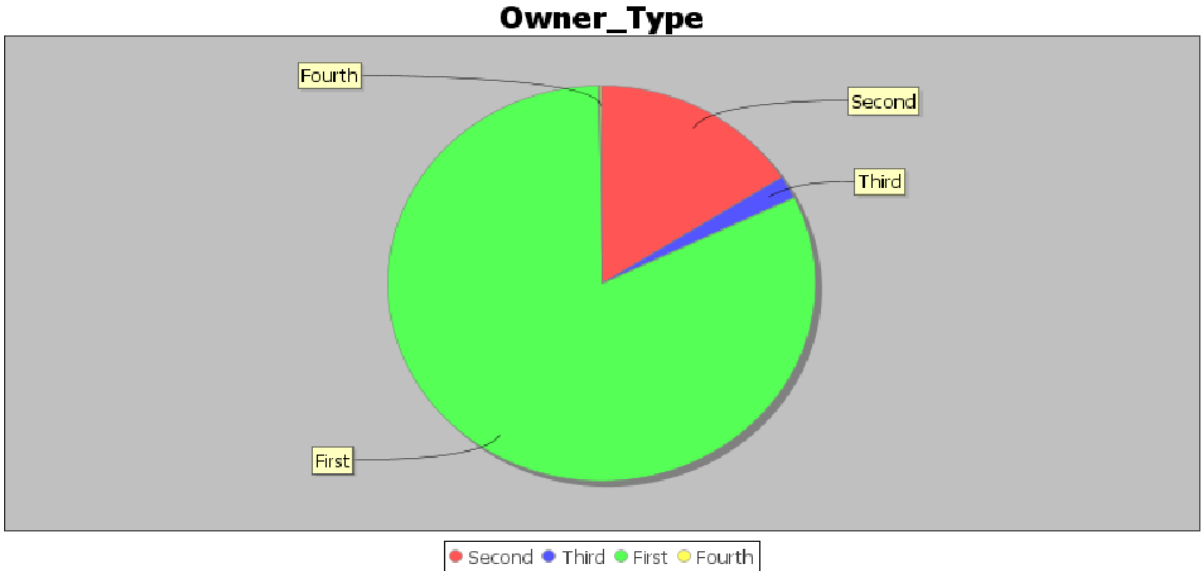


Year



Seats





INTERPRETATION

In this code we are passing the cleaned data set and plot different pie chart for different features. Here We are mainly using features like fuel type, year, seats, owner type .First graph represent the amount of car users using different cars of specific fuel type where we can found that Disel is the main category of fuel that are opted by car users. Next we are finding the pie chart of different manufactures in a year which has more sale. In this we can found that Maruti and Hyundai was sold more in a year. In the third graph we can see that more people prefer 5 seaters when they buy a car. In the forth graph it is clear that all users are preferring first hand car to buy.

Barchart.java

```
package com.example;

import java.io.FileReader;

import java.io.IOException;

import java.io.Reader;

import java.util.HashMap;

import java.util.Map;

import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVRecord;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;
```

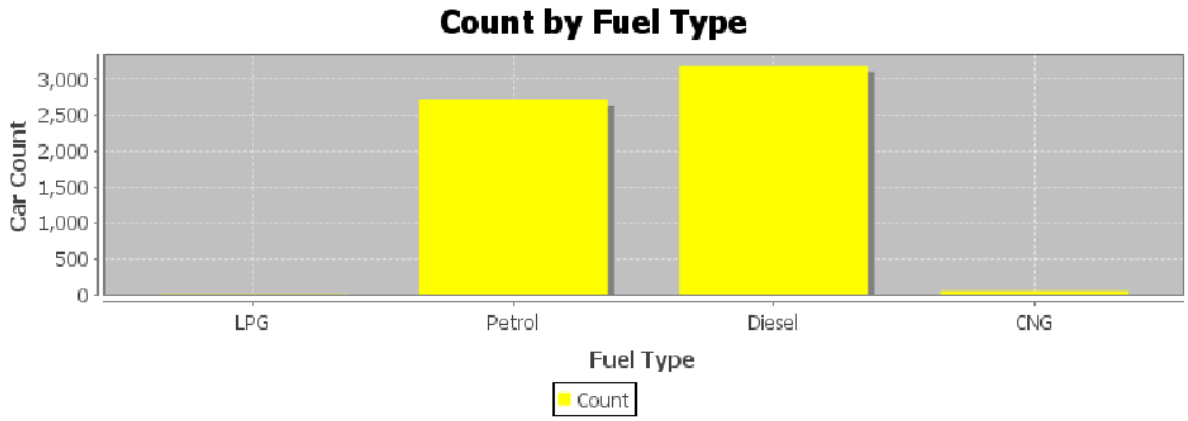
```
import org.jfree.chart.plot.CategoryPlot;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.chart.renderer.category.BarRenderer;

import org.jfree.chart.renderer.category.StandardBarPainter;

import org.jfree.data.category.DefaultCategoryDataset;

import javax.swing.*;

import java.awt.*;

public class Barchart extends JFrame {

public Barchart(String title) {

super(title);

// Load the data and count occurrences

Map<String, Integer> Manufacturer = loadDataAndCount("output.csv",
"Manufacturer");

Map<String, Integer> Year = loadDataAndCount("output.csv", "Year");

Map<String, Integer> Location = loadDataAndCount("output.csv", "Location");

Map<String, Integer> Transmission= loadDataAndCount("output.csv",
"Transmission");

Map<String, Integer> Fuel_Type = loadDataAndCount("output.csv", "Fuel_Type");

Map<String, Integer> seatsData = loadDataAndCount("output.csv", "Seats");

// Create the charts

JFreeChart yearChart = createChart(Year, "Count by Year", "Year", "Car Count");

JFreeChart TransmissionChart = createChart(Transmission, "Count by
Transmission", "Transmission", "Car Count");

JFreeChart LocationChart = createChart(Location, "Count by Location", "Location",
"Car Count");

JFreeChart facturerChart = createChart(Manufacturer, "Count by Manufacturer",
"Manufacturer", "Car Count");
```

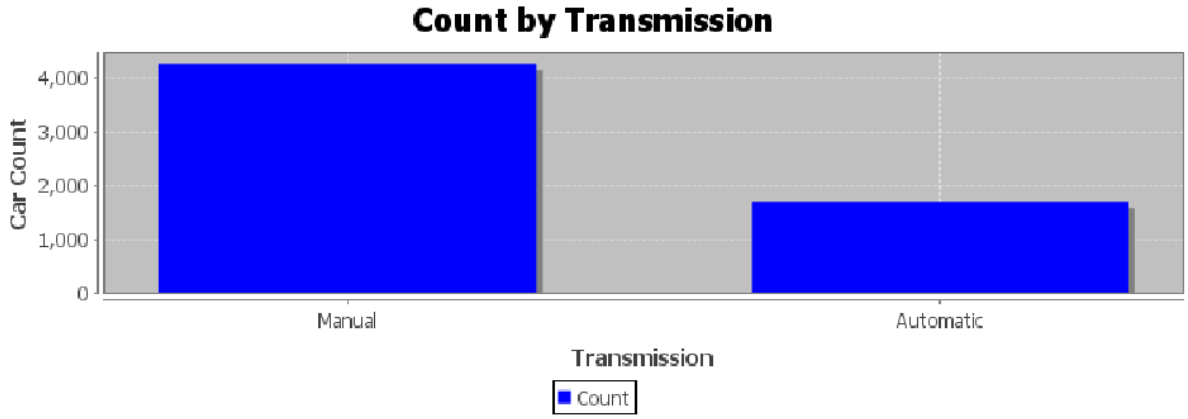
```
JFreeChart Fuel_TypeChart = createChart(Fuel_Type, "Count by Fuel Type", "Fuel Type", "Car Count");

JFreeChart seatsChart = createChart(seatsData, "Count by Seats", "Seats", "Car Count");

// Create a panel and add charts

JPanel panel = new JPanel(new GridLayout(3, 2));

panel.add(new ChartPanel(yearChart));

panel.add(new ChartPanel(Fuel_TypeChart));

panel.add(new ChartPanel(seatsChart));

panel.add(new ChartPanel(TransmissionChart));

panel.add(new ChartPanel(LocationChart));

panel.add(new ChartPanel(facturerChart));

add(panel, BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

pack();

setVisible(true);

}

private static Map<String, Integer> loadDataAndCount(String csvFile, String field) {

Map<String, Integer> dataCount = new HashMap<>();

try (Reader reader = new FileReader(csvFile)) {

Iterable<CSVRecord> records =
CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(reader);

for (CSVRecord record : records) {

String key = record.get(field);

dataCount.put(key, dataCount.getOrDefault(key, 0) + 1);

}

} catch (IOException e) {
```

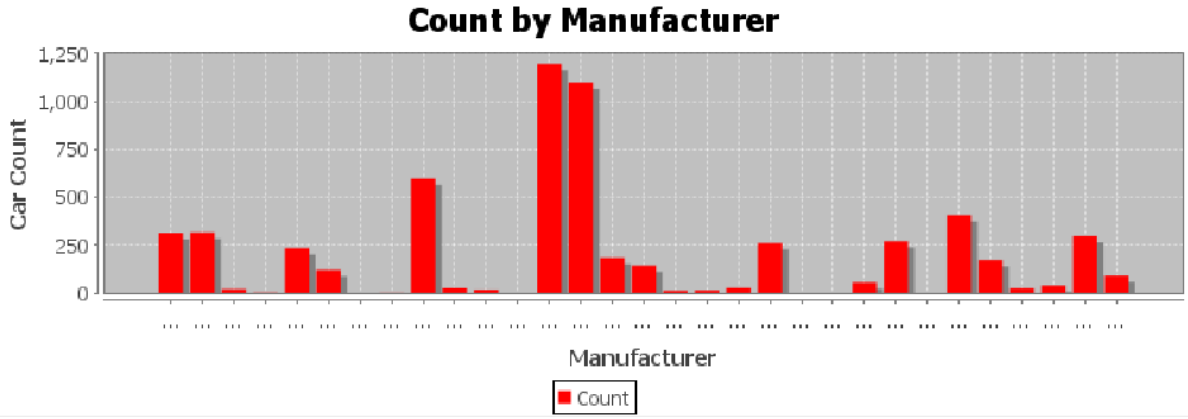
```
e.printStackTrace();

}

return dataCount;

}

private static JFreeChart createChart(Map<String, Integer> data, String title, String
xAxisLabel, String yAxisLabel) {

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

for (Map.Entry<String, Integer> entry : data.entrySet()) {

dataset.addValue(entry.getValue(), "Count", entry.getKey());

}

JFreeChart barChart = ChartFactory.createBarChart(

title, // chart title

xAxisLabel, // domain axis label

yAxisLabel, // range axis label

dataset, // data

PlotOrientation.VERTICAL, // orientation

true, // include legend

true, // tooltips

false // URLs?

);

CategoryPlot plot = barChart.getCategoryPlot();

plot.setDomainGridlinesVisible(true);

plot.setRangeGridlinesVisible(true);

BarRenderer renderer = (BarRenderer) plot.getRenderer();

Color[] colors = new Color[] {Color.RED, Color.BLUE, Color.GREEN,
Color.YELLOW, Color.ORANGE, Color.CYAN };
```

```
for (int i = 0; i < dataset.getColumnCount(); i++) {

renderer.setSeriesPaint(0, colors[i % colors.length]);

renderer.setBarPainter(new StandardBarPainter()); // Ensure solid colors without
gradient

}

return barChart;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

Barchart example = new Barchart("Bar Chart");

example.setSize(800, 600);

example.setLocationRelativeTo(null);

example.setVisible(true);

});

}

}
```

OUTPUT







INTERPRETATION

In the first graph it the year wise count of cars which are produced in specific years. from that graph we can see that in 2022 the more are produced in the early years it was less and there is huge increment in car production in this years. In the second we can clearly see that the 5 seater cars are more produced . form the 3rd bar graph we can see that the number of cars sales in Mumbi and Hyderabad are more but in other states it is comparatively less. And in the forth graph we can see that more people are preferring diesel cars more than petrol cars .In the fifth graph we can see that more people are preferring Manual transmission cars. In the last graph we can see that mainly Maruti is selling more number of cars.

Scatterplot.jav

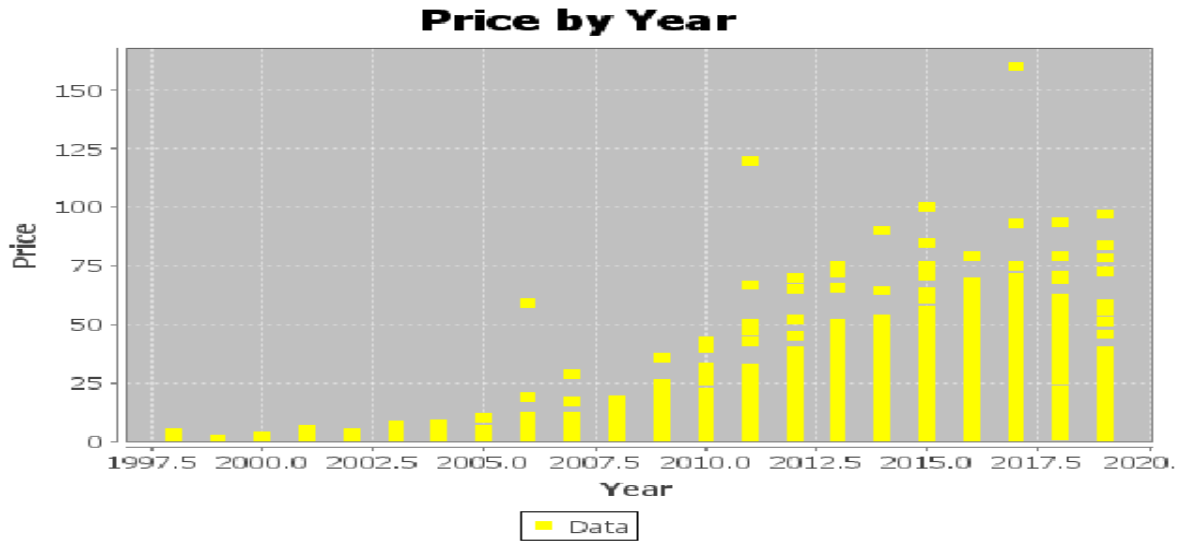
```
package com.example;

import java.io.FileReader;

import java.io.IOException;

import java.io.Reader;

import java.util.ArrayList;

import java.util.List;

import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVRecord;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.chart.plot.XYPlot;

import org.jfree.chart.renderer.category.BarRenderer;

import org.jfree.chart.renderer.category.StandardBarPainter;

import org.jfree.chart.renderer.xy.XYItemRenderer;

import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;

import javax.swing.*;

import java.awt.*;

public class ScatterPlot extends JFrame {

public ScatterPlot(String title) {

super(title);

// Load the data
```

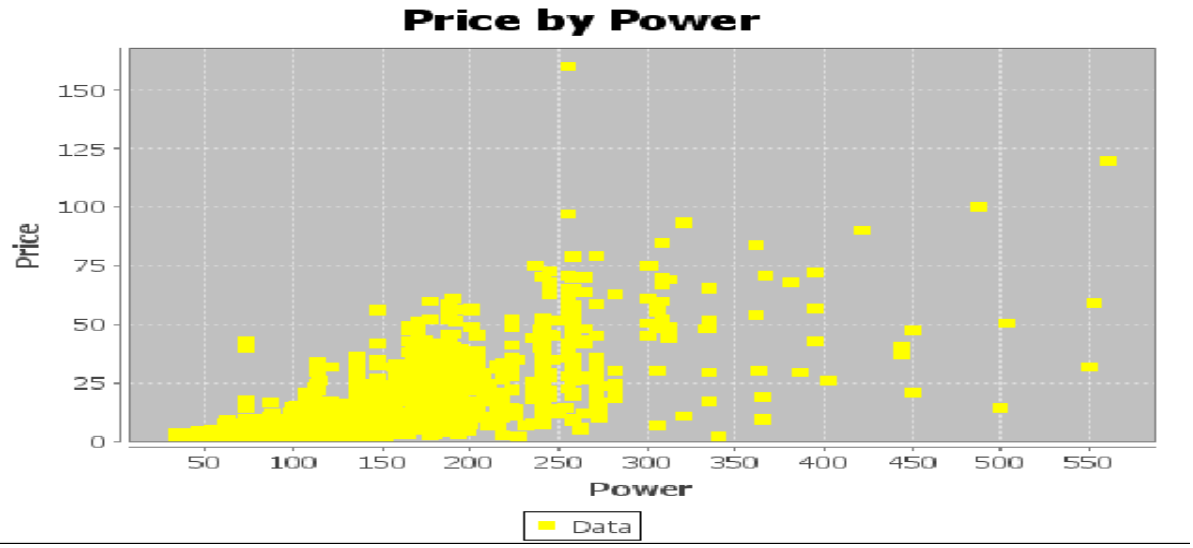
```
String csvFile = "output.csv";

double[][] yearData = loadData(csvFile, "Year", "Price");

double[][] kmData = loadData(csvFile, "Kilometers_Driven", "Price");

double[][] engineData = loadData(csvFile, "Engine CC", "Price");

double[][] powerData = loadData(csvFile, "Power", "Price");

double[][] seatsData = loadData(csvFile, "Seats", "Price");

double[][] mileageData = loadData(csvFile, "Mileage Km/L", "Price");

// Create the charts

JFreeChart yearChart = createChart(yearData, "Price by Year", "Year", "Price");

JFreeChart kmChart = createChart(kmData, "Price by Kilometers Driven",
"Kilometers Driven", "Price");

JFreeChart engineChart = createChart(engineData, "Price by Engine CC", "Engine
CC", "Price");

JFreeChart powerChart = createChart(powerData, "Price by Power", "Power",
"Price");

JFreeChart seatsChart = createChart(seatsData, "Price by Seats", "Seats", "Price");

JFreeChart mileageChart = createChart(mileageData, "Price by Mileage", "Mileage
Km/L", "Price");

// Create a panel and add charts

JPanel panel = new JPanel(new GridLayout(2, 3));

panel.add(new ChartPanel(yearChart));

panel.add(new ChartPanel(kmChart));

panel.add(new ChartPanel(engineChart));

panel.add(new ChartPanel(powerChart));

panel.add(new ChartPanel(seatsChart));

panel.add(new ChartPanel(mileageChart));

add(panel, BorderLayout.CENTER);
```

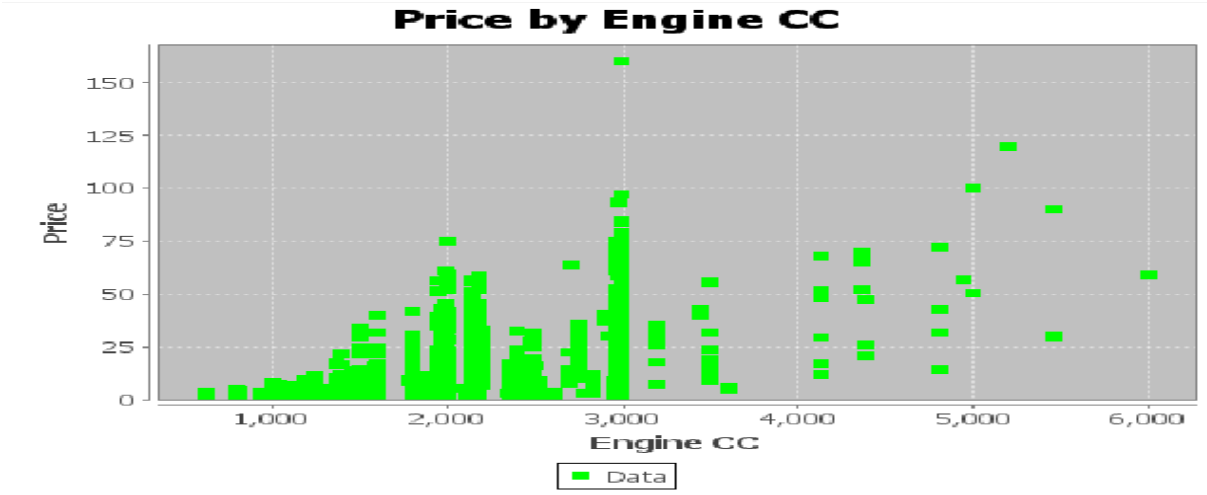
```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

pack();

setVisible(true);

}

private static double[][] loadData(String csvFile, String xField, String yField) {

try (Reader reader = new FileReader(csvFile)) {

Iterable<CSVRecord> records =
CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(reader);

// Convert the iterable to a list to get the size

List<CSVRecord> recordList = new ArrayList<>();

for (CSVRecord record : records) {

recordList.add(record);

}

int recordCount = recordList.size();

double[][] data = new double[2][recordCount];

for (int i = 0; i < recordCount; i++) {

CSVRecord record = recordList.get(i);

data[0][i] = Double.parseDouble(record.get(xField));

data[1][i] = Double.parseDouble(record.get(yField));

}

return data;

} catch (IOException e) {

e.printStackTrace();

return null;

} catch (NumberFormatException e) {

e.printStackTrace();
```

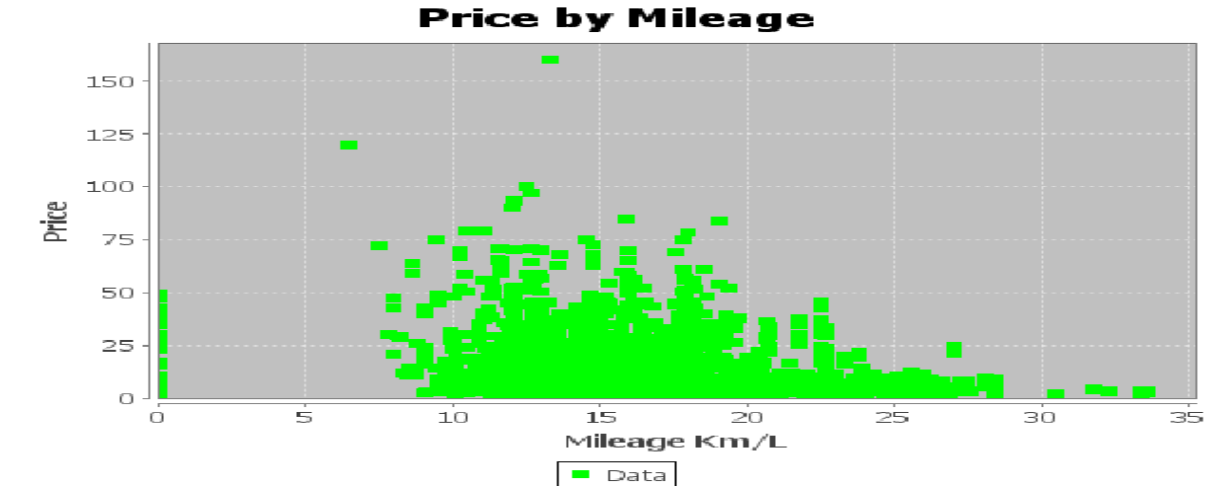
```
return null;

}

}

private static JFreeChart createChart(double[][] data, String title, String xAxisLabel,
String yAxisLabel) {

XYSeries series = new XYSeries("Data");

for (int i = 0; i < data[0].length; i++) {

series.add(data[0][i], data[1][i]);

}

XYSeriesCollection dataset = new XYSeriesCollection(series);

JFreeChart chart = ChartFactory.createScatterPlot(title, xAxisLabel, yAxisLabel,
dataset, PlotOrientation.VERTICAL, true, true, false);

XYPlot scatterPlot = (XYPlot) chart.getPlot();

XYItemRenderer scatterRenderer = scatterPlot.getRenderer();

scatterRenderer.setSeriesPaint(0,Color.GREEN );

return chart;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

ScatterPlot example = new ScatterPlot("Scatter Plot");

example.setSize(800, 600);

example.setLocationRelativeTo(null);

example.setVisible(true);

});

}

}
```

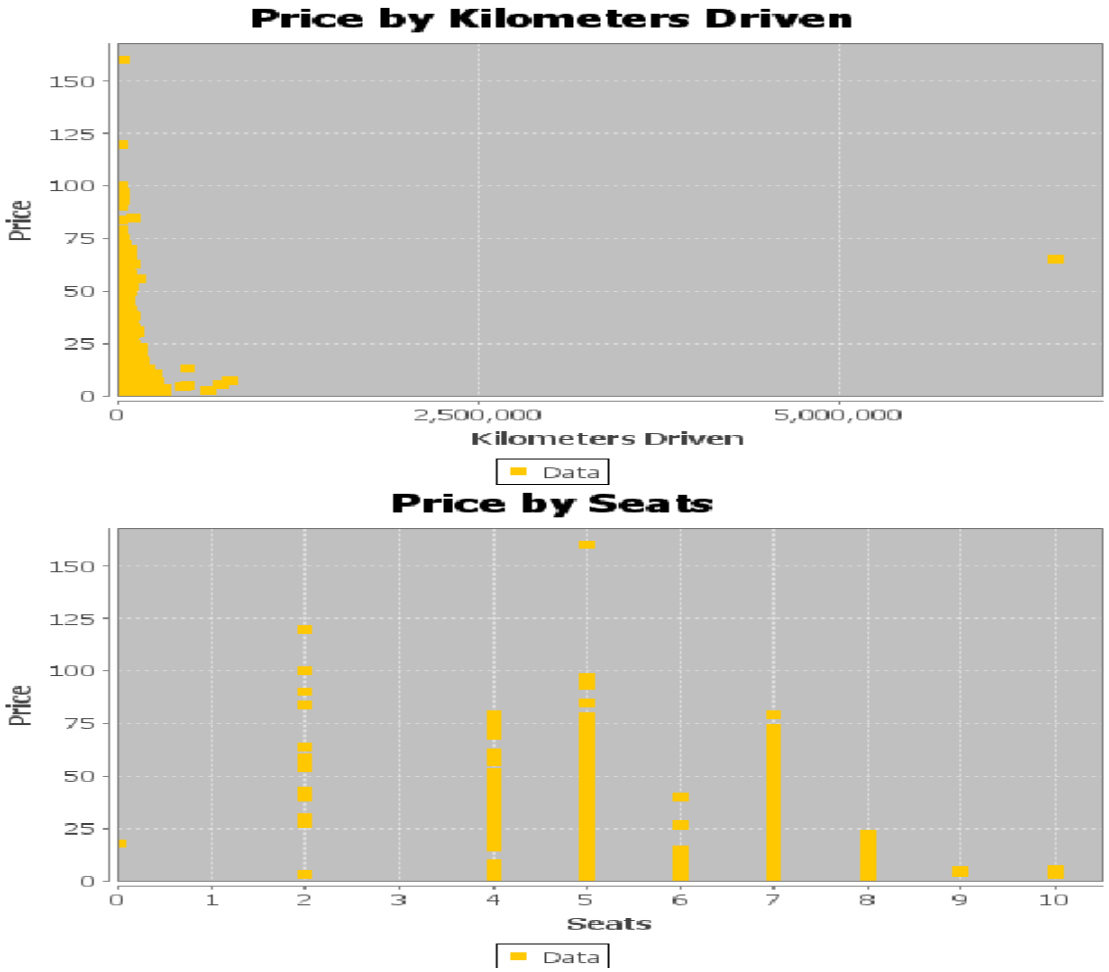
OUTPUT


Price by Year
Price
150
125
100
75
50
25
0
1997.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.
Year
Data


Price by Power
Price
150
125
100
75
50
25
0
50
100
150
200
250
300
350
400
450
500
550
Power
Data


Price by Engine CC
Price
150
125
100
75
50
25
0
1,000
2,000
3,000
4,000
5,000
6,000
Engine CC
Data


Price by Mileage
Price
150
125
100
75
50
25
0
0
5
10
15
20
25
30
35
Mileage Km/L
Data



INTERPRETATION

In the first graph represent the relationship between price and year in a scatter plot .from that graph we can see that as year increases price is also increasing and car price have a more hike in year 2015 to 2020 that's why more points are in that region in scatter plot. In the second graph we can see that price by power scatter plot where as car having more power have more price and we can see that in the range of 150 and 200 power more cars are sold. In the third scatter chart we can see that more points are scattered in between 4000 to 5000 cc category. And the points are more together in 2000 and 3000 cc cars that means at this price range more people are preferring 2000 to 3000 cc cars. In the fourth chart we can see that more people are preferring cars with 15 to 20 km average milage that's why the data points are more scattered there. In the fifth chart we can see that most of the car has driven under 2lak kilometre so most of the people prefer that type of cars. In the sixth chart it a scatter plot for seat by price, in 5 seaters cars more data points are scattered so we can see that more people are preferring five seaters cars.

Linechart.java

package com.example;

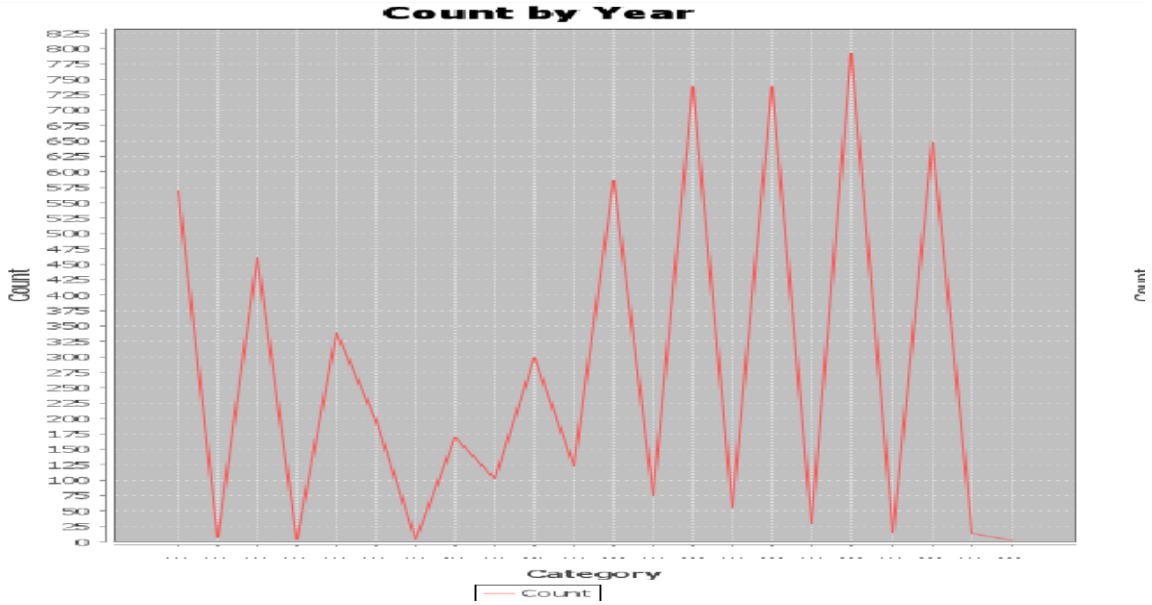
```
import org.apache.commons.csv.CSVFormat;

import org.apache.commons.csv.CSVRecord;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.CategoryPlot;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.data.category.DefaultCategoryDataset;

import javax.swing.*;

import java.awt.*;

import java.io.FileReader;

import java.io.IOException;

import java.io.Reader;

import java.util.HashMap;

import java.util.Map;

public class LineChart extends JFrame {

public LineChart(String title) {

super(title);

// Load the data and count occurrences

Map<String, Integer> yearData = loadDataAndCount("output.csv", "Year");

Map<String, Integer> Location = loadDataAndCount("output.csv", "Location");

Map<String, Integer> Manufacturer= loadDataAndCount("output.csv",
"Manufacturer");

// Create the charts

JFreeChart yearChart = createChart(yearData, "Count by Year");

JFreeChart LocationChart = createChart(Location, "count by Location");
```

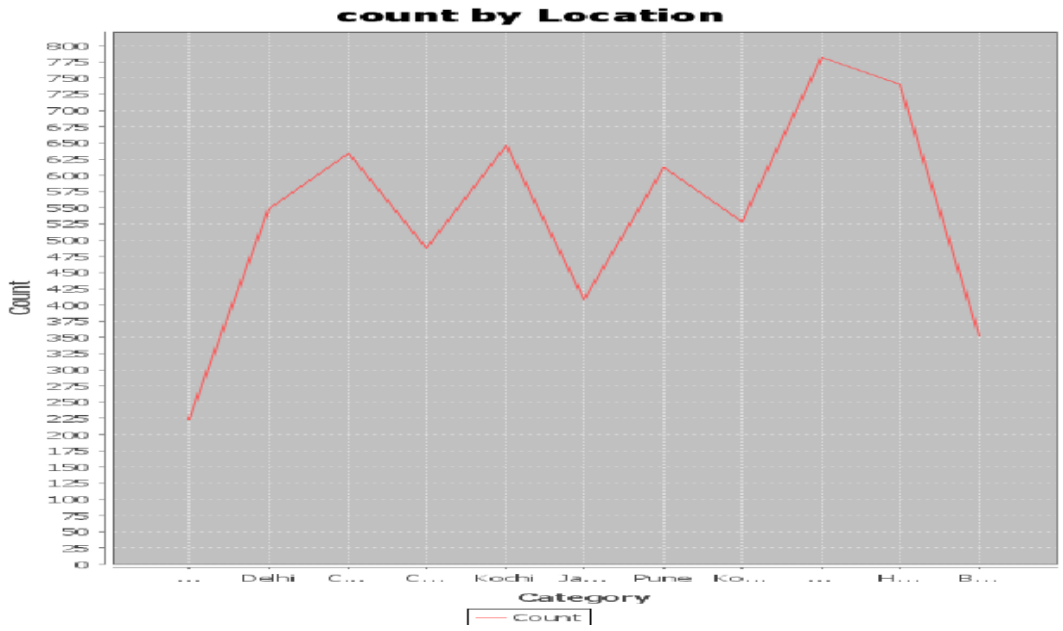
```
JFreeChart ManufacturerChart = createChart(Manufacturer, "Count by Manufacturer");

// Create a panel and add charts

JPanel panel = new JPanel(new GridLayout(1, 3));

panel.add(new ChartPanel(yearChart));

panel.add(new ChartPanel(LocationChart));

panel.add(new ChartPanel(ManufacturerChart));

add(panel, BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

pack();

setVisible(true);

}

private static Map<String, Integer> loadDataAndCount(String csvFile, String field) {

Map<String, Integer> dataCount = new HashMap<>();

try (Reader reader = new FileReader(csvFile)) {

Iterable<CSVRecord> records = CSVFormat.DEFAULT.withFirstRecordAsHeader().parse(reader);

for (CSVRecord record : records) {

String key = record.get(field);

dataCount.put(key, dataCount.getOrDefault(key, 0) + 1);

}

} catch (IOException e) {

e.printStackTrace();

}

return dataCount;

}
```

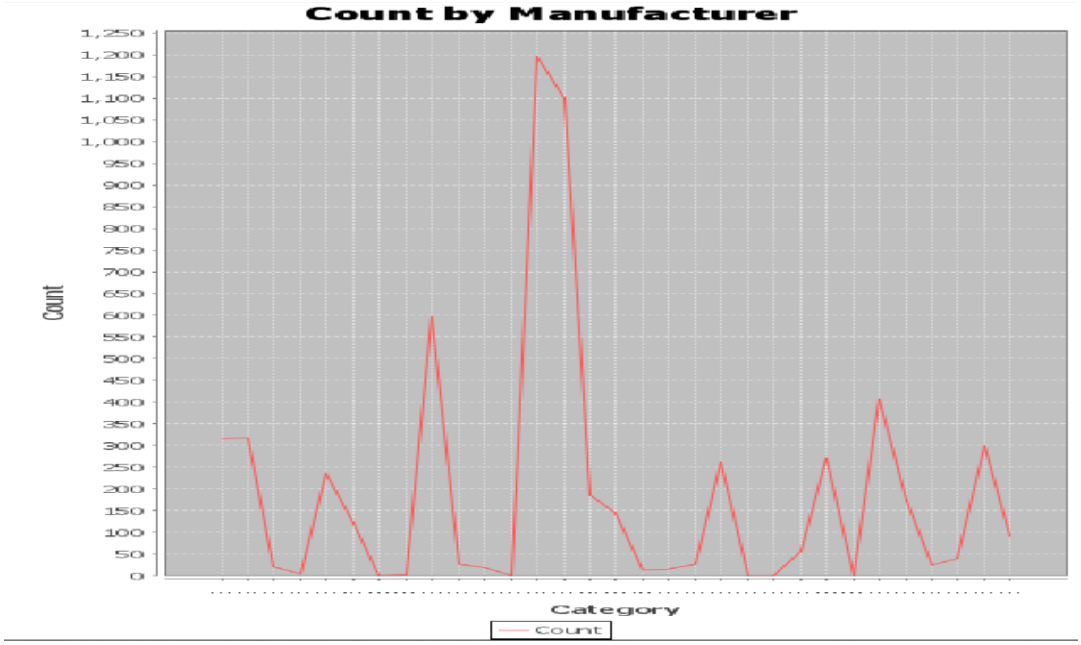
```
private static JFreeChart createChart(Map<String, Integer> data, String title) {
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
for (Map.Entry<String, Integer> entry : data.entrySet()) {
dataset.addValue(entry.getValue(), "Count", entry.getKey());
}
JFreeChart chart = ChartFactory.createLineChart(
title, // chart title
"Category", // domain axis label
"Count", // range axis label
dataset,PlotOrientation.VERTICAL,true, true, false
// data
);
CategoryPlot plot = (CategoryPlot) chart.getPlot();
plot.setDomainGridlinesVisible(true);
plot.setRangeGridlinesVisible(true);
return chart;
}
public static void main(String[] args) {
SwingUtilities.invokeLater(() -> {
LineChart example = new LineChart("Line Chart Example");
example.setSize(800, 600);
example.setLocationRelativeTo(null);
example.setVisible(true);
});
}
```

}

OUTPUT


Count by Year
Count
Category
Count


count by Location
Count
Dalhi
Kochi
Pune
Category
Count



INTERPRETATION

In the first line chart we can see more fluctuation of line in the year 2015 to 2020 that indicate there is a car production hike in the year 2015 to 2020. In the second chart we can see that in regions like Mumbai there is more increment in the car production but in Delhi the car production is less compare to other states. In the last line chart we can see that Maruti has more sale during every year.