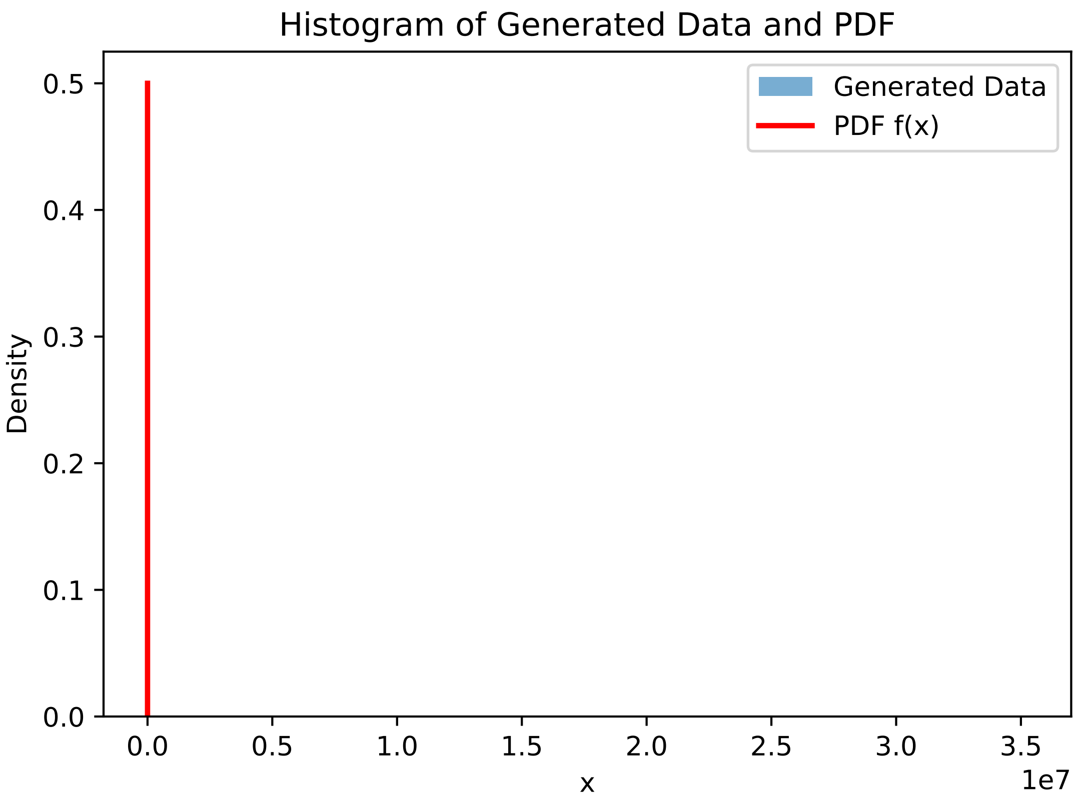
Final Exam

Abdallah chidjou

December 13, 2024

Question 1

```
import numpy as np
import matplotlib.pyplot as plt

# d. Use python, and write a function which takes N = 2000
# and generates N observations from the distribution.

def generate_observations(N=2000):
    # Generate N uniform random variables in [0, 1]
    U = np.random.uniform(0, 1, N)
    # Apply the inverse CDF formula to generate X
    X = (1 / (1 - U)**2) - 1
    return X

# Generate 2000 observations
N = 2000
observations = generate_observations(N)

# Plot the histogram of the generated data
plt.hist(observations, bins=50, density=True, alpha=0.6, label='Generated Data')

# Define the original PDF function
def pdf(x):
    return 1 / (2 * (1 + x)**1.5)

# Generate x values for plotting the PDF
x_vals = np.linspace(0, 1000, 100)  # Limit x-axis to match the filtered data
y_vals = pdf(x_vals)

# Plot the PDF on top of the histogram
plt.plot(x_vals, y_vals, 'r-', linewidth=2, label='PDF f(x)')
plt.xlabel('x')
plt.ylabel('Density')
plt.title('Histogram of Generated Data and PDF')
plt.legend()
plt.show()
```



Question 3

```
# a. Define the quadratic function
def quadratic_function(a, b, c, x):
    return a * x**2 + b * x + c

# b. Define the gradient function
def grad_function(a, b, x):
    return 2 * a * x + b

# c. Gradient descent for minimizing the quadratic function
def gradient_descent(a, b, c, learning_rate=0.1, tol=1e-5, max=1000):
    x = 0
    for i in range(max):
        grad = grad_function(a, b, x)
        # Check for convergence
        if abs(grad) < tol:
            print(f"Converged after {i} iterations.")
            return x
        # Update x using gradient descent
        x = x - learning_rate * grad
    print("Did not converge within the maximum number of iterations.")
    return x
```

```
# d. Test the program with a=3, b=-12, c=9
a, b, c = 3, -12, 9
x_min = gradient_descent(a, b, c)
```

```
## Converged after 16 iterations.
```

```
print(f"The minimum value occurs at x = {x_min}")
```

```
## The minimum value occurs at x = 1.9999991410065407
```

```
print(f"The minimum value of f(x) = {quadratic_function(a, b, c, x_min)}")
```

```
## The minimum value of f(x) = -2.9999999999977867
```

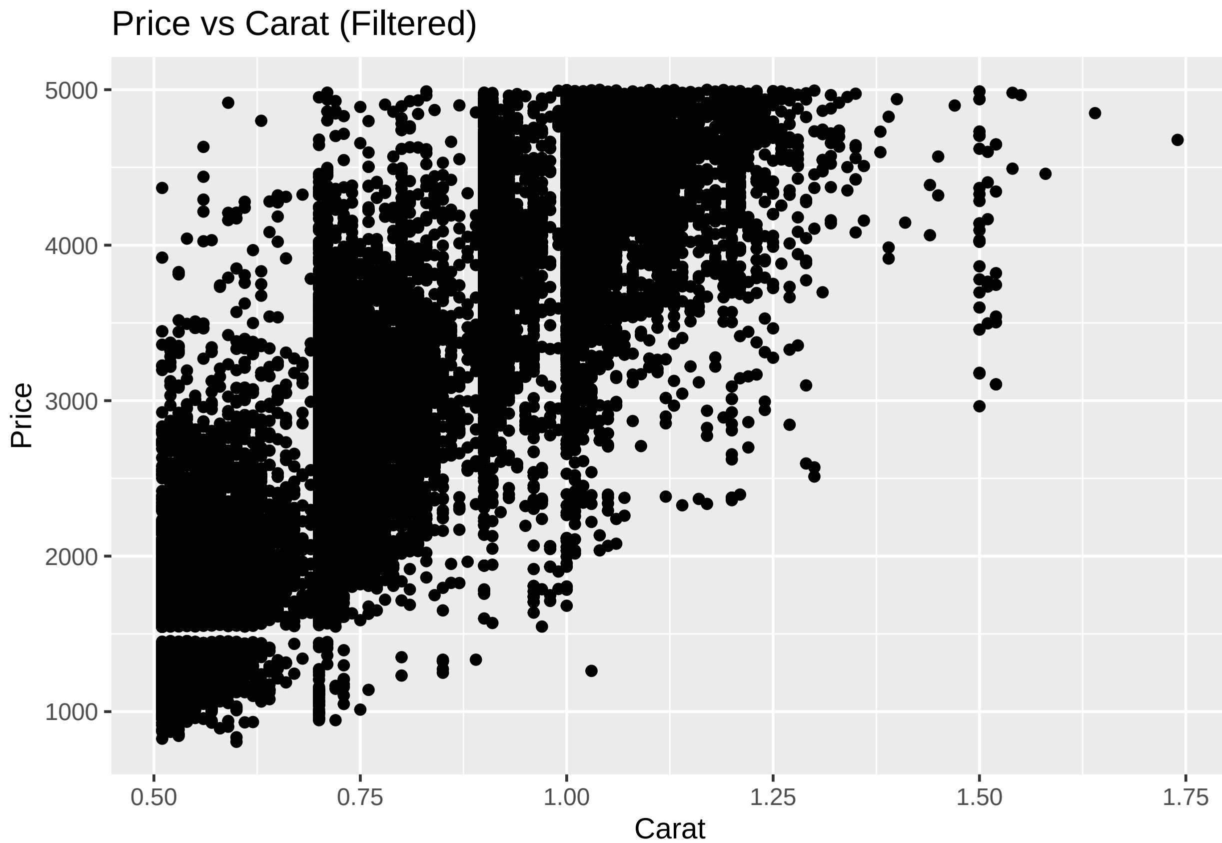
```
#  e. Explanation
# The convergence depends on the learning rate, tolerance, and the problem's characteristics.
# If the gradient became smaller than `tol`, the optimization converged successfully.
# If the maximum number of iterations was reached, it suggests either a slow convergence or a poorly ch
```

Question 4

```
# Load the necessary library
library(ggplot2)

# Load the dataset
dataset <- read.csv("diamonds.csv")

# a. Focus on the diamond observations whose carat is greater
# than 0.5 and whose price is less than 5000.
filtered_data <- subset(dataset, carat > 0.5 & price < 5000)

head(filtered_data)
```

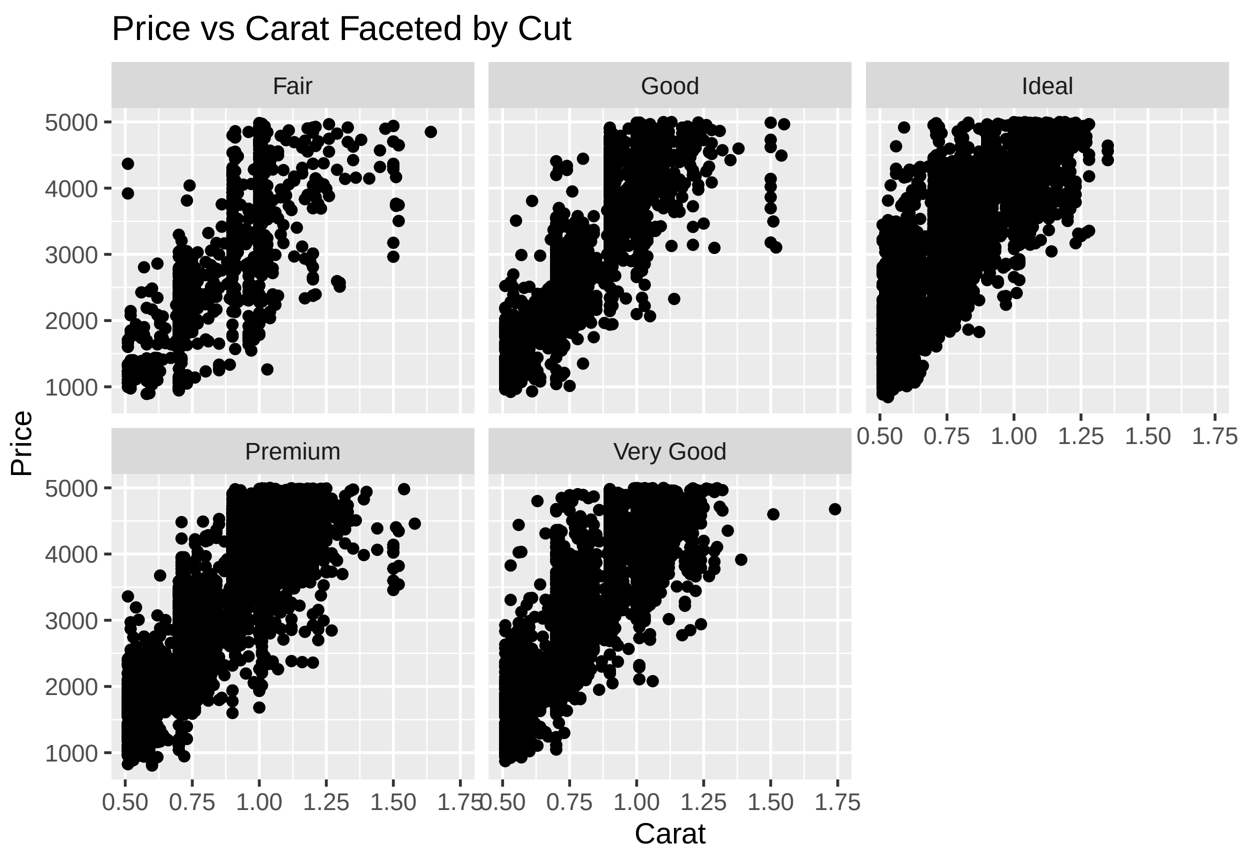
```
##    carat       cut color clarity depth table price    x    y    z
## 91  0.70     Ideal     E     SI1  62.5    57  2757 5.70 5.72 3.57
## 92  0.86      Fair     E     SI2  55.1    69  2757 6.45 6.33 3.52
## 93  0.70     Ideal     G     VS2  61.6    56  2757 5.70 5.67 3.50
## 94  0.71 Very Good     E     VS2  62.4    57  2759 5.68 5.73 3.56
## 95  0.78 Very Good     G     SI2  63.8    56  2759 5.81 5.85 3.72
## 96  0.70      Good     E     VS2  57.5    58  2759 5.85 5.90 3.38
```

```
# b. Use ggplot and study the relationship between the two
# variables of price and carat from part (a) data. Explain
# which graphical display should be used and why.
ggplot(filtered_data, aes(x = carat, y = price,)) +
  geom_point() +
  ggtitle("Price vs Carat (Filtered)") +
  xlab("Carat") +
  ylab("Price")
```

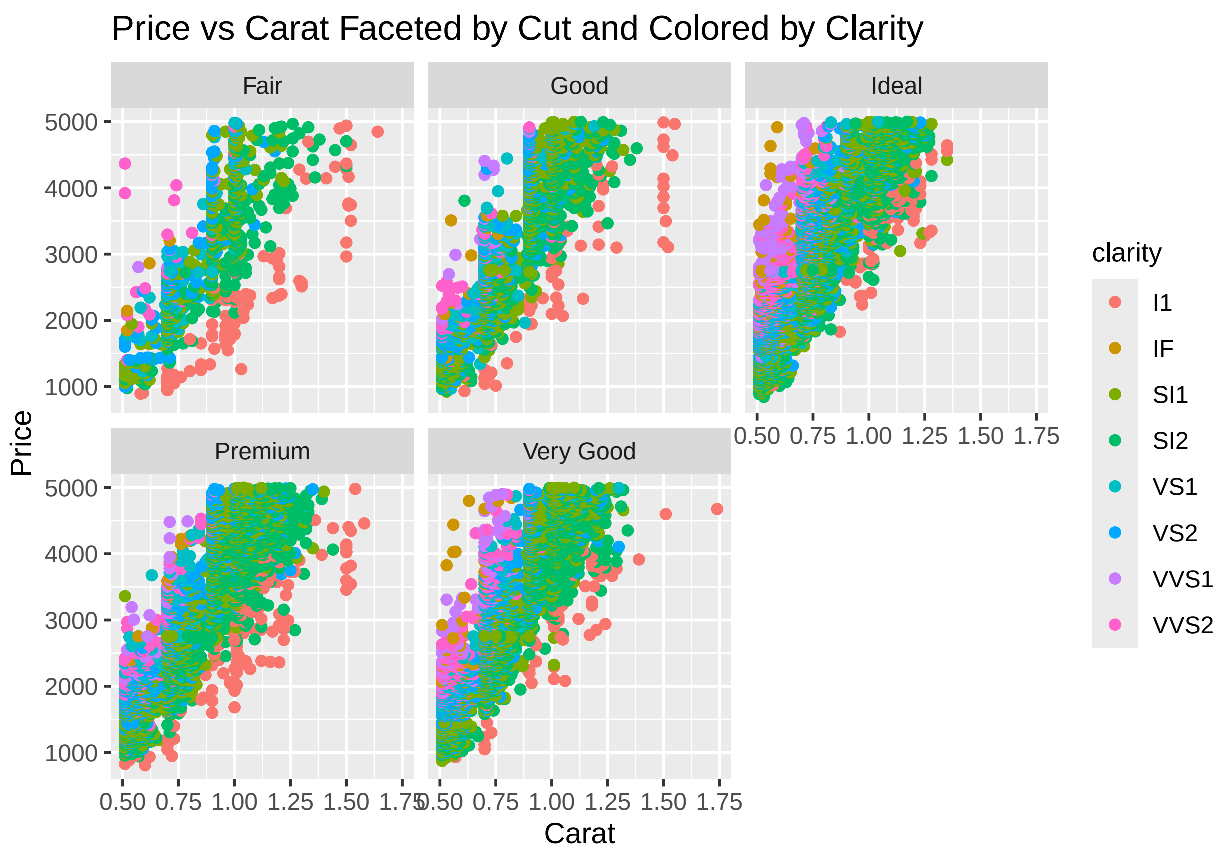



```r
# The scatterplot is used here because it effectively shows
# the relationship between two continuous variables price and carat.
```

```r
# c. Facet the plot by `cut` so that each facet corresponds
# to a different diamond cut.

# Facet by cut
ggplot(filtered_data, aes(x = carat, y = price)) +
  geom_point() +
  facet_wrap(~ cut) +
  ggtitle("Price vs Carat Faceted by Cut") +
  xlab("Carat") +
  ylab("Price")
```




```r
# d) Color the points by 'clarity'. Carefully add the labels
# for the x-axis, y-axis, title and legend.
ggplot(filtered_data, aes(x = carat, y = price, color = clarity)) +
  geom_point() +
  facet_wrap(~ cut) +
  ggtitle("Price vs Carat Faceted by Cut and Colored by Clarity") +
  xlab("Carat") +
  ylab("Price") +
  theme(legend.title = element_text(size = 10))
```



Question 5

```
import pandas as pd
import matplotlib.pyplot as plt

# a. Load the dataset as a data frame. Convert the ‘Date‘
# column to a datetime object and extract the year.

data = pd.read_csv('sales.csv')
data.head()
```

```
##        Date Region  Sales
## 0  2021-01  North  17131
## 1  2021-02  North  10819
## 2  2021-03  North   5597
## 3  2021-04  North  10350
## 4  2021-05  North  12136
```

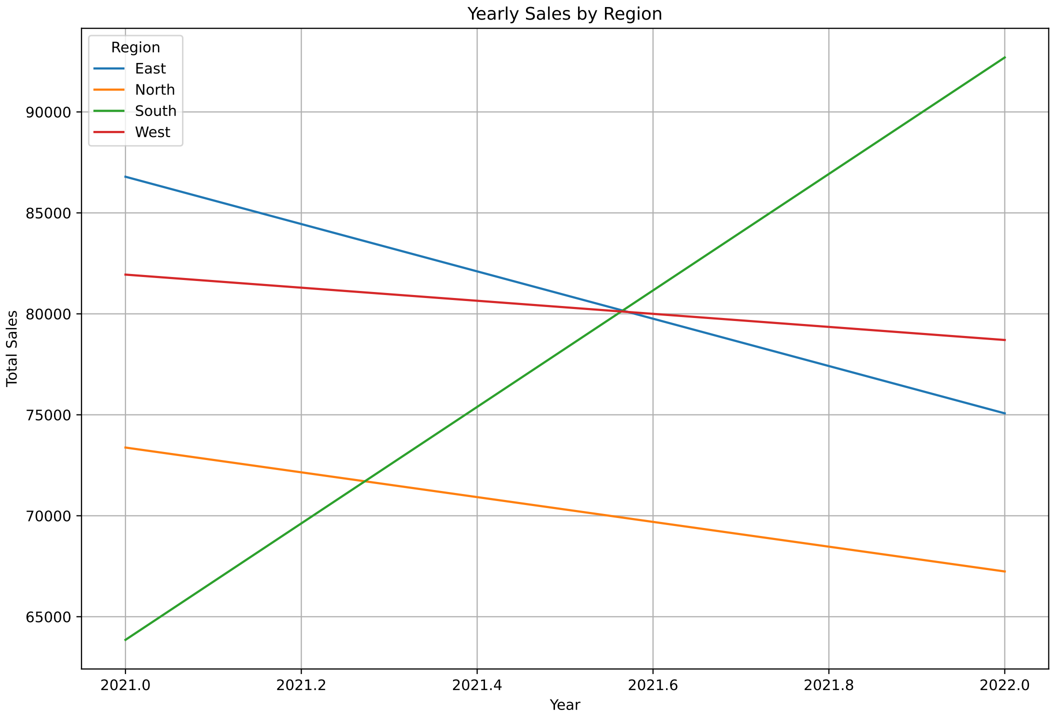
```
data['Date'] = pd.to_datetime(data['Date'])
data['Year'] = data['Date'].dt.year.astype(int)


data.head()
```

```
##         Date Region  Sales  Year
## 0 2021-01-01  North  17131  2021
## 1 2021-02-01  North  10819  2021
## 2 2021-03-01  North   5597  2021
## 3 2021-04-01  North  10350  2021
## 4 2021-05-01  North  12136  2021
```

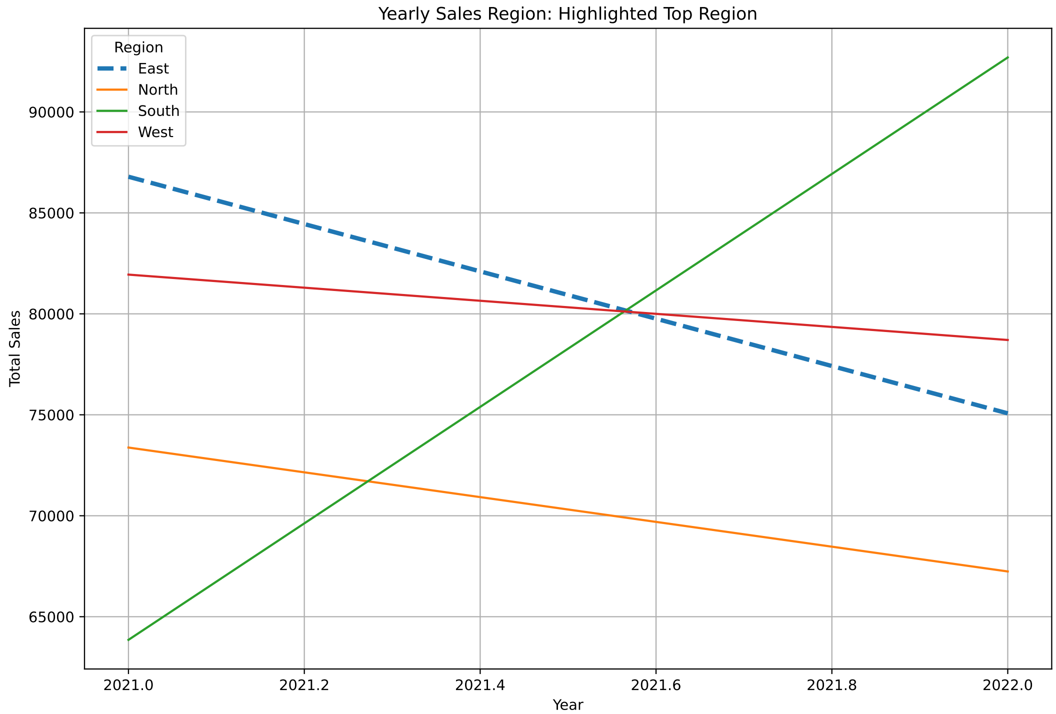
```python
# b. Group the data by ‘Region‘ and calculate the total sales for each year.
sales_region = data.groupby(['Region', 'Year'])['Sales'].sum().reset_index()
sales_region
```

```
##   Region  Year  Sales
## 0   East  2021  86791
## 1   East  2022  75072
## 2  North  2021  73378
## 3  North  2022  67241
## 4  South  2021  63853
## 5  South  2022  92698
## 6   West  2021  81943
## 7   West  2022  78705
```

```python
# c. Create a line plot to show the yearly sales trend for each region.
# Then, add appropriate labels for the x-axis and y-axis and a title for the plot.
plt.figure(figsize=(12, 8))
regions = sales_region['Region'].unique()

for region in regions:
    region_data = sales_region[sales_region['Region'] == region]
    plt.plot(region_data['Year'], region_data['Sales'], label=region)

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.title('Yearly Sales by Region')
plt.legend(title='Region')
plt.grid()
plt.show()
```




```
# d. Highlight the region with the highest total sales across all years in the plot using a thicker lin

# Identify the region with the highest total sales
region_total_sales = sales_region.groupby('Region')['Sales'].sum()
highest_sales_region = region_total_sales.idxmax()

# Re-plot the sales trend with highlighted region
plt.figure(figsize=(12, 8))

for region in regions:
    region_data = sales_region[sales_region['Region'] == region]
    if region == highest_sales_region:
        plt.plot(region_data['Year'], region_data['Sales'], label=region, linewidth=3, linestyle='--')
    else:
        plt.plot(region_data['Year'], region_data['Sales'], label=region)

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Total Sales')
plt.title('Yearly Sales Region: Highlighted Top Region')
plt.legend(title='Region')
plt.grid()
plt.show()
```



Yearly Sales Region: Highlighted Top Region
Region
East
North
South
West
90000
85000
80000
75000
70000
65000
Total Sales
2021.0
2021.2
2021.4
2021.6
2021.8
2022.0
Year