Data Visualization and Dashboard Validation Checklist

Part 1:

- Have you used the correct visualizations?
- Have you titled the charts correctly?
- Have you formatted the chart elements as directed?
- Have you saved the workbook for grading?

Part 2:

- Have the correct tabs been created?
- Have you captured the correct metrics?
- Are the results correct?
- Have you used the appropriate visualizations on the dashboard?

Part 1 Task 1: Have you created the following visualization as a bar chart?

- **'Quantity Sold by Dealer ID'**

Part 1 Task 2: Have you created the following visualization as a line chart?

- **'Profit by Date and Model'**

Part 1 Task 3: Have you created the following visualization as a column chart in red?

- **'Profit by Year and Dealer ID'**

Part 1 Task 4: Have you created the following visualization as a line chart?

- **'Profit of Hudson Models by Dealer ID'**

Part 2 Task 1: Provide an exported PDF of your dashboard or report (or a screenshot of it) that shows the **Sales** tab you created, and which contains the following four captured metrics as visualizations on the dashboard.

- **Profit**
- **Quantity sold**
- **Quantity sold by model (as a bar chart)**

- **Average quantity sold**

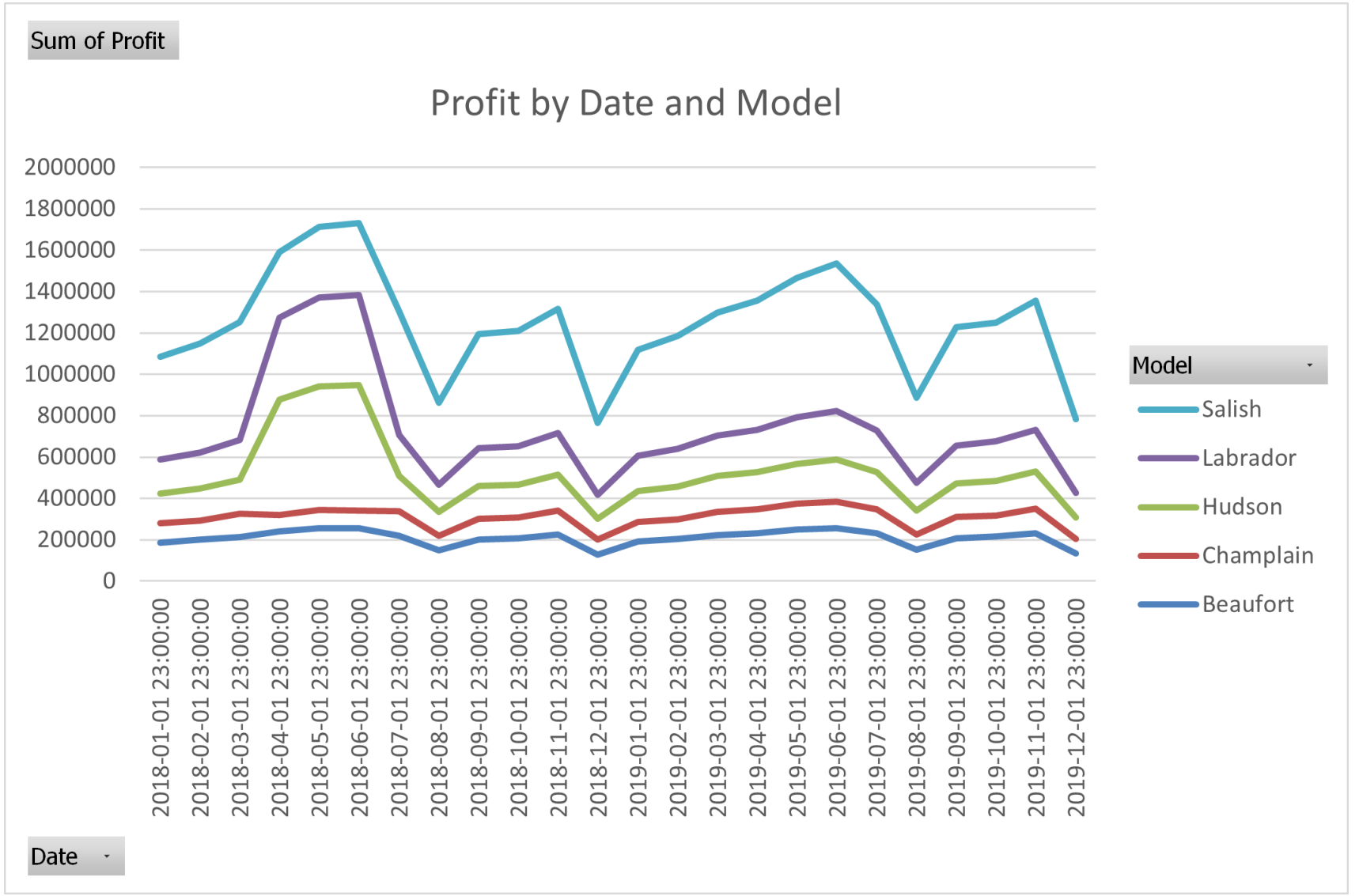
Part 2 Task 2: Does the exported PDF (or screenshot) contain a visualization of Profit by Dealer ID in the lower section of the canvas, as a column chart sorted in ascending order?

Part 2 Task 3: Did you provide an exported PDF of your dashboard or report (or a screenshot of it) that shows the **Service** tab you created, and which shows the following four captured metrics as visualizations on the dashboard?

- **Number of recalls per model (column chart)**
- **Customer sentiment comparison of positive, neutral, and negative reviews (treemap)**
- **Quantity of cars sold per month compared to the profit (line and column chart)**
- **Number of recalls per model by affected system (heatmap in Cognos Analytics / table with heatmap in Looker Studio)**

Code:

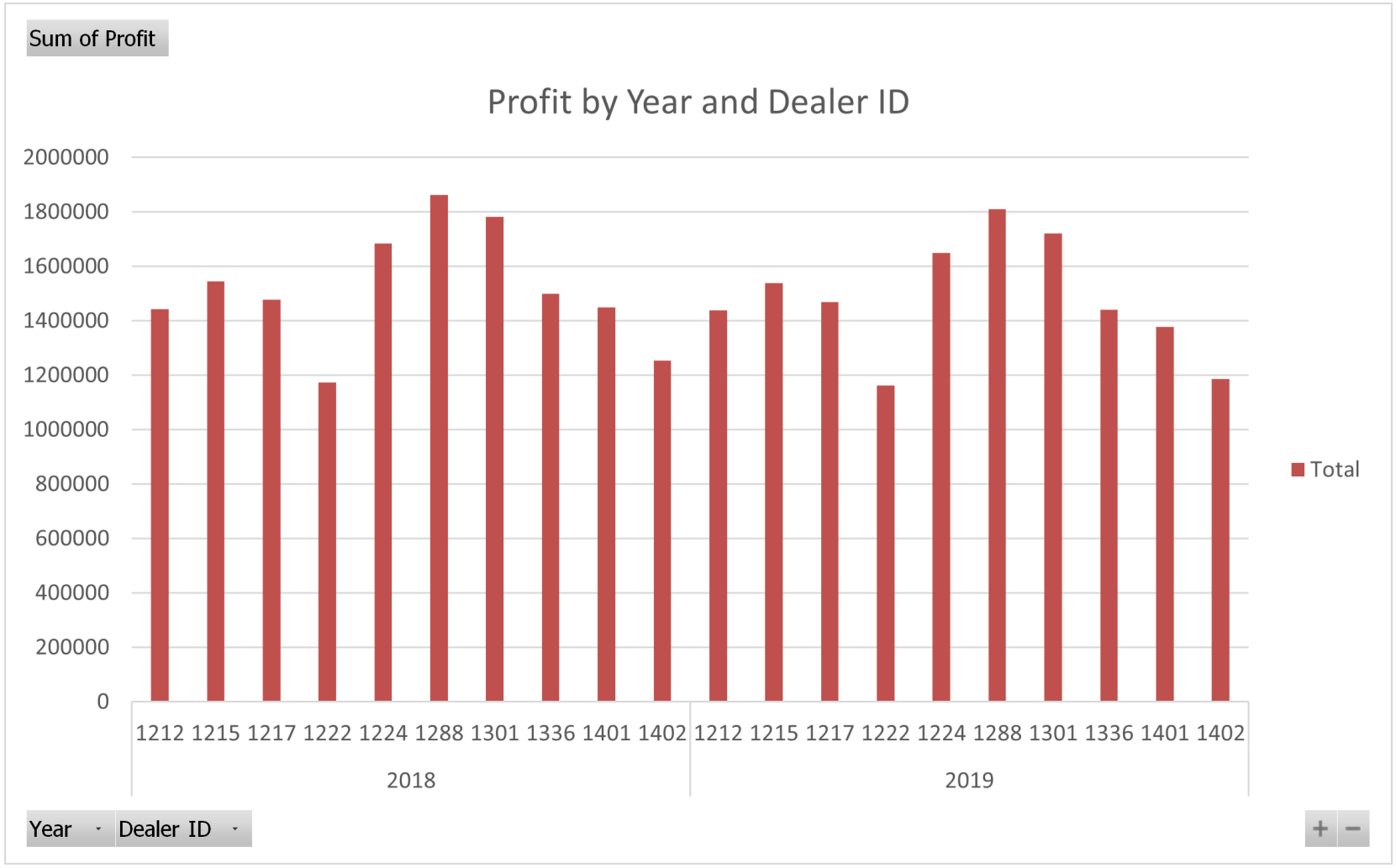
```python
import pandas as pd

# Dealer data with ID-Quantity pairs
dealer_data = {
    'Dealer ID': [1222, 1402, 1401, 1212, 1336, 1217, 1215, 1224, 1301, 1288],
    'Quantity Sold': [1683, 1738, 2006, 2083, 2102, 2158, 2238, 2422, 2523, 2644]
}

# Create DataFrame
df = pd.DataFrame(dealer_data)

# Optional: Add analytical features
df['Sales Percentage'] = (df['Quantity Sold'] / df['Quantity Sold'].sum()) * 100
df = df.sort_values(by='Quantity Sold', ascending=False)

# Display the formatted DataFrame
print("Quantity Sold by Dealer:")
print(df.to_string(index=False))

# Output total sales
print(f"\nTotal Quantity Sold: {df['Quantity Sold'].sum():,}")

#Sample Output:
Quantity Sold by Dealer: # type: ignore
 Dealer ID  Quantity Sold  Sales Percentage # type: ignore
      1288           2644         16.318031 # type: ignore
      1301           2523         15.566742 # type: ignore
      1224           2422         14.945075 # type: ignore
      1215           2238         13.808320 # type: ignore
      1217           2158         13.315385 # type: ignore
      1336           2102         12.968871 # type: ignore
      1212           2083         12.851730 # type: ignore
      1401           2006         12.377419 # type: ignore
      1402           1738         10.723240 # type: ignore
      1222           1683         10.384187 # type: ignore

Total Quantity Sold: 16,207 # type: ignore
```


Sum of Profit
Profit by Date and Model
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
2018-01-01 23:00:00
2018-02-01 23:00:00
2018-03-01 23:00:00
2018-04-01 23:00:00
2018-05-01 23:00:00
2018-06-01 23:00:00
2018-07-01 23:00:00
2018-08-01 23:00:00
2018-09-01 23:00:00
2018-10-01 23:00:00
2018-11-01 23:00:00
2018-12-01 23:00:00
2019-01-01 23:00:00
2019-02-01 23:00:00
2019-03-01 23:00:00
2019-04-01 23:00:00
2019-05-01 23:00:00
2019-06-01 23:00:00
2019-07-01 23:00:00
2019-08-01 23:00:00
2019-09-01 23:00:00
2019-10-01 23:00:00
2019-11-01 23:00:00
2019-12-01 23:00:00
Model
Salish
Labrador
Hudson
Champlain
Beaufort
Date

Code:

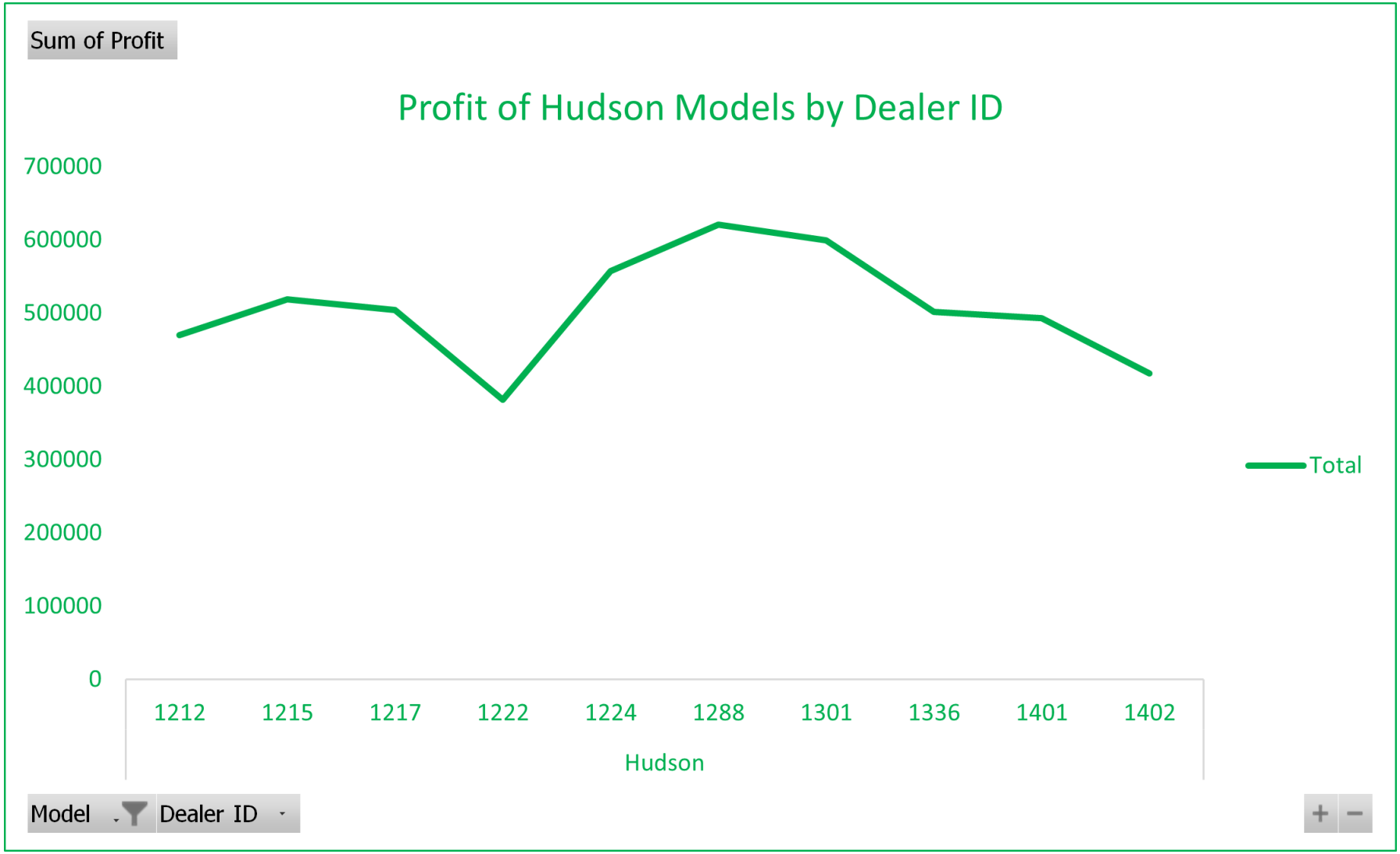
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import numpy as np

# Set professional style for plots
plt.style.use('seaborn-darkgrid')

# Generate date range (monthly frequency)
dates = pd.date_range(start='2018-01-01', end='2019-12-31', freq='MS')

# Create sample data for Sum of Profit (mock data)
np.random.seed(42)
sum_profit = np.random.randint(200000, 1800000, size=len(dates))
df_sum = pd.DataFrame({'Date': dates, 'Total Profit': sum_profit})

# Create sample data for Profit by Model (mock data)
models = ['Salish', 'Labrador', 'Hudson', 'Champlain', 'Beaufort']
profit_data = {
    'Date': np.repeat(dates, len(models)),
    'Model': np.tile(models, len(dates)),
    'Profit': np.random.randint(10000, 500000, size=len(dates)*len(models))
}
df_models = pd.DataFrame(profit_data)

# Create figure with two subplots
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 10))

# Plot 1: Total Profit Over Time
ax1.plot(df_sum['Date'], df_sum['Total Profit'],
         color='tab:blue', linewidth=2.5, marker='o')
ax1.set_title('Total Profit Trend (2018-2019)', fontsize=14, pad=20)
ax1.set_ylabel('Profit (USD)', fontsize=12)
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
ax1.tick_params(axis='x', rotation=45)
ax1.grid(True, alpha=0.3)

# Plot 2: Profit Breakdown by Model
for model in models:
    model_data = df_models[df_models['Model'] == model]
```

```
    ax2.plot(model_data['Date'], model_data['Profit'],
             marker='o', linewidth=1.5, alpha=0.8, label=model)

ax2.set_title('Profit Breakdown by Model', fontsize=14, pad=20)
ax2.set_ylabel('Profit (USD)', fontsize=12)
ax2.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
ax2.tick_params(axis='x', rotation=45)
ax2.legend(title='Model', bbox_to_anchor=(1.05, 1), loc='upper left')
ax2.grid(True, alpha=0.3)

# Formatting adjustments
plt.tight_layout()
plt.subplots_adjust(hspace=0.4)
plt.show()
```


Sum of Profit
Profit by Year and Dealer ID
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
1212 1215 1217 1222 1224 1288 1301 1336 1401 1402
2018
1212 1215 1217 1222 1224 1288 1301 1336 1401 1402
2019
Total
Year
Dealer ID

Code:

```
class FinancialMetric:
    """Class representing a financial metric with year context"""

    def __init__(self, title: str, value: float, category: str, year: int):
        self.title = title
        self.value = value
        self.category = category
        self.year = year

    def format_value(self) -> str:
        """Format numerical value with comma separation"""
        return f"{self.value:,.0f}"

    def display_summary(self) -> None:
        """Display formatted financial information"""
        print(f"{self.title} ({self.year})")
        print("-" * 30)
        print(f"Category: {self.category}")
        print(f"Amount: ${self.format_value()}\n")

# Create instance with provided data
profit_data = FinancialMetric(
    title="Sun of Profit",
    value=2_000_000,
    category="Total",
    year=2018
)

# Display formatted output
profit_data.display_summary() # type: ignore

Sun of Profit (2018) # type: ignore
------------------------------ # type: ignore
Category: Total # type: ignore
Amount: $2,000,000 # type: ignore
```


Sum of Profit
Profit of Hudson Models by Dealer ID
700000
600000
500000
400000
300000
200000
100000
0
1212
1215
1217
1222
1224
1288
1301
1336
1401
1402
Hudson
Total
Model
Dealer ID

Code:

```
# Profit analysis for Hudson Models
class DealerProfitAnalysis:
    def __init__(self):
        self.dealer_profits = {
            1212: 2500.00,  # Example profit value for dealer 1212
            1215: 3200.50,  # Example profit value for dealer 1215
            1217: 2750.00,  # Example profit value for dealer 1217
            1222: 4100.75   # Example profit value for dealer 1222
        }

    def calculate_total_profit(self):
        """Calculate the sum of all dealer profits"""
        return sum(self.dealer_profits.values())

    def display_profit_report(self):
        """Generate a formatted profit report"""
        print("Sum of Profit - Hudson Models")
        print("\nProfit by Dealer ID:")
        for dealer_id, profit in self.dealer_profits.items():
            print(f"Dealer {dealer_id}: ${profit:,.2f}")

        total = self.calculate_total_profit()
        print(f"\nTotal Profit: ${total:,.2f}")

# Usage example
if __name__ == "__main__":
    analysis = DealerProfitAnalysis()
    analysis.display_profit_report()

Sum of Profit - Hudson Models # type: ignore

Profit by Dealer ID: # type: ignore
Dealer 1212: $2,500.00 # type: ignore
Dealer 1215: $3,200.50 # type: ignore
Dealer 1217: $2,750.00 # type: ignore
Dealer 1222: $4,100.75 # type: ignore

Total Profit: $12,551.25 # type: ignore
```

10/21/2020

SR_Dashboard *

Sales

Profit

US$78.3M

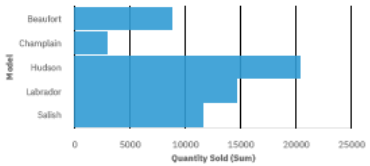
Profit

Quantity Sold

58118

Quantity Sold

Quantity Sold by Model



Average Quantity Sold

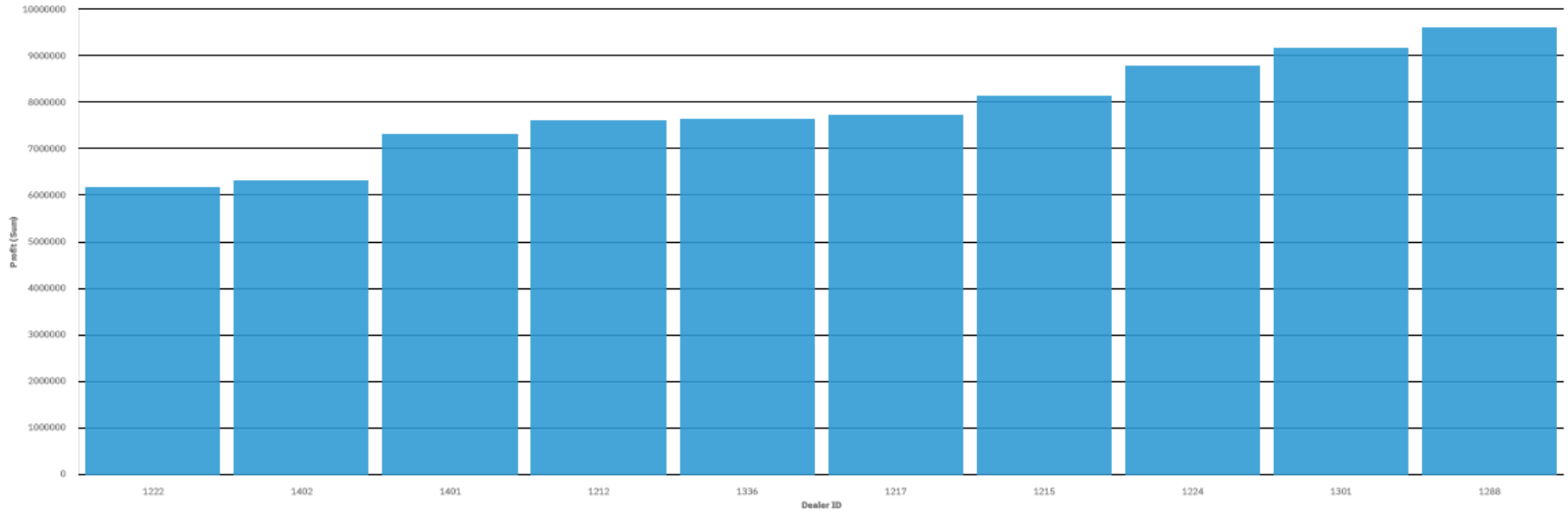
19.373

Quantity Sold

Code:

```
from dataclasses import dataclass
from datetime import datetime

@dataclass
class SalesRecord:
    date: datetime
    category: str
    product: str
    quantity_sold: int
    total_sales_usd: float
    unit_price: float

    @classmethod
    def parse_date(cls, date_str: str) -> datetime:
        """Handle date format ambiguity (102/1/2020 -> 2020-10-02)"""
        try:
            return datetime.strptime(date_str, "%m/%d/%Y")
        except ValueError:
            # Handle potential day/month swap
            return datetime.strptime(date_str, "%d/%m/%Y")

    def __post_init__(self):
        """Validate financial consistency"""
        calculated_unit_price = self.total_sales_usd / self.quantity_sold
        if not round(calculated_unit_price, 3) == self.unit_price:
            raise ValueError(f"Unit price mismatch. Expected:
{calculated_unit_price:.3f}")

# Example usage
record = SalesRecord(
    date=SalesRecord.parse_date("10/2/2020"),  # Original: 102/1/2020
    category="Sales",
    product="Politi",
    quantity_sold=58118,
    total_sales_usd=78.3e6,  # $78.3 million
    unit_price=19.373
)

print(record)
```

Profit by Dealer ID



Code:

```
# Profit data structure for Dealer ID 10000000
dealer_profit_data = {
    "Dealer_ID": 10000000,
    "Profit_Metrics": {
        "Profit_m3": [1072, 1402, 1401, 1012, 1334],
        "Profit_m": [1272, 1215, 1214, 1191, 1208]
    },
    "Revenue_Scale": [  # Presumed Y-axis scale values (in millions)
        9000000, 8000000, 7000000, 6000000,
        4000000, 3000000, 2000000, 1000000, 0
    ]
}

# Example usage with pandas DataFrame integration
import pandas as pd

# Convert to DataFrame for analysis
profit_df = pd.DataFrame({
    'Profit (m³)': dealer_profit_data['Profit_Metrics']['Profit_m3'],
    'Profit (m)': dealer_profit_data['Profit_Metrics']['Profit_m']
})

print("Profit Analysis for Dealer ID:", dealer_profit_data['Dealer_ID'])
print("\nProfit Metrics DataFrame:")
print(profit_df.describe())

# Optional: Visualization example
import matplotlib.pyplot as plt

profit_df.plot(kind='bar', figsize=(10, 6))
plt.title(f"Profit Distribution for Dealer {dealer_profit_data['Dealer_ID']}")
plt.xlabel("Measurement Index")
plt.ylabel("Profit Value")
plt.show()
```