Manufacturing-Line Productivity Analysis

28/12/24

This is a manufacturing-Line Productivity Analysis for a bottling company. Some of the steps i've taken to clean the data include;

- first i imported my libraries.
- i imported the dataset.
- i made sure there were no null(empty) values.
- i made sure there were no duplicates.
- i calculated the processing time.

After cleaning and optimizing the dataset for interpreation.

- i started my analysis, identifying key insights, like; group analyis, daily productivity, etc.
- lastly i created the visualizations for my insights, adding more insights as i progressed.

The company has 4 operators, and their products with their flavours include:

- Orange 600ml (OR-600)
- Lemon Lime 600ml (LE-600)
- Cola 600ml (CO-600)
- Diet Cola 600ml (DC-600)
- Root Beer 600ml (RB-600)
- Cola 2L (CO-2L)

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import openpyxl
import numpy as np
```

In [2]:
```
df = pd.read_excel(r'C:\Users\Isaac\Documents\Data Analysis\Manufacturing Line\Manu
```

In [3]:
```
df
```

Out[3]:

	Date	Product	Batch	Operator	Start Time	End Time
0	2024-08-29	OR-600	422111	Mac	11:50:00	14:05:00
1	2024-08-29	LE-600	422112	Mac	14:05:00	15:45:00
2	2024-08-29	LE-600	422113	Mac	15:45:00	17:35:00
3	2024-08-29	LE-600	422114	Mac	17:35:00	19:15:00
4	2024-08-29	LE-600	422115	Charlie	19:15:00	20:39:00
5	2024-08-29	LE-600	422116	Charlie	20:39:00	21:39:00
6	2024-08-29	LE-600	422117	Charlie	21:39:00	22:54:00
7	2024-08-30	CO-600	422118	Dee	04:05:00	06:05:00
8	2024-08-30	CO-600	422119	Dee	06:05:00	07:30:00
9	2024-08-30	CO-600	422120	Dee	07:30:00	09:22:00
10	2024-08-30	CO-600	422121	Dennis	09:22:00	10:37:00
11	2024-08-30	CO-600	422122	Dennis	10:37:00	12:02:00
12	2024-08-30	CO-600	422123	Dennis	12:02:00	14:15:00
13	2024-08-30	CO-600	422124	Dennis	14:15:00	15:55:00
14	2024-08-30	CO-600	422125	Charlie	15:55:00	17:15:00
15	2024-08-30	CO-600	422126	Charlie	17:15:00	18:59:00
16	2024-08-30	CO-600	422127	Charlie	18:59:00	20:22:00
17	2024-08-30	CO-600	422128	Charlie	20:22:00	22:14:00
18	2024-08-30	CO-600	422129	Charlie	22:14:00	23:29:00
19	2024-08-31	CO-600	422130	Dee	07:45:00	09:05:00
20	2024-08-31	CO-600	422131	Dee	09:05:00	10:35:00
21	2024-08-31	CO-600	422132	Dee	10:35:00	11:35:00
22	2024-08-31	DC-600	422133	Dee	11:35:00	12:55:00
23	2024-08-31	DC-600	422134	Mac	12:55:00	14:45:00
24	2024-08-31	DC-600	422135	Mac	14:45:00	16:30:00
25	2024-08-31	DC-600	422136	Mac	16:30:00	17:30:00
26	2024-09-02	RB-600	422137	Dee	01:00:00	02:45:00
27	2024-09-02	RB-600	422138	Dee	02:45:00	04:05:00
28	2024-09-02	RB-600	422139	Dee	04:05:00	05:40:00
29	2024-09-02	RB-600	422140	Dee	05:40:00	07:43:00

	Date	Product	Batch	Operator	Start Time	End Time
30	2024-09-02	RB-600	422141	Dennis	07:43:00	08:50:00
31	2024-09-02	RB-600	422142	Dennis	08:50:00	10:20:00
32	2024-09-02	RB-600	422143	Dennis	10:20:00	12:18:00
33	2024-09-02	CO-2L	422144	Dennis	12:18:00	14:50:00
34	2024-09-02	CO-2L	422145	Charlie	14:50:00	16:50:00
35	2024-09-02	CO-2L	422146	Charlie	16:50:00	19:30:00
36	2024-09-02	CO-2L	422147	Charlie	19:30:00	22:55:00
37	2024-09-03	CO-2L	422148	Mac	22:55:00	1900-01-01 01:05:00

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Date          0
Product       0
Batch         0
Operator      0
Start Time    0
End Time      0
dtype: int64
```

In [5]:
```
df.duplicated()
```

Out[5]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
33    False
34    False
35    False
36    False
37    False
dtype: bool
```

In [6]:
```
df['Start Time'] = pd.to_datetime(df['Start Time'],format='%H:%M:%S')
df['End Time'] = pd.to_datetime(df['End Time'],format='%H:%M:%S')
```

In [7]:
```
df
```

Out[7]:

	Date	Product	Batch	Operator	Start Time	End Time
0	2024-08-29	OR-600	422111	Mac	1900-01-01 11:50:00	1900-01-01 14:05:00
1	2024-08-29	LE-600	422112	Mac	1900-01-01 14:05:00	1900-01-01 15:45:00
2	2024-08-29	LE-600	422113	Mac	1900-01-01 15:45:00	1900-01-01 17:35:00
3	2024-08-29	LE-600	422114	Mac	1900-01-01 17:35:00	1900-01-01 19:15:00
4	2024-08-29	LE-600	422115	Charlie	1900-01-01 19:15:00	1900-01-01 20:39:00
5	2024-08-29	LE-600	422116	Charlie	1900-01-01 20:39:00	1900-01-01 21:39:00
6	2024-08-29	LE-600	422117	Charlie	1900-01-01 21:39:00	1900-01-01 22:54:00
7	2024-08-30	CO-600	422118	Dee	1900-01-01 04:05:00	1900-01-01 06:05:00
8	2024-08-30	CO-600	422119	Dee	1900-01-01 06:05:00	1900-01-01 07:30:00
9	2024-08-30	CO-600	422120	Dee	1900-01-01 07:30:00	1900-01-01 09:22:00
10	2024-08-30	CO-600	422121	Dennis	1900-01-01 09:22:00	1900-01-01 10:37:00
11	2024-08-30	CO-600	422122	Dennis	1900-01-01 10:37:00	1900-01-01 12:02:00
12	2024-08-30	CO-600	422123	Dennis	1900-01-01 12:02:00	1900-01-01 14:15:00
13	2024-08-30	CO-600	422124	Dennis	1900-01-01 14:15:00	1900-01-01 15:55:00
14	2024-08-30	CO-600	422125	Charlie	1900-01-01 15:55:00	1900-01-01 17:15:00
15	2024-08-30	CO-600	422126	Charlie	1900-01-01 17:15:00	1900-01-01 18:59:00
16	2024-08-30	CO-600	422127	Charlie	1900-01-01 18:59:00	1900-01-01 20:22:00
17	2024-08-30	CO-600	422128	Charlie	1900-01-01 20:22:00	1900-01-01 22:14:00
18	2024-08-30	CO-600	422129	Charlie	1900-01-01 22:14:00	1900-01-01 23:29:00
19	2024-08-31	CO-600	422130	Dee	1900-01-01 07:45:00	1900-01-01 09:05:00
20	2024-08-31	CO-600	422131	Dee	1900-01-01 09:05:00	1900-01-01 10:35:00
21	2024-08-31	CO-600	422132	Dee	1900-01-01 10:35:00	1900-01-01 11:35:00
22	2024-08-31	DC-600	422133	Dee	1900-01-01 11:35:00	1900-01-01 12:55:00
23	2024-08-31	DC-600	422134	Mac	1900-01-01 12:55:00	1900-01-01 14:45:00
24	2024-08-31	DC-600	422135	Mac	1900-01-01 14:45:00	1900-01-01 16:30:00
25	2024-08-31	DC-600	422136	Mac	1900-01-01 16:30:00	1900-01-01 17:30:00
26	2024-09-02	RB-600	422137	Dee	1900-01-01 01:00:00	1900-01-01 02:45:00
27	2024-09-02	RB-600	422138	Dee	1900-01-01 02:45:00	1900-01-01 04:05:00
28	2024-09-02	RB-600	422139	Dee	1900-01-01 04:05:00	1900-01-01 05:40:00
29	2024-09-02	RB-600	422140	Dee	1900-01-01 05:40:00	1900-01-01 07:43:00

	Date	Product	Batch	Operator	Start Time	End Time
30	2024-09-02	RB-600	422141	Dennis	1900-01-01 07:43:00	1900-01-01 08:50:00
31	2024-09-02	RB-600	422142	Dennis	1900-01-01 08:50:00	1900-01-01 10:20:00
32	2024-09-02	RB-600	422143	Dennis	1900-01-01 10:20:00	1900-01-01 12:18:00
33	2024-09-02	CO-2L	422144	Dennis	1900-01-01 12:18:00	1900-01-01 14:50:00
34	2024-09-02	CO-2L	422145	Charlie	1900-01-01 14:50:00	1900-01-01 16:50:00
35	2024-09-02	CO-2L	422146	Charlie	1900-01-01 16:50:00	1900-01-01 19:30:00
36	2024-09-02	CO-2L	422147	Charlie	1900-01-01 19:30:00	1900-01-01 22:55:00
37	2024-09-03	CO-2L	422148	Mac	1900-01-01 22:55:00	1900-01-01 01:05:00

Calculating Processing Time

In [8]:
```
df['Start Time'] = pd.to_datetime(df['Start Time'])
df['End Time'] = pd.to_datetime(df['End Time'])
```

In [9]:
```
df['Processing Time'] = (df['End Time'] - df['Start Time']).dt.total_seconds()/3600

print(df[['Date','Product','Processing Time']])
```

	Date	Product	Processing Time
0	2024-08-29	OR-600	2.250000
1	2024-08-29	LE-600	1.666667
2	2024-08-29	LE-600	1.833333
3	2024-08-29	LE-600	1.666667
4	2024-08-29	LE-600	1.400000
5	2024-08-29	LE-600	1.000000
6	2024-08-29	LE-600	1.250000
7	2024-08-30	CO-600	2.000000
8	2024-08-30	CO-600	1.416667
9	2024-08-30	CO-600	1.866667
10	2024-08-30	CO-600	1.250000
11	2024-08-30	CO-600	1.416667
12	2024-08-30	CO-600	2.216667
13	2024-08-30	CO-600	1.666667
14	2024-08-30	CO-600	1.333333
15	2024-08-30	CO-600	1.733333
16	2024-08-30	CO-600	1.383333
17	2024-08-30	CO-600	1.866667
18	2024-08-30	CO-600	1.250000
19	2024-08-31	CO-600	1.333333
20	2024-08-31	CO-600	1.500000
21	2024-08-31	CO-600	1.000000
22	2024-08-31	DC-600	1.333333
23	2024-08-31	DC-600	1.833333
24	2024-08-31	DC-600	1.750000
25	2024-08-31	DC-600	1.000000
26	2024-09-02	RB-600	1.750000
27	2024-09-02	RB-600	1.333333
28	2024-09-02	RB-600	1.583333
29	2024-09-02	RB-600	2.050000
30	2024-09-02	RB-600	1.116667
31	2024-09-02	RB-600	1.500000
32	2024-09-02	RB-600	1.966667
33	2024-09-02	CO-2L	2.533333
34	2024-09-02	CO-2L	2.000000
35	2024-09-02	CO-2L	2.666667
36	2024-09-02	CO-2L	3.416667
37	2024-09-03	CO-2L	-21.833333

In [10]:
```
df
```

Out[10]:

	Date	Product	Batch	Operator	Start Time	End Time	Processing Time
0	2024-08-29	OR-600	422111	Mac	1900-01-01 11:50:00	1900-01-01 14:05:00	2.250000
1	2024-08-29	LE-600	422112	Mac	1900-01-01 14:05:00	1900-01-01 15:45:00	1.666667
2	2024-08-29	LE-600	422113	Mac	1900-01-01 15:45:00	1900-01-01 17:35:00	1.833333
3	2024-08-29	LE-600	422114	Mac	1900-01-01 17:35:00	1900-01-01 19:15:00	1.666667
4	2024-08-29	LE-600	422115	Charlie	1900-01-01 19:15:00	1900-01-01 20:39:00	1.400000
5	2024-08-29	LE-600	422116	Charlie	1900-01-01 20:39:00	1900-01-01 21:39:00	1.000000
6	2024-08-29	LE-600	422117	Charlie	1900-01-01 21:39:00	1900-01-01 22:54:00	1.250000
7	2024-08-30	CO-600	422118	Dee	1900-01-01 04:05:00	1900-01-01 06:05:00	2.000000
8	2024-08-30	CO-600	422119	Dee	1900-01-01 06:05:00	1900-01-01 07:30:00	1.416667
9	2024-08-30	CO-600	422120	Dee	1900-01-01 07:30:00	1900-01-01 09:22:00	1.866667
10	2024-08-30	CO-600	422121	Dennis	1900-01-01 09:22:00	1900-01-01 10:37:00	1.250000
11	2024-08-30	CO-600	422122	Dennis	1900-01-01 10:37:00	1900-01-01 12:02:00	1.416667
12	2024-08-30	CO-600	422123	Dennis	1900-01-01 12:02:00	1900-01-01 14:15:00	2.216667
13	2024-08-30	CO-600	422124	Dennis	1900-01-01 14:15:00	1900-01-01 15:55:00	1.666667
14	2024-08-30	CO-600	422125	Charlie	1900-01-01 15:55:00	1900-01-01 17:15:00	1.333333
15	2024-08-30	CO-600	422126	Charlie	1900-01-01 17:15:00	1900-01-01 18:59:00	1.733333
16	2024-08-30	CO-600	422127	Charlie	1900-01-01 18:59:00	1900-01-01 20:22:00	1.383333
17	2024-08-30	CO-600	422128	Charlie	1900-01-01 20:22:00	1900-01-01 22:14:00	1.866667
18	2024-08-30	CO-600	422129	Charlie	1900-01-01 22:14:00	1900-01-01 23:29:00	1.250000

	Date	Product	Batch	Operator	Start Time	End Time	Processing Time
19	2024-08-31	CO-600	422130	Dee	1900-01-01 07:45:00	1900-01-01 09:05:00	1.333333
20	2024-08-31	CO-600	422131	Dee	1900-01-01 09:05:00	1900-01-01 10:35:00	1.500000
21	2024-08-31	CO-600	422132	Dee	1900-01-01 10:35:00	1900-01-01 11:35:00	1.000000
22	2024-08-31	DC-600	422133	Dee	1900-01-01 11:35:00	1900-01-01 12:55:00	1.333333
23	2024-08-31	DC-600	422134	Mac	1900-01-01 12:55:00	1900-01-01 14:45:00	1.833333
24	2024-08-31	DC-600	422135	Mac	1900-01-01 14:45:00	1900-01-01 16:30:00	1.750000
25	2024-08-31	DC-600	422136	Mac	1900-01-01 16:30:00	1900-01-01 17:30:00	1.000000
26	2024-09-02	RB-600	422137	Dee	1900-01-01 01:00:00	1900-01-01 02:45:00	1.750000
27	2024-09-02	RB-600	422138	Dee	1900-01-01 02:45:00	1900-01-01 04:05:00	1.333333
28	2024-09-02	RB-600	422139	Dee	1900-01-01 04:05:00	1900-01-01 05:40:00	1.583333
29	2024-09-02	RB-600	422140	Dee	1900-01-01 05:40:00	1900-01-01 07:43:00	2.050000
30	2024-09-02	RB-600	422141	Dennis	1900-01-01 07:43:00	1900-01-01 08:50:00	1.116667
31	2024-09-02	RB-600	422142	Dennis	1900-01-01 08:50:00	1900-01-01 10:20:00	1.500000
32	2024-09-02	RB-600	422143	Dennis	1900-01-01 10:20:00	1900-01-01 12:18:00	1.966667
33	2024-09-02	CO-2L	422144	Dennis	1900-01-01 12:18:00	1900-01-01 14:50:00	2.533333
34	2024-09-02	CO-2L	422145	Charlie	1900-01-01 14:50:00	1900-01-01 16:50:00	2.000000
35	2024-09-02	CO-2L	422146	Charlie	1900-01-01 16:50:00	1900-01-01 19:30:00	2.666667
36	2024-09-02	CO-2L	422147	Charlie	1900-01-01 19:30:00	1900-01-01 22:55:00	3.416667
37	2024-09-03	CO-2L	422148	Mac	1900-01-01 22:55:00	1900-01-01 01:05:00	-21.833333

Analysis

Group Analysis

Grouping operators and products to find the total processing time and batch count

In [11]:
```
operator_analysis = df.groupby('Operator')['Processing Time'].sum().reset_index()
print(operator_analysis)
```

```
  Operator  Processing Time
0  Charlie        19.300000
1      Dee        17.166667
2   Dennis        13.666667
3      Mac        -9.833333
```

In [12]:
```
product_batch_count = df.groupby('Product')['Batch'].count().reset_index()
print(product_batch_count)
```

```
  Product  Batch
0   CO-2L      5
1  CO-600     15
2  DC-600      4
3  LE-600      6
4  OR-600      1
5  RB-600      7
```

Daily Productivity

Analyzing productivity per day by grouping by the date

In [13]:
```
daily_productivity = df.groupby('Date')['Processing Time'].sum().reset_index()
print(daily_productivity)
```

```
        Date  Processing Time
0 2024-08-29        11.066667
1 2024-08-30        19.400000
2 2024-08-31         9.750000
3 2024-09-02        21.916667
4 2024-09-03       -21.833333
```

Identifying the most productive operator

In [14]:
```
most_productive_operator = operator_analysis.sort_values('Processing Time', ascendi
print(most_productive_operator)
```

```
Operator           Charlie
Processing Time       19.3
Name: 0, dtype: object
```

Processing time per product

Analyzing the products and their processing time

In [15]:
```
pt_per_product = df.groupby('Product')['Processing Time'].mean().sort_values(ascend
print(pt_per_product)
```

```
  Product  Processing Time
0  OR-600         2.250000
1  RB-600         1.614286
2  CO-600         1.548889
3  DC-600         1.479167
4  LE-600         1.469444
5   CO-2L        -2.243333
```

Aggregate of max & min of processing time

finding the max and min processing time for each operator

In [16]:
```
result = df.groupby('Operator').agg({'Processing Time':['max','min']})
result
```

Out[16]:

	Processing Time	
	max	min
Operator		
Charlie	3.416667	1.000000
Dee	2.050000	1.000000
Dennis	2.533333	1.116667
Mac	2.250000	-21.833333

Visualizations

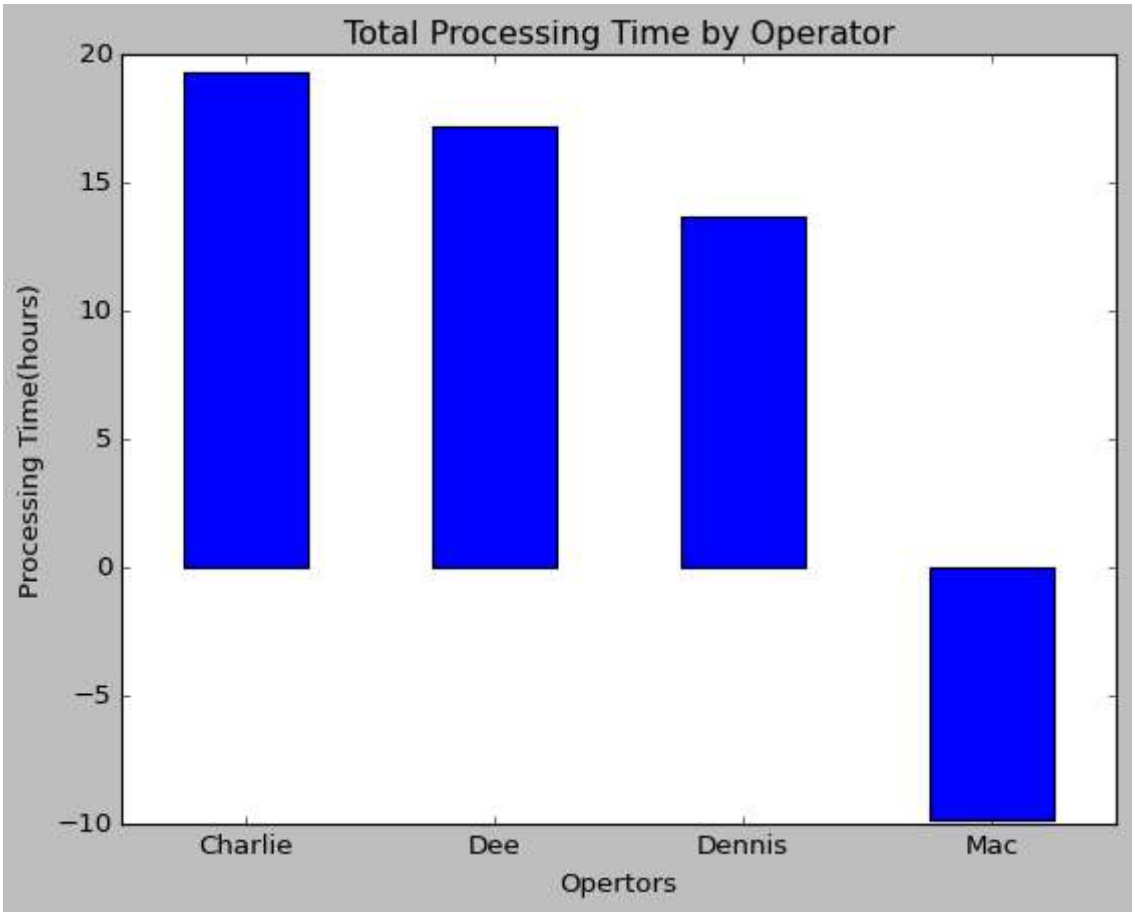
Bar Chart

The visualization indicates Charlie has the most processing time amongst the operators, at nearly 20+ hours, Dee is at 15+ hours, Dennis is at 10+ hours, Mac is less than 10 hours.

In [17]:
```
plt.style.use('classic')
```

In [18]:
```
plt.figure(figsize=(10,6))
operator_analysis.plot(kind='bar',x='Operator',y='Processing Time',legend=False)
plt.title('Total Processing Time by Operator')
plt.ylabel('Processing Time(hours)')
plt.xlabel('Opertors')
plt.xticks(rotation=360)
plt.show()
```

```
<Figure size 800x480 with 0 Axes>
```

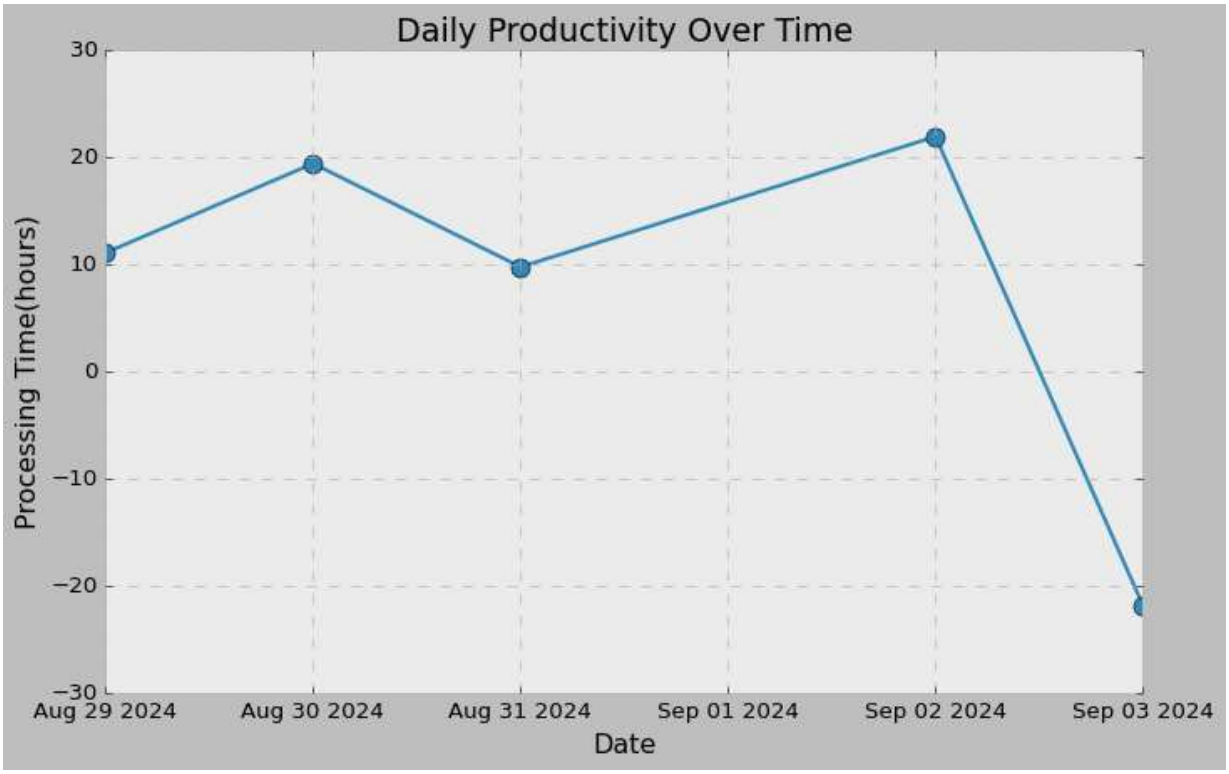


Line Plot

Over time we see that productivity rises from august 29 and it dips in August 31 (it was a saturday), it peaked september 2 which was a monday, there was a drop september 3, this needs to be analyzed futrher.

In [19]:
```
plt.style.use('bmh')
```

In [20]:
```
plt.figure(figsize=(10,6))
plt.plot(daily_productivity['Date'],daily_productivity['Processing Time'], marker='
plt.title('Daily Productivity Over Time')
plt.ylabel('Processing Time(hours)')
plt.xlabel('Date')
plt.xticks(rotation=360)
plt.show()
```



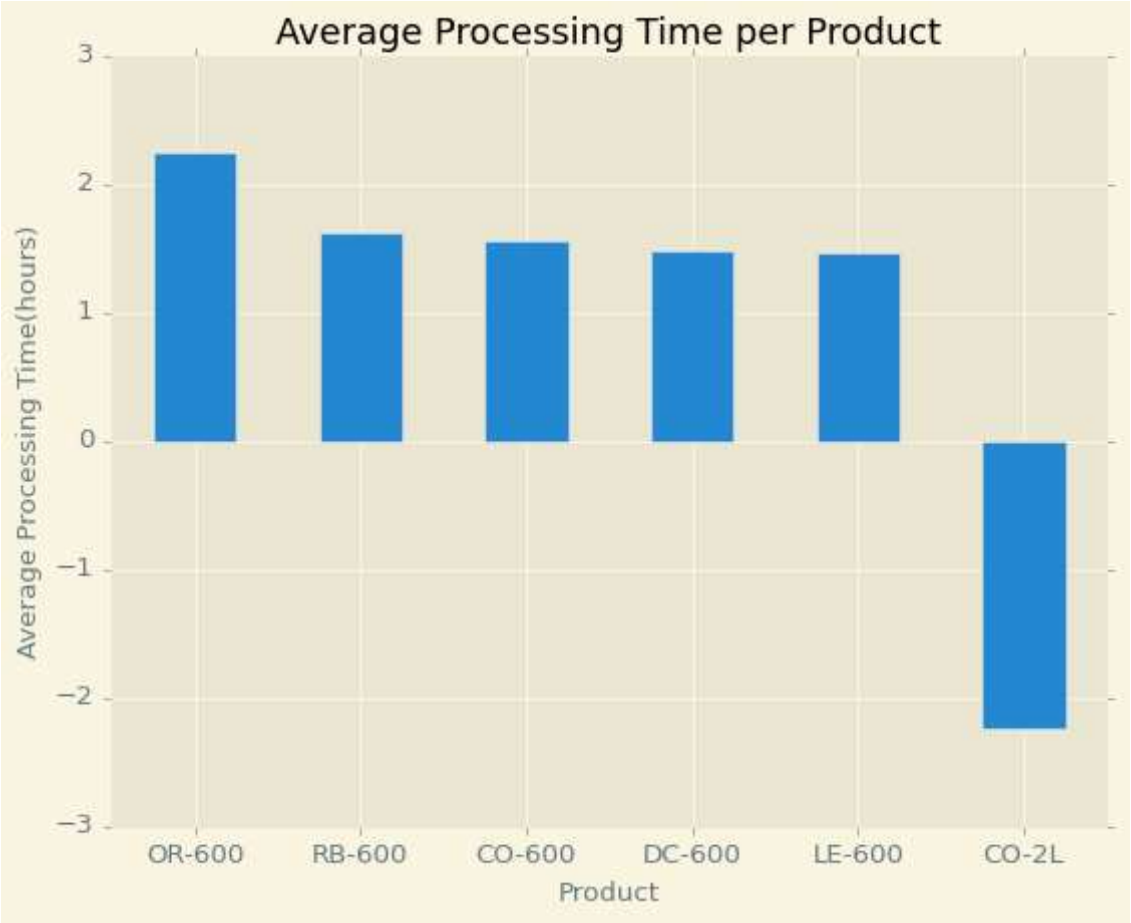
Bar chart

Orange 600ml has the highest processing time, Root Beer 600ml edges Cola 600ml by a bit, Diet Cola 600ml and Lemon Lime 600ml have the same average processing time, while Cola 2L has the lowest average processing time.

In [21]:
```
plt.style.use('Solarize_Light2')
```

In [22]:
```
plt.figure(figsize=(10,6))
pt_per_product.plot(kind='bar',x='Product',y='Processing Time',legend=False)
plt.title('Average Processing Time per Product')
plt.xlabel('Product')
plt.ylabel('Average Processing Time(hours)')
plt.xticks(rotation=360)
plt.show()
```

```
<Figure size 800x480 with 0 Axes>
```



Stacked Bar Chart

- We can see from the analysis for each product how every operator performed; for CO-2L, Charlie spent 8 hours, for CO-600 Dee spent 9 hours, for DC-600 Mac spent 4 hours, for LE-600 Mac spent 5 hours, for OR-600 Mac spent 2 hours, for RB-600 Dennis spent 4 hours.
- For the visualization we see how much each product's processing time took for each operator.

In [23]:
```
grouped_data = df.groupby(['Product','Operator'])['Processing Time'].sum().unstack(
print(grouped_data)
```

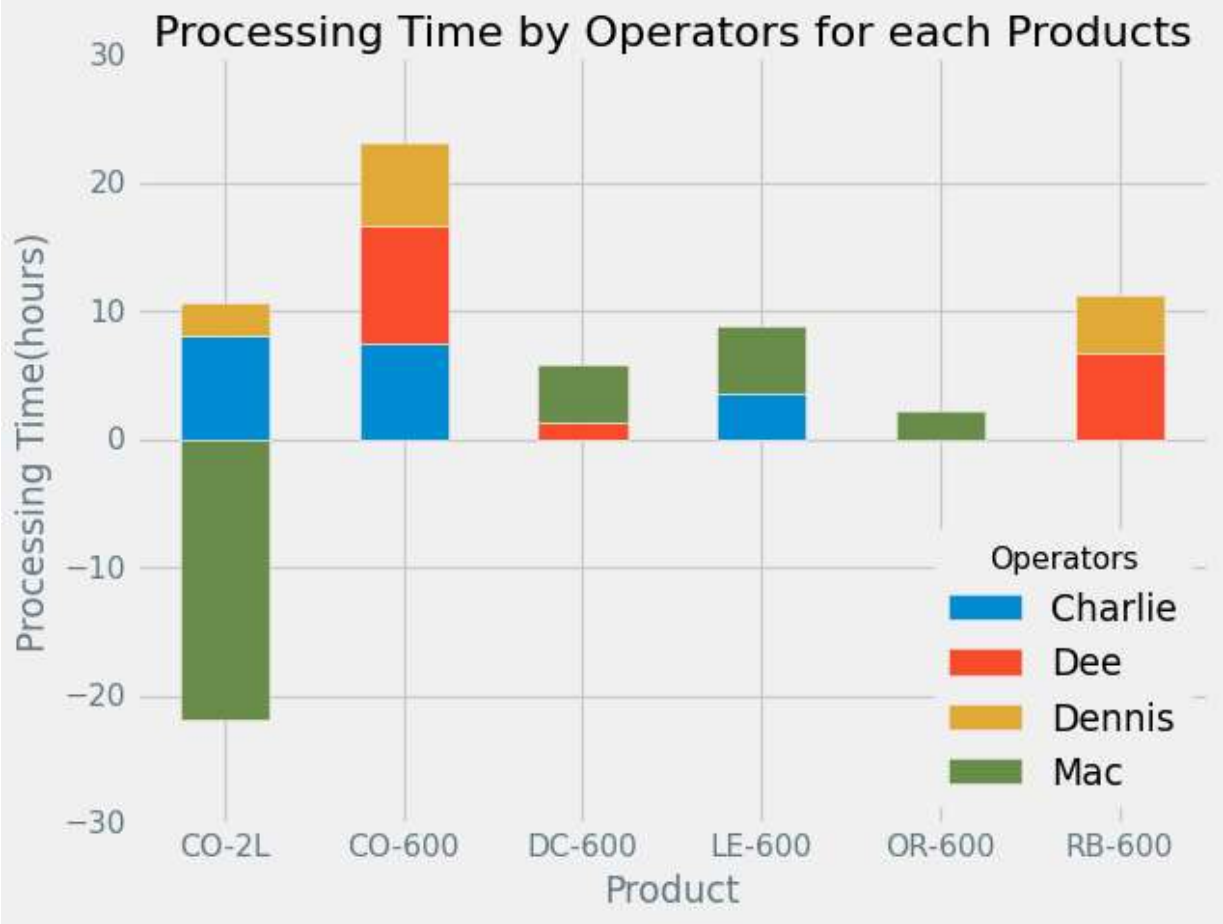
```
Operator   Charlie       Dee    Dennis        Mac
Product
CO-2L     8.083333       NaN  2.533333 -21.833333
CO-600    7.566667  9.116667  6.550000        NaN
DC-600         NaN  1.333333       NaN   4.583333
LE-600    3.650000       NaN       NaN   5.166667
OR-600         NaN       NaN       NaN   2.250000
RB-600         NaN  6.716667  4.583333        NaN
```

In [24]:
```
plt.style.use('fivethirtyeight')
```

In [25]:
```
plt.figure(figsize=(10,6))
grouped_data.plot(kind='bar',stacked=True)
```

```
plt.title('Processing Time by Operators for each Products')
plt.xlabel('Product')
plt.ylabel('Processing Time(hours)')
plt.legend(title='Operators',loc='lower right')
plt.xticks(rotation=360)
plt.show()
```

```
<Figure size 800x480 with 0 Axes>
```



Pie Chart

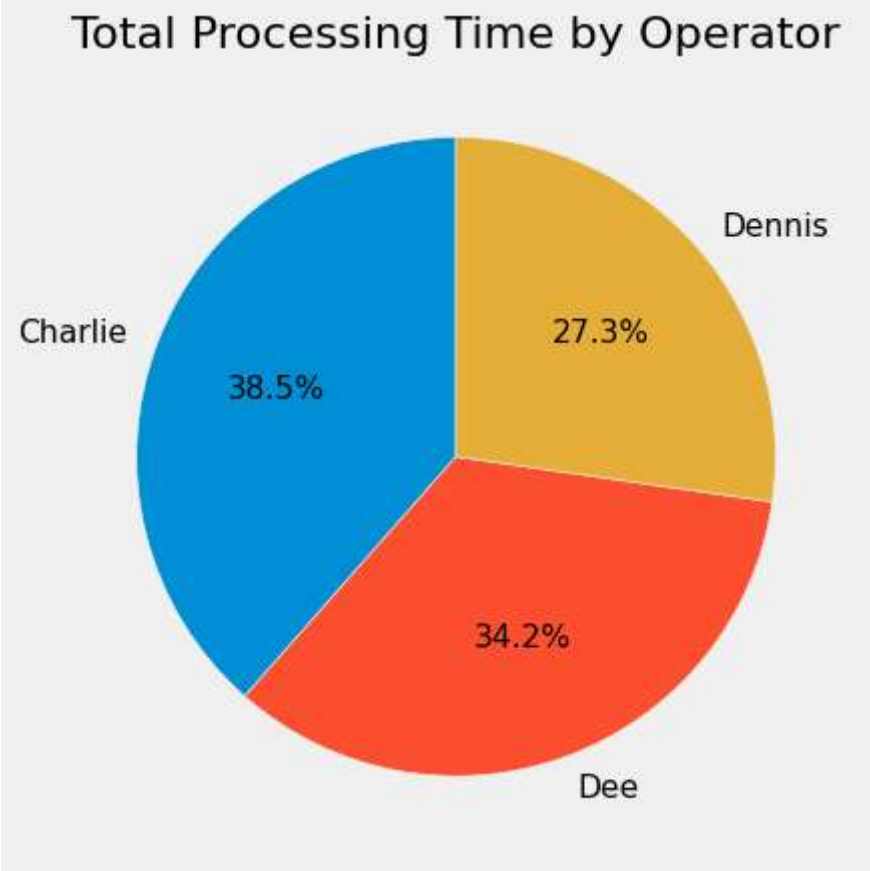
- The pie chart shows the percentage from each operator, Charlie has the highest percentage.
- I could not add Mac because, the pie chart doesn't read negative values.

In [26]:
```
operator_total = df.groupby('Operator')['Processing Time'].sum().head(3)
print(operator_total)
```

```
Operator
Charlie    19.300000
Dee        17.166667
Dennis     13.666667
Name: Processing Time, dtype: float64
```

In [27]:
```
plt.style.use('fivethirtyeight')
```

In [28]:
```
plt.figure(figsize=(10,6))
operator_total.plot(kind='pie',autopct='%1.1f%%',startangle=90)
plt.title('Total Processing Time by Operator')
plt.ylabel('')
plt.show()
```



Heatmap

The heat map conveys the same insight as the stacked bar chart, with more specificity with the amount of hours spent by each operator.

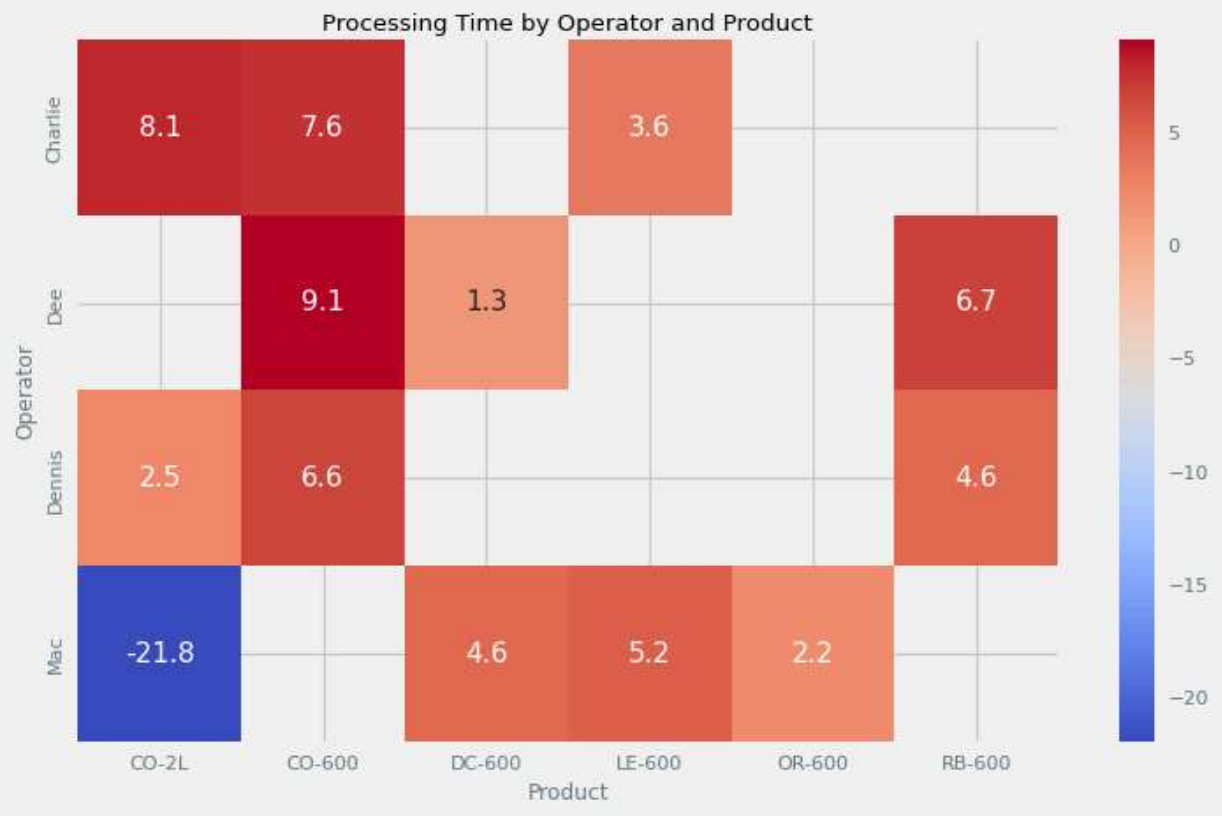
In [29]:
```
heatmap_data = df.pivot_table(values='Processing Time',index='Operator',columns='Pr
print(heatmap_data)
```

```
Product       CO-2L    CO-600    DC-600    LE-600  OR-600    RB-600
Operator
Charlie    8.083333  7.566667       NaN  3.650000     NaN       NaN
Dee             NaN  9.116667  1.333333       NaN     NaN  6.716667
Dennis     2.533333  6.550000       NaN       NaN     NaN  4.583333
Mac      -21.833333       NaN  4.583333  5.166667    2.25       NaN
```

In [30]:
```
plt.style.use('seaborn-v0_8-notebook')
```

In [31]:
```
plt.figure(figsize=(10,6))
sns.heatmap(heatmap_data,annot=True,cmap='coolwarm',fmt='.1f')
plt.title('Processing Time by Operator and Product')
plt.xlabel('Product')
```

```
plt.ylabel('Operator')
plt.show()
```

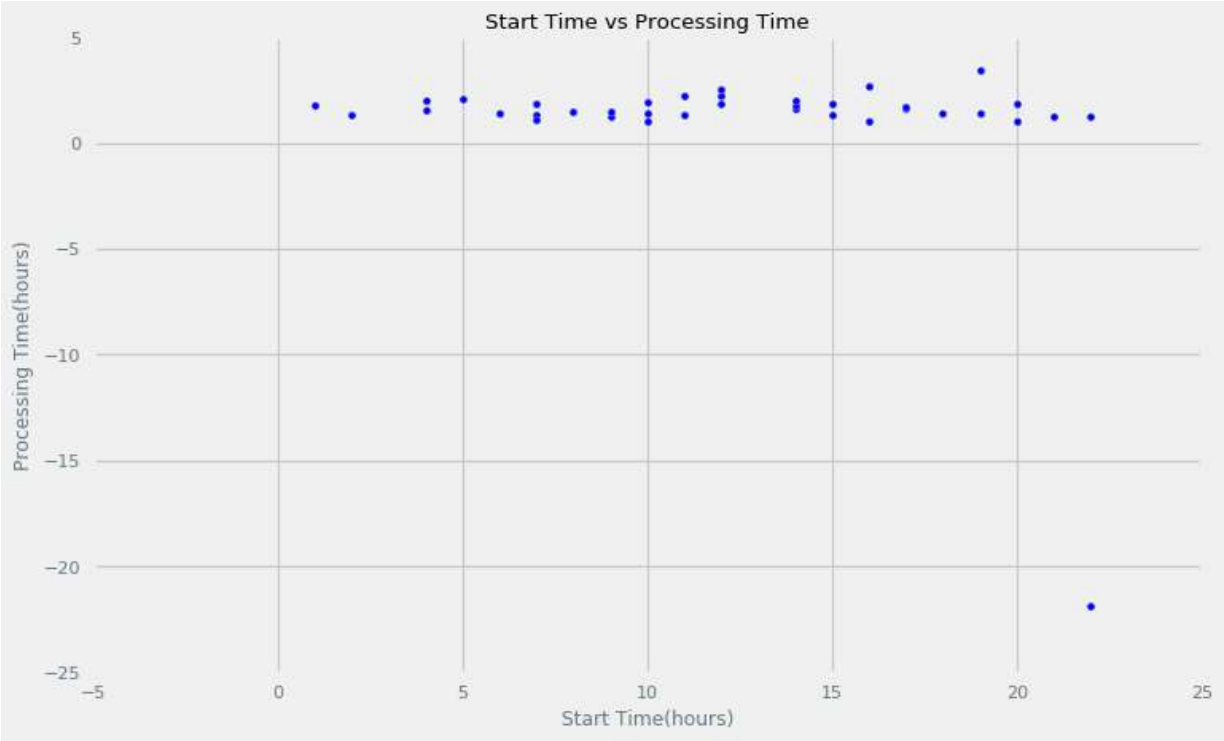


Scatter Plot

The start time and the processing time for all the products is between 0 to 5 hours, that means all operators get to work and they start processing.

In [32]:
```
plt.style.use('_classic_test_patch')
```

In [33]:
```
plt.figure(figsize=(10,6))
plt.scatter(df['Start Time'].dt.hour, df['Processing Time'])
plt.title('Start Time vs Processing Time')
plt.xlabel('Start Time(hours)')
plt.ylabel('Processing Time(hours)')
plt.show()
```


Start Time vs Processing Time
Processing Time(hours)
Start Time(hours)
5
0
−5
−10
−15
−20
−25
−5
0
5
10
15
20
25