In [1]:

```
import json
from os import listdir
from os.path import isfile, join
```

In [2]:

```
import pandas as pd
```

Load in Data

- Open all json files that contain traces and append them to a list with all of the traces.
    - This makes the data ready to use with pandas dataframes and easy to create visualizations with Altair

In [12]:

```
baseDirectory = 'data/synthetic/20210302-hipster-shop'
directories = listdir(baseDirectory)
```

In [13]:

```
traces = []
for directory in directories:
    thisDirectory = baseDirectory + '/' + directory
    try:
        with open(thisDirectory) as f:
          data = json.load(f)
    except:
        continue
    traces.append(data)
```

In [15]:

```
names = []
durations = []
traceIDs = []
for trace in traces:

    for element in trace:
        traceIDs.append(element['traceId'])
        names.append(element['name'])
        durations.append(element['duration'])
```

In [16]:

```
traceDf = pd.DataFrame({'Resource Name' : names, 'Duration':durations, 'Tra
traceDf
```

Out[16]:

	Resource Name	Duration	Trace_ID
0	/getcart	190000	550997223f8c4b30
1	/getrecommendations	226000	550997223f8c4b30
2	/getproducts	5000	550997223f8c4b30
3	/cart	334000	550997223f8c4b30
4	/product	501000	592363a229596c88
...	...	...	...
19558	/cart	397000	529885a8ac3c2592
19559	/cart	485000	b9ae10ad77e3ecee
19560	/getproducts	95000	b9ae10ad77e3ecee
19561	/getcart	94000	b9ae10ad77e3ecee
19562	/getrecommendations	433000	b9ae10ad77e3ecee

19563 rows × 3 columns

Three hypotheses

```
1. The provided data will have traces that use some resources more
 than others. These resources may be taxing on the system if they t
ake too long to complete and are commonly used.

2. Traces within this system architecture will have different run t
imes based on how many resources they use to complete the task at h
and.

3. Specific traces may have errors that lead to longer runtimes tha
t we must identify to trouble shoot and improve system performance.
```

Hypothesis #1

```
The provided data will have traces that use some resources more tha
n others. These resources may be taxing on the system if they take
 too long to complete and are commonly used.
```

In [18]:
```
averagetraceDf = traceDf.groupby('Resource Name')['Duration'].agg(['mean',
averagetraceDf
```

Out[18]:

	Resource Name	Mean Duration	count
0	/address	5.068270e+04	208
1	/adrequest	2.460234e+05	728
2	/cart	3.605121e+05	3644
3	/checkout	1.625361e+06	119
4	/creditcardinfo	2.560921e+04	87
5	/currency	1.820784e+05	51
6	/getcart	1.001764e+05	3731
7	/getconversion	1.483261e+05	138
8	/getproducts	5.542275e+04	5100
9	/getquote	3.072727e+05	121
10	/getrecommendations	2.564613e+05	4372
11	/money	4.876087e+04	138
12	/orderresult	4.457746e+04	71
13	/placeorder	7.807731e+05	119
14	/product	5.275755e+05	728
15	/sendorderconfirmation	4.840920e+05	87
16	/shipping	3.523140e+05	121

In [19]:

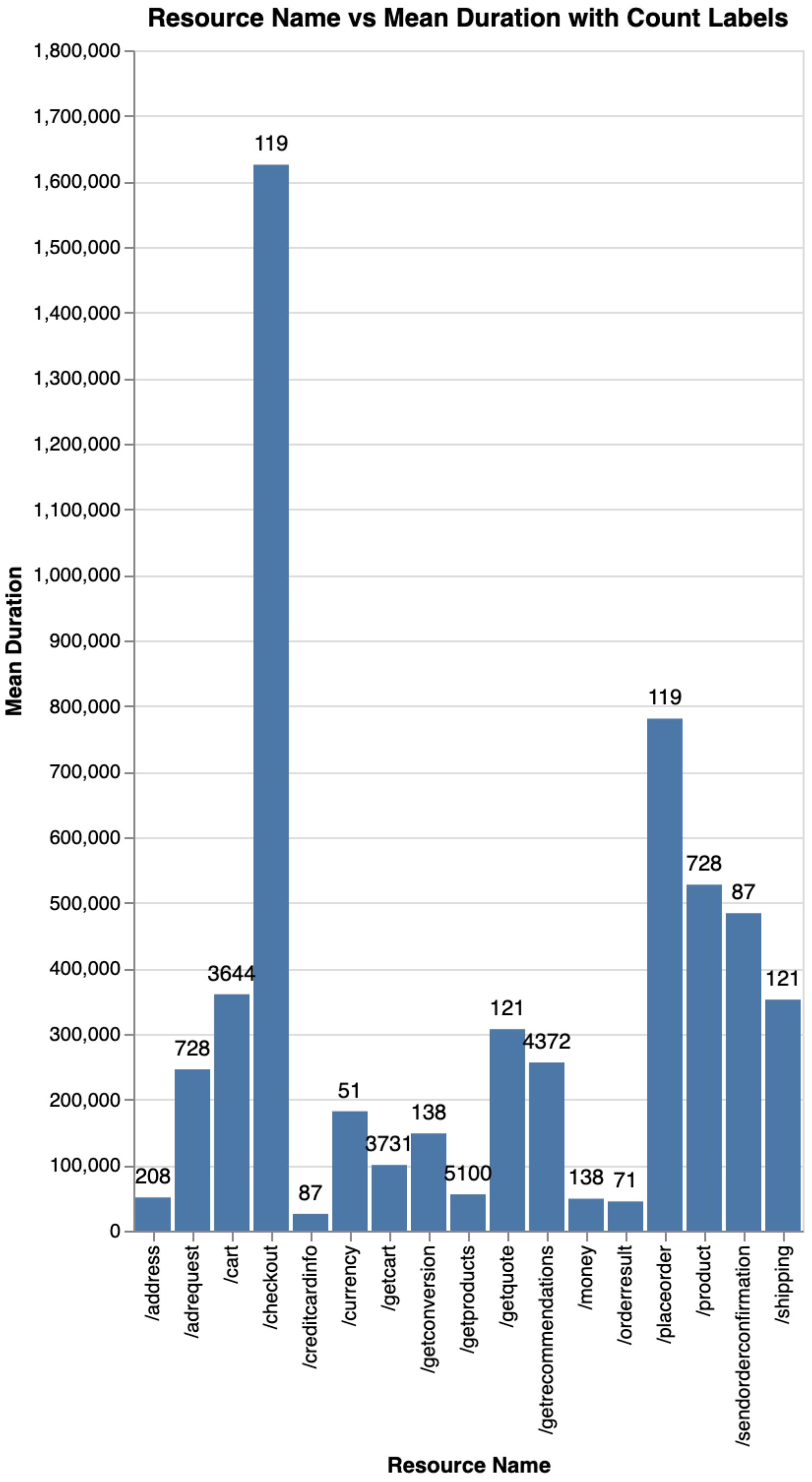
```
import altair as alt

source = averagetraceDf

bars = alt.Chart(source, title='Resource Name vs Mean Duration with Count L
    x='Resource Name',
    y='Mean Duration'
)

text = bars.mark_text(
    baseline='middle',
    dy=-10  # Nudges text to right so it doesn't appear on top of the bar
).encode(
    text='count:Q'
)
(bars + text).properties(height=600)
```

Out[19]:



The bar plot above shows how each resource have very different durations. /Checkout on average takes the longest time to complete but is only used 119 times in our trace dataset. For the purpose of minimizing durations throughout our system, I believe it is important to focus on processes such as /cart and /getrecommendations that are used commonly and have a relatively long duration.

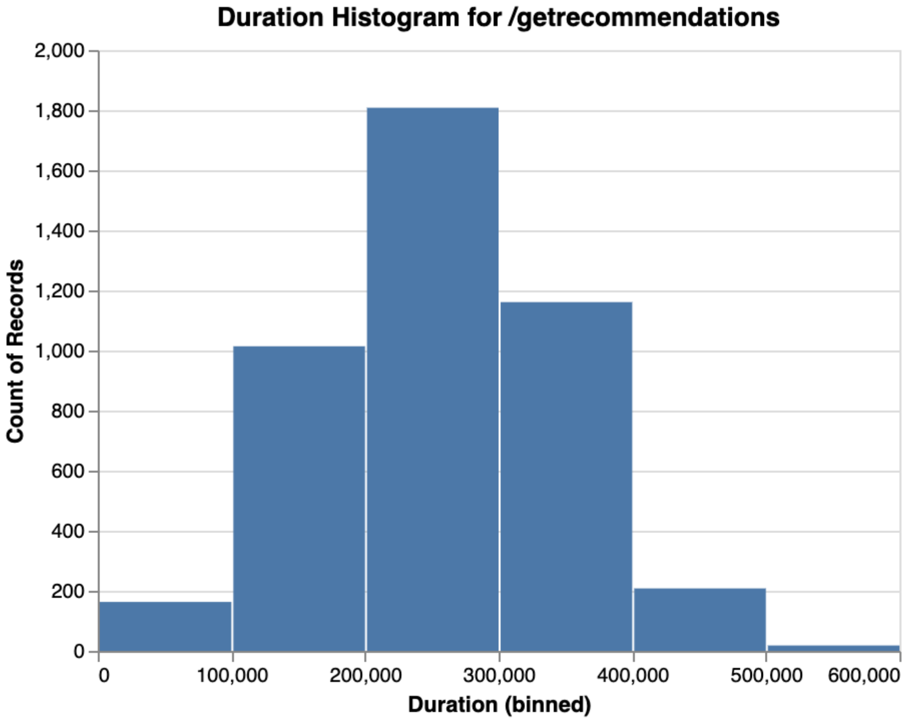
Refine

With a better understanding of average durations for each proess and how often they occur in our trace data, it would be helpful to see the duration distributions for concerning process that take a long time and occur frequently.

In [29]:
```
def histogram(resourceName):
    source = traceDf[traceDf['Resource Name'] == resourceName]
    return alt.Chart(source, title = "Duration Histogram for " + resourceNa
        alt.X("Duration:Q", bin=True),
        y='count()',
    )
```

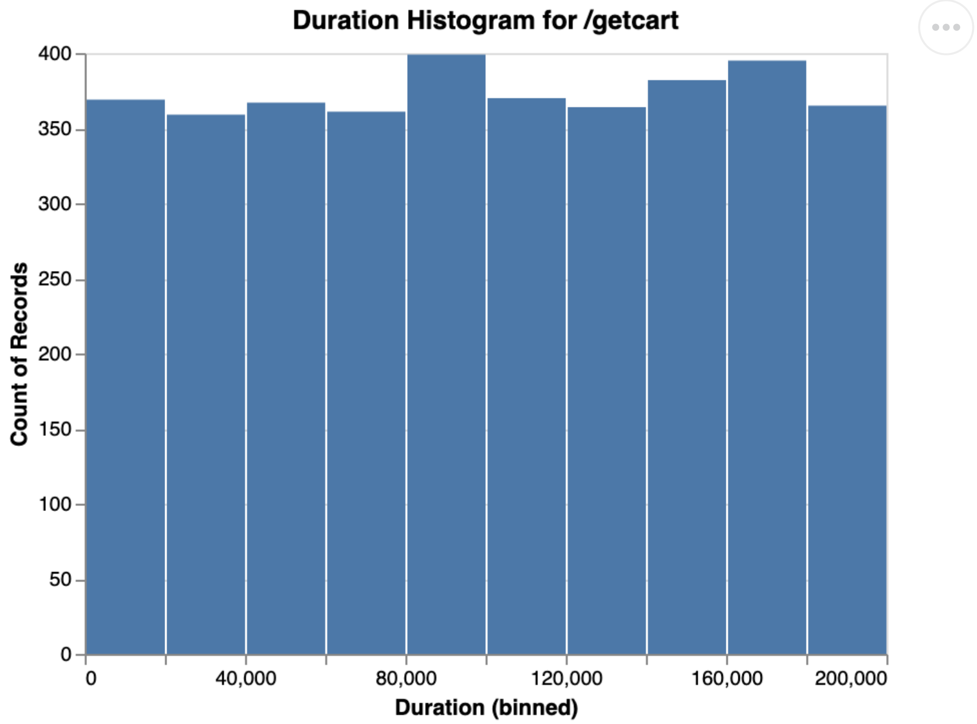
In [30]:
```
histogram("/getrecommendations")
```

Out[30]:



In [31]:
```
histogram("/getcart")
```

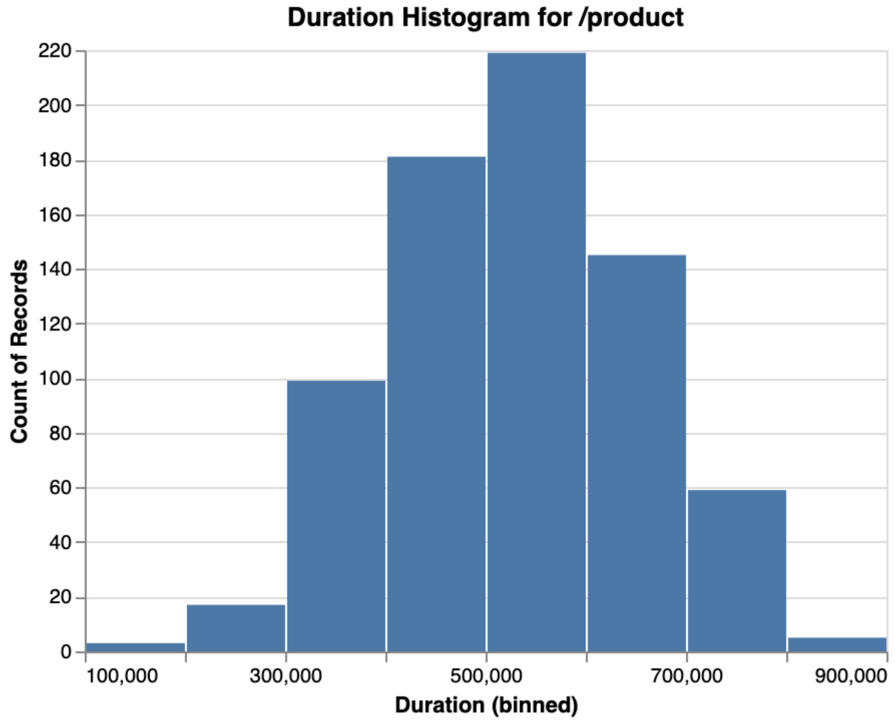
Out[31]:



In [32]:

```
histogram("/product")
```

Out[32]:



Generally, it appears that the distribution of resources are relatively normal. However, /getcart has a normal distribution. For these concerning resources, I am curious to what is causing very low and very high durations. If a system engineer can pin point whatever is significantly improving durations, they could figure out how to improve the systems performance as a whole.

Hypothesis #2

```
Traces within this system architecture will have different run time
s based on how many resources they use to complete the task at han
d.
```

In [9]:

```
traceNetDurationResourceCount = traceDf.groupby('Trace_ID')['Duration'].agg
traceNetDurationResourceCount
```

Out[9]:

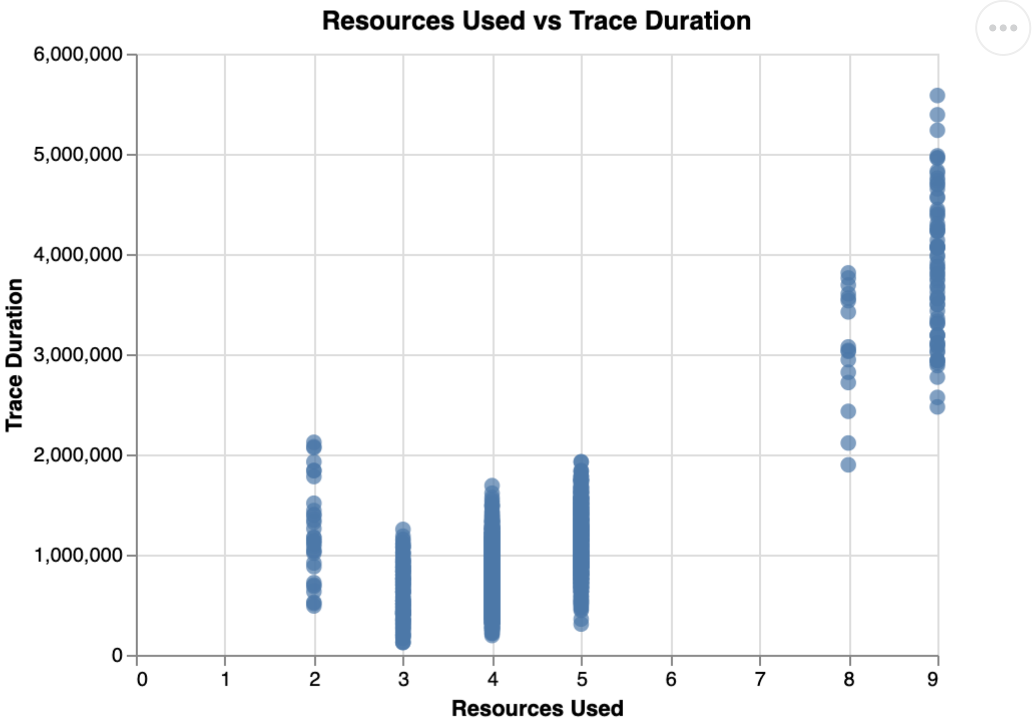
	Trace_ID	Trace Duration	Resources Used
0	00068a67bc793add	858000	4
1	000ba11f524d250f	809000	4
2	00200ac17dbf541d	593000	4
3	0021a1abb3546ead	759000	5
4	0044a79dfe680331	572000	4
...	...	...	...
4658	ffdb50d2a2d18c5b	795000	4
4659	ffe88d8088ee8d5e	997000	4
4660	ffea94949d425fe9	4380000	9
4661	fff712da02528d14	959000	4
4662	fffe889dcec9d1b5	607000	4

4663 rows × 3 columns

In [10]:

```
source = traceNetDurationResourceCount
alt.Chart(source, title = 'Resources Used vs Trace Duration').mark_circle(s
    x='Resources Used',
    y='Trace Duration',
    tooltip = ['Trace_ID','Resources Used', 'Trace Duration']
)
```

Out[10]:



The plot above suggests that generally more resources used does lead to a greater duration. Additionally, it is interesting to see the range of duration for each number of resources used. When 2 resources are used they take a surprisingly long time compared to the other groups. I assume that when 2 resources are used they are long resources that take a long time to complete.

Hypothesis #3

```
Specific traces may have errors that lead to longer runtimes that w
e must identify to trouble shoot and improve system performance.
```

In [34]:
```
traceDf
```

Out[34]:

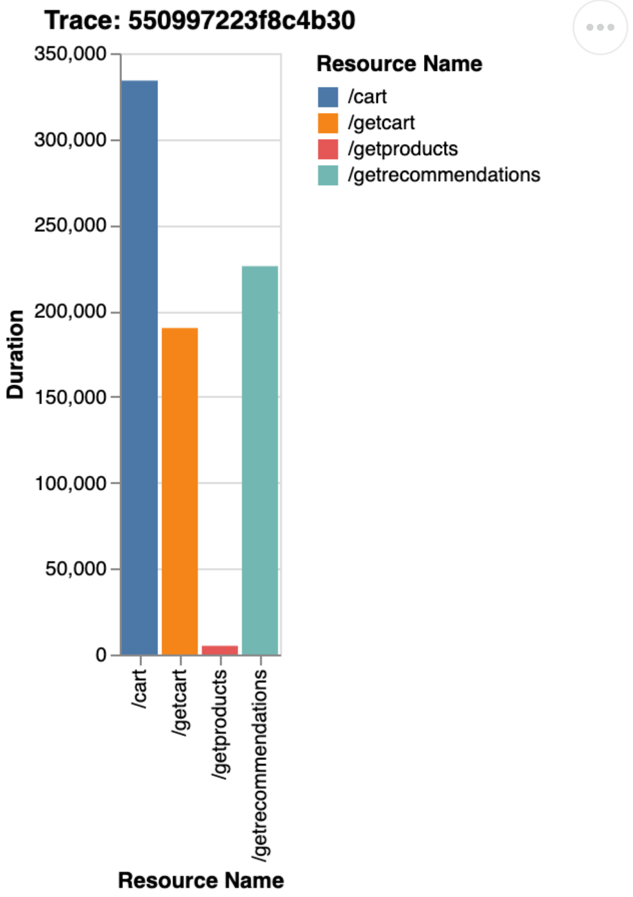
	Resource Name	Duration	Trace_ID
0	/getcart	190000	550997223f8c4b30
1	/getrecommendations	226000	550997223f8c4b30
2	/getproducts	5000	550997223f8c4b30
3	/cart	334000	550997223f8c4b30
4	/product	501000	592363a229596c88
...	...	...	...
19558	/cart	397000	529885a8ac3c2592
19559	/cart	485000	b9ae10ad77e3ecee
19560	/getproducts	95000	b9ae10ad77e3ecee
19561	/getcart	94000	b9ae10ad77e3ecee
19562	/getrecommendations	433000	b9ae10ad77e3ecee

19563 rows × 3 columns

In [47]:
```
def plotTrace(traceID):
    source = traceDf[traceDf['Trace_ID'] == traceID]
    return alt.Chart(source, title='Trace: ' + traceID).mark_bar().encode(
        x='Resource Name',
        y='Duration',
        color = 'Resource Name:N'
    )
```

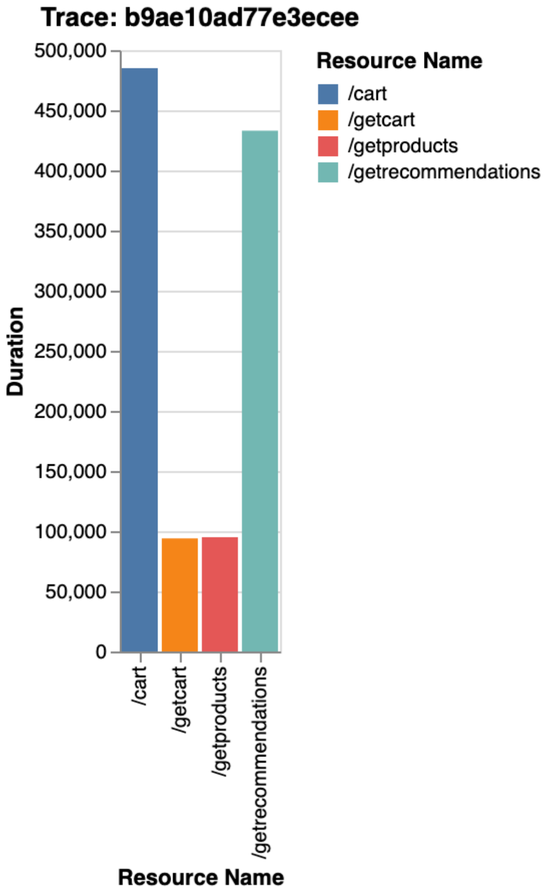
```
In [48]: plotTrace('550997223f8c4b30')
```

Out[48]:



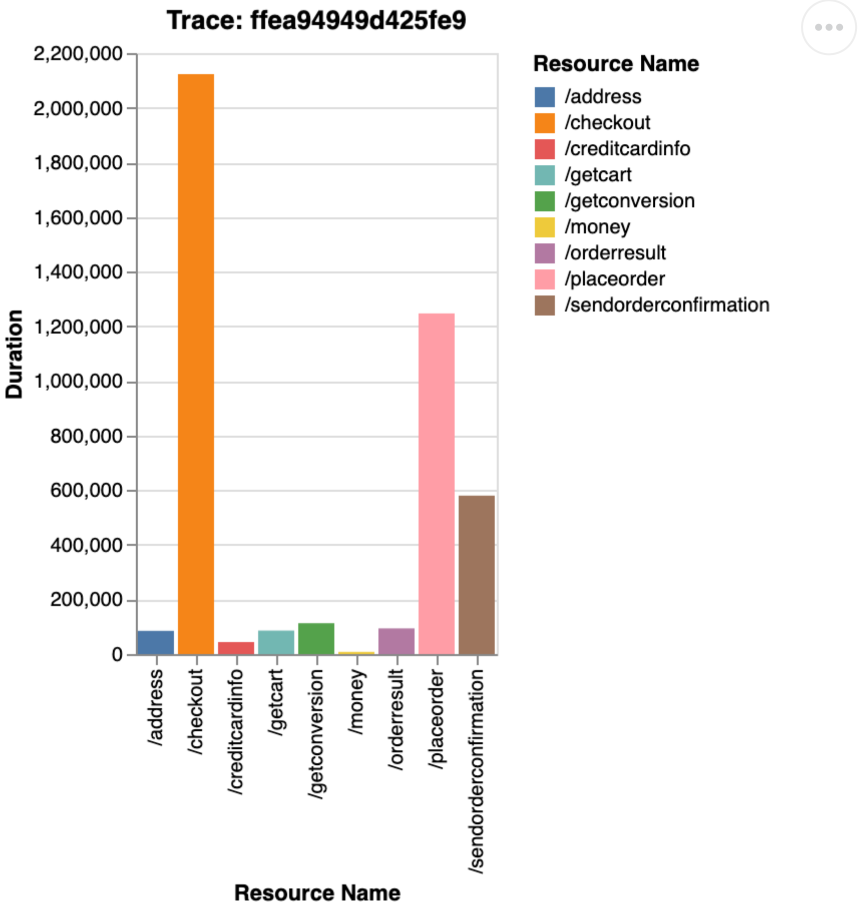
In [49]:
```
plotTrace('b9ae10ad77e3ecee')
```

Out[49]:



```
In [50]: plotTrace('ffea94949d425fe9')
```

Out[50]:



These detailed looks into specific traces are helpful at understanding how our system processes specific requests. Additionally, these trace visualizations show how individual traces can be improved and which aspects take up the majority of the time. Systems engineer would likely use these visualizations to test new implementation and identify issues within the system.

In [ ]: