LockerDome Technical Assessment

Overview

This Notebook details the exploration and analysis of data from a single advertiser between the dates 12/01/2020 and 02/28/2021.

Project Goals

Through the lens of optimizing for LockerDome:

- Using raw historical campaign data, make concrete optimization recommendations for Creative.ID, Slot.ID, and Device variables.
- Deliver results as tables and graphics that illustrate results
- Detail any additional observations

Metrics

The main metric in this analysis will be **Net Revenue** (Gross - Publisher Split)

Data

The data used in this analysis describes a single advertiser's campaigns. The data is imported and previewed below.

Note: project_data.csv is too large to upload to GitHub. If this repo is cloned, a copy of project_data.csv must be added to the data folder.

For the reader's convenience, the data can be downloaded here.

In [1]:
```
# Import Key DataFrame Libraries
import pandas as pd
import numpy as np
```

In [2]:
```
#Import and Preview Data
df = pd.read_csv('data/project_data.csv')
df.head()
```

Out[2]:

	Slot.ID	Placement	Ad.Slot.Page.Layout	Creative.ID	Content.Type	Content.Has.Play
0	1	NaN	NaN	1	advertorial	
1	2	external_in_content	full_article	1	advertorial	
2	3	external_above_content	full_article	1	advertorial	
3	4	external_below_content	full_article	1	advertorial	

	Slot.ID	Placement	Ad.Slot.Page.Layout	Creative.ID	Content.Type	Content.Has.Play
4	5	external_below_content	full_article	1	advertorial	

In [3]:
```
# Inspect Column Datatypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5484486 entries, 0 to 5484485
Data columns (total 13 columns):
 #   Column                   Dtype
---  ------                   -----
 0   Slot.ID                  int64
 1   Placement                object
 2   Ad.Slot.Page.Layout      object
 3   Creative.ID              int64
 4   Content.Type             object
 5   Content.Has.Play.Button  object
 6   Date                     object
 7   Device                   object
 8   Size                     object
 9   Publisher.Split          float64
 10  Impressions              int64
 11  Referrals                int64
 12  Conversions              int64
dtypes: float64(1), int64(5), object(7)
memory usage: 544.0+ MB
```

In [4]:
```
# Inspect Missing Values
df.isna().sum()
```

Out[4]:
```
Slot.ID                       0
Placement                 12326
Ad.Slot.Page.Layout       12330
Creative.ID                   0
Content.Type                  0
Content.Has.Play.Button       0
Date                          0
Device                        0
Size                          0
Publisher.Split               0
Impressions                   0
Referrals                     0
Conversions                   0
dtype: int64
```

Feature Creation/DataType Correction

Through 1/7, conversion value is assumed to be $35.00. All dates following, the conversion value is assumed to be \$40.00. This information can be used to calculate gross revenue.

To help create these features efficiently, the Date column will be converted to the `datetime64` data type. Using a lambda function and the corrected Date column, a conversion_value column can be properly populated.

In [5]:
```
# Convert Date column from object to datetime
```

```
df['Date'] = pd.to_datetime(df['Date'])
```

In [6]:
```
# Create conversion value column
value_change_date = pd.to_datetime('01/08/2021')
df['conversion_value'] = df['Date'].map(lambda x: 35 if x < value_change_date el
```

In [7]:
```
# Gross revenue = conversions * conversion value
df['gross_revenue'] = df['Conversions']*df['conversion_value']
```

To find net revenue, we must subtract publisher costs from the gross revenue.

The `Publisher.Split` column is defined as the amount paid to publishers for their impressions.

In [8]:
```
# Net revenue = Gross revenue - Publisher Split
df['net_revenue'] = df['gross_revenue'] - df['Publisher.Split']
```

In [9]:
```
#Preview Current DataFrame
df.head()
```

Out[9]:

	Slot.ID	Placement	Ad.Slot.Page.Layout	Creative.ID	Content.Type	Content.Has.Play
0	1	NaN	NaN	1	advertorial	
1	2	external_in_content	full_article	1	advertorial	
2	3	external_above_content	full_article	1	advertorial	
3	4	external_below_content	full_article	1	advertorial	
4	5	external_below_content	full_article	1	advertorial	

Dealing with Missing Values

A small percentage of the values (roughly 0.22%) in columns `Placement` and `Ad.Slot.Page.Layout` are missing. Upon inspection, none of the rows with missing data produced any profit, leading to the assumption that more data in those rows may be misleading.

For the purpose of this analysis, rows with missing columns will be removed.

In [10]:
```
# checking if rows with missing Placement values produce any revenue
df[df['Placement'].isna()]['net_revenue'].sum()
```

Out[10]:
```
0.0
```

In [11]:
```
# Checking if rows with missing Ad.Slot.Page.Layout produce any revenue
```

```
df[df['Ad.Slot.Page.Layout'].isna()]['net_revenue'].sum()
```

Out[11]:
```
0.0
```

In [12]:
```
# Drop Missing Data and Reset Index
df.dropna(inplace=True)
df.reset_index(inplace=True, drop=True)
```

In [13]:
```
df.head()
```

Out[13]:

	Slot.ID	Placement	Ad.Slot.Page.Layout	Creative.ID	Content.Type	Content.Has.Play
0	2	external_in_content	full_article	1	advertorial	
1	3	external_above_content	full_article	1	advertorial	
2	4	external_below_content	full_article	1	advertorial	
3	5	external_below_content	full_article	1	advertorial	
4	6	external_feed	homepage	1	advertorial	

Recap: At this point, the data is in the correct datatype, new features have been created (including our primary metric), and missing values have been dealt with.

Data Analysis

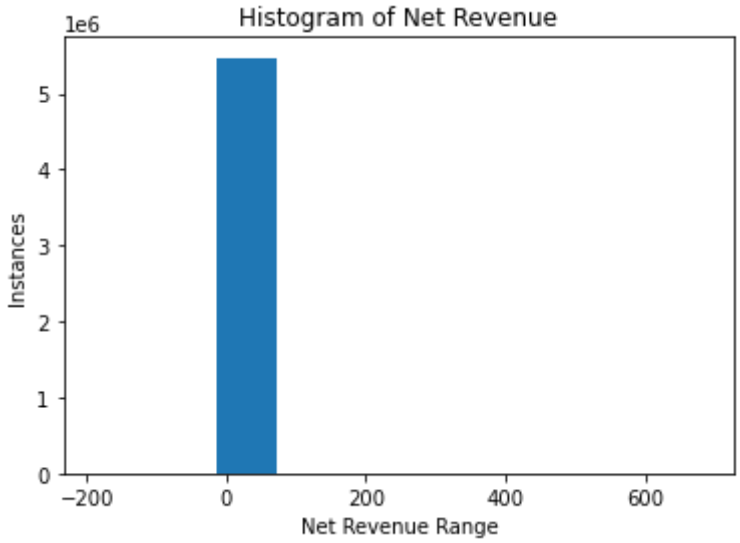
With the properly structured data, a combination of charts and pivot tables can be used to draw conclusions and make recommendations on ways to optimize the campaign.

Check for Outliers

Before aggregating data into pivot tables, it is useful to check for data outliers that may skew results.

In [14]:
```
# import graphing packages
import matplotlib.pyplot as plt
%matplotlib inline
```

In [15]:
```
plt.hist(df['net_revenue'])
plt.title('Histogram of Net Revenue')
plt.xlabel('Net Revenue Range')
plt.ylabel('Instances')
plt.show()
```

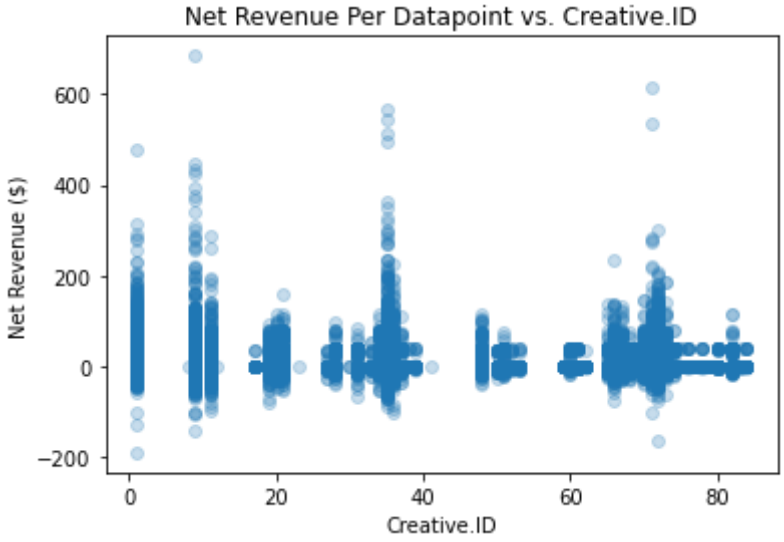


Without adjusting histogram parameters, it appears that the majority of instances hover around a value of 0, with some outliers ranging from -200 to 600 and above. These outliers cannot be discounted without further investigation, as they may be a result of other variables.

To explore that possibility, The net revenue of each instance can be plotted against the Creative.ID, Slot.ID, and Device.

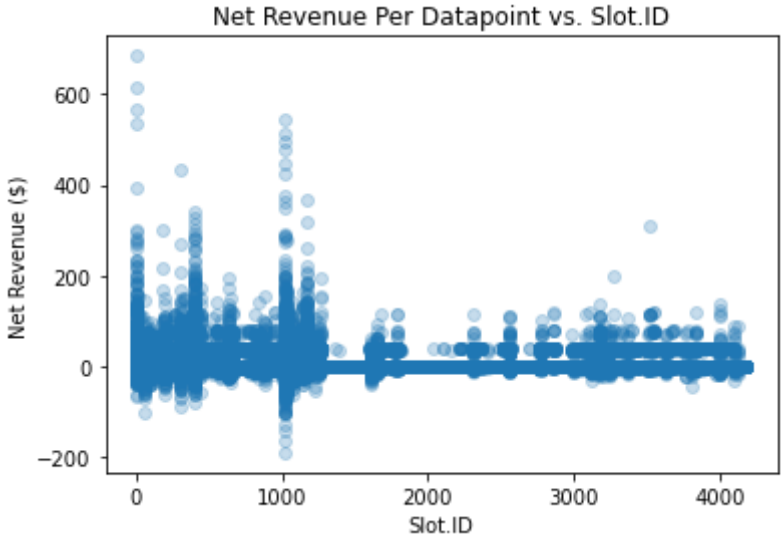
These plots can function as a histogram with an extra dimension.

In [16]:
```
#Revenue vs Creative ID
plt.scatter(df['Creative.ID'], df['net_revenue'], alpha=0.25)
plt.title('Net Revenue Per Datapoint vs. Creative.ID')
plt.xlabel('Creative.ID')
plt.ylabel('Net Revenue ($)')
plt.show()
```



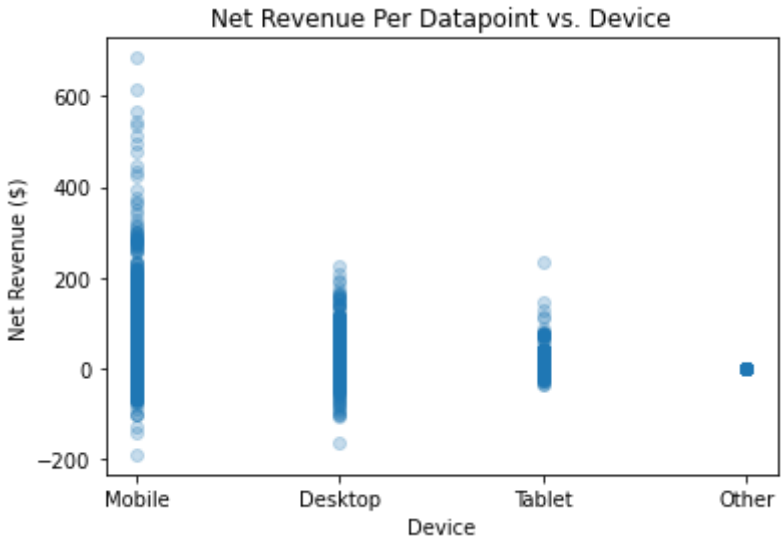
In [17]:
```
#Revenue vs Slot ID
plt.scatter(df['Slot.ID'], df['net_revenue'], alpha=0.25)
plt.title('Net Revenue Per Datapoint vs. Slot.ID')
plt.xlabel('Slot.ID')
```

```
plt.ylabel('Net Revenue ($)')
plt.show()
```



In [18]:

```
#Revenue vs Device
plt.scatter(df['Device'], df['net_revenue'], alpha=0.25)
plt.title('Net Revenue Per Datapoint vs. Device')
plt.xlabel('Device')
plt.ylabel('Net Revenue ($)')
plt.show()
```



These graphs make outliers less obvious. For the purpose of this analysis, all values currently in the working dataframe will be kept.

Pivot Tables

To make insites possible to the human eye, data will be aggregated into pivot tables.

Net Revenue as a Function of Creative ID and Device

The following pivot table aggregates the sum of the net revenue made by each Creative.ID by Device Type.

In [19]:
```
# Create Pivot Table
create_by_device = pd.pivot_table(df, values='net_revenue',index = ['Creative.ID
                                  columns=['Device'], aggfunc=np.sum, fill_value=0)
# Display Pivot Table
create_by_device
```

Out[19]:

Device	Desktop	Mobile	Other	Tablet
Creative.ID				
1	40.00	38572.25	0	0.00
8	0.00	0.00	0	0.00
9	2064.91	26339.46	0	8.45
11	1043.61	10309.43	0	0.00
12	0.00	0.00	0	0.00
17	0.00	0.00	0	-12.67
19	1511.70	2595.74	0	-15.69
20	2433.33	4115.19	0	91.02
21	0.00	7943.12	0	0.00
23	0.00	0.00	0	0.00
27	-266.12	0.00	0	0.00
28	1827.19	240.00	0	0.00
30	0.00	0.00	0	0.00
31	1528.80	160.00	0	0.00
32	0.00	0.00	0	0.00
33	-66.43	0.00	0	0.00
34	-18.53	6189.41	0	40.21
35	3704.43	41026.76	0	1698.70
36	14287.98	155.00	0	1678.93
37	1616.21	0.00	0	-24.50
38	11.14	0.00	0	11.36
39	9.36	0.00	0	50.67
41	0.00	0.00	0	0.00
48	0.00	3962.65	0	0.00
50	0.00	-123.06	0	0.00
51	0.00	252.97	0	246.99
52	0.00	130.19	0	0.00
53	0.00	50.01	0	0.00
59	0.00	0.00	0	-8.35
60	75.00	0.00	0	146.96

Device	Desktop	Mobile	Other	Tablet
Creative.ID				
61	0.00	0.00	0	-7.47
62	0.00	0.00	0	-34.64
65	0.00	1659.69	0	-14.65
66	53.00	3989.41	0	-21.58
67	1321.74	2921.32	0	1270.90
68	-482.14	115.00	0	-62.15
69	-165.66	-9.62	0	0.00
70	3257.14	3748.24	0	-44.29
71	1958.44	14545.86	0	0.00
72	15537.76	15993.47	0	549.28
73	714.34	4511.93	0	-41.78
74	671.39	-63.93	0	-44.79
75	58.10	0.00	0	0.00
76	-22.00	165.60	0	-7.31
77	0.00	0.00	0	-21.30
78	0.00	0.00	0	136.76
79	0.00	0.00	0	-7.42
80	40.00	0.00	0	36.03
82	0.00	1141.81	0	0.00
83	0.00	-38.40	0	0.00
84	0.00	205.62	0	0.00

This pivot table is useful because it reveals which Creative IDs net the highest for each device type. This can be more easily read with the values sorted in descending order.

Below are previews of the pivot table ordered by each device in descending order.

Also, because there is no data in the `Other` column, it will be removed.

In [20]:
```
# Drop 'Other' Column
create_by_device.drop(columns=['Other'], inplace=True)
```

In [21]:
```
# Preview table ordered by Desktop
create_by_device.sort_values(by='Desktop', ascending=False).head(10)
```

Out[21]:

Device	Desktop	Mobile	Tablet
Creative.ID			

Creative.ID	Desktop	Mobile	Tablet
72	15537.76	15993.47	549.28
36	14287.98	155.00	1678.93
35	3704.43	41026.76	1698.70
70	3257.14	3748.24	-44.29
20	2433.33	4115.19	91.02
9	2064.91	26339.46	8.45
71	1958.44	14545.86	0.00
28	1827.19	240.00	0.00
37	1616.21	0.00	-24.50
31	1528.80	160.00	0.00

In [22]:
```
# Preview table ordered by Mobile
create_by_device.sort_values(by='Mobile', ascending=False).head(10)
```

Out[22]:

Creative.ID	Desktop	Mobile	Tablet
35	3704.43	41026.76	1698.70
1	40.00	38572.25	0.00
9	2064.91	26339.46	8.45
72	15537.76	15993.47	549.28
71	1958.44	14545.86	0.00
11	1043.61	10309.43	0.00
21	0.00	7943.12	0.00
34	-18.53	6189.41	40.21
73	714.34	4511.93	-41.78
20	2433.33	4115.19	91.02

In [23]:
```
# Preview table ordered by Tablet
create_by_device.sort_values(by='Tablet', ascending=False).head(10)
```

Out[23]:

Creative.ID	Desktop	Mobile	Tablet
35	3704.43	41026.76	1698.70
36	14287.98	155.00	1678.93
67	1321.74	2921.32	1270.90

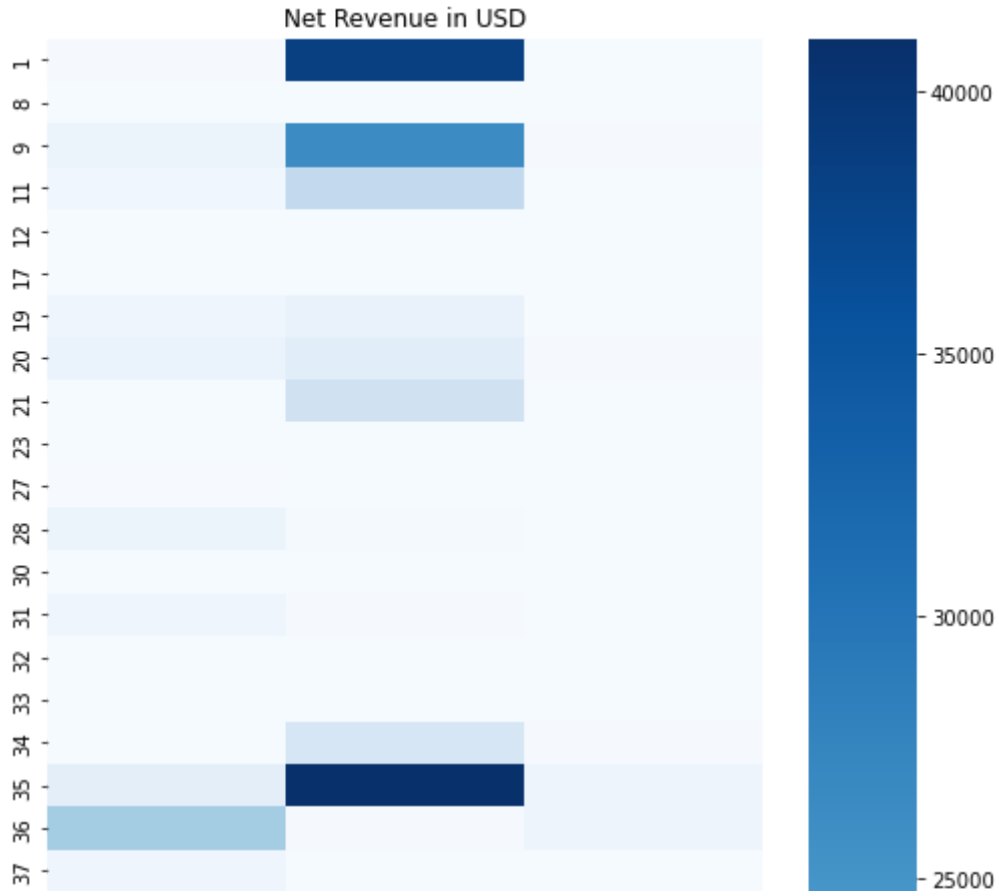
Creative.ID	Desktop	Mobile	Tablet
72	15537.76	15993.47	549.28
51	0.00	252.97	246.99
60	75.00	0.00	146.96
78	0.00	0.00	136.76
20	2433.33	4115.19	91.02
39	9.36	0.00	50.67
34	-18.53	6189.41	40.21

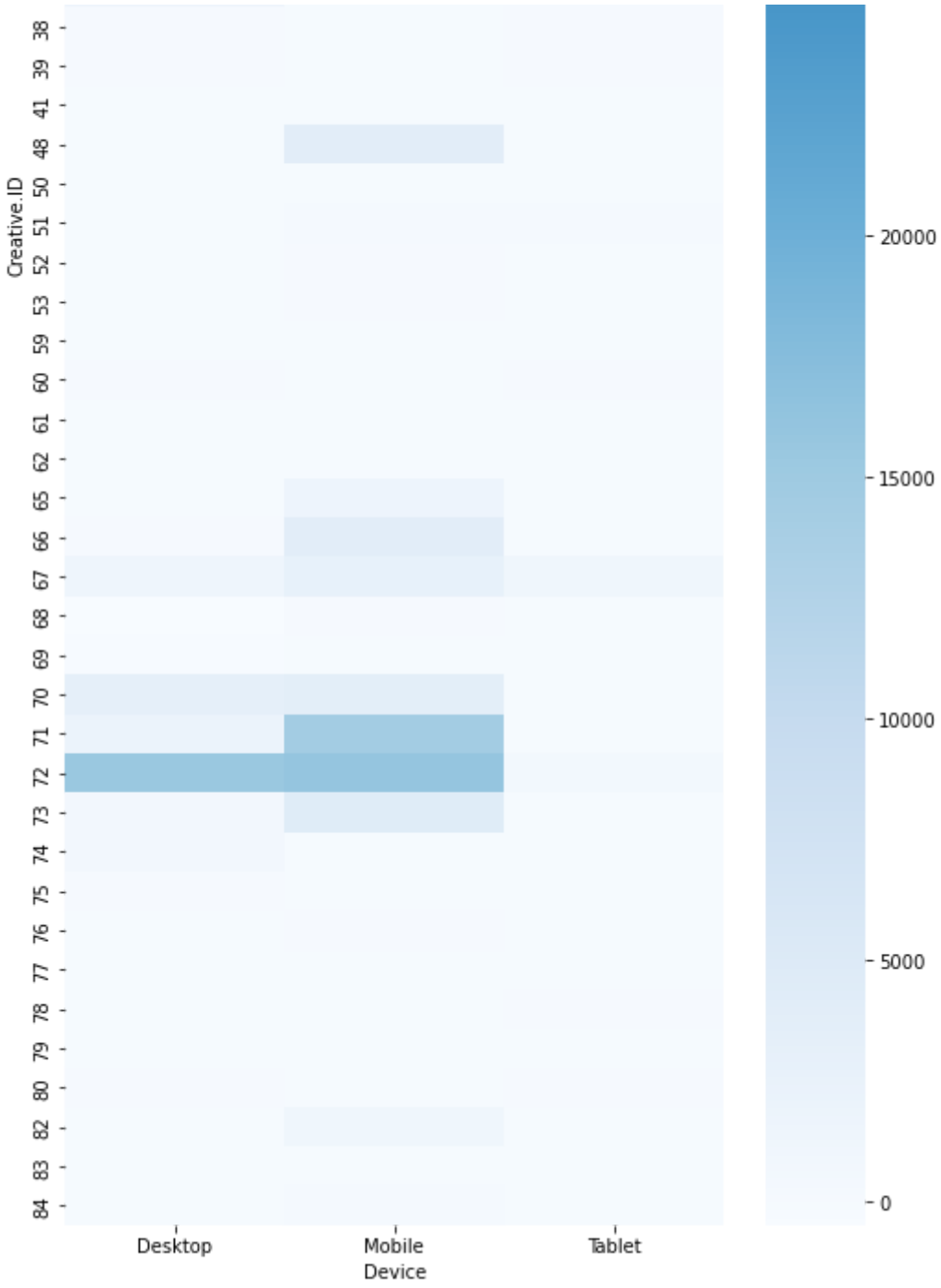
Pivot Table Visualization

If a precise numerical value is not neccessary for an initial performance review, a heatmap representation of the above pivot table delivers information efficiently.

In [24]:
```
# Import Packages
import seaborn as sns
```

In [25]:
```
# Create Heatmap with Seaborn
plt.figure(figsize=(8,20))
sns.heatmap(create_by_device, cmap='Blues')
plt.title('Net Revenue in USD')
plt.show()
```





Adding Slot.ID Variable to Pivot Table

Adding Slot.ID as an index to the above Pivot table might allow for a more detailed visual analysis.

In [26]:
```
# Create Pivot Table
add_slot_pivot = pd.pivot_table(df, values='net_revenue',index = ['Creative.ID',
                                columns=['Device'], aggfunc=np.sum, fill_value=0)
# Drop 'Other' column
add_slot_pivot.drop(columns=['Other'], inplace=True)
```

With the added dimension, however, the pivot table becomes so long that it can no longer reasonably be analyzed with the human eye alone. A heatmap similarly would be too condensed

for a viewer to gather information.

In [27]:
```
print('Shape of New Pivot Table')
add_slot_pivot.shape
```

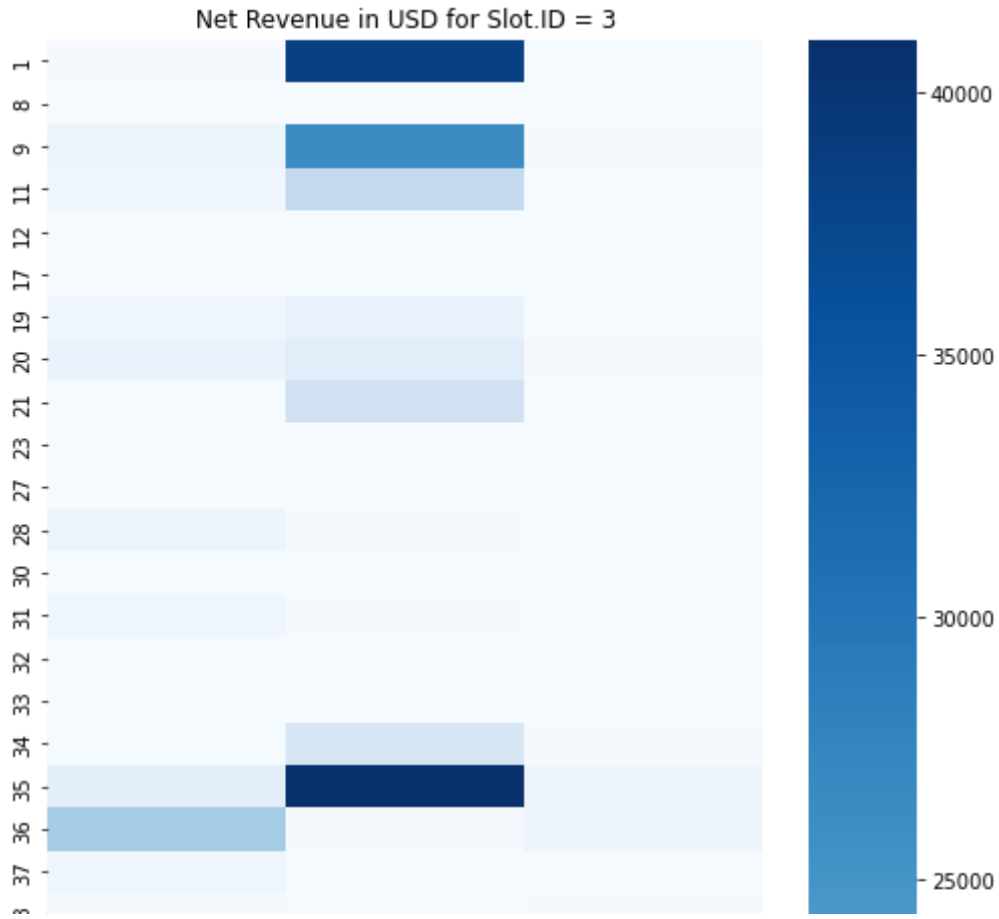
Out[27]:
```
Shape of New Pivot Table
(113289, 3)
```

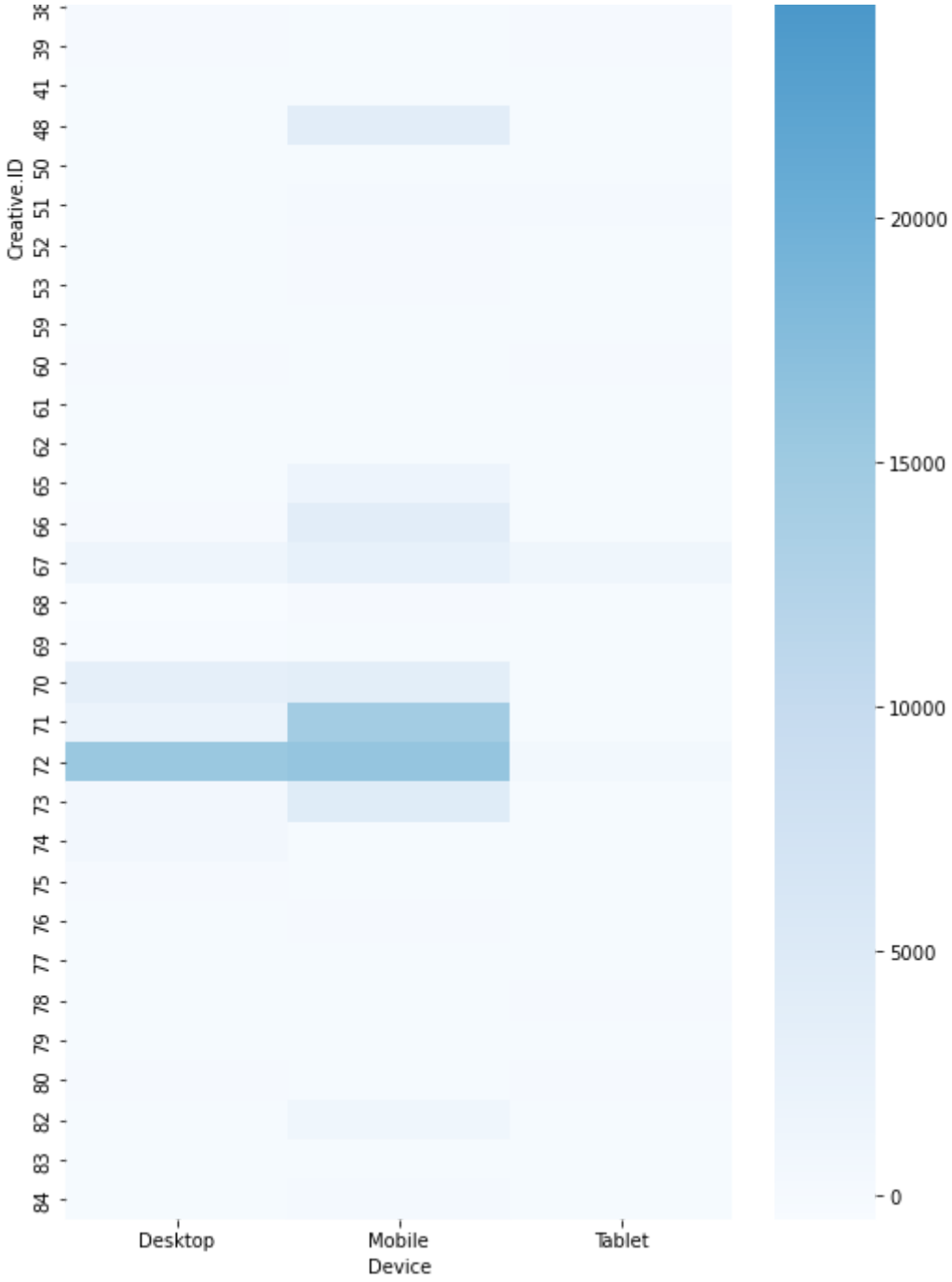
A more useful tool that integrates the Creative.ID, Slot.ID, and Device variables would be a tool that takes a user-selected Slot.ID value as input and returns a pivot table or heat map that illustrates the best strategy for the selected ad unit.

In [28]:
```
# Create function that takes Slot.ID as input and creates a heatmap
def generate_heatmap(slot_id):
    filtered_df = df[df['Slot.ID']==slot_id] #filter df to only contain selected
    pivot_table = create_by_device = pd.pivot_table(df, values='net_revenue',ind
                                                    columns=['Device'], aggfunc=
    pivot_table.drop(columns=['Other'], inplace=True) # Drop 'Other' Column (Emp

    #generate heatmap
    plt.figure(figsize=(8,20))
    sns.heatmap(create_by_device, cmap='Blues')
    plt.title(f'Net Revenue in USD for Slot.ID = {slot_id}')
    plt.show()
```

In [29]:
```
# Test Function
generate_heatmap(3)
```





Deployment

To make use of the above function outside of this notebook, a .py script was created that can be run from the terminal.

The script can be found in the folder labeled deployment.

Note: due to GitHub file size restrictions, this notebook must be successfully run beginning to end before the script is used locally for the first time.

In [30]:
```
# Export Processed Dataframe to be utilized in external python script
df.to_csv('./deployment/processed_data.csv')
```

Conclusions

- Data collected on this campaign can be pivoted into useful and readable summary tables and heat maps that enable any audience to visually audit the campaign's performance.
- When the entire dataset is aggregated and analyzed across all Slot.IDs, Creative IDs perform significantly differently when presented on different devices.
- If data is filtered by Sort.ID prior to pivot table and heat map creation, top performing ad content can be curiated for each slot and for each device.
- The top performing Creative ID across all slots are 72, 35, and 35 when viewed on Desktop, Mobile, and Tablet respectively.

Recommendations

- Use heatmap function to curate ad content for each slot ID.
- Use master heatmap created above to analyze the performance of existing ad content and explore if the ad requires updating or modification to better match the performance of higher ranked creative content.
- Consider retiring lowest performing content and increasing the spread of more successful content.

Future Work

- Explore why some values were missing from initial dataset
- Explore if data aggregation is being skewed by initial post date
- Modify pivot table functions to reflect user defined dates or a range of dates
- Expand python script to take user input for the date being analyzed
- Expand python script to create heatmap for all Slot.IDs or a selection of Slot.IDs if prompted.
- Extrapolate the data to predict future performance with Machine Learning

---