EDA Analysis

Course Name: DSC530-T301

Student Name: Harish Kaparwan

Assignment # : Quiz #3

Professor Name: Dr. Parajulee

Load and transform data.

If you are seeing below issue then you have to make sure your matplotlib version i.e. matplotlib==3.2.0

The number of FixedLocator locations (2), usually from a call to set_ticks, does not match the number of ticklabels (1).

In order to execute below program we need to make sure we ave below package and version.

```
pip show matplotlib
pip install matplotlib==3.2.0
pip show matplotlib
pip install missingno
pip install pandas_profiling
sudo pip install -U joblib==1.0.1
pip install phik
sudo pip install autoviz
sudo pip install typing_extensions
sudo conda install -c pyviz hvplot
sudo pip install typing-extensions==4.1.1
sudo pip install dtale
sudo pip install labellines
sudo pip install sweetviz
sudo pip install --upgrade xarray==2022.3.0
sudo pip install matplotlib-label-lines
```

if you are seeing terminator issue then please see below link:-

https://github.com/ydataai/pandas-profiling/issues/476

if you see error - AttributeError: module 'dask.array' has no attribute 'lib', then please use below steps

pip install --upgrade xarray==2022.3.0

In order to see 'labellines' package you need to follow below steps

sudo pip install matplotlib-label-lines

In [1]:
```
import pandas as pd
import phik
from typing import Tuple, Union, Optional
from phik.binning import auto_bin_data
from phik.phik import phik_from_rebinned_df
import numpy as np
```

In [2]:
```
df = pd.read_csv('data/itunes_data.csv')
#print(df.head())
df['Minutes'] = df['Milliseconds'] / (1000 * 60)
df['MB'] = df['Bytes'] / 1000000
df.drop(['Milliseconds', 'Bytes'], axis=1, inplace=True)
df.info()
df.describe()
#print(df['Minutes'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3503 entries, 0 to 3502
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Track      3503 non-null   object 
 1   Composer   2525 non-null   object 
 2   UnitPrice  3503 non-null   float64
 3   Genre      3503 non-null   object 
 4   Album      3503 non-null   object 
 5   Artist     3503 non-null   object 
 6   Minutes    3503 non-null   float64
 7   MB         3503 non-null   float64
dtypes: float64(3), object(5)
memory usage: 219.1+ KB
```

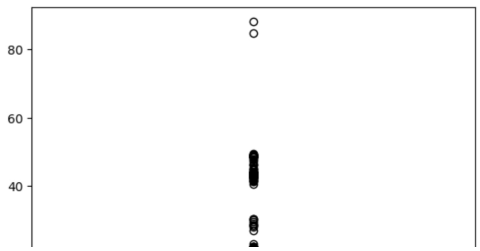
Out[2]:

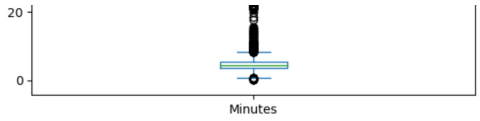
	UnitPrice	Minutes	MB
count	3503.000000	3503.000000	3503.000000
mean	1.050805	6.559987	33.510207
std	0.239006	8.916757	105.392534
min	0.990000	0.017850	0.038747
25%	0.990000	3.454683	6.342566
50%	0.990000	4.260567	8.107896
75%	0.990000	5.360750	10.266789
max	1.990000	88.115883	1059.546140

Boxplots and Boxenplots

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
df['Minutes'].plot.box()
plt.show()
```





In [5]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
#plt.title('Neighbourhood Group')
#plt.pie(data.neighbourhood_group.value_counts(), labels = data.neighbourhood_group.value_counts().index,autopct='%1.1f%%')

df['Minutes'].plot.box()
plt.tight_layout()  # auto-adjust margins
```

This should show up automatially in Jupyter notebooks, if not, try running the magic command `%matplotlib` or `%matplotlib inline` in a code cell.

In [6]:
```
df['Minutes'].plot.box()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f87777f3bb0>
```

In [7]:
```
import seaborn as sns
_ = sns.boxenplot(y=df['Minutes'])
```

In [8]:
```
# plot multiple columns at once
sns.boxenplot(data=df[['Minutes', 'MB']])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f87777f3bb0>
```

In [9]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.boxenplot(y=df['Minutes'])
plt.tight_layout()  # auto-adjust margins
```

To hide the text output, send it to the special variable `_`.

In [10]:
```
sns.boxenplot(y=df['Minutes'])
plt.yscale('log')
```

In [11]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.boxenplot(y=df['Minutes'])
plt.yscale('log')
plt.tight_layout()  # auto-adjust margins
```

In [12]:
```
df['Minutes'].plot.box()
plt.yscale('log')
```

In [13]:
```
# another way to use a log scale
df['Minutes'].plot.box(logy=True)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8777ee1610>
```

In [14]:
```
df['Minutes'].describe()
```

Out[14]:
```
count    3503.000000
mean        6.559987
std         8.916757
min         0.017850
25%         3.454683
50%         4.260567
75%         5.360750
max        88.115883
Name: Minutes, dtype: float64
```

Violin Plots

In [15]:
```
sns.histplot(x=df['Minutes'], kde=True)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8777ee1610>
```

In [16]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.histplot(x=df['Minutes'], kde=True)
plt.tight_layout()  # auto-adjust margins
```

In [17]:
```
sns.violinplot(data=df, x='Minutes')
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8778757be0>
```

In [18]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.violinplot(data=df, x='Minutes')
plt.tight_layout()  # auto-adjust margins
```

In [19]:
```
top_5_genres = df['Genre'].value_counts().index[:5]
top_5_data = data=df[df['Genre'].isin(top_5_genres)]
```

In [20]:
```
sns.violinplot(data=top_5_data, x='Minutes', y='Genre')
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8778767850>
```

In [21]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.violinplot(data=top_5_data, x='Minutes', y='Genre')
plt.tight_layout()  # auto-adjust margins
```

Scatter plots

In [22]:
```
plt.scatter(df['Minutes'], df['MB'])
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x7f8778192eb0>
```

In [23]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
plt.scatter(df['Minutes'], df['MB'])
plt.tight_layout()  # auto-adjust margins
```

In [24]:
```
plt.scatter(df['Minutes'], df['MB'], alpha=0.1)
```

Out[24]:
```
<matplotlib.collections.PathCollection at 0x7f8778767160>
```

In [25]:
```
plt.scatter(df['Minutes'], df['MB'])
plt.xlabel('Minutes')
plt.ylabel('MB')
```

Out[25]:
```
Text(9.444444444444452, 0.5, 'MB')
```

In [26]:
```
plt.scatter(df['Minutes'], df['MB'])
plt.xlabel('Minutes')
plt.ylabel('MB')
```

Out[26]:
```
Text(9.444444444444452, 0.5, 'MB')
```

In [27]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.scatterplot(data=df, x='Minutes', y='MB')
plt.tight_layout()  # auto-adjust margins
```

In [28]:
```
sns.scatterplot(data=df, x='Minutes', y='MB')
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8778dd4970>
```

In [29]:
```
sns.scatterplot(data=top_5_data, x='Minutes', y='MB', hue='Genre')
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8778dd4970>
```

In [30]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.scatterplot(data=top_5_data, x='Minutes', y='MB', hue='Genre')
plt.tight_layout()  # auto-adjust margins
```

Pairplots and Correlograms

In [31]:
```
sns.pairplot(data=df)
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x7f87784935e0>
```

In [32]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.pairplot(data=df)
plt.tight_layout()  # auto-adjust margins
```

In [33]:
```
sns.heatmap(df.corr(), annot=True)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f875b7ef0d0>
```

In [34]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.heatmap(df.corr(), annot=True)
plt.tight_layout()  # auto-adjust margins
```

In [35]:
```
sns.heatmap(df.corr(method='spearman'), annot=True)
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f875d4700a0>
```

In [36]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image -- more on this is chapter 5
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.heatmap(df.corr(method='spearman'), annot=True)
plt.tight_layout()  # auto-adjust margins
```

Missing value plot

In [37]:
```
import missingno as msno
msno.matrix(df)
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f875da4c2b0>
```

In [38]:
```
# save figure for book
f = msno.matrix(df, figsize=(5.5, 5.5))
f.patch.set_facecolor('w')  # sets background color behind axis labels
f2 = f.get_figure()
```

EDA packages - pandas-profiling

Install pandas profiling with pip instead of conda - problems with conda version

In [39]:
```
import pandas_profiling
pandas_profiling.__version__

# Same as phik.phik_matrix except for the default value of njobs
def phik_matrix_nJobsDefVal(
    df: pd.DataFrame,
    interval_cols: Optional[list] = None,
    bins: Union[int, list, np.ndarray, dict] = 10,
    quantile: bool = False,
    noise_correction: bool = True,
    dropna: bool = True,
    drop_underflow: bool = True,
    drop_overflow: bool = True,
    verbose: bool = True,
    njobs: int = 1,
) -> pd.DataFrame:
    """
    Correlation matrix of bivariate gaussian derived from chi2-value
    Chi2-value gets converted into correlation coefficient of bivariate gauss
    with correlation value rho, assuming giving binning and number of records.
    Correlation coefficient value is between 0 and 1.
    Bivariate gaussian's range is set to [-5,5] by construction.
    :param pd.DataFrame data_binned: input data
    :param list interval_cols: column names of columns with interval variables.
    :param bins: number of bins, or a list of bin edges (same for all columns), or a dictionary where per column the bins are specified. (default=10)\
    E.g.: bins = {'mileage':5, 'driver_age':[18,25,35,45,55,65,125]}
    :param quantile: when bins is an integer, uniform bins (False) or bins based on quantiles (True)
    :param bool noise_correction: apply noise correction in phik calculation
    :param bool dropna: remove NaN values with True
    :param bool drop_underflow: do not take into account records in underflow bin when True (relevant when binning\
    a numeric variable)
    :param bool drop_overflow: do not take into account records in overflow bin when True (relevant when binning\
    a numeric variable)
    :param bool verbose: if False, do not print all interval columns that are guessed
    :param int njobs: number of parallel jobs used for calculation of phik. default is -1. 1 uses no parallel jobs.
    :return: phik correlation matrix
    """

    data_binned, binning_dict = auto_bin_data(
        df=df,
        interval_cols=interval_cols,
        bins=bins,
        quantile=quantile,
        dropna=dropna,
        verbose=verbose,
    )
    return phik_from_rebinned_df(
        data_binned,
        noise_correction,
        dropna=dropna,
        drop_underflow=drop_underflow,
        drop_overflow=drop_overflow,
        njobs=njobs,
    )
```

```
    )

phik.phik_matrix = phik_matrix_nJobsDefVal
```

In [40]:
```
from pandas_profiling import ProfileReport
```

In [41]:
```
report = ProfileReport(df)
```

In [42]:
```
report.to_widgets()
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=5.0), HTML(value='')))

HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0, max=1.0), HTML(value='')))

HBox(children=(HTML(value='Render widgets'), FloatProgress(value=0.0, max=1.0), HTML(value='')))
VBox(children=(Tab(children=(Tab(children=(GridBox(children=(VBox(children=(GridspecLayout(children=(HTML(valu…
```

In [43]:
```
# create our own histogram of the length of the tracks
df['Track'].str.len().plot.hist(bins=50)
```

Out[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f874c327790>
```

In [44]:
```
import numpy as np
df_log = df.copy()
df_log['log(Minutes)'] = np.log(df_log['Minutes'])
df_log['log(MB)'] = np.log(df_log['MB'])
sns.jointplot(x="log(Minutes)", y="log(MB)", data=df_log, kind="hex")
```

Out[44]:
```
<seaborn.axisgrid.JointGrid at 0x7f8766d4ed00>
```

In [45]:
```
# saving figure for book
sns.jointplot(x="log(Minutes)", y="log(MB)", data=df_log, kind="hex")
plt.tight_layout()  # auto-adjust margins
```

In [46]:
```
report.to_notebook_iframe()
```

```
HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

Pandas Profiling Report

Overview | Variables | Interactions | Correlations | Missing values | Sample

8	Night Of The Long Knives	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.428
9	Spellbound	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	4.514

Last rows

	Track	Composer	UnitPrice	Genre	Album
3493	Symphony No. 2, Op. 16 - "The Four Temperaments": II. Allegro Comodo e Flemmatico	Carl Nielsen	0.99	Classical	Nielsen: The Six Sympl
3494	24 Caprices, Op. 1, No. 24, for Solo Violin, in A Minor	Niccolò Paganini	0.99	Classical	Great Recordings of th
3495	Étude 1, In C Major - Preludio (Presto) - Liszt	NaN	0.99	Classical	Liszt - 12 Études D'Exe
3496	Erlkonig, D.328	NaN	0.99	Classical	Great Recordings of th
3497	Concerto for Violin, Strings and Continuo in G Major, Op. 3, No. 9: I. Allegro	Pietro Antonio Locatelli	0.99	Classical	Locatelli: Concertos fo
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della Via Appia	NaN	0.99	Classical	Respighi:Pines of Rom
3499	String Quartet No. 12 in C Minor, D. 703 "Quartettsatz": II. Andante - Allegro assai	Franz Schubert	0.99	Classical	Schubert: The Late Stri
3500	L'orfeo, Act 3, Sinfonia (Orchestra)	Claudio Monteverdi	0.99	Classical	Monteverdi: L'Orfeo
3501	Quintet for Horn, Violin, 2 Violas, and Cello in E Flat Major, K. 407/386c: III. Allegro	Wolfgang Amadeus Mozart	0.99	Classical	Mozart: Chamber Musi
3502	Koyaanisqatsi	Philip Glass	0.99	Soundtrack	Koyaanisqatsi (Soundtr

Report generated with pandas-profiling.

Other EDA packages: autoviz, sweetviz, d-tale

In [47]:
```
from autoviz.AutoViz_Class import AutoViz_Class
```

```
Alert! from version 0.1.43, after importing, you must do '%matplotlib inline' to display charts in Jupyter Notebooks.
    AV = AutoViz_Class()
    dfte = AV.AutoViz(filename, sep=',', depVar='', dfte=None, header=0, verbose=0, lowess=False,
                chart_format='svg',max_rows_analyzed=150000,max_cols_analyzed=30, save_plot_dir=None)
Note: verbose=0 or 1 generates charts and displays them in your local Jupyter notebook.
      verbose=2 does not display plots but saves them in AutoViz_Plots folder in local machine.
Updated: chart_format='bokeh' generates and displays charts in your local Jupyter notebook.
      chart_format='server' generates and displays charts in the browser - one tab for each chart.
      chart_format='html' silently saves charts HTML format - they are also interactive!
```

In [48]:
```
# the documentation is not great for autoviz, but you can use a dataframe like so
AutoViz_Class().AutoViz(filename="", dfte=df)
```

```
Shape of your Data Set loaded: (3503, 8)
############################################################################
###################### C L A S S I F Y I N G  V A R I A B L E S  ############################
############################################################################
Classifying variables in data set...
    8 Predictors classified...
        No variables removed since no ID or low-information variables found in data set
Number of All Scatter Plots = 3

**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/Users/harishkaparwan/nltk_data'
    - '/opt/anaconda3/nltk_data'
    - '/opt/anaconda3/share/nltk_data'
    - '/opt/anaconda3/lib/nltk_data'
    - '/usr/share/nltk_data'
```

```
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************

Could not draw wordcloud plot for Track

**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/Users/harishkaparwan/nltk_data'
    - '/opt/anaconda3/nltk_data'
    - '/opt/anaconda3/share/nltk_data'
    - '/opt/anaconda3/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************

Could not draw wordcloud plot for Composer

**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/Users/harishkaparwan/nltk_data'
    - '/opt/anaconda3/nltk_data'
    - '/opt/anaconda3/share/nltk_data'
    - '/opt/anaconda3/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************

Could not draw wordcloud plot for Album

**********************************************************************
  Resource punkt not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('punkt')

  For more information see: https://www.nltk.org/data.html

  Attempted to load tokenizers/punkt/PY3/english.pickle

  Searched in:
    - '/Users/harishkaparwan/nltk_data'
    - '/opt/anaconda3/nltk_data'
    - '/opt/anaconda3/share/nltk_data'
    - '/opt/anaconda3/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
    - ''
**********************************************************************

Could not draw wordcloud plot for Artist
All Plots done
Time to run AutoViz = 1 seconds

 ###################### AUTO VISUALIZATION Completed ######################
```

Out[48]:

	Track	Composer	UnitPrice	Genre	Album	Artist	Minutes	MB
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	5.728650	11.170334
1	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.427700	6.713451
2	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.898767	7.636561
3	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.513900	6.852860
4	Snowballed	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.385033	6.599424
...	...	...	...	...	...	...	...	...
3498	Pini Di Roma (Pinien Von Rom) \ I Pini Della Via Appia	NaN	0.99	Classical	Respighi:Pines of Rome	Eugene Ormandy	4.779017	4.718950
3499	String Quartet No. 12 in C Minor, D. 703 "Quartettsatz": II. Andante - Allegro assai	Franz Schubert	0.99	Classical	Schubert: The Late String Quartets & String Quintet (3 CD's)	Emerson String Quartet	2.320000	2.283131
3500	L'orfeo, Act 3, Sinfonia (Orchestra)	Claudio Monteverdi	0.99	Classical	Monteverdi: L'Orfeo	C. Monteverdi, Nigel Rogers - Chiaroscuro; London Baroque; London Cornett & Sackbu	1.110650	1.189062
3501	Quintet for Horn, Violin, 2 Violas, and Cello in E Flat Major, K. 407/386c: III. Allegro	Wolfgang Amadeus Mozart	0.99	Classical	Mozart: Chamber Music	Nash Ensemble	3.688850	3.665114
3502	Koyaanisqatsi	Philip Glass	0.99	Soundtrack	Koyaanisqatsi (Soundtrack from the Motion Picture)	Philip Glass Ensemble	3.433417	3.305164

3503 rows × 8 columns

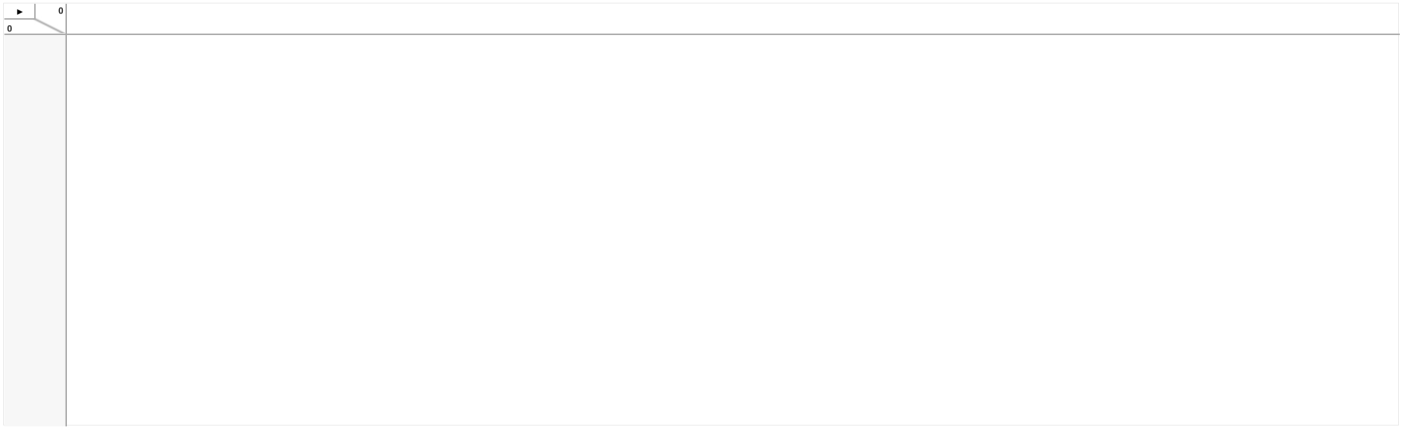
Sweetviz

In [49]:
```
import sweetviz as sv

sv = sv.analyze(df)
sv.show_notebook()
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, layout=Layout(flex='2'), max=9.0), HTML(value='')), la…
```

In [50]:
```
import dtale
import dask.array as da
dtale.show(df)
```



Out[50]:

Visualization best practices

In [51]:
```
df['Genre'].value_counts()[:5].plot.bar()
plt.xlabel('Genre')
```

Out[51]:
```
Text(0.5, 0, 'Genre')
```

In [52]:
```
df['Genre'].value_counts()[:5]
```

Out[52]:
```
Rock                  1297
Latin                  579
Metal                  374
Alternative & Punk     332
Jazz                   130
Name: Genre, dtype: int64
```

In [53]:
```
sns.countplot(y='Genre', data=df, order=df['Genre'].value_counts().index[:5], color='darkblue')
```

Out[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873c8c0550>
```

In [54]:
```
sns.countplot(y='Genre', data=df, order=df['Genre'].value_counts().index[:5])
plt.tight_layout()
```

Choosing the right method for plotting

In [55]:
```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///data/chinook.db')

with engine.connect() as connection:
    sql_df = pd.read_sql_table('invoices', connection)
```

In [56]:
```
sql_df.head()
```

Out[56]:

	InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
0	1	2	2009-01-01	Theodor-Heuss-Straße 34	Stuttgart	None	Germany	70174	1.98
1	2	4	2009-01-02	Ullevålsveien 14	Oslo	None	Norway	0171	3.96
2	3	8	2009-01-03	Grétrystraat 63	Brussels	None	Belgium	1000	5.94
3	4	14	2009-01-06	8210 111 ST NW	Edmonton	AB	Canada	T6G 2C7	8.91
4	5	23	2009-01-11	69 Salem Street	Boston	MA	USA	2113	13.86

In [57]:
```
sql_df.groupby('BillingCountry').sum().sort_values(by='Total', ascending=False)[:3]
```

Out[57]:

BillingCountry	InvoiceId	CustomerId	Total
USA	19103	2002	523.06
Canada	11963	1309	303.96
France	7168	1435	195.10

In [58]:
```
top_3_countries = sql_df.groupby('BillingCountry').sum().sort_values(by='Total', ascending=False)[:3].index.values
```

In [59]:
```
top 3 countries
```

In [59]:
```
array(['USA', 'Canada', 'France'], dtype=object)
```

In [60]:
```
sql_df.set_index('InvoiceDate', inplace=True)
```

In [61]:
```
gb = sql_df[sql_df['BillingCountry'].isin(top_3_countries)]. \
        groupby([pd.Grouper(freq='M'), 'BillingCountry']).sum(). \
        groupby(level=-1).cumsum()
```

In [62]:
```
gb
```

Out[62]:

InvoiceDate	BillingCountry	InvoiceId	CustomerId	Total
2009-01-31	**Canada**	4	14	8.91
	USA	5	23	13.86
2009-02-28	**France**	17	82	5.94
	USA	18	39	14.85
2009-03-31	**Canada**	22	45	17.82
...	**...**	...	...	...
2013-10-31	**USA**	17477	1913	514.15
2013-11-30	**France**	7168	1435	195.10
	USA	17882	1933	515.14
2013-12-31	**Canada**	11963	1309	303.96
	USA	19103	2002	523.06

108 rows × 3 columns

In [63]:
```
gb.reset_index(inplace=True)
```

In [64]:
```
gb.head()
```

Out[64]:

	InvoiceDate	BillingCountry	InvoiceId	CustomerId	Total
0	2009-01-31	Canada	4	14	8.91
1	2009-01-31	USA	5	23	13.86
2	2009-02-28	France	17	82	5.94
3	2009-02-28	USA	18	39	14.85
4	2009-03-31	Canada	22	45	17.82

In [65]:
```
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry')
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873c8c0550>
```

In [66]:
```
# saving figure for book
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry')
plt.tight_layout()
```

Making plots redundant

In [67]:
```
# black and white redundancy
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', style='BillingCountry')
```

Out[67]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873c8c0550>
```

In [68]:
```
# save figure for book
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', style='BillingCountry')
plt.tight_layout()
```

In [69]:
```
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', style='BillingCountry', dashes=[(2, 1), (5, 2), ''])
```

Out[69]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873c8c0550>
```

In [70]:
```
from labellines import labelLines
```

In [71]:
```
f = plt.figure()
ax = f.gca()
for country in top_3_countries:
    c_df = gb[gb['BillingCountry'] == country]
    ax.plot(c_df['InvoiceDate'], c_df['Total'], label=country)

labelLines(ax.get_lines())

plt.xlabel('Year')
plt.ylabel('Cumulative Sales ($)')

plt.tight_layout()
```

In [72]:
```
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', size='BillingCountry', sizes=[1, 5, 1])
```

Out[72]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873d3c8310>
```

Another way to collect and plot the same data

In [73]:
```
top_3_df_gb = sql_df[sql_df['BillingCountry'].isin(top_3_countries)].groupby([pd.Grouper(freq='M'), 'BillingCountry']).sum()
dfs = []
for country in top_3_countries:
    c_df = top_3_df_gb.xs(country, level=1).cumsum()
    c_df['Country'] = country
    dfs.append(c_df)

full_c_df = pd.concat(dfs)
```

In [74]:
```
full_c_df.head()
```

Out[74]:

InvoiceDate	InvoiceId	CustomerId	Total	Country
2009-01-31	5	23	13.86	USA
2009-02-28	18	39	14.85	USA
2009-03-31	80	121	28.71	USA
2009-04-30	106	140	42.57	USA
2009-06-30	220	205	61.38	USA

In [75]:
```
full_c_df.reset_index(inplace=True)
```

In [76]:
```
sns.lineplot(data=full_c_df, x='InvoiceDate', y='Total', hue='Country')
```

Out[76]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f873d3c8310>
```

Here is a start towards how we could collect the data using a SQL query.

We would need to either modify the SQL query to join the same data for the top 3 countries, or do 3 separate SQL queries and join the resulting dataframes.

In [77]:
```
engine = create_engine('sqlite:///data/chinook.db')

query = """SELECT SUM(Total) OVER (ORDER BY InvoiceId) as Total, strftime("%m-%Y", InvoiceDate) as 'month-year', BillingCountry
FROM invoices
WHERE BillingCountry="USA"
GROUP BY strftime("%m-%Y", InvoiceDate);
"""

with engine.connect() as connection:
    sql_df2 = pd.read_sql_query(query, connection)
```

In [78]:
```
sql_df2.head()
```

Out[78]:

	Total	month-year	BillingCountry
0	13.86	01-2009	USA
1	14.85	02-2009	USA
2	16.83	03-2009	USA
3	30.69	04-2009	USA
4	34.65	06-2009	USA

In [79]:
```
# convert to datetime for better plotting
sql_df2['month-year'] = pd.to_datetime(sql_df2['month-year'])
```

In [80]:
```
sns.lineplot(data=sql_df2, x='month-year', y='Total', hue='BillingCountry')
plt.xlabel('Year')
```

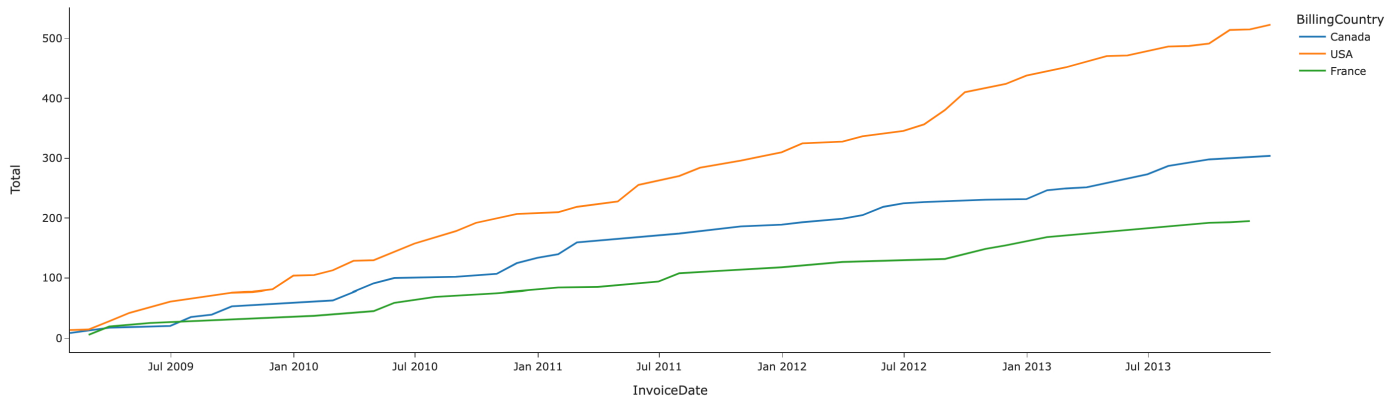
Out[80]:
```
Text(0.5, 39.00000000000002, 'Year')
```

Saving Images

In [81]:
```
f = plt.figure(figsize=(5, 5))
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry')
plt.tight_layout()
plt.savefig('cumulative_sales_lineplot.png', facecolor='w', dpi=300)
```

Plotly

In [82]:
```
import plotly.express as px
px.line(gb, x='InvoiceDate', y='Total', color='BillingCountry', template='simple_white')
```

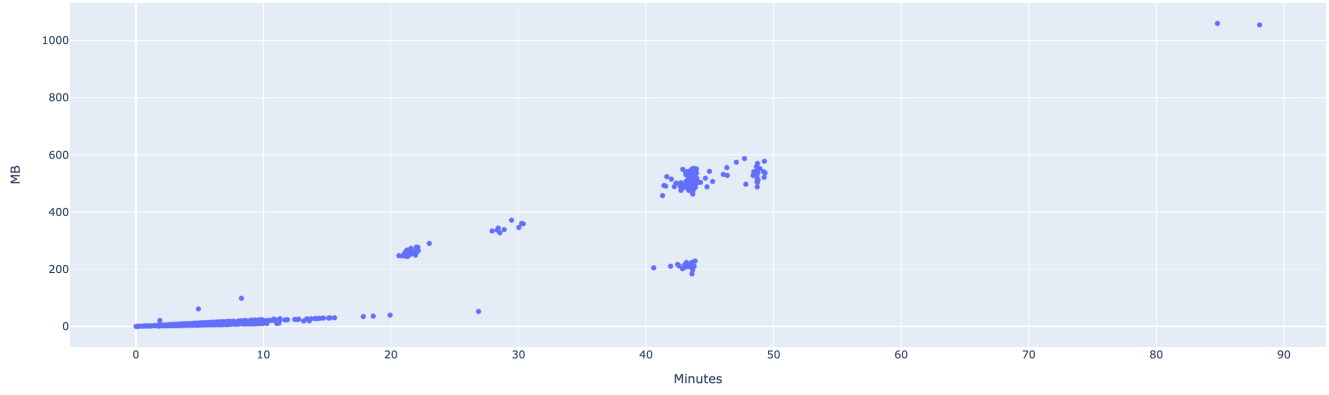


In [83]:
```
# recall our original DataFrame has information on the tracks in our iTunes data
df.head()
```

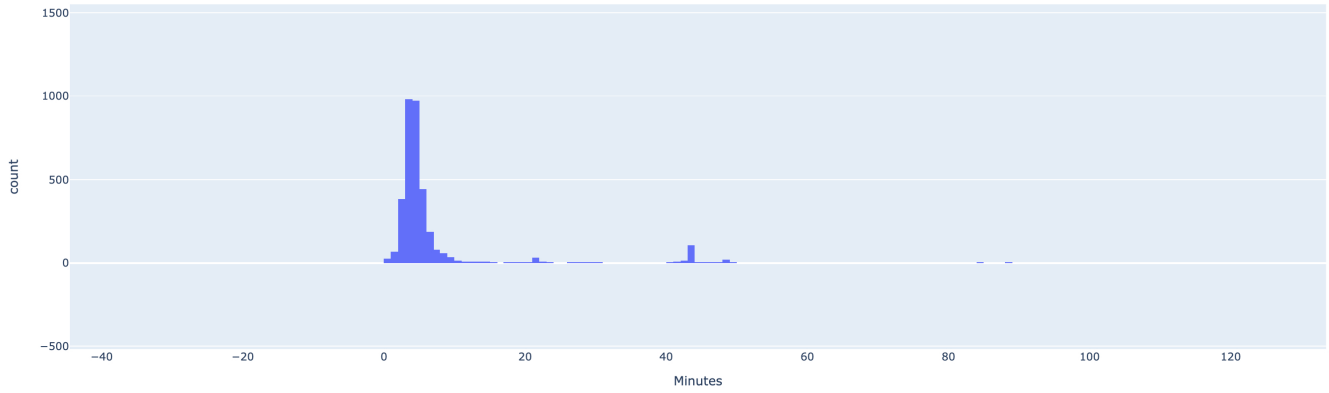
Out[83]:

	Track	Composer	UnitPrice	Genre	Album	Artist	Minutes	MB
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	5.728650	11.170334
1	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.427700	6.713451
2	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.898767	7.636561
3	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.513900	6.852860
4	Snowballed	Angus Young, Malcolm Young, Brian Johnson	0.99	Rock	For Those About To Rock We Salute You	AC/DC	3.385033	6.599424

In [84]:
```
px.scatter(df, x='Minutes', y='MB')
```



In [85]:
```
px.histogram(df, x='Minutes')
```



In [86]:
```
# recall this is our invoices table from the chinook iTunes database
sql_df.head()
```

Out[86]:

InvoiceDate	InvoiceId	CustomerId	BillingAddress	BillingCity	BillingState	BillingCountry	BillingPostalCode	Total
2009-01-01	1	2	Theodor-Heuss-Straße 34	Stuttgart	None	Germany	70174	1.98
2009-01-02	2	4	Ullevålsveien 14	Oslo	None	Norway	0171	3.96
2009-01-03	3	8	Grétrystraat 63	Brussels	None	Belgium	1000	5.94
2009-01-06	4	14	8210 111 ST NW	Edmonton	AB	Canada	T6G 2C7	8.91
2009-01-11	5	23	69 Salem Street	Boston	MA	USA	2113	13.86

In [87]:
```
gb_countries = sql_df.groupby('BillingCountry').sum()
gb_countries.reset_index(inplace=True)
gb_countries.head()
```

Out[87]:

	BillingCountry	InvoiceId	CustomerId	Total
0	Argentina	1729	392	37.62
1	Australia	1043	385	37.62
2	Austria	1568	49	42.62
3	Belgium	1428	56	37.62
4	Brazil	7399	329	190.10

In [88]:
```
px.choropleth(gb_countries, locations="BillingCountry",
              locationmode='country names',
              color="Total")
```