EDA

Load and transform data.

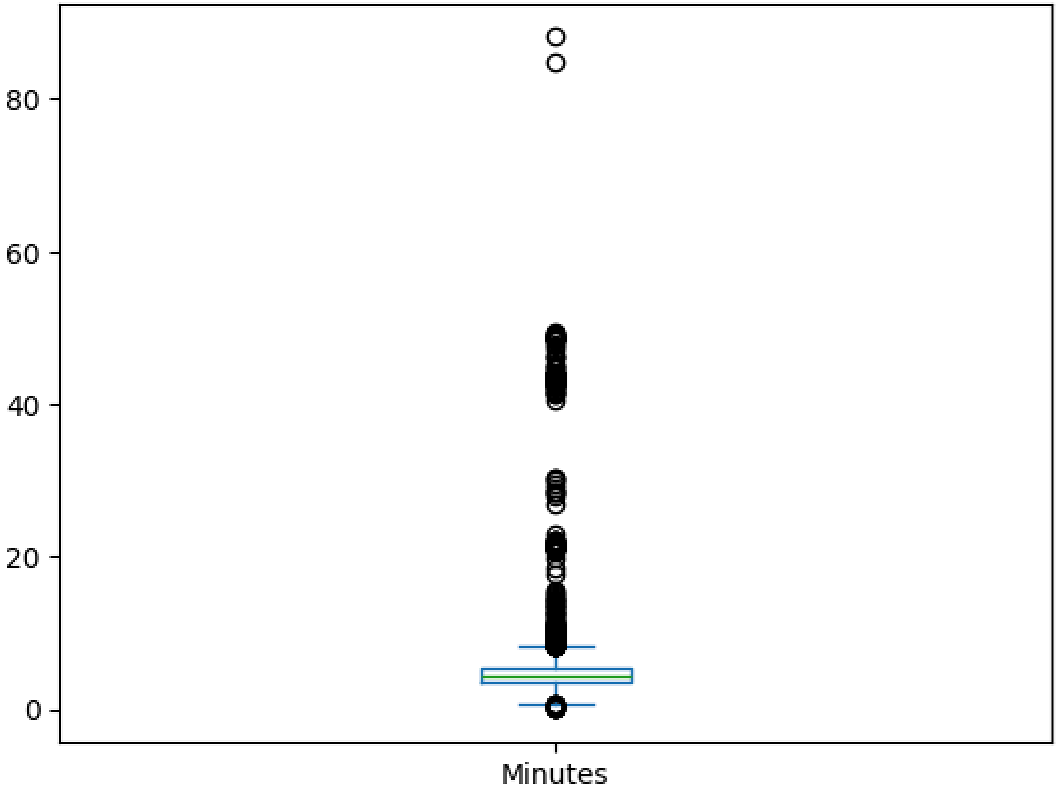
```
In [1]: import pandas as pd
```

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

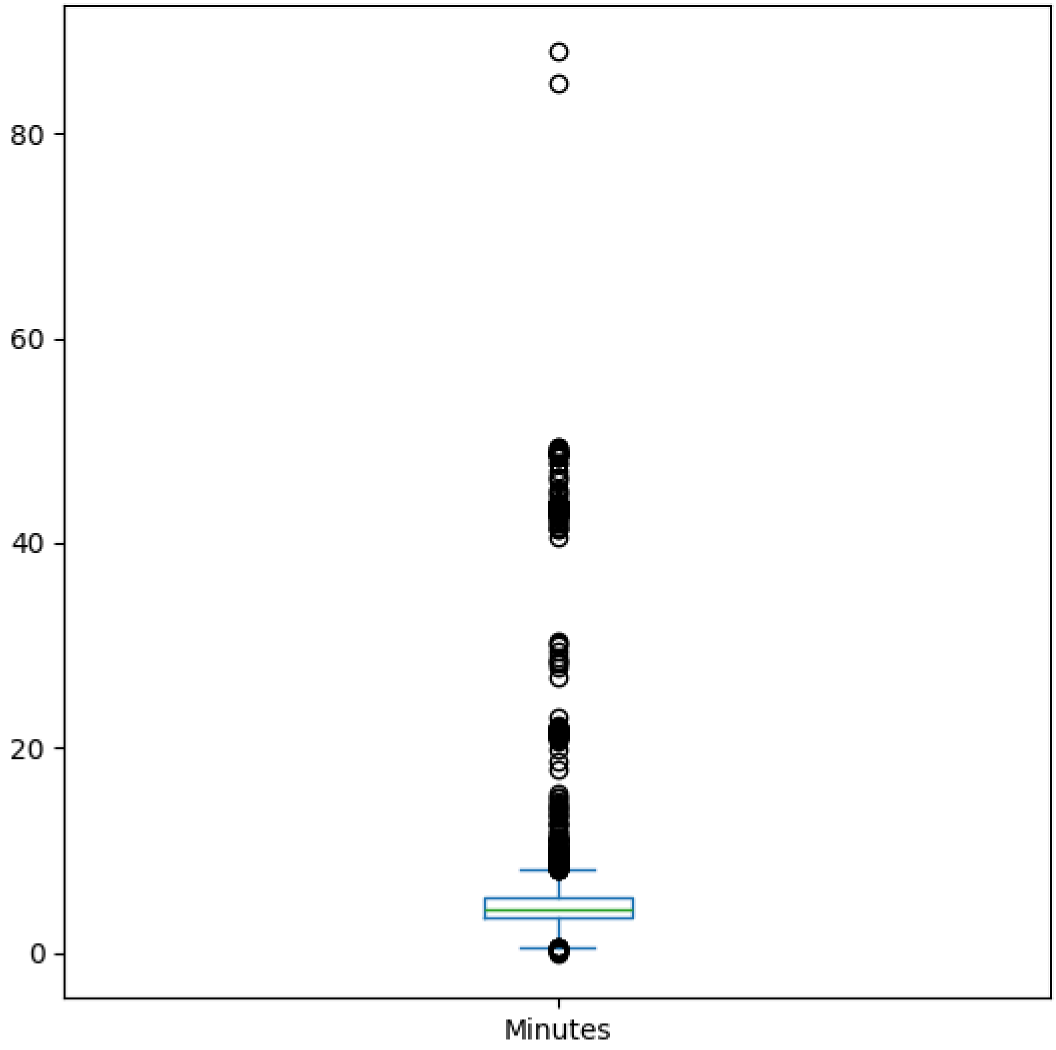
Boxplots and Boxenplots

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

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



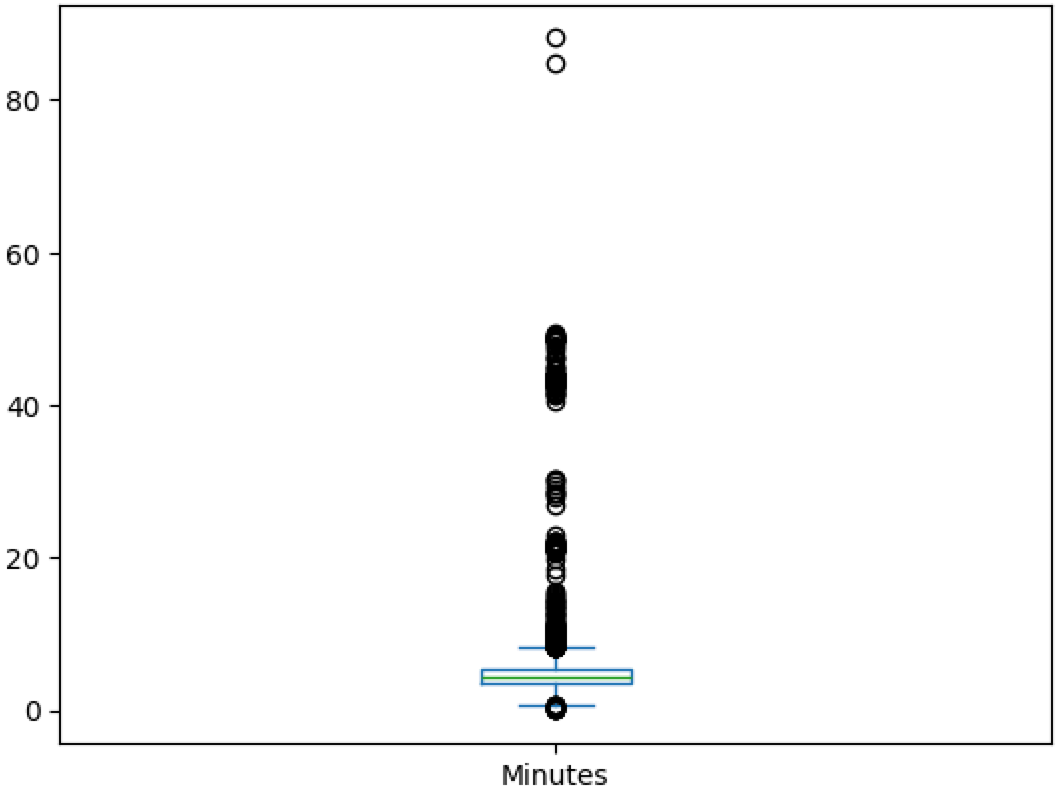
```
In [5]: # save figure for book
        f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
        f.patch.set_facecolor('w')  # sets background color behind axis labels
        df['Minutes'].plot.box()
        plt.tight_layout()  # auto-adjust margins
```



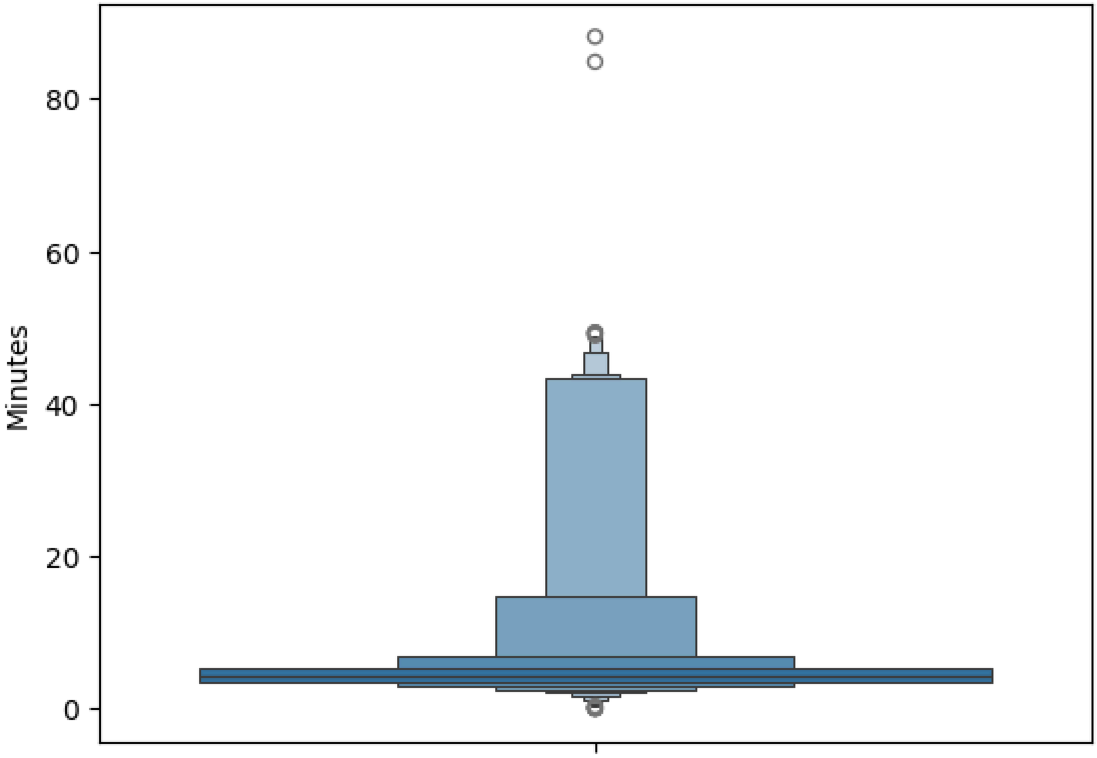
This should show up automatially in Jupyter notebooks, if not, try running the magic command `%matplotlib` or `%matplotlib inline` in a code cell.

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

```
Out[6]: <Axes: >
```

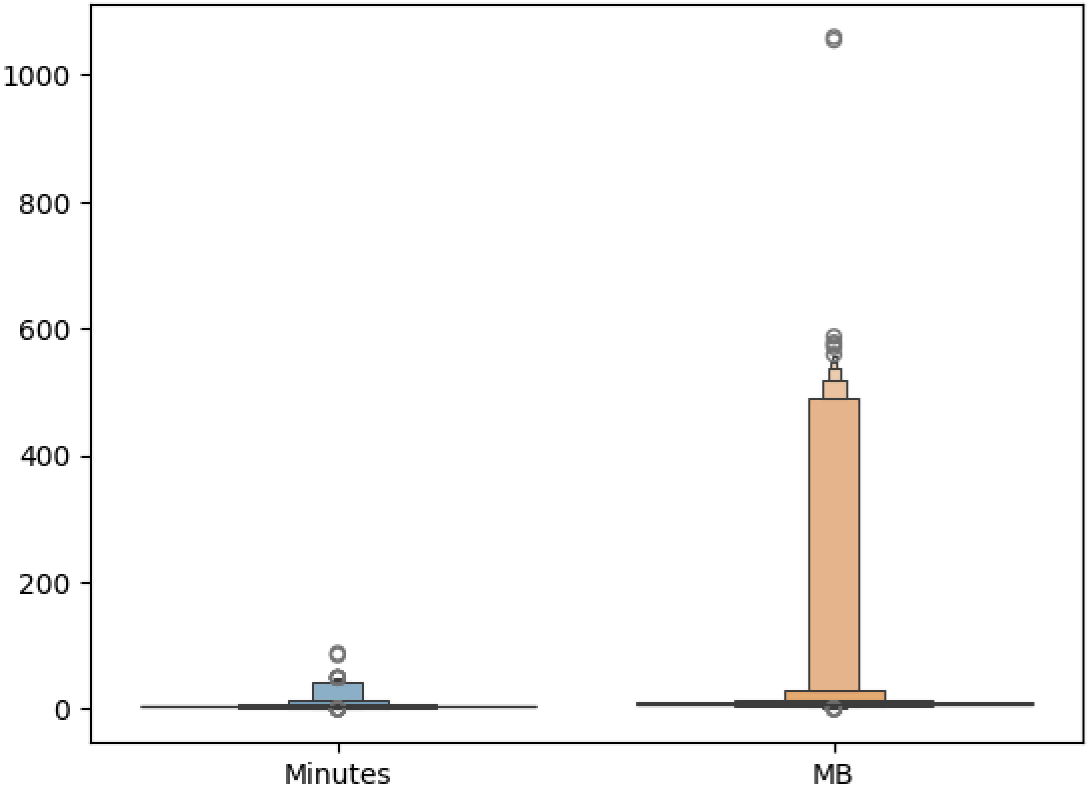


In [7]:
```
import seaborn as sns
_ = sns.boxenplot(y=df['Minutes'])
```

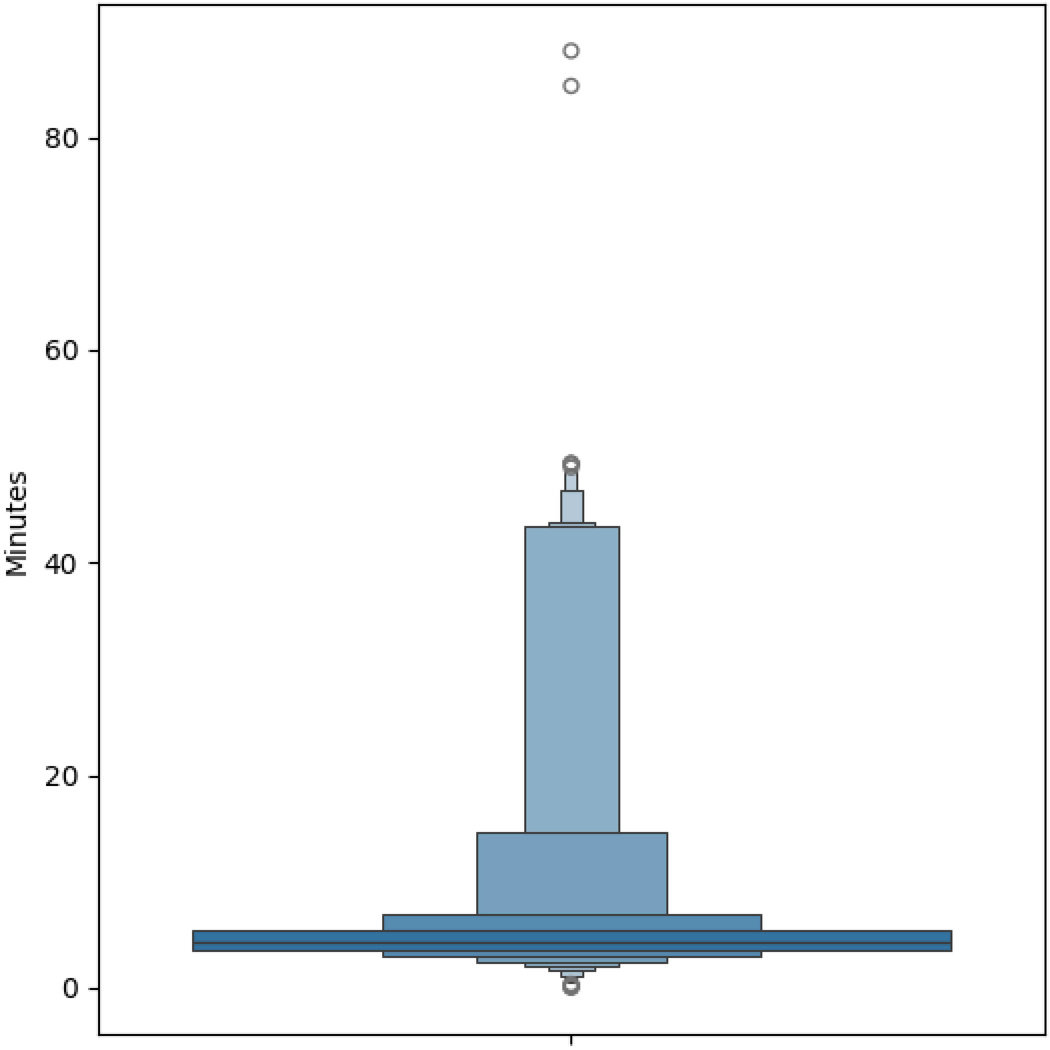


In [8]:
```
# plot multiple columns at once
sns.boxenplot(data=df[['Minutes', 'MB']])
```

```
Out[8]: <Axes: >
```

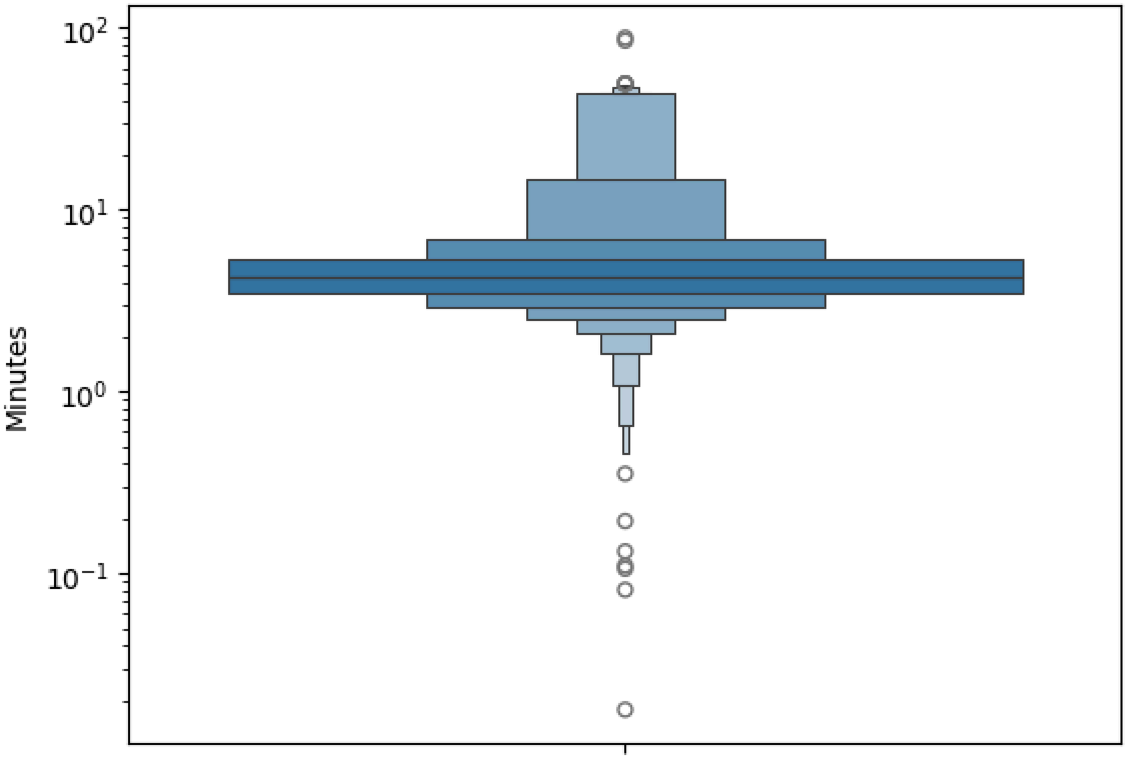


```
In [9]: # save figure for book
        f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
        f.patch.set_facecolor('w')  # sets background color behind axis labels
        sns.boxenplot(y=df['Minutes'])
        plt.tight_layout()  # auto-adjust margins
```

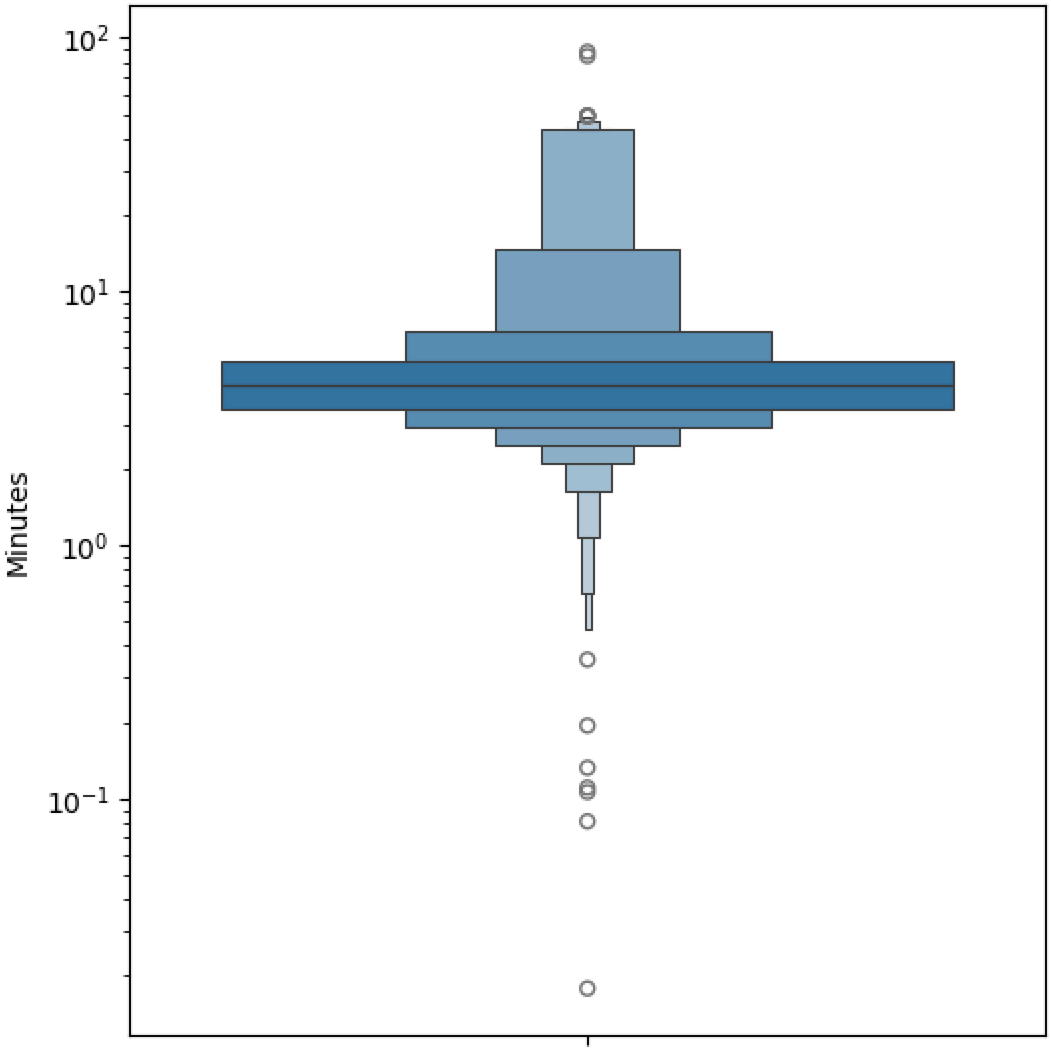


To hide the text output, send it to the special variable `_`.

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

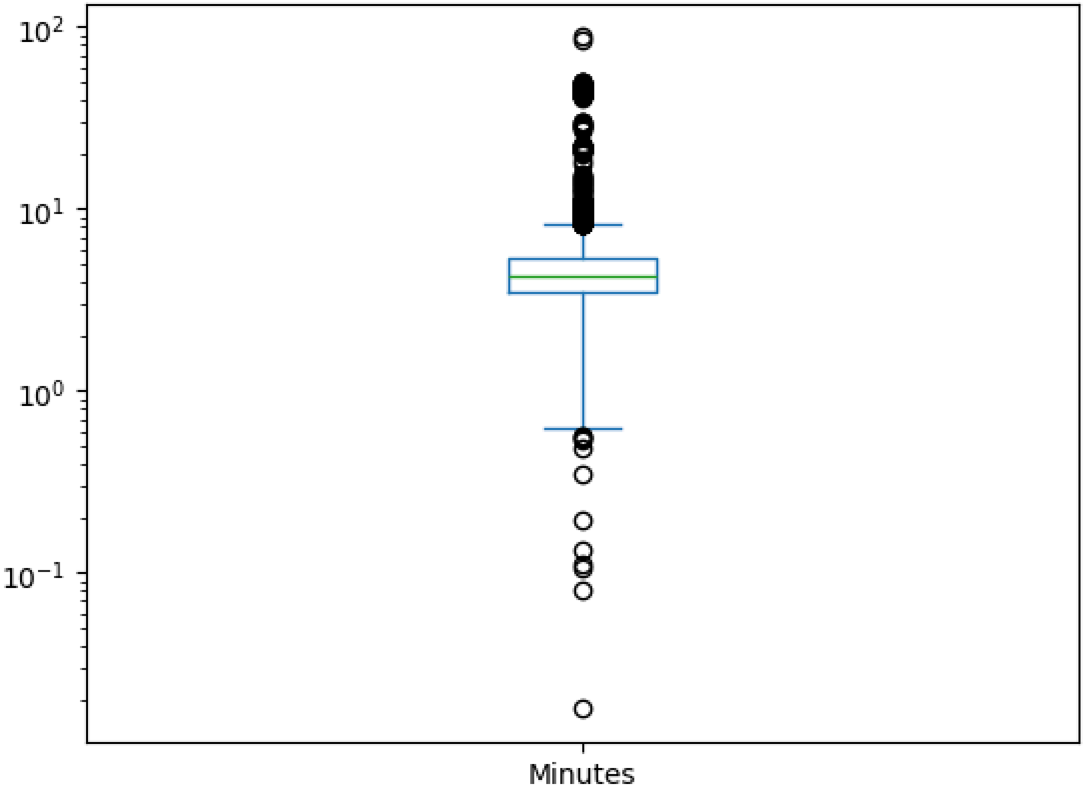


In [11]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.boxenplot(y=df['Minutes'])
plt.yscale('log')
plt.tight_layout()  # auto-adjust margins
```



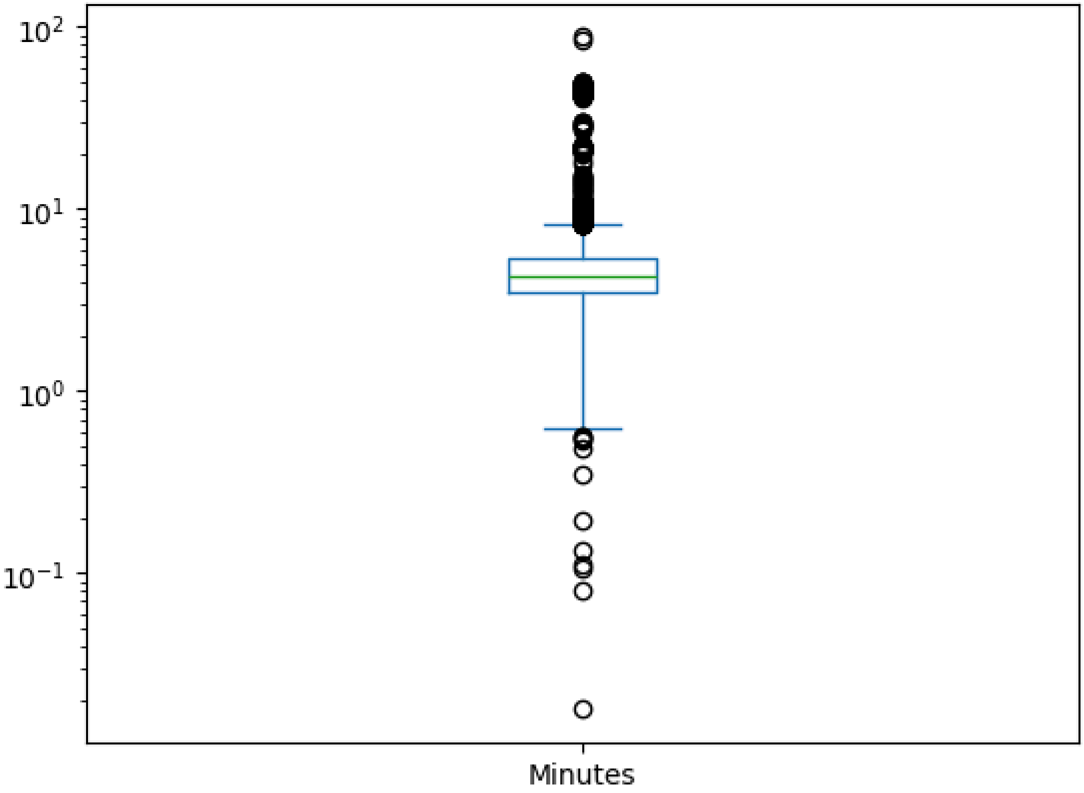
In [12]:

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



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

Out[13]:
```
<Axes: >
```



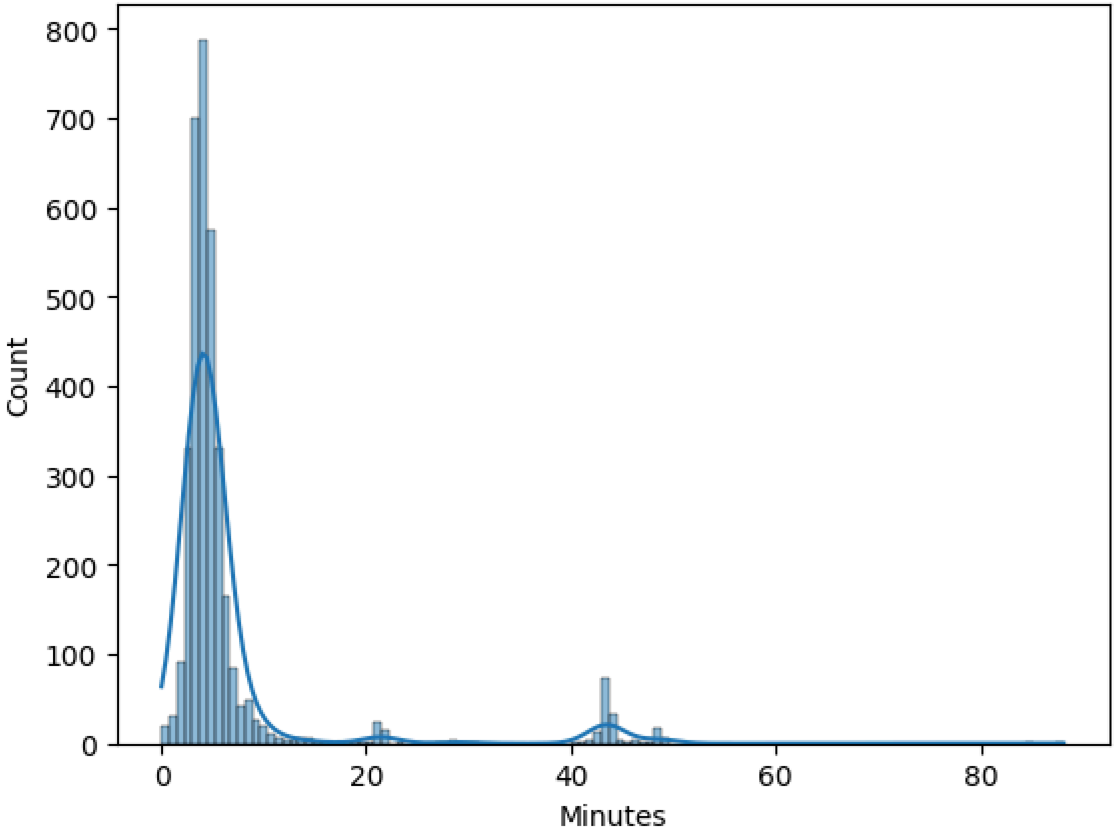
In [14]:
```
df['Minutes'].describe()
```

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
```

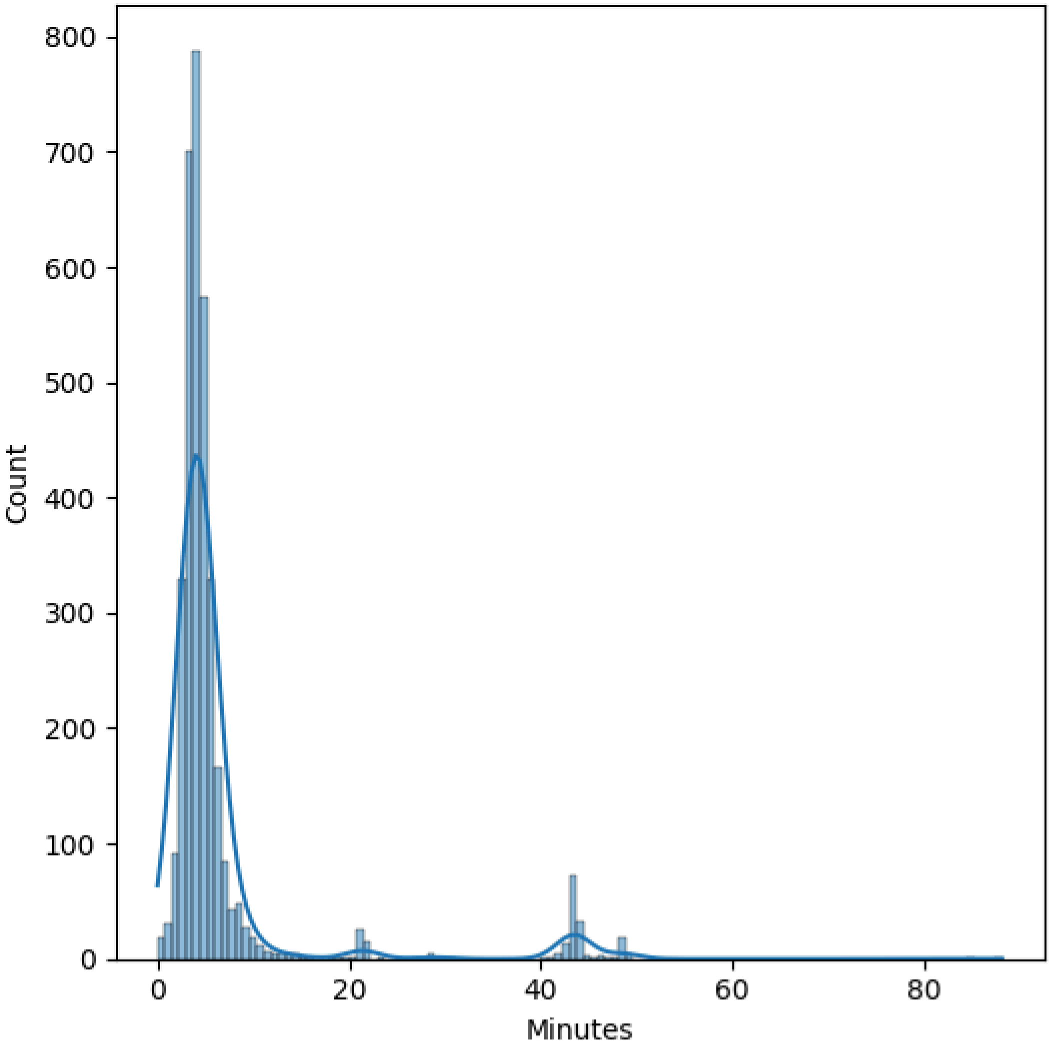
Violin Plots

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

Out[15]:
```
<Axes: xlabel='Minutes', ylabel='Count'>
```

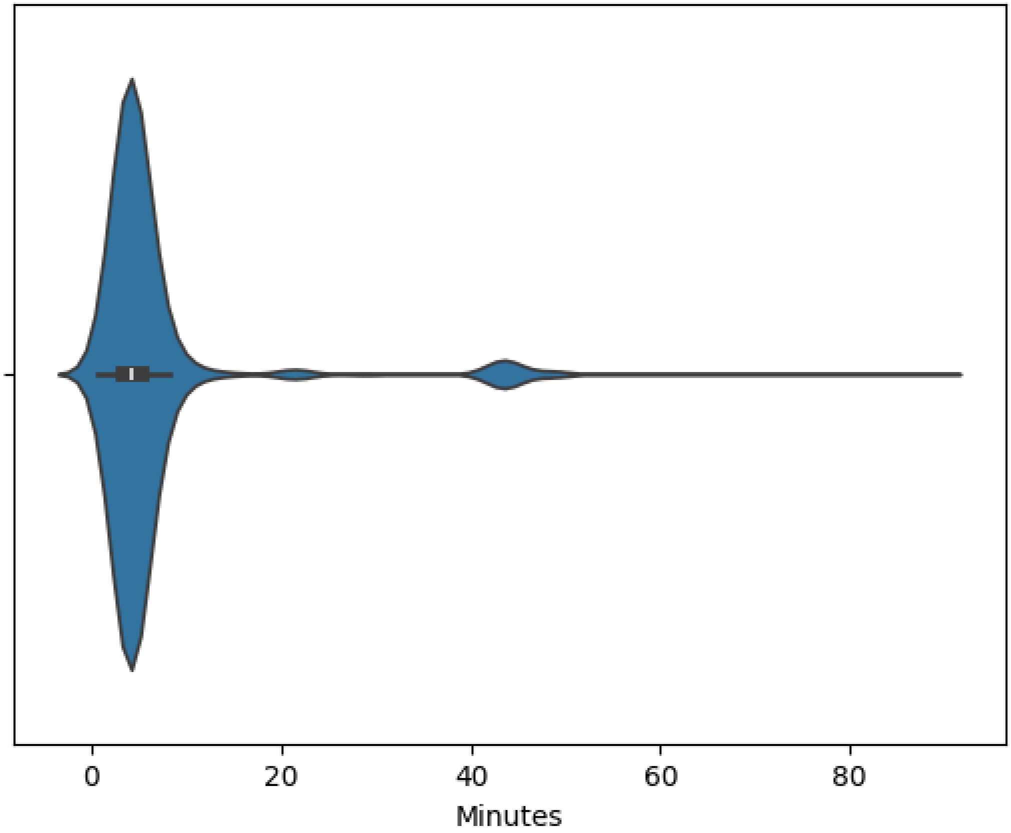


In [16]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.histplot(x=df['Minutes'], kde=True)
plt.tight_layout()  # auto-adjust margins
```

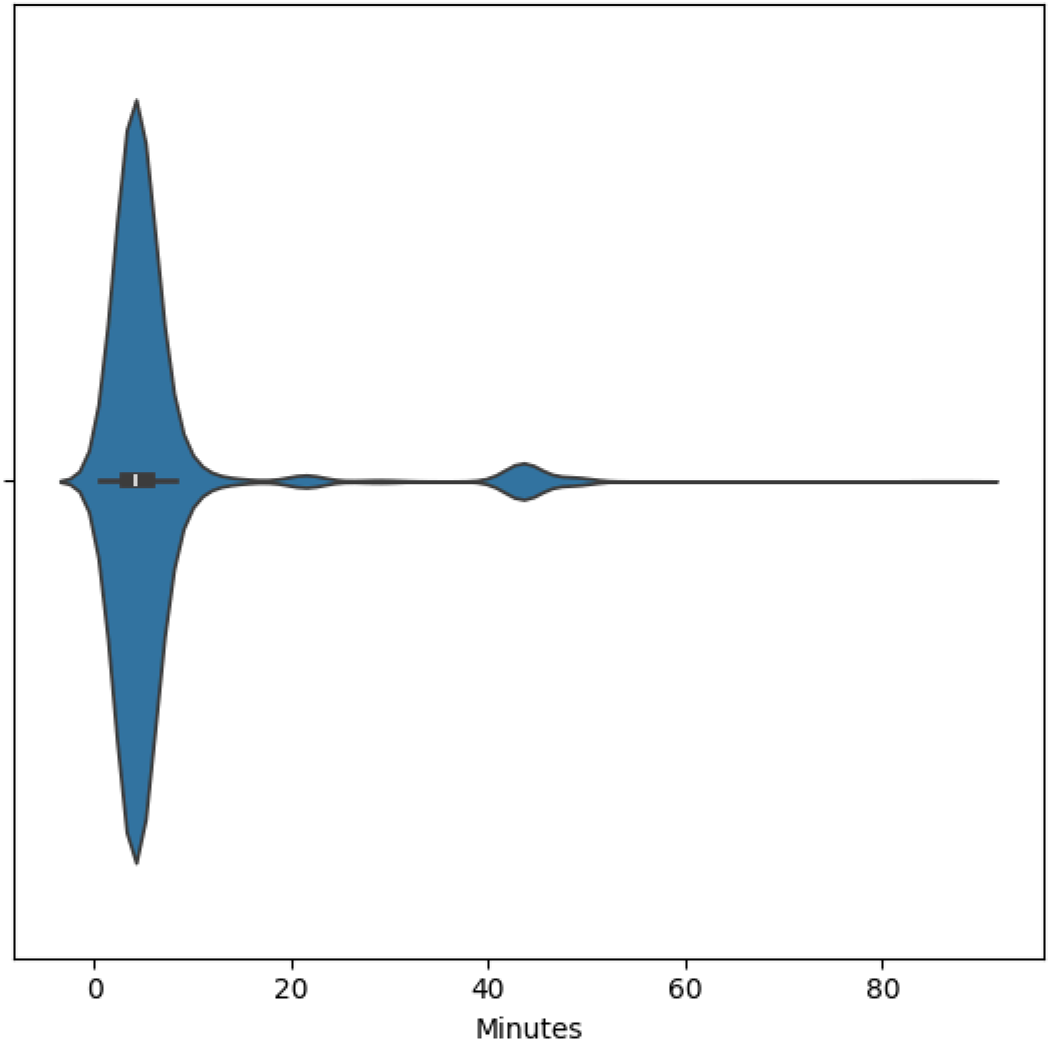


In [17]:
```
sns.violinplot(data=df, x='Minutes')
```

Out[17]:
```
<Axes: xlabel='Minutes'>
```



In [18]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
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]:

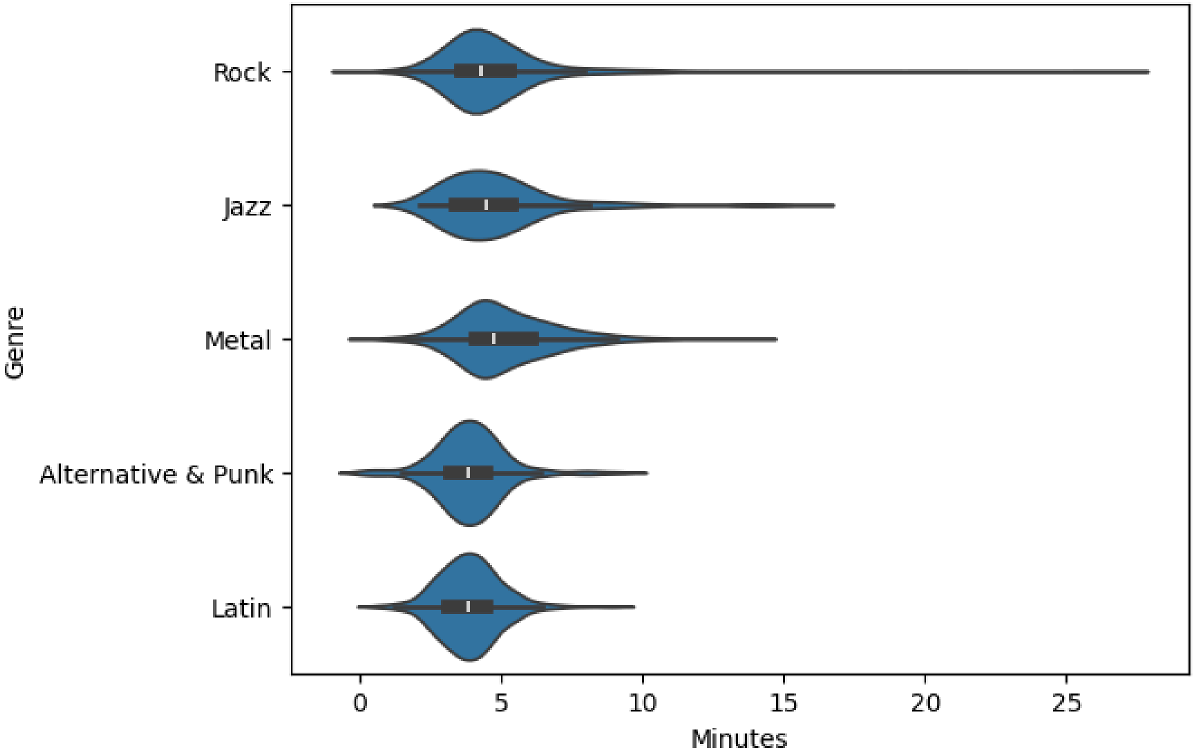
```
top_5_genres = df['Genre'].value_counts().index[:5]
top_5_data = data=df[df['Genre'].isin(top_5_genres)]
```

In [20]:

```
sns.violinplot(data=top_5_data, x='Minutes', y='Genre')
```

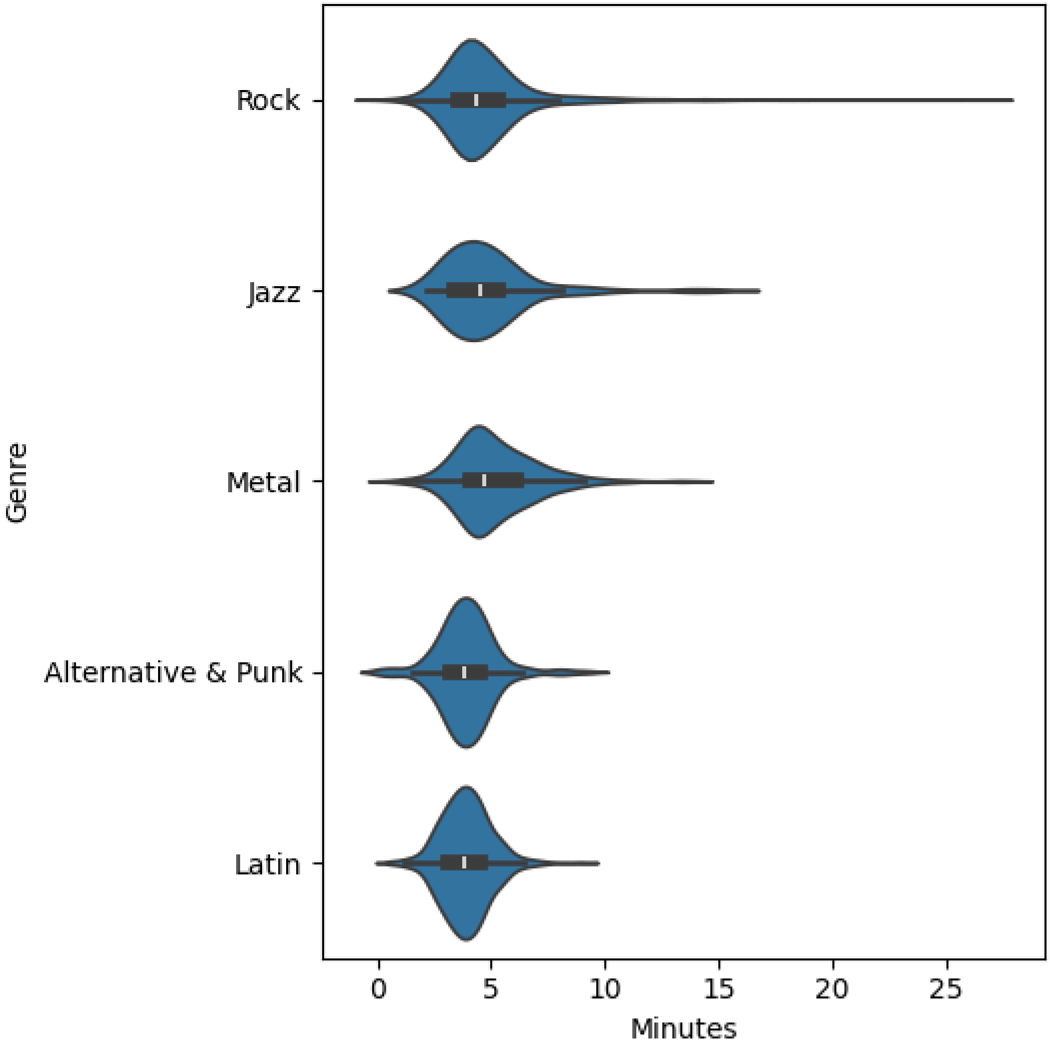
Out[20]:

```
<Axes: xlabel='Minutes', ylabel='Genre'>
```



In [21]:

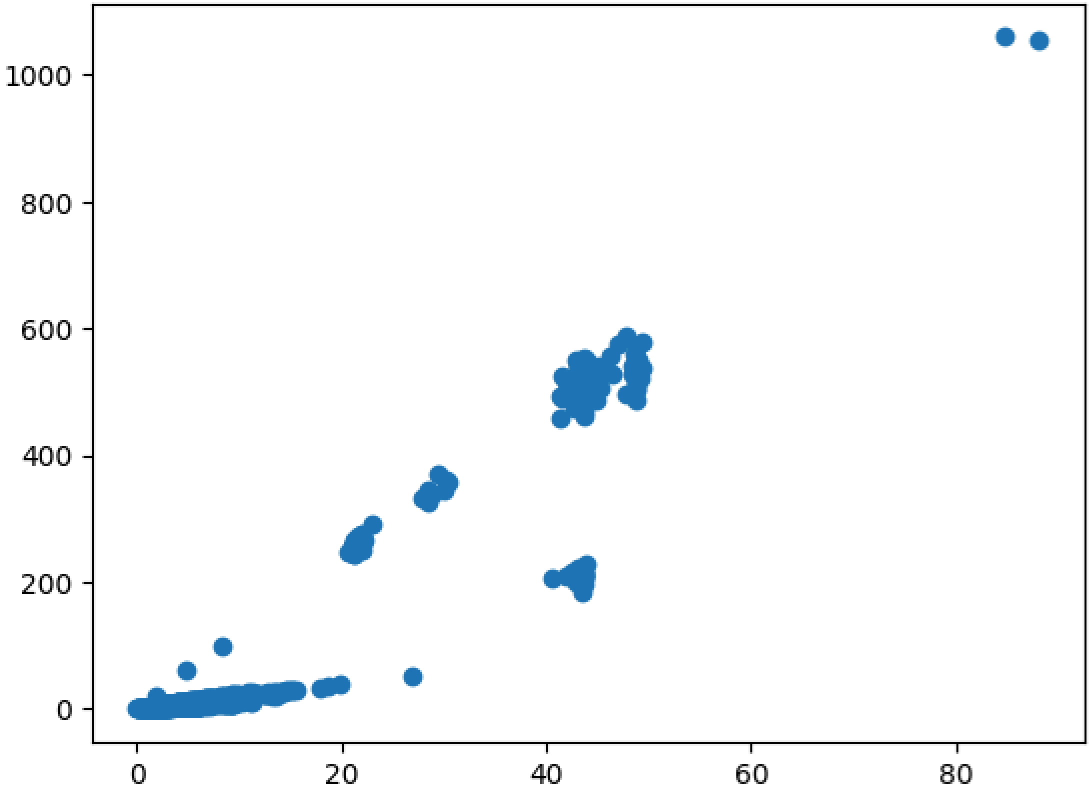
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.violinplot(data=top_5_data, x='Minutes', y='Genre')
plt.tight_layout()  # auto-adjust margins
```



Scatter plots

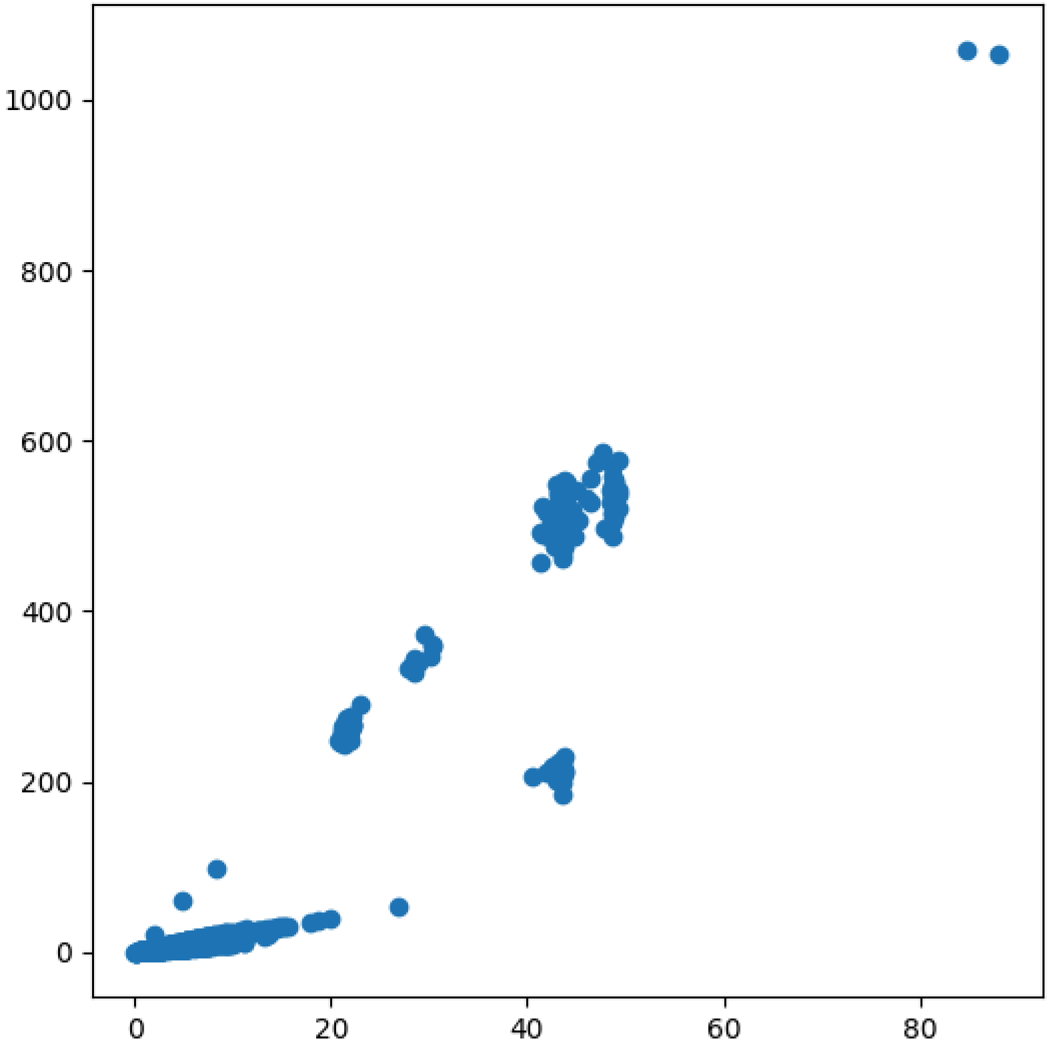
In [22]:
```
plt.scatter(df['Minutes'], df['MB'])
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x7efc5efa79d0>
```



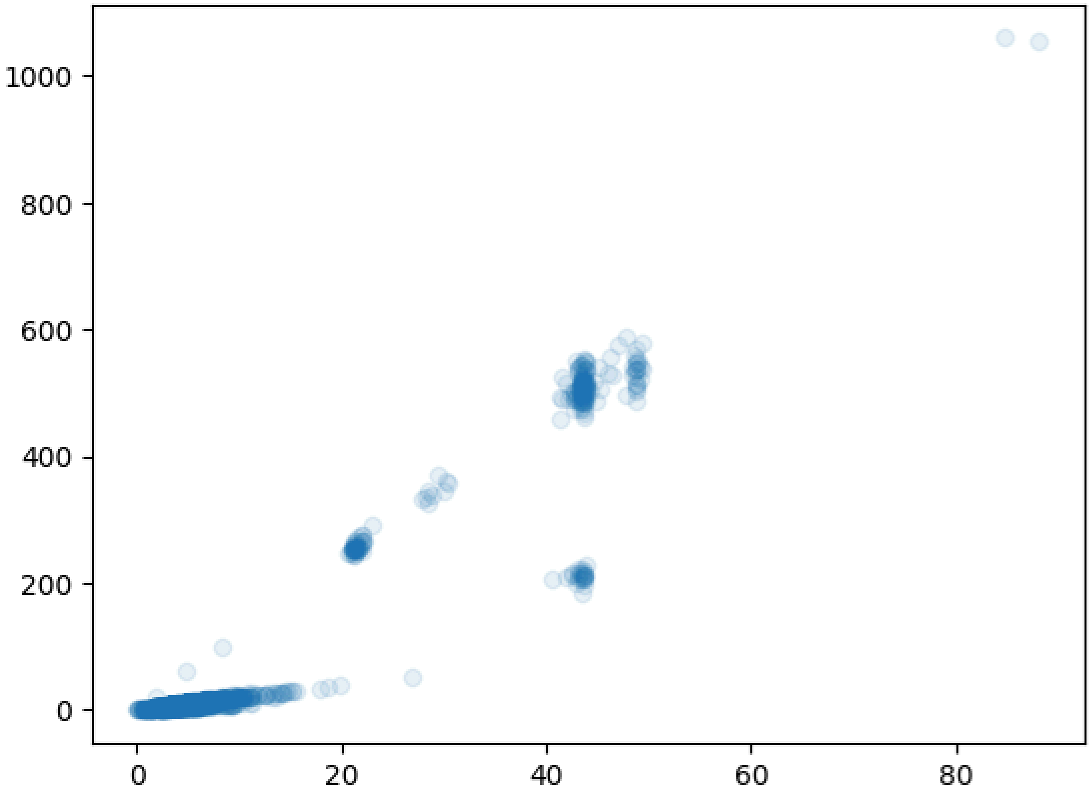
In [23]:

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



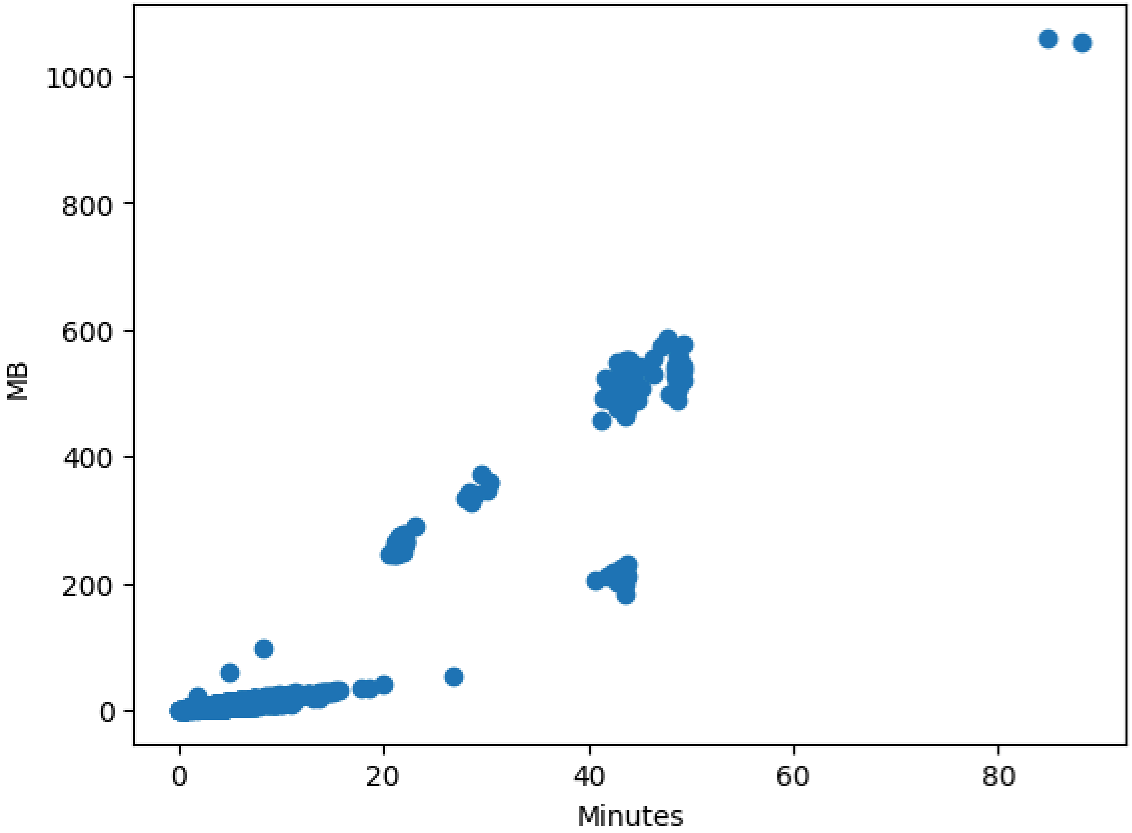
```
In [24]: plt.scatter(df['Minutes'], df['MB'], alpha=0.1)
```

```
Out[24]: <matplotlib.collections.PathCollection at 0x7efc5eea2990>
```



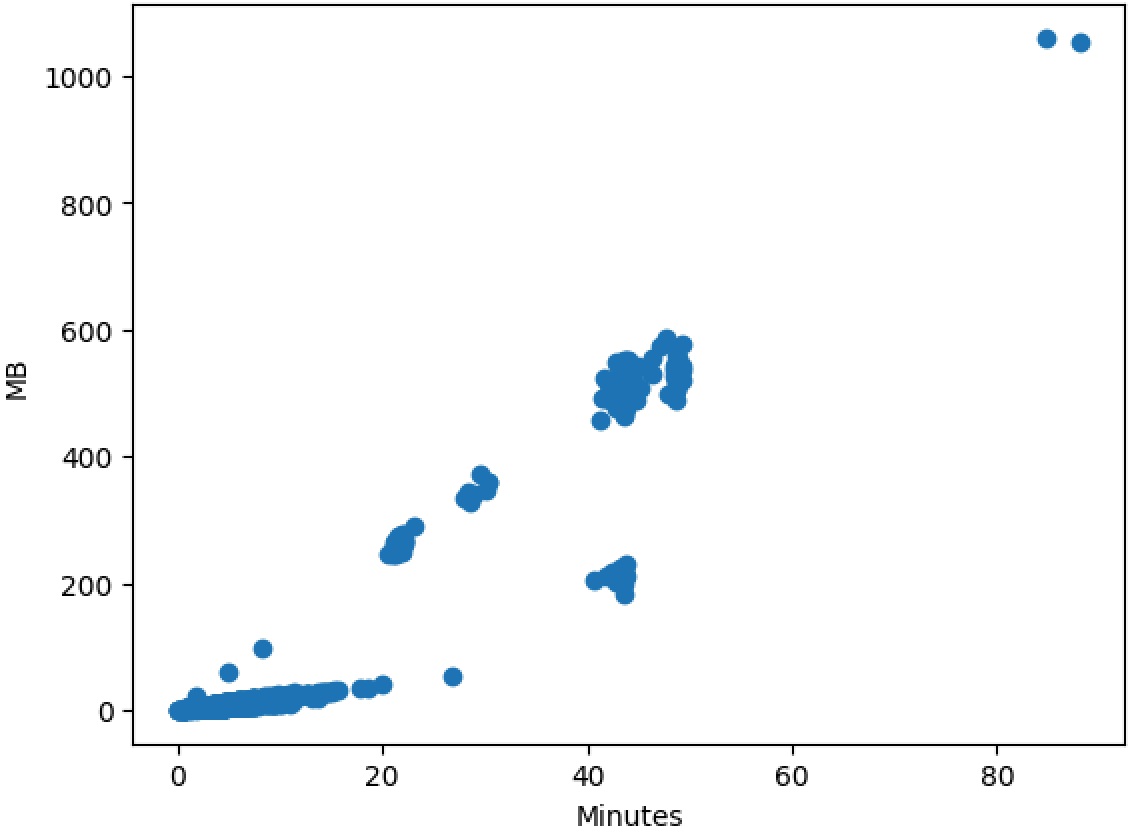
```
In [25]: plt.scatter(df['Minutes'], df['MB'])
         plt.xlabel('Minutes')
         plt.ylabel('MB')
```

```
Out[25]: Text(0, 0.5, 'MB')
```



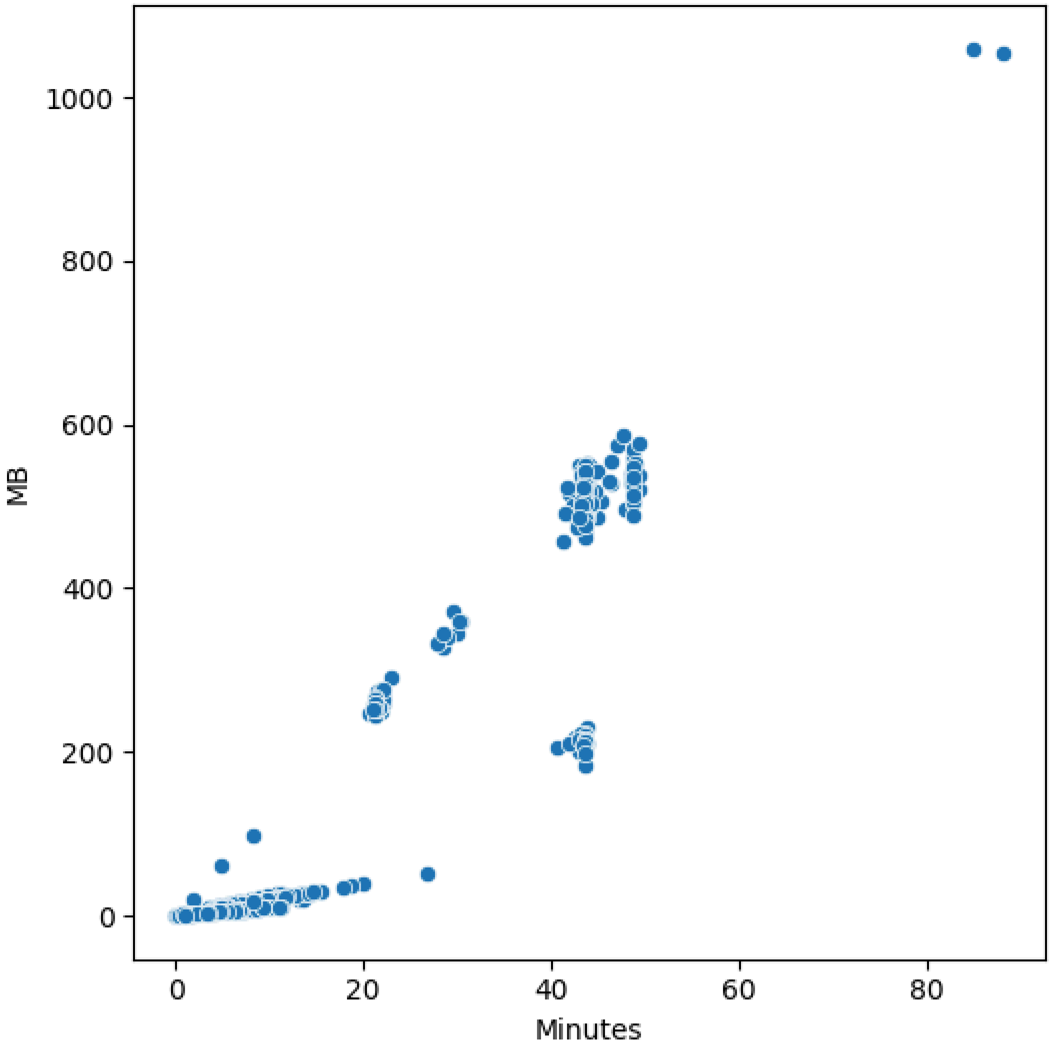
```
In [26]: plt.scatter(df['Minutes'], df['MB'])
         plt.xlabel('Minutes')
         plt.ylabel('MB')
```

```
Out[26]: Text(0, 0.5, 'MB')
```



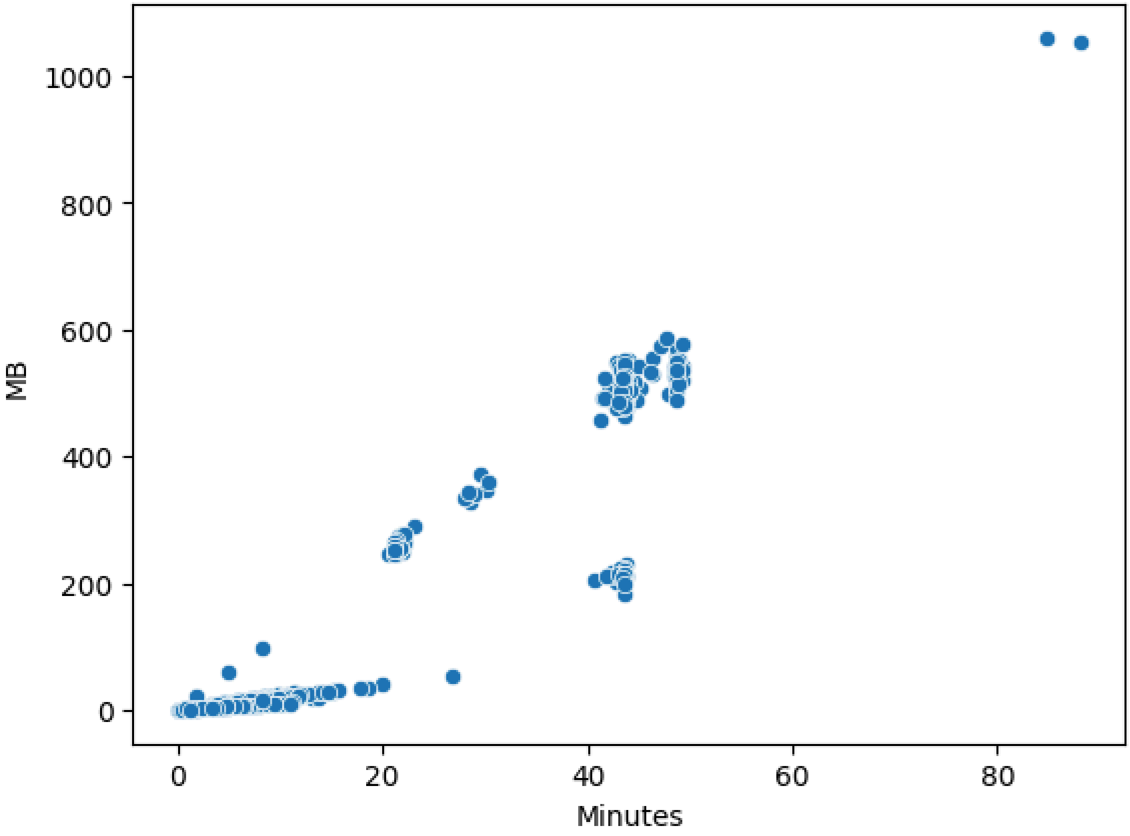
In [27]:

```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
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
```



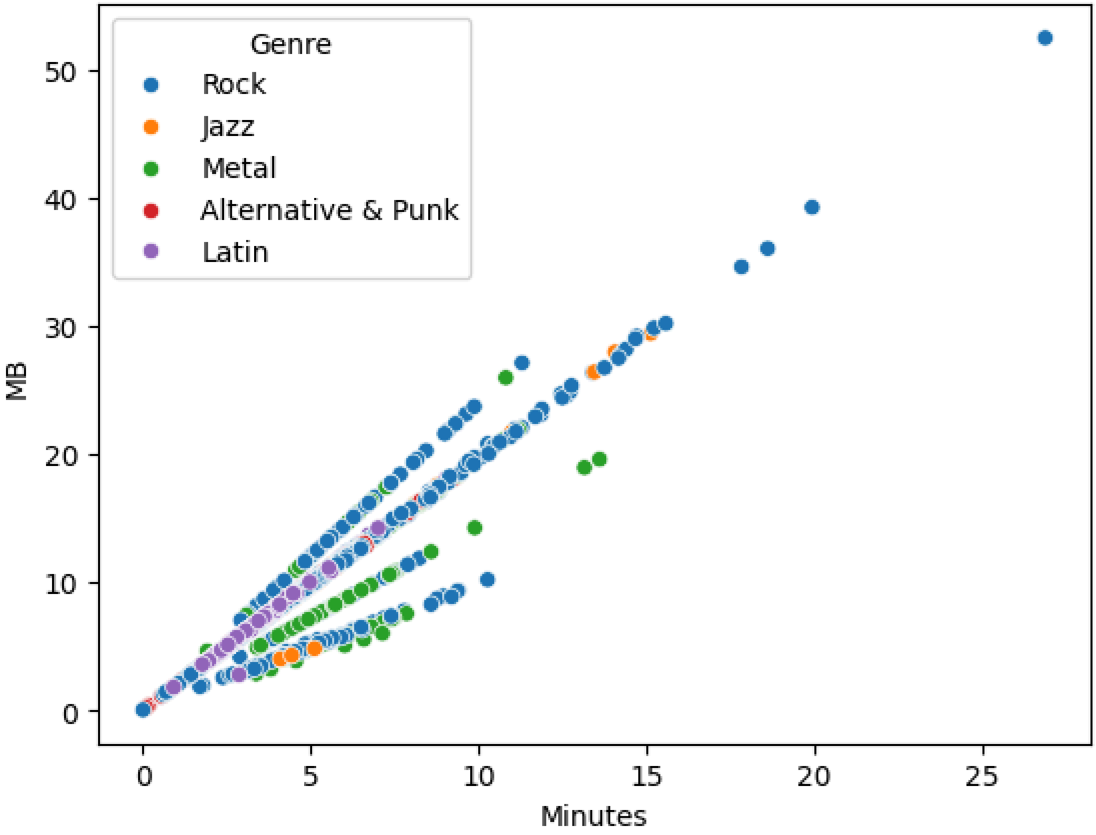
```
In [28]: sns.scatterplot(data=df, x='Minutes', y='MB')
```

```
Out[28]: <Axes: xlabel='Minutes', ylabel='MB'>
```

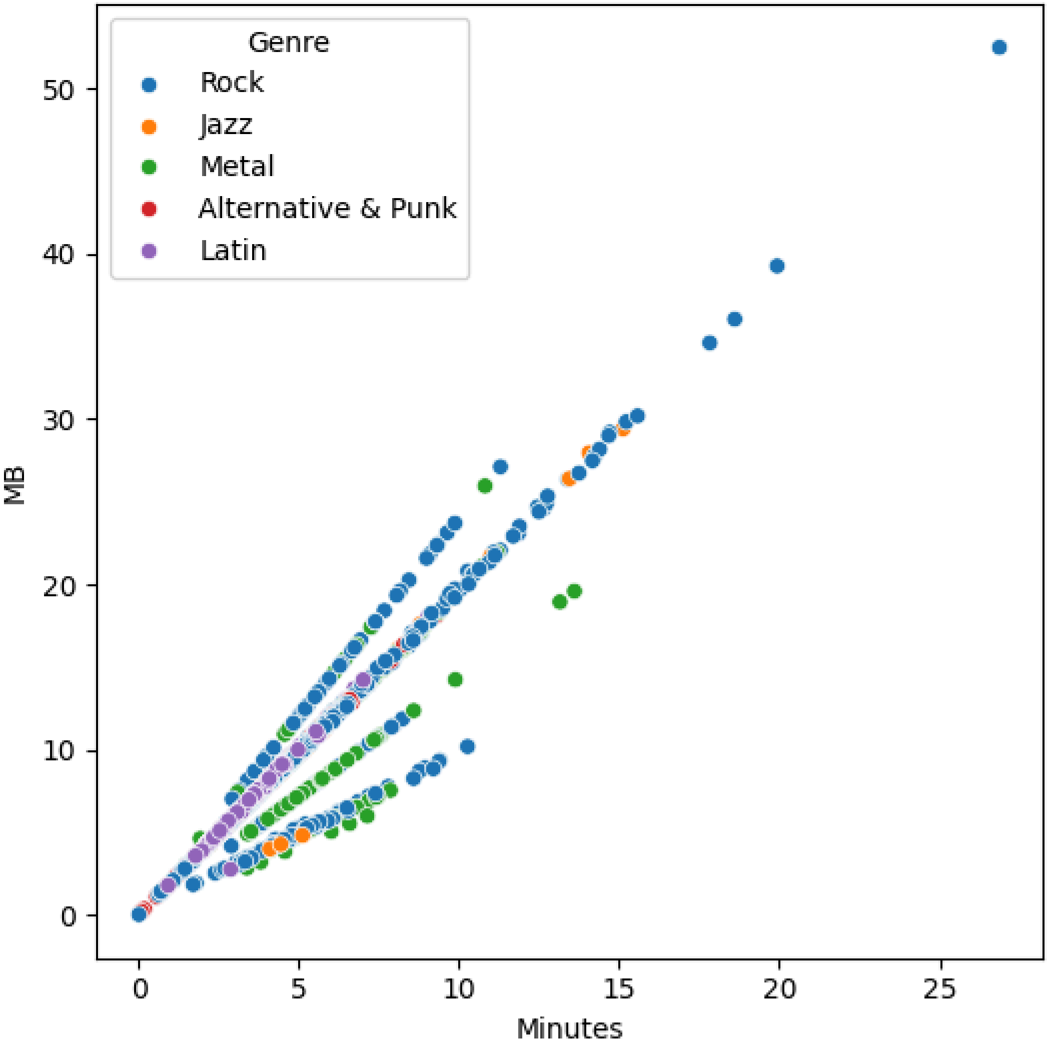


```
In [29]: sns.scatterplot(data=top_5_data, x='Minutes', y='MB', hue='Genre')
```

```
Out[29]: <Axes: xlabel='Minutes', ylabel='MB'>
```



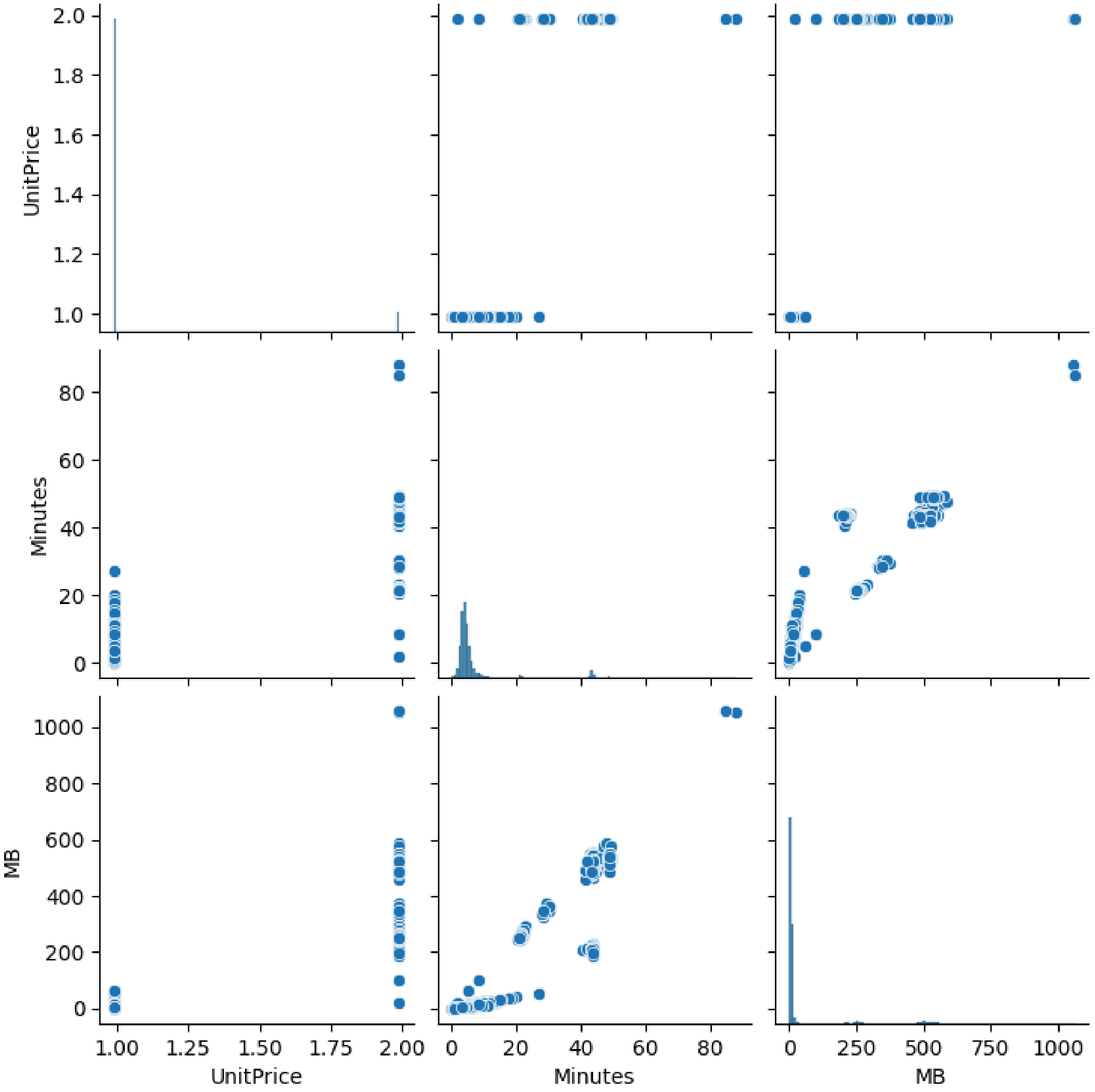
In [30]:
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.scatterplot(data=top_5_data, x='Minutes', y='MB', hue='Genre')
plt.tight_layout()  # auto-adjust margins
```



Pairplots and Correlograms

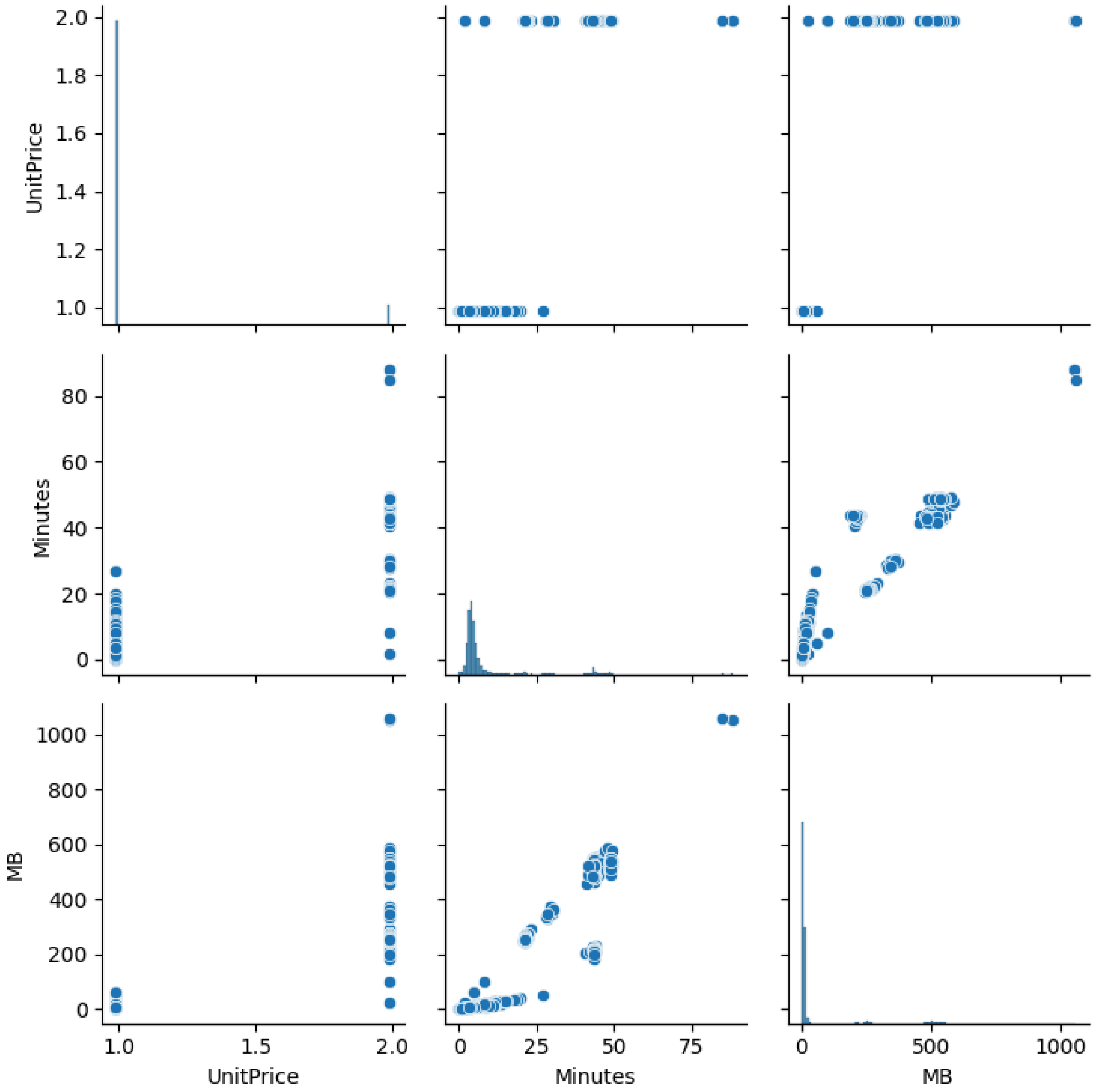
In [31]:
```
sns.pairplot(data=df)
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x7efc61de1fd0>
```



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

```
<Figure size 550x550 with 0 Axes>
```

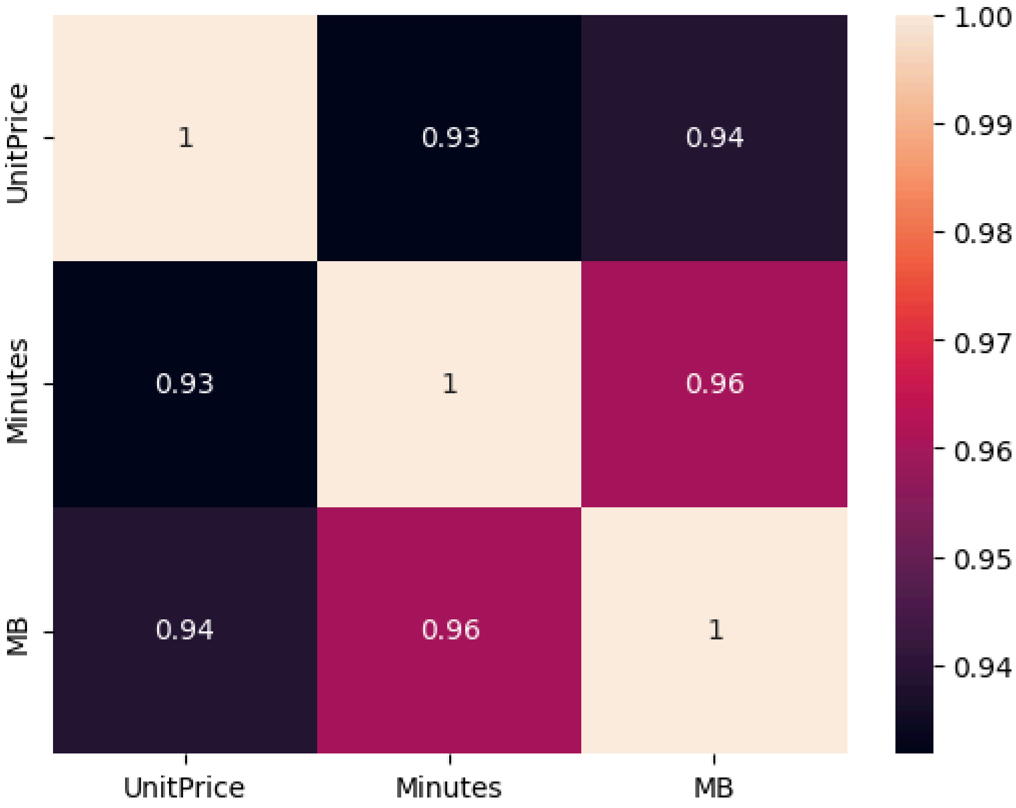


In [35]:

```
import numpy as np


sns.heatmap(df.select_dtypes(include=np.number).corr(), annot=True)
```

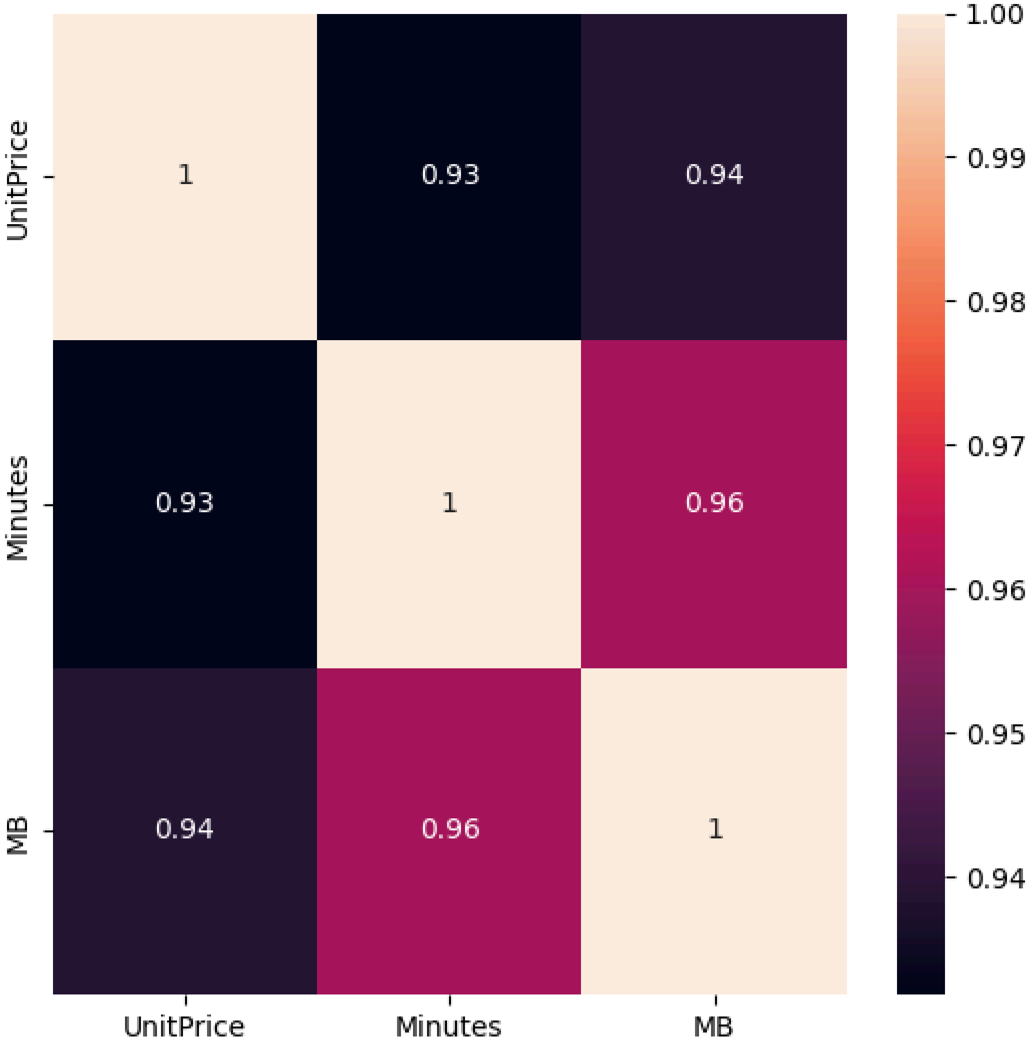
Out[35]:

```
<Axes: >
```



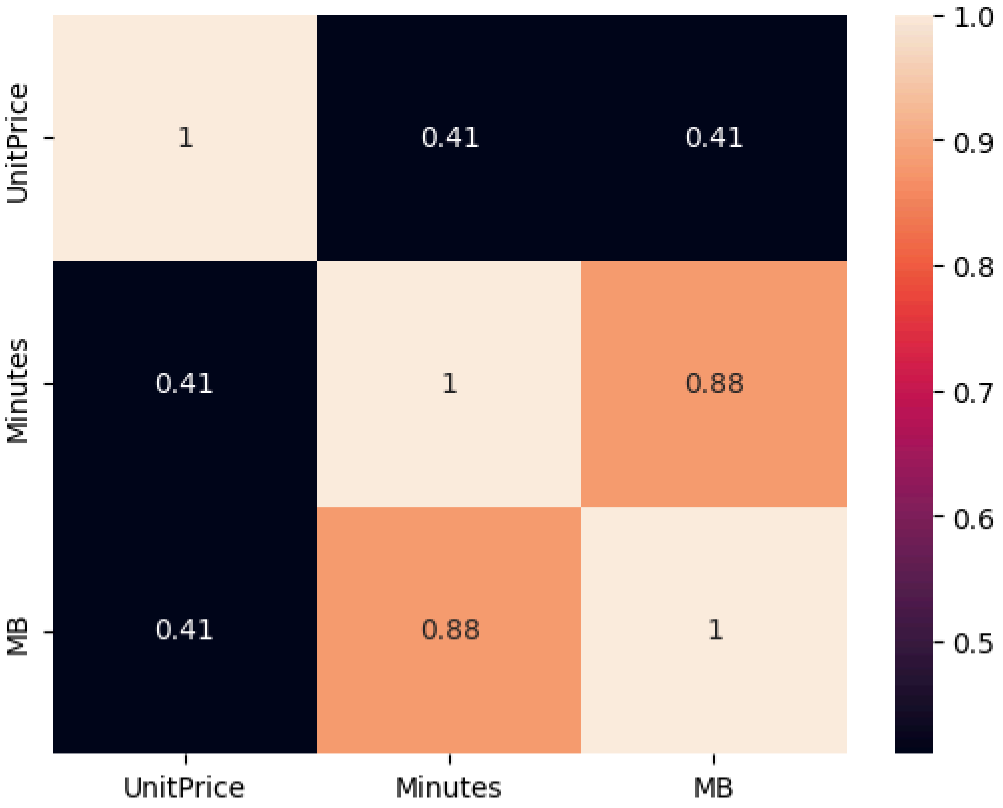
In [38]:

```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.heatmap(df.select_dtypes(include=np.number).corr(), annot=True)
plt.tight_layout()  # auto-adjust margins
```



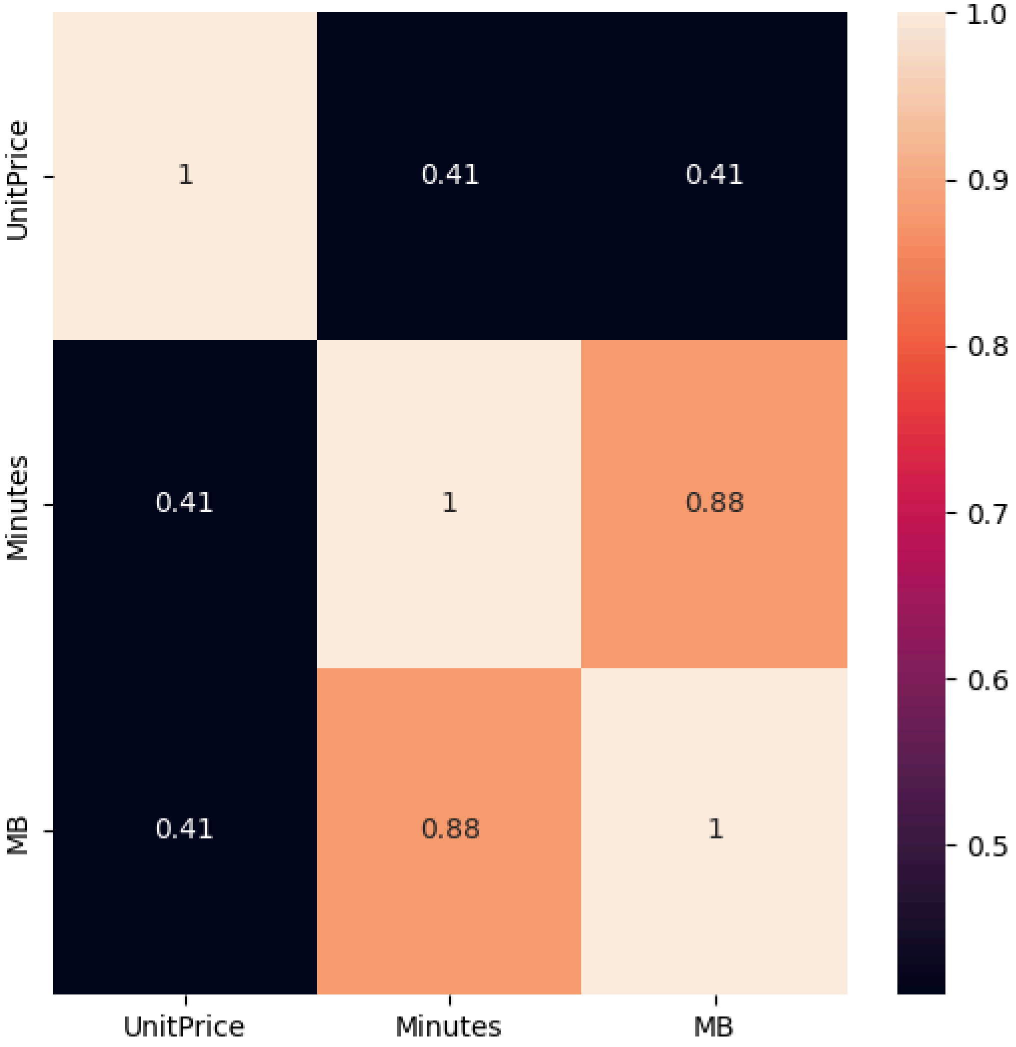
```
In [40]: sns.heatmap(df.select_dtypes(include=np.number).corr(method='spearman'), ann
```

```
Out[40]: <Axes: >
```



In [41]:

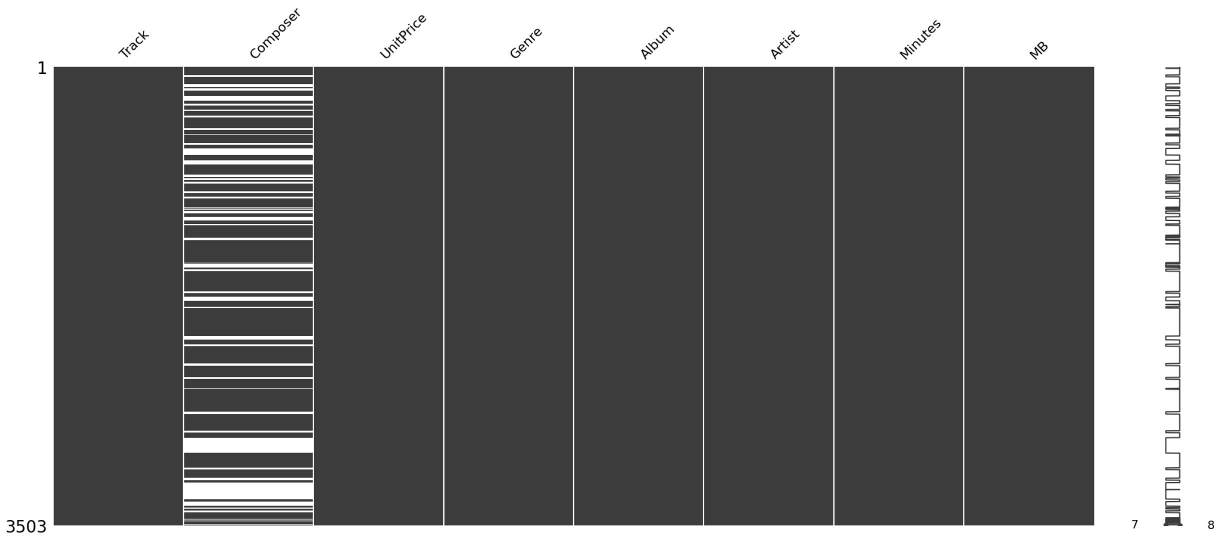
```
# save figure for book
f = plt.figure(figsize=(5.5, 5.5))  # this changes the size of the image --
f.patch.set_facecolor('w')  # sets background color behind axis labels
sns.heatmap(df.select_dtypes(include=np.number).corr(method='spearman'), ann
plt.tight_layout()  # auto-adjust margins
```



Missing value plot

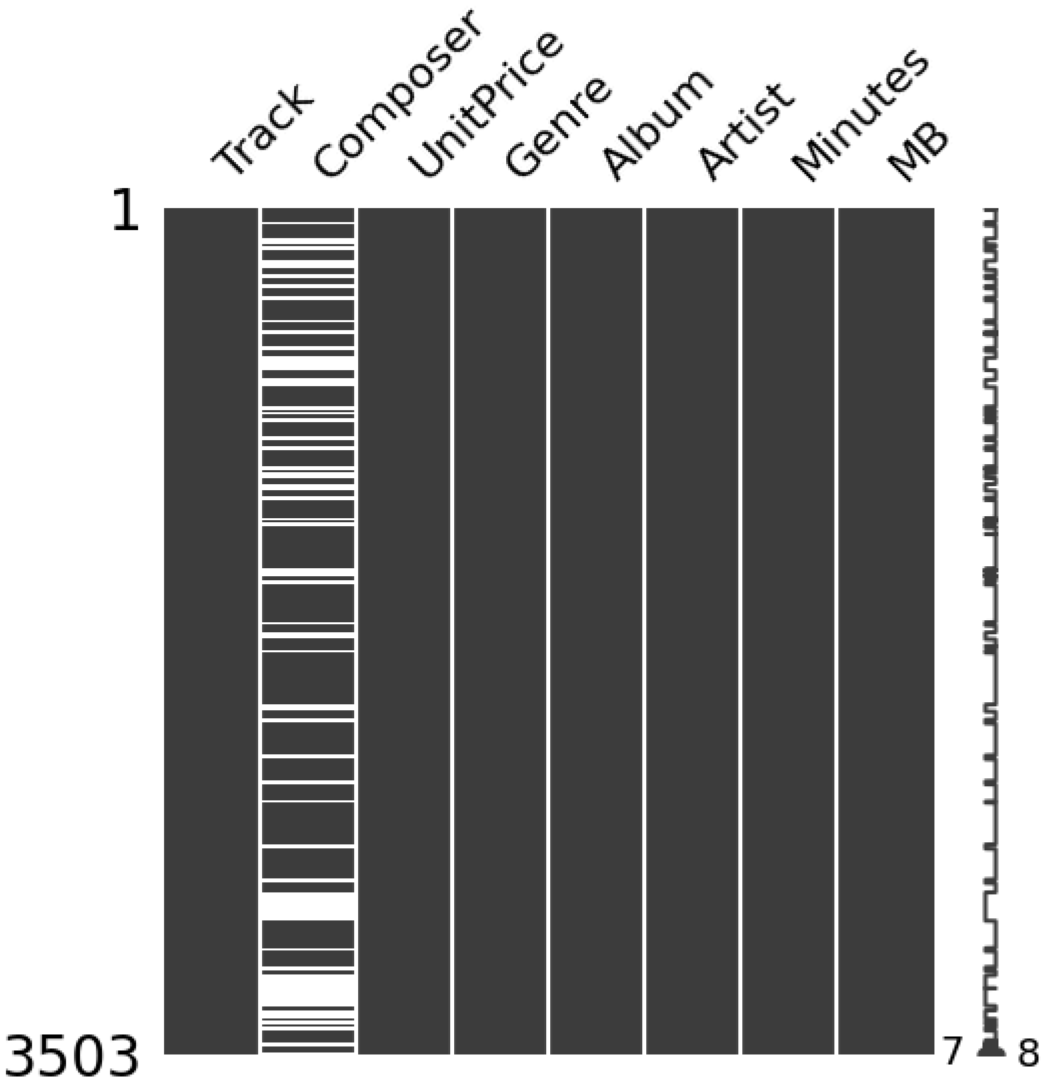
```
In [45]: import missingno as msno
         msno.matrix(df)
```

```
Out[45]: <Axes: >
```



In [46]:

```
# 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

Could Not Install (using `pip`), sorry

In [49]:
```
import pandas_profiling
pandas_profiling.__version__
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[49], line 1
----> 1 import pandas_profiling
      2 pandas_profiling.__version__

ModuleNotFoundError: No module named 'pandas_profiling'
```

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

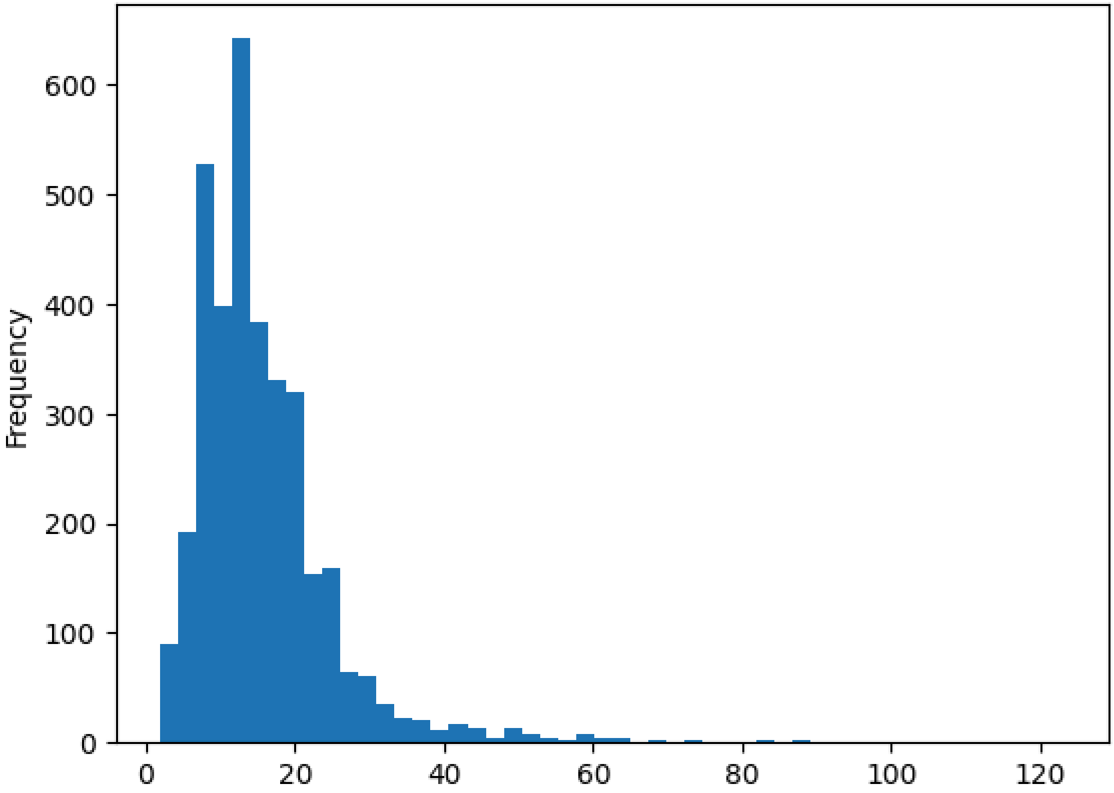
In [ ]:
```
report = ProfileReport(df)
```

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

```
Summarize dataset:   0%|          | 0/21 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render widgets:   0%|          | 0/1 [00:00<?, ?it/s]
VBox(children=(Tab(children=(Tab(children=(GridBox(children=(VBox(children=(G
ridspecLayout(children=(HTML(valu…
```

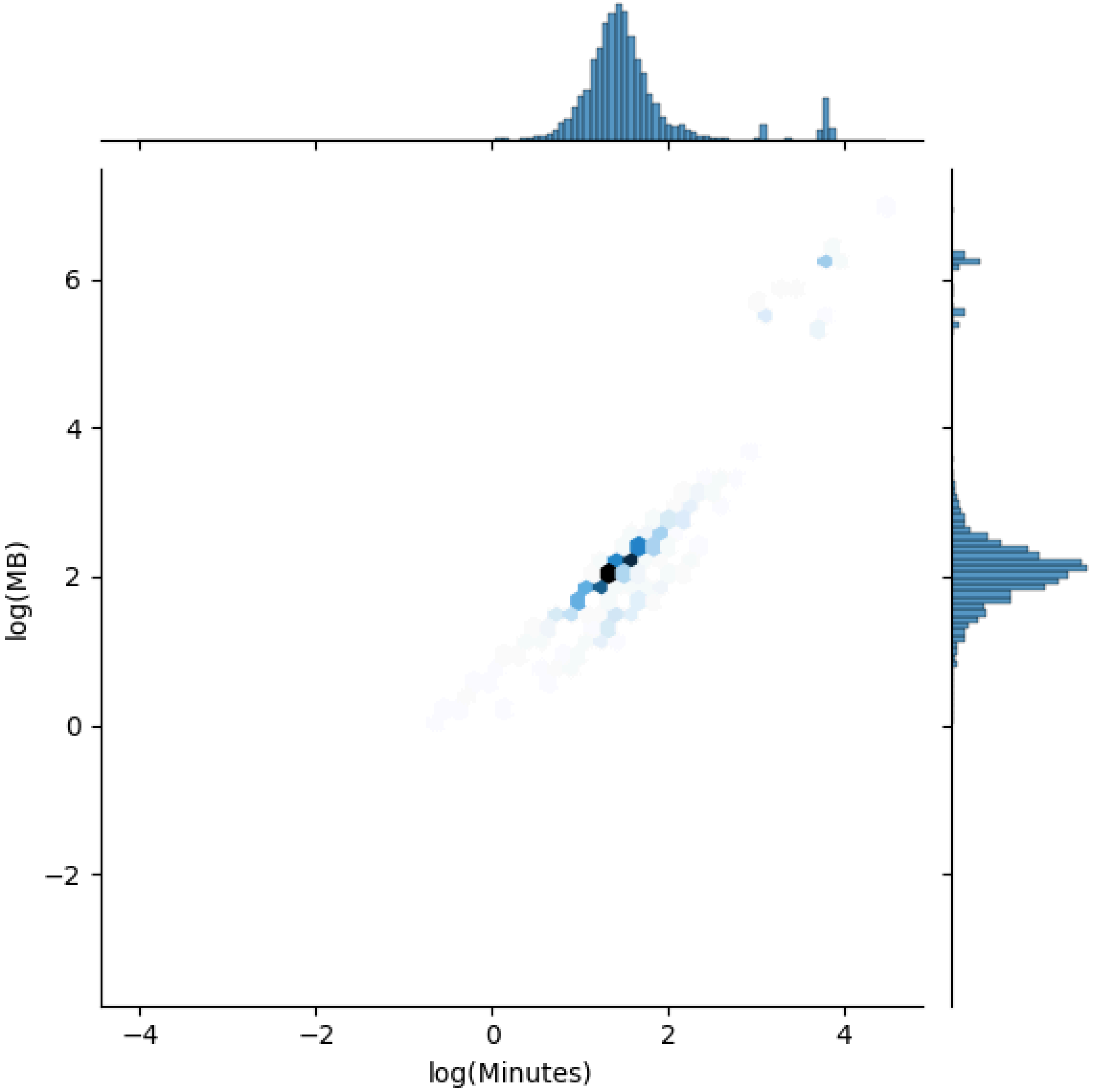
In [50]:
```
# create our own histogram of the length of the tracks
df['Track'].str.len().plot.hist(bins=50)
```

Out[50]:
```
<Axes: ylabel='Frequency'>
```

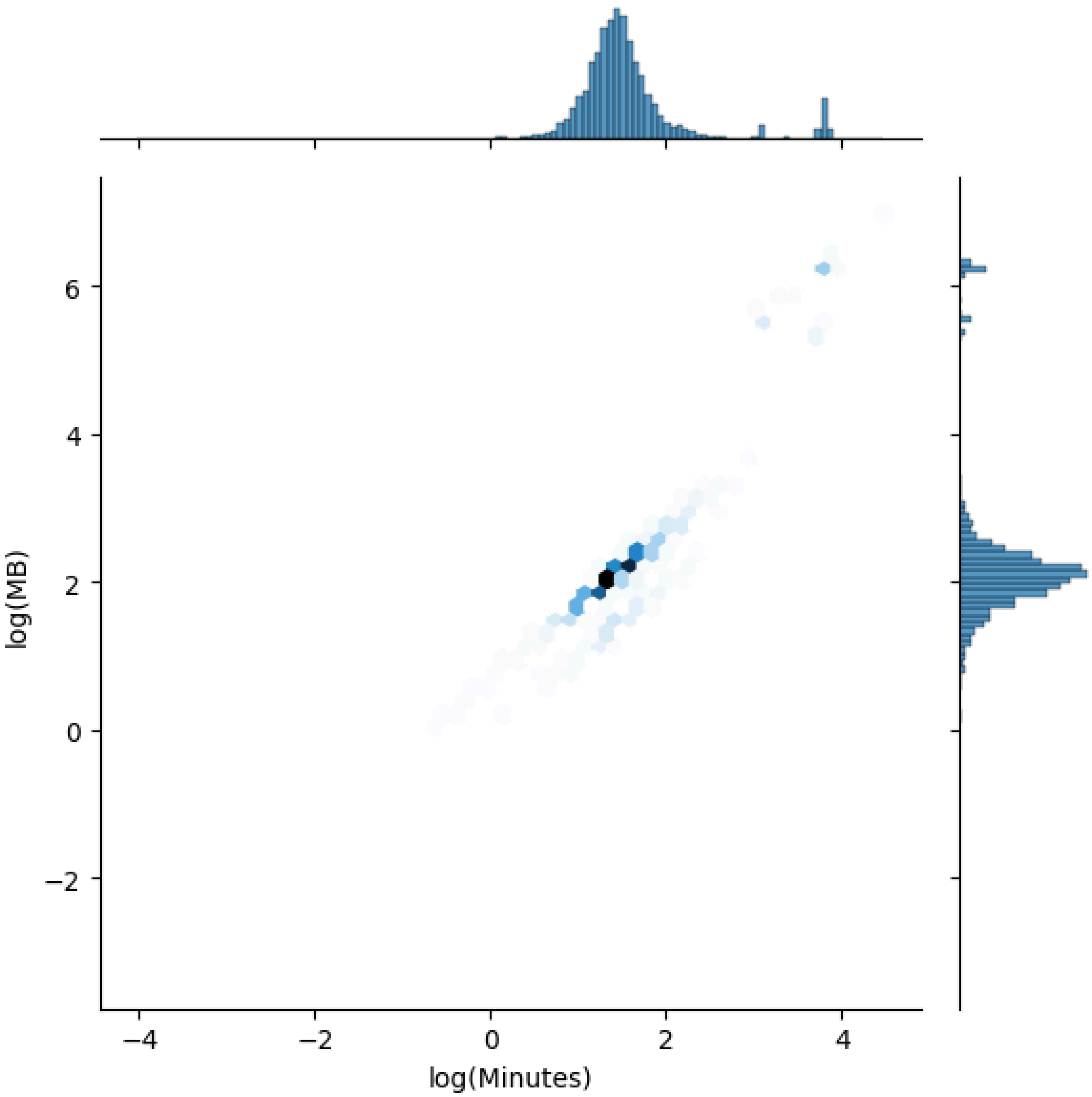


In [51]:
```
import numpy as np
df_log = df.copy()
df_log['log(Minutes)'] = np.log(df_log['Minutes'])
df_log['log(MB)'] = np.log(df_log['MB'])
sns.jointplot(x="log(Minutes)", y="log(MB)", data=df_log, kind="hex")
```

Out[51]:
```
<seaborn.axisgrid.JointGrid at 0x7efc61de3b60>
```



```
In [52]: # saving figure for book
         sns.jointplot(x="log(Minutes)", y="log(MB)", data=df_log, kind="hex")
         plt.tight_layout()  # auto-adjust margins
```



```
In [54]: report.to_notebook_iframe()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[54], line 1
----> 1 report.to_notebook_iframe()

NameError: name 'report' is not defined
```

Other EDA packages: autoviz, sweetviz, d-tale

```
In [55]: from autoviz.AutoViz_Class import AutoViz_Class
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
File ~/CS-620/.venv/lib64/python3.13/site-packages/xgboost/compat.py:105
    104 try:
--> 105     import pkg_resources
    106     pkg_resources.get_distribution('dask')

ModuleNotFoundError: No module named 'pkg_resources'

During handling of the above exception, another exception occurred:

NameError                                 Traceback (most recent call last)
Cell In[55], line 1
----> 1 from autoviz.AutoViz_Class import AutoViz_Class

File ~/CS-620/.venv/lib64/python3.13/site-packages/autoviz/__init__.py:3
      1 name = "autoviz"
      2 from .__version__ import __version__, __holo_version__
----> 3 from .AutoViz_Class import AutoViz_Class
      4 from .AutoViz_Class import data_cleaning_suggestions
      5 from .AutoViz_Class import FixDQ

File ~/CS-620/.venv/lib64/python3.13/site-packages/autoviz/AutoViz_Class.py:2
9
     27 from pandas_dq import Fix_DQ, dq_report
     28 ######################################################################
#####################
---> 29 from autoviz.AutoViz_Holo import AutoViz_Holo
     30 from autoviz.AutoViz_Utils import draw_pivot_tables, draw_scatters
     31 from autoviz.AutoViz_Utils import draw_pair_scatters, draw_barplots,
draw_heatmap

File ~/CS-620/.venv/lib64/python3.13/site-packages/autoviz/AutoViz_Holo.py:3
      1 from holoviews.ipython import display
----> 3 from autoviz.AutoViz_Utils import classify_print_vars, find_remove_du
plicates
      4 import numpy as np
      5 import pandas as pd

File ~/CS-620/.venv/lib64/python3.13/site-packages/autoviz/AutoViz_Utils.py:2
098
   2092     return train, test
   2095 ######################################################################
##############
   2096 ################      Find top features using XGB     ###############
####
   2097 ######################################################################
###########
-> 2098 from xgboost import XGBClassifier, XGBRegressor
   2101 def find_top_features_xgb(train, preds, numvars, target, modeltype, c
orr_limit=0.7, verbose=0):
   2102     """
   2103     This is a fast utility that uses XGB to find top features.
   2104     It returns a list of important features.
   2105     Since it is XGB, you don't have to restrict the input to just num
eric vars.
```

```
   2106     You can send in all kinds of vars and it will take care of transf
orming it. Sweet!
   2107     """

File ~/CS-620/.venv/lib64/python3.13/site-packages/xgboost/__init__.py:6
      1 """XGBoost: eXtreme Gradient Boosting library.
      2 
      3 Contributors: https://github.com/dmlc/xgboost/blob/master/CONTRIBUTOR
S.md
      4 """
----> 6 from .core import (
      7     DMatrix,
      8     DeviceQuantileDMatrix,
      9     Booster,
     10     DataIter,
     11     build_info,
     12     _py_version,
     13 )
     14 from .training import train, cv
     15 from . import rabit  # noqa

File ~/CS-620/.venv/lib64/python3.13/site-packages/xgboost/core.py:21
     18 import numpy as np
     19 import scipy.sparse
---> 21 from .compat import STRING_TYPES, DataFrame, py_str, PANDAS_INSTALLED
     22 from .libpath import find_lib_path
     23 from ._typing import (
     24     CStrPptr,
     25     c_bst_ulong,
   (...)     36     CupyT,
     37 )

File ~/CS-620/.venv/lib64/python3.13/site-packages/xgboost/compat.py:108
    106     pkg_resources.get_distribution('dask')
    107     DASK_INSTALLED = True
--> 108 except pkg_resources.DistributionNotFound:
    109     dask = None
    110     DASK_INSTALLED = False

NameError: name 'pkg_resources' is not defined
```

ERROR: Could not find a version that satisfies the requirement pkg_resources (from versions: none) ERROR: No matching distribution found for pkg_resources

[notice] A new release of pip is available: 23.2.1 -> 25.1.1 [notice] To update, run: pip install --upgrade pip

In [56]:

```
# the documentation is not great for autoviz, but you can use a dataframe li
AutoViz_Class().AutoViz(filename="", dfte=df)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[56], line 2
      1 # the documentation is not great for autoviz, but you can use a dataf
rame like so
----> 2 AutoViz_Class().AutoViz(filename="", dfte=df)

NameError: name 'AutoViz_Class' is not defined
```

Sweetviz

In [ ]:

```
```

In [57]:

```
import sweetviz as sv

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

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[57], line 1
----> 1 import sweetviz as sv
      3 sv = sv.analyze(df)
      4 sv.show_notebook()

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/__init__.py:15
     12 __license__ = "MIT"
     14 # These are the main API functions
---> 15 from sweetviz.sv_public import analyze, compare, compare_intra
     16 from sweetviz.feature_config import FeatureConfig
     18 # This is the main report class; holds the report data
     19 # and is used to output the final report

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/sv_public.py:4
      1 from typing import Union, List, Tuple
      2 import pandas as pd
----> 4 import sweetviz.dataframe_report
      5 from sweetviz.feature_config import FeatureConfig
      8 def analyze(source: Union[pd.DataFrame, Tuple[pd.DataFrame, str]],
      9             target_feat: str = None,
     10             feat_cfg: FeatureConfig = None,
     11             pairwise_analysis: str = 'auto'):

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/dataframe_report.
py:10
      8 from sweetviz.sv_types import NumWithPercent, FeatureToProcess, Featu
reType
      9 import sweetviz.from_dython as associations
---> 10 import sweetviz.series_analyzer as sa
     11 import sweetviz.utils as su
     12 from sweetviz.graph_associations import GraphAssoc

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/series_analyzer.p
y:4
      2 from sweetviz.sv_types import NumWithPercent, FeatureType, FeatureToP
rocess
      3 from sweetviz.type_detection import determine_feature_type
----> 4 import sweetviz.series_analyzer_numeric
      5 import sweetviz.series_analyzer_cat
      6 import sweetviz.series_analyzer_text

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/series_analyzer_n
umeric.py:3
      1 import numpy as np
      2 import pandas as pd
----> 3 from sweetviz.graph_numeric import GraphNumeric
      4 import sweetviz.sv_html as sv_html
      5 from sweetviz.sv_types import NumWithPercent, FeatureType, FeatureToP
rocess

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/graph_numeric.py:
8
      5 import warnings
      7 from sweetviz.config import config
```

```
----> 8 from sweetviz import sv_html_formatters
      9 from sweetviz.sv_types import FeatureType, FeatureToProcess
     10 import sweetviz.graph

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/sv_html_formatter
s.py:3
      1 from decimal import Decimal
      2 import numpy as np
----> 3 from sweetviz.graph_associations import CORRELATION_ERROR
      4 from sweetviz.graph_associations import CORRELATION_IDENTICAL
      7 def fmt_int_commas(value: float) -> str:

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/graph_association
s.py:6
      4 import matplotlib.pyplot as plt
      5 from sweetviz.sv_types import FeatureType
----> 6 import sweetviz.graph
      7 from sweetviz.config import config
      8 import itertools

File ~/CS-620/.venv/lib64/python3.13/site-packages/sweetviz/graph.py:8
      6 from io import BytesIO
      7 import base64
----> 8 from pkg_resources import resource_filename
      9 from pandas.plotting import register_matplotlib_converters
     11 from sweetviz import sv_html_formatters

ModuleNotFoundError: No module named 'pkg_resources'
```

In [58]:

```
import dtale
dtale.show(df)
```

▶ 8 3503	Track	
0	For Those About To Rock (We Salute You)	Angus Young, Malcolm Youn
1	Put The Finger On You	Angus Young, Malcolm Youn
2	Let's Get It Up	Angus Young, Malcolm Youn
3	Inject The Venom	Angus Young, Malcolm Youn
4	Snowballed	Angus Young, Malcolm Youn
5	Evil Walks	Angus Young, Malcolm Youn
6	C.O.D.	Angus Young, Malcolm Youn
7	Breaking The Rules	Angus Young, Malcolm Youn
8	Night Of The Long Knives	Angus Young, Malcolm Youn
9	Spellbound	Angus Young, Malcolm Youn
10	Balls to the Wall	nan
11	Fast As a Shark	F. Baltes, S. Kaufman, U. Dir
12	Restless and Wild	F. Baltes, R.A. Smith-Diesel,
13	Princess of the Dawn	Deaffy & R.A. Smith-Diesel
14	Go Down	AC/DC
15	Dog Eat Dog	AC/DC
16	Let There Be Rock	AC/DC

Out[58]:

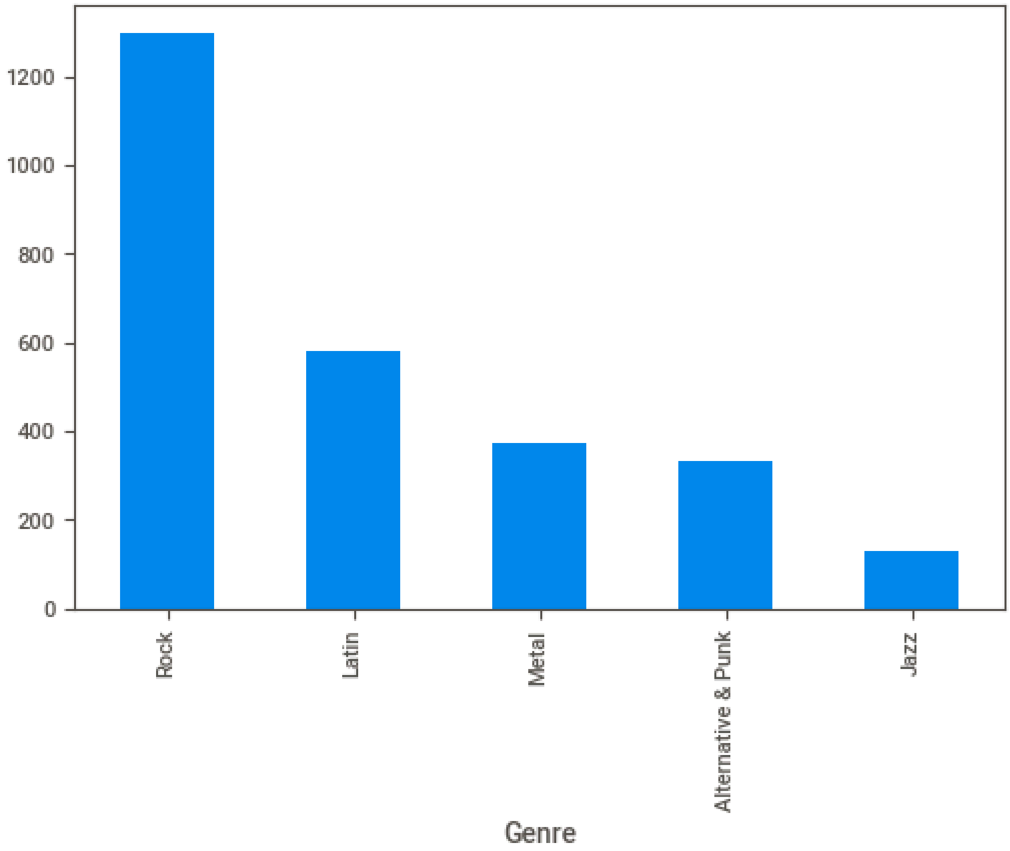
Visualization best practices

In [ ]:

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

Out[ ]:

```
Text(0.5, 0, 'Genre')
```

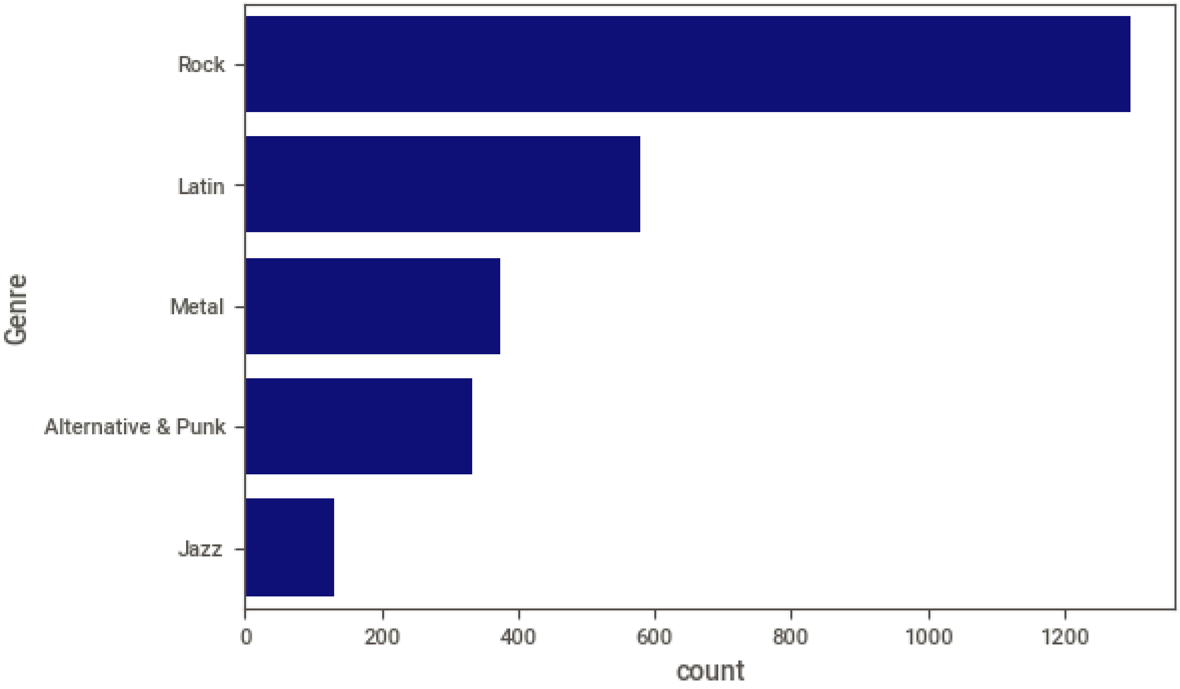


```
In [ ]: df['Genre'].value_counts()[:5]
```

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

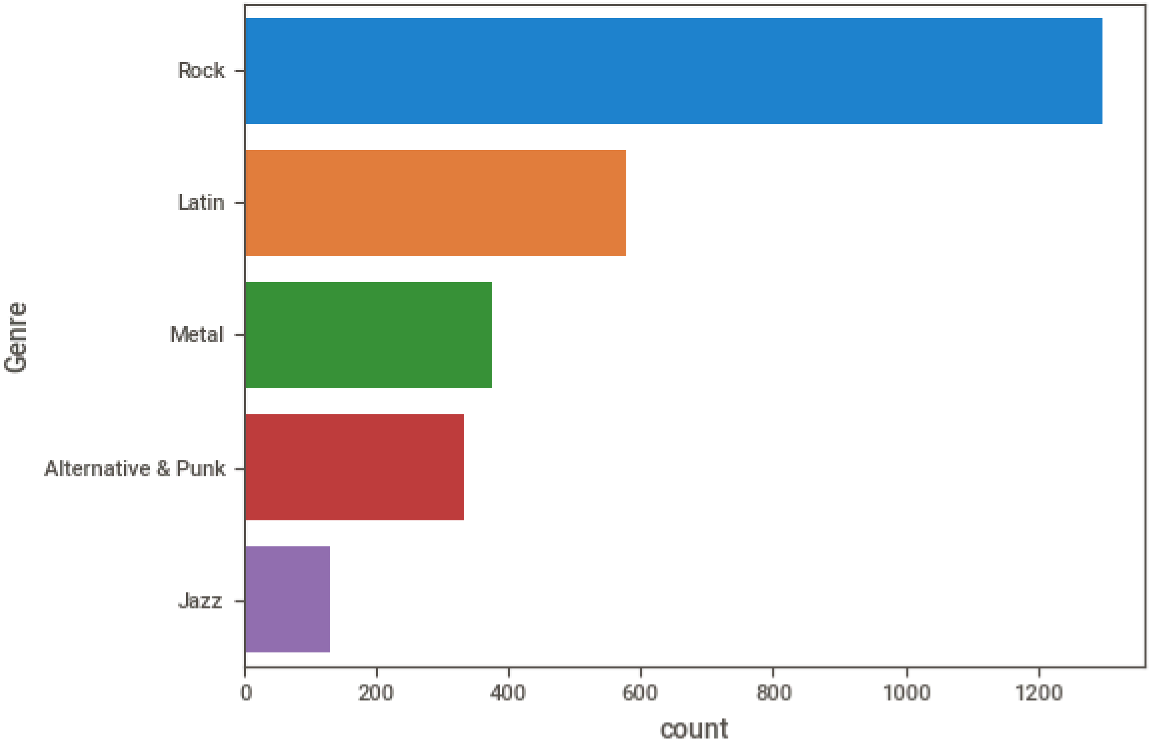
```
In [ ]: sns.countplot(y='Genre', data=df, order=df['Genre'].value_counts().index[:5]
```

```
Out[ ]: <AxesSubplot:xlabel='count', ylabel='Genre'>
```



In [ ]:

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



Choosing the right method for plotting

In [ ]:

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

Out[ ]:

	InvoiceId	CustomerId	InvoiceDate	BillingAddress	BillingCity	BillingState	BillingCountr
0	1	2	2009-01-01	Theodor-Heuss-Straße 34	Stuttgart	None	German
1	2	4	2009-01-02	Ullevålsveien 14	Oslo	None	Norwa
2	3	8	2009-01-03	Grétrystraat 63	Brussels	None	Belgiur
3	4	14	2009-01-06	8210 111 ST NW	Edmonton	AB	Canad
4	5	23	2009-01-11	69 Salem Street	Boston	MA	US

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

Out[ ]:

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

In [ ]:
```
top_3_countries = sql_df.groupby('BillingCountry').sum().sort_values(by='Tot
```

In [ ]:
```
top_3_countries
```

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

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

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

In [ ]:
```
gb
```

Out[ ]:

		InvoiceId	CustomerId	Total
InvoiceDate	**BillingCountry**			
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 [ ]:

```
gb.reset_index(inplace=True)
```

In [ ]:

```
gb.head()
```

Out[ ]:

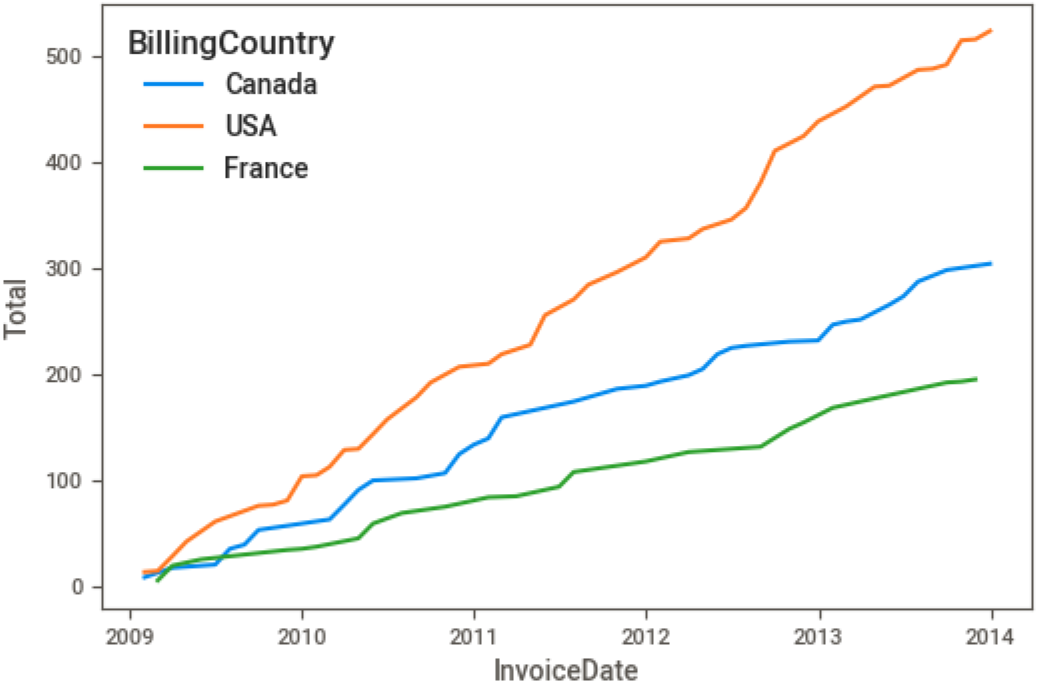
	InvoiceDate	BillingCountry	InvoiceId	CustomerId	Total
0	2009-01-31	Canada	4	14	8.91
1	2009-01-31	USA	5	23	13.86
2	2009-02-28	France	17	82	5.94
3	2009-02-28	USA	18	39	14.85
4	2009-03-31	Canada	22	45	17.82

In [ ]:

```
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry')
```

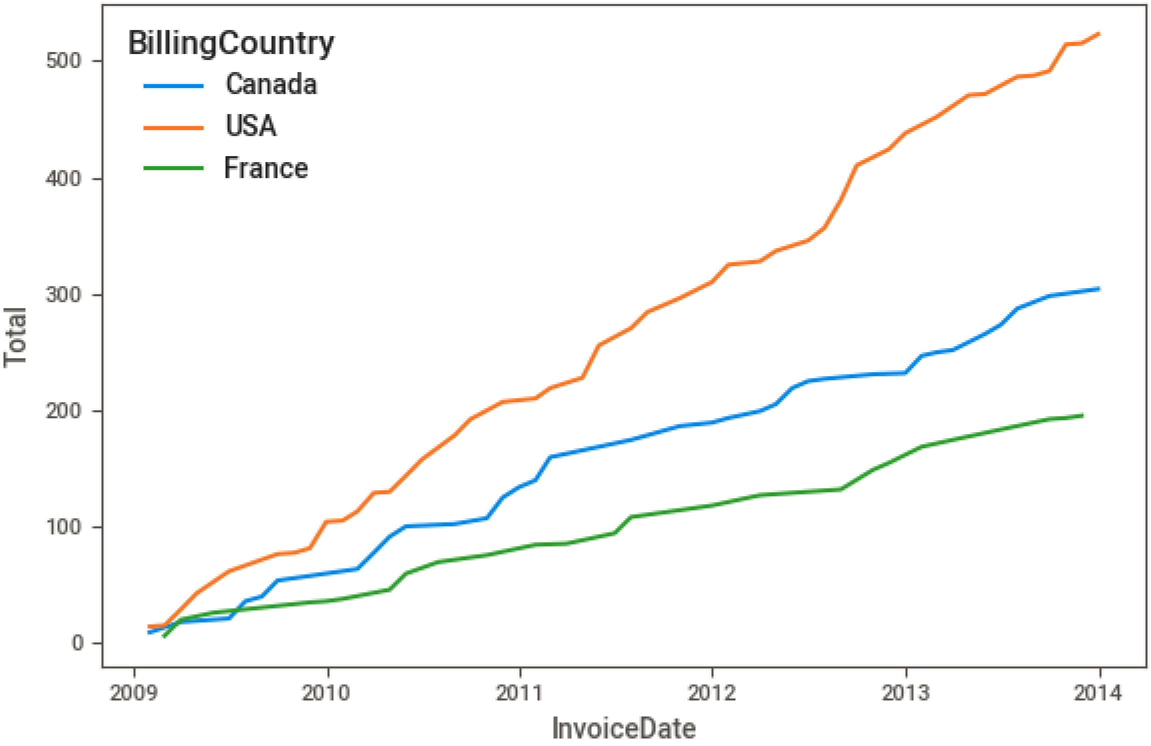
Out[ ]:

```
<AxesSubplot:xlabel='InvoiceDate', ylabel='Total'>
```



In [ ]:

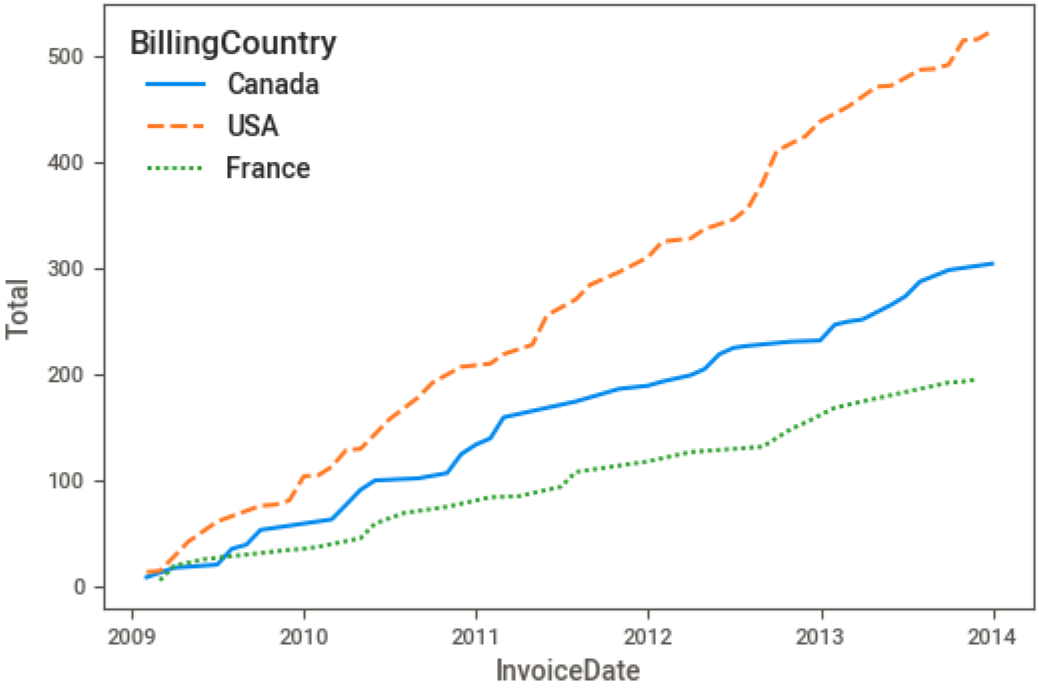
```
# saving figure for book
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry')
plt.tight_layout()
```



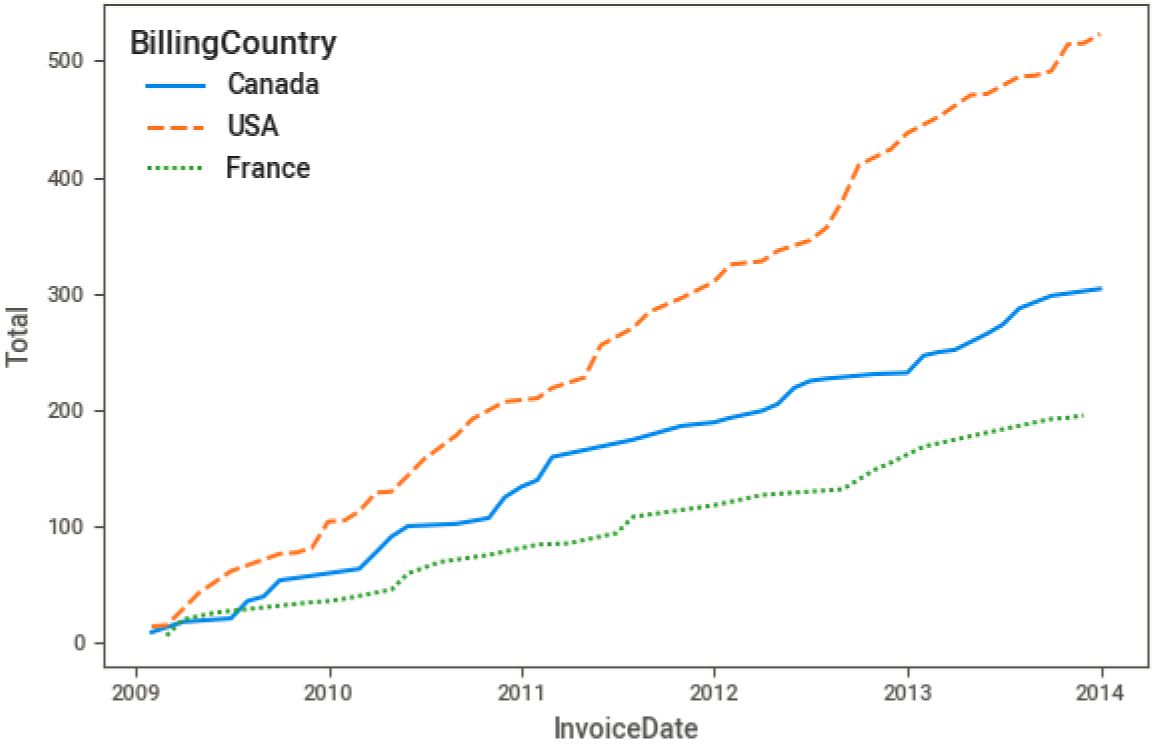
Making plots redundant

In [ ]:

```
# black and white redundancy
sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', styl
Out[ ]: <AxesSubplot:xlabel='InvoiceDate', ylabel='Total'>
```

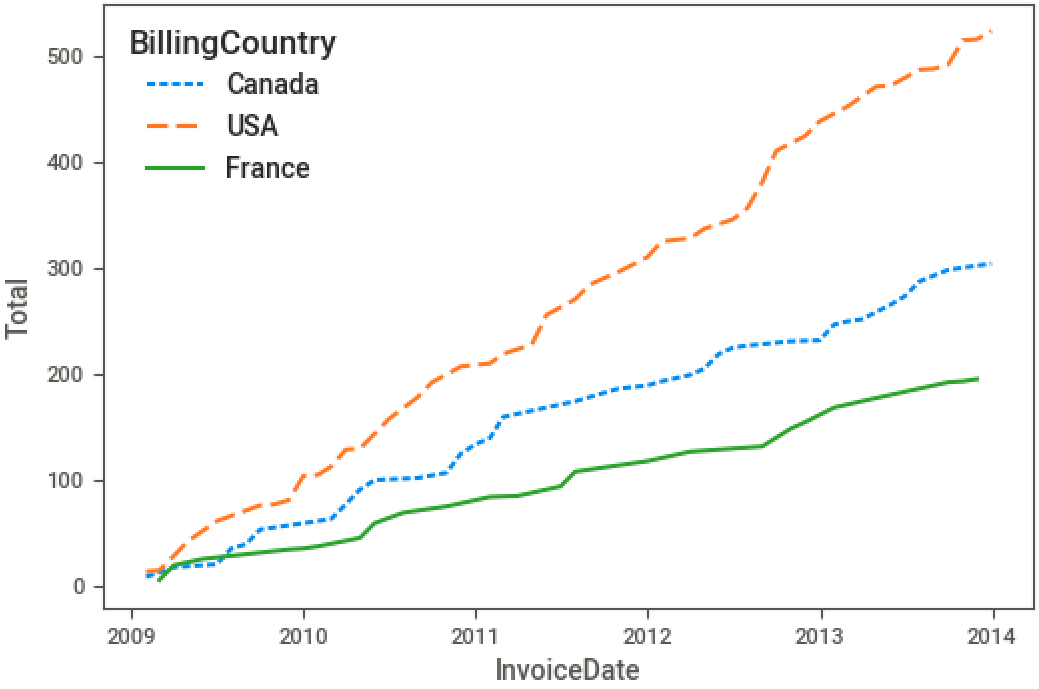


```
In [ ]: # save figure for book
        sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', styl
        plt.tight_layout()
```



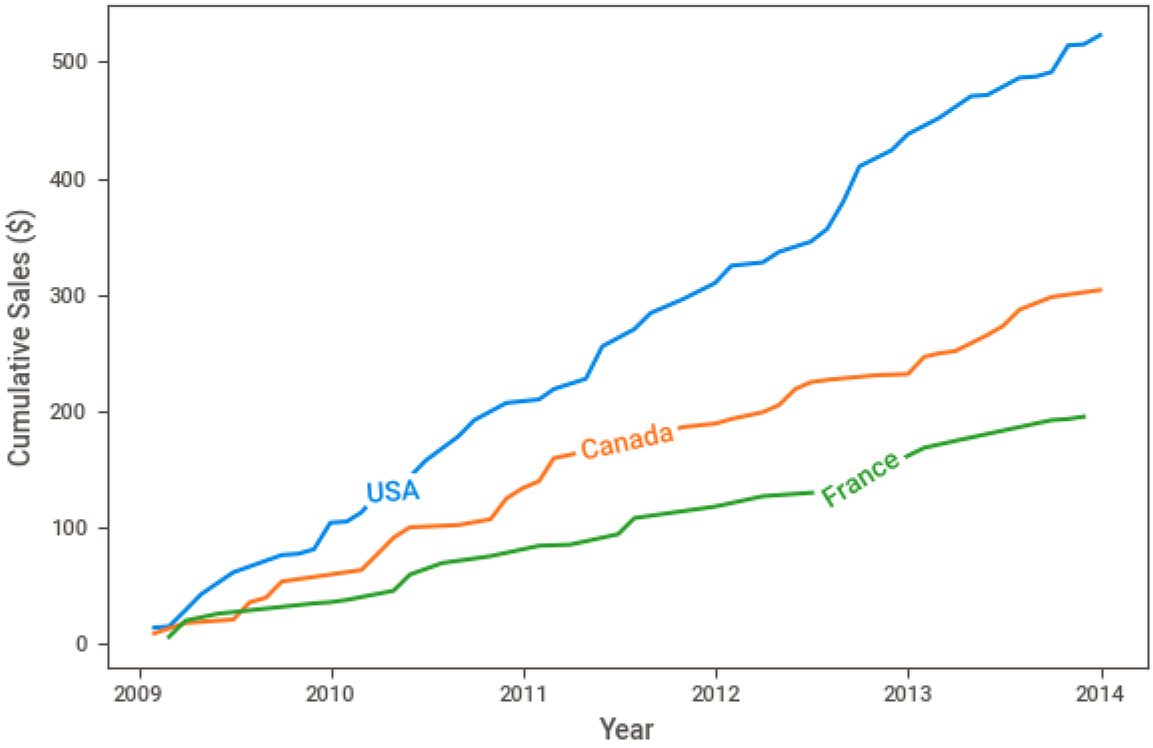
```
In [ ]: sns.lineplot(data=gb, x='InvoiceDate', y='Total', hue='BillingCountry', styl
```

```
Out[ ]: <AxesSubplot:xlabel='InvoiceDate', ylabel='Total'>
```



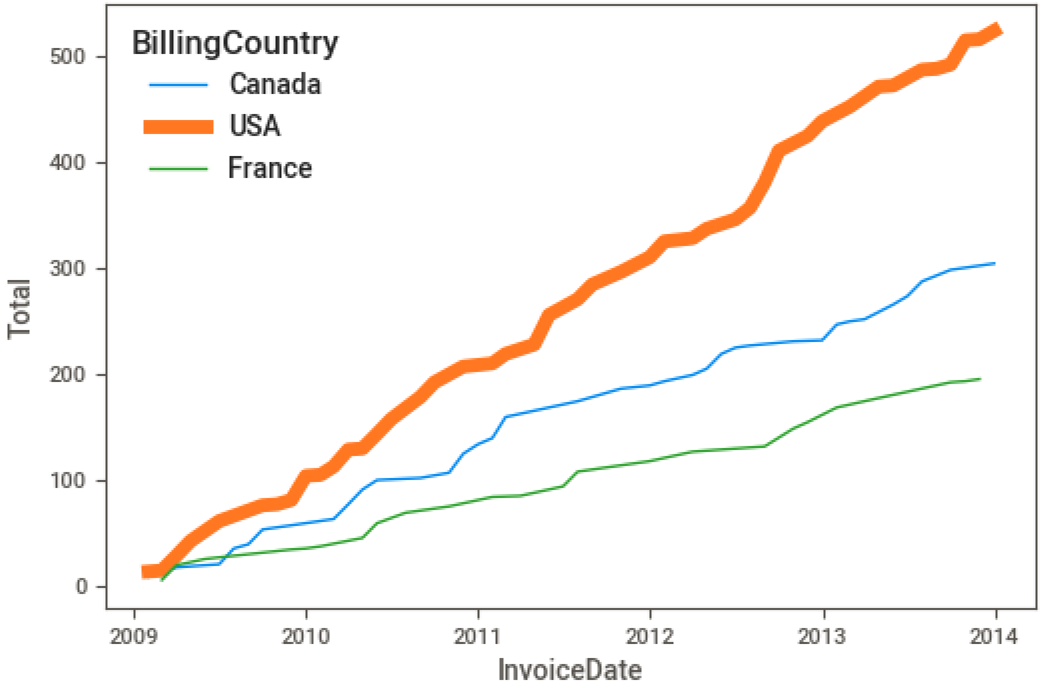
In [ ]:
```
from labellines import labelLines
```

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

```
Out[ ]: <AxesSubplot:xlabel='InvoiceDate', ylabel='Total'>
```



Another way to collect and plot the same data

```
In [ ]: top_3_df_gb = sql_df[sql_df['BillingCountry'].isin(top_3_countries)].groupby
        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 [ ]:

```
full_c_df.head()
```

Out[ ]:

	InvoiceId	CustomerId	Total	Country
InvoiceDate				
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 [ ]:

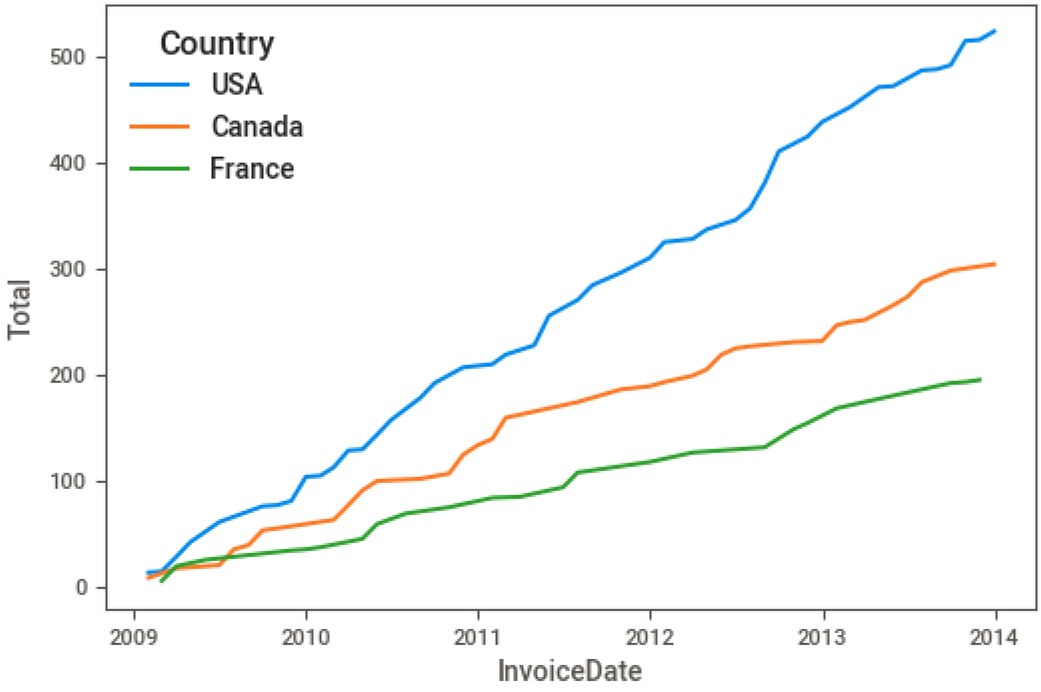
```
full_c_df.reset_index(inplace=True)
```

In [ ]:

```
sns.lineplot(data=full_c_df, x='InvoiceDate', y='Total', hue='Country')
```

Out[ ]:

```
<AxesSubplot:xlabel='InvoiceDate', ylabel='Total'>
```



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 [ ]:

```
engine = create_engine('sqlite:///data/chinook.db')

query = """SELECT SUM(Total) OVER (ORDER BY InvoiceId) as Total, strftime("%
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 [ ]:

```
sql_df2.head()
```

Out[ ]:

	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 [ ]:

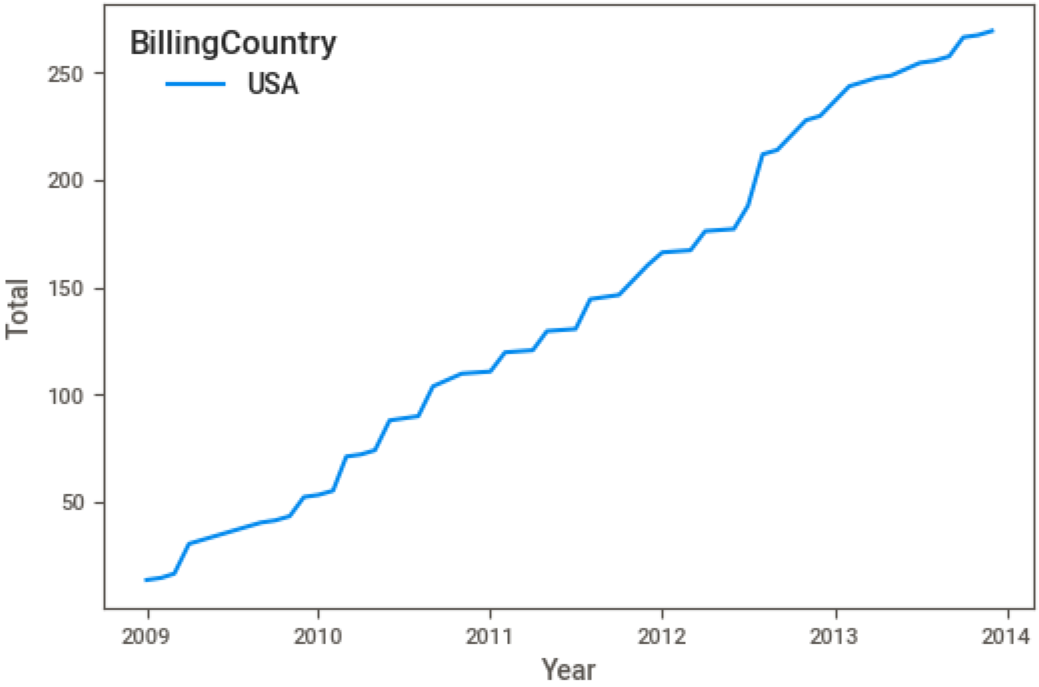
```
# convert to datetime for better plotting
sql_df2['month-year'] = pd.to_datetime(sql_df2['month-year'])
```

In [ ]:

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

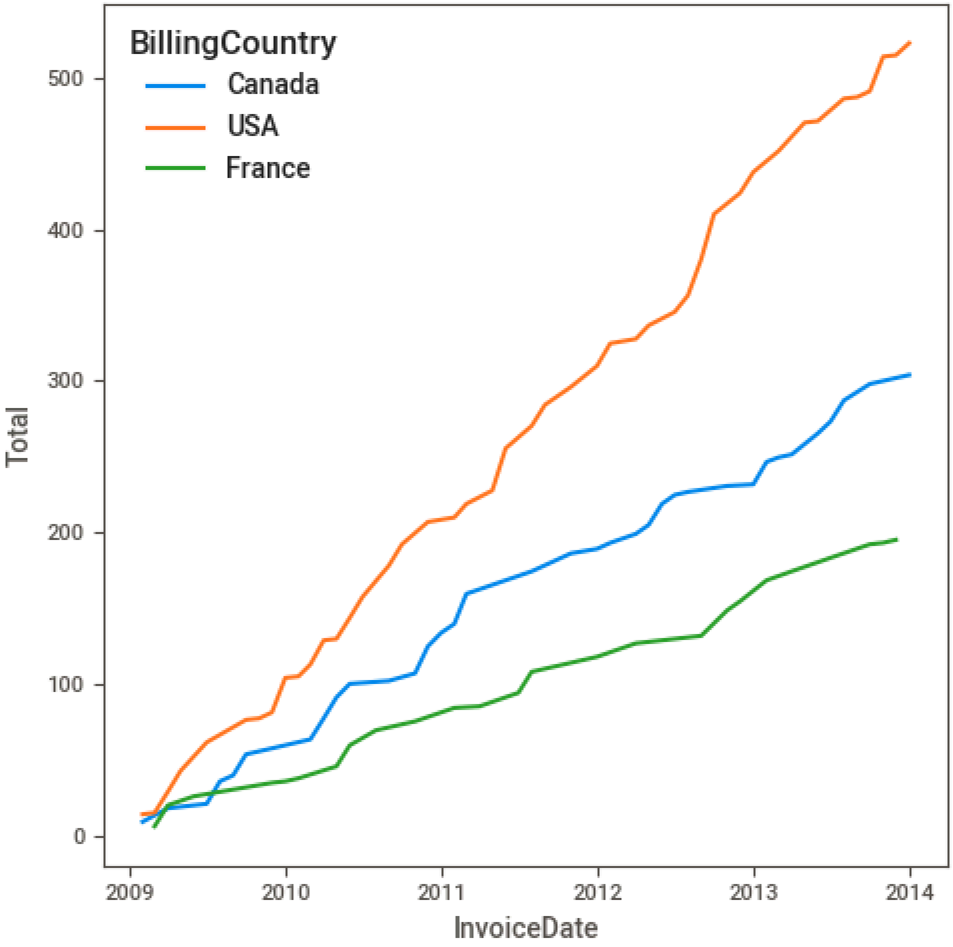
Out[ ]:

```
Text(0.5, 0, 'Year')
```



Saving Images

In [ ]:
```
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 [ ]:

```
import plotly.express as px
px.line(gb, x='InvoiceDate', y='Total', color='BillingCountry', template='si
```

In [ ]:

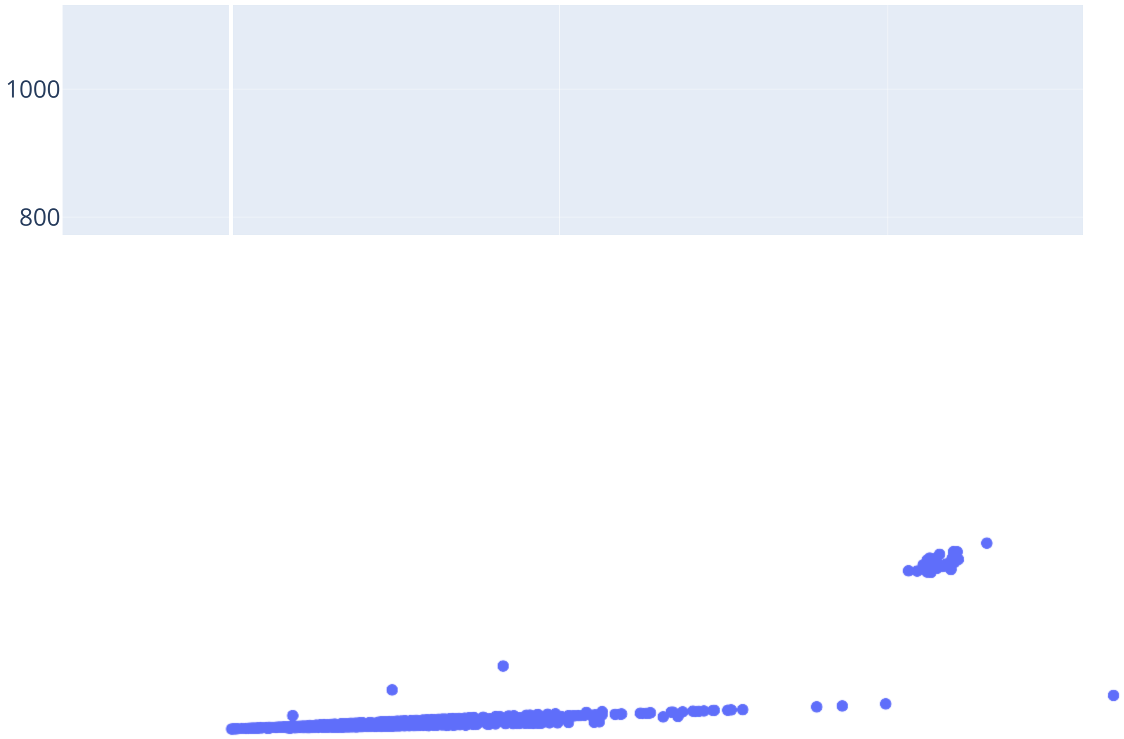
```
# recall our original DataFrame has information on the tracks in our iTunes
df.head()
```

Out[ ]:

	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

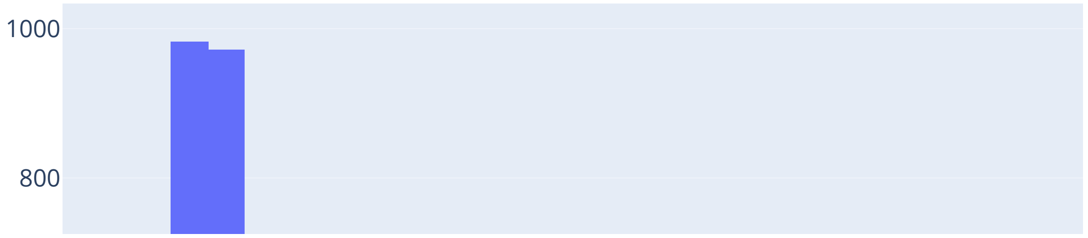
In [ ]:

```
px.scatter(df, x='Minutes', y='MB')
```



In [ ]:

```
px.histogram(df, x='Minutes')
```



In [ ]:

```
# recall this is our invoices table from the chinook iTunes database
sql_df.head()
```

Out[ ]:

	InvoiceId	CustomerId	BillingAddress	BillingCity	BillingState	BillingCountry
InvoiceDate						
2009-01-01	1	2	Theodor-Heuss-Straße 34	Stuttgart	None	Germany
2009-01-02	2	4	Ullevålsveien 14	Oslo	None	Norway
2009-01-03	3	8	Grétrystraat 63	Brussels	None	Belgium
2009-01-06	4	14	8210 111 ST NW	Edmonton	AB	Canada
2009-01-11	5	23	69 Salem Street	Boston	MA	USA

In [ ]:

```
gb_countries = sql_df.groupby('BillingCountry').sum()
gb_countries.reset_index(inplace=True)
gb_countries.head()
```

Out[ ]:

	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 [ ]:

```
px.choropleth(gb_countries, locations="BillingCountry",
              locationmode='country names',
              color="Total")
```