In [2]:
```
# Import the needed libraries. In this case, I needed Pandas to work with the data frame, Matpl
# Do not forget to install the libraries first using !pip install in Jupyter or pip install in
```

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy.stats import chi2_contingency
```

In [40]:
```
# load the data. Preparing the dataset is the most crucial part: without a perfectly workable d
# In this case, I cleaned using Excel, so I did not have to use Pandas here to clean the data.
```

In [3]:
```
try:
    df = pd.read_excel("C:/Users/lucas/OneDrive/Documentos/Sample Project.xlsx")
except FileNotFoundError:
    print("Error: File not found. Please check the path.")
    exit()
```

In [4]:
```
# First, summarise your data to check if the column names and their types are correct; otherwis
```

In [5]:
```
print("\nData Summary:")
print(df.info())
```

```
Data Summary:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 463 entries, 0 to 462
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   MSS name              463 non-null    object
 1   Author                456 non-null    object
 2   Genre                 459 non-null    object
 3   Catholic Origin       463 non-null    int64
 4   Epistolary work       463 non-null    int64
 5   Pedagogical Material  462 non-null    float64
 6   Timeframe             463 non-null    int64
 7   Accessibility         463 non-null    object
 8   Place of Storage      463 non-null    object
 9   Section in Catalogi   463 non-null    object
 10  Current Identifier    463 non-null    object
dtypes: float64(1), int64(3), object(7)
memory usage: 39.9+ KB
None
```

In [6]:
```
# ALWAYS describe the data in order to have a general look at the data.
```

In [7]:
```
print("\nData Description:")
print(df.describe(include='all'))
```

Data Description:

	MSS name	Author	Genre
count	463	456	459
unique	456	104	53
top	Johannes glo. ii. fo., " suos tam."	Augustine	Phil. Treatise
freq	2	88	122
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	Catholic Origin	Epistolary work	Pedagogical Material	Timeframe
count	463.000000	463.000000	462.000000	463.000000
unique	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN
mean	0.758099	0.205184	0.339827	2.503240
std	0.428698	0.404273	0.474164	0.901832
min	0.000000	0.000000	0.000000	1.000000
25%	1.000000	0.000000	0.000000	1.000000
50%	1.000000	0.000000	0.000000	3.000000
75%	1.000000	0.000000	1.000000	3.000000
max	1.000000	1.000000	1.000000	4.000000

	Accessibility	Place of Storage
count	463	463
unique	7	10
top	Public	Armariolo
freq	241	201
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN
max	NaN	NaN

	Section in Catalogi	Current Identifier
count	463	463
unique	15	70
top	Isti Libri Infra Scripti Inventi Fuerunt in Co...	Non-surviving
freq	115	367
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN
max	NaN	NaN

In [8]:
```
# Here is the list of all authors. Very commonly, one manuscript can have multiple authors. Fo
```

In [9]:
```
all_authors = []
for auth_str in df['Author'].dropna():
    auths = [auth.strip() for auth in auth_str.split('+')]
    for auth in auths:
        if auth.upper() != "NA":
            all_authors.append(auth)
```

```python
author_counts = Counter(all_authors)

print("\nAuthor Counts (Descending Order):")
for author, count in author_counts.most_common():
    print(f"{author}: {count}")
```

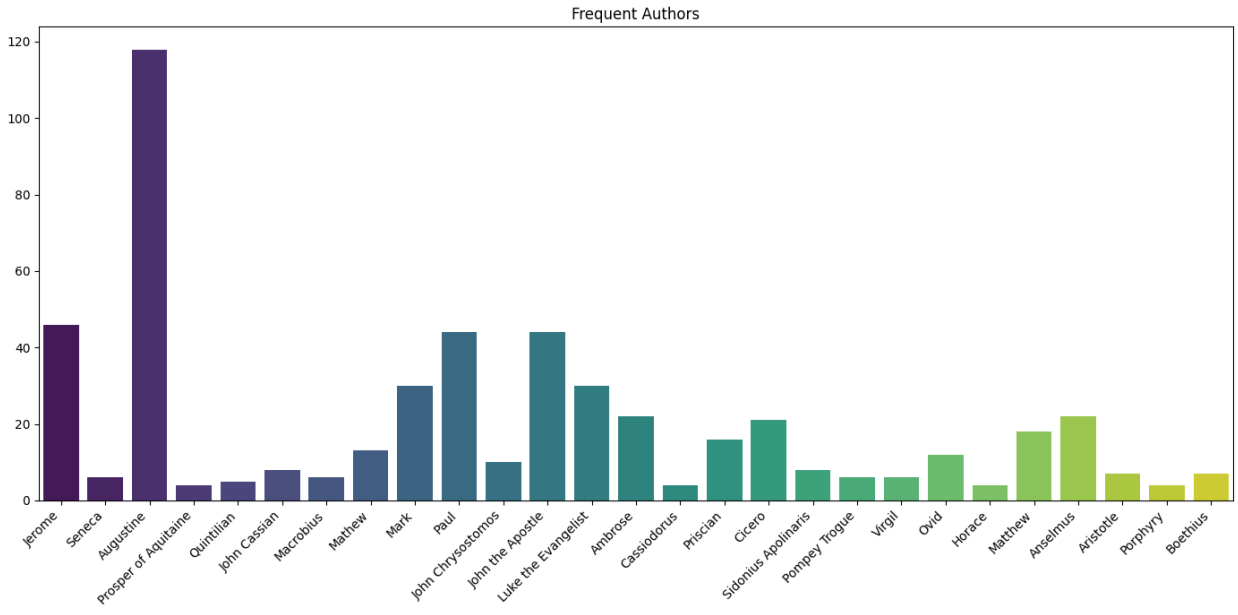
Author Counts (Descending Order):
Augustine: 118
Jerome: 46
Paul: 44
John the Apostle: 44
Mark: 30
Luke the Evangelist: 30
Ambrose: 22
Anselmus: 22
Cicero: 21
Matthew: 18
Priscian: 16
Mathew: 13
Ovid: 12
John Chrysostomos: 10
John Cassian: 8
Sidonius Apolinaris: 8
Aristotle: 7
Boethius: 7
Seneca: 6
Macrobius: 6
Pompey Trogue: 6
Virgil: 6
Quintilian: 5
Prosper of Aquitaine: 4
Cassiodorus: 4
Horace: 4
Porphyry: 4
Prudentius: 3
Cyprian: 3
Terence: 3
Lucan: 3
Juvenal: 3
Donatus: 3
Statius: 2
Theodolus: 2
Aulus Gellius: 2
Tertullian: 2
Eusebius: 2
Clemens of Rome: 2
Victorinus: 1
Remigius of Reims: 1
Eutropis: 1
Esopus: 1
Servius: 1
Marcus Aurelius: 1
Homer: 1
Claudius: 1
Avian: 1
Persius: 1
Cato: 1
Persio: 1
Arator: 1
Martial: 1
Augutine: 1
Plato: 1
Valerius Maximus: 1
Palladius: 1
Justinus: 1
Tropius: 1
Paul Orosius: 1
Claudian: 1

```
Ennodius: 1
Sallust: 1
Prudence: 1
Justinian: 1
Josephus: 1
Fulgentius: 1
Orosius: 1
Epictetus: 1
Augustinus: 1
Rufinus of Aquileia: 1
Hilarius: 1
Julianus Pomerius: 1
Pliny the Elder: 1
Ciryl of Alexandria: 1
```

In [10]:
```
# Visualising the most frequent authors. I have ignored any author with less than 3 entries si
```

In [11]:
```
authors = []
for auth_str in df['Author'].dropna():
    auths = [auth.strip() for auth in auth_str.split('+')]
    for auth in auths:
        if auth.upper() != "NA":
            authors.append(auth)

author_counts = Counter(authors)
frequent_authors = {k: v for k, v in author_counts.items() if v > 3}

plt.figure(figsize=(14, 7))
sns.barplot(x=list(frequent_authors.keys()), y=list(frequent_authors.values()), palette='viridi
plt.xticks(rotation=45, ha='right')
plt.title('Frequent Authors')
plt.tight_layout()
plt.show()
```

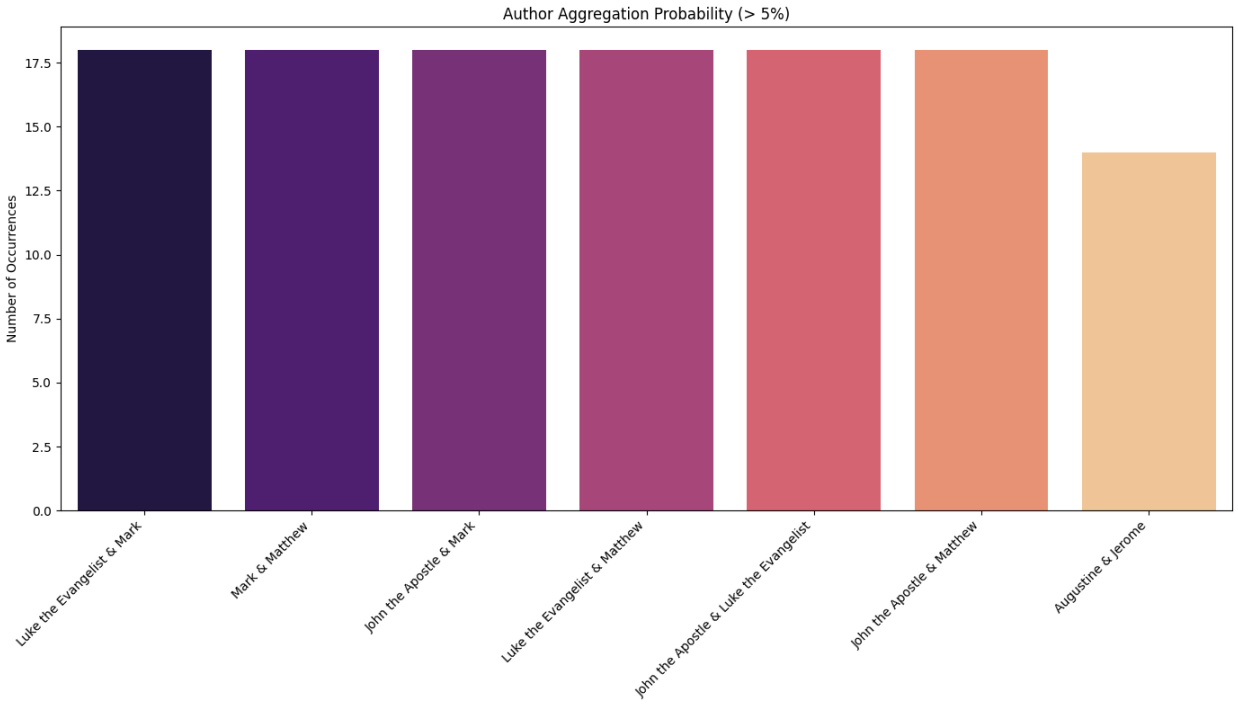
```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\1950978962.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(frequent_authors.keys()), y=list(frequent_authors.values()), palette='viri
dis')
```



In [12]:
```
# An analysis of the most frequent authors combined in the same manuscript. Logically, the ones
```

In [13]:
```
author_pairs = []
for auth_str in df['Author'].dropna():
    auths = [auth.strip() for auth in auth_str.split('+')]
    valid_auths = [auth for auth in auths if auth.upper() != "NA"]
    if len(valid_auths) > 1:
        for i in range(len(valid_auths)):
            for j in range(i + 1, len(valid_auths)):
                author_pairs.append(tuple(sorted((valid_auths[i], valid_auths[j]))))

pair_counts = Counter(author_pairs)

total_pairs = sum(pair_counts.values())

filtered_pairs = {pair: count for pair, count in pair_counts.items() if (count / total_pairs)

plt.figure(figsize=(14, 8))
if filtered_pairs:
    pair_labels = [f"{pair[0]} & {pair[1]}" for pair in filtered_pairs.keys()]
    sns.barplot(x=pair_labels, y=list(filtered_pairs.values()), palette='magma')
    plt.xticks(rotation=45, ha='right')
    plt.title("Author Aggregation Probability (> 5%)")
    plt.ylabel("Number of Occurrences")
    plt.tight_layout()
    plt.show()
else:
    print("No author pairs with more than 5% probability.")
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\2649549164.py:19: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=pair_labels, y=list(filtered_pairs.values()), palette='magma')
```



In [14]:
```
# A list of all genres. Naming and cataloguing a genre is always tricky: we may name genres di
# Therefore, I tried to be flat and generic in naming them.
```

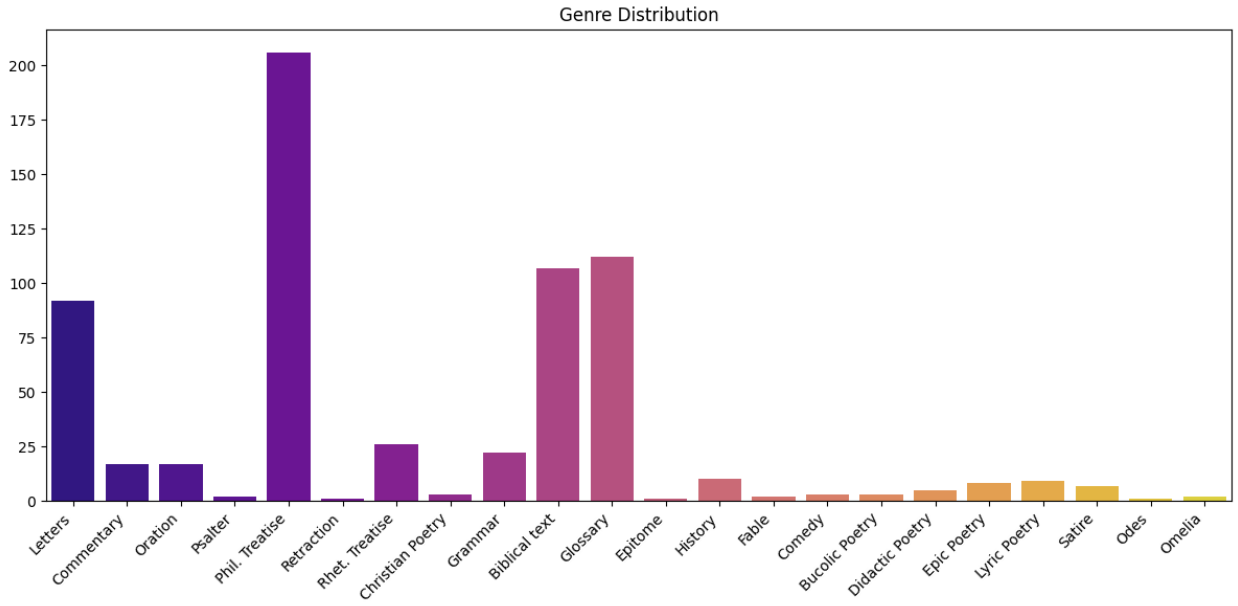
In [15]:
```
all_genres = []
for genre_str in df['Genre'].dropna():
    genres = [genre.strip() for genre in genre_str.split('+')]
    for genre in genres:
        all_genres.append(genre)

genre_counts = Counter(all_genres)

print("\nGenre Counts (Descending Order):")
for genre, count in genre_counts.most_common():
    print(f"{genre}: {count}")
```

```
Genre Counts (Descending Order):
Phil. Treatise: 206
Glossary: 112
Biblical text: 107
Letters: 92
Rhet. Treatise: 26
Grammar: 22
Commentary: 17
Oration: 17
History: 10
Lyric Poetry: 9
Epic Poetry: 8
Satire: 7
Didactic Poetry: 5
Christian Poetry: 3
Comedy: 3
Bucolic Poetry: 3
Psalter: 2
Fable: 2
Omelia: 2
Retraction: 1
Epitome: 1
Odes: 1
```

In [16]:
```
# Visualisation for the genres.
```

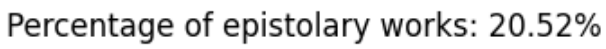
In [17]:
```
genres = []
for genre_str in df['Genre'].dropna():
    genres.extend([genre.strip() for genre in genre_str.split('+')])

genre_counts = Counter(genres)
plt.figure(figsize=(12, 6))
sns.barplot(x=list(genre_counts.keys()), y=list(genre_counts.values()), palette='plasma')
plt.xticks(rotation=45, ha='right')
plt.title('Genre Distribution')
plt.tight_layout()
plt.show()
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\2841190918.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(genre_counts.keys()), y=list(genre_counts.values()), palette='plasma')
```



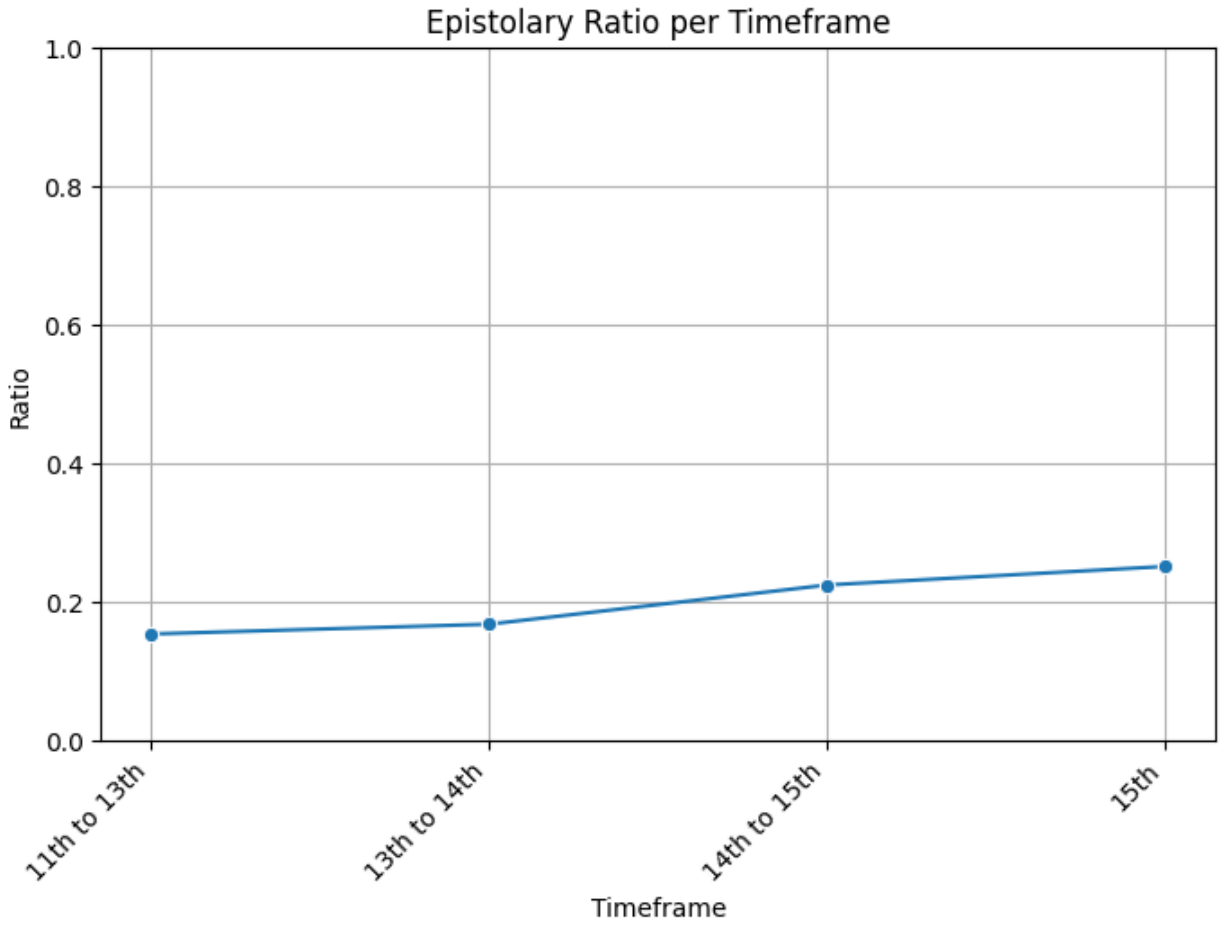
In [18]:
```
# Percentage of epistolary works considering all the manuscripts.
```

In [19]:
```
total_epistolary = df['Epistolary work'].sum()
total_works = len(df)
epistolary_ratio_general = total_epistolary / total_works

plt.figure(figsize=(6, 1))
plt.text(0.5, 0.5, f"Percentage of epistolary works: {epistolary_ratio_general:.2%}", ha='cent
plt.axis('off')
plt.show()
```



In [20]:
```
# Considering the works catalogued in each century, I calculate the evolution of the percentage
```

In [21]:
```
epistolary_ratios_timeframe = df.groupby('Timeframe')['Epistolary work'].mean()

plt.figure(figsize=(8, 5))
sns.lineplot(x=epistolary_ratios_timeframe.index, y=epistolary_ratios_timeframe.values, marker=
plt.xticks(epistolary_ratios_timeframe.index, labels = ["11th to 13th", "13th to 14th", "14th
plt.title('Epistolary Ratio per Timeframe')
plt.xlabel("Timeframe")
plt.ylabel("Ratio")
plt.ylim(0, 1)
plt.grid(True)
plt.show()

print("\nEpistolary Ratios per Timeframe:")
print(epistolary_ratios_timeframe)
```




```
Epistolary Ratios per Timeframe:
Timeframe
1    0.152542
2    0.166667
3    0.223242
4    0.250000
Name: Epistolary work, dtype: float64
```

In [22]:
```
# To better evaluate whether there were significant changes in the epistolary works per cent, 
```

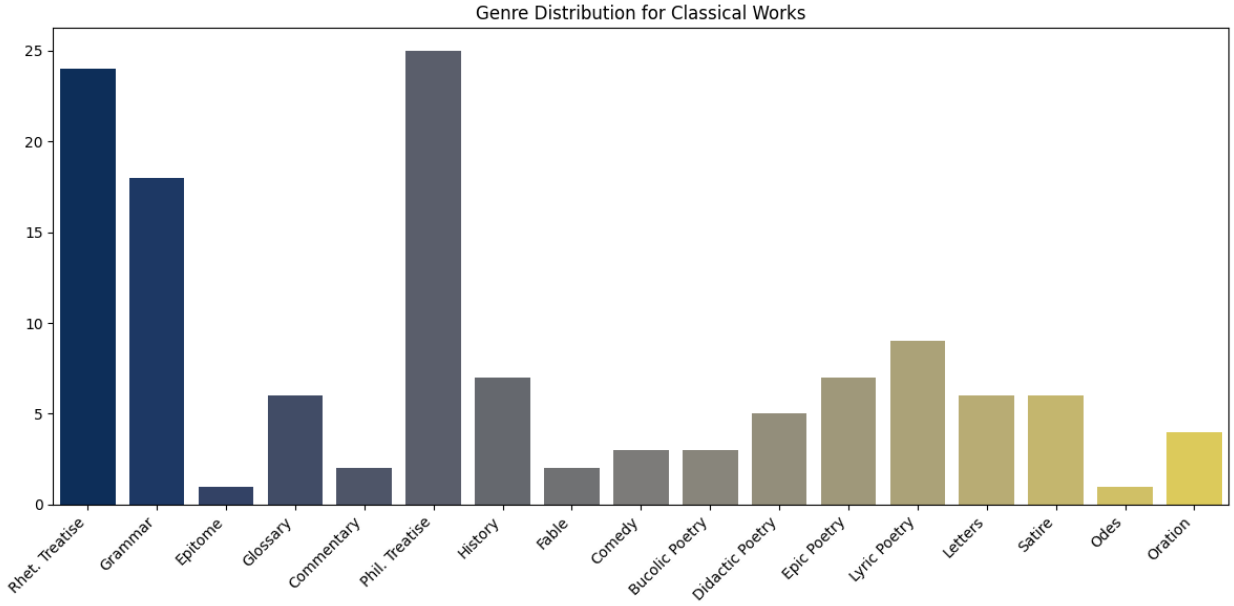
In [23]:
```
change = epistolary_ratios_timeframe.iloc[-1] - epistolary_ratios_timeframe.iloc[0]
print(f"The epistolary ratio changed by {change:.2%} from the first to the last timeframe.")
```

```
The epistolary ratio changed by 9.75% from the first to the last timeframe.
```

In [24]:
```
# Genres written only by Classical authors (non-Christian).
```

In [25]:
```
non_catholic_df = df[df['Catholic Origin'] == 0]

genres_non_catholic = []
for genre_str in non_catholic_df['Genre'].dropna():
    genres_non_catholic.extend([genre.strip() for genre in genre_str.split('+')])

genre_counts_non_catholic = Counter(genres_non_catholic)

plt.figure(figsize=(12, 6))
sns.barplot(x=list(genre_counts_non_catholic.keys()), y=list(genre_counts_non_catholic.values(
plt.xticks(rotation=45, ha='right')
plt.title('Genre Distribution for Classical Works')
plt.tight_layout()
plt.show()
```

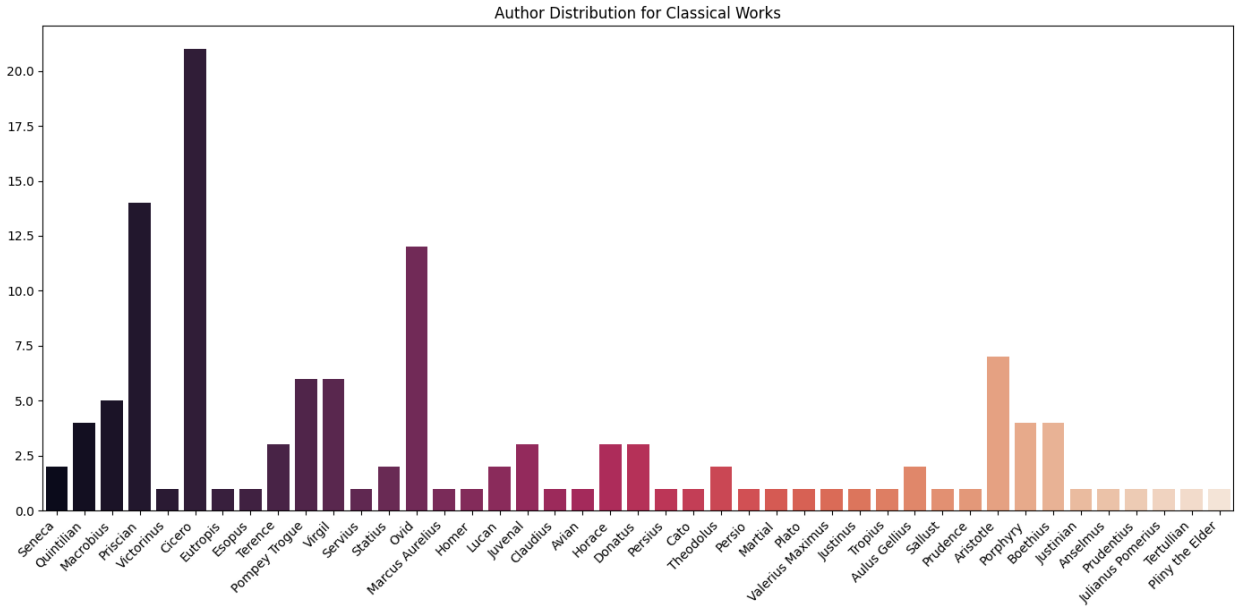
```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\3307341772.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(genre_counts_non_catholic.keys()), y=list(genre_counts_non_catholic.values
())), palette='cividis')
```



In [26]:
```
# Classical authors.
```

In [27]:
```
authors_non_catholic = []
for auth_str in non_catholic_df['Author'].dropna():
    auths = [auth.strip() for auth in auth_str.split('+')]
    for auth in auths:
        if auth.upper() != "NA":
            authors_non_catholic.append(auth)

author_counts_non_catholic = Counter(authors_non_catholic)

plt.figure(figsize=(14, 7))
sns.barplot(x=list(author_counts_non_catholic.keys()), y=list(author_counts_non_catholic.value
plt.xticks(rotation=45, ha='right')
plt.title('Author Distribution for Classical Works')
plt.tight_layout()
plt.show()
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\4253173714.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(author_counts_non_catholic.keys()), y=list(author_counts_non_catholic.valu
es()), palette='rocket')
```



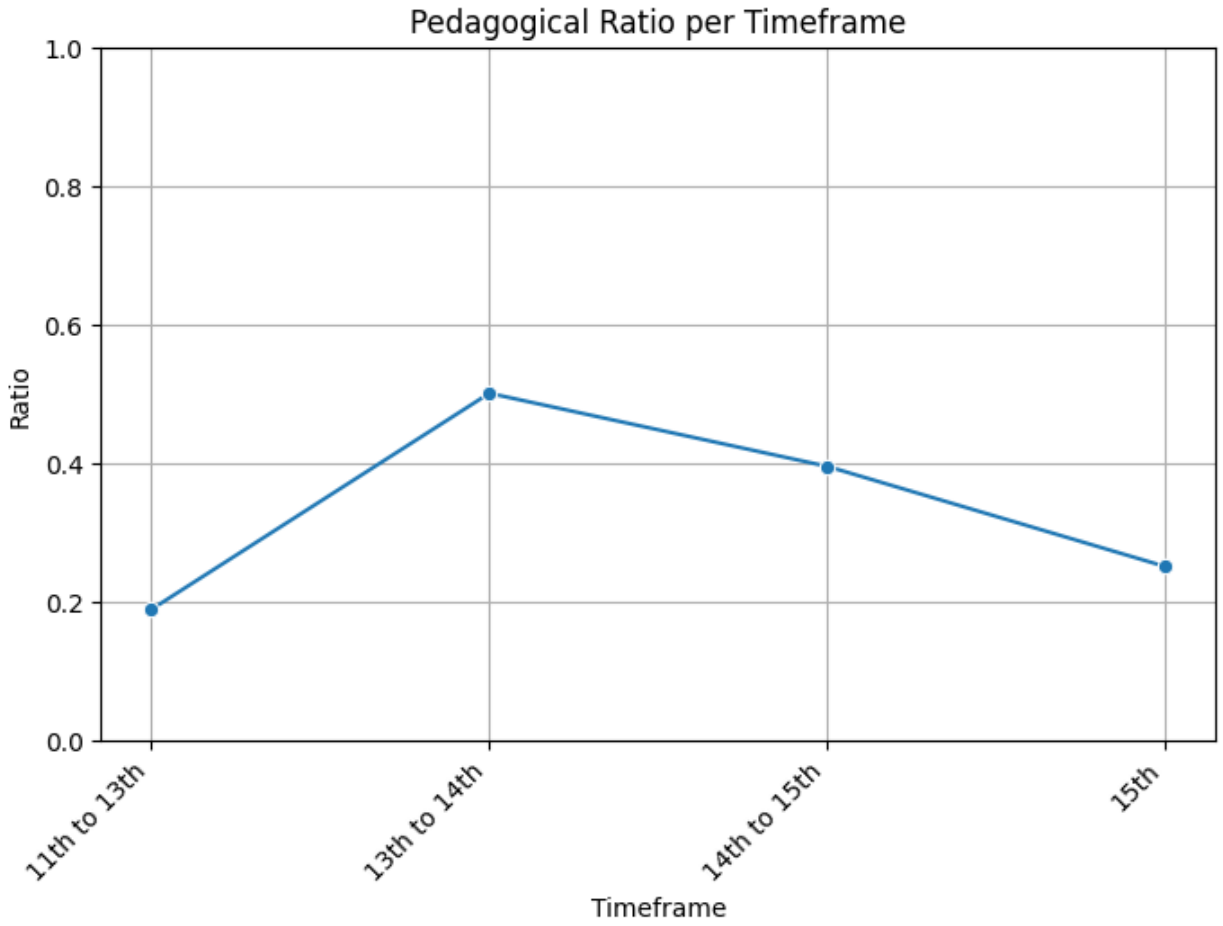
In [28]:
```
# Percentage of manuscripts with educational material, be it a Glossary, rhetorical treatise or
```

In [29]:
```
total_pedagogical = df['Pedagogical Material'].sum()
total_works = len(df)
pedagogical_ratio_general = total_pedagogical / total_works


plt.figure(figsize=(6, 1))
plt.text(0.5, 0.5, f"Percentage of works related to educational material: {pedagogical_ratio_g
plt.axis('off')
plt.show()
```

Percentage of works related to educational material: 33.91%

In [30]:
```
# The percentage of educational material in each data frame is increased.
```

In [31]:
```
pedagogical_ratios_timeframe = df.groupby('Timeframe')['Pedagogical Material'].mean()

plt.figure(figsize=(8, 5))
sns.lineplot(x=pedagogical_ratios_timeframe.index, y=pedagogical_ratios_timeframe.values, mark
plt.xticks(pedagogical_ratios_timeframe.index, labels = ["11th to 13th", "13th to 14th", "14th
plt.title('Pedagogical Ratio per Timeframe')
plt.xlabel("Timeframe")
plt.ylabel("Ratio")
plt.ylim(0, 1)
plt.grid(True)
plt.show()

print("\nEvolution of educational material per timeframe:")
print(pedagogical_ratios_timeframe)
```




```
Evolution of educational material per timeframe:
Timeframe
1    0.188034
2    0.500000
3    0.394495
4    0.250000
Name: Pedagogical Material, dtype: float64
```

In [32]:
```
# How much the per cent of educational material change from the first to the last timeframe.
```

In [33]:
```
change = pedagogical_ratios_timeframe.iloc[-1] - pedagogical_ratios_timeframe.iloc[0]
print(f"The pedagogical ratio changed by {change:.2%} from the first to the last timeframe.")
```

```
The pedagogical ratio changed by 6.20% from the first to the last timeframe.
```

In [34]:
```
df['Current Identifier'] = df['Current Identifier'].astype(str)
```

In [35]:
```
existing_df = df[df['Current Identifier'] != "Non-Surviving"]
```

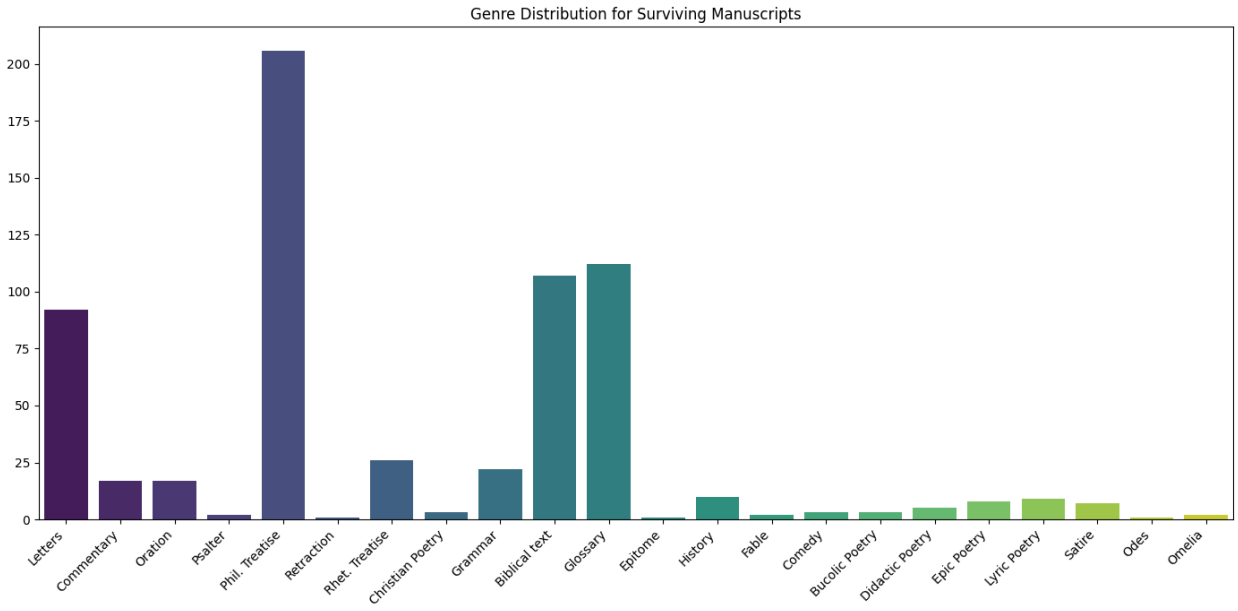
In [36]:
```
# Considering only the surviving manuscripts, the number of authors and genres that survived.
```

In [37]:
```
genres_existing = []
for genre_str in existing_df['Genre'].dropna():
    genres_existing.extend([genre.strip() for genre in genre_str.split('+')])

genre_counts_existing = Counter(genres_existing)
```

In [38]:
```
plt.figure(figsize=(14, 7))
sns.barplot(x=list(genre_counts_existing.keys()), y=list(genre_counts_existing.values()), pale
plt.xticks(rotation=45, ha='right')
```

```
plt.title('Genre Distribution for Surviving Manuscripts')
plt.tight_layout()
plt.show()
```

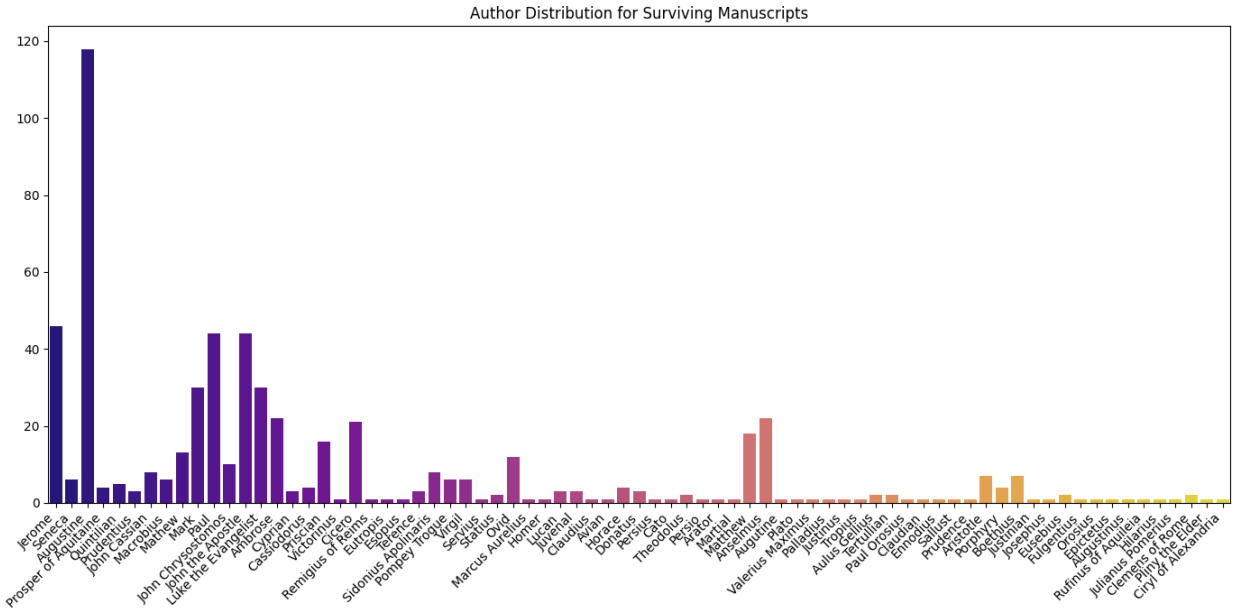
```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\2173918355.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(genre_counts_existing.keys()), y=list(genre_counts_existing.values()), pal
ette='viridis')
```



In [39]:
```
authors_existing = []
for auth_str in existing_df['Author'].dropna():
    auths = [auth.strip() for auth in auth_str.split('+')]
    for auth in auths:
        if auth.upper() != "NA":
            authors_existing.append(auth)
```

In [40]:
```
author_counts_existing = Counter(authors_existing)

plt.figure(figsize=(14, 7))
sns.barplot(x=list(author_counts_existing.keys()), y=list(author_counts_existing.values()), pa
plt.xticks(rotation=45, ha='right')
plt.title('Author Distribution for Surviving Manuscripts')
plt.tight_layout()
plt.show()
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\3587652149.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(author_counts_existing.keys()), y=list(author_counts_existing.values()), p
alette='plasma')
```



In [41]:
```
# Percentage of authors who were also letter-writers.
```

In [42]:
```
letter_authors = []
for index, row in df.iterrows():
    if (isinstance(row['Genre'], str) and 'Letters' in row['Genre']) or row['Epistolary work']
        auth_str = row['Author']
        if isinstance(auth_str, str):  # Check if auth_str is a string *before* splitting
            auths = [auth.strip() for auth in auth_str.split('+')]
            for auth in auths:
                if auth.upper() != "NA":
                    letter_authors.append(auth)

letter_author_counts = Counter(letter_authors)
```

In [43]:
```
total_authors_count = sum(author_counts.values())
letter_authors_count = sum(letter_author_counts.values())
percentage_letter_authors = (letter_authors_count / total_authors_count) * 100 if total_author

print(f"Percentage of authors who wrote letters: {percentage_letter_authors:.2f}%")
```

```
Percentage of authors who wrote letters: 22.83%
```

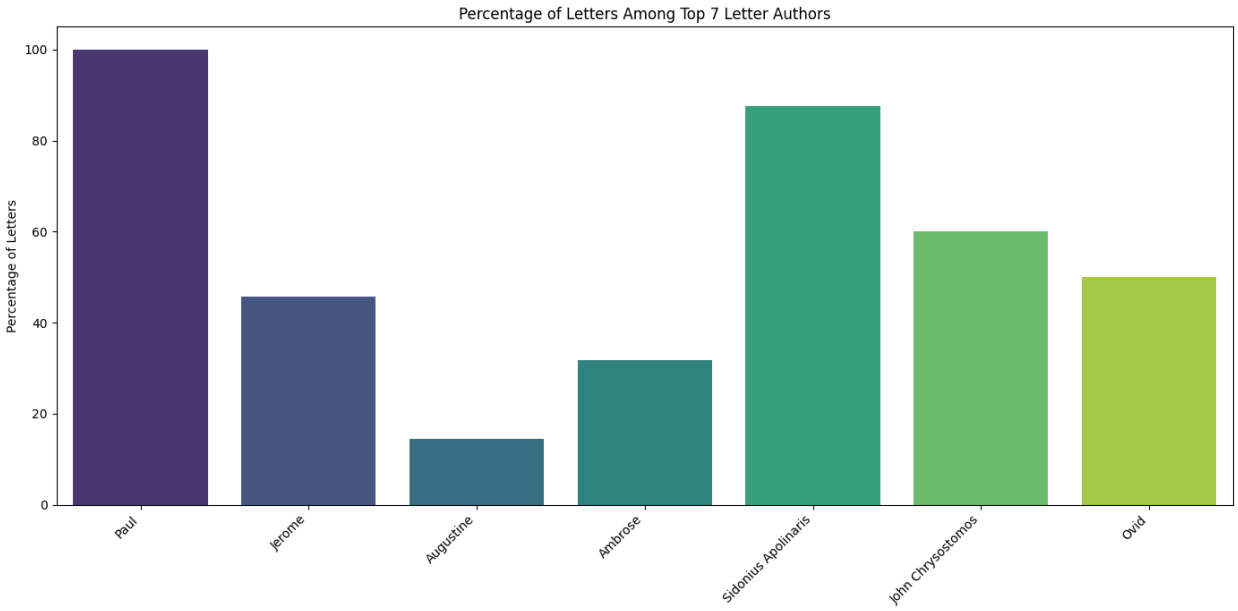
In [44]:
```
# Authors with more letters in the collection. Ovid is a surprise.
```

In [45]:
```
top_7_letter_authors = letter_author_counts.most_common(7)

top_author_letter_percentages = {}
for author, letter_count in top_7_letter_authors:
    author_total_count = author_counts.get(author, 0) #Handle cases where author is not in the
    percentage = (letter_count / author_total_count) * 100 if author_total_count > 0 else 0
    top_author_letter_percentages[author] = percentage
```

In [46]:
```
authors = list(top_author_letter_percentages.keys())
letter_percentages = list(top_author_letter_percentages.values())

plt.figure(figsize=(14, 7))
sns.barplot(x=authors, y=letter_percentages, palette='viridis')
plt.xticks(rotation=45, ha='right')
plt.title('Percentage of Letters Among Top 7 Letter Authors')
plt.ylabel('Percentage of Letters')
```

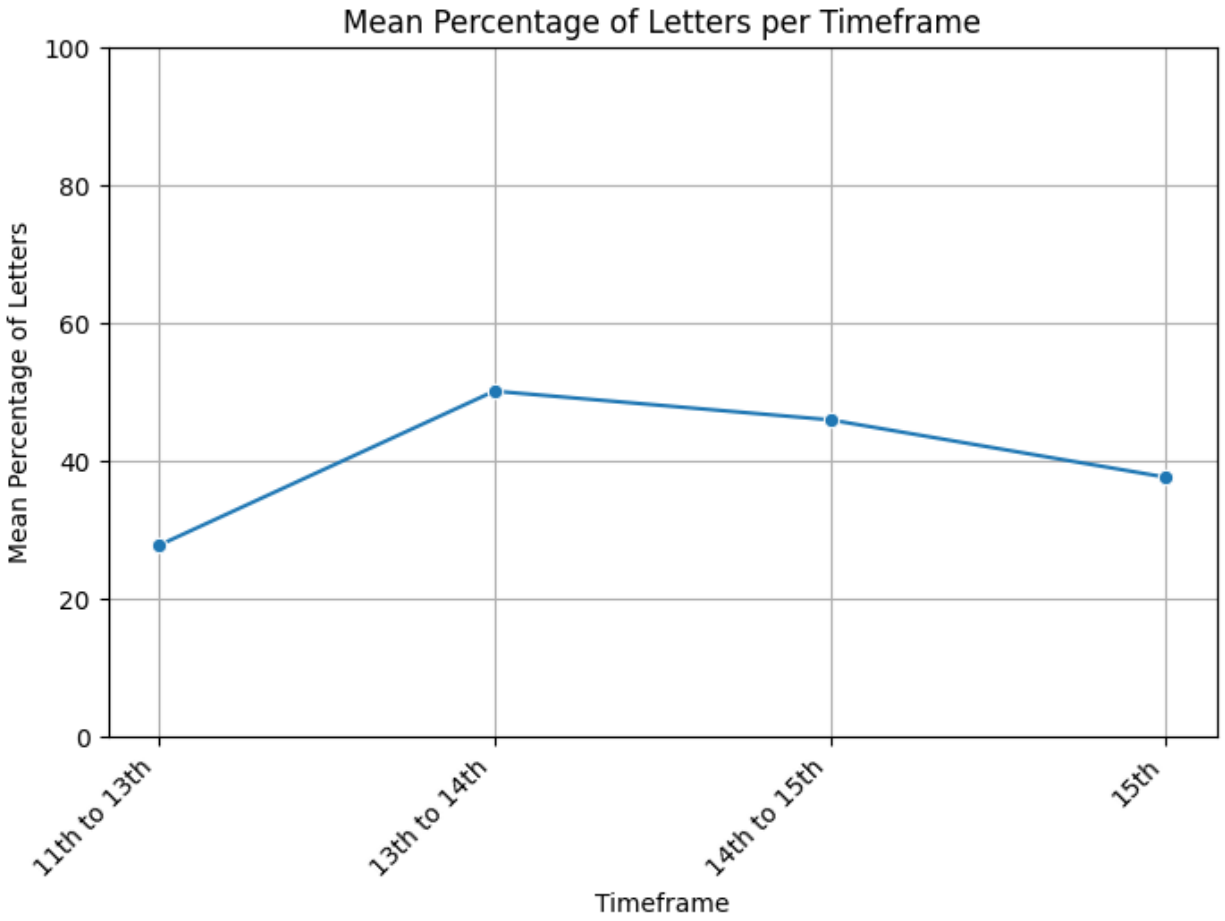
```
plt.tight_layout()
plt.show()
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\3452270182.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign
the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=authors, y=letter_percentages, palette='viridis')
```



In [47]:
```
timeframe_letter_percentages = df.groupby('Timeframe').apply(lambda x: (x[(x['Genre'].str.conta

plt.figure(figsize=(8, 5))
sns.lineplot(x=timeframe_letter_percentages.index, y=timeframe_letter_percentages.values, mark
plt.xticks(timeframe_letter_percentages.index, labels = ["11th to 13th", "13th to 14th", "14th
plt.title('Mean Percentage of Letters per Timeframe')
plt.xlabel('Timeframe')
plt.ylabel('Mean Percentage of Letters')
plt.ylim(0, 100)
plt.grid(True)
plt.show()
```

```
C:\Users\lucas\AppData\Local\Temp\ipykernel_24612\3585540898.py:1: DeprecationWarning: DataFram
eGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future v
ersion of pandas the grouping columns will be excluded from the operation. Either pass `include
_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby
to silence this warning.
  timeframe_letter_percentages = df.groupby('Timeframe').apply(lambda x: (x[(x['Genre'].str.con
tains('Letters', na=False)) | (x['Epistolary work'] == 1)]['Author'].dropna().apply(lambda auth
ors: [a.strip() for a in authors.split('+')]).explode().isin(authors).sum() / x['Author'].dropn
a().apply(lambda authors: [a.strip() for a in authors.split('+')]).explode().isin(authors).sum
())*100)
```



In [48]:
```
# Applying statistical models to see if there are correlations between the variables.
```

In [49]:
```
categorical_cols = ['Genre', 'Author', 'Epistolary work', 'Pedagogical Material', 'Current Ide

for col1 in categorical_cols:
    for col2 in categorical_cols:
        if col1 != col2:  # Avoid comparing a column to itself
            try:
                # Create contingency table
                contingency_table = pd.crosstab(df[col1], df[col2])

                #Check if there is data to do the test
                if contingency_table.shape[0] > 1 and contingency_table.shape[1] > 1:
                    # Perform chi-square test
                    chi2, p, dof, expected = chi2_contingency(contingency_table)

                    print(f"\nChi-Square Test: {col1} vs. {col2}")
                    print(f"Chi-square statistic: {chi2}")
                    print(f"P-value: {p}")
                    if p < 0.05:
                        print("There is a statistically significant association between the va
                    else:
                        print("There is no statistically significant association between the v
                else:
                    print(f"\nChi-Square Test: {col1} vs. {col2}")
                    print("Not enough data to perform the Chi-Square Test")
            except Exception as e:
                print(f"Error performing Chi-Square Test for {col1} vs. {col2}: {e}")
```

```
Chi-Square Test: Genre vs. Author
Chi-square statistic: 11149.012147685396
P-value: 0.0
There is a statistically significant association between the variables.

Chi-Square Test: Genre vs. Epistolary work
Chi-square statistic: 413.6963083403354
P-value: 8.380157688583263e-58
There is a statistically significant association between the variables.

Chi-Square Test: Genre vs. Pedagogical Material
Chi-square statistic: 290.24487617963155
P-value: 8.095015121905868e-35
There is a statistically significant association between the variables.

Chi-Square Test: Genre vs. Current Identifier
Chi-square statistic: 2484.899338687168
P-value: 1.0
There is no statistically significant association between the variables.

Chi-Square Test: Author vs. Genre
Chi-square statistic: 11149.012147685396
P-value: 0.0
There is a statistically significant association between the variables.

Chi-Square Test: Author vs. Epistolary work
Chi-square statistic: 311.8205304740274
P-value: 7.269163507211365e-23
There is a statistically significant association between the variables.

Chi-Square Test: Author vs. Pedagogical Material
Chi-square statistic: 237.41108115625434
P-value: 1.2823541664465012e-12
There is a statistically significant association between the variables.

Chi-Square Test: Author vs. Current Identifier
Chi-square statistic: 6839.877396650207
P-value: 0.6971208442068523
There is no statistically significant association between the variables.

Chi-Square Test: Epistolary work vs. Genre
Chi-square statistic: 413.6963083403355
P-value: 8.380157688582784e-58
There is a statistically significant association between the variables.

Chi-Square Test: Epistolary work vs. Author
Chi-square statistic: 311.82053047402746
P-value: 7.269163507211159e-23
There is a statistically significant association between the variables.

Chi-Square Test: Epistolary work vs. Pedagogical Material
Chi-square statistic: 5.01723136366571
P-value: 0.025096265441536227
There is a statistically significant association between the variables.

Chi-Square Test: Epistolary work vs. Current Identifier
Chi-square statistic: 107.428675707231
P-value: 0.0021060007333283935
There is a statistically significant association between the variables.

Chi-Square Test: Pedagogical Material vs. Genre
Chi-square statistic: 290.2448761796316
```

```
P-value: 8.09501512190587e-35
There is a statistically significant association between the variables.

Chi-Square Test: Pedagogical Material vs. Author
Chi-square statistic: 237.41108115625434
P-value: 1.2823541664465012e-12
There is a statistically significant association between the variables.

Chi-Square Test: Pedagogical Material vs. Epistolary work
Chi-square statistic: 5.01723136366571
P-value: 0.025096265441536227
There is a statistically significant association between the variables.

Chi-Square Test: Pedagogical Material vs. Current Identifier
Chi-square statistic: 86.09013260937661
P-value: 0.07997556761509794
There is no statistically significant association between the variables.

Chi-Square Test: Current Identifier vs. Genre
Chi-square statistic: 2484.8993386871684
P-value: 1.0
There is no statistically significant association between the variables.

Chi-Square Test: Current Identifier vs. Author
Chi-square statistic: 6839.877396650207
P-value: 0.6971208442068523
There is no statistically significant association between the variables.

Chi-Square Test: Current Identifier vs. Epistolary work
Chi-square statistic: 107.42867570723101
P-value: 0.0021060007333283622
There is a statistically significant association between the variables.

Chi-Square Test: Current Identifier vs. Pedagogical Material
Chi-square statistic: 86.09013260937662
P-value: 0.07997556761509785
There is no statistically significant association between the variables.
```

In [ ]: