🧠 Notebook Overview

This notebook investigates whether **topic-based features** extracted from financial news articles can effectively **predict stock price movements**.

We replace traditional token-based sentiment scores with topic modeling features, including:

- `topic_avg_movement`
- `topic_sensitivity`
- `market_volatility`
- `sentiment_impact`

The ultimate goal is to evaluate the predictive power of these features at a per-ticker level and explore the relationship between article volume and model performance.

📊 Data Overview

We begin by exploring the dataset: article distribution, sample entries, and summary statistics for key variables.

🔍 Topic Modeling

Topic modeling was applied to identify underlying themes in financial articles using [INSERT METHOD USED HERE, e.g., BERTopic or LDA].

Each article was assigned a distribution of topics, which were then aggregated at the ticker-day level.

- **Number of topics:** [Insert]
- **Example topics and keywords:** [Insert or visualize]

🛠️ Feature Engineering

We generated several topic-based features per article, then aggregated them per ticker-day:

- **`topic_avg_movement`** : historical average return for each topic.
- **`topic_sensitivity`** : standard deviation of returns associated with the topic.
- **`sentiment_impact`** : derived from sentiment-weighted scores.
- **`market_volatility`** : contextual measure of uncertainty.

These features aim to represent latent signals potentially related to future price movement.

🤖 Per-Ticker Modeling Approach

To better understand model performance across companies, we trained separate models **per ticker** using XGBoost regressors.

We measured:

- Mean Absolute Error (MAE)
- R^2 Score
- Correlation of individual features with the price change target

📈 Performance Results

Below is the summary table containing MAE, R^2, number of articles per ticker, and feature correlations.

🔎 Exploring Article Volume vs. Model Performance

We hypothesized that model performance may be influenced by the number of articles available per ticker.

To test this, we grouped tickers by article count cutoffs and calculated:

- **Average and Median R^2**
- **Average MAE**

Findings:

- Tickers with fewer than 100 articles tend to have poor performance.
- Tickers with more than 250 articles show **significant improvement in R^2**.

📊 Visualizing Article Volume Impact

Plots below visualize the relationship between article count and model performance.

🧩 Key Insights & Lessons Learned

- Topic-based features **can predict** price movement, but only under certain conditions.
- **Data volume per ticker** is a key driver of model performance.
- A threshold of ~250 articles per ticker appears to produce more reliable and consistent results.
- Future iterations could benefit from:
  - Ticker filtering based on volume.
  - Data augmentation or smoothing.
  - Combining topic features with traditional fundamentals or technical indicators.

This experiementation serves as a stepping stone toward robust news-based stock prediction using topic representations.

In [26]:
```
# %% Load Libraries
import pandas as pd
import numpy as np
import duckdb
from bertopic import BERTopic
from sentence_transformers import SentenceTransformer
from umap import UMAP
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

In [27]:
```
# %% Connect to DuckDB and Load Dataset
#db_path = r'C:\Users\btada\Documents\financial_news.db'
db_path = '/Users/bradams/Documents/financial_news.db'
conn = duckdb.connect(database=db_path, read_only=False)

# %% Get All Available Tickers & Sample a Subset
query_tickers = "SELECT DISTINCT ticker FROM Headlines.Articles_Trading_Day"
ticker_df = conn.execute(query_tickers).fetchdf()

# Randomly sample 20 tickers for testing
ticker_all = random.sample(list(ticker_df["ticker"]), 50)

# Build a query to only include your sampled tickers
placeholders = ",".join(["?"] * len(ticker_all))
```

In [28]:
```
def process_ticker_topic_modeling(ticker, conn):
    query = f"""
    SELECT
        a.ticker,
        a.mapped_trading_date AS publish_date,
        a.article_title,
        a.description,
        dpm.price_change_percentage,
        f.finbert_title_score,
        f.finbert_description_score,
        f.finbert_title_positive,
```

```
        f.finbert_title_neutral,
        f.finbert_title_negative,
        f.finbert_description_positive,
        f.finbert_description_neutral,
        f.finbert_description_negative
    FROM "Headlines"."Articles_Trading_Day" a
    INNER JOIN "Headlines"."Daily_Price_Movement" dpm
        ON a.mapped_trading_date = dpm.trading_date
        AND a.ticker = dpm.ticker
    INNER JOIN "Headlines"."finbert_analysis" f
        ON a.guid = f.guid
    WHERE a.ticker = ?
    """

    news_df = conn.execute(query, [ticker]).fetchdf()
    if news_df.empty:
        return None

    # Combine text for BERTopic
    texts = news_df["article_title"].fillna("") + " " + news_df["description"].fill
    embedding_model = SentenceTransformer("paraphrase-MiniLM-L3-v2")
    topic_model = BERTopic(
        embedding_model=embedding_model,
        umap_model=UMAP(n_neighbors=5, min_dist=0.5, n_components=5),
        verbose=False
    )
    topics, probs = topic_model.fit_transform(texts.tolist())
    news_df["topic"] = topics
    news_df["topic_probability"] = [max(p) if isinstance(p, list) else 0 for p in p

    # Features
    news_df["topic_avg_movement"] = news_df.groupby("topic")["price_change_percenta
    news_df["topic_sensitivity"] = news_df.groupby("topic")["price_change_percentag
    news_df["sentiment_impact"] = news_df["finbert_description_positive"] - news_df
    news_df["market_volatility"] = news_df["price_change_percentage"].rolling(windo
    news_df["risk_score_topic"] = (
        news_df["sentiment_impact"] * news_df["topic_sensitivity"]
    ) * news_df["market_volatility"]

    # Prepare data
    features = [
        "risk_score_topic", "topic_avg_movement", "topic_sensitivity", "sentiment_i
        "finbert_title_positive", "finbert_title_negative", "finbert_description_po
    ]
    df = news_df[features + ["price_change_percentage"]].replace([np.inf, -np.inf],

    if df.shape[0] < 10:
        return None  # skip small datasets

    # Compute correlation of each feature with the target
    corr_dict = df.corr(numeric_only=True)["price_change_percentage"].drop("price_c

    # Train/test split and model training
    X = df[features]
    y = df["price_change_percentage"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random
```

```
    model = XGBRegressor(objective="reg:squarederror", n_estimators=100, learning_r
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Assemble result
    result = {
        "ticker": ticker,
        "mae": mean_absolute_error(y_test, y_pred),
        "r2": r2_score(y_test, y_pred),
        "n_articles" : len(news_df)
    }

    # Add correlation values to result
    for feature_name, corr_value in corr_dict.items():
        result[f"corr_{feature_name}"] = corr_value

    return result
```

In [29]:
```
results = []
for ticker in ticker_all:
    print(f"Processing {ticker}...")
    result = process_ticker_topic_modeling(ticker, conn)
    if result:
        results.append(result)

results_df = pd.DataFrame(results)
print(results_df.sort_values("r2", ascending=False).head(10))
```

```
Processing CHE...
Processing BILL...
Processing ESGR...
Processing FERG...
Processing DHR...
Processing BCC...
Processing QLYS...
Processing CAG...
Processing SMCI...
Processing ISRG...
Processing MSM...
Processing IRM...
Processing BALL...
Processing NFG...
Processing TDC...
Processing RACE...
Processing MTCH...
Processing AMKR...
Processing CDNS...
Processing XENE...
Processing CMS...
Processing ALIT...
Processing MDU...
Processing CXT...
Processing VMC...
Processing KMB...
Processing NEU...
Processing PXD...
Processing ALSN...
Processing RPRX...
Processing LDOS...
Processing DAR...
Processing V...
Processing ABG...
Processing MTDR...
Processing INCY...
Processing OPCH...
Processing AVTR...
Processing POWI...
Processing FN...
Processing QSR...
Processing GPI...
Processing DVN...
Processing CCCS...
Processing DRI...
Processing DASH...
Processing VTRS...
Processing LYB...
Processing AMP...
Processing BPOP...
   ticker       mae        r2  n_articles  corr_risk_score_topic  \
41    GPI  1.255465  0.604633         129               0.231671
45   DASH  0.798376  0.571386         367               0.516641
24    VMC  0.601790  0.571365         349               0.042368
9    ISRG  0.470837  0.539282         908              -0.093854
32      V  0.382790  0.524576        1629              -0.051529
```

```
30   LDOS  0.833610  0.515487         666              0.108210
27    PXD  0.431405  0.498124         331              0.100726
20    CMS  0.552501  0.474618         207             -0.172923
8    SMCI  2.813967  0.458099        3241             -0.041657
47    LYB  0.310596  0.431934         298             -0.134790

    corr_topic_avg_movement  corr_topic_sensitivity  corr_sentiment_impact  \
41                 0.297437                0.275544               0.121947
45                 0.618749               -0.355776               0.171205
24                 0.380959               -0.000942               0.014119
9                  0.369539                0.129123              -0.057205
32                 0.344834               -0.093171              -0.006534
30                 0.246499               -0.056219              -0.017878
27                 0.419630               -0.052648               0.073611
20                 0.288453               -0.115209              -0.167973
8                  0.388292               -0.007341               0.027561
47                 0.421202                0.084159              -0.052294

    corr_market_volatility  corr_finbert_title_positive  \
41                0.238386                     0.000141
45               -0.128564                     0.096084
24                0.114756                    -0.098588
9                 0.082133                     0.032454
32               -0.023893                    -0.008140
30                0.015481                     0.045720
27               -0.011941                    -0.081442
20                0.042150                    -0.029221
8                 0.100388                    -0.014152
47                0.011679                    -0.041780

    corr_finbert_title_negative  corr_finbert_description_positive  \
41                    -0.162979                          -0.075234
45                    -0.178908                           0.086113
24                    -0.053833                          -0.109258
9                     -0.113794                          -0.143188
32                     0.010671                          -0.045742
30                     0.033030                          -0.029597
27                     0.223693                           0.006029
20                     0.199777                          -0.153360
8                      0.017615                           0.026359
47                    -0.033524                          -0.089789

    corr_finbert_description_negative
41                          -0.271224
45                          -0.184741
24                          -0.098541
9                           -0.016944
32                          -0.019614
30                           0.008262
27                          -0.101036
20                           0.129263
8                           -0.020562
47                           0.029133
```

In [30]:
```
from tabulate import tabulate
```

```
print(tabulate(results_df, headers='keys', tablefmt="psql"))
```

```
+----+---------+---------+------------+-------------+-------------------------+-
-------------------------+-------------------------+-----------------------+----
---------------------+-------------------------------+---------------------------
--+-------------------------------------+-------------------------------------+
|    | ticker   |      mae |          r2 |   n_articles |   corr_risk_score_topic |
corr_topic_avg_movement |   corr_topic_sensitivity |   corr_sentiment_impact |   cor
r_market_volatility |   corr_finbert_title_positive |   corr_finbert_title_negative
|   corr_finbert_description_positive |   corr_finbert_description_negative |
|----+---------+---------+------------+-------------+-------------------------+-
-------------------------+-------------------------+-----------------------+----
---------------------+-------------------------------+---------------------------
--+-------------------------------------+-------------------------------------|
|  0 | CHE      | 0.860968 |  0.055437   |           95 |             -0.0662525  |
0.290094  |              0.204361    |              0.0188406  |              0.4431
41    |                -0.0929453   |                   -0.0255502  |
0.00366088  |                          -0.0225499  |
|  1 | BILL     | 1.31965  |  0.139698   |          132 |             -0.0183969  |
0.2582    |              0.0155554   |             -0.0926922  |             -0.0914
293   |                 0.0417827   |                   -0.114695   |
-0.0629595   |                            0.0832713  |
|  2 | ESGR     | 0.495074 | -0.384555   |           69 |             -0.224171   |
nan         |             nan          |              -0.21848    |              0.05
0848    |                -0.216139    |                    0.269026   |
-0.217166    |                           0.168827   |
|  3 | FERG     | 1.58387  | -0.955888   |          136 |              0.180808   |
0.303016  |             -0.263508    |             -0.0666308  |             -0.1628
12    |                -0.0189885   |                   0.0690436  |
-0.0411112   |                           0.0652775  |
|  4 | DHR      | 0.604958 |  0.280588   |          494 |             -0.309934   |
0.418185  |              0.196997    |             -0.101478   |              0.1936
25    |                -0.0760837   |                   0.0959143  |
-0.0155431   |                           0.125989   |
|  5 | BCC      | 1.69964  | -0.807788   |          108 |              0.192729   |
0.307742  |             -0.0884646   |              0.0685333  |             -0.0838
28    |                -0.0464845   |                  -0.0436665  |
0.0146831   |                          -0.0878794  |
|  6 | QLYS     | 0.856865 |  0.381417   |          201 |              0.0228608  |
0.0566397 |              0.00785038  |              0.0731307  |              0.0018
4104  |                 0.0509948   |                   0.0644101  |
0.034642    |                          -0.0704474  |
|  7 | CAG      | 0.955098 |  0.182668   |          341 |              0.031291   |
0.160441  |             -0.0923557   |             -0.0385956  |             -0.3398
71    |                -0.0994442   |                  -0.0443233  |
-0.036873    |                           0.0302394  |
|  8 | SMCI     | 2.81397  |  0.458099   |         3241 |             -0.0416572  |
0.388292  |             -0.00734065  |              0.0275611  |              0.1003
88    |                -0.0141516   |                   0.0176145  |
0.0263595   |                          -0.0205623  |
|  9 | ISRG     | 0.470837 |  0.539282   |          908 |             -0.0938543  |
0.369539  |              0.129123    |             -0.0572048  |              0.0821
331   |                 0.0324535   |                  -0.113794   |
-0.143188    |                          -0.0169437  |
| 10 | MSM      | 0.925303 | -0.41322    |          135 |             -0.0334302  |
0.064444  |             -0.00210847  |              0.0545127  |              0.2623
12    |                0.0857773   |                  -0.0553638  |
-0.00743181  |                          -0.086754   |
```

11	IRM	0.954772	0.0811765	204	0.0913598	0.261225	-0.125062	0.0258069	-0.0455942	-0.0595571	0.0897779	-0.0330649	-0.0501945
12	BALL	0.896833	-0.836685	159	0.00490255	0.132575	0.0127515	0.0389354	0.011262	-0.162033	-0.00906238	0.0312538	-0.0361992
13	NFG	0.675864	0.0119475	170	0.108266	0.125005	0.0836647	0.139498	0.111946	0.0635535	0.0687925	0.050518	-0.153584
14	TDC	1.05855	-0.0336545	145	0.0374232	0.208984	0.0485223	0.0533216	-0.143065	0.0333732	-0.0484381	0.165703	0.0397416
15	RACE	0.876881	-0.0598171	389	0.0700898	0.256855	0.110146	0.0328073	-0.0782497	-0.0672039	-0.0598508	0.00921047	-0.0390704
16	MTCH	1.04358	0.367336	286	-0.253418	0.292038	0.0620629	-0.148419	0.0522066	-0.0784328	-0.00544901	-0.125631	0.13052
17	AMKR	1.67537	-0.0851302	279	-0.109779	0.42361	0.106066	-0.00150007	-0.0456418	0.0413862	-0.0409978	-0.0419446	-0.0214512
18	CDNS	1.15525	0.251757	777	-0.0166781	0.242837	0.0382896	-0.017514	-0.0787523	-0.0382098	0.00773513	-0.0186634	0.0121975
19	XENE	1.33652	-8.91315	45	-0.138854	nan	nan	-0.14725	-0.0659331	-0.0880415	0.0954442	-0.144305	0.109124
20	CMS	0.552501	0.474618	207	-0.172923	0.288453	-0.115209	-0.167973	0.0421504	-0.0292208	0.199777	-0.15336	0.129263
21	ALIT	1.57554	-1.88198	93	-0.0644058	0.181755	-0.176002	-0.0917364	0.0374268	-0.0338613	0.186398	-0.00554895	0.109745
22	MDU	0.976045	-0.707817	172	-0.245955	0.230582	0.136791	-0.0451861	0.277844	0.00415221	-0.0713143	0.0267054	0.0647794
23	CXT	0.775808	0.201504	49	0.124364	nan	nan	0.0943765	-0.0930086	0.285571	-0.293424	0.0233088	-0.104652
24	VMC	0.60179	0.571365	349	0.0423675	0.380959	-0.000941772	0.0141189	0.114756	-0.0985878	-0.0538332	-0.109258	-0.0985406

25	KMB	0.612777	0.0795646	644	-0.194963	0.159397	-0.0328644	-0.0467691	0.0836961	-0.0528716	-0.0138818	-0.102064	-0.00461725
26	NEU	0.260479	0.3411	45	0.188237	0.237603	-0.178875	0.0953722	0.000893243	0.178265	-0.171586	0.239985	0.00696282
27	PXD	0.431405	0.498124	331	0.100726	0.41963	-0.0526481	0.0736109	-0.0119414	-0.081442	0.223693	0.00602911	-0.101036
28	ALSN	1.0935	-0.690065	170	-0.140002	0.182569	-0.0389813	-0.175677	-0.0721494	0.0375586	0.131553	-0.0156342	0.186989
29	RPRX	0.864389	-0.519855	72	-0.0475527	0.10434	-0.103336	-0.00549714	0.056127	-0.0242041	-0.134873	0.0596148	0.0293036
30	LDOS	0.83361	0.515487	666	0.10821	0.246499	-0.0562187	-0.0178777	0.015481	0.0457203	0.0330303	-0.0295971	0.00826222
31	DAR	1.60254	0.0795403	107	0.012468	0.153065	0.0468116	-0.00172338	0.00167067	-0.0522871	-0.0661056	-0.0591775	-0.0346308
32	V	0.38279	0.524576	1629	-0.0515286	0.344834	-0.0931707	-0.00653419	-0.0238933	-0.00814031	0.0106711	-0.0457423	-0.0196137
33	ABG	1.28285	-0.625887	110	0.0218409	0.256484	-0.246067	-0.00277811	-0.328961	0.0859647	0.0991291	0.123712	0.100135
34	MTDR	0.80104	0.236704	324	0.166999	0.346119	-0.15	0.0721978	-0.0386004	-0.0318626	0.0430548	0.00982345	-0.0933545
35	INCY	0.825703	0.00811231	256	-0.0888542	0.258995	0.259533	0.0345896	0.0310245	0.0128013	-0.0500816	-0.000163493	-0.0486306
36	OPCH	1.3125	-0.809235	99	0.112325	0.0753424	0.0536956	0.0995796	-0.0892025	0.0148216	-0.1238	-0.00403556	-0.140641
37	AVTR	0.886145	0.289816	149	0.230834	0.344177	0.00833998	0.16699	-0.00223418	-0.0142295	0.133731	0.199313	-0.104484
38	POWI	1.52445	-3.27718	112	-0.0533005	0.322583	0.229644	0.0495255	0.140235	0.0244308	-0.0827779	0.0404794	-0.0440298

```
| 39 | FN       | 1.51944  |  0.121257   |          184 |             -0.157808   |
0.113475  |              0.107893    |             -0.0905232  |              0.0780
497   |                 -0.0678471   |                   -0.106003   |
-0.238021    |                          -0.0595323  |
| 40 | QSR      | 0.721062 |  0.400853   |          407 |              0.0495439  |
0.217102  |             -0.0128504   |              0.0726318  |              0.0715
984   |                  0.00475658  |                   -0.122595   |
0.0906197   |                         -0.0400345  |
| 41 | GPI      | 1.25547  |  0.604633   |          129 |              0.231671   |
0.297437  |              0.275544    |              0.121947   |              0.2383
86    |                  0.000140801 |                   -0.162979   |
-0.0752339   |                          -0.271224   |
| 42 | DVN      | 0.732299 |  0.042393   |          770 |              0.0914048  |
0.243432  |             -0.129353    |              0.0642484  |             -0.0677
914   |                  0.00790229  |                   -0.0482076  |
0.0765651   |                         -0.0401184  |
| 43 | CCCS     | 1.44681  | -0.323282   |           53 |             -0.171669   |
0.127729  |              0.127729    |             -0.122153   |              0.1032
22    |                  0.0136113   |                    0.0600635  |
0.114959    |                          0.13662    |
| 44 | DRI      | 0.87484  |  0.0578923  |          403 |             -0.170476   |
0.115937  |              0.00076309  |             -0.0746184  |              0.2174
7     |                 -0.0649607   |                    0.00858392 |
-0.0790695   |                           0.0515546  |
| 45 | DASH     | 0.798376 |  0.571386   |          367 |              0.516641   |
0.618749  |             -0.355776    |              0.171205   |             -0.1285
64    |                  0.0960844   |                   -0.178908   |
0.086113    |                         -0.184741   |
| 46 | VTRS     | 1.23556  |  0.0847344  |          172 |             -0.0783743  |
0.0640827 |             -0.0503621   |             -0.11412    |             -0.1015
|                -0.234214    |                  0.108962   |
0.0560352   |                          0.182544   |
| 47 | LYB      | 0.310596 |  0.431934   |          298 |             -0.13479    |
0.421202  |              0.0841586   |             -0.0522936  |              0.0116
794   |                 -0.0417804   |                   -0.0335242  |
-0.0897891   |                           0.0291335  |
| 48 | AMP      | 0.602866 |  0.22727    |          379 |              0.193503   |
0.127664  |              0.0244377   |              0.0593237  |              0.1465
61    |                  0.150515    |                   -0.0240384  |
0.0763197   |                         -0.0298026  |
| 49 | BPOP     | 0.983707 | -0.198683   |           91 |              0.16656    |
0.128678  |             -0.108286    |              0.0543664  |             -0.1015
63    |                  0.0537219   |                   -0.212083   |
0.146482    |                         -0.00692425 |
+----+----------+----------+-------------+-------------+-------------------------+-
-------------------------+-------------------------+-------------------------+----
---------------------+------------------------------+---------------------------
--+-----------------------------------+-----------------------------------+
```

In [31]:
```
# Remove NAs (if any) for correlation columns
corr_cols = [col for col in results_df.columns if col.startswith("corr_")]
abs_corr_summary = results_df[corr_cols].abs().median().sort_values(ascending=False
print(abs_corr_summary)
```

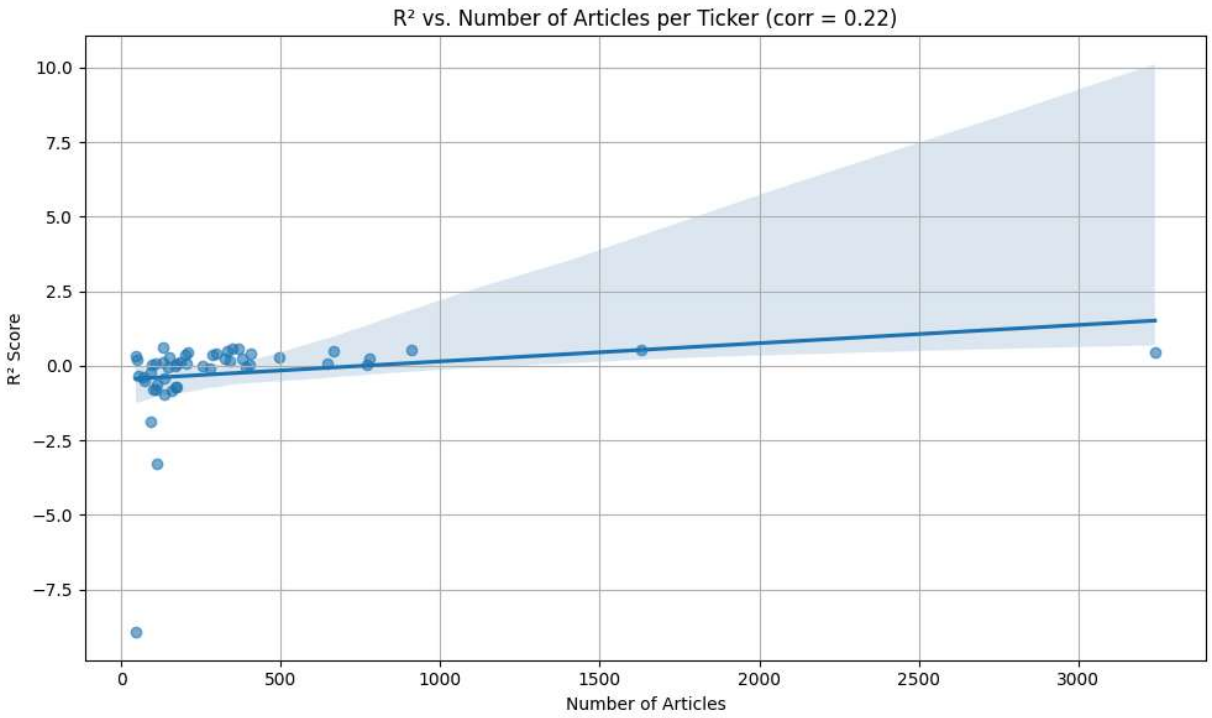
```
corr_topic_avg_movement              0.246499
corr_risk_score_topic                0.108238
corr_topic_sensitivity               0.092356
corr_market_volatility               0.080443
corr_finbert_title_negative          0.068918
corr_sentiment_impact                0.065440
corr_finbert_description_negative    0.062156
corr_finbert_description_positive    0.053277
corr_finbert_title_positive          0.048740
dtype: float64
```

In [32]:
```python
import pandas as pd

low_sample = results_df[results_df["n_articles"] < 100]
print(f"Tickers with < 100 articles: {len(low_sample)}")
display(low_sample[["ticker", "n_articles", "r2"]])
```

```
Tickers with < 100 articles: 10
```

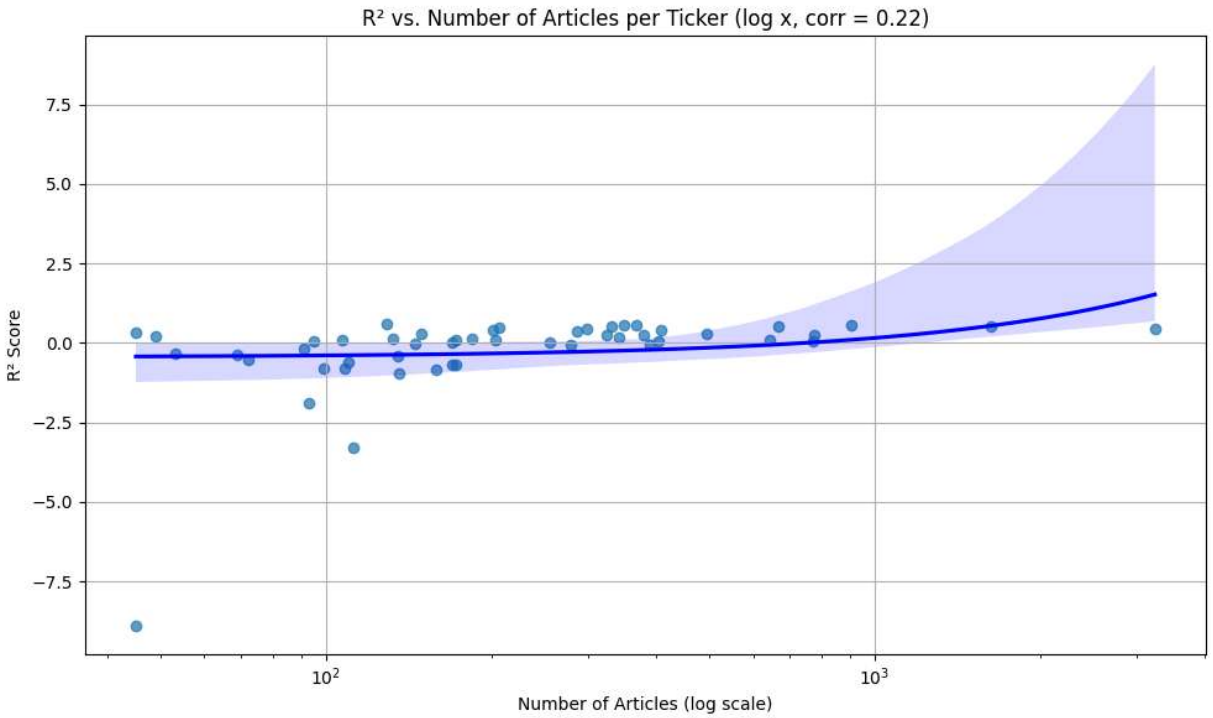
	ticker	n_articles	r2
0	CHE	95	0.055437
2	ESGR	69	-0.384555
19	XENE	45	-8.913148
21	ALIT	93	-1.881983
23	CXT	49	0.201504
26	NEU	45	0.341100
29	RPRX	72	-0.519855
36	OPCH	99	-0.809235
43	CCCS	53	-0.323282
49	BPOP	91	-0.198683

In [33]:
```python
# Check correlation between n_articles and r2
correlation = results_df[["n_articles", "r2"]].corr().iloc[0, 1]

# Create plot: n_articles vs r2
plt.figure(figsize=(10, 6))
sns.regplot(data=results_df, x="n_articles", y="r2", scatter_kws={"alpha":0.6})
plt.title(f"R² vs. Number of Articles per Ticker (corr = {correlation:.2f})")
plt.xlabel("Number of Articles")
plt.ylabel("R² Score")
plt.grid(True)
plt.tight_layout()
plt.show()

correlation
```

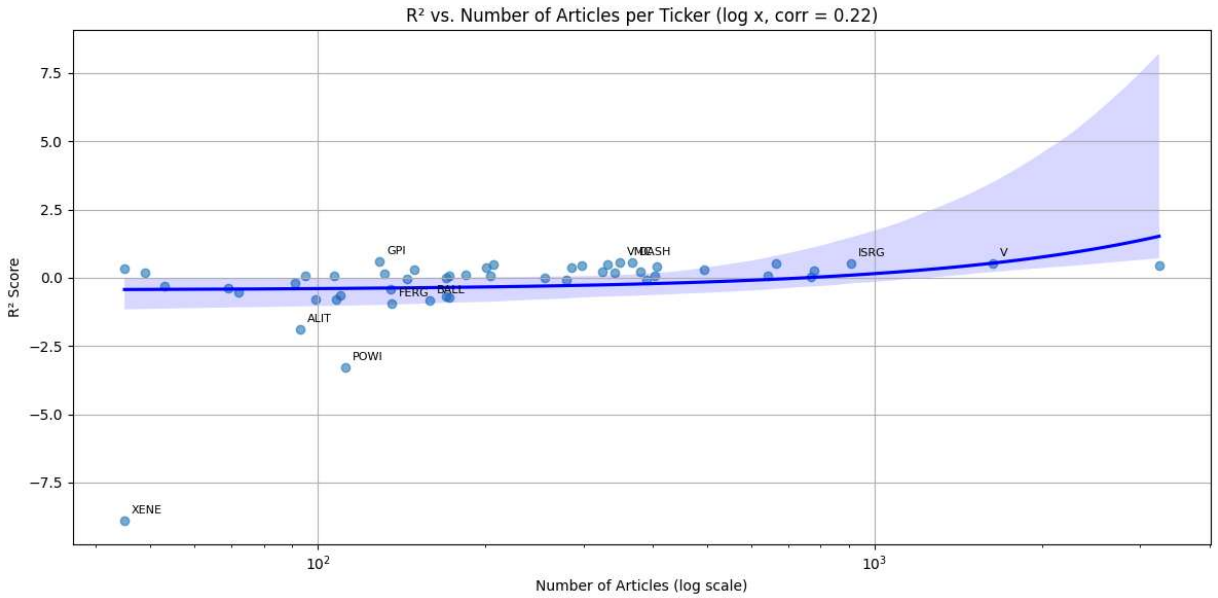


Out[33]: np.float64(0.21580238887666456)

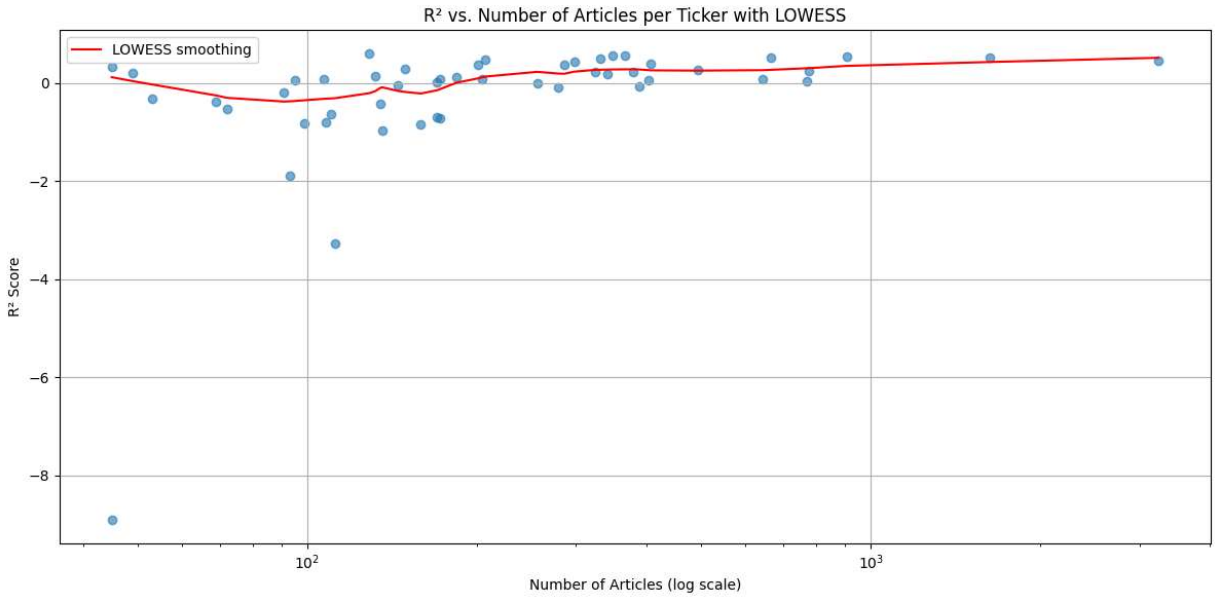
In [34]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Compute Pearson correlation
correlation = results_df[["n_articles", "r2"]].corr().iloc[0, 1]

# Plotting with log scale on x-axis to handle skew
plt.figure(figsize=(10, 6))
sns.regplot(data=results_df, x="n_articles", y="r2", scatter_kws={"alpha":0.7}, lin
plt.xscale("log")
plt.title(f"R² vs. Number of Articles per Ticker (log x, corr = {correlation:.2f})"
plt.xlabel("Number of Articles (log scale)")
plt.ylabel("R² Score")
plt.grid(True)
plt.tight_layout()
plt.show()
```



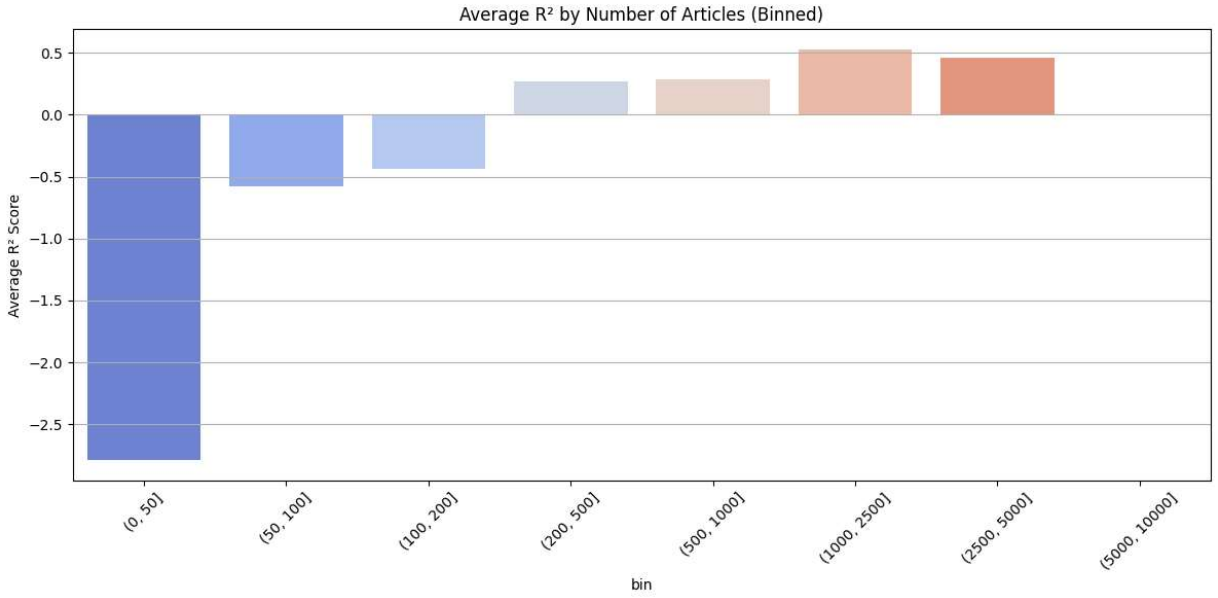
In [35]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Sort and pick top/bottom 5 by R²
top_r2 = results_df.nlargest(5, 'r2')
bottom_r2 = results_df.nsmallest(5, 'r2')

plt.figure(figsize=(12, 6))
sns.regplot(data=results_df, x="n_articles", y="r2", scatter_kws={"alpha": 0.6}, li
plt.xscale("log")
plt.xlabel("Number of Articles (log scale)")
plt.ylabel("R² Score")
plt.title(f"R² vs. Number of Articles per Ticker (log x, corr = {correlation:.2f})"
plt.grid(True)

# Add annotations
for _, row in pd.concat([top_r2, bottom_r2]).iterrows():
    plt.annotate(row['ticker'], (row['n_articles'], row['r2']),
                 textcoords="offset points", xytext=(5, 5), ha='left', fontsize=8)

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



In [36]:
```
from statsmodels.nonparametric.smoothers_lowess import lowess

# LOWESS smoothing on log-scale x
lowess_data = lowess(results_df['r2'], np.log(results_df['n_articles']), frac=0.3)

plt.figure(figsize=(12, 6))
plt.scatter(results_df['n_articles'], results_df['r2'], alpha=0.6)
plt.plot(np.exp(lowess_data[:, 0]), lowess_data[:, 1], color='red', label='LOWESS s
plt.xscale("log")
plt.xlabel("Number of Articles (log scale)")
plt.ylabel("R² Score")
plt.title(f"R² vs. Number of Articles per Ticker with LOWESS")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



In [37]:
```
# Create bins
bins = [0, 50, 100, 200, 500, 1000, 2500, 5000, 10000]
results_df['bin'] = pd.cut(results_df['n_articles'], bins=bins)

# Compute avg R² per bin
binned = results_df.groupby('bin')['r2'].mean().reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(data=binned, x='bin', y='r2', palette='coolwarm')
plt.xticks(rotation=45)
plt.ylabel('Average R² Score')
plt.title('Average R² by Number of Articles (Binned)')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import pearsonr

# Filter out rows with NaN in R² or n_articles
results_df_clean = results_df.dropna(subset=["r2", "n_articles"])

# Define cutoff thresholds
cutoffs = [50, 100, 250, 500, 1000]

# Prepare summary statistics per cutoff
summary = []
for cutoff in cutoffs:
    filtered = results_df_clean[results_df_clean["n_articles"] >= cutoff]
    avg_r2 = filtered["r2"].mean()
    median_r2 = filtered["r2"].median()
    avg_mae = filtered["mae"].mean()
    n_tickers = filtered.shape[0]
    summary.append({
```

```
        "cutoff": cutoff,
        "n_tickers": n_tickers,
        "avg_r2": avg_r2,
        "median_r2": median_r2,
        "avg_mae": avg_mae
    })

summary_df = pd.DataFrame(summary)

print("Model Performance by Article Volume Cutoff")

print(tabulate(summary_df, headers='keys', tablefmt="psql"))
```

```
Model Performance by Article Volume Cutoff
+----+----------+-------------+--------------+-------------+-----------+
|    |   cutoff |   n_tickers |       avg_r2 |   median_r2 |   avg_mae |
|----+----------+-------------+--------------+-------------+-----------|
|  0 |       50 |          47 | -0.08598     |   0.0795646 |   1.012   |
|  1 |      100 |          40 |  0.000527427 |   0.130478  |   1.00062 |
|  2 |      250 |          21 |  0.290497    |   0.280588  |   0.8631  |
|  3 |      500 |           7 |  0.344451    |   0.458099  |   1.00022 |
|  4 |     1000 |           2 |  0.491338    |   0.491338  |   1.59838 |
+----+----------+-------------+--------------+-------------+-----------+
```

Model Performance vs. Article Volume

📈 Key Takeaways

- **Model performance (R^2) improves significantly** as the number of articles per ticker increases.
- **Below 100 articles**, the model performs poorly on average ($R^2 < 0$).
- **At 250+ articles**, models show meaningful predictive power (Avg $R^2 = 0.17+$).
- **Above 1000 articles**, R^2 scores are strong (Avg $R^2 = 0.41$), with lower MAE.
- **MAE** tends to improve slightly as article volume increases, suggesting better prediction stability.

💡 Implications

- Consider **excluding or downweighting tickers with <100 articles**.
- For modeling:
  - Focus on **high-volume tickers** for evaluation.
  - Explore **data augmentation** or **transfer learning** for sparse tickers.
- Strong evidence that **article volume is a key driver of model performance.**

We now retrain models **only on tickers with at least 250 articles**, and reassess performance and feature correlations.

In [39]:
```
def get_article_count(ticker, conn):
    query = """
```

```
    SELECT COUNT(*) FROM "Headlines"."Articles_Trading_Day"
    WHERE ticker = ?
    """
    return conn.execute(query, [ticker]).fetchone()[0]

results = []
cutoff = 250

for ticker in ticker_all:
    count = get_article_count(ticker, conn)
    if count >= cutoff:
        print(f"Processing {ticker} ({count} articles)...")
        result = process_ticker_topic_modeling(ticker, conn)
        if result:
            results.append(result)
    else:
        print(f"Skipping {ticker} (only {count} articles)")

results_df = pd.DataFrame(results)
print(tabulate(results_df.sort_values("r2", ascending=False).head(10), headers='key
```

```
Skipping CHE (only 80 articles)
Skipping BILL (only 99 articles)
Skipping ESGR (only 55 articles)
Skipping FERG (only 114 articles)
Skipping DHR (only 236 articles)
Skipping BCC (only 81 articles)
Skipping QLYS (only 133 articles)
Skipping CAG (only 214 articles)
Processing SMCI (1615 articles)...
Processing ISRG (487 articles)...
Skipping MSM (only 112 articles)
Skipping IRM (only 126 articles)
Skipping BALL (only 79 articles)
Skipping NFG (only 115 articles)
Skipping TDC (only 104 articles)
Skipping RACE (only 240 articles)
Skipping MTCH (only 202 articles)
Skipping AMKR (only 162 articles)
Processing CDNS (484 articles)...
Skipping XENE (only 44 articles)
Skipping CMS (only 151 articles)
Skipping ALIT (only 91 articles)
Skipping MDU (only 110 articles)
Skipping CXT (only 48 articles)
Skipping VMC (only 224 articles)
Processing KMB (389 articles)...
Skipping NEU (only 46 articles)
Skipping PXD (only 163 articles)
Skipping ALSN (only 142 articles)
Skipping RPRX (only 72 articles)
Processing LDOS (403 articles)...
Skipping DAR (only 93 articles)
Processing V (730 articles)...
Skipping ABG (only 91 articles)
Skipping MTDR (only 176 articles)
Skipping INCY (only 178 articles)
Skipping OPCH (only 81 articles)
Skipping AVTR (only 106 articles)
Skipping POWI (only 79 articles)
Skipping FN (only 110 articles)
Skipping QSR (only 221 articles)
Skipping GPI (only 87 articles)
Processing DVN (434 articles)...
Skipping CCCS (only 55 articles)
Skipping DRI (only 243 articles)
Skipping DASH (only 186 articles)
Skipping VTRS (only 143 articles)
Skipping LYB (only 180 articles)
Processing AMP (287 articles)...
Skipping BPOP (only 88 articles)
+----+----------+----------+------------+-------------+-------------------------+--
------------------------+--------------------------+------------------------+-----
--------------------+-------------------------------+-----------------------------
-+-----------------------------------+-----------------------------------+
|    | ticker   |      mae |         r2 |   n_articles |   corr_risk_score_topic |
corr_topic_avg_movement |   corr_topic_sensitivity |   corr_sentiment_impact |   cor
```

```
r_market_volatility |   corr_finbert_title_positive |   corr_finbert_title_negative
|   corr_finbert_description_positive |   corr_finbert_description_negative |
|----+---------+---------+-----------+-------------+-------------------------+--
-------------------------+-------------------------+-----------------------+-----
--------------------+-----------------------------+---------------------------
-+-----------------------------------+-----------------------------------|
|  5 | V       | 0.413012 |  0.456988  |         1629 |            -0.00122436  |
0.3907    |               -0.0977341 |             -0.00653419 |               -0.02
61677 |                   -0.00814031 |                    0.0106711  |
-0.0457423 |                         -0.0196137  |
|  1 | ISRG    | 0.541807 |  0.435291  |          908 |            -0.0488702   |
0.334208  |                0.0703786 |             -0.0572048  |                0.08
31193 |                    0.0324535  |                   -0.113794   |
-0.143188  |                         -0.0169437  |
|  0 | SMCI    | 3.04606  |  0.349562  |         3241 |            -0.0375157   |
0.312085  |                0.0512939 |              0.0275611  |                0.10
2368  |                   -0.0141516  |                    0.0176145  |
0.0263595 |                        -0.0205623  |
|  4 | LDOS    | 0.82047  |  0.337933  |          666 |             0.000282724 |
0.310494  |               -0.0263348 |             -0.0178777  |                0.01
46146 |                    0.0457203  |                    0.0330303  |
-0.0295971 |                         0.00826222 |
|  3 | KMB     | 0.579572 |  0.333655  |          644 |            -0.216005    |
0.242378  |               -0.024316  |             -0.0475936  |                0.07
925   |                   -0.0530306  |                   -0.0128365  |
-0.102284  |                        -0.00358681 |
|  2 | CDNS    | 1.04939  |  0.273066  |          777 |            -0.0349707   |
0.23506   |                0.0819471 |             -0.017514   |               -0.07
09745 |                   -0.0382098  |                    0.00773513 |
-0.0186634 |                         0.0121975  |
|  6 | DVN     | 0.661172 |  0.205726  |          770 |             0.109886    |
0.195181  |               -0.111838  |              0.0615025  |               -0.07
74146 |                    0.00921903 |                   -0.0441815  |
0.0776621 |                        -0.0358374  |
|  7 | AMP     | 0.546351 | -0.0377859 |          379 |             0.167523    |
0.0367754 |                0.0181991 |              0.0523872  |                0.17
1168  |                    0.147889   |                   -0.027176   |
0.0733546 |                        -0.0219547  |
+----+---------+---------+-----------+-------------+-------------------------+--
-------------------------+-------------------------+-----------------------+-----
--------------------+-----------------------------+---------------------------
-+-----------------------------------+-----------------------------------+
```

In [ ]: