In the previous notebook (https://www.kaggle.com/code/hrishikguha/wefarm-embedding-similarity-scoring), I imported about 2000000 rows of the wefarm dataset, for initial analysis, and computed embeddings and similarity scores with target financial phrases. In this notebook, I import the scored parquet file generated in the previous notebook, see what portion of the questions are financial, and finally, apply clustering techniques to group the financial questions thematically.

```
import polars as pl
import os

# ---  CONFIGURATION AND LOADING ---
FINAL_SCORED_PATH = "all_10_parts_scored.parquet"
TEXT_COL = "question_content" # Column name for the question text

print(f"Attempting to load data from: {FINAL_SCORED_PATH}")

try:
    # Load the scored data
    scored_df_final = pl.read_parquet(FINAL_SCORED_PATH)
    print(f"✅ Data loaded successfully. Total rows: {scored_df_final.shape[0]}")
except Exception as e:
    print(f"❌ ERROR: Failed to load file. Ensure the file name is exactly 'all_10_parts_scored.parquet'
    print(f"Error details: {e}")
    # Stop execution if data cannot be loaded
    raise

# Filter down to only the questions classified as financial (is_financial == True)
financial_questions_df = scored_df_final.filter(pl.col("is_financial"))

print("-" * 60)

# --- QUANTITATIVE INSIGHTS: VOLUME AND CONFIDENCE ---

total_questions = scored_df_final.shape[0]
financial_count = financial_questions_df.shape[0]
percent_financial = (financial_count / total_questions) * 100 if total_questions > 0 else 0

# Calculate summary statistics for the confidence scores (sim_fin_max)
confidence_summary = financial_questions_df.select(pl.col("sim_fin_max")).describe()


print("🎯 INITIAL INSIGHTS: VOLUME & CONFIDENCE")
print("-" * 60)
print(f"Total Questions Analyzed: {total_questions}")
print(f"Questions Flagged as Financial: {financial_count} ({percent_financial:.2f}%)")
print("\nConfidence Score (sim_fin_max) Summary for Financial Questions:")
print(confidence_summary)
print("=" * 60)


# ---  QUALITATIVE INSIGHTS: TOP 10 THEMES ---

print("\n🔍 QUALITATIVE INSIGHTS: TOP 10 HIGHEST CONFIDENCE QUESTIONS")
print("------------------------------------------------------------")

# Sort by the highest score and select the top 10
top_hits_display = financial_questions_df.sort("sim_fin_max", descending=True).head(10).select([
    pl.col(TEXT_COL),
    pl.col("sim_fin_max").round(4).alias("Sim_Score"),
])

# Use the corrected Polars configuration to ensure full text is printed
with pl.Config(tbl_cols=-1, set_tbl_width_chars=500 ):
    print(top_hits_display)
```

```
Attempting to load data from: all_10_parts_scored.parquet
✅ Data loaded successfully. Total rows: 2000000
------------------------------------------------------------
```

```
🎯 INITIAL INSIGHTS: VOLUME & CONFIDENCE
------------------------------------------------------------
Total Questions Analyzed: 2000000
Questions Flagged as Financial: 1520 (0.08%)

Confidence Score (sim_fin_max) Summary for Financial Questions:
shape: (9, 2)
┌────────────┬─────────────┐
│ statistic  ┆ sim_fin_max │
│ ---        ┆ ---         │
│ str        ┆ f64         │
╞════════════╪═════════════╡
│ count      ┆ 1520.0      │
│ null_count ┆ 0.0         │
│ mean       ┆ 0.636932    │
│ std        ┆ 0.034592    │
│ min        ┆ 0.600021    │
│ 25%        ┆ 0.614206    │
│ 50%        ┆ 0.627178    │
│ 75%        ┆ 0.647536    │
│ max        ┆ 0.862286    │
└────────────┴─────────────┘

============================================================

🔍 QUALITATIVE INSIGHTS: TOP 10 HIGHEST CONFIDENCE QUESTIONS
------------------------------------------------------------
shape: (10, 2)
┌──────────────────────────────┬───────────┐
│ question_content             ┆ Sim_Score │
│ ---                          ┆ ---       │
│ str                          ┆ f64       │
╞══════════════════════════════╪═══════════╡
│ How can i get loans          ┆ 0.8623    │
│ How can I get some loans?    ┆ 0.8209    │
│ How can I get some loans?    ┆ 0.8209    │
│ How can I get some loans?    ┆ 0.8209    │
│ Q Where Can I Get Loans?     ┆ 0.8149    │
│ Q Where Can I Get Loans?     ┆ 0.8149    │
│ Q How can i get access to loan… ┆ 0.803  │
│ Q How can i get access to loan… ┆ 0.803  │
│ Q                            ┆ 0.7984    │
│ How can one get grants from …  ┆           │
│ Q. How Can I Get That Loans? ┆ 0.7953    │
└──────────────────────────────┴───────────┘
```

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import os
import matplotlib.pyplot as plt
# Filter down to financial questions and extract embeddings
financial_df = scored_df_final.filter(pl.col("is_financial"))
if financial_df.shape[0] == 0:
    raise ValueError("No financial questions found for clustering.")

financial_embeddings = np.stack(financial_df["embedding"].to_list())
print(f"Loaded {financial_df.shape[0]} financial questions.")
```

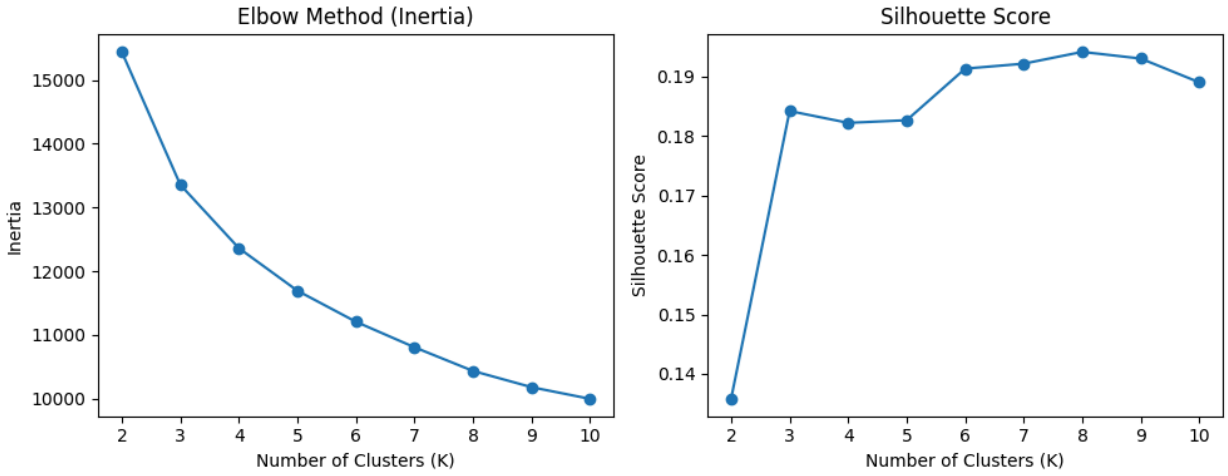
```
Loaded 1520 financial questions.
```

```
# ---  OPTIMAL K DETERMINATION ---

def find_optimal_k(embeddings, k_range=(2, 10)):
    """Calculates Inertia (Elbow) and Silhouette Score for a range of K values."""
    inertias = []
    silhouette_scores = {}

    k_values = range(k_range[0], k_range[1] + 1)

    print("\nCalculating Elbow and Silhouette scores for K in range 2-10...")

    for k in k_values:
        # Run K-Means with the current K
        kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
```

```
        clusters = kmeans.fit_predict(embeddings)

        # 1. Elbow Method Metric (Inertia)
        inertias.append(kmeans.inertia_)

        # 2. Silhouette Score Metric (Measures density/separation)
        # Skip Silhouette for K=1, as it requires at least two clusters
        if k > 1:
            score = silhouette_score(embeddings, clusters)
            silhouette_scores[k] = score
      # Plotting for visual determination
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    plt.plot(k_values, inertias, marker='o')
    plt.title('Elbow Method (Inertia)')
    plt.xlabel('Number of Clusters (K)')
    plt.ylabel('Inertia')

    plt.subplot(1, 2, 2)
    plt.plot(list(silhouette_scores.keys()), list(silhouette_scores.values()), marker='o')
    plt.title('Silhouette Score')
    plt.xlabel('Number of Clusters (K)')
    plt.ylabel('Silhouette Score')

    plt.tight_layout()
    plt.show()

    # Recommend the K with the highest Silhouette Score (most separated/dense clusters)
    if silhouette_scores:
        optimal_k = max(silhouette_scores, key=silhouette_scores.get)
        print(f"\nRecommended K based on highest Silhouette Score: {optimal_k}")
        return optimal_k
```

```
# Execute K determination and use the result
K_CLUSTERS = find_optimal_k(financial_embeddings)
```

```
Calculating Elbow and Silhouette scores for K in range 2-10...
```




```
Recommended K based on highest Silhouette Score: 8
```

```
# ---  FINAL CLUSTERING AND REPRODUCIBILITY FIX ---

print("-" * 60)
#print(f"Running final K-Means with K={K_CLUSTERS}...")

# Run the FINAL model with the determined K
kmeans_final = KMeans(n_clusters=8, random_state=42, n_init=10)
clusters = kmeans_final.fit_predict(financial_embeddings)

# Add the cluster label back to the Polars DataFrame
```

```
financial_df = financial_df.with_columns(
    pl.Series(name="cluster_label", values=clusters)
)

# FIX: Save the Cluster Centers for Reproducibility
CLUSTER_CENTERS_PATH = "cluster_centers_financial.npy"
np.save(CLUSTER_CENTERS_PATH, kmeans_final.cluster_centers_)
print(f"✅ Cluster Centers saved for future prediction/reproducibility: {CLUSTER_CENTERS_PATH}")
print("-" * 60)

# ---  CLUSTER ANALYSIS AND QUALITY EVALUATION ---

print("🎯 CLUSTER ANALYSIS: THEME DISCOVERY AND EVALUATION")
print("-" * 60)

# Calculate Cluster Quality Metrics
final_silhouette_score = silhouette_score(financial_embeddings, clusters)

print(f"Total Questions Clustered: {financial_df.shape[0]}")
print(f"Final Silhouette Score (Quality Metric): {final_silhouette_score:.4f} (Higher is Better)")

# Group by the new cluster label and count questions in each group
cluster_counts = financial_df.group_by("cluster_label").agg(
    pl.count().alias("Count"),
    (pl.col("sim_fin_max").mean()).round(4).alias("Avg_Sim_Score")
).sort("Count", descending=True)

print("\nCluster Sizes and Average Confidence:")
print(cluster_counts)
```

```
------------------------------------------------------------
✅ Cluster Centers saved for future prediction/reproducibility: cluster_centers_financial.npy
------------------------------------------------------------
🎯 CLUSTER ANALYSIS: THEME DISCOVERY AND EVALUATION
------------------------------------------------------------
Total Questions Clustered: 1520
Final Silhouette Score (Quality Metric): 0.1942 (Higher is Better)

Cluster Sizes and Average Confidence:
shape: (8, 3)
┌───────────────┬───────┬───────────────┐
│ cluster_label ┆ Count ┆ Avg_Sim_Score │
│ ---           ┆ ---   ┆ ---           │
│ i32           ┆ u32   ┆ f64           │
╞═══════════════╪═══════╪═══════════════╡
│ 5             ┆ 277   ┆ 0.6361        │
│ 2             ┆ 259   ┆ 0.6167        │
│ 1             ┆ 225   ┆ 0.633         │
│ 3             ┆ 216   ┆ 0.6588        │
│ 7             ┆ 213   ┆ 0.6306        │
│ 4             ┆ 159   ┆ 0.6396        │
│ 6             ┆ 107   ┆ 0.6603        │
│ 0             ┆ 64    ┆ 0.6378        │
└───────────────┴───────┴───────────────┘
/tmp/ipython-input-4294223265.py:34: DeprecationWarning: `pl.count()` is deprecated. Please use `pl.len(
  pl.count().alias("Count"),
```

The translation is not always great but I decided to list the full text of 10 questions from each cluster to understand the exact themes. This gives us a basic idea. The cluster themes go as follows: "Saving", "Loans", "Cash Crops", "Government assistance, grants and subisidies", "Earnings from Farm", "Managing Farm Finance", "Farm Inputs", and generic questions about which farming and poultry uses are the best. Going through example question text(listed below) for each cluster might be useful. I have not listed cluster no.1 due to poor readability and generic nature.

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 0
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
```

```
cluster_0_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_0_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🏅 Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🏅 Top 10 Questions for Cluster 0 ---
Total questions in Cluster 0: 225
----------------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.7157
| Question: I WOULD LIKE TO START SAVING MONEY, WHAT CAN I DO?
========================================
| RANK: 2
| Score: 0.7157
| Question: I WOULD LIKE TO START SAVING MONEY, WHAT CAN I DO?
========================================
| RANK: 3
| Score: 0.7157
| Question: I WOULD LIKE TO START SAVING MONEY, WHAT CAN I DO?
========================================
| RANK: 4
| Score: 0.7117
| Question: Q hello How Can I Save My Money
========================================
| RANK: 5
| Score: 0.7117
| Question: Q hello How Can I Save My Money
========================================
| RANK: 6
| Score: 0.7117
| Question: Q hello How Can I Save My Money
========================================
| RANK: 7
| Score: 0.6643
| Question: Q I WANT TO KEEP HENS INEED HELP IN SOME MONEY
========================================
| RANK: 8
| Score: 0.6643
| Question: Q I WANT TO KEEP HENS INEED HELP IN SOME MONEY
========================================
| RANK: 9
| Score: 0.6640
| Question: Q why do i need to save money?
========================================
| RANK: 10
| Score: 0.6522
| Question: Which is the best method of saving
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 2
N_DISPLAY = 10
```

```
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_2_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_2_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🏅 Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🏅 Top 10 Questions for Cluster 2 ---
Total questions in Cluster 2: 259
----------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.6404
| Question: Q I want to start Potatoe farming what is the best type for cash crop
========================================
| RANK: 2
| Score: 0.6404
| Question: Q I want to start Potatoe farming what is the best type for cash crop
========================================
| RANK: 3
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 4
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 5
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 6
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 7
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 8
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 9
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
| RANK: 10
| Score: 0.6337
| Question: What are best cash crops that a farmer can plant to earn a living faster?
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 3
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_3_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_3_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🏅 Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🏅 Top 10 Questions for Cluster 3 ---
Total questions in Cluster 3: 259
----------------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.7903
| Question: Q#Which bank can access farmers loans?
========================================
| RANK: 2
| Score: 0.7815
| Question: Q, which bank offers loans to small scale farmers?
========================================
| RANK: 3
| Score: 0.7422
| Question: Q How Can I Get Loans For Farming,
========================================
| RANK: 4
| Score: 0.7422
| Question: Q How Can I Get Loans For Farming,
========================================
| RANK: 5
| Score: 0.7422
| Question: Q How Can I Get Loans For Farming,
========================================
| RANK: 6
| Score: 0.7291
| Question: Q Where can i get loans to boost my farming
========================================
| RANK: 7
| Score: 0.7291
| Question: Q Where can i get loans to boost my farming
========================================
| RANK: 8
| Score: 0.7268
| Question: Q.which bank gives farmers loans at low interest
========================================
| RANK: 9
| Score: 0.7268
| Question: Q.which bank gives farmers loans at low interest
========================================
| RANK: 10
| Score: 0.7268
| Question: Q.which bank gives farmers loans at low interest
```

```
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 4
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_4_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_4_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🎖️ Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🎖️ Top 10 Questions for Cluster 4 ---
Total questions in Cluster 4: 259
----------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.7984
| Question: Q
How can one get grants from the government to do farming
========================================
| RANK: 2
| Score: 0.7441
| Question: Do You Offer Any Assistance In Agriculture
========================================
| RANK: 3
| Score: 0.7441
| Question: Do You Offer Any Assistance In Agriculture
========================================
| RANK: 4
| Score: 0.7441
| Question: Do You Offer Any Assistance In Agriculture
========================================
| RANK: 5
| Score: 0.7329
| Question: Q, how can i get subsidies fertilizer offered by the ministry of agriculture
========================================
| RANK: 6
| Score: 0.7175
| Question: How can we farm support farmers
========================================
| RANK: 7
| Score: 0.7175
| Question: How can we farm support farmers
========================================
| RANK: 8
| Score: 0.7167
| Question: Q Where(Internationally) can one get agricultural aid in form of funds?
========================================
| RANK: 9
| Score: 0.7075
| Question: What are the conditions for a farmer to be funded?
```

```
========================================
| RANK: 10
| Score: 0.7075
| Question: What are the conditions for a farmer to be funded?
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 5
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_5_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_5_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🎖️ Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🎖️ Top 10 Questions for Cluster 5 ---
Total questions in Cluster 5: 259
----------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.7334
| Question: Q how can i get money to earn in my farm?
========================================
| RANK: 2
| Score: 0.7334
| Question: Q how can i get money to earn in my farm?
========================================
| RANK: 3
| Score: 0.7242
| Question: QHOW CAN I ACQUIRE SOME MONEY FOR FARM INPUTS
========================================
| RANK: 4
| Score: 0.7242
| Question: QHOW CAN I ACQUIRE SOME MONEY FOR FARM INPUTS
========================================
| RANK: 5
| Score: 0.7242
| Question: QHOW CAN I ACQUIRE SOME MONEY FOR FARM INPUTS
========================================
| RANK: 6
| Score: 0.7242
| Question: QHOW CAN I ACQUIRE SOME MONEY FOR FARM INPUTS
========================================
| RANK: 7
| Score: 0.7213
| Question: Q
How can one manage farm finance
========================================
| RANK: 8
| Score: 0.7213
| Question: Q
```

```
How can one manage farm finance
========================================
| RANK: 9
| Score: 0.7213
| Question: Q
How can one manage farm finance
========================================
| RANK: 10
| Score: 0.7210
| Question: Q HOW CAN I GET MONEY THROUGH FARMING
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 6
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_6_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_6_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🎖️ Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🎖️ Top 10 Questions for Cluster 6 ---
Total questions in Cluster 6: 259
----------------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.8623
| Question: How can i get loans
========================================
| RANK: 2
| Score: 0.8209
| Question: How can I get some loans?
========================================
| RANK: 3
| Score: 0.8209
| Question: How can I get some loans?
========================================
| RANK: 4
| Score: 0.8209
| Question: How can I get some loans?
========================================
| RANK: 5
| Score: 0.8149
| Question: Q Where Can I Get Loans?
========================================
| RANK: 6
| Score: 0.8149
| Question: Q Where Can I Get Loans?
========================================
| RANK: 7
| Score: 0.8030
```

```
| Question: Q How can i get access to loans?
========================================
| RANK: 8
| Score: 0.8030
| Question: Q How can i get access to loans?
========================================
| RANK: 9
| Score: 0.7953
| Question: Q. How Can I Get That Loans?
========================================
| RANK: 10
| Score: 0.7536
| Question: can we get loans
========================================
```

```
# --- CONFIGURATION ---
TARGET_CLUSTER = 7
N_DISPLAY = 10
TEXT_COL = "question_content"

# Filter the DataFrame for the target cluster
cluster_7_df = financial_df.filter(pl.col("cluster_label") == TARGET_CLUSTER)

# Sort by the highest similarity score (confidence) and select the top N_DISPLAY
top_10_questions = cluster_7_df.sort("sim_fin_max", descending=True).head(N_DISPLAY)

# Extract the relevant data into a list of Python dictionaries
questions_data = top_10_questions.select([TEXT_COL, "sim_fin_max", "cluster_label"]).to_dicts()

# --- MANUAL PRINTING FOR FULL TEXT ---

print(f"\n--- 🏅 Top {N_DISPLAY} Questions for Cluster {TARGET_CLUSTER} ---")
print(f"Total questions in Cluster {TARGET_CLUSTER}: {cluster_2_df.shape[0]}")
print("----------------------------------------------------------------------")

# Manually print each question, ensuring full text is displayed
for i, row in enumerate(questions_data):
    # Print a separator for readability
    print("=" * 40)
    print(f"| RANK: {i+1}")
    print(f"| Score: {row['sim_fin_max']:.4f}")
    # Print the full question text
    print(f"| Question: {row[TEXT_COL]}")

print("=" * 40)
```

```
--- 🏅 Top 10 Questions for Cluster 7 ---
Total questions in Cluster 7: 259
----------------------------------------------------------------------
========================================
| RANK: 1
| Score: 0.7187
| Question: Q CAN YOU CREDIT FARM IMPUTS
========================================
| RANK: 2
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 3
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 4
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 5
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 6
| Score: 0.6996
| Question: How can I apply for farm imputs?
```

```
========================================
| RANK: 7
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 8
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 9
| Score: 0.6996
| Question: How can I apply for farm imputs?
========================================
| RANK: 10
| Score: 0.6923
| Question: How Can I Benefit In We Farm
========================================
```

```
Start coding or generate with AI.
```