```
from faker import Faker

# Initialize Faker
fake = Faker()

# Generate synthetic user data
user_data = []
for _ in range(num_users):
    user_data.append({
        'user_id': fake.uuid4(),
        'username': fake.user_name(),
        'age': fake.random_int(18, 70),
        'gender': fake.random_element(['Male', 'Female']),
        'location': fake.country()
    })

# Generate synthetic item data
item_data = []
for _ in range(num_items):
    item_data.append({
        'item_id': fake.uuid4(),
        'title': fake.catch_phrase(),
        'category': fake.random_element(['Electronics', 'Clothing', 'Books', 'Home']),
        'price': fake.random_number(2),
        'popularity': fake.random_int(1, 100)
    })

# Convert user and item data to DataFrames
users_df = pd.DataFrame(user_data)
items_df = pd.DataFrame(item_data)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-a9907a405bd9> in <cell line: 8>()
      6 # Generate synthetic user data
      7 user_data = []
----> 8 for _ in range(num_users):
      9     user_data.append({
     10         'user_id': fake.uuid4(),

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

Next steps: Explain error

```
from faker import Faker
import numpy as np
import pandas as pd

# Initialize Faker to generate synthetic data
fake = Faker()

# Set random seed for reproducibility
np.random.seed(42)

# Define the number of users and items
num_users = 1000
num_items = 500

# Generate synthetic user data
user_data = []
for _ in range(num_users):
    user_data.append({
        'user_id': fake.uuid4(),  # Generate a unique user ID
        'age': np.random.randint(18, 65),  # Generate random age between 18 and 65
        'gender': fake.random_element(['Male', 'Female', 'Other']),  # Generate random gender
        # Add more user attributes as needed
    })

# Generate synthetic item data
item_data = []
for _ in range(num_items):
    item_data.append({
        'item_id': fake.uuid4(),  # Generate a unique item ID
        'category': fake.random_element(['Electronics', 'Clothing', 'Books', 'Home & Kitchen']),  # Generate random
        'price': round(np.random.uniform(10, 200), 2),  # Generate random price between 10 and 200
        # Add more item attributes as needed
```

```
    })

# Convert user and item data to pandas DataFrame
users_df = pd.DataFrame(user_data)
items_df = pd.DataFrame(item_data)

# Display the first few rows of user and item data
print("User Data:")
print(users_df.head())
print("\nItem Data:")
print(items_df.head())
```

```
User Data:
                                user_id  age gender
0  5a785578-52e4-4cb7-9c1e-c6c03a750a78   56  Other
1  d1862dcc-aaec-419d-b96f-16a3a5e4a7bd   46   Male
2  4f143929-06c9-48cd-8d65-a29e8376ab87   32   Male
3  406a607b-7463-4218-84fc-35c41ecaaa97   60  Other
4  95f09a0f-e351-4aed-8b2d-672817120d44   25  Other

Item Data:
                                item_id        category   price
0  5059fe38-da0f-494c-a008-7ef2f301ca31  Home & Kitchen   71.25
1  862fc122-da03-42d5-aea1-51f28e37ea86        Clothing  163.88
2  639836ef-d1bd-4c45-854b-4dd93a13ec55     Electronics   58.38
3  5eeeba6b-0e16-4045-a312-6330f047f974     Electronics  139.49
4  eb28b17c-f386-46ae-ab5f-4ac213979b03  Home & Kitchen  154.44
```

```
pip install faker
```

```
Collecting faker
  Downloading Faker-24.3.0-py3-none-any.whl (1.8 MB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 9.8 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.10/dist-packages (from faker) (2.8
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.4->f
Installing collected packages: faker
Successfully installed faker-24.3.0
```

```
import pandas as pd
from faker import Faker
import random
from datetime import datetime, timedelta

# Initialize Faker to generate synthetic data
fake = Faker()

# Set random seed for reproducibility
random.seed(42)

# Define the number of users, items, and user engagement events
num_users = 1000
num_items = 500
num_events = 1000

# Generate synthetic user data
user_data = []
for _ in range(num_users):
    user_data.append({
        'user_id': fake.uuid4(),  # Generate a unique user ID
        'age': random.randint(18, 65),  # Generate a random age between 18 and 65
        'gender': random.choice(['Male', 'Female', 'Other']),  # Randomly select gender
        # Add more user attributes as needed
    })

# Generate synthetic item data
item_data = []
for _ in range(num_items):
    item_data.append({
        'item_id': fake.uuid4(),  # Generate a unique item ID
        'category': random.choice(['Electronics', 'Clothing', 'Books', 'Home & Kitchen']),  # Randomly select catego
```

```
        'price': round(random.uniform(10, 200), 2),  # Generate a random price between 10 and 200
        # Add more item attributes as needed
    })

# Generate synthetic user engagement data
user_engagement_data = []

for _ in range(num_events):
    user_id = random.choice(user_data)['user_id']  # Randomly select a user ID from user data
    item_id = random.choice(item_data)['item_id']  # Randomly select an item ID from item data
    timestamp = fake.date_time_between(start_date='-30d', end_date='now')  # Generate a random timestamp within the
    event_type = random.choice(['click', 'order'])  # Randomly select the event type (click or order)

    user_engagement_data.append({
        'user_id': user_id,
        'item_id': item_id,
        'timestamp': timestamp,
        'event_type': event_type
    })

# Create DataFrames from the synthetic user, item, and user engagement data
users_df = pd.DataFrame(user_data)
items_df = pd.DataFrame(item_data)
user_engagement_df = pd.DataFrame(user_engagement_data)

# Display the first few rows of the synthetic user, item, and user engagement data
print("Synthetic User Data:")
print(users_df.head())
print("\nSynthetic Item Data:")
print(items_df.head())
print("\nSynthetic User Engagement Data:")
```

```
print(user_engagement_df.head())
```

```
Synthetic User Data:
                                user_id  age gender
0  078a794c-0bf9-48c9-9bca-326a3f126f29   58   Male
1  698073be-883e-4117-bcd5-0f8dc9535cc0   19  Other
2  b2dfd310-e5ac-4308-a66e-b2dff368e2aa   35   Male
3  456471be-4b8b-46fc-bec7-82d35b81bb20   32   Male
4  c1f44db2-055d-45fd-9f82-0625cf3c7655   65   Male

Synthetic Item Data:
                                item_id     category   price
0  d0e8909a-a930-4522-8758-68f9cdad2395     Clothing   40.83
1  8d808eae-086c-400f-bb14-cc5066b97e10     Clothing  147.02
2  c77a9e24-a36b-4c9f-bee5-cad4167fa9b8  Electronics   88.06
3  374fffd5-cb1d-4f15-bcc6-f978473cfc61     Clothing  194.22
4  693c3441-435d-4320-9710-6281f20efdc2        Books  152.95

Synthetic User Engagement Data:
                                user_id                                item_id  \
0  eb6ba928-39d2-4347-9d58-8feac4fe9ee7  f9c35bc1-359e-4b02-9521-944f2487f097
1  922f3b5e-8da1-45d1-b966-8b2937e363ca  99b570f1-2b14-46b8-bb74-82a6aa057655
2  5c3e4f8e-ca73-4bbb-8afb-7d15c9851ee1  27f9475b-1f3e-4b32-8ef1-f4f5c44f1e56
3  5008bc86-05c1-44c0-bfa9-39cb113a0f6d  53932805-ffc3-455c-880a-90882cb03107
4  27153da9-f88a-4ae0-921d-4ab48ffbc802  2dcc9a5e-b360-4fd8-899a-dff8b0e6d168

                   timestamp event_type
0 2024-03-14 12:23:09.607069      order
1 2024-03-16 21:09:57.072796      order
2 2024-03-01 10:25:10.993901      order
3 2024-03-01 04:32:55.130427      click
4 2024-03-12 08:38:03.920221      click
```

```
import numpy as np

# Create a pivot table to represent user-item interactions
interaction_matrix = pd.crosstab(user_engagement_df['user_id'], user_engagement_df['item_id'])

# Calculate the co-visitation matrix
co_visitation_matrix = np.dot(interaction_matrix.T, interaction_matrix)

# Print the co-visitation matrix
print("Co-visitation Matrix:")
print(co_visitation_matrix)
```

```
Co-visitation Matrix:
[[2 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 3 ... 0 0 0]
 ...
 [0 0 0 ... 2 0 0]
 [0 0 0 ... 0 2 0]
 [0 0 0 ... 0 0 1]]
```

```
import numpy as np

# Assuming co_visitation_matrix is your co-visitation matrix
# Threshold for co-visitation count
threshold = 5
N=10
# Filter items based on threshold
candidate_items = set(np.where(co_visitation_matrix >= threshold)[0])

# Rank candidate items based on co-visitation counts
ranked_candidate_items = sorted(candidate_items, key=lambda x: np.sum(co_visitation_matrix[x]), reverse=True)

# Select top-N candidate items
top_n_candidates = ranked_candidate_items[:N]

# Print top-N candidate items
print("Top-N Candidate Items:", top_n_candidates)
```

```
Top-N Candidate Items: [302, 177, 66, 377, 370, 330, 119, 343, 12, 21]
```

```
# Extract unique item IDs and user IDs from user engagement data
unique_item_ids = user_engagement_df['item_id'].unique()
unique_user_ids = user_engagement_df['user_id'].unique()

# Print unique item IDs and user IDs
print("Unique Item IDs:", unique_item_ids)
print("Unique User IDs:", unique_user_ids)
```

```
'ebcd5618-a566-419c-8003-392d2a97552e'
'0338b8d0-f05e-43f6-8983-095197aba26e'
'61f1b68a-cf51-4204-9212-30c52e504287'
'e3d5a296-b12f-48a9-b4c4-a1513b1c81a8'
'fbe335f3-defe-4172-b8f5-4f186a654d8f'
'a3f26784-411c-4ba5-bdff-dd84a3914f50'
'ddae1969-f6d2-4508-92ab-c370c6d5e441'
'f89e295c-5a09-494d-afe5-be32f6cc9f61'
'dcb4da46-d19b-4567-b33f-97ea5b107b05'
'a21a6a9a-678d-40a9-9c43-cf85c0caceb3'
'6bd213e8-fa4b-4b31-b42e-69819fef53fb'
'746f7a0d-00ac-488a-b6b0-5fd2456af590'
'4a41f69e-89f9-4cbf-b879-eb2d74962b7c'
'cc3c27a2-a0a2-47db-a677-e8893bb82f76'
'ac72aad2-9efa-4cb3-ba16-10dcce6b0920'
'8b9a5090-7237-4f2c-afbf-537de636f170'
'10e52bc6-c86f-4bfd-8469-39d54cd40d32'
'30d61a99-a747-4eb2-9e47-4912f8e2b4bb'
'b2822251-2173-45ad-b8b5-222102f67497'
```

```
# Assuming co_visitation_matrix is your co-visitation matrix
# Threshold for co-visitation count
threshold = 10

# Filter items based on threshold
candidate_items = set(np.where(co_visitation_matrix >= threshold)[0])

# Rank candidate items based on co-visitation counts
ranked_candidate_items = sorted(candidate_items, key=lambda x: np.sum(co_visitation_matrix[x]), reverse=True)

# Select top-N candidate items
top_n_candidates = ranked_candidate_items[:5]

# Print top-N candidate items
print("Top-N Candidate Items:", top_n_candidates)
```

```
Top-N Candidate Items: []
```

```
import numpy as np

# Assuming co_visitation_matrix is your co-visitation matrix
# Adjust display options to print the entire matrix
np.set_printoptions(threshold=np.inf)

# Print the co-visitation matrix
print("Co-visitation Matrix:")
print(co_visitation_matrix)
```

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```python
# Assuming item_to_index is a dictionary mapping item IDs to numeric indices
# Retrieve the item IDs corresponding to the numeric identifiers
top_n_items = [item_id for item_id, index in item_to_index.items() if index in [302, 177, 66, 377, 370, 330, 119, 34

# Print the top-N candidate item IDs
print("Top-N Candidate Item IDs:", top_n_items)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-aac3449e0f8c> in <cell line: 3>()
      1 # Assuming item_to_index is a dictionary mapping item IDs to numeric indices
      2 # Retrieve the item IDs corresponding to the numeric identifiers
----> 3 top_n_items = [item_id for item_id, index in item_to_index.items() if index in [302, 177, 66, 377, 370, 330, 119, 343, 12, 21]]
      4
      5 # Print the top-N candidate item IDs

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

Next steps: Explain error

```
# Define item_to_index dictionary mapping item IDs to numeric indices
item_to_index = {item_id: index for index, item_id in enumerate(unique_item_ids)}

# Retrieve the item IDs corresponding to the numeric identifiers
top_n_items = [item_id for item_id, index in item_to_index.items() if index in [302, 177, 66, 377, 370, 330, 119, 34

# Print the top-N candidate item IDs
print("Top-N Candidate Item IDs:", top_n_items)
```

```
19e-f3c3-4c7c-8b48-c53f70e30fbc', '4994f84c-fe0b-4fcf-87d2-f4848681e72a', 'ed32f1c8-64b2-491c-8c0a-182b2c80d487']
```

```
from sklearn.preprocessing import OneHotEncoder

# One-hot encode item categories
encoder = OneHotEncoder(sparse=False)
encoded_categories = encoder.fit_transform(items_df['category'].values.reshape(-1, 1))
category_columns = encoder.get_feature_names(['category'])
item_category_df = pd.DataFrame(encoded_categories, columns=category_columns, index=items_df.index)

# Merge user features with user-item interactions
user_interaction_df = pd.merge(user_engagement_df, users_df, on='user_id', how='left')

# Merge item features with user-item interactions
user_item_interaction_df = pd.merge(user_interaction_df, item_category_df, left_on='item_id', right_index=True, how=

# Add derived features or interaction features if needed
# For example, you can calculate interaction counts, average ratings, etc.

# Print the final feature-engineered DataFrame
print(user_item_interaction_df.head())
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was rena
  warnings.warn(
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-17-c875abb044fa> in <cell line: 6>()
      4 encoder = OneHotEncoder(sparse=False)
      5 encoded_categories = encoder.fit_transform(items_df['category'].values.reshape(-1, 1))
----> 6 category_columns = encoder.get_feature_names(['category'])
      7 item_category_df = pd.DataFrame(encoded_categories, columns=category_columns, index=items_df.index)
      8

AttributeError: 'OneHotEncoder' object has no attribute 'get_feature_names'
```

Next steps: Explain error

```
from sklearn.preprocessing import OneHotEncoder

# One-hot encode item categories
encoder = OneHotEncoder(sparse=False)
encoded_categories = encoder.fit_transform(items_df['category'].values.reshape(-1, 1))
category_columns = encoder.get_feature_names_out(['category'])  # Corrected
item_category_df = pd.DataFrame(encoded_categories, columns=category_columns, index=items_df.index)

# Merge user features with user-item interactions
user_interaction_df = pd.merge(user_engagement_df, users_df, on='user_id', how='left')

# Merge item features with user-item interactions
user_item_interaction_df = pd.merge(user_interaction_df, item_category_df, left_on='item_id', right_index=True, how=

# Add derived features or interaction features if needed
# For example, you can calculate interaction counts, average ratings, etc.

# Print the final feature-engineered DataFrame
print(user_item_interaction_df.head())
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was rena
  warnings.warn(
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-18-31c7f387d465> in <cell line: 13>()
     11
     12 # Merge item features with user-item interactions
---> 13 user_item_interaction_df = pd.merge(user_interaction_df, item_category_df, left_on='item_id',
right_index=True, how='left')
     14
     15 # Add derived features or interaction features if needed

 ------------------------------------ 2 frames ------------------------------------
/usr/local/lib/python3.10/dist-packages/pandas/core/reshape/merge.py in _maybe_coerce_merge_keys(self)
   1338                     inferred_right in string_types and inferred_left not in string_types
   1339                 ):
-> 1340                     raise ValueError(msg)
   1341
   1342             # datetimelikes must match exactly

ValueError: You are trying to merge on object and int64 columns. If you wish to proceed you should use
pd.concat
```

Next steps: Explain error

```
# Convert index of item_category_df to int64
item_category_df.index = item_category_df.index.astype('int64')

# Merge item features with user-item interactions
user_item_interaction_df = pd.concat(user_interaction_df, item_category_df, left_on='item_id', right_index=True, how
```

```
<ipython-input-20-4eaafff54721>:5: FutureWarning: In a future version of pandas all arguments of concat except f
  user_item_interaction_df = pd.concat(user_interaction_df, item_category_df, left_on='item_id', right_index=Tru
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-4eaafff54721> in <cell line: 5>()
      3
      4 # Merge item features with user-item interactions
----> 5 user_item_interaction_df = pd.concat(user_interaction_df, item_category_df, left_on='item_id',
right_index=True, how='left')

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

TypeError: concat() got an unexpected keyword argument 'left_on'
```

Next steps: Explain error

```
# Convert index of item_category_df to int64
item_category_df.index = item_category_df.index.astype('int64')

# Merge item features with user-item interactions using pd.concat
user_item_interaction_df = pd.concat([user_interaction_df, item_category_df], axis=1)

# Print the final feature-engineered DataFrame
print(user_item_interaction_df.head())
```

```
                                user_id                                 item_id  \
0  eb6ba928-39d2-4347-9d58-8feac4fe9ee7  f9c35bc1-359e-4b02-9521-944f2487f097
1  922f3b5e-8da1-45d1-b966-8b2937e363ca  99b570f1-2b14-46b8-bb74-82a6aa057655
2  5c3e4f8e-ca73-4bbb-8afb-7d15c9851ee1  27f9475b-1f3e-4b32-8ef1-f4f5c44f1e56
3  5008bc86-05c1-44c0-bfa9-39cb113a0f6d  53932805-ffc3-455c-880a-90882cb03107
4  27153da9-f88a-4ae0-921d-4ab48ffbc802  2dcc9a5e-b360-4fd8-899a-dff8b0e6d168

                   timestamp event_type  age  gender  category_Books  \
0 2024-03-14 12:23:09.607069      order   44   Other             0.0
1 2024-03-16 21:09:57.072796      order   39  Female             0.0
2 2024-03-01 10:25:10.993901      order   41  Female             0.0
3 2024-03-01 04:32:55.130427      click   51   Other             0.0
4 2024-03-12 08:38:03.920221      click   34    Male             1.0

   category_Clothing  category_Electronics  category_Home & Kitchen
0                1.0                   0.0                      0.0
1                1.0                   0.0                      0.0
2                0.0                   1.0                      0.0
3                1.0                   0.0                      0.0
4                0.0                   0.0                      0.0
```

```
from IPython.display import display
display(user_item_interaction_df)  # Enhanced display
```

	user_id	item_id	timestamp	event_type	age	gender	category_Books	category_Clothing	cate
0	eb6ba928-39d2-4347-9d58-8feac4fe9ee7	f9c35bc1-359e-4b02-9521-944f2487f097	2024-03-14 12:23:09.607069	order	44	Other	0.0	1.0	
1	922f3b5e-8da1-45d1-b966-8b2937e363ca	99b570f1-2b14-46b8-bb74-82a6aa057655	2024-03-16 21:09:57.072796	order	39	Female	0.0	1.0	
2	5c3e4f8e-ca73-4bbb-8afb-7d15c9851ee1	27f9475b-1f3e-4b32-8ef1-f4f5c44f1e56	2024-03-01 10:25:10.993901	order	41	Female	0.0	0.0	
3	5008bc86-05c1-44c0-bfa9-39cb113a0f6d	53932805-ffc3-455c-880a-90882cb03107	2024-03-01 04:32:55.130427	click	51	Other	0.0	1.0	
4	27153da9-f88a-4ae0-921d-4ab48ffbc802	2dcc9a5e-b360-4fd8-899a-dff8b0e6d168	2024-03-12 08:38:03.920221	click	34	Male	1.0	0.0	
...	...	...	...	...	...	...	...	...	
995	57339cb4-0887-482c-9b39-539bd8843d7b	53932805-ffc3-455c-880a-90882cb03107	2024-03-16 15:55:42.949715	click	61	Female	NaN	NaN	
	ea6b1211-681e-4e7f	f5614740-5feb-46f0	2024-03-17						

996	681e-4e71-b8bc-2ced857f443d	5feb-46f0-bb6f-deaff236f7ba	2024-03-17 23:10:08.629570	click	46	Male	NaN	NaN
997	d474876c-13e5-4ffc-8f39-fd567c948f5c	ab3a6f33-bd78-448d-8034-e6bb648c86c0	2024-03-10 19:46:13.719560	order	44	Female	NaN	NaN
998	e3d5a296-b12f-48a9-b4c4-a1513b1c81a8	693c3441-435d-4320-9710-6281f20efdc2	2024-03-10 05:52:55.261985	order	54	Other	NaN	NaN
999	e2deb4c1-7365-40bf-a391-ba4884ae15cd	5edcfe60-c7b8-4cb5-b9be-b0e705deb43a	2024-03-07 18:46:09.538832	order	34	Other	NaN	NaN

1000 rows × 10 columns

Next steps: Generate code with `user_item_interaction_df` View recommended plots

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

```
# Reset the index of user_item_interaction_df
user_item_interaction_df_reset_index = user_item_interaction_df.reset_index()

# Create a dictionary mapping item IDs to categories
item_id_to_categories = user_item_interaction_df_reset_index.set_index('item_id')[['category_Books', 'category_Cloth

# Retrieve the categories for the top-N items
top_n_item_categories = []

for item_id in top_n_item_ids:
    categories = []
    for category_column in ['category_Books', 'category_Clothing', 'category_Electronics', 'category_Home & Kitchen'
        if item_id_to_categories.get(item_id, {}).get(category_column):
            categories.append(category_column.split('_')[-1])  # Extract category name from column name
    top_n_item_categories.append(categories)

# Print the categories for the top-N items
print("Categories for the Top-N Items:")
for item_id, categories in zip(top_n_item_ids, top_n_item_categories):
    print(f"Item ID: {item_id}, Categories: {categories}")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-27-ecf4335fa53d> in <cell line: 5>()
      3
      4 # Create a dictionary mapping item IDs to categories
----> 5 item_id_to_categories = user_item_interaction_df_reset_index.reset_index('item_id')[['category_Books',
'category_Clothing', 'category_Electronics', 'category_Home & Kitchen']].to_dict(orient='index')
      6
      7 # Retrieve the categories for the top-N items

                                 4 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in _validate_index_level(self, level)
   2048                 )
   2049         elif level != self.name:
-> 2050             raise KeyError(
   2051                 f"Requested level ({level}) does not match index name ({self.name})"
   2052             )

KeyError: 'Requested level (item_id) does not match index name (None)'
```

Next steps: Explain error

```
# Reset the index of user_item_interaction_df
user_item_interaction_df_reset_index = user_item_interaction_df.reset_index()

# Create a dictionary mapping item IDs to categories
item_id_to_categories = user_item_interaction_df_reset_index.set_index('item_id')[['category_Books', 'category_Cloth

# Retrieve the categories for the top-N items
top_n_item_categories = []

for item_id in top_n_item_ids:
    categories = []
    for category_column in ['category_Books', 'category_Clothing', 'category_Electronics', 'category_Home & Kitchen'
        if item_id_to_categories.get(item_id, {}).get(category_column):
            categories.append(category_column.split('_')[-1])  # Extract category name from column name
    top_n_item_categories.append(categories)

# Print the categories for the top-N items
print("Categories for the Top-N Items:")
for item_id, categories in zip(top_n_item_ids, top_n_item_categories):
    print(f"Item ID: {item_id}, Categories: {categories}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-28-385804c73ae9> in <cell line: 5>()
      3
      4 # Create a dictionary mapping item IDs to categories
----> 5 item_id_to_categories = user_item_interaction_df_reset_index.set_index('item_id')[['category_Books',
'category_Clothing', 'category_Electronics', 'category_Home & Kitchen']].to_dict(orient='index')
      6
      7 # Retrieve the categories for the top-N items

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in to_dict(self, orient, into)
   2061         elif orient == "index":
   2062             if not self.index.is_unique:
-> 2063                 raise ValueError("DataFrame index must be unique for orient='index'.")
   2064             return into_c(
   2065                 (t[0], dict(zip(self.columns, map(maybe_box_native, t[1:]))))

ValueError: DataFrame index must be unique for orient='index'.
```

Next steps: Explain error

```
# Assuming user_item_interaction_df is your DataFrame and Top-N Candidate Item IDs is the list of item IDs
top_n_candidate_item_ids = ['dfd07d70-c92d-4993-9886-8beb55927b8d', '3457c925-d3e0-4d52-bb0c-7ad1963b310e', 'a0759c1

# Filter user_item_interaction_df for the top-N candidate item IDs
top_n_items_df = user_item_interaction_df[user_item_interaction_df['item_id'].isin(top_n_candidate_item_ids)]

# Print the filtered DataFrame
print("Items from Top-N Candidate Item IDs:")
print(top_n_items_df)
```

112	1.0	0.0
135	0.0	0.0
213	0.0	1.0
361	0.0	0.0
483	0.0	0.0
542	NaN	NaN
552	NaN	NaN
574	NaN	NaN
595	NaN	NaN
620	NaN	NaN
643	NaN	NaN
644	NaN	NaN
670	NaN	NaN
671	NaN	NaN
709	NaN	NaN
720	NaN	NaN
816	NaN	NaN
822	NaN	NaN
944	NaN	NaN
958	NaN	NaN
981	NaN	NaN
989	NaN	NaN
996	NaN	NaN

```
# Assuming user_item_interaction_df is your DataFrame and the list of item IDs is provided
item_ids_to_print = ['dfd07d70-c92d-4993-9886-8beb55927b8d', '3457c925-d3e0-4d52-bb0c-7ad1963b310e', 'a0759c1a-19b6-

# Filter user_item_interaction_df for the specified item IDs
items_to_print_df = user_item_interaction_df[user_item_interaction_df['item_id'].isin(item_ids_to_print)]

# Print the filtered DataFrame
print("Items with specified Item IDs:")
print(items_to_print_df)
```

112	1.0	0.0
135	0.0	0.0
213	0.0	1.0
361	0.0	0.0
483	0.0	0.0
542	NaN	NaN
552	NaN	NaN
574	NaN	NaN
595	NaN	NaN
620	NaN	NaN
643	NaN	NaN
644	NaN	NaN
670	NaN	NaN
671	NaN	NaN
709	NaN	NaN
720	NaN	NaN
816	NaN	NaN
822	NaN	NaN
944	NaN	NaN
958	NaN	NaN
981	NaN	NaN
989	NaN	NaN
996	NaN	NaN

```
from IPython.display import display
display(items_to_print_df)  # Enhanced display
```

.mestamp	event_type	age	gender	category_Books	category_Clothing	category_Electronics	category_Home & Kitchen
24-03-22 3.906624	click	32	Male	0.0	1.0	0.0	0.0
24-03-09 7.768706	click	52	Male	0.0	1.0	0.0	0.0
24-03-08 1.645462	click	65	Male	0.0	0.0	1.0	0.0
24-03-19 3.460866	order	45	Other	0.0	0.0	1.0	0.0
24-02-25 5.614512	click	24	Other	0.0	1.0	0.0	0.0
24-02-23 9.099391	click	27	Male	0.0	0.0	0.0	1.0
24-02-24 1.753372	order	22	Female	1.0	0.0	0.0	0.0

24-03-07 3.170246	click	51	Female	0.0	1.0	0.0	0.0
24-02-26 5.872378	click	31	Male	NaN	NaN	NaN	NaN
24-03-20 2.040779	order	42	Female	NaN	NaN	NaN	NaN
24-03-04 0.413665	click	21	Other	NaN	NaN	NaN	NaN
24-03-19 8.110209	order	61	Male	NaN	NaN	NaN	NaN
24-03-05 3.218564	click	24	Other	NaN	NaN	NaN	NaN
24-02-24 3.201230	click	47	Male	NaN	NaN	NaN	NaN

24-02-26 9.126919	order	47	Male	NaN	NaN	NaN	NaN
24-03-08 4.749567	order	45	Other	NaN	NaN	NaN	NaN
24-03-16 1.485444	click	43	Female	NaN	NaN	NaN	NaN
24-03-12 9.680258	click	32	Female	NaN	NaN	NaN	NaN
24-02-24 3.544301	click	38	Female	NaN	NaN	NaN	NaN
24-03-15 9.987880	click	55	Other	NaN	NaN	NaN	NaN
24-02-29 3.833887	click	44	Female	NaN	NaN	NaN	NaN
24-02-24 5.754763	click	54	Female	NaN	NaN	NaN	NaN

24-03-08 7.479051	order	64	Other	NaN	NaN	NaN	NaN
24-03-02 0.433251	click	50	Other	NaN	NaN	NaN	NaN
24-03-21 3.358246	order	62	Female	NaN	NaN	NaN	NaN

Next steps: Generate code with `items_to_print_df` View recommended plots

24-03-17 3.629570	click	46	Male	NaN	NaN	NaN	NaN

```
# Assuming user_item_interaction_df is your DataFrame and the list of item IDs is provided
item_ids_to_print = ['dfd07d70-c92d-4993-9886-8beb55927b8d', '3457c925-d3e0-4d52-bb0c-7ad1963b310e', 'a0759c1a-19b6-

# Filter user_item_interaction_df for the specified item IDs
items_to_print_df = user_item_interaction_df[user_item_interaction_df['item_id'].isin(item_ids_to_print)]

# Group by item ID and print all occurrences of the specified items
grouped_items_df = items_to_print_df.groupby('item_id')
for item_id, group_df in grouped_items_df:
    print(f"Item ID: {item_id}, Occurrences: {len(group_df)}")
    print(group_df)
```