Step-by-Step Demo/Tutorial: AWS SageMaker Feature Engineering with Feature Store

This tutorial will guide you through performing feature engineering on the Mall_Customers.csv dataset using Amazon SageMaker. We'll use SageMaker Feature Store to create a feature group, ingest engineered features, and make them accessible both online (for real-time inference) and offline (for batch processing via Amazon Athena). The offline store will be in S3, and we'll query it using Athena.

Prerequisites:

- You have access to an AWS account with SageMaker Studio or a SageMaker Notebook Instance set up in the us-east-1 region.
- The IAM role AmazonSageMaker-ExecutionRole-20250818T084167 is attached to your SageMaker execution environment (this role should have permissions for SageMaker, S3, Athena, and Glue).
- The dataset Mall_Customers.csv is already uploaded to your S3 bucket: s3://custom-sagemaker-bucket-s3-feature-engineering123/Mall_Customers.csv.
- Install required libraries in your SageMaker notebook (if not pre-installed): Run !pip install sagemaker pandas in a cell.

High-Level Overview:

1. Launch a SageMaker Notebook.
2. Load and engineer features from the CSV.
3. Create a Feature Group in SageMaker Feature Store.
4. Ingest data into the Feature Store (enabling online and offline access).
5. Verify online access via SageMaker SDK.
6. Query offline features via Athena.

I'll provide the complete code as a series of notebook cells. You can copy-paste these into a new Jupyter notebook (.ipynb) in SageMaker Studio. At the end, I'll explain how to download the .ipynb file.

Step 1: Launch SageMaker Studio or Notebook Instance

- Go to the AWS Management Console > Amazon SageMaker > Studio (or create a Notebook Instance if preferred).
- Create a new notebook with a Python 3 kernel (e.g., Data Science image).
- Ensure the execution role is AmazonSageMaker-ExecutionRole-20250818T084167.
- Open a new notebook file (e.g., name it FeatureEngineeringDemo.ipynb).

Step 2: Install Dependencies and Import Libraries

In the first cell of your notebook, install any missing libraries and import required modules.

```
Cell 1: Install and Import
!pip install -U sagemaker pandas boto3
import pandas as pd
import numpy as np
import time
import boto3
import sagemaker
from sagemaker.session import Session
from sagemaker.feature_store.feature_group import FeatureGroup
from sagemaker.feature_store.feature_definition import FeatureDefinition, FeatureTypeEnum
# Set up SageMaker session
sagemaker_session = sagemaker.Session()
region = 'us-east-1' # Your region
bucket = 'custom-sagemaker-bucket-s3-feature-engineering123' # Your S3 bucket
role = 'arn:aws:iam::YOUR_ACCOUNT_ID:role/AmazonSageMaker-ExecutionRole-20250818T084167' # Replace YOUR_ACCOUNT_ID with your AWS account ID
prefix = 'feature-store-demo'
# Note: Replace YOUR_ACCOUNT_ID above with your actual AWS account ID (find it in AWS Console > IAM)
```

Step 3: Load Data from S3

Load the Mall_Customers.csv from your S3 bucket into a Pandas DataFrame.

```
# Cell 2: Load Data

s3_path = f's3://{bucket}/Mall_Customers.csv'
df = pd.read_csv(s3_path)
# Quick inspection
print(df.head())
print(df.info())
```

Step 4: Perform Feature Engineering

Clean and engineer new features:

- Clean Annual_Income: Remove "EUR " and ".00" to make it numeric.
- Rename Genre to Gender for clarity.
- Create new features:
    - Age_Group: Categorize age into bins (e.g., Young, Adult, Senior).
    - Income_Spending_Ratio: Annual_Income / Spending_Score (handle division by zero if needed).

    - High_Spender: Binary flag if Spending_Score > 50.

- Add required columns for Feature Store: RecordId (unique identifier, use CustomerID) and EventTime (timestamp for ingestion).

```
# Cell 3: Feature Engineering

# Clean Annual_Income
df['Annual_Income'] = df['Annual_Income'].str.replace('EUR ', '').str.replace('.00', '').astype(float)
# Rename Genre to Gender
df.rename(columns={'Genre': 'Gender'}, inplace=True)
# New features
df['Age_Group'] = pd.cut(df['Age'], bins=[0, 25, 45, 100], labels=['Young', 'Adult', 'Senior'])
df['Income_Spending_Ratio'] = df['Annual_Income'] / df['Spending_Score'].replace(0, np.nan) # Avoid div by zero
df['High_Spender'] = (df['Spending_Score'] > 50).astype(int)
# Add required columns for Feature Store
df['RecordId'] = df['CustomerID'].astype(str) # Unique record identifier
df['EventTime'] = time.time() # Unix timestamp for ingestion
# Convert categoricals to string
df['Age_Group'] = df['Age_Group'].astype(str)
df['Gender'] = df['Gender'].astype(str)
# Drop CustomerID if not needed as feature
df.drop(columns=['CustomerID'], inplace=True)
# Inspection
print(df.head())
print(df.dtypes)
```

Step 5: Define and Create Feature Group

Define the schema for the Feature Group and create it. Enable both online and offline stores.

```
# Cell 4: Define Feature Group
feature_group_name = 'mall-customers-features'
# Define feature definitions based on DataFrame
feature_definitions = [
FeatureDefinition('RecordId', FeatureTypeEnum.STRING),
FeatureDefinition('Gender', FeatureTypeEnum.STRING),
FeatureDefinition('Age', FeatureTypeEnum.INTEGRAL),
FeatureDefinition('Annual_Income', FeatureTypeEnum.FRACTIONAL),
FeatureDefinition('Spending_Score', FeatureTypeEnum.INTEGRAL),
FeatureDefinition('Age_Group', FeatureTypeEnum.STRING),
FeatureDefinition('Income_Spending_Ratio', FeatureTypeEnum.FRACTIONAL),
FeatureDefinition('High_Spender', FeatureTypeEnum.INTEGRAL),
FeatureDefinition('EventTime', FeatureTypeEnum.FRACTIONAL)
]
# Create Feature Group
feature_group = FeatureGroup(
name=feature_group_name,
sagemaker_session=sagemaker_session,
feature_definitions=feature_definitions
)
# Create the group with offline and online stores enabled
feature_group.create(
s3_uri=f's3://{bucket}/{prefix}',
record_identifier_name='RecordId',
event_time_feature_name='EventTime',
role_arn=role,
enable_online_store=True # Enables online store
)
# Wait for creation (poll status)
status = feature_group.describe()['FeatureGroupStatus']
while status == 'Creating':
print('Waiting for Feature Group Creation...')
time.sleep(5)
status = feature_group.describe()['FeatureGroupStatus']
print(f'Feature Group {feature_group_name} created successfully!')
```

Step 6: Ingest Data into Feature Store

Ingest the engineered DataFrame into the Feature Group.

```
# Cell 5: Ingest Data
feature_group.ingest(data_frame=df, max_workers=3, wait=True)
print('Data ingested successfully!')
```

Step 7: Verify Online Store Access

Retrieve a sample feature record in real-time from the online store.

```
# Cell 6: Query Online Store
runtime_client = boto3.client('sagemaker-featurestore-runtime', region_name=region)
# Get a single record
response = runtime_client.get_record(
FeatureGroupName=feature_group_name,
RecordIdentifierValueAsString='1' # Example RecordId (from original CustomerID=1)
)
print(response)
```

You should see the feature values for that record.

Step 8: Access Offline Store via Athena

The offline store is in S3 (under s3://custom-sagemaker-bucket-s3-feature-engineering123/feature-store-demo/...) and registered in AWS Glue as a database/table for Athena querying.

Steps to Query via Athena:

1. Go to AWS Console > Amazon Athena.
2. In the Query Editor, select the data source as AwsDataCatalog.
3. The database name is auto-generated as sagemaker_featurestore (default). If not visible, run a Glue Crawler on the offline S3 path or wait ~1 hour for auto-sync.
4. The table name is <feature_group_name>_<account_id>_<region> (e.g., mall-customers-features_123456789012_us-east-1).
5. Run a sample query

*SELECT * FROM "sagemaker_featurestore"."mall-customers-features_123456789012_us-east-1" LIMIT 10;*

   - Replace the table name with your exact one (check in Glue Console > Databases > Tables).
   - Columns include your features plus metadata like write_time, is_deleted, etc.

6. For time-based queries: Use eventtime (e.g., WHERE eventtime > UNIX_TIMESTAMP('2025-08-18')).
7. If the table isn't visible:
   1. Go to AWS Glue > Crawlers > Create Crawler.
   2. Set crawler to scan the offline S3 path (from Feature Group description: feature_group.describe()['OfflineStoreConfig']['S3StorageConfig']['S3Uri']).

3. Run the crawler to populate the Glue catalog.

In your notebook, you can also query Athena programmatically:

```
# Cell 7: Query Athena from Notebook (Optional)

athena_query = feature_group.athena_query()
table_name = athena_query.table_name
# Run query
athena_query.run(query_string=f'SELECT * FROM "{table_name}" LIMIT 5',
output_location=f's3://{bucket}/query_results/')
athena_query.wait()
result_df = athena_query.as_dataframe()
print(result_df)
```

Step 9: Cleanup (Optional)

Delete the Feature Group when done.

```
# Cell 8: Cleanup

feature_group.delete()
```