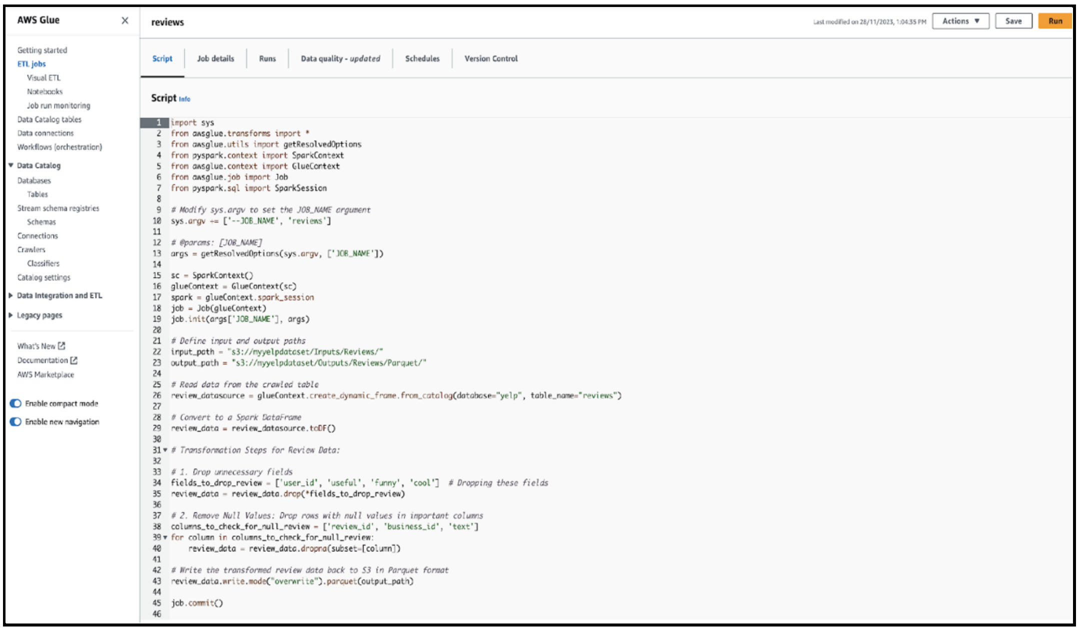
The 'reviews' Transformation Script

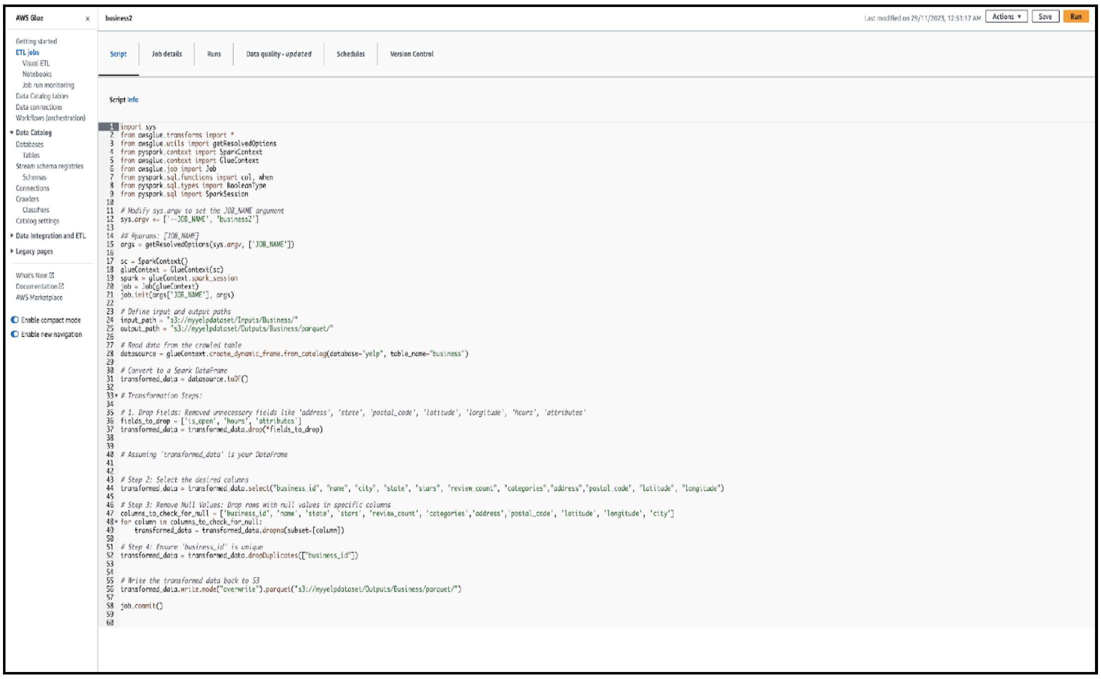
Each line of code serving a specific purpose in reshaping the 'Reviews' dataset.



```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql import SparkSession

# Modify sys.argv to set the JOB_NAME argument
sys.argv += ['--JOB_NAME', 'reviews']

# @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Define input and output paths
input_path = "s3://myyelpdataset/Inputs/Reviews/"
output_path = "s3://myyelpdataset/Outputs/Reviews/Parquet/"

# Read data from the crawled table
review_datasource = glueContext.create_dynamic_frame.from_catalog(database="yelp", table_name="reviews")

# Convert to a Spark DataFrame
review_data = review_datasource.toDF()

# Transformation Steps for Review Data:

# 1. Drop unnecessary fields
fields_to_drop_review = ['user_id', 'useful', 'funny', 'cool']  # Dropping these fields
review_data = review_data.drop(*fields_to_drop_review)

# 2. Remove Null Values: Drop rows with null values in important columns
columns_to_check_for_null_review = ['review_id', 'business_id', 'text']
for column in columns_to_check_for_null_review:
    review_data = review_data.dropna(subset=[column])

# Write the transformed review data back to S3 in Parquet format
review_data.write.mode("overwrite").parquet(output_path)

job.commit()

```

The 'business2' Transformation Script

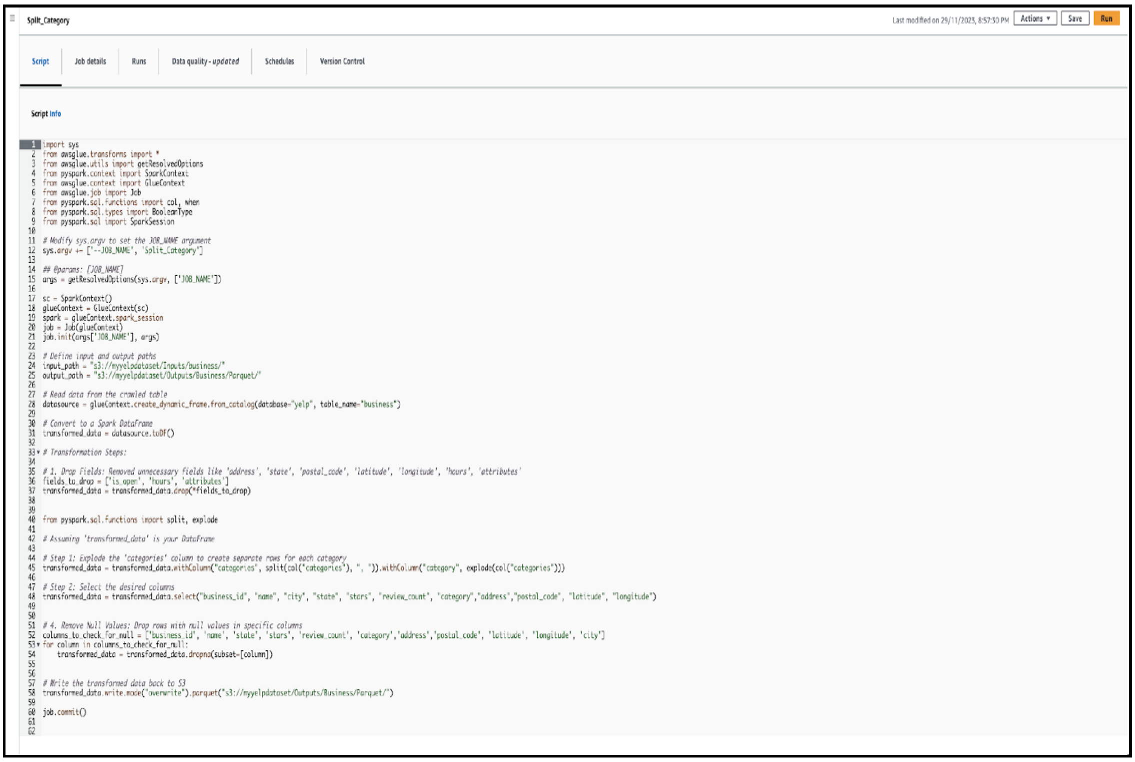
Each line of code in this script was strategically written to address specific aspects of the 'Business' dataset



```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import col, when
from pyspark.sql.types import BooleanType
from pyspark.sql import SparkSession

# Modify sys.argv to set the JOB_NAME argument
sys.argv += ['--JOB_NAME', 'business2']

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Define input and output paths
input_path = "s3://myyelpdataset/Inputs/Business/"
output_path = "s3://myyelpdataset/Outputs/Business/parquet/"

# Read data from the crawled table
datasource = glueContext.create_dynamic_frame.from_catalog(database="yelp", table_name="business")

# Convert to a Spark DataFrame
transformed_data = datasource.toDF()

# Transformation Steps:

# 1. Drop Fields: Removed unnecessary fields like 'address', 'state', 'postal_code', 'latitude', 'longitude', 'hours', 'attributes'
fields_to_drop = ['is_open', 'hours', 'attributes']
transformed_data = transformed_data.drop(*fields_to_drop)


# Assuming 'transformed_data' is your DataFrame


# Step 2: Select the desired columns
transformed_data = transformed_data.select("business_id", "name", "city", "state", "stars", "review_count", "categories","address","postal_code", "latitude", "longitude")

# Step 3: Remove Null Values: Drop rows with null values in specific columns
columns_to_check_for_null = ['business_id', 'name', 'state', 'stars', 'review_count', 'categories','address','postal_code', 'latitude', 'longitude', 'city']
for column in columns_to_check_for_null:
    transformed_data = transformed_data.dropna(subset=[column])

# Step 4: Ensure 'business_id' is unique
transformed_data = transformed_data.dropDuplicates(["business_id"])


# Write the transformed data back to S3
transformed_data.write.mode("overwrite").parquet("s3://myyelpdataset/Outputs/Business/parquet/")

job.commit()


```

'Split_Category' Transformation Script

This script was the key to unlocking a more nuanced view of the 'Business' dataset by splitting the 'Category' field into individual rows.



```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import col, when
from pyspark.sql.types import BooleanType
from pyspark.sql import SparkSession

# Modify sys.argv to set the JOB_NAME argument
sys.argv += ['--JOB_NAME', 'Split_Category']

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Define input and output paths
input_path = "s3://myyelpdataset/Inputs/business/"
output_path = "s3://myyelpdataset/Outputs/Business/Parquet/"

# Read data from the crawled table
datasource = glueContext.create_dynamic_frame.from_catalog(database="yelp", table_name="business")

# Convert to a Spark DataFrame
transformed_data = datasource.toDF()

# Transformation Steps:

# 1. Drop Fields: Removed unnecessary fields like 'address', 'state', 'postal_code', 'latitude', 'longitude', 'hours', 'attributes'
fields_to_drop = ['is_open', 'hours', 'attributes']
transformed_data = transformed_data.drop(*fields_to_drop)


from pyspark.sql.functions import split, explode

# Assuming 'transformed_data' is your DataFrame

# Step 1: Explode the 'categories' column to create separate rows for each category
transformed_data = transformed_data.withColumn("categories", split(col("categories"), ", ")).withColumn("category", explode(col("categories")))

# Step 2: Select the desired columns
transformed_data = transformed_data.select("business_id", "name", "city", "state", "stars", "review_count", "category","address","postal_code", "latitude", "longitude")


# 4. Remove Null Values: Drop rows with null values in specific columns
columns_to_check_for_null = ['business_id', 'name', 'state', 'stars', 'review_count', 'category','address','postal_code', 'latitude', 'longitude', 'city']
for column in columns_to_check_for_null:
    transformed_data = transformed_data.dropna(subset=[column])


# Write the transformed data back to S3
transformed_data.write.mode("overwrite").parquet("s3://myyelpdataset/Outputs/Business/Parquet/")

job.commit()
```