Task.py:

import os

import sys

import logging

import requests

import geohash2

from dotenv import load\_dotenv

from pyspark.sql import SparkSession, DataFrame

from pyspark.sql.functions import col, lit, udf

from pyspark.sql.types import StringType, DoubleType

from typing import Optional, Tuple

# Configure logging

logging.basicConfig(

    level=logging.INFO,

    format='%(asctime)s - %(levelname)s - %(message)s',

    handlers=[logging.StreamHandler(sys.stdout)]

)

logger = logging.getLogger(\_\_name\_\_)

# Load environment variables

load\_dotenv()

def create\_spark\_session() -> SparkSession:

    """

    Create and return a Spark session

    """

    return SparkSession.builder \

        .appName("Spark ETL Task") \

        .config("spark.sql.files.maxPartitionBytes", "128MB") \

        .config("spark.sql.adaptive.enabled", "true") \

        .getOrCreate()

def fetch\_geocoding(address: str) -> Tuple[Optional[float], Optional[float]]:

    """

    Fetch latitude and longitude for an address using OpenCage Geocoding API

    """

    api\_key = os.getenv("OPENCAGE\_API\_KEY")

    if not api\_key:

        raise ValueError("OpenCage API key is not set in the environment variables")

    try:

        response = requests.get(

            f"https://api.opencagedata.com/geocode/v1/json?q={address}&key={api\_key}",

            timeout=10

        )

        response.raise\_for\_status()

        data = response.json()

        if data['results']:

            location = data['results'][0]['geometry']

            return location['lat'], location['lng']

        return None, None

    except requests.RequestException as e:

        logger.error(f"Error fetching geocoding for address '{address}': {e}")

        return None, None

def generate\_geohash(lat: float, lon: float, precision: int = 4) -> Optional[str]:

    """

    Generate a geohash for the given latitude and longitude

    """

    try:

        return geohash2.encode(lat, lon, precision=precision)

    except Exception as e:

        logger.error(f"Error generating geohash for lat={lat}, lon={lon}: {e}")

        return None

def load\_restaurant\_data(spark: SparkSession, file\_path: str) -> DataFrame:

    """

    Load restaurant data from a CSV file and handle null values for latitude and longitude

    """

    restaurant\_df = spark.read.csv(file\_path, header=True, inferSchema=True)

    # Filter and correct null latitude and longitude

    udf\_fetch\_geocoding = udf(lambda address: fetch\_geocoding(address), StringType())

    restaurant\_df = restaurant\_df.withColumn(

        "latitude",

        col("latitude").cast(DoubleType())

    ).withColumn(

        "longitude",

        col("longitude").cast(DoubleType())

    ).withColumn(

        "geohash",

        udf(lambda lat, lon: generate\_geohash(lat, lon), StringType())(col("latitude"), col("longitude"))

    )

    return restaurant\_df

def load\_weather\_data(spark: SparkSession, file\_path: str) -> DataFrame:

    """

    Load weather data from a CSV file

    """

    return spark.read.csv(file\_path, header=True, inferSchema=True)

def enrich\_data(restaurant\_df: DataFrame, weather\_df: DataFrame) -> DataFrame:

    """

    Enrich restaurant data with weather data by performing a left join on geohash

    """

    enriched\_df = restaurant\_df.join(weather\_df, "geohash", "left\_outer")

    return enriched\_df

def save\_enriched\_data(df: DataFrame, output\_path: str):

    """

    Save the enriched DataFrame in Parquet format

    """

    df.write.mode("overwrite").partitionBy("geohash").parquet(output\_path)

    logger.info(f"Enriched data saved to {output\_path}")

def main():

    """

    Main ETL process

    """

    spark = create\_spark\_session()

    try:

        restaurant\_data\_path = "data/restaurant\_data.csv"

        weather\_data\_path = "data/weather\_data.csv"

        output\_path = "output/enriched\_data"

        # Check if input files exist

        if not os.path.exists(restaurant\_data\_path):

            logger.error(f"Restaurant data file not found: {restaurant\_data\_path}")

            raise FileNotFoundError(f"File not found: {restaurant\_data\_path}")

        if not os.path.exists(weather\_data\_path):

            logger.error(f"Weather data file not found: {weather\_data\_path}")

            raise FileNotFoundError(f"File not found: {weather\_data\_path}")

        # Load datasets

        restaurant\_df = load\_restaurant\_data(spark, restaurant\_data\_path)

        weather\_df = load\_weather\_data(spark, weather\_data\_path)

        # Enrich data

        enriched\_df = enrich\_data(restaurant\_df, weather\_df)

        # Save enriched data

        save\_enriched\_data(enriched\_df, output\_path)

        logger.info("ETL process completed successfully")

    except Exception as e:

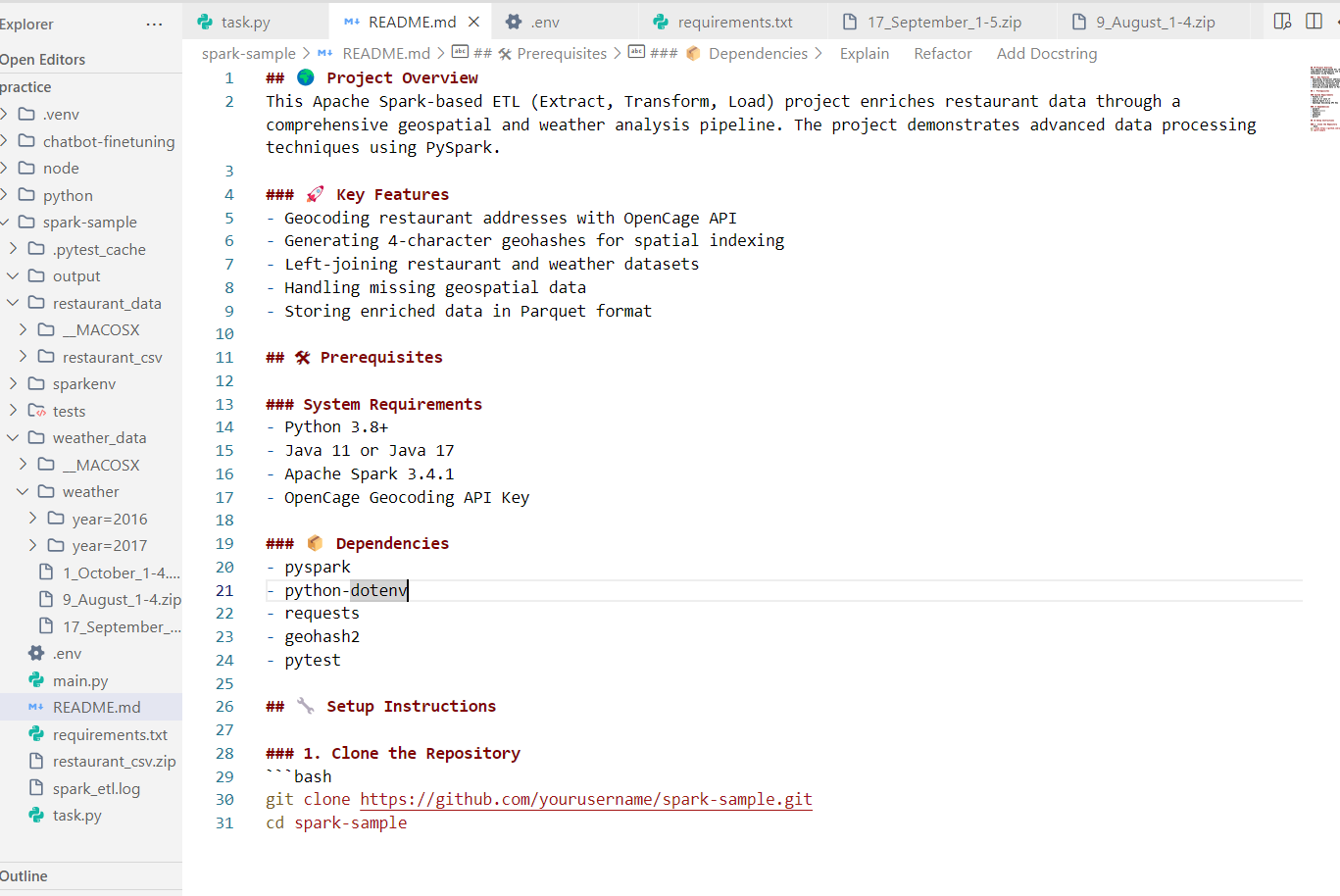
        logger.error(f"ETL process failed: {e}")

    finally:

        spark.stop()

if \_\_name\_\_ == "\_\_main\_\_":

    main()



<https://github.com/dilrabonu/Machine-Learning/blob/main/task.py>

<https://github.com/dilrabonu/Machine-Learning/blob/main/README.md>