```
'''
-- 首先在MySQL中创建sales表
-- 先在MySQL中创建一个名为spark的数据库（如果不存在）
-- CREATE DATABASE IF NOT EXISTS spark;
-- USE spark;

CREATE TABLE IF NOT EXISTS sales (
    sales_id INT,                -- 销售ID
    customer_id INT,             -- 客户ID
    product_id INT,              -- 产品ID
    sale_datetime DATETIME, -- 销售日期时间
    price DECIMAL(10,2),       -- 价格（最多10位，2位小数）
    quantity INT                 -- 数量
);

-- 解释:
-- 1. sales_id: 每条销售记录的唯一标识符
-- 2. customer_id: 购买产品的客户编号
-- 3. product_id: 被销售的产品编号
-- 4. sale_datetime: 销售发生的具体时间
-- 5. price: 产品单价，DECIMAL(10,2)表示最多10位数，其中2位是小数
-- 6. quantity: 购买数量
'''

# 导入必要的库
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType
from pyspark.sql.functions import to_timestamp, col
import pymysql

# ------------------------------
# 第一部分：创建Spark会话
# ------------------------------
# Spark会话是与Spark集群通信的入口点
# 相当于打开一个Spark工作环境
# 创建Spark会话(Spark 세션 생성)
spark = SparkSession.builder ₩
    .appName("Sales Data to MySQL") ₩
```

```
    .config("spark.sql.legacy.timeParserPolicy", "LEGACY") ₩
    .getOrCreate()

# 教学重点：
# 1. SparkSession: Spark 2.0+的核心入口点
# 2. .config(): 用于设置Spark配置参数
# 3. "LEGACY": 允许解析两位数的年份（如"13"表示2013年）

# ------------------------------
# 第二部分：定义数据模式
# ------------------------------
# 为CSV文件定义结构（模式），告诉Spark每列的数据类型
#  定义CSV文件模式(CSV 파일 스키마 정의)
schema = StructType([
    StructField("sales_id", IntegerType(), True),
    StructField("customer_id", IntegerType(), True),
    StructField("product_id", IntegerType(), True),
    StructField("sale_datetime", StringType(), True),
    StructField("price", DoubleType(), True),
    StructField("quantity", IntegerType(), True)
])

# 读取数据(데이터 읽기)
df = spark.read ₩
    .option("header", False) ₩
    .option("delimiter", ",") ₩
    .schema(schema) ₩
    .csv("file:///home/hadoop/Sales.txt")

# 日期格式转换(날짜 형식 변환)
df = df.withColumn("sale_datetime",
                   to_timestamp(col("sale_datetime"), "MM/dd/yy HH:mm:ss"))

print(f"数据读取成功! 总计 {df.count()} 条记录") #f"데이터 읽기 성공! 총 {df.count()} 건")


# MySQL保存函数 (用于foreachPartition)(MySQL 저장 함수 (foreachPartition용))
def save_partition_to_mysql(partition):
```

```
    # 为每个分区创建MySQL连接(각 파티션별로 MySQL 연결 생성)
    conn = pymysql.connect(
        host='localhost',
        user='root',
        password='123456',
        database='spark',
        charset='utf8mb4'
    )

    try:
        with conn.cursor() as cursor:
            sql = """
            INSERT INTO sales (sales_id, customer_id, product_id, sale_datetime, price, quantity)
            VALUES (%s, %s, %s, %s, %s, %s)
            """

            batch = []
            for row in partition:
                batch.append((
                    int(row.sales_id),
                    int(row.customer_id),
                    int(row.product_id),
                    row.sale_datetime,  # Python datetime 对象
                    float(row.price),
                    int(row.quantity)
                ))

                # 当批量大小达到1000时保存(배치 크기가 1000이 되면 저장)
                if len(batch) >= 1000:
                    cursor.executemany(sql, batch)
                    conn.commit()
                    batch = []

            # 保存剩余数据(남은 데이터 저장)
            if batch:
                cursor.executemany(sql, batch)
                conn.commit()
```

```
        except Exception as e:
            print(f"分区错误: {e}")
            conn.rollback()
        finally:
            conn.close()


# 清空表 (可选):테이블 초기화 (선택사항)
def truncate_table():
    conn = pymysql.connect(
        host='localhost',
        user='root',
        password='123456',
        database='spark',
        charset='utf8mb4'
    )
    try:
        with conn.cursor() as cursor:
            cursor.execute("TRUNCATE TABLE sales")
            conn.commit()
            print("表已清空") #테이블 초기화 완료
    finally:
        conn.close()


#  清空表:테이블 초기화
truncate_table()

# 使用foreachPartition进行分布式保存(foreachPartition을 사용하여 분산 저장)
print("正在保存数据到MySQL...")
df.foreachPartition(save_partition_to_mysql)
print("数据已成功保存到MySQL!")

# 数据验证
df.show(5)

# 关闭Spark会话
spark.stop()
```