```
from pyspark.sql import SparkSession
from pyspark.sql.types import StringType

from pyspark.sql.functions import col, udf, to_timestamp, to_date, collect_list,
unix_timestamp
from pyspark.pandas.window import Rolling
import pyspark.sql.functions as f
import pyspark.pandas as ps

ps.set_option('compute.ops_on_diff_frames', True)

spark = SparkSession.builder.appName("DSW_Final_Project").getOrCreate()

start_date = "2017-01-01"
end_date = "2019-01-01"

def rolling_mean(df, col_name, window_size):
    rolling_mean = df[col_name].rolling(window_size).mean()
    df[f"Mean of {col_name} for {window_size} days"] = rolling_mean
    return df

def rolling_std(df, col_name, window_size):
    rolling_mean = df[col_name].rolling(window_size).std()
    df[f"Std of {col_name} for {window_size} days"] = rolling_mean
    return df


time_series_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("delimiter", ";")\
    .load("s3://dsw-bittweet-bucket/BTC_Prices.csv")

dollar_cols = ['open', 'close', 'high', 'low', 'volume', 'marketCap']
for col in dollar_cols:
    time_series_df = time_series_df.withColumn(col, f.regexp_replace(col, "\$",
""))
    time_series_df = time_series_df.withColumn(col, f.regexp_replace(col, ",", ""))
    time_series_df = time_series_df.withColumn(col,
time_series_df[col].cast("float"))

# convert PySpark DataFrame to Pandas DataFrame
time_series_df = time_series_df.withColumn("date",
to_date(to_timestamp(time_series_df.timestamp), 'yyyy-MM-dd'))
time_series_df = time_series_df.filter(time_series_df.date.between(start_date,
end_date))

pandas_time_series_df = time_series_df.to_pandas_on_spark()
pandas_time_series_df = pandas_time_series_df.sort_values(by=['date'])
print('here1')
# Rolling Means

pandas_time_series_df = rolling_mean(pandas_time_series_df, "open", 3)
print('h1')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "open", 7)
print('h2')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "open", 30)
print('h3')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "close", 3)
```

```
print('h4')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "close", 7)
print('h5')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "close", 30)
print('h6')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "volume", 3)
print('h7')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "volume", 7)
print('h8')
pandas_time_series_df = rolling_mean(pandas_time_series_df, "volume", 30)

pandas_time_series_df.to_spark().repartition(1).write.csv('s3://dsw-bittweet-
bucket/intermediate1.csv', header=True, sep = ',')

pandas_time_series_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("sep", ",")\

.load('s3://dsw-bittweet-bucket/intermediate1.csv').to_pandas_on_spark().sort_value
s(by=['date'])

print(pandas_time_series_df.head(10))


# Rolling STDs
print('h9')
pandas_time_series_df = rolling_std(pandas_time_series_df, "open", 3)
print('h10')
pandas_time_series_df = rolling_std(pandas_time_series_df, "open", 7)
print('h11')
pandas_time_series_df = rolling_std(pandas_time_series_df, "open", 30)
print('h12')
pandas_time_series_df = rolling_std(pandas_time_series_df, "close", 3)
print('h13')

pandas_time_series_df.to_spark().repartition(1).write.csv('s3://dsw-bittweet-
bucket/intermediate2.csv', header=True, sep = ',')

pandas_time_series_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("sep", ",")\

.load('s3://dsw-bittweet-bucket/intermediate2.csv').to_pandas_on_spark().sort_value
s(by=['date'])

pandas_time_series_df = rolling_std(pandas_time_series_df, "close", 7)
print('h14')
pandas_time_series_df = rolling_std(pandas_time_series_df, "close", 30)
print('h15')
pandas_time_series_df = rolling_std(pandas_time_series_df, "volume", 3)
print('h16')
pandas_time_series_df = rolling_std(pandas_time_series_df, "volume", 7)
print('h17')
pandas_time_series_df = rolling_std(pandas_time_series_df, "volume", 30)

# Avg Price and Previous Day Avg price
# pandas_time_series_df['Avg Price'] = (pandas_time_series_df['High'] +
pandas_time_series_df['Low']) / 2
# pandas_time_series_df['Prev Day Avg Price'] = pandas_time_series_df['Avg
Price'].shift(1)

#  Previous Day close price
print('h18')
pandas_time_series_df['Prev Day Close Price'] =
pandas_time_series_df['close'].shift(1)
print('h19')
pandas_time_series_df['Next Day Close Price'] =
pandas_time_series_df['close'].shift(-1)

print('h20')

pandas_time_series_df.to_spark().repartition(1).write.csv('s3://dsw-bittweet-
bucket/intermediate3.csv', header=True, sep = ',')

pandas_time_series_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("sep", ",")\

.load('s3://dsw-bittweet-bucket/intermediate3.csv').to_pandas_on_spark().sort_value
s(by=['date'])


twitter_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("sep", ",")\
    .load("s3://dsw-bittweet-bucket/preprocessed_tweets.csv")
print('here2')
twitter_df = twitter_df.to_pandas_on_spark()
twitter_df = twitter_df.sort_values(by=['date'])
twitter_df = rolling_mean(twitter_df, 'sentiment', 3)
twitter_df = rolling_mean(twitter_df, 'sentiment', 7)
twitter_df = rolling_mean(twitter_df, 'sentiment', 30)

twitter_df = rolling_std(twitter_df, 'sentiment', 3)
twitter_df = rolling_std(twitter_df, 'sentiment', 7)
twitter_df = rolling_std(twitter_df, 'sentiment', 30)
print('here2.25')
twitter_df = twitter_df.set_index('date')


print('here2.5')
pandas_time_series_df = pandas_time_series_df.set_index('date')
print('here3')
joined_spark_df = pandas_time_series_df.join(twitter_df,on = 'date', how = "left",
lsuffix = 'left_').reset_index()

print(joined_spark_df.head(10))

joined_spark_df = joined_spark_df.sort_values(by=['date'])

joined_spark_df = joined_spark_df.dropna()

print('here4')
joined_spark_df.to_spark().repartition(1).write.csv('s3://dsw-bittweet-bucket/
final_dataset.csv', header=True, sep = ',')

spark.stop()
```