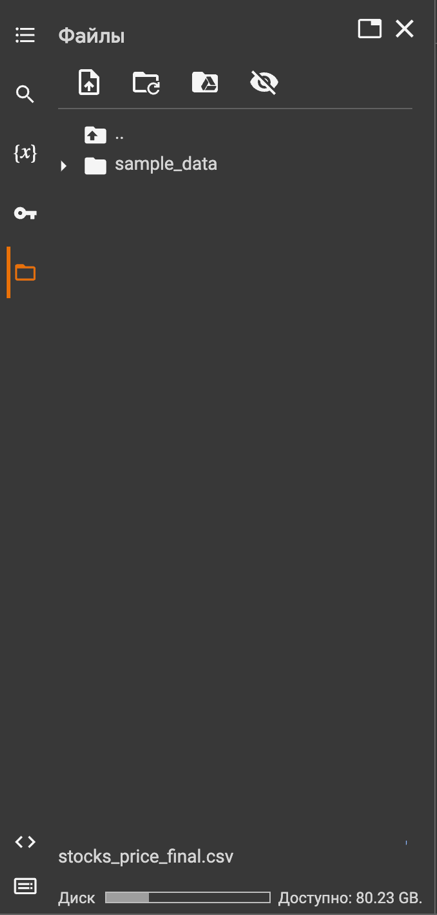
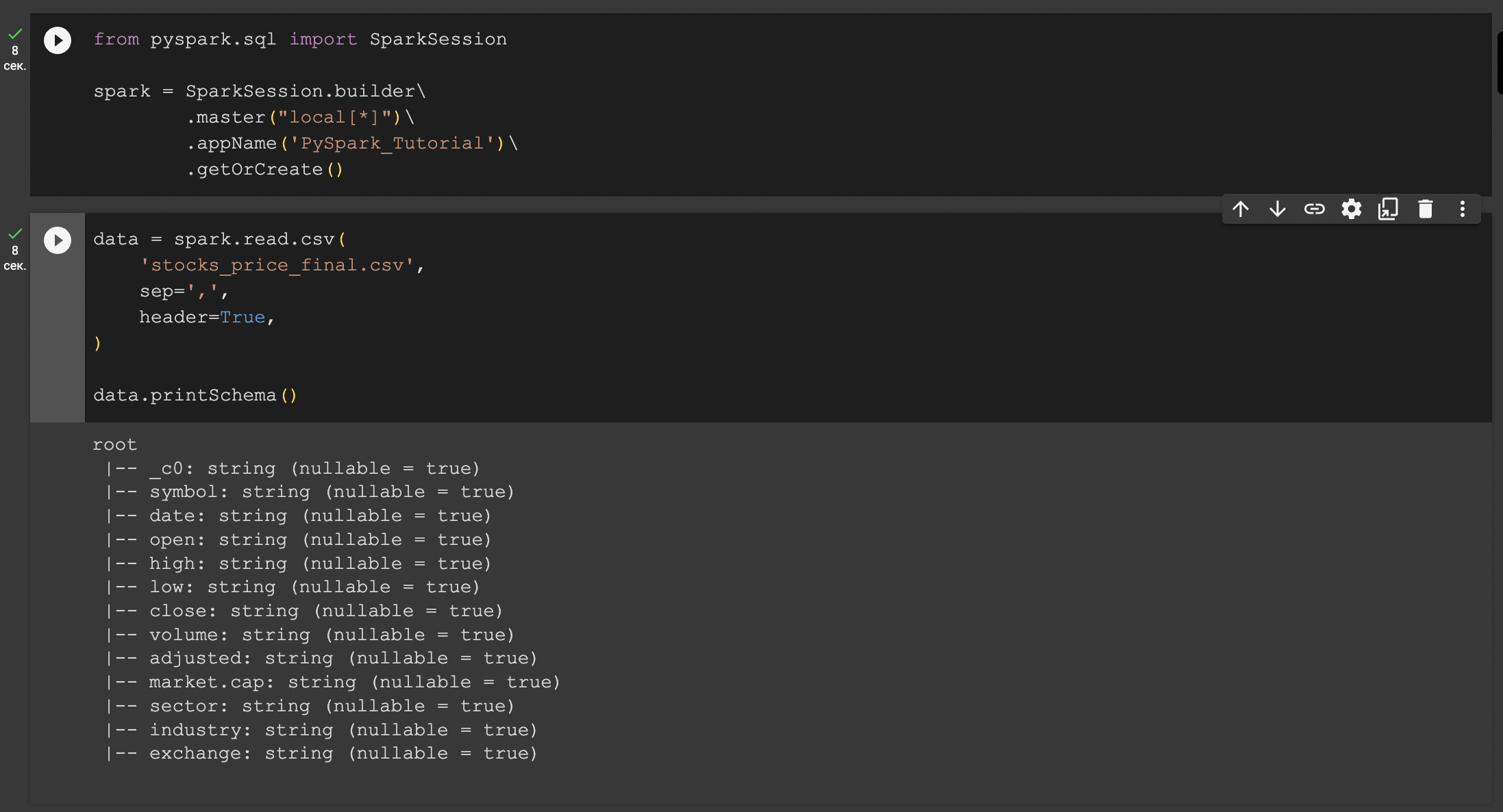
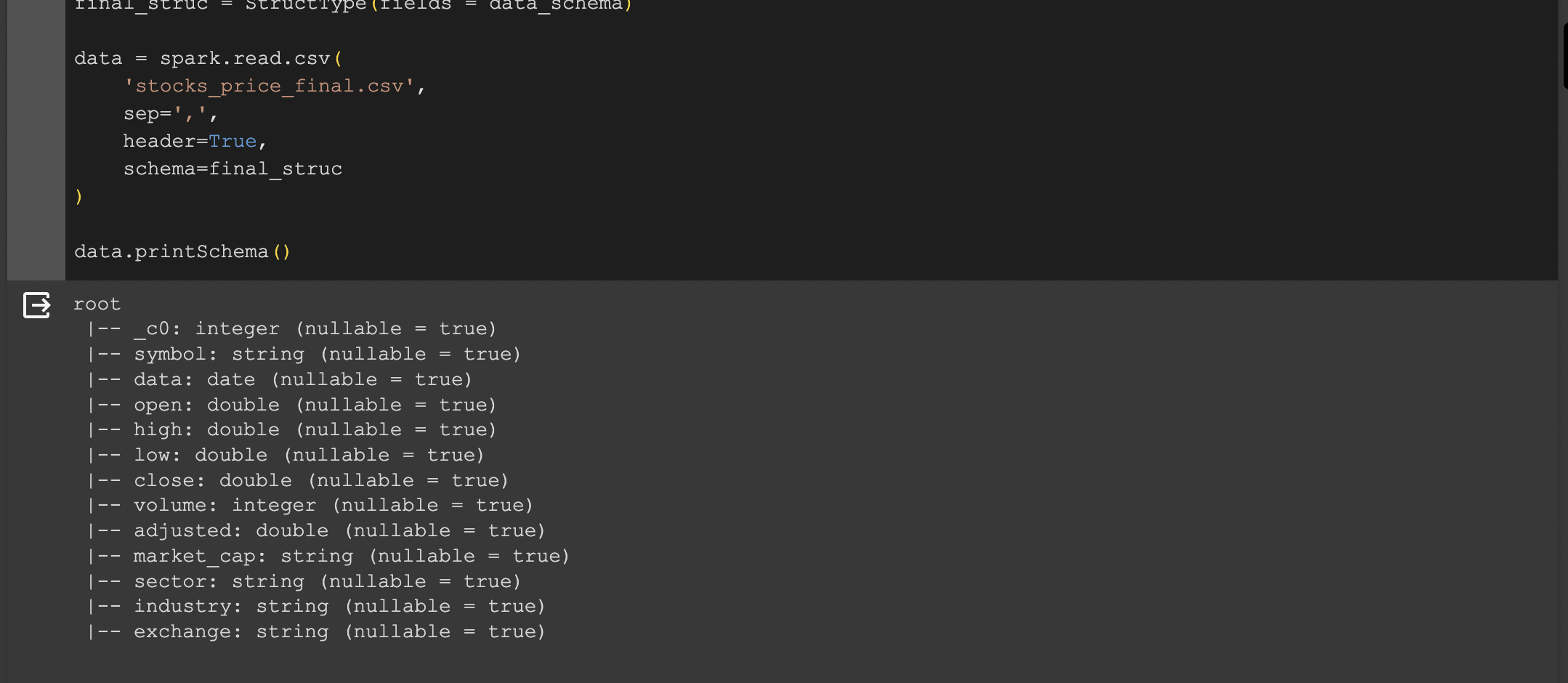
**1 часть – Google Colab**

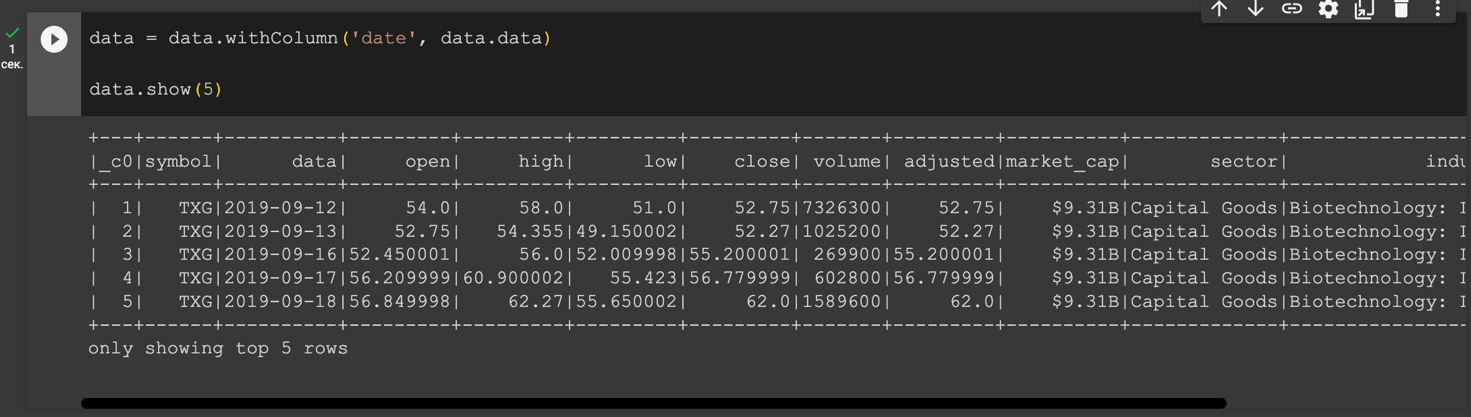
****

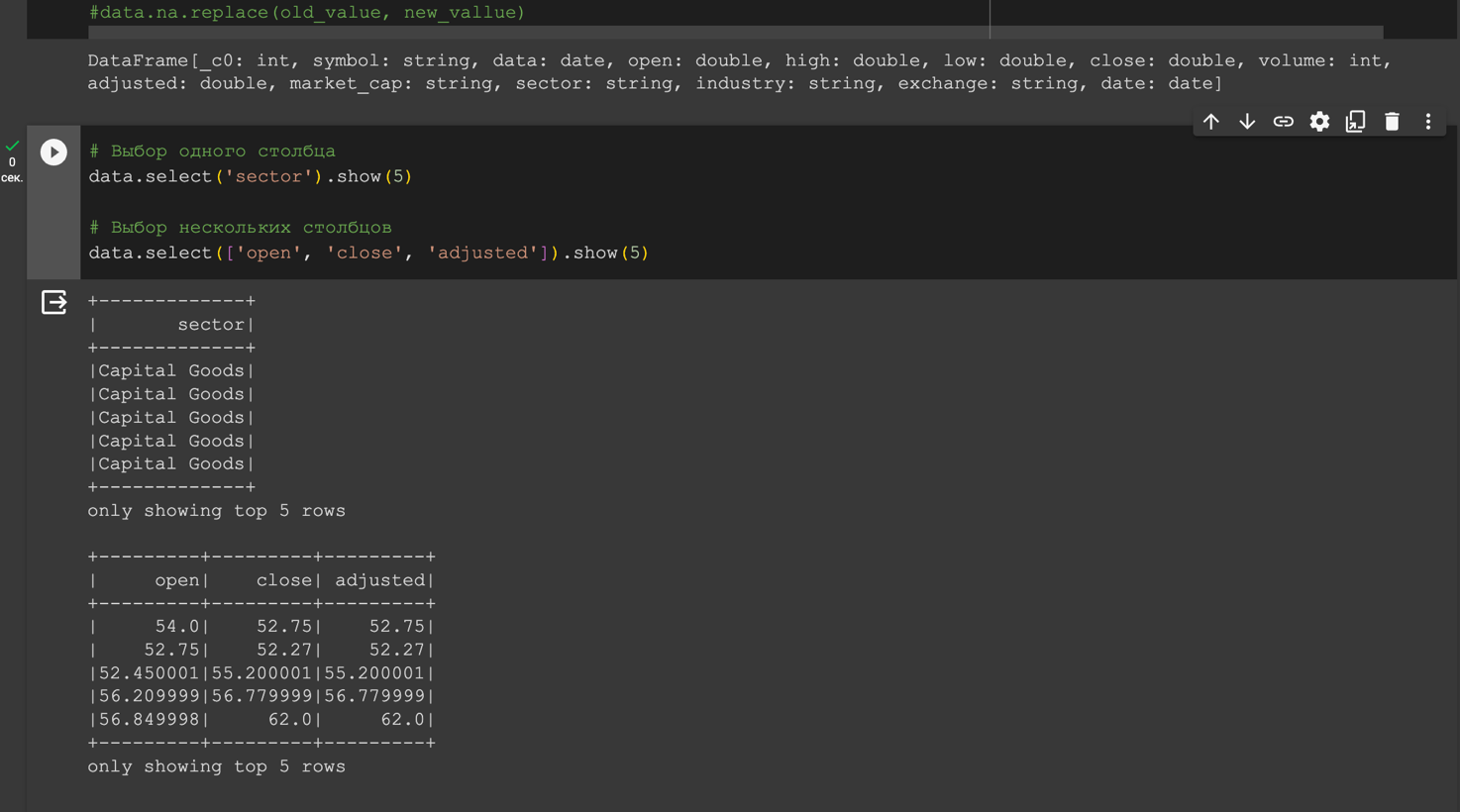
****

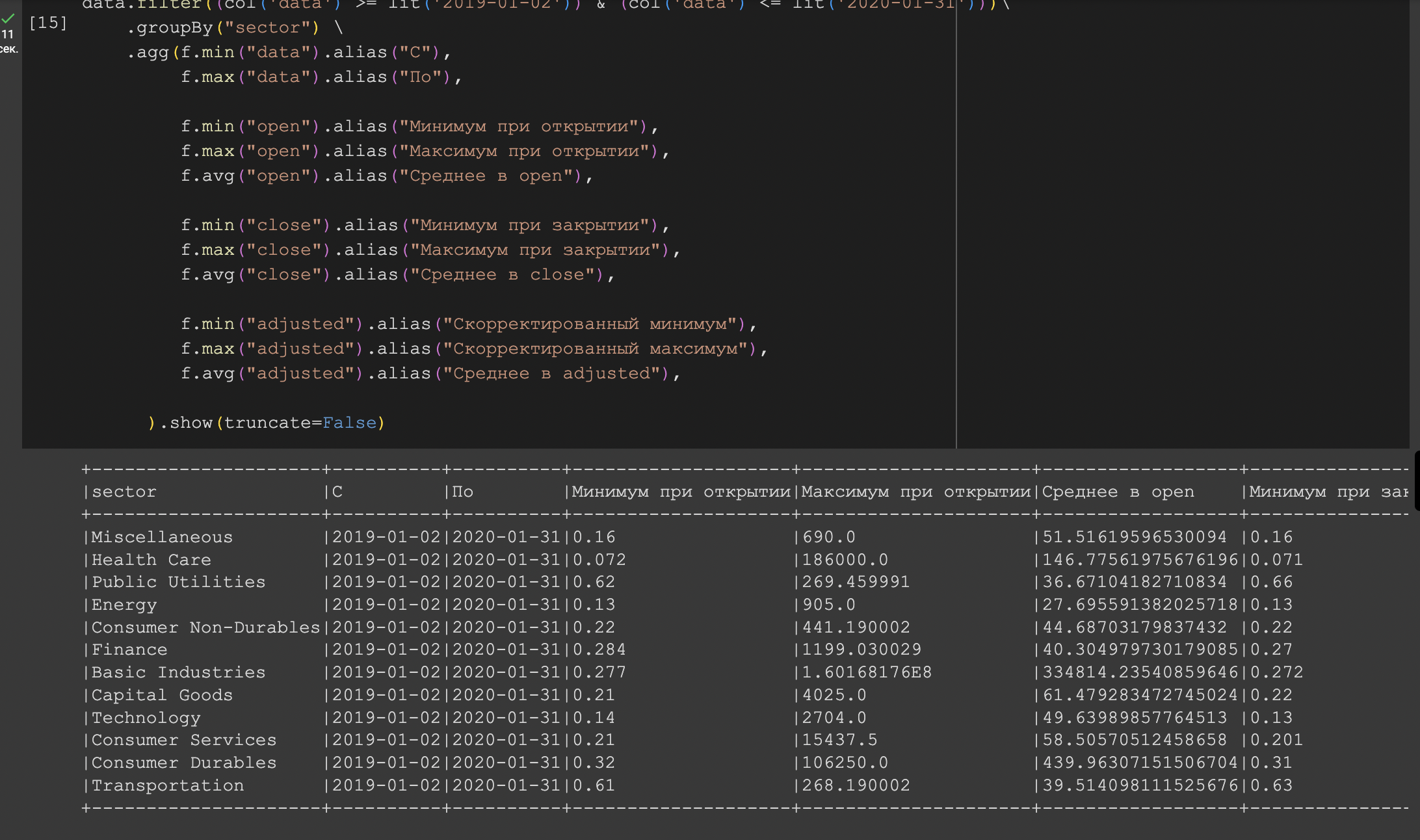
****

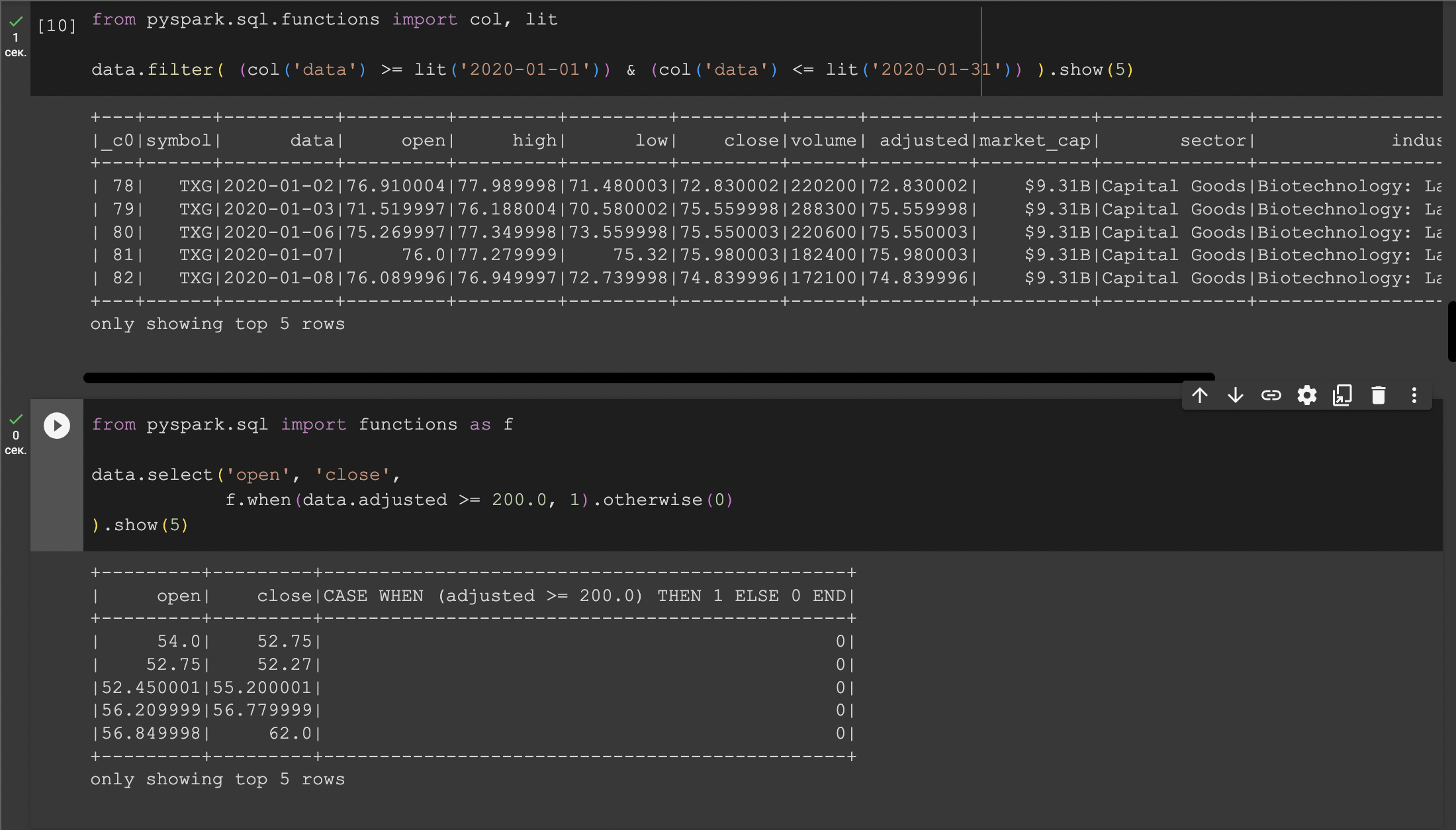
****

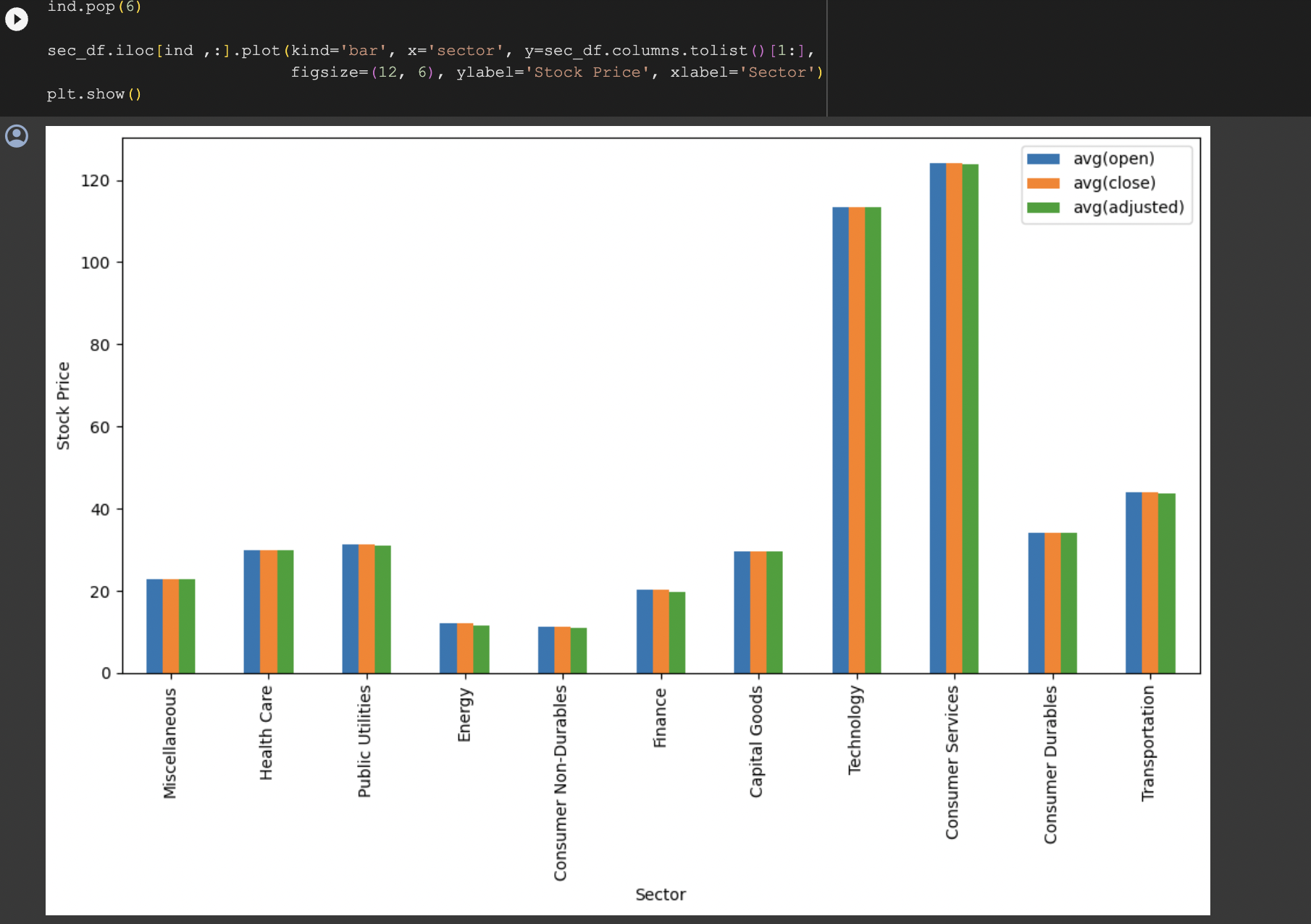
****

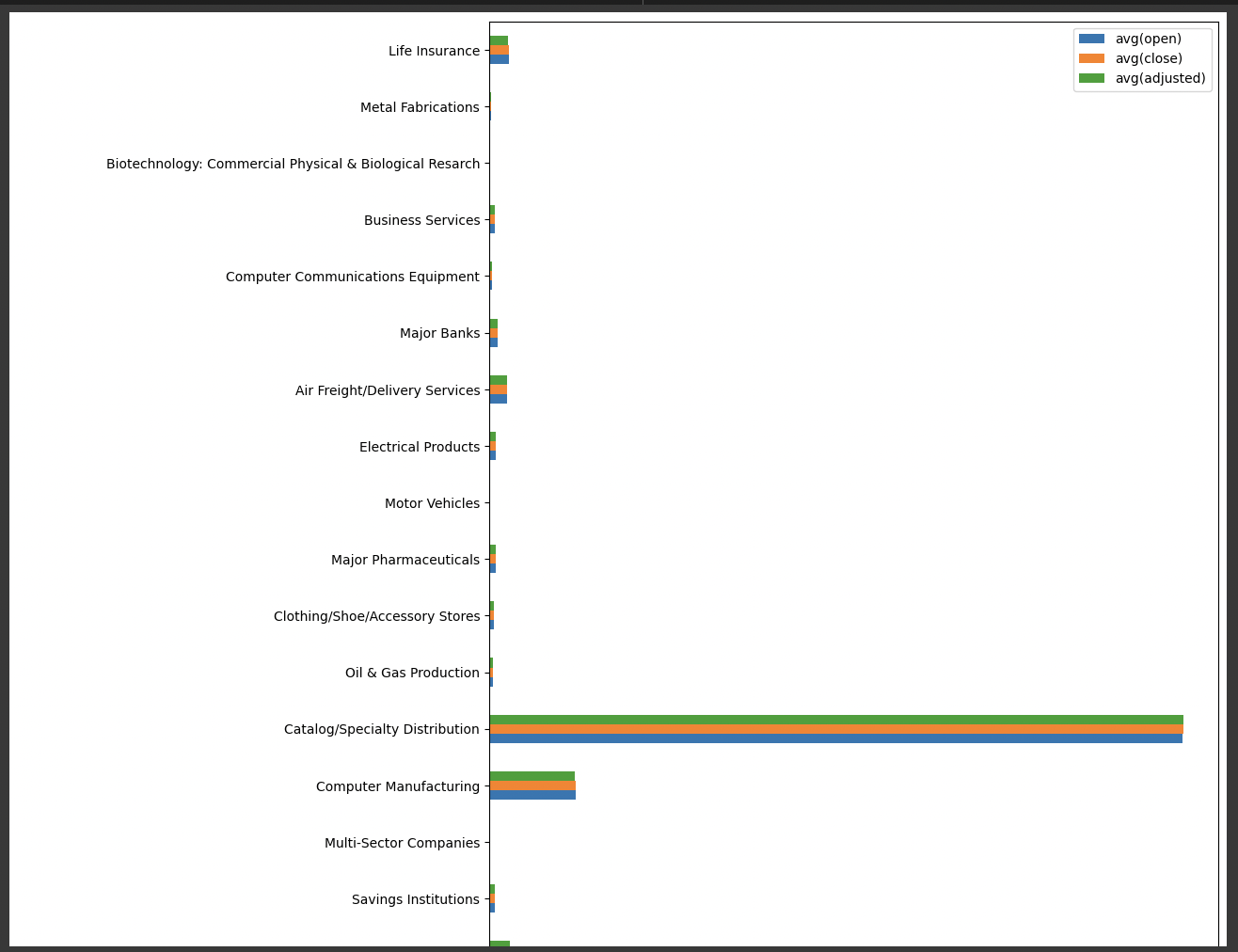
****

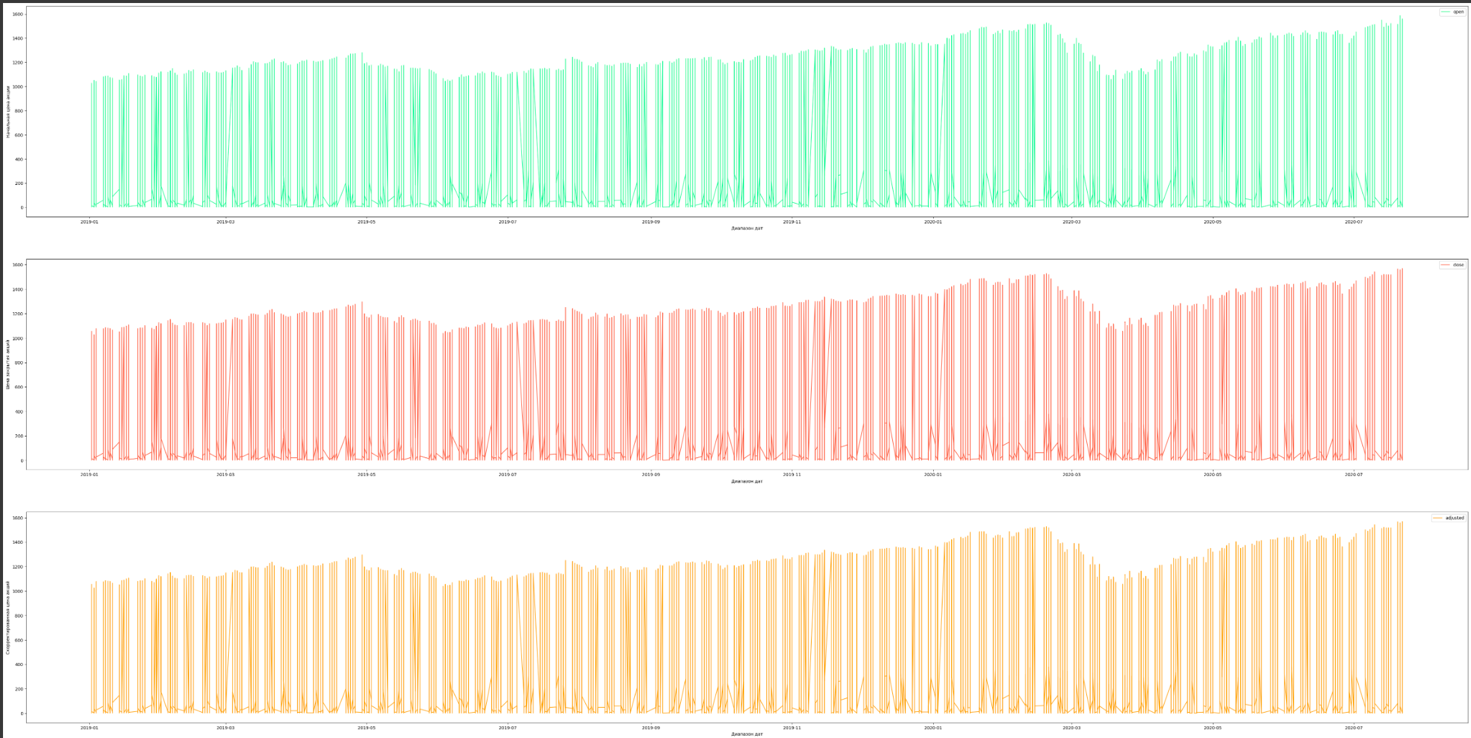
****

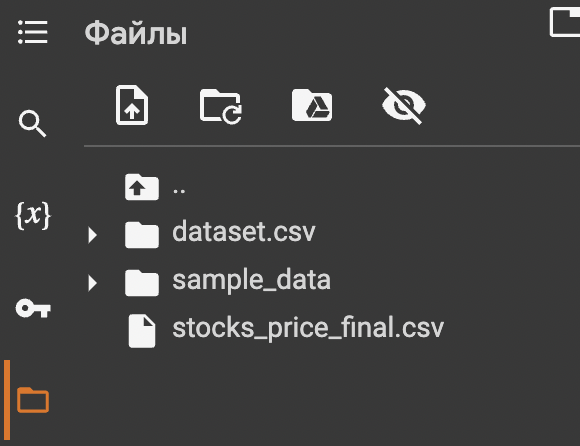
****

****

****

****

****

****

**2 часть – Zeppelin**

import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder().master("local[\*]").appName("PySpark\_Tutorial").getOrCreate()

=======

val df = spark.read.format("csv").option("header", "true").load("/home/av/Downloads/stocks\_price\_final.csv")

df.show()

=======

val data = df.limit(5)

data.show()

=======

data.write.csv("/home/av/Downloads/data.csv")

data.select("date", "open", "close", "adjusted").write.json("/home/av/Downloads/data.json")

=======

data.printSchema()

=======

import org.apache.spark.sql.types.\_

val data\_schema = Array(

StructField("\_c0", IntegerType, true),

StructField("symbol", StringType, true),

StructField("data", DateType, true),

StructField("open", DoubleType, true),

StructField("high", DoubleType, true),

StructField("low", DoubleType, true),

StructField("close", DoubleType, true),

StructField("volume", IntegerType, true),

StructField("adjusted", DoubleType, true),

StructField("market.cap", StringType, true),

StructField("sector", StringType, true),

StructField("industry", StringType, true),

StructField("exchange", StringType, true)

)

========

val final\_struc =StructType(fields = data\_schema )

val df = spark.read.format("csv").option("header", "true").schema(final\_struc ).load("/home/av/Downloads/stocks\_price\_final.csv")

df.printSchema()

========

val tst\_josn = spark.read.json("/home/av/Downloads/data.json")

tst\_josn.show()

=========

data.schema

========

data.dtypes

========

data.take(5)

========

data.first()

========

data.first()

========

data.describe().show()

========

data.columns

========

data.count()

========

data.distinct().count()

========

import org.apache.spark.sql.Row

import org.apache.spark.sql.functions.lit

val data\_new = data.withColumn("test\_new", lit("test")) // lit - создает столбец test\_new и заполняет его значениями "test"

data\_new.show(5)

========

data\_new.withColumnRenamed("test\_new","test\_new\_1").show()

========

data\_new.drop("test\_new").show()

========

val data\_null = data.withColumn("close", when(col("close")===62,null).otherwise(col("close")))

========

val data\_null\_new=data\_null.withColumnRenamed("market.cap","market\_cap")

========

data\_null\_new.na.drop().show()

========

// # Замена отсутствующих значений средним

import org.apache.spark.sql.functions.mean

========

data.select("sector").show()

========

data.select("open", "close","adjusted").show()

========

data.filter( (col("open") >= 54 )).show()