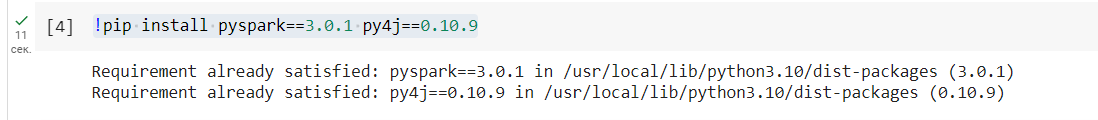
!pip install pyspark==3.0.1 py4j==0.10.9



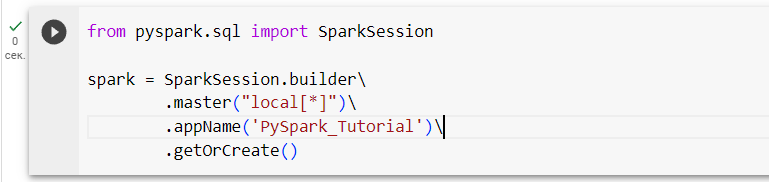
from pyspark.sql import SparkSession

spark = SparkSession.builder\

        .master("local[\*]")\

        .appName('PySpark\_Tutorial')\

        .getOrCreate()



data = spark.read.csv(

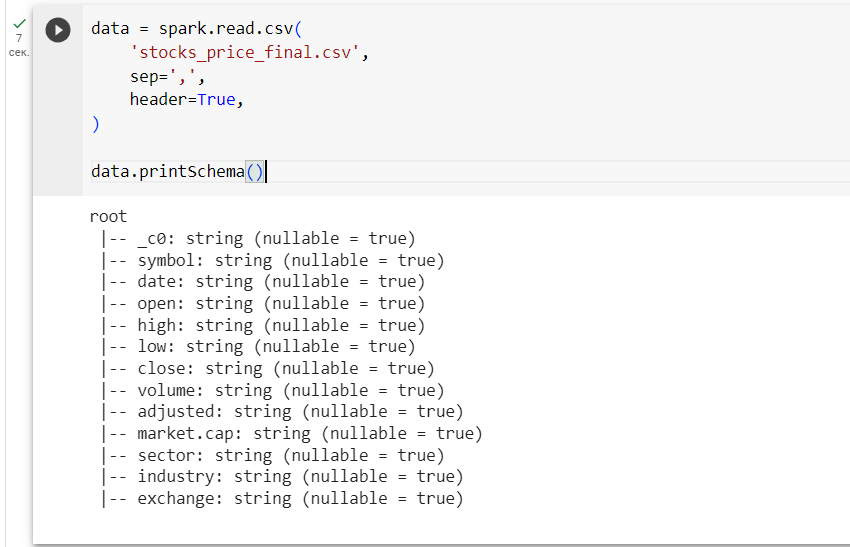
    'stocks\_price\_final.csv',

    sep=',',

    header=True,

)

data.printSchema()



from pyspark.sql.types import \*

data\_schema = [

               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),

            ]

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

data = spark.read.csv(

    'stocks\_price\_final.csv',

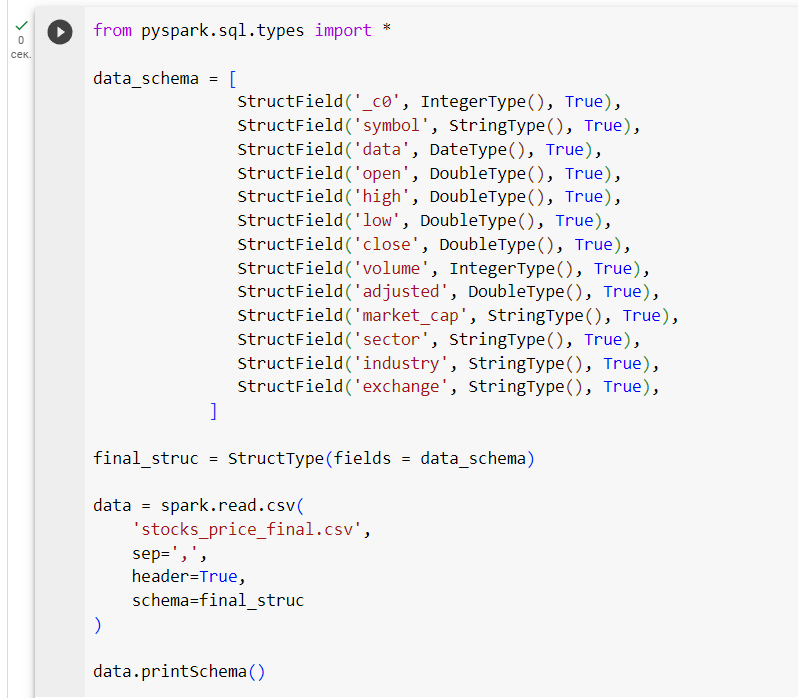
    sep=',',

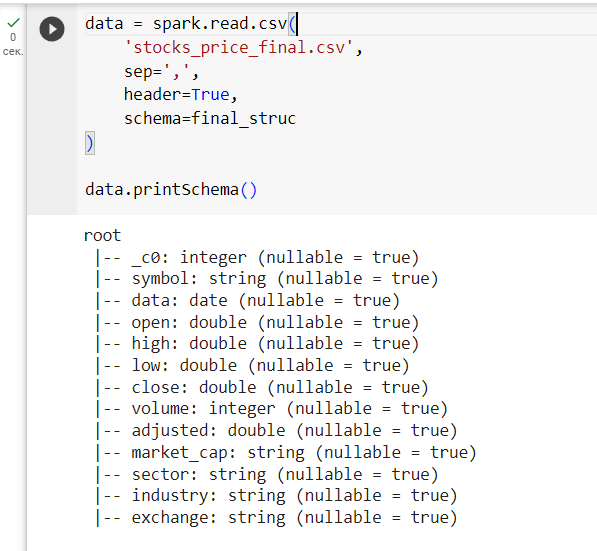
    header=True,

    schema=final\_struc

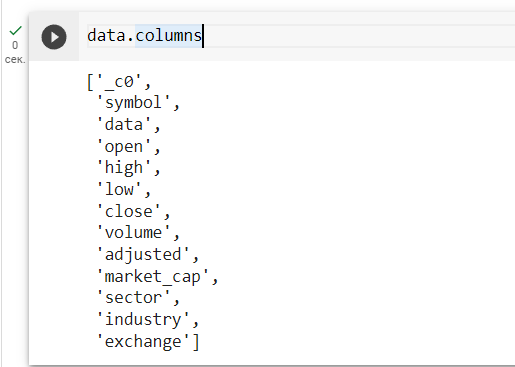
)

data.printSchema()



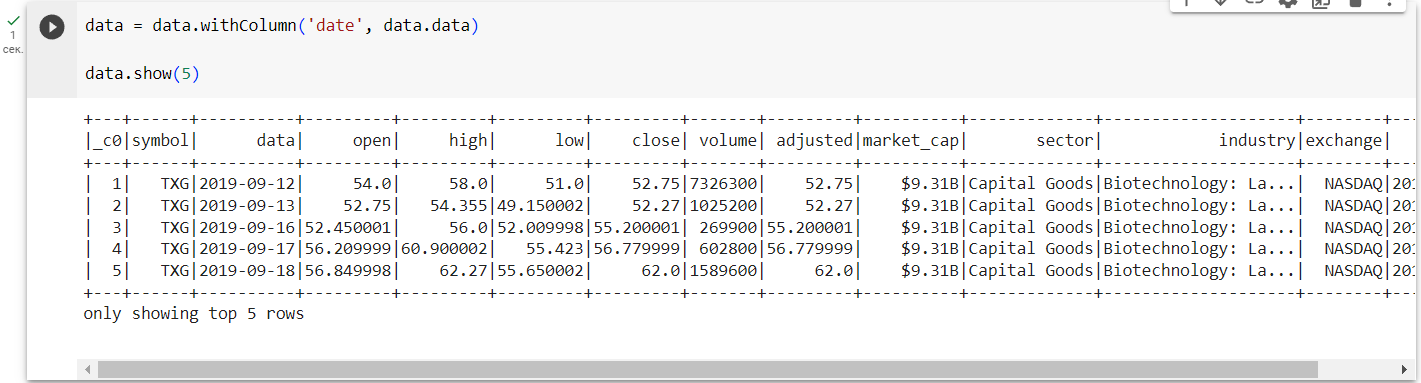


data.columns

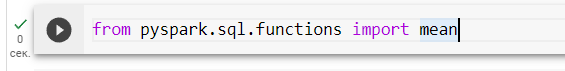


data = data.withColumn('date', data.data)

data.show(5)



from pyspark.sql.functions import mean



# Удаление строк с пропущенными значениями

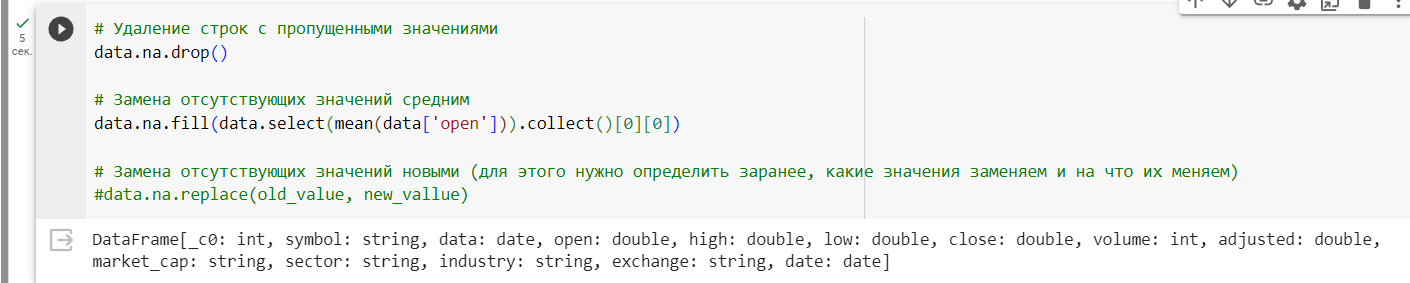
data.na.drop()

# Замена отсутствующих значений средним

data.na.fill(data.select(mean(data['open'])).collect()[0][0])

# Замена отсутствующих значений новыми (для этого нужно определить заранее, какие значения заменяем и на что их меняем)

#data.na.replace(old\_value, new\_vallue)

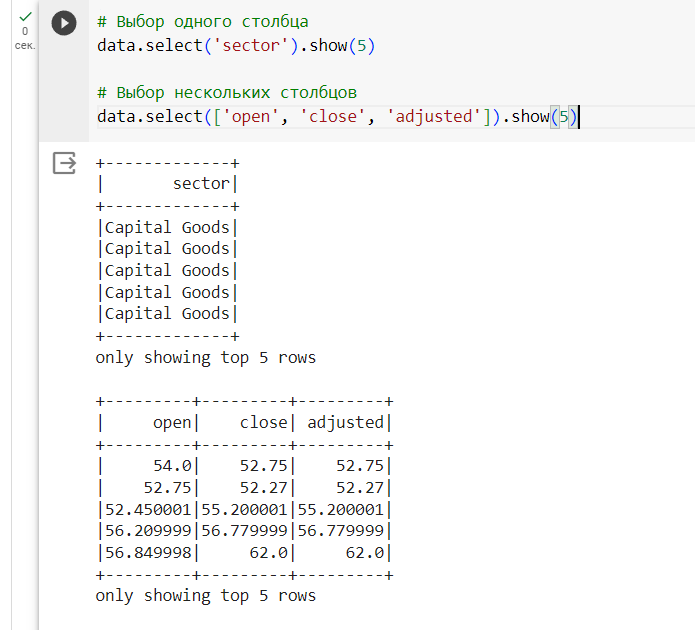


# Выбор одного столбца

data.select('sector').show(5)

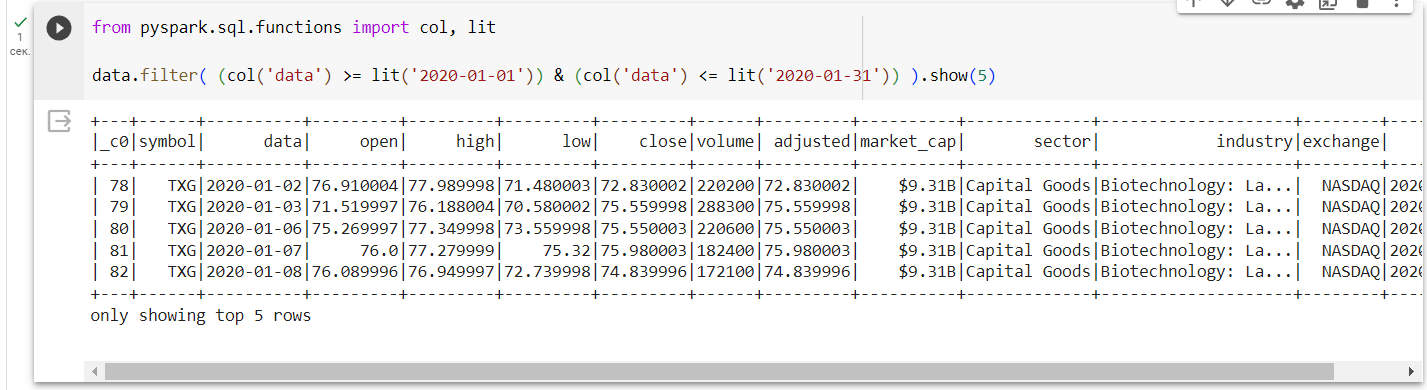
# Выбор нескольких столбцов

data.select(['open', 'close', 'adjusted']).show(5)



from pyspark.sql.functions import col, lit

data.filter( (col('data') >= lit('2020-01-01')) & (col('data') <= lit('2020-01-31')) ).show(5)

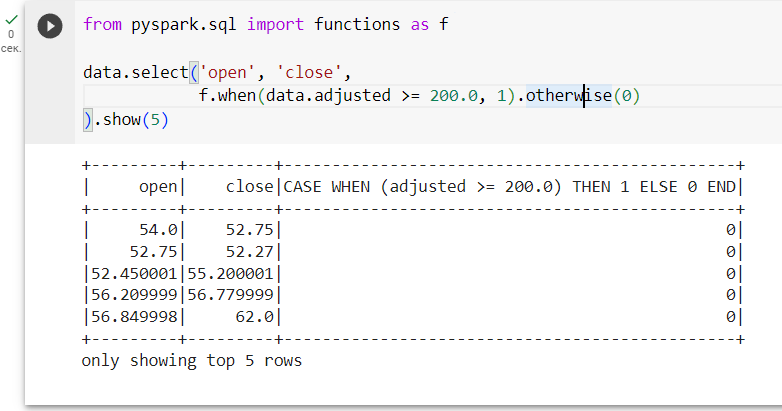


from pyspark.sql import functions as f

data.select('open', 'close',

            f.when(data.adjusted >= 200.0, 1).otherwise(0)

).show(5)

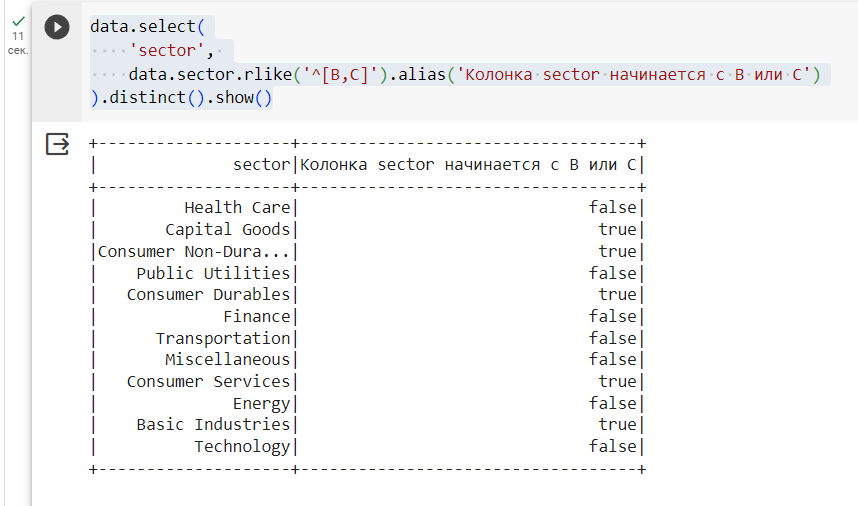


data.select(

    'sector',

    data.sector.rlike('^[B,C]').alias('Колонка sector начинается с B или C')

).distinct().show()

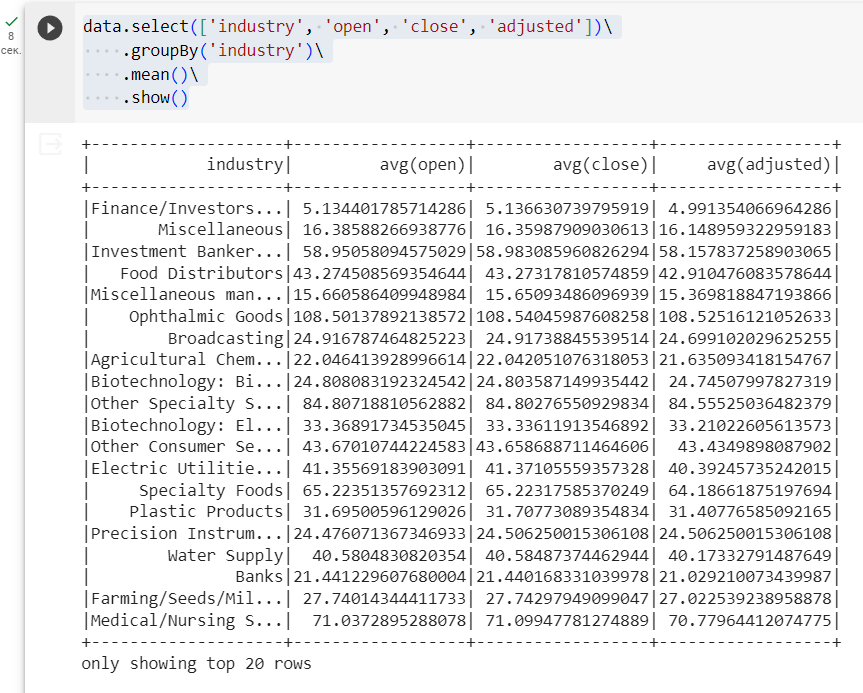


data.select(['industry', 'open', 'close', 'adjusted'])\

    .groupBy('industry')\

    .mean()\

    .show()



from pyspark.sql import functions as f

data.filter((col('data') >= lit('2019-01-02')) & (col('data') <= lit('2020-01-31')))\

    .groupBy("sector") \

    .agg(f.min("data").alias("С"),

         f.max("data").alias("По"),

         f.min("open").alias("Минимум при открытии"),

         f.max("open").alias("Максимум при открытии"),

         f.avg("open").alias("Среднее в open"),

         f.min("close").alias("Минимум при закрытии"),

         f.max("close").alias("Максимум при закрытии"),

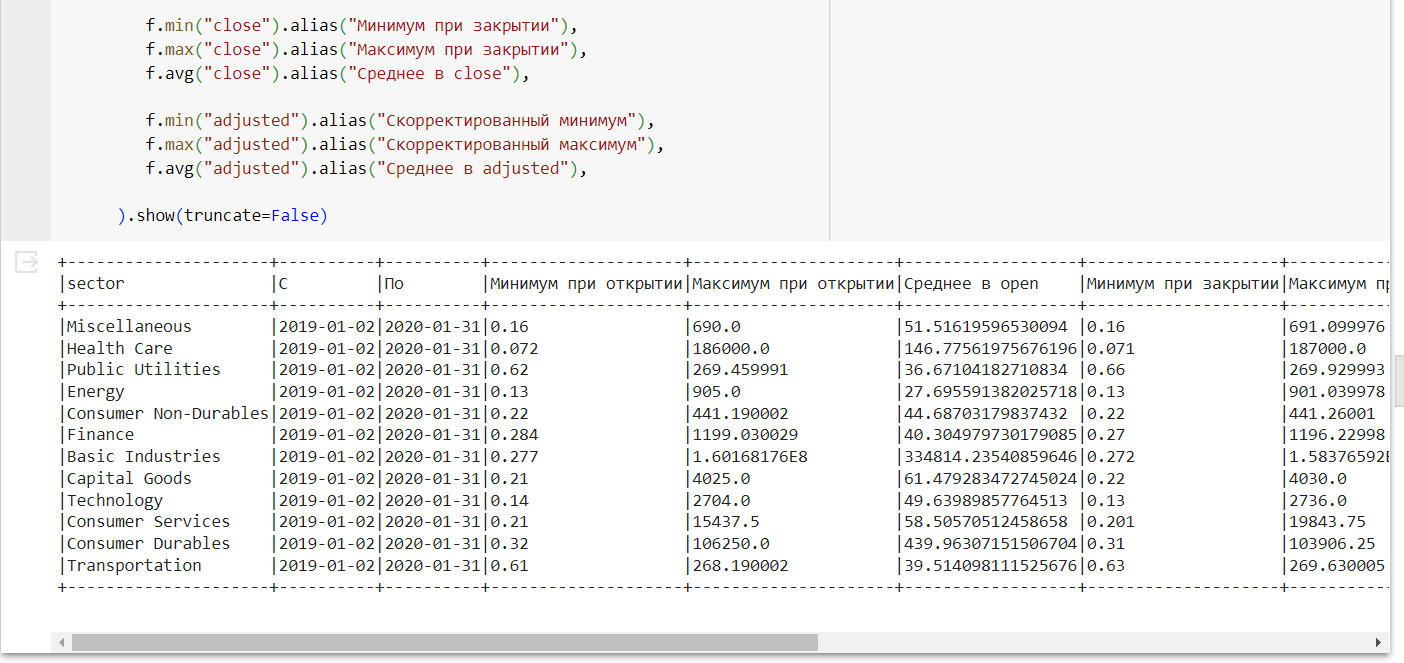
         f.avg("close").alias("Среднее в close"),

         f.min("adjusted").alias("Скорректированный минимум"),

         f.max("adjusted").alias("Скорректированный максимум"),

         f.avg("adjusted").alias("Среднее в adjusted"),

      ).show(truncate=False)



from matplotlib import pyplot as plt

sec\_df =  data.select(['sector',

                       'open',

                       'close',

                       'adjusted']

                     )\

                     .groupBy('sector')\

                     .mean()\

                     .toPandas()

ind = list(range(12))

ind.pop(6)

sec\_df.iloc[ind ,:].plot(kind='bar', x='sector', y=sec\_df.columns.tolist()[1:],

                         figsize=(12, 6), ylabel='Stock Price', xlabel='Sector')

plt.show()

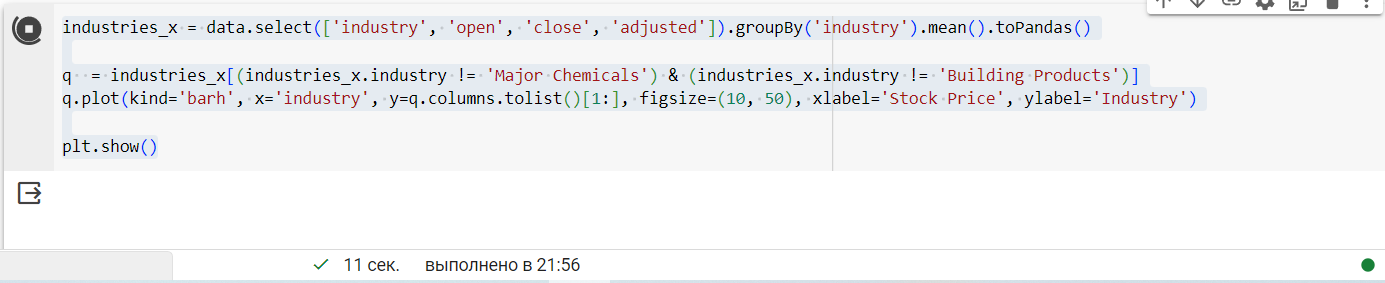


industries\_x = data.select(['industry', 'open', 'close', 'adjusted']).groupBy('industry').mean().toPandas()

q  = industries\_x[(industries\_x.industry != 'Major Chemicals') & (industries\_x.industry != 'Building Products')]

q.plot(kind='barh', x='industry', y=q.columns.tolist()[1:], figsize=(10, 50), xlabel='Stock Price', ylabel='Industry')

plt.show()



from pyspark.sql.functions import col

tech = data.where(col('sector') == 'Technology')\

           .select('data', 'open', 'close', 'adjusted')

fig, axes = plt.subplots(nrows=3, ncols=1, figsize =(60, 30))

tech.toPandas().plot(kind='line', x='data', y='open',

                     xlabel='Диапазон дат', ylabel='Начальная цена акции',

                     ax=axes[0], color='mediumspringgreen')

tech.toPandas().plot(kind='line', x='data', y='close',

                     xlabel='Диапазон дат', ylabel='Цена закрытия акций',

                     ax=axes[1], color='tomato')

tech.toPandas().plot(kind='line', x='data', y='adjusted',

                     xlabel='Диапазон дат', ylabel='Скорректированная цена акций',

                     ax=axes[2], color='orange')

plt.show()



# CSV

data.write.csv('dataset.csv')

# JSON

data.write.save('dataset.json', format='json')

# Parquet

data.write.save('dataset.parquet', format='parquet')

# Запись выбранных данных в различные форматы файлов

# CSV

data.select(['data', 'open', 'close', 'adjusted'])\

    .write.csv('dataset.csv')

# JSON

data.select(['data', 'open', 'close', 'adjusted'])\

    .write.save('dataset.json', format='json')

# Parquet

data.select(['data', 'open', 'close', 'adjusted'])\

    .write.save('dataset.parquet', format='parquet')

