SDE Assignment 3

Name: Sonia Goyal
Roll No: M21AIE258

Hardware

Disk storage: 500 GB Flash storage
Machine: MacOS Monterey version 12.5.1
Processor 2.6 GHz 6-Core Intel Core i7
System Memory: 16 GB 2667 MHz DDR4
Graphics: Graphics Intel UHD Graphics 630 1536 MB
Cache: L1: 32768

Software

OS: MacBook Pro (16-inch, 2019)
Python Version: 3.9
Pandas Version:
Spark Version:

YouTube Link: https://www.youtube.com/watch?v=Byp3UADHSgI
GitHub Link: https://github.com/sonia-goyal/benchmarks/tree/main

TASKS:

a) **Dataset:** The dataset for benchmarking is "Stock Market Data - Nifty 50 Stocks (1 min) data" which has been downloaded from Kaggle.
The dataset contains historical daily prices for Nifty 100 stocks and indices currently trading on the Indian Stock Market.
   - Data samples are of 5-minute intervals and the availability of data is from Jan 2015 to Feb 2022.
   - The dataset has OHLCV (Open, High, Low, Close, and Volume) data, and 55 more such technical indicators.
   - The data size is around 33 GB and divided in total of 51 file where each file approximately 630 MB.

b) I have tried data loading with different data sizes using both spark and pandas. As shown in Figure 1 spark is performing much better than pandas. For small data size the loading time is almost same but as the data size is growing the taken by pandas is increasing exponentially but spark is taking almost same time. For exact data please refer to Table 1.1 for pandas metadata and Table 1.2 for spark metadata.

SNo.	data_size(GB)	memory_used	exec_time	number_of_rows
0	3.093414	2.241718	42.50005	3259414
1	4.955037	1.59565	71.93664	5215169
2	9.913227	2.015606	187.6712	10436261
3	14.86589	1.667667	474.1745	15657358
4	19.81792	2.112713	845.1835	20872659

Table 1.1 – Pandas stats (exec times are in sec)

SNo.	data_size(GB)	memory_used	exec_time	number_of_rows
0	3.09341377	0.070014954	6.517884254	3259419
1	4.95503674	0.070030212	1.370390177	5215177
2	9.91322701	0.070091248	2.045832872	10436277
3	14.8658916	0.070095062	2.686369658	15657382
4	19.8179219	0.070114136	3.583834171	20872691

Table 1.2 – Spark stats (exec times are in sec)

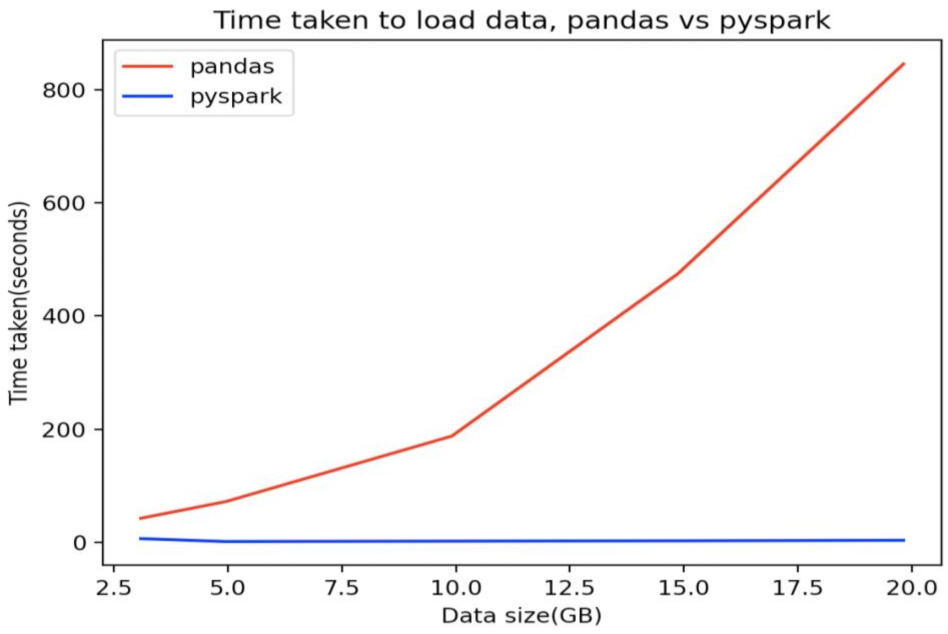

Figure 1

c) **Benchmarking:**

1. **Performance:**
   Here 3 queries are performed to check performance and data size is kept at max 20 GB. Table 3.1 and Table 3.2 respectively shows the time taken pandas and spark to execute the below queries

   SQL 1: select max(close) from data
   SQL 2: select count(distinct volume) from data
   SQL 3: select sum(high) from data group by date

Sno.	**data_size(GB)**	**max_query_exec_time**	**count_query_exec_time**	**group_query_exec_time**
0	1.86034783	0.01421523	0.02866006	2.20010996
1	4.336572	0.04470205	0.09147334	6.51582122
2	6.18931502	0.11151123	0.1406281	17.912967
3	8.05540706	0.15490413	0.27962804	28.2730529
4	9.91322701	0.17139792	0.27673602	40.2435093
5	14.8658916	0.39513874	0.54161525	79.988287
6	20.4424677	0.64294791	0.79739189	113.199371

Table 3.1 – Pandas stats (exec times are in sec)

Sno.	**data_size(GB)**	**max_query_exec_time**	**count_query_exec_time**	**group_query_exec_time**
0	1.86034783	3.27840424	2.91969013	6.76744103
1	4.336572	2.6944797	3.34207511	7.85592604
2	6.18931502	3.45613122	4.40558171	8.94303775
3	8.05540706	6.06586599	6.69820714	12.0819032
4	9.91322701	6.16575098	8.15435791	14.16731
5	14.8658916	9.22923899	11.3875952	20.9417582
6	20.4424677	13.2934787	16.500205	27.646225

Table 3.2 – Spark stats (exec times are in sec)

SQL1: For the max query we can see that pyspark is taking more time to execute than pandas.

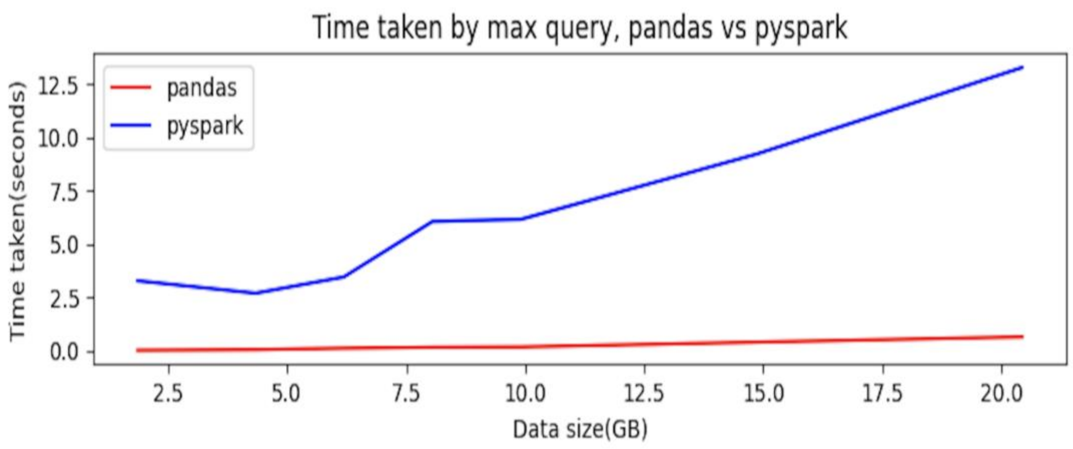

Figure 2

SQL2: For counting distinct in a column pyspark is taking more time than pandas.

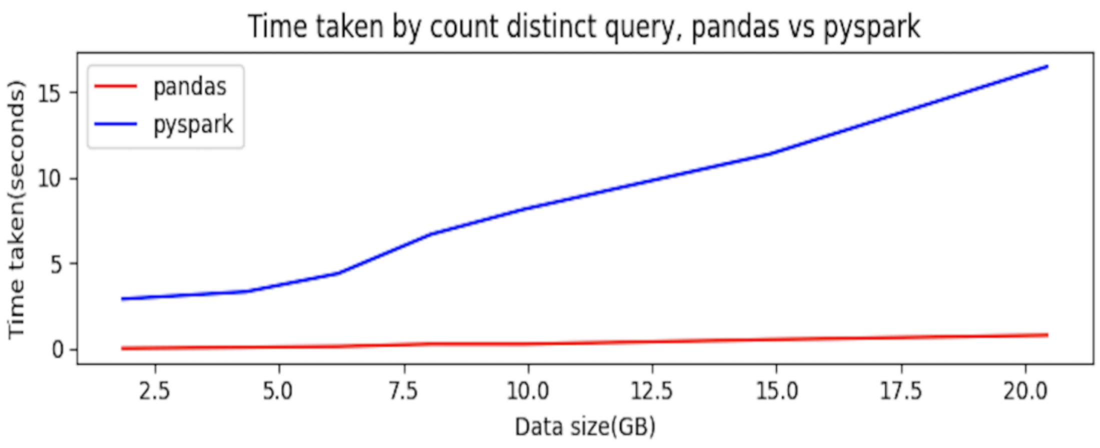

Figure 3

SQL3: The third query is doing a group by and then performing an aggregate operation. In this query spark is performing better than pandas. In my opinion this query needs to get the data in memory and then perform group by operation hence we can see the benefit of spark here. Pandas has been optimized to perform some intrinsic operation like max and count query that's why it is performing better in those. If we are to perform complex operations or join queries I am sure spark will perform much better than pandas

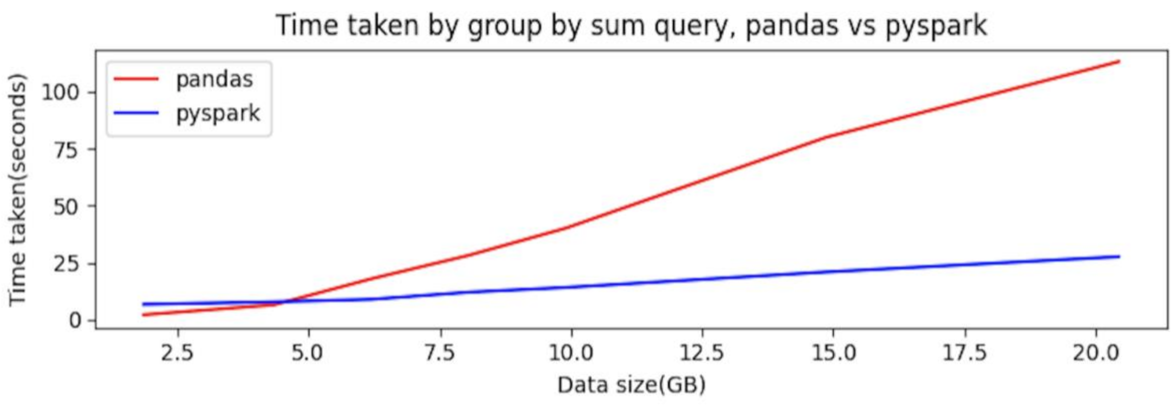

Figure 4

2. **Scalability:** This parameter shows the pandas scalability power. Pandas was able to handle 30 GB of data and it crashed out of memory at 35 GB. I have replaces the execution time for “out of memory” datapoint with 0 to showcase it in the graph.

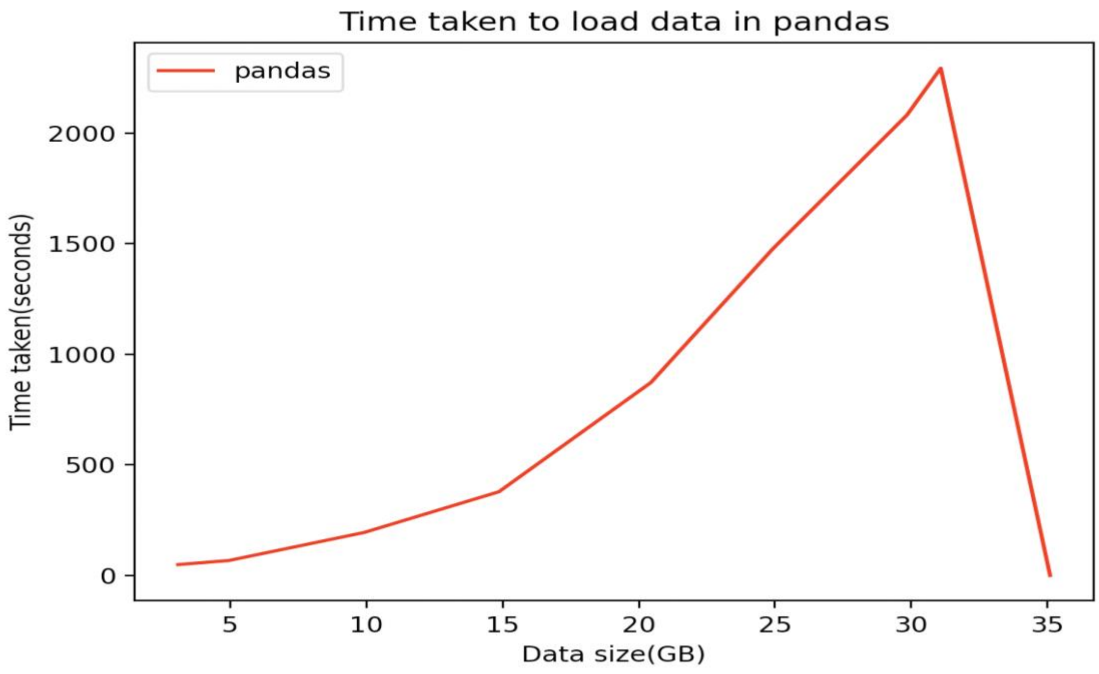

Figure 5

Sno	data_size(GB)	memory_used(GB)	exec_time (sec)	number_of_rows
0	3.09341377	2.20763397	49.0851409	3259414
1	4.95503674	1.65602112	67.5956991	5215169
2	9.91322701	2.68391037	194.110595	10436261
3	14.8658916	1.61630249	378.438765	15657358
4	20.4424677	1.91514587	872.211073	21524572
5	24.9052939	2.32999802	1475.32522	26235597
6	29.8422313	2.72985458	2082.08851	31450906
7	31.0791579	2.85693741	2293.56616	32754739
8	35.0917906	2.85693741	0	32754739

Table 3.3 Pandas scalability

3. **Data Sorting:** Sort the ‘close’ column in pandas. The time taken to sort increases with data size.

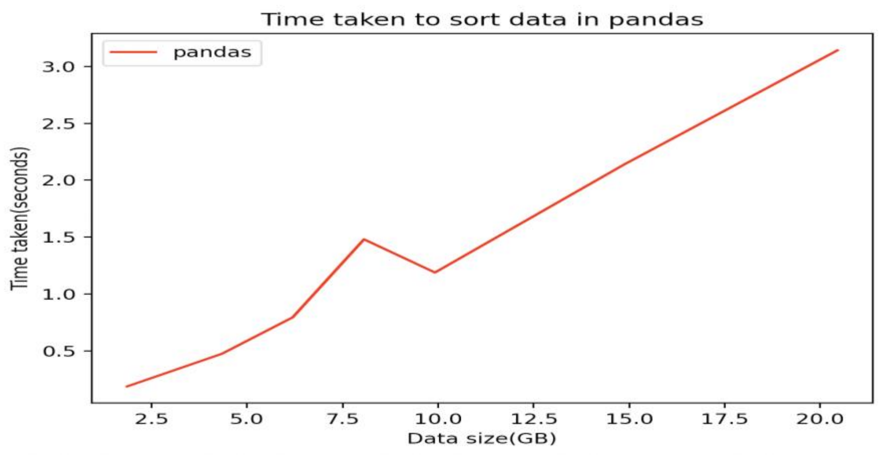

Figure 6

4. **Sum query:**

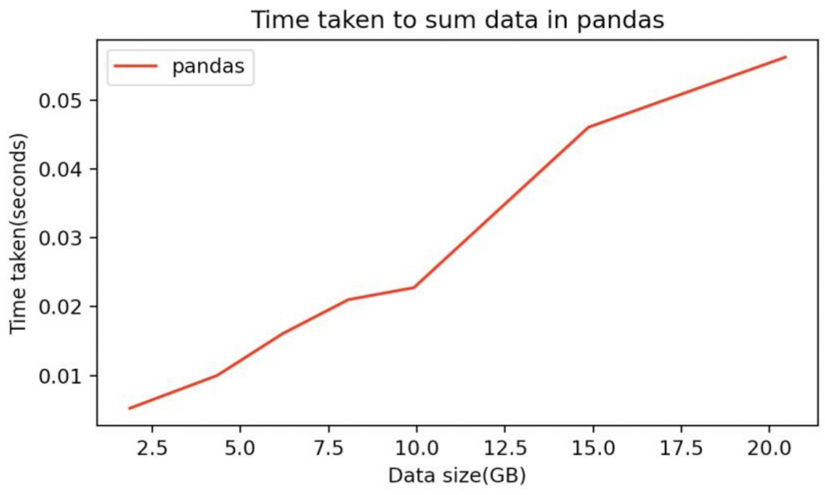

Figure 7

5. **Parallelism :** True parallelism is not possible in Pandas. It can give look alike of parallelism using multi-threading. The main task of Spark is distributed computing.