Single or Cluster/Distributed		S	S	C	S	C	S	C	
Program Type		Native Python	Pyspark RDD	Pyspark Rdd	Pyspark Dataframe	Pyspark Dataframe	CockroachDB SQL	CockroachDB SQL (6 nodes)	Distr. Python (6 Nodes)
Units of Shakespere	Size Of Processed File (MB)								
1	5	0.213	3.2119	5.456666667	1.21	2.19	1.55	0.797	0.09
16	81	1.517	15.04	10.08	6.29	3.72	11.733	10.761	0.58
64	325	5.69	58.807	16	17.75	4.85	46.189	43.798	2.08
256	1300	24.17	233.554	64.12	87.45	17.614	308.452	201.897	7.99
1024	5203	94.217	930	236	298	67.64	1096.809	812.267	28.26
2048	10406	196.075	1874	432	510.398	120.74	fail- timeout (more than 1 hour)	1628.146	
Notes			much much slower but less memory. no use of disk	1h5 and 5 minutes for 5 test runs	started having to write to disk at 256. Fail at 2048 due to not enough disk space (30 GB allocated, probably like 13 GB free)	20 minutes for 5 test runs		Run-time isn't as dependent on nodes until it gets to larger datsets. Picture diagrams to show how work is being divided among nodes.	

Node vs Cluster Performance Difference	(Single/Cluster)				
Units of Shakespere	Pyspark DF	Pyspark RDD	SQL	Native Python	
1	55.25%	58.86%	194.48%	236.67%	<--
16	169.09%	149.21%	109.03%	261.55%	
64	365.98%	367.54%	105.46%	273.56%	
256	496.48%	364.25%	152.78%	302.50%	
1024	440.57%	394.07%	135.03%	333.39%	
2048	422.72%	433.80%	N/A	N/A	
Notes	In both cases I see a turning point it the cluster getting faster somewhere between 1 and 16, and performance improvements seem to peek at 5x for the DF and 4.3x for the RDD, even though the cluster has 6 worker nodes.				
	Further questions: do more experiments with quantities between 1 and 16 to determine the "turning point" and 16 and 64 to get the increase smoothed out.				
	Further questions: does the size of the cluster change where these break even points are?