

Relazione Big Data

Progetto I: Daily Historical Stock Prices

Baldazzi Teodoro Sferrazza Stefano
teo.baldazzi@stud.uniroma3.it ste.sferrazza@stud.uniroma3.it

Maggio 2020

1 Introduzione

La seguente relazione illustra e documenta il lavoro di analisi, progettazione e realizzazione svolto come primo progetto del corso "Big Data" nell'anno accademico 2019/2020.

Di seguito i dettagli sui membri del gruppo:

- Baldazzi Teodoro, matr. 492471;
- Sferrazza Stefano, matr. 499264.

Il repository con il progetto svolto, le pseudocodifiche e i risultati può essere trovato al seguente link:

<https://github.com/StefanoSferrazza/BigData>, nonché allegato a questa relazione in formato compresso.

La relazione è strutturata come segue:

- la Sezione 2 introduce il dataset e le interpretazioni adottate in fase di analisi dei vari esercizi;
- la Sezione 3 fornisce una breve panoramica della struttura del progetto, per semplificarne l'esplorazione, nonché delle versioni degli esercizi sviluppate nelle varie tecnologie;
- la Sezione 4 illustra una delle possibili pseudo-codifiche degli esercizi in Hadoop e Spark, e la corrispondente implementazione in Hive;
- la Sezione 5 presenta e descrive in dettaglio le scelte implementative delle distinte versioni degli esercizi, nelle tre tecnologie;
- la Sezione 6 illustra i risultati delle esecuzioni nelle tre tecnologie, mostrando l'equivalenza dei record e del formato in output.
- la Sezione 7 descrive in dettaglio gli insiemi di test, con i corrispettivi grafici e tabelle, eseguiti in locale e su cluster, per ogni esercizio e in ciascuna tecnologia, con dimensioni crescenti dell'input.

2 Analisi Dataset ed Esercizi

In questa Sezione è presentato il dataset *Daily Historical Stock Prices* fornito, con particolare riferimento alle proprietà osservate in fase di analisi dei dati e un primo riferimento alle soluzioni conseguentemente adottate per favorire la corretta esecuzione degli esercizi, illustrate in maggior dettaglio nelle Sezioni successive. Inoltre sono sinteticamente descritti tali esercizi e le interpretazioni assunte per le fasi successive del lavoro. Si noti che, nel seguito della trattazione, i termini Esercizi e Job saranno impiegati come sinonimi.

Analisi Dataset

Il dataset comprende due file .CSV: *historical_stock_prices* (di seguito HSP) e *historical_stocks* (di seguito HS). Relativamente alle caratteristiche osservate sui dati in essi presenti, si osserva:

- entrambi i file impiegano la virgola (",") come carattere separatore dei vari campi;
- HSP è il file di maggiore dimensione (pari a circa 2 GB) e descrive l'andamento giornaliero di un insieme di azioni in borsa. I record sono prevalentemente puliti, privi dunque di caratteri speciali o valori non corrispondenti ai tipi previsti per i corrispondenti campi;
- HS ha dimensione minore (pari a circa 450 KB) e fornisce informazioni su ciascuna azione, quali l'azienda e il settore di appartenenza; si noti che ciascun settore comprenda una o più aziende e ciascuna azienda sia associata a una o più azioni. Il file presenta diversi record con dati sporchi o assenti, per i quali dunque si ritiene necessaria una fase di pre-processamento: esempio tipico è la presenza di un valore nullo o vuoto in corrispondenza dei campi "settore" o "azienda". Si è inoltre osservata la presenza di una virgola nel nome di alcune aziende, nello specifico quelle terminanti per ", Inc." e sempre tra virgolette. Come vedremo, sono state richieste opportune strategie di parsing e filtraggio per mantenere i dati corretti e completi.

Analisi Esercizi

Le specifiche progettuali richiedono lo sviluppo di 3 esercizi, ciascuno corrispondente a un job MapReduce (op-pure a una chain di job, come vedremo nelle successive Sezioni) nelle tecnologie Hadoop, Hive e Spark. Per tutti gli esercizi, ogni richiesta associata a un prezzo fa riferimento ai prezzi di chiusura "close", mentre "open", "adj_close", "lowThe" e "highThe" non sono mai impiegati. Con riferimento ai singoli esercizi richiesti e alle informazioni sul dataset sopra fornite, si possono fare alcune osservazioni.

Primo Esercizio. Questo esercizio richiede informazioni ottenibili dal solo HSP e le specifiche fornite sono sufficienti per il suo svolgimento: variazione di quotazione e volume medio sono definiti come descritto e prezzo minimo e massimo si riferiscono a close minimo e massimo dell'intervallo per ciascun ticker.

Secondo Esercizio. Questo esercizio richiede di considerare sia HSP che HS. Dall'analisi si possono dedurre distinte interpretazioni per calcolare volume annuale, variazione annuale media e quotazione giornaliera media:

- come media di tutti i ticker del settore, trascurando dunque l'azienda di appartenenza di ciascun ticker;
- calcolando prima la media dei ticker di ciascuna azienda e poi la media delle aziende di ciascun settore;

Entrambe le versioni sono state oggetto di sviluppo e testing in questo progetto. Inoltre, per quanto concerne il calcolo della variazione annuale di un'azienda, si può considerare una distinta interpretazione rispetto alla media delle variazioni annuali dei ticker. Difatti, per variazione annuale di un'azienda si intende la differenza percentuale tra quotazione di fine anno e quotazione di inizio anno per l'azienda; inoltre la quotazione di un'azienda in un dato istante è definita come la somma delle quotazioni delle azioni ad essa appartenenti e attive in quel dato istante. Pertanto, la definizione impiegata in questo progetto per variazione annuale di un'azienda è la differenza percentuale tra la sommatoria dei close delle azioni alla fine dell'anno e la sommatoria dei close delle azioni all'inizio dell'anno.

Terzo Esercizio. Questo esercizio richiede di considerare sia HSP che HS. Il calcolo del trend come variazione di quotazione delle aziende è stato effettuato facendo riferimento alle due interpretazioni di cui sopra. Inoltre è stato considerato lo scenario facoltativo correlato (non presente nel progetto originale ma proposto dal docente via mail) in cui si voglia associare non solo aziende con medesimo trend, ma anche quelle che presentano similarità, definita a partire dalla distanza tra trend e calcolata mediante distanza euclidea.

3 Presentazione Progetto

In questa Sezione è illustrata la struttura del progetto allegato, per facilitarne l'esplorazione e la comprensione, con particolare riferimento alle versioni degli esercizi sviluppate nelle varie tecnologie e su cui sono stati effettuati test e comparazioni delle performance.

Struttura progetto

Il progetto è articolato come segue:

- in *src/main/java* sono presenti le implementazioni degli esercizi in Hadoop, Hive e Spark, nonché le classi di supporto *Utilities* per il pre-processing e il post-processing dei record;
- in *pseudo* sono presenti le pseudocodifiche degli esercizi in Hadoop e Spark, mentre le corrispondenti implementazioni in Hive sono nella source folder di cui sopra;
- in *resultsFirst10* sono presenti i primi 10 record degli output dei job in Hadoop, Hive e Spark;
- in *resultsComplete* sono presenti i risultati completi delle versioni principali degli esercizi.

Versioni Esercizi

In questa attività progettuale sono state sviluppate distinte versioni dei vari esercizi. Tali versioni si distinguono non solo per le interpretazioni delle varie richieste nelle specifiche progettuali, come introdotto in Sezione 2, ma anche per le scelte adottate nell'implementazione ai fini di presentare comparazioni delle performance interessanti e di una certa rilevanza. Di seguito sono brevemente presentate rispetto alla tecnologia adottata: nella Sezione 5 sono analizzate con maggior dettaglio.

Hadoop. Nella tecnologia Hadoop sono state sviluppate le seguenti versioni degli esercizi.

- esercizio 1: in questa tecnologia presenta una singola versione;
- esercizio 2: presenta una versione *basic*, in cui non si considera l'azienda e i campi richiesti sono calcolati ragionando in termini di ticker e settore, una versione *complex*, che al contrario considera anche l'azienda, e una versione *complex con combiner* per l'analisi degli effetti del combiner sulle prestazioni;
- esercizio 3: in questa tecnologia è presente una singola versione.

Hive. Nella tecnologia Hive sono state sviluppate le seguenti versioni degli esercizi.

- esercizio 1: presenta una versione *v1*, caratterizzata da una singola temporary table, una versione *v2*, che invece presenta un maggior numero di temporary tables, e una versione *v3*, analoga a *v1* fatta eccezione per il formato dei record in output;
- esercizio 2: presenta una versione *basic*, in cui non si considera l'azienda e i campi richiesti sono calcolati ragionando in termini di ticker e settore, ed una versione *complex*, che al contrario considera anche l'azienda;
- esercizio 3: presenta due versioni *v1* e *v2*, che si distinguono per la definizione e il calcolo della variazione annuale media, come introdotto in Sezione 2, una versione *v3* analoga a *v1* fatta eccezione per il formato dei record in output, e infine una versione *facoltativa* che implementa la similarità tra aziende per le quali non ne esistano altre con trend equivalente.

Spark. Nella tecnologia Spark sono state sviluppate le seguenti versioni dei job.

- esercizio 1: in questa tecnologia è presente una singola versione;
- esercizio 2: presenta una versione *singleRow* ed una versione *distributedRows*. Le funzioni utilizzate nelle due versioni sono identiche, ma si ha una differenza nella distribuzione delle funzioni utilizzate su più strutture intermedie al fine di osservare se questo comportasse delle differenze in termini di prestazioni.
- esercizio 3: in questa tecnologia è presente una singola versione.

4 Pseudocodifica

In questa Sezione sono presentate le pseudocodifiche delle principali soluzioni per ciascun esercizio, nelle tecnologie Hadoop e Spark, e la corrispondente implementazione in Hive. Per ciascuna sono fornite pre-condizioni e post-condizioni. Si rimanda alla Sezione 5 per la dettagliata discussione sulle scelte implementative. Nella cartella *pseudo* del progetto sono presenti i file con le seguenti pseudocodifiche.

4.1 Hadoop

<p>HADOOP EX1</p> <p>INPUT:</p> <ul style="list-style-type: none">- path to document "historical_stock_prices.csv".- path for output result. <p>PRE-COND:</p> <ul style="list-style-type: none">- path to historical_stock_prices.csv needs to be valid.- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields: <table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- OPEN</td><td>Float</td><td>open price</td></tr><tr><td>- CLOSE</td><td>Float</td><td>close price</td></tr><tr><td>- ADJ_CLOSE</td><td>Float</td><td>adjusted close price</td></tr><tr><td>- LOW</td><td>Float</td><td>min price</td></tr><tr><td>- HIGH</td><td>Float</td><td>max price</td></tr><tr><td>- VOLUME</td><td>Long</td><td>number of transactions</td></tr><tr><td>- DATE</td><td>Date</td><td>date in format yyyy-mm-dd</td></tr></table> <ul style="list-style-type: none">- output path needs to be valid. <p>POST-COND:</p> <ul style="list-style-type: none">- output file needs to be a document, separated by commas, with 5 fields: <table><tr><td>- TICKER</td><td>String</td><td></td></tr><tr><td>- DELTA_QUOTATION_%</td><td>String</td><td>(Integer with %) percentage change rounded of ticker value in period 2008-2018</td></tr><tr><td>- MIN_CLOSE</td><td>Float</td><td>min "close" value reported in period 2008-2018</td></tr><tr><td>- MAX_CLOSE</td><td>Float</td><td>max "close" value reported in period 2008-2018</td></tr><tr><td>- AVG_VOLUME</td><td>Long</td><td>average "volume" in period 2008-2018</td></tr></table> <ul style="list-style-type: none">- document rows need to be sorted by descending DELTA_QUOTATION	- TICKER	String	unique share symbol	- OPEN	Float	open price	- CLOSE	Float	close price	- ADJ_CLOSE	Float	adjusted close price	- LOW	Float	min price	- HIGH	Float	max price	- VOLUME	Long	number of transactions	- DATE	Date	date in format yyyy-mm-dd	- TICKER	String		- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of ticker value in period 2008-2018	- MIN_CLOSE	Float	min "close" value reported in period 2008-2018	- MAX_CLOSE	Float	max "close" value reported in period 2008-2018	- AVG_VOLUME	Long	average "volume" in period 2008-2018	<pre>ex1Mapper(key, value) skip first row //header check_correctness_data(value) //correct types close,volume,date <- extractRelevantValues(value) //relevant values are "close","volume","date" key <- extractTicker(value) if (2008 <= date.year() && date.year() <= 2018): EMIT(ticker, (close, volume, date)) ex1Reducer(key, values) initialize firstClose, lastClose, firstDate, lastDate, minClose, maxClose, sumVolumes, numValues for each text in values: update firstClose, lastClose, firstDate, lastDate, minClose, maxClose, sumVolumes, numValues if(text.date < firstDate) firstDate = text.date firstClose = text.close if(text.date > lastDate) lastDate = text.date lastClose = text.close if(text.close < minClose) minClose = close if(text.close > maxClose) maxClose = close sumVolumes += text.volume numValues++ deltaQuotation = ((lastClose - firstClose) / firstClose) * 100 avgVolume = sumVolumes / numValues EMIT(ticker, (deltaQuotation,minClose,maxClose,avgVolume))</pre>
- TICKER	String	unique share symbol																																						
- OPEN	Float	open price																																						
- CLOSE	Float	close price																																						
- ADJ_CLOSE	Float	adjusted close price																																						
- LOW	Float	min price																																						
- HIGH	Float	max price																																						
- VOLUME	Long	number of transactions																																						
- DATE	Date	date in format yyyy-mm-dd																																						
- TICKER	String																																							
- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of ticker value in period 2008-2018																																						
- MIN_CLOSE	Float	min "close" value reported in period 2008-2018																																						
- MAX_CLOSE	Float	max "close" value reported in period 2008-2018																																						
- AVG_VOLUME	Long	average "volume" in period 2008-2018																																						

Figure 1: Hadoop Esercizio 1 con pre e post condizioni

<p>HADOOP EX 2 BASIC</p> <p>INPUT:</p> <ul style="list-style-type: none">- path to document "historical_stock_prices.csv".- path to document "historical_stocks.csv".- path for temp result.- path for output result. <p>PRE-COND:</p> <ul style="list-style-type: none">- path to historical_stock_prices.csv needs to be valid.- path to historical_stocks.csv needs to be valid.- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields: <table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- OPEN</td><td>Float</td><td>open price</td></tr><tr><td>- CLOSE</td><td>Float</td><td>close price</td></tr><tr><td>- ADJ_CLOSE</td><td>Float</td><td>adjusted close price</td></tr><tr><td>- LOW</td><td>Float</td><td>min price</td></tr><tr><td>- HIGH</td><td>Float</td><td>max price</td></tr><tr><td>- VOLUME</td><td>Long</td><td>number of transactions</td></tr><tr><td>- DATE</td><td>Date</td><td>date in format yyyy-mm-dd</td></tr></table> <ul style="list-style-type: none">- historical_stocks.csv needs to be a document, separated by commas, with 5 fields: <table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- EXCHANGE</td><td>String</td><td>NYSE or NASDAQ</td></tr><tr><td>- NAME</td><td>String</td><td>company name</td></tr><tr><td>- SECTOR</td><td>String</td><td>sector name</td></tr><tr><td>- INDUSTRY</td><td>String</td><td>industry name</td></tr></table> <ul style="list-style-type: none">- temp path needs to be valid.- output path needs to be valid. <p>POST-COND:</p> <ul style="list-style-type: none">- output file needs to be a document, separated by commas, with 5 fields: <table><tr><td>- SECTOR</td><td>String</td><td></td></tr><tr><td>- YEAR</td><td>String</td><td>between 2008-2018</td></tr><tr><td>- avgVolume</td><td>Long</td><td>average annual volume of sector reported in period 2008-2018</td></tr><tr><td>- DELTA_QUOTATION_%</td><td>String</td><td>(Integer with %) percentage change rounded of volume reported in period 2008-2018</td></tr><tr><td>- avgDailyClose</td><td>Long</td><td>average daily quotation of sector reported in period 2008-2018</td></tr></table>	- TICKER	String	unique share symbol	- OPEN	Float	open price	- CLOSE	Float	close price	- ADJ_CLOSE	Float	adjusted close price	- LOW	Float	min price	- HIGH	Float	max price	- VOLUME	Long	number of transactions	- DATE	Date	date in format yyyy-mm-dd	- TICKER	String	unique share symbol	- EXCHANGE	String	NYSE or NASDAQ	- NAME	String	company name	- SECTOR	String	sector name	- INDUSTRY	String	industry name	- SECTOR	String		- YEAR	String	between 2008-2018	- avgVolume	Long	average annual volume of sector reported in period 2008-2018	- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of volume reported in period 2008-2018	- avgDailyClose	Long	average daily quotation of sector reported in period 2008-2018	<pre>ex2HSPMapper(key, value) skip first row //header check_correctness_data(value) //correct types and date in correct period close,volume,date <- extractRelevantValues(value) //relevant values are "close","volume","date" key <- extractTicker(value) if (2008 <= date.year() && date.year() <= 2018): EMIT(ticker, (close, volume, date)) ex2HSMapper(key, value) skip first row //header check_correctness_data(value) //correct types and date in correct period sector <- extractRelevantValues(value) //relevant value is "sector" key <- extractTicker(value) EMIT(ticker, sector) ex2JoinReducer(key, values) initialize maps actionYearFirstDate, actionYearLastDate, actionYearFirstClose, actionYearLastClose, actionYearSumVolume, actionYearSumDailyClose, actionYearNumRows sector <- values.get() close,volume,date <- values.get() for each text in values: update actionYearFirstDate, actionYearLastDate, actionYearFirstClose, actionYearLastClose, actionYearSumVolume, actionYearSumDailyClose, actionYearNumRows for each year in actionYearFirstClose: lastClose <- actionYearLastClose.get(year) firstClose <- actionYearFirstClose.get(year) deltaQuotation <- ((lastClose - firstClose) / firstClose) * 100 sumVolume <- actionYearSumVolume.get(year) sumDailyClose <- actionYearSumDailyClose.get(year) yearRow <- actionYearNumRows.get(year) EMIT((sector,year), (sumVolume,deltaQuotation,sumDailyClose,yearRow)) ex2Mapper(key, value) sumVolume,deltaQuotation,sumDailyClose,yearRow <- extractRelevantValues(value) EMIT((sector,year), (sumVolume,deltaQuotation,sumDailyClose,yearRow)) ex2Reducer(key, values) initialize sectorSumVolume, sectorSumDeltaQuotation, sectorSumDailyClose, sectorYearRows, counterRows for each text in values: update sectorSumVolume, sectorSumDeltaQuotation, sectorSumDailyClose, sectorYearRows, counterRows avgSumVolume <- sectorSumVolume/counterRows avgDeltaQuot <- sectorSumDeltaQuotation/counterRows avgDailyClose <- sectorSumDailyClose/sectorYearRows EMIT((sector,year), (avgSumVolume,avgDeltaQuot,avgDailyClose))</pre>
- TICKER	String	unique share symbol																																																					
- OPEN	Float	open price																																																					
- CLOSE	Float	close price																																																					
- ADJ_CLOSE	Float	adjusted close price																																																					
- LOW	Float	min price																																																					
- HIGH	Float	max price																																																					
- VOLUME	Long	number of transactions																																																					
- DATE	Date	date in format yyyy-mm-dd																																																					
- TICKER	String	unique share symbol																																																					
- EXCHANGE	String	NYSE or NASDAQ																																																					
- NAME	String	company name																																																					
- SECTOR	String	sector name																																																					
- INDUSTRY	String	industry name																																																					
- SECTOR	String																																																						
- YEAR	String	between 2008-2018																																																					
- avgVolume	Long	average annual volume of sector reported in period 2008-2018																																																					
- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of volume reported in period 2008-2018																																																					
- avgDailyClose	Long	average daily quotation of sector reported in period 2008-2018																																																					

Figure 2: Hadoop Esercizio 2 Basic con pre e post condizioni

<p>HADOOP EX 2 WITH COMPANIES</p> <p>INPUT:</p> <ul style="list-style-type: none">- path to document "historical_stock_prices.csv".- path to document "historical_stocks.csv".- path for temp results.- path for output result. <p>PRE-COND:</p> <ul style="list-style-type: none">- path to historical_stock_prices.csv needs to be valid.- path to historical_stocks.csv needs to be valid.- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields: <table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- OPEN</td><td>Float</td><td>open price</td></tr><tr><td>- CLOSE</td><td>Float</td><td>close price</td></tr><tr><td>- ADJ_CLOSE</td><td>Float</td><td>adjusted close price</td></tr><tr><td>- LOW</td><td>Float</td><td>min price</td></tr><tr><td>- HIGH</td><td>Float</td><td>max price</td></tr><tr><td>- VOLUME</td><td>Long</td><td>number of transactions</td></tr><tr><td>- DATE</td><td>Date</td><td>date in format yyyy-mm-dd</td></tr></table> <ul style="list-style-type: none">- historical_stocks.csv needs to be a document, separated by commas, with 5 fields: <table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- EXCHANGE</td><td>String</td><td>NYSE or NASDAQ</td></tr><tr><td>- NAME</td><td>String</td><td>company name</td></tr><tr><td>- SECTOR</td><td>String</td><td>sector name</td></tr><tr><td>- INDUSTRY</td><td>String</td><td>industry name</td></tr></table> <ul style="list-style-type: none">- temp paths need to be valid.- output path needs to be valid. <p>POST-COND:</p> <ul style="list-style-type: none">- output file needs to be a document, separated by commas, with 5 fields: <table><tr><td>- SECTOR</td><td>String</td><td></td></tr><tr><td>- YEAR</td><td>String</td><td>between 2008-2018</td></tr><tr><td>- avgVolume</td><td>Long</td><td>average annual volume of sector reported in period 2008-2018</td></tr><tr><td>- DELTA_QUOTATION_%</td><td>String</td><td>(Integer with %) percentage change rounded of volume reported in period 2008-2018</td></tr><tr><td>- avgDailyClose</td><td>Long</td><td>average daily quotation of sector reported in period 2008-2018</td></tr></table>	- TICKER	String	unique share symbol	- OPEN	Float	open price	- CLOSE	Float	close price	- ADJ_CLOSE	Float	adjusted close price	- LOW	Float	min price	- HIGH	Float	max price	- VOLUME	Long	number of transactions	- DATE	Date	date in format yyyy-mm-dd	- TICKER	String	unique share symbol	- EXCHANGE	String	NYSE or NASDAQ	- NAME	String	company name	- SECTOR	String	sector name	- INDUSTRY	String	industry name	- SECTOR	String		- YEAR	String	between 2008-2018	- avgVolume	Long	average annual volume of sector reported in period 2008-2018	- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of volume reported in period 2008-2018	- avgDailyClose	Long	average daily quotation of sector reported in period 2008-2018	<pre>ex2HSPMapper(key, value) skip first row //header check_correctness_data(value) //correct types and date in correct period close,volume,date <- extractRelevantValues(value) //relevant values are "close","volume","date" key <- extractTicker(value) if (2008 <= date.year() && date.year() <= 2018): EMIT(ticker, (close, volume, date)) ex2HSMapper(key, value) skip first row //header check_correctness_data(value) //correct types and date in correct period sector <- extractRelevantValues(value) //relevant value is "sector" key <- extractTicker(value) EMIT(ticker, (sector,company)) ex2JoinReducer(key, values) initialize maps actionYearFirstDate, actionYearLastDate, actionYearFirstClose, actionYearLastClose, actionYearSumVolume, actionYearSumDailyClose, actionYearNumRows company,sector <- values.get() close,volume,date <- values.get() for each text in values: update actionYearFirstDate, actionYearLastDate, actionYearFirstClose, actionYearLastClose, actionYearSumVolume, actionYearSumDailyClose, actionYearNumRows for each year in actionYearFirstClose: lastClose <- actionYearLastClose.get(year) firstClose <- actionYearFirstClose.get(year) sumVolume <- actionYearSumVolume.get(year) sumDailyClose <- actionYearSumDailyClose.get(year) yearRow <- actionYearNumRows.get(year) EMIT((company,sector,year),(sumVolume,lastClose,firstClose, sumDailyClose,yearRow)) ex2CompanyMapper(key, value) sumVolume,lastClose,firstClose,sumDailyClose,yearRow <- extractRelevantValues(value) EMIT((sector,year), (sumVolume,lastClose,firstClose,sumDailyClose,yearRow)) ex2CompanyReducer(key, values) initialize companySumYearVolume, companySumLastCloses, companySumFirstCloses, companySumDailyCloses, companyYearRows for each text in values: update companySumYearVolume, companySumLastCloses, companySumFirstCloses, companySumDailyCloses, companyYearRows companyDeltaQuotation <- ((companySumLastCloses- companySumFirstCloses)/companySumFirstCloses)*100 companyAvgDailyClose <- companySumDailyCloses/companyYearRows EMIT((sector,year), (companySumYearVolume,companyDeltaQuotation,companyAvgDailyClose)) ex2SectorMapper(key, value) companySumYearVolume,companyDeltaQuotation,companyAvgDailyClose <- extractRelevantValues(value) EMIT((sector,year), (companySumYearVolume,companyDeltaQuotation,companyAvgDailyClose)) ex2SectorReducer(key, values) initialize sectorSumVolume, sectorSumDeltaQuotation, sectorSumDailyClose, counterCompanies for each text in values: update sectorSumVolume, sectorSumDeltaQuotation, sectorSumDailyClose, counterCompanies avgSumVolume <- sectorSumVolume / counterCompanies avgDeltaQuot = sectorSumDeltaQuotation / counterCompanies avgDailyClose = sectorSumDailyClose / counterCompanies EMIT((sector,year), (avgSumVolume,avgDeltaQuot,avgDailyClose))</pre>
- TICKER	String	unique share symbol																																																					
- OPEN	Float	open price																																																					
- CLOSE	Float	close price																																																					
- ADJ_CLOSE	Float	adjusted close price																																																					
- LOW	Float	min price																																																					
- HIGH	Float	max price																																																					
- VOLUME	Long	number of transactions																																																					
- DATE	Date	date in format yyyy-mm-dd																																																					
- TICKER	String	unique share symbol																																																					
- EXCHANGE	String	NYSE or NASDAQ																																																					
- NAME	String	company name																																																					
- SECTOR	String	sector name																																																					
- INDUSTRY	String	industry name																																																					
- SECTOR	String																																																						
- YEAR	String	between 2008-2018																																																					
- avgVolume	Long	average annual volume of sector reported in period 2008-2018																																																					
- DELTA_QUOTATION_%	String	(Integer with %) percentage change rounded of volume reported in period 2008-2018																																																					
- avgDailyClose	Long	average daily quotation of sector reported in period 2008-2018																																																					

Figure 3: Hadoop Esercizio 2 Complex con pre e post condizioni

<div>HADOOP EX 3</div> <div>INPUT:</div> <div><ul style="list-style-type: none">- path to document "historical_stock_prices.csv".- path to document "historical_stocks.csv".- path for temp result.- path for output result.</div> <div>PRE-COND:</div> <div><ul style="list-style-type: none">- path to historical_stock_prices.csv needs to be valid.- path to historical_stocks.csv needs to be valid.- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields:<table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- OPEN</td><td>Float</td><td>open price</td></tr><tr><td>- CLOSE</td><td>Float</td><td>close price</td></tr><tr><td>- ADJ_CLOSE</td><td>Float</td><td>adjusted close price</td></tr><tr><td>- LOW</td><td>Float</td><td>min price</td></tr><tr><td>- HIGH</td><td>Float</td><td>max price</td></tr><tr><td>- VOLUME</td><td>Long</td><td>number of transactions</td></tr><tr><td>- DATE</td><td>Date</td><td>date in format yyyy-mm-dd</td></tr></table><ul style="list-style-type: none">- historical_stocks.csv needs to be a document, separated by commas, with 5 fields:<table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- EXCHANGE</td><td>String</td><td>NYSE or NASDAQ</td></tr><tr><td>- NAME</td><td>String</td><td>company name</td></tr><tr><td>- SECTOR</td><td>String</td><td>sector name</td></tr><tr><td>- INDUSTRY</td><td>String</td><td>industry name</td></tr></table><ul style="list-style-type: none">- temp path needs to be valid.- output path needs to be valid.</div> <div>POST-COND:</div> <div><ul style="list-style-type: none">- output file needs to be a document, separated by commas, with 5 fields:<table><tr><td>- SIMILARITIES</td><td>String[]</td><td>list of companies with same trend in year 2016-2018</td></tr><tr><td>- TREND</td><td>Integer[3]</td><td>percentage variation of company in year 2016,2017,2018</td></tr></table></div>	- TICKER	String	unique share symbol	- OPEN	Float	open price	- CLOSE	Float	close price	- ADJ_CLOSE	Float	adjusted close price	- LOW	Float	min price	- HIGH	Float	max price	- VOLUME	Long	number of transactions	- DATE	Date	date in format yyyy-mm-dd	- TICKER	String	unique share symbol	- EXCHANGE	String	NYSE or NASDAQ	- NAME	String	company name	- SECTOR	String	sector name	- INDUSTRY	String	industry name	- SIMILARITIES	String[]	list of companies with same trend in year 2016-2018	- TREND	Integer[3]	percentage variation of company in year 2016,2017,2018	<div>ex3HSPMapper(key, value)</div> <div>skip first row //header</div> <div>check_correctness_data(value) //correct types and date in correct period</div> <div>close,volume,date <- extractRelevantValues(value) //relevant values are "close","volume","date"</div> <div>key <- extractTicker(value)</div> <div>if (2008 <= date.year() && date.year() <= 2018):</div> <div>EMIT(ticker, (close, volume, date))</div> <div>ex3HSMapper(key, value)</div> <div>skip first row //header</div> <div>check_correctness_data(value) //correct types and date in correct period</div> <div>company <- extractRelevantValues(value) //relevant value is "company"</div> <div>key <- extractTicker(value)</div> <div>EMIT(ticker, company)</div> <div>ex3JoinReducer(key,values)</div> <div>SETUP phase</div> <div>initialize maps companyYearStartQuotation, companyYearEndQuotation, companyAnnualVariations</div> <div>REDUCE phase</div> <div>joinValuesOn(key)</div> <div>initialize maps yearFirstDate,yearLastDate,yearFirstClose,yearLastClose</div> <div>for each value in values</div> <div>year <- value.year</div> <div>firstDate <- MIN(value.date,yearFirstDate.get(year))</div> <div>lastDate <- MAX(value.date,yearLastDate.get(year))</div> <div>firstClose <- close associated to firstDate</div> <div>lastClose <- close associated to lastDate</div> <div>update maps pair with key year</div> <div>for each year in yearFirstDate</div> <div>firstClose <- yearFirstClose.get(year)</div> <div>lastClose <- yearLastClose.get(year)</div> <div>sumFirstCloses <- companyYearStartQuotation.get(company,year) + firstClose</div> <div>sumLastCloses <- companyYearEndQuotation.get(company,year) + lastClose</div> <div>update maps companyYearStartQuotation, companyYearEndQuotation with key company,year</div> <div>CLEANUP phase</div> <div>for each company,year in companyYearStartQuotation</div> <div>sumFirstCloses <- companyYearStartQuotation.get(companyYear)</div> <div>sumLastCloses <- companyYearStartQuotation.get(companyYear)</div> <div>companyAnnualVariation <- ((sumLastCloses - sumFirstCloses) /sumFirstCloses)*100</div> <div>update companyAnnualVariations with key company and value companyAnnualVariation for respective year</div> <div>EMIT(company, (companyAnnualVariations.get(company)))</div> <div>Ex3Mapper(key,value)</div> <div>checkCorrectValues(value)</div> <div>EMIT(key,value)</div> <div>Ex3Reducer(key,values)</div> <div>SETUP phase</div> <div>initialize maps companiesNumbers,companiesQuotations</div> <div>EMIT(header)</div> <div>REDUCE phase</div> <div>for each value in values</div> <div>companies <- append value.company</div> <div>counterCompanies <- companies.length</div> <div>if(counterCompanies>1)</div> <div>update maps companiesNumbers,companiesQuotations</div> <div>CLEANUP phase</div> <div>for each comps in companiesNumbers</div> <div>numberOfCompanies <- companiesNumbers.get(comps)</div> <div>trend <- companiesQuotations.get(comps)</div> <div>EMIT(numberOfCompanies, (comps,trend))</div>
- TICKER	String	unique share symbol																																												
- OPEN	Float	open price																																												
- CLOSE	Float	close price																																												
- ADJ_CLOSE	Float	adjusted close price																																												
- LOW	Float	min price																																												
- HIGH	Float	max price																																												
- VOLUME	Long	number of transactions																																												
- DATE	Date	date in format yyyy-mm-dd																																												
- TICKER	String	unique share symbol																																												
- EXCHANGE	String	NYSE or NASDAQ																																												
- NAME	String	company name																																												
- SECTOR	String	sector name																																												
- INDUSTRY	String	industry name																																												
- SIMILARITIES	String[]	list of companies with same trend in year 2016-2018																																												
- TREND	Integer[3]	percentage variation of company in year 2016,2017,2018																																												

Figure 4: Hadoop Esercizio 3 con pre e post condizioni

4.2 Spark

<pre> SPARK EX 1 INPUT: - path to document "historical_stock_prices.csv". - path for output result. PRE-COND: - path to historical_stock_prices.csv needs to be valid. - historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields: - TICKER String unique share symbol - OPEN Float open price - CLOSE Float close price - ADJ_CLOSE Float adjusted close price - LOW Float min price - HIGH Float max price - VOLUME Long number of transactions - DATE Date date in format yyyy-mm-dd - output path needs to be valid. POST-COND: - output file needs to be a document, separated by commas, with 5 fields: - TICKER String - DELTA_QUOTATION_% String (Integer with %) percentage change rounded of ticker value in period 2008-2018 - MIN_CLOSE Float min "close" value reported in period 2008-2018 - MAX_CLOSE Float max "close" value reported in period 2008-2018 - AVG_VOLUME Long average "volume" in period 2008-2018 - document rows need to be sorted by descending DELTA_QUOTATION </pre>	<pre> main valuesHSP <- checkline(inputHSP).prepareValues valuesHSP.prepareValues(line) .reducer(tuple1, tuple2) .produceResults(tuple) .descendentSort(result.percentageChange) checkline(line) //filter dirty rows if (check_correctness_data(line) && //correct types 2008 <= date.year() && date.year() <= 2018): return true return false prepareValues(line) //organize data for later manipulation firstDate,lastDate <- line.date firstClose,lastClose,minClose,maxClose <- line.close volume <- line.volume counter <- 1 key <- ticker value <- firstDate,lastDate,firstClose,lastClose, minClose,maxClose,volume,counter EMIT(key,value) reducer(tuple1, tuple2) //aggregate on ticker firstDate <- MIN(tuple1.date , tuple2.date) lastDate <- MAX(tuple1.date , tuple2.date) firstClose <- close associated to firstDate lastClose <- close associated to lastDate minClose <- MIN(tuple1.close , tuple2.close) maxClose <- MAX(tuple1.close , tuple2.close) sumVolume <- tuple1.volume + tuple2.volume counter <- tuple1.counter + tuple2.counter value <- firstDate,lastDate,firstClose,lastClose, minClose,maxClose,sumVolume,counter EMIT(value) produceResults(tuple) //calculate results ticker <- tuple.ticker percentageChange <- ((tuple.lastClose - tuple.firstClose) / tuple.firstClose) * 100 minPrice <- tuple.minPrice maxPrice <- tuple.maxPrice avgVolume <- tuple.volume / tuple.counter result <- ticker, percentageChange, minPrice, maxPrice, avgVolume EMIT(result) </pre>
---	--

Figure 5: Spark Esercizio 1 con pre e post condizioni

SPARK EX 2

INPUT:

- path to document "historical_stock_prices.csv".
- path to document "historical_stocks.csv".
- path for output result.

PRE-COND:

- path to historical_stock_prices.csv needs to be valid.
- path to historical_stocks.csv needs to be valid.
- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields:

- TICKER	String	unique share symbol
- OPEN	Float	open price
- CLOSE	Float	close price
- ADJ_CLOSE	Float	adjusted close price
- LOW	Float	min price
- HIGH	Float	max price
- VOLUME	Long	number of transactions
- DATE	Date	date in format yyyy-mm-dd

- historical_stocks.csv needs to be a document, separated by commas, with 5 fields:

- TICKER	String	unique share symbol
- EXCHANGE	String	NYSE or NASDAQ
- NAME	String	company name
- SECTOR	String	sector name
- INDUSTRY	String	industry name

- output path needs to be valid.

POST-COND:

- output file needs to be a document, separated by commas, with 5 fields:

- SECTOR	String	
- YEAR	String	between 2008-2018
- avgVolume	Long	average annual volume of sector reported in period 2008-2018
- DELTA_QUOTATION_%	String	(Integer with percentage change rounded of volume reported in period 2008-2018)
- avgDailyClose	Long	average daily quotation of sector reported in period 2008-2018

main

```
valuesHSP <- checkInputHSP(inputHSP).prepareValuesHSP
valuesHS <- checkInputHSP(inputHS).prepareValuesHS
valuesHSP.join(valuesHS) .reorganizeValuesAfterJoin
    .reduce_sumVolumesTicker_findFirstLastClose.map_fromTickerToCompany
    .reduce_sumVolumeCompany_SumFirstLastCloseCompany_sumCloseSumCounterCompan
    .map_varYearCompany_dailyQuotCompany_fromCompanyToSector
    .reduce_sumVolumes_sumVars_sumQuots.map_avgSector.sortBy(key);

checkInputHS(line) //filter dirty rows
if(check_correctness_data(line))//correct types
    return true
return false

checkInputHSP(line) //filter dirty rows
if (check_correctness_data(line) && //correct types
    2008 <= date.year() && date.year() <= 2018):
    return true
return false

prepareValuesHS(line)//take relevant fields from historical_stocks
company,sector <- extractValues(line)
key <- line.ticker
value <- company,sector
EMIT(key,value)

prepareValuesHSP(line)//take relevant fields from historical_stock_prices
close,volume,date <- extractValues(line)
key <- line.ticker
value <- close,volume,date
EMIT(key, value)

reorganizeValuesAfterJoin(tuple) //organize values for later computation
volume <- tuple.volume
firstDate,lastDate <- tuple.date
firstClose,lastClose,close <- tuple.close
counterDays <- 1
company <- tuple.company
sector <- tuple.sector
key <- tuple.ticker + tuple.year
value <-
volume,firstDate,lastDate,firstClose,lastClose,close,counterDays,company,sector
EMIT(key, value)

reduce_sumVolumesTicker_findFirstLastClose(tuple1, tuple2)
//aggregate values on tickers
sumVolume <- tuple1.volume + tuple2.volume
firstDate <- MIN(tuple1.date, tuple2.date)
lastDate <- MAX(tuple1.date, tuple2.date)
firstClose <- close associated to firstDate
lastClose <- close associated to lastDate
sumClose <- tuple1.close + tuple2.close
counterDays <- tuple1.counterDays + tuple2.counterDays
company <- tuple1.company
sector <- tuple1.sector
value <-
sumVolume,firstDate,lastDate,firstClose,lastClose,sumClose,counterDays,company,sector
EMIT(value)

map_fromTickerToCompany(tuple) //change key to company year
sumVolume,firstClose,lastClose,sumClose,counterDays,sector <- extractValues(tuple)
key <- tuple.company + tuple.year
value <- sumVolume,firstClose,lastClose,sumClose,counterDays,sector
EMIT(key,value)

reduce_sumVolumeCompany_SumFirstLastCloseCompany_sumCloseSumCounterCompany
(tuple1, tuple2) //aggregate on company,year
sumVolumeCompany <- tuple1.volume + tuple2.volume
sumFirstCloses <- tuple1.firstClose + tuple2.firstClose
sumLastCloses <- tuple1.lastClose + tuple2.lastClose
sumCloses <- tuple1.close + tuple2.close
smCounterDays <- tuple1.counterDays + tuple2.counterDays
sector <- tuple1.sector
value <- sumVolumeCompany,sumFirstCloses,sumLastCloses,
sumCloses,sumCounterDays,sector
EMIT(value)

map_varYearCompany_dailyQuotCompany_fromCompanyToSector(tuple)
//aggregate companies and change key to sector year
sumVolume <- tuple.sumVolumeCompany
varYear <- ((tuple.lastCloses - tuple.firstCloses) / tuple.firstCloses) * 100
dailyQuot <- tuple.sumCloses / tuple.sumCounterDays
counterCompanies <- 1
key <- tuple.sector + tuple.year
value <- sumVolume,varYear,dailyQuot,counterCompanies
EMIT(key,value)

reduce_sumVolumes_sumVars_sumQuots(tuple1, tuple2) //aggregato on sector year
sumVolumes <- tuple1.volume + tuple2.volume
sumVars <- tuple1.varYear + tuple2.varYear
sumQuots <- tuple1.dailyQuot + tuple2.dailyQuot
counterCompanies <- tuple1.counterCompanies + tuple2.counterCompanies
value <- sumVolumes,sumVars,sumQuots,counterCompanies
EMIT(value)

map_avgSector(tuple) //compute results
sectorYear <- tuple.key
avgVolumes <- tuple.sumVolumes / tuple.counterCompanies
avgVars <- tuple.sumVars / tuple.counterCompanies
avgDailyQuots <- tuple.sumQuots / tuple.counterCompanies
result <- sectorYear,avgVolumes,avgVars,avgDailyQuots
EMIT(result)
```

Figure 6: Spark Esercizio 2 complex con pre e post condizioni

<div>SPARK EX 3</div> <div>INPUT:</div> <div><ul style="list-style-type: none">- path to document "historical_stock_prices.csv".- path to document "historical_stocks.csv".- path for temp result.- path for output result.</div> <div>PRE-COND:</div> <div><ul style="list-style-type: none">- path to historical_stock_prices.csv needs to be valid.- path to historical_stocks.csv needs to be valid.- historical_stock_prices.csv needs to be a document, separated by commas, with 8 fields:<table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- OPEN</td><td>Float</td><td>open price</td></tr><tr><td>- CLOSE</td><td>Float</td><td>close price</td></tr><tr><td>- ADJ_CLOSE</td><td>Float</td><td>adjusted close price</td></tr><tr><td>- LOW</td><td>Float</td><td>min price</td></tr><tr><td>- HIGH</td><td>Float</td><td>max price</td></tr><tr><td>- VOLUME</td><td>Long</td><td>number of transactions</td></tr><tr><td>- DATE</td><td>Date</td><td>date in format yyyy-mm-dd</td></tr></table><ul style="list-style-type: none">- historical_stocks.csv needs to be a document, separated by commas, with 5 fields:<table><tr><td>- TICKER</td><td>String</td><td>unique share symbol</td></tr><tr><td>- EXCHANGE</td><td>String</td><td>NYSE or NASDAQ</td></tr><tr><td>- NAME</td><td>String</td><td>company name</td></tr><tr><td>- SECTOR</td><td>String</td><td>sector name</td></tr><tr><td>- INDUSTRY</td><td>String</td><td>industry name</td></tr></table><ul style="list-style-type: none">- temp path needs to be valid.- output path needs to be valid.</div> <div>POST-COND:</div> <div><ul style="list-style-type: none">- output file needs to be a document, separated by commas, with 5 fields:<table><tr><td>- SIMILARITIES</td><td>String[]</td><td>list of companies with same trend in year 2016-2018</td></tr><tr><td>- TREND</td><td>Integer[3]</td><td>percentage variation of company in year 2016,2017,2018</td></tr></table></div>	- TICKER	String	unique share symbol	- OPEN	Float	open price	- CLOSE	Float	close price	- ADJ_CLOSE	Float	adjusted close price	- LOW	Float	min price	- HIGH	Float	max price	- VOLUME	Long	number of transactions	- DATE	Date	date in format yyyy-mm-dd	- TICKER	String	unique share symbol	- EXCHANGE	String	NYSE or NASDAQ	- NAME	String	company name	- SECTOR	String	sector name	- INDUSTRY	String	industry name	- SIMILARITIES	String[]	list of companies with same trend in year 2016-2018	- TREND	Integer[3]	percentage variation of company in year 2016,2017,2018	<div>main</div> <div><pre>valuesHSP <- checkInputHSP(inputHSP).prepareValuesHSP valuesHS <- checkInputHSP(inputHS).prepareValuesHS valuesHSP.join(valuesHS).reorganizeValuesAfterJoin .reduce_findFirstLastCloses.map_fromTickerToCompany .reduce_sumFirstLastCloses.map_calculateVarPercCompanyYear_changeKeyToCompany .reduce_unifyTrends.checkAllYearPresent .invertKey_fromCompany_toVarYear.reduce_companySameTrend .mapByNumberSimilarCompaniesTrend.sortBy(similarities.length); checkInputHS(line) //filter dirty rows if(check_correctness_data(line))//correct types return true return false checkInputHSP(line) //filter dirty rows if (check_correctness_data(line) && //correct types 2016 <= date.year() && date.year() <= 2018): return true return false prepareValuesHS(line) //take relevant fields from historical_stocks company,sector <- extractValues(line) key <- line.ticker value <- company,sector EMIT(key,value) prepareValuesHSP(line) //take relevant fields from historical_stock_prices close,volume,date <- extractValues(line) key <- line.ticker value <- close,volume,date EMIT(key,value) reorganizeValuesAfterJoin(tuple) //prepare values for later computation firstDate,lastDate <- tuple.date firstClose,lastClose <- tuple.close company <- tuple.company key <- tuple.ticker + tuple.year value <- firstDate,lastDate,firstClose,lastClose,company EMIT(key,value) reduce_findFirstLastCloses(tuple1,tuple2) //aggregate on ticker firstDate <- MIN(tuple1.date,tuple2.date) lastDate <- MAX(tuple1.date,tuple2.date) firstClose <- close associated to firstDate lastClose <- close associated to lastDate company <- tuple1.company value <- firstDate,lastDate,firstClose,lastClose,company EMIT(value) map_fromTickerToCompany(tuple) //change key to company firstClose <- tuple.firstClose lastClose <- tuple.lastClose key <- tuple.company + tuple.year value <- firstClose,lastClose EMIT(key,value) reduce_sumFirstLastCloses(tuple1,tuple2) //aggregate on company sumFirstCloses <- tuple1.firstClose + tuple2.firstClose sumLastCloses <- tuple1.lastClose + tuple2.lastClose value <- sumFirstCloses,sumLastCloses EMIT(value) map_calculateVarPercCompanyYear_changeKeyToCompany(tuple) //calculate percentage variation and change key to company varYear2016,varYear2017,varYear2018 <- null varYear <- ((tuple.sumLastCloses - tuple.sumFirstCloses) / tuple.sumFirstCloses)*100 update correspondent VarYear variables according to tuple.year key <- tuple.company value <- varYear2016,varYear2017,varYear2018 EMIT(key,value) reduce_unifyTrends(tuple1,tuple2) //unify percentage variation of year on a single record varYear2016 <- tuple1.varYear2016 varYear2017 <- tuple1.varYear2017 varYear2018 <- tuple1.varYear2017 if(varYear2016==NULL) varYear2016 <- tuple2.varYear2016 if(varYear2017==NULL) varYear2017 <- tuple2.varYear2017 if(varYear2018==NULL) varYear2018 <- tuple2.varYear2018 value <- varYear2016,varYear2017,varYear2018 EMIT(value) checkAllYearPresent(tuple) //filter those that doesn't have var percentage for 2016-2017-2018 if(tuple.varYear2016==NULL tuple.varYear2017==NULL tuple.varYear2018==NULL) return false return true invertKey_fromCompany_toVarYear(tuple) //change key to trend varYear2016,varYear2017,varYear2018 <- extractValues(tuple) key <- varYear2016,varYear2017,varYear2018 value <- tuple.company EMIT(key,value) reduce_companySameTrend(tuple1,tuple2) //unify companies with same trend value <- tuple1.company, tuple2.company EMIT(value) mapByNumberSimilarCompaniesTrend(tuple) //calculate number of similarities to order results and change key to that value key <- tuple.companies.length value <- tuple.companies + tuple.varYear2016 + tuple.varYear2017 + tuple.varYear2018 EMIT(key,value)</pre></div>
- TICKER	String	unique share symbol																																												
- OPEN	Float	open price																																												
- CLOSE	Float	close price																																												
- ADJ_CLOSE	Float	adjusted close price																																												
- LOW	Float	min price																																												
- HIGH	Float	max price																																												
- VOLUME	Long	number of transactions																																												
- DATE	Date	date in format yyyy-mm-dd																																												
- TICKER	String	unique share symbol																																												
- EXCHANGE	String	NYSE or NASDAQ																																												
- NAME	String	company name																																												
- SECTOR	String	sector name																																												
- INDUSTRY	String	industry name																																												
- SIMILARITIES	String[]	list of companies with same trend in year 2016-2018																																												
- TREND	Integer[3]	percentage variation of company in year 2016,2017,2018																																												

Figure 7: Spark Esercizio 3 con pre e post condizioni

4.3 Hive

```
HIVE EX 1

DROP TABLE if exists ticker_firstlastvalues;

CREATE TEMPORARY TABLE ticker_firstlastvalues
AS
SELECT ticker as ticker,
       MIN(day) as first_date,
       MAX(day) as last_date,
       MIN(close) as min_close,
       MAX(close) as max_close,
       FLOOR(AVG(volume)) as avg_volume
FROM historical_stock_prices
WHERE year(day) between '2008' and '2018'
GROUP BY ticker;

----- VERSION WITHOUT TEMPORARY TABLES -----
DROP TABLE if exists ex1_hive;

CREATE TABLE ex1_hive
ROW FORMAT DELIMITED FIELDS TERMINATED by ','
AS
SELECT tfc.ticker as ticker,
       ROUND(((last_close - first_close)/first_close)*100, 0) as d_quot,
       min_close,
       max_close,
       avg_volume
FROM ( SELECT tfd.ticker as ticker,
              hsp.close as first_close,
              min_close,
              max_close,
              avg_volume
        FROM ticker_firstlastvalues tfd
        JOIN historical_stock_prices hsp
          ON (tfid.ticker = hsp.ticker and tfd.first_date = hsp.day)
      ) tfc
JOIN ( SELECT tld.ticker as ticker,
              hsp.close as last_close
        FROM ticker_firstlastvalues tld
        JOIN historical_stock_prices hsp
          ON (tld.ticker = hsp.ticker and tld.last_date = hsp.day)
      ) tlc
ON (tfc.ticker = tlc.ticker)
ORDER BY d_quot desc;
```

Figure 8: Hive Esercizio 1 hql

```

HIVE EX 2 BASIC

DROP TABLE if exists ticker_sector;

CREATE TEMPORARY TABLE ticker_sector
AS
SELECT hs.ticker as ticker,
       sector,
       volume,
       close,
       day
FROM historical_stocks hs JOIN historical_stock_prices hsp
ON hs.ticker = hsp.ticker
WHERE year(day) between '2008' and '2018'
and sector != 'N/A';

DROP TABLE if exists ticker_sector_year;

CREATE TEMPORARY TABLE ticker_sector_year
AS
SELECT ticker,
       sector,
       year(day) as year,
       SUM(volume) as volumes,
       SUM(close) as closes,
       MIN(day) as first_date,
       MAX(day) as last_date,
       COUNT(*) as num
FROM ticker_sector
GROUP BY ticker, sector, year(day);

DROP TABLE if exists ex2basic_hive;

CREATE TABLE ex2basic_hive
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
AS
SELECT tmp.sector as sector,
       tmp.year as year,
       cast(AVG(tmp.volumes) as BIGINT) as avgVolume,
       ROUND(AVG(((ts.close - first_close) / first_close) * 100), 2) as
delta_quot,
       ROUND(SUM(tmp.closes) / SUM(num), 2)
FROM
    ( SELECT ty.ticker,
            ty.sector,
            ty.year,
            ty.volumes,
            ty.closes,
            ts.close as first_close,
            last_date,
            num
      FROM ticker_sector_year ty JOIN ticker_sector ts
      ON ty.ticker = ts.ticker
      and ty.sector = ts.sector
      and ty.year = year(ts.day)
      and ty.first_date = ts.day ) tmp
JOIN ticker_sector ts
ON tmp.ticker = ts.ticker
and tmp.sector = ts.sector
and tmp.year = year(ts.day)
and last_date = ts.day
GROUP BY tmp.sector, tmp.year;
maxClose = close
sumVolumes += text.volume
numValues++

deltaQuotation = ((lastClose - firstClose) / firstClose) * 100
avgVolume = sumVolumes / numValues

EMIT(ticker, (deltaQuotation,minClose,maxClose,avgVolume))

```

Figure 9: Hive Esercizio 2 Basic hql

```

HIVE EX 2 COMPLEX

DROP TABLE if exists ticker_firstlastdateyear;
CREATE TEMPORARY TABLE ticker_firstlastdateyear
AS
SELECT hs.ticker,
       company,
       sector,
       year(day) as year,
       SUM(volume) as totvolume_ticker,
       MIN(day) as firstdate_ticker,
       MAX(day) as lastdate_ticker,
       SUM(close) as totclose_ticker,
       COUNT(*) as totcount_ticker
FROM historical_stocks hs JOIN historical_stock_prices hsp
ON hs.ticker = hsp.ticker
WHERE year(day) between '2008' and '2018'
      and sector != 'N/A'
      and company != 'N/A'
GROUP BY hs.ticker, company, sector, year(day);

DROP TABLE if exists ticker_firstlastcloseyear;
CREATE TEMPORARY TABLE ticker_firstlastcloseyear
AS
SELECT first.ticker as ticker,
       company,
       sector,
       first.year as year,
       totvolume_ticker,
       firstclose_ticker,
       lastclose_ticker,
       totclose_ticker,
       totcount_ticker
FROM ( SELECT tfldy.ticker as ticker,
       company, sector, year, totvolume_ticker,
       hsp.close as firstclose_ticker,
       totclose_ticker, totcount_ticker
       FROM ticker_firstlastdateyear tfldy JOIN historical_stock_prices hsp
       ON (tfldy.ticker = hsp.ticker and tfldy.firstdate_ticker = hsp.day)
     ) first
JOIN ( SELECT tfldy.ticker as ticker,
       year as year,
       hsp.close as lastclose_ticker
       FROM ticker_firstlastdateyear tfldy JOIN historical_stock_prices
       hsp
       ON (tfldy.ticker = hsp.ticker and tfldy.lastdate_ticker =
       hsp.day)
     ) last
ON (first.ticker = last.ticker and first.year = last.year);

DROP TABLE if exists company_quotationyear;
CREATE TEMPORARY TABLE company_quotationyear
AS
SELECT company,
       sector,
       year,
       totvolume_company,
       (((lastcloses_company - firstcloses_company) / firstcloses_company)
* 100) as delta_quot,
       (totclose_company / totcount_company) as avg_dailyquot
FROM ( SELECT company,
       sector,
       year,
       SUM(totvolume_ticker) as totvolume_company,
       SUM(firstclose_ticker) as firstcloses_company,
       SUM(lastclose_ticker) as lastcloses_company,
       SUM(totclose_ticker) as totclose_company,
       SUM(totcount_ticker) as totcount_company
       FROM ticker_firstlastcloseyear
       GROUP BY company, sector, year
     ) tmp;

DROP TABLE if exists ex2complex_hive;
CREATE TABLE ex2complex_hive
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
AS
SELECT sector,
       year,
       cast(AVG(totvolume_company) as BIGINT) as avg_volume,
       ROUND(AVG(delta_quot), 2) as delta_quot,
       ROUND(AVG(avg_dailyquot), 2) as avg_dailyquot
FROM company_quotationyear
GROUP BY sector, year;

```

Figure 10: Hive Esercizio 2 Complex hql

```

HIVE EX3

DROP TABLE if exists ticker_firstlastdateyear;

CREATE TEMPORARY TABLE ticker_firstlastdateyear
AS
SELECT ticker,
       year(day) as year,
       MIN(day) as first_date,
       MAX(day) as last_date
FROM historical_stock_prices
WHERE year(day) between '2016' and '2018'
GROUP BY ticker, year(day);

----- quotation - PERSONAL VERSION: sum first_close and sum last_close,
then delta_quot -----
DROP TABLE if exists ticker_quotationyear;

CREATE TEMPORARY TABLE ticker_quotationyear
AS
SELECT first.ticker as ticker,
       first.year as year,
       firstclose_ticker,
       lastclose_ticker
FROM ( SELECT tfldy.ticker as ticker,
              tfldy.year as year,
              hsp.close as firstclose_ticker
        FROM ticker_firstlastdateyear tfldy JOIN historical_stock_prices hsp
        ON (tfldy.ticker = hsp.ticker and tfldy.first_date = hsp.day)
      ) first
JOIN ( SELECT tfldy.ticker as ticker,
              tfldy.year as year,
              hsp.close as lastclose_ticker
        FROM ticker_firstlastdateyear tfldy JOIN historical_stock_prices hsp
        ON (tfldy.ticker = hsp.ticker and tfldy.last_date = hsp.day)
      ) last
ON (first.ticker = last.ticker and first.year = last.year);

DROP TABLE if exists company_quotationyear;

CREATE TEMPORARY TABLE company_quotationyear
AS
SELECT company,
       COLLECT_SET( CONCAT( cast(year as STRING), ":",
       cast(cast(((lastcloses_company - firstcloses_company)/
       firstcloses_company)*100 as BIGINT)
       as STRING), "%")) as quot_years
FROM (
  SELECT company, year,
         SUM(firstclose_ticker) as firstcloses_company,
         SUM(lastclose_ticker) as lastcloses_company
  FROM historical_stocks hs JOIN ticker_quotationyear tqy
  ON hs.ticker = tqy.ticker
  GROUP BY company, year
) tmp
GROUP BY company;

DROP TABLE if exists ex3_hive;

CREATE TABLE ex3_hive
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
AS
SELECT CONCAT ("{" , CONCAT_WS(';', comp_list), "}") as companies,
       quot_years
FROM ( SELECT COLLECT_SET(company) as comp_list,
       quot_years
       FROM ( SELECT company,
                     CONCAT_WS(';', quot_years) as quot_years
               FROM company_quotationyear
               WHERE size(quot_years)==3
             ) comp_totquot
       GROUP BY quot_years
       ex3_hive_tot
WHERE size(comp_list)>1
ORDER BY size(comp_list) desc;

```

Figure 11: Hive Esercizio 3 Standard hql

5 Implementazione

In questa Sezione vengono presentate più nel dettaglio le scelte implementative adottate durante lo sviluppo dei vari Job nelle varie tecnologie. Inizialmente si illustrano le decisioni comuni ai vari esercizi, per poi focalizzarsi, rispettivamente, su Hadoop, Hive e Spark. Tutte le seguenti Sotto-Sezioni fanno riferimento al codice del progetto, fornito in allegato a questa relazione.

5.1 Scelte generali

- Ogni job è stato progettato per funzionare individualmente e a prescindere da una eventuale pre-pulizia dei dati. In una fase successiva si è considerata la possibilità di aumentare l'efficienza dei vari job, spostando l'operazione di pulizia e filtraggio dei dati a monte dell'elaborazione e dando in input ai job direttamente dei dati puliti. Questa scelta è stata implementata e ha portato a dei risultati leggermente positivi nel Job1 e Job2, con diminuzione dei tempi di esecuzione tra il 5% ed il 10%, ma sempre sotto i 2 secondi di guadagno. Nel Job3, al contrario, portava a dei risultati differenti, in quanto la pulizia dei dati di HS svolta specificatamente per il Job3 considerava soltanto i campi per esso rilevanti, ovvero "ticker" e "company", mentre la pulizia dei dati generalizzata considerava anche il campo "sector". Questo ha portato ad un pruning di valori maggiore, incorretto per il Job3, in quanto esso necessita dei soli campi "ticker" e "company", e il risparmio di pochi secondi non compensa la perdita di dati utili e rilevanti per l'analisi richiesta. Al contrario, una pulizia dei dati a monte ad-hoc per ogni Job è stata ritenuta superflua, oltre che potenzialmente penalizzante nel caso in cui non fosse richiesto di eseguire più volte lo stesso job sullo stesso input.
- Si è costruita una classe *Utility* per metodi di supporto presenti in uno o più Job, come il metodo *inputExist(String s)* che verifica l'esistenza della stringa passata, per controllare se un campo dei record fosse presente ma vuoto.
- Per la separazione dei record presenti in HS è stato necessario utilizzare un'espressione regolare, in quanto, da un'analisi del dataset, si è riscontrato che in alcuni campi "company" fosse presente una virgola che avrebbe portato a suddividere in più campi la riga, facendo erroneamente risultare il dato come sporco. Fortunatamente, in presenza delle virgole all'interno del campo, il campo stesso viene delimitato da delle virgolette, che permettono la corretta identificazione del dato utilizzando un'espressione regolare all'interno dello split.
- Si è preferito utilizzare, come risultati intermedi delle varie elaborazioni dei vari job, dei campi in formato Text sui quali viene eseguito il parsing all'inizio di ogni fase. Questo ha permesso di mantenere bassa la complessità generale del progetto, evitando di dover scrivere numerose classi per oggetti intermedi che, tra l'altro, avrebbero occupato maggiore spazio.
- I file in output sono stati forniti senza alcuna particolare formattazione per facilitare la conversione a seconda dell'utilizzo che se ne debba fare: si è comunque prediletta una formattazione dei dati comoda per una divisione nel tipico formato .csv, usando la virgola come carattere separatore. Inoltre nelle tecnologie Hadoop e Spark, per ogni Job viene inserito in testa al file di output un'intestazione esplicativa dei valori delle colonne.

5.2 Hadoop

ESERCIZIO 1

Questo esercizio presenta una sola implementazione in quanto non sono state trovate differenti interpretazioni per esso. Consiste in un singolo job e comprende un solo Mapper (*Ex1Mapper*) ed un solo Reducer (*Ex1Reducer*).

Ex1Mapper si occupa di prendere i dati da HSP, controllare che i campi delle varie righe non siano nulli o di un formato non idoneo e controllare che siano all'interno dell'intervallo temporale richiesto. Fornisce in output la data, il valore di close ed il volume associati ad ogni ticker.

Ex1Reducer prende in input i valori passati da *Ex1Mapper* ed esegue le seguenti elaborazioni:

- calcola la variazione percentuale della quotazione andando a calcolare quanto valesse il "ticker" all'inizio del periodo temporale e quanto valesse alla fine.
- calcola la media del volume del ticker, andando a sommare tutti i valori "volume" di ogni ticker e dividendolo per il numero di record incontrati.
- calcola il minimo ed il massimo "close" registrato nel periodo di interesse.

In output restituisce come chiave "ticker" e come value i valori sopraindicati, ordinando le righe per variazione percentuale decrescente. Si è scelto di utilizzare una classe ad-hoc per rappresentare i risultati del Job1 *Result_Ex1*, che implementa *Comparable*, la quale verrà riutilizzata anche dal Job1 di Spark.

Non è stato inserito un Combiner in quanto avrebbe richiesto un refactoring delle implementazioni di *Ex1Mapper* e *Ex1Reducer* e non è stato ritenuto significativo il suo sviluppo, potendo già osservare la differenza di prestazioni utilizzando o meno un Combiner dal Job2.

ESERCIZIO 2

Questo esercizio si è presentato come il più complesso in quanto sono state fornite diverse interpretazioni.

La discussione ora viene suddivisa descrivendo nel dettaglio:

- *Ex2_basic* versione nella quale vengono considerati solo "ticker" e "settore".
- *Ex2_complex* versione nella quale vengono considerati "ticker", "company" e "sector", a sua volta questa variante è stata suddivisa in:
 - *Ex2_Companies* versione che non prevede l'utilizzo di Combiner.
 - *Ex2_Companies_Combiner* versione che sfrutta due Combiner.

Esercizio 2 Basic. Questa è la versione corrispondente all'interpretazione più semplice dei dati. Come anticipato in Sezione 2, qui il campo "company" di HS non viene preso in considerazione e questo permette di avere uno step di MapReduce in meno. Comprende tre Mapper differenti:

- *Ex2HSMapper*
- *Ex2HSPMapper*
- *Ex2Mapper*

e due Reducer:

- *Ex2JoinReducer*
- *Ex2Reducer*

I Mapper *Ex2HSMapper* e *Ex2HSPMapper* costituiscono il punto di partenza del Job, incaricati di prendere i dati dai rispettivi dataset, verificarne la correttezza in termini di validazione del tipo di dato ed appartenenza all'intervallo corretto, in seguito estraggono i dati rilevanti per il job.

Il join implementato è un *Reduce Side Join*, ovvero sfrutta lo shuffle & sort per aggregare i risultati in uscita dai due Mapper rispetto a "ticker" e con l'ausilio di separatori personalizzati che permettono, lato Reducer, di distinguere la provenienza tra HSP e HS.

L'output di entrambi i primi due Mapper viene dunque dato in input a *Ex2JoinReducer*. Questo ottiene sia i dati rilevanti appartenenti ad HS che ad HSP, unificati grazie alla chiave "ticker" e distinti l'uno dagli altri grazie all'utilizzo degli appositi separatori inseriti. In questa fase vengono aggregati i valori dei due dataset per calcolare, per ogni anno e ticker, quale sia il valore iniziale e finale del ticker, oltre al volume annuale, la somma delle close annuali ed il numero di giorni dell'anno in cui si registrano record riguardanti il ticker. Per raggiungere tale scopo si sono utilizzate delle mappe, come in altre parti del progetto, per evitare la creazione di tipi di dati appositi, al fine di aumentare la leggibilità del programma e diminuire la quantità di codice introdotto.

Alla fine del reducer si calcola la variazione annuale del ticker usando i valori del ticker stesso ad inizio e fine

anno, per poi cambiare la chiave utilizzata per settore e anno, così da poter aggregare per lo stesso.

Ex2Mapper prende le righe prodotte da *Ex2JoinReducer* ed esegue un mapping unitario, in quanto la chiave settore e anno viene già definita nello step precedente.

Infine *Ex2Reducer* si occupa di aggregare i valori per settore ed anno andando a sommare tutti i volumi dei ticker, le variazioni percentuali annuali ed andandole a dividere per il numero di ticker nel settore. Viene calcolato anche quanto valga in media un ticker del settore, andando a sommare tutte le close di tutti i ticker del settore ed andando a dividere tale numero per la somma del numero di record registrati per tutti i ticker del settore.

In uscita da *Ex2Reducer* si ha il risultato finale, che viene ordinato per settore e anno.

Esercizio 2 Complex. Questa versione del Job2 è più complessa rispetto alla precedente perchè, considerando anche il campo "company", necessita di un'ulteriore elaborazione e di conseguenza di un'ulteriore fase MapReduce. Vengono utilizzati quattro Mapper:

- *Ex2HSPMapper*
- *Ex2HSMapper_Companies*
- *Ex2CompanyMapper_Companies*
- *Ex2SectorMapper_Companies*

e tre Reducer:

- *Ex2JoinReducer_Companies*
- *Ex2CompanyReducer_Companies*
- *Ex2SectorReducer_Companies*

Ex2HSPMapper è lo stesso usato in precedenza in quanto i campi di HSP necessari restano invariati.

Ex2HSMapper_Companies è un Mapper molto simile al Mapper per HS usato nella versione precedente, ma che include tra i dati rilevanti estratti anche "company".

I record estratti dai due precedenti Mapper vengono riuniti da *Ex2JoinReducer_Companies* con una logica uguale alla versione "Basic" dove le uniche differenze sono l'aggiunta della gestione del campo "company" e rimandare il calcolo della variazione percentuale al successivo Reducer.

Ex2CompanyMapper_Companies è il Mapper unitario incaricato di gestire i record con chiave "company" e "year", anche se questa viene già impostata dal reducer precedente.

Ex2CompanyReducer_Companies rappresenta la vera differenza rispetto alla precedente versione, in quanto permette di calcolare il valore iniziale e finale dell'azienda nel suo complesso, sommando i valori iniziali e finali dei ticker della stessa. Questo inoltre calcola anche la media del valore di un ticker dell'azienda andando a sommare tutte le close dei ticker e dividendo per la somma del numero di record di ogni ticker. *Ex2SectorMapper_Companies* è il Mapper unitario incaricato di gestire i record con chiave "sector" e "year", già impostata dal precedente Reducer. *Ex2SectorReducer_Companies* è l'ultimo Reducer di questo Job, incaricato di aggregare i valori per settore e anno, andando a calcolare la media del volume delle aziende del settore, la media delle variazioni percentuali del settore e la media del valore medio dei ticker nelle aziende del settore.

Esercizio 2 Complex with Combiner è l'ultima versione del Job2 sviluppata. Presenta le stesse elaborazioni della precedente, andando però ad aggiungere due Combiner al fine di ridurre l'overhead di comunicazione:

- *Ex2CombinerCompany_withCompany*
- *Ex2CombinerSector_withCompany*

Ex2CombinerCompany_withCompany si occupa di aggregare i dati prima di *Ex2CompanyReducer_Companies* senza effettuare le operazioni di calcolo della variazione percentuale dell'azienda e calcolo della media delle close dei ticker dell'azienda.

Ex2CombinerSector_withCompany si occupa di aggregare i dati prima di *Ex2SectorReducer_Companies* senza effettuare il calcolo delle medie dei valori finali.

ESERCIZIO 3

Questo esercizio presenta in Hadoop una sola implementazione. Poiché richiede di calcolare i trend delle aziende tra 2016 e 2018 ed aggregarle rispetto all'uguaglianza di tali trend, necessita dell'impiego di un join tra HSP e HS (similmente all'Esercizio 2).

Consiste in due job in chain e comprende 3 Mapper differenti:

- *Ex3HSMapper*
- *Ex3HSPMapper*
- *Ex3Mapper*

e due Reducer:

- *Ex3JoinReducer*
- *Ex3Reducer*

I Mapper *Ex2HSMapper* e *Ex2HSPMapper* sono impiegati per verificare la correttezza e la validità dei dati in input, nonché per estrarre i campi rilevanti.

Similmente al caso precedente, il join è un Reduce Side Join e fa uso di separatori personalizzati per distinguere i dati provenienti dai due Mapper. Tuttavia *Ex3JoinReducer* è stato implementato con un approccio differente dal suo corrispondente nel Job 2: tale variazione è stata adottata per fornire maggiore eterogeneità tra le strategie implementative impiegate e per risparmiare, come vedremo, un intero step MapReduce. Nello specifico:

1. *Ex3JoinReducer* comprende una classe privata *Pair*, che definisce una coppia `<String,Integer>` utile per i passi seguenti.
2. il metodo *setup* di Reducer è impiegato per inizializzare 3 mappe, definite come variabili d'istanza e che avranno il compito di memorizzare i risultati al termine del metodo *reduce*; tali mappe conterranno, rispettivamente, la quotazione iniziale, finale e la variazione di quotazione di ciascuna azienda in ogni anno.
3. nel *reduce* si calcolano anzitutto, per il singolo ticker e per ciascun anno, le date iniziali e finali in cui esso compare e i valori di close ad esse associati; si ricava inoltre dal join il nome dell'azienda cui appartiene l'azione corrente.
4. a questo punto si aggiornano le due mappe della quotazione finale e iniziale delle aziende con i corrispettivi valori ricavati dalla corrente iterazione del reduce, eventualmente sommati ai precedenti se presenti.
5. l'implementazione scelta qui prevede di impiegare metodo il *cleanup* del Reducer per effettuare alcune operazioni al termine del *reduce*: difatti tale metodo è eseguito al termine di tutte le iterazioni del *reduce*, dunque le mappe prima discusse ora contengono i valori complessivi di quotazione iniziale e finale di ciascuna azienda negli anni 2016-2018.
6. ciò permette di evitare un ulteriore step MapReduce, poiché si calcola la variazione di quotazione di ciascuna azienda nei tre anni direttamente nel *cleanup*, si concatenano i risultati come stringa e infine si forniscono in output.

Ex3Mapper ed *Ex3Reducer* sono impiegati per raggruppare tutte le aziende caratterizzate dal medesimo trend. Per fare ciò, *Ex3Mapper* rende la stringa, identificante il trend nei tre anni, chiave e azienda valore: in tal modo si sfrutta lo *shuffle & sort* per raggruppare le aziende rispetto ai trend. Infine, *Ex3Reducer* nel *reduce* verifica che il numero di aziende con un certo trend sia superiore a 1, ovvero che siano state trovate almeno due aziende con stesse variazioni di quotazione, nel qual caso le memorizza in una mappa inizializzata nel *setup*; il *cleanup* ordina tale mappa rispetto al numero di aziende (mediante un metodo della classe *Utilities*) e fornisce in output il trend in comune, il set di aziende con tale trend e il loro numero (utile per effettuare confronti tra diverse implementazioni, come discusso in Sezione 6).

5.3 Hive

INITIALIZATION

Prima di presentare le scelte implementative adottate in ciascun esercizio, facciamo qui riferimento al file *init_hive.hql*, impiegato per caricare i dataset del progetto nelle tabelle corrispondenti *historical_stock_prices* e *historical_stocks*. In particolare, si evidenzia come anche in Hive sia stato necessario tenere in considerazione come alcuni record di HS presentassero il nome dell'azienda tra "" e con una virgola all'interno. Dovendo mantenere come *field delimiter* la virgola, a causa della struttura dei dati in input, e per evitare che ciò comportasse uno split non corretto dei nomi dell'azienda, si è scelto come *row format* di *historical_stocks* la libreria di Hadoop API SerDe: essa ha permesso di definire il comportamento corretto da adottare nella situazione di cui sopra, ovvero il riconoscimento della virgola come parte del nome dell'azienda.

ESERCIZIO 1

In Hive sono state implementate tre versioni del primo esercizio:

- *ex1_hive_v1.hql*, la versione principale, descritta di seguito.
- *ex1_hive_v2.hql*, con la stessa logica di *ex1_hive_v1.hql* ma caratterizzata da un maggior numero di temporary tables: è stata sviluppata per poter verificare gli effetti dell'impiego di tabelle temporanee sulle performance.
- *ex1_hive_v3.hql*, analoga a *ex1_hive_v1.hql* ma con miglior formato output, utilizzato per produrre i risultati presentati nella relazione.

Con riferimento a *ex1_hive_v1.hql*

1. è creata la tabella temporanea *ticker_firstlastvalues*: essa memorizza, per ciascun ticker in *historical_stock_prices* compreso tra il 2008 e il 2018, la prima e l'ultima data di comparsa, il valore di close minimo e massimo e il volume medio.
2. ora si procede a ricavare, mediante join tra *ticker_firstlastvalues* e *historical_stock_prices*, la *first_close* e la *last_close*, ovvero, rispettivamente, la close in corrispondenza della prima e l'ultima data per quel ticker; si noti che il join sia eseguito con *historical_stock_prices* come secondo membro, poiché di dimensione maggiore (tale aspetto è presente in tutte le implementazioni e non sarà menzionato nuovamente).
3. infine, si crea la tabella *ex1_hive* e vi si memorizzano, per ciascun ticker, il volume medio, la close minima e massima e la variazione di quotazione, calcolata come da definizione e arrotondata: i record sono ordinati rispetto a tale variazione in modo decrescente.

ESERCIZIO 2

In Hive sono state implementate due versioni del secondo esercizio:

- *ex2basic_hive.hql*, che non tiene conto delle aziende per il calcolo dei trend dei settori.
- *ex2complex_hive.hql*, che invece considera sia ticker che aziende.

Con riferimento a *ex2basic_hive.hql*

1. è creata la tabella temporanea *ticker_sector*, che memorizza il nome del settore, il volume, la close e il giorno per ogni occorrenza di ciascun ticker, a partire dal join tra *historical_stock_prices* e *historical_stocks*, e limitato al solo periodo compreso tra il 2008 e il 2018; si noti come occorra filtrare i record con settore nullo per garantire il corretto output del job.
2. è creata la tabella temporanea *ticker_sector_year*, su cui sono salvati, per ciascun ticker e anno, la somma dei volumi del ticker, la somma delle close, la prima data di comparsa e l'ultima, e infine il numero di occorrenze, necessarie per il calcolo della quotazione giornaliera media.
3. ora, con un approccio simile a *ex1_hive_v1.hql*, si ricavano le close iniziali e finali del ticker, a partire dai join di *ticker_sector_year* con *ticker_sector*: infine si ricavano, a partire dalle definizioni fornite nelle specifiche progettuali, per ciascun settore, il volume annuale medio, la variazione di quotazione annuale media e la quotazione giornaliera media in quell'anno. Come si può osservare, abbiamo qui ragionato solo in termini di ticker e settori, non considerando le aziende.

Con riferimento a *ex2complex_hive.hql*

1. è creata la tabella temporanea *ticker_firstlastdateyear*, che memorizza, a partire dal join tra *historical_stock_prices* e *historical_stocks*, per ogni ticker in un dato anno compreso tra 2008 e 2018, i nomi di compagnia e settore, la somma dei volumi, prima e ultima data di comparsa, la somma delle close e il numero di occorrenze.
2. è creata la tabella temporanea *ticker_firstlastcloseyear* in cui, similmente a *ex2basic_hive.hql*, si ottengono i valori di close iniziali e finali per ciascun ticker in quell'anno.
3. sono calcolati, per ogni azienda in ogni anno, la somma dei volumi, delle close iniziali e finali, dei close totali e del numero complessivo di occorrenze, a partire dalle informazioni sopra ottenute per i vari ticker; è creata la tabella temporanea *company_quotationyear*, che memorizza, per ciascuna azienda in ogni anno, il volume totale precedentemente ottenuto, la variazione di quotazione annuale dell'azienda e la sua quotazione giornaliera media.
4. come si è osservato, in *ex2complex_hive.hql* si ricavano le informazioni sui settori a partire da quelle delle aziende. Infine si memorizzano nella tabella conclusiva i valori medi, per un settore in un anno, di volume, variazione di quotazione e quotazione giornaliera.

ESERCIZIO 3

In Hive sono state implementate quattro versioni del terzo esercizio:

- *ex3_hive_v1.hql*, la versione principale, descritta di seguito: essa interpreta la variazione di quotazione per un'azienda come la differenza percentuale tra le close finali totali e iniziali dei ticker di quell'azienda.
- *ex3_hive_v2.hql*, che interpreta la variazione di quotazione di un'azienda come la media delle variazioni di quotazione dei ticker di quell'azienda.
- *ex3_hive_v3.hql*, analoga a *ex3_hive_v1.hql* ma con miglior formato output, tra cui il numero di aziende con un certo trend, utilizzato per produrre i risultati presentati nella relazione. item *ex3_hive_facoltativo.hql*, simile a *ex1_hive_v1.hql* ma che tiene conto, per le aziende con trend non equivalente ad altri, di un tentativo di aggregazione basata su somiglianza tra trend (definita mediante distanza euclidea tra i trend di due aziende nei tre anni in questione).

Con riferimento a *ex3_hive_v1.hql*

1. è creata la tabella temporanea *ticker_firstlastdateyear*, in cui si memorizza, a partire da *historical_stock_prices*, per ciascun ticker negli anni compresi tra 2016 e 2018, la prima e ultima data di comparsa del ticker in quell'anno.
2. è creata la tabella temporanea *ticker_quotationyear*, in cui si memorizza, per ciascun ticker e anno, le close iniziali e finali, ottenute a partire dal join di *ticker_firstlastdateyear* con *historical_stock_prices*
3. dal join di *ticker_quotationyear* con *historical_stocks* si ricavano, per ciascuna azienda in un anno, le close iniziali e finali, definite come somme, rispettivamente, delle close iniziali e finali dei ticker appartenenti a quell'azienda
4. è creata la tabella temporanea *company_quotationyear*, che memorizza, per ogni azienda, il set contenente i trend di quell'azienda nei tre anni 2016, 2017 e 2018.
5. infine si converte tale set in una stringa e si effettua una *collect* rispetto ad essa, di modo da ottenere tutte le aziende con medesimo trend in tutti e tre gli anni. Nella tabella conclusiva *ex3_hive* si memorizza la stringa ottenuta da tale *collect* e il trend delle aziende in essa presenti: i risultati sono filtrati affinché siano presenti solo insiemi di aziende superiori a 1 con un certo trend comune, e sono ordinati in modo decrescente rispetto a tale valore. Nella versione *ex3_hive_v3.hql* tale numero è anche parte dell'output finale.

Infine, alcune osservazioni su *ex3_hive_facoltativo.hql*.

- la differenza rispetto a *ex3_hive_v1.hql* consiste nel fatto che si distinguono i gruppi di aziende con trend equivalente da quelli di aziende con trend simile. Per calcolare la similarità tra due trend, si considera la distanza euclidea su tre coordinate (corrispondenti alle variazioni di quotazione nei tre anni) e si raggruppano inizialmente le aziende a coppie nel caso in cui il valore di similarità tra i trend sia superiore a un certo *threshold* modificabile.
- una volta definito il valore di similarità, si procede in maniera analoga al caso precedente e infine si uniscono i risultati ottenuti dalla ricerca di equivalenza tra trend e quelli ricavati dalla ricerca di similarità.

5.4 Spark

ESERCIZIO 1

L'implementazione del Job 1 in Spark non presenta differenti varianti. Sono state scritte diverse Lambda Functions personalizzate, che sono descritte nel seguito:

- *checkLine*, è una funzione di filtraggio, viene utilizzata per rimuovere le righe associate a dati sporchi o ad una data esterna all'intervallo richiesto.
- *prepareValues*, è un'operazione di mapping, utilizzata solo per replicare i valori date e close al fine di produrre un formato adatto all'elaborazione richiesta dal reducer successivo.
- *reducer*, si occupa di trovare prima ed ultima data utile di ogni ticker ed il valore di close ad esse associato, oltre a trovare i valori minimi e massimi di close ed il volume totale.
- *produceResults*, è un mapper incaricato di effettuare le aggregazioni richieste per calcolare la variazione percentuale del ticker ed il volume medio, producendo in uscita un'istanza Comparable di "Result.Ex1", al fine di poter ordinare i risultati.

ESERCIZIO 2

Questo esercizio è stato presentato in due versioni:

- *Ex2_spark_distributedRow*
- *Ex2_spark_singleRow*

Le due versioni utilizzano le stesse funzioni ed implementano entrambe il corrispettivo della versione Complex implementata in Hadoop e Hive, ma si differenziano per l'utilizzo di risultati intermedi inseriti nello heap delle variabili. Questo sarà rilevante ai soli fini dello studio della differenza di performance tra i due.

Le funzioni che vengono utilizzate sono:

- *checkInputHSP*, è una funzione di filtraggio, viene utilizzata per rimuovere le righe associate a dati sporchi o ad una data esterna all'intervallo richiesto.
- *checkInputHS*, come la precedente, si occupa del filtraggio di righe associate a dati sporchi.
- *prepareValuesHS*, mappa i dati in ingresso di HS, prelevando quelli rilevanti per il job.
- *prepareValuesHSP*, come la precedente, per HSP.
- *reorganizeValuesAfterJoin*, viene utilizzata subito dopo aver eseguito il join dei valori prelevati da HS e HSP, per preparare i valori per il successivo reducer, infine imposta come chiave (ticker,anno).
- *reduce_sumVolumeTicker_findFirstLastClose*, vengono effettuate le prime aggregazioni di dati ovvero:
 - somma di tutte le close
 - trovare i primi valori delle close di ogni ticker e anno
- *map_fromTickerToCompany*, si occupa soltanto del cambio di chiave da (ticker,anno) a (compagnia,anno).
- *reduce_sumVolumeCompany_sumFirstLastCloseCompany_sumCloseSumCounterCompany*, si occupa di alcune aggregazioni:
 - somma dei volumi
 - somma dei primi valori dei close
 - somma degli ultimi valori dei close
 - somma di tutte le close
 - somma del numero di record di ogni ticker
- *map_varYearCompany_dailyQuotCompany_fromCompanyToSector* si occupa di calcolare la variazione percentuale ed il valore medio della quotazione giornaliera dell'azienda. In seguito cambia la chiave in (set-tore,anno).
- *reduce_sumVolumes_sumVars_sumQuots* si occupa di sommare volumi, variazioni e quotazioni, oltre a contare quante aziende ci siano nel settore.
- *map_avgSector* calcola i risultati finali dividendo le somme precedentemente calcolate per il numero di aziende.

ESERCIZIO 3

L'implementazione di questo esercizio non presenta varianti. È stata implementata senza produrre risultati intermedi ed utilizza le funzioni:

- *checkInputHS*, filtra righe con valori sporchi.
- *checkInputHSP*, è una funzione di filtraggio, viene utilizzata per rimuovere le righe associate a dati sporchi o ad una data all'esterno dell'intervallo richiesto.
- *prepareValuesHS*, mappa i dati in ingresso di HS, prelevando quelli rilevanti al job.
- *prepareValuesHSP*, come la precedente, per HSP.
- *reorganizeValuesAfterJoin* viene chiamata subito dopo aver eseguito il join tra i valori di HS e HSP, ed è utilizzata per replicare alcuni valori al fine di prepararli ad un'elaborazione effettuata nel reducer successivo, impostando come chiave (ticker, anno).
- *reduce_findFirstLastCloses* si occupa di trovare i valori iniziali e finali delle close di ogni ticker di ogni anno.
- *map_fromTickerToCompany* effettua un mapping cambiando la chiave a (ticker,anno) a (compagnia,anno).
- *reduce_sumFirstLastCloses* somma i rispettivi valori di close iniziali e finali per ogni azienda, al fine di calcolare quanto valga l'azienda ad inizio e fine di ogni anno.
- *map_calculateVarPercCompanyYear_changeKeyToCompany* calcola la variazione percentuale annua di ogni azienda, cambia la chiave ad azienda ed inserisce il valore calcolato nell'anno cui si riferisce.
- *reduce_unifyTrends* unisce le variazioni percentuali di anni differenti della stessa azienda.
- *checkAllYearPresent* costituisce un'operazione di filtraggio, nella quale si controlla che siano stati calcolate le variazioni percentuali di tutte e tre gli anni.
- *invertKey_fromCompany_toVarYear* cambia la chiave da compagnia a trend, dove il trend è l'insieme delle variazioni percentuali per gli anni 2016, 2017 e 2018.
- *reduce_companySameTrend* aggrega tutte le aziende con lo stesso trend.
- *mapByNumberSimilarCompaniesTrend* calcola quante siano le aziende con lo stesso trend e cambia la chiave da trend a numero di similarità, così da poter fornire un ordinamento decrescente per numero di similarità.

6 Presentazione Risultati

In questa Sezione sono illustrati i primi record appartenenti ai risultati delle esecuzioni nelle tre tecnologie. L'obiettivo è mostrare l'equivalenza dei record e del formato in output. I risultati delle versioni implementate sono visibili nella cartella *resultsFirst10* del progetto.

HADOOP EX1	HIVE EX1	SPARK EX1
TICKER,VARIAZIONE_QUOTAZIONE_%,PREZZO_MIN,PREZZO_MAX,VOLUME_MEDIO EAF,267757%,0.002,22.37,1245139 ORGS,217416%,0.00314,19.8,8570 PUB,179900%,0.009,135.0,34449 RMP,121082%,0.03,78.54985,120487 CTZ,120300%,0.005,26.73,52050 CCD,111250%,0.015,25.55,105416 SAB,110429%,1.3985,318800.0,1695378 KE,99400%,0.015,22.2,73249 LN,96700%,0.015,49.63,394344 GHG,64850%,0.005,24.31,607659	EAF,267757%,0.002,22.37,1245139 ORGS,217416%,0.00314,19.8,8570 PUB,179900%,0.009,135.0,34449 RMP,121082%,0.03,78.54985,120487 CTZ,120300%,0.005,26.73,52050 CCD,111250%,0.015,25.55,105416 SAB,110429%,1.3985,318800.0,1695378 KE,99400%,0.015,22.2,73249 LN,96700%,0.015,49.63,394344 GHG,64850%,0.005,24.31,607659	TICKER,VARIAZIONE_QUOTAZIONE_%,PREZZO_MIN,PREZZO_MAX,VOLUME_MEDIO EAF,267757%,0.002,22.37,1245139 ORGS,217416%,0.00314,19.8,8570 PUB,179900%,0.009,135.0,34449 RMP,121082%,0.03,78.54985,120487 CTZ,120300%,0.005,26.73,52050 CCD,111250%,0.015,25.55,105416 SAB,110429%,1.3985,318800.0,1695378 KE,99400%,0.015,22.2,73249 LN,96700%,0.015,49.63,394344 GHG,64850%,0.005,24.31,607659

Figure 12: Esercizio 1 in Hadoop, Hive, Spark

HADOOP EX2 BASIC	HIVE EX2 BASIC
SETTORE,ANNO,VOLUME_ANNUALE_MEDIO,VARIAZIONE_ANNUALE_MEDIA_%,QUOTAZIONE_GIORNALIERA_MEDIA BASIC INDUSTRIES,2008,560950485,-42.05%,40.0 BASIC INDUSTRIES,2009,589384165,82.32%,25.7 BASIC INDUSTRIES,2010,476571424,32.89%,32.35 BASIC INDUSTRIES,2011,448450099,-11.34%,42.74 BASIC INDUSTRIES,2012,377483105,8.19%,42.37 BASIC INDUSTRIES,2013,367271657,18.85%,133.95 BASIC INDUSTRIES,2014,350472233,-3.35%,109.44 BASIC INDUSTRIES,2015,388588042,123.46%,40.48 BASIC INDUSTRIES,2016,470968318,47.54%,31.4 BASIC INDUSTRIES,2017,383340553,16.69%,35.56 BASIC INDUSTRIES,2018,232851237,-4.05%,37.82	BASIC INDUSTRIES,2008,560950485,-42.05,40.0 BASIC INDUSTRIES,2009,589384165,82.32,25.7 BASIC INDUSTRIES,2010,476571424,32.89,32.35 BASIC INDUSTRIES,2011,448450099,-11.34,42.74 BASIC INDUSTRIES,2012,377483105,8.19,42.37 BASIC INDUSTRIES,2013,367271657,18.85,133.95 BASIC INDUSTRIES,2014,350472233,-3.35,109.44 BASIC INDUSTRIES,2015,388588042,123.46,40.48 BASIC INDUSTRIES,2016,470968318,47.54,31.4 BASIC INDUSTRIES,2017,383340553,16.69,35.56 BASIC INDUSTRIES,2018,232851237,-4.05,37.82

Figure 13: Esercizio 2 Basic in Hadoop e Hive

HADOOP EX2 COMPS	HIVE EX2 COMPLEX	SPARK EX2
SETTORE,ANNO,VOLUME_ANNUALE_MEDIO, VARIAZIONE_ANNUALE_MEDIA_%,QUOTAZIONE_GIORNALIERA_MEDIA BASIC INDUSTRIES,2008,560950485,-42.05%,604.1 BASIC INDUSTRIES,2009,589384165,82.32%,224.26 BASIC INDUSTRIES,2010,476571424,32.89%,32.13 BASIC INDUSTRIES,2011,448450099,-11.34%,443.95 BASIC INDUSTRIES,2012,377483105,8.19%,124.53 BASIC INDUSTRIES,2013,367271657,18.85%,238.53 BASIC INDUSTRIES,2014,350472233,-3.35%,233.44 BASIC INDUSTRIES,2015,390174115,124.0%,41.16 BASIC INDUSTRIES,2016,472822524,47.85%,30.67 BASIC INDUSTRIES,2017,384814939,16.81%,35.03 BASIC INDUSTRIES,2018,233739982,-4.02%,37.55	BASIC INDUSTRIES,2008,560950485,-42.05,604.1 BASIC INDUSTRIES,2009,589384165,82.32,224.26 BASIC INDUSTRIES,2010,476571424,32.89,32.13 BASIC INDUSTRIES,2011,448450099,-11.34,443.94 BASIC INDUSTRIES,2012,377483105,8.19,124.53 BASIC INDUSTRIES,2013,367271657,18.85,238.53 BASIC INDUSTRIES,2014,350472233,-3.35,233.44 BASIC INDUSTRIES,2015,390174115,124.0,41.16 BASIC INDUSTRIES,2016,472822524,47.85,30.67 BASIC INDUSTRIES,2017,384814939,16.81,35.03 BASIC INDUSTRIES,2018,233739982,-4.02,37.55	SETTORE,ANNO,VOLUME_ANNUALE_MEDIO, VARIAZIONE_ANNUALE_MEDIA_%,QUOTAZIONE_GIORNALIERA_MEDIA BASIC INDUSTRIES,2008,560950485,-42.05%,604.1 BASIC INDUSTRIES,2009,589384165,82.32%,224.26 BASIC INDUSTRIES,2010,476571424,32.89%,32.13 BASIC INDUSTRIES,2011,448450099,-11.34%,443.95 BASIC INDUSTRIES,2012,377483105,8.19%,124.53 BASIC INDUSTRIES,2013,367271657,18.85%,238.53 BASIC INDUSTRIES,2014,350472233,-3.35%,233.44 BASIC INDUSTRIES,2015,390174115,124.0%,41.16 BASIC INDUSTRIES,2016,472822524,47.85%,30.67 BASIC INDUSTRIES,2017,384814939,16.81%,35.03 BASIC INDUSTRIES,2018,233739982,-4.02%,37.55

Figure 14: Esercizio 2 Complex in Hadoop, Hive, Spark

Si osservi come per i primi tre casi, (rispettivamente illustrati in Figure 12, 13 e 14), i risultati siano equivalenti sia in termini di valori che di ordinamento. Per quanto concerne gli ultimi due, ovvero quelli relativi all'esercizio 3 e alla sua versione facoltativa, non si può dire altrettanto. In effetti, benché i risultati siano coerenti, distinte tecnologie forniscono distinti ordinamenti, a parità del numero di aziende caratterizzate dal medesimo trend. Si è pensato dunque di integrare l'output di questo esercizio con il numero di aziende con lo stesso trend, così da semplificarne il confronto visivo. Si osservi inoltre che per la Figure 16 siano stati volutamente omissi alcuni risultati intermedi, al fine di mostrare anche record con i valori di similarità.

HADOOP EX3
n. aziende trend comune,(AZIENDE_CON_TREND_COMUNE):, 2016: VAR_ANN_%,2017: VAR_ANN_%,2018: VAR_ANN_% 10,{EATON VANCE NEXTSHARES TRUST,VANGUARD SHORT-TERM TREASURY ETF,WISDOMTREE BARCLAYS INTEREST RATE HEDGED U.S. AGGREGATE BOND F,ISHARES 1-3 YEAR CREDIT BOND ETF,ISHARES SHORT TREASURY BOND ETF,IVY NEXTSHARES,FIRST TRUST ENHANCED SHORT MATURITY ETF,POWERSHARES VARIABLE RATE INVESTMENT GRADE PORTFOLIO,ISHARES 1-3 YEAR TREASURY BOND ETF,ISHARES 0-5 YEAR INVESTMENT GRADE CORPORATE BOND ETF}: 2016:0%,2017:0%,2018:0% 7,{VANGUARD MORTGAGE-BACKED SECURITIES ETF,VANGUARD INTERMEDIATE-TERM TREASURY ETF,VANGUARD SHORT-TERM CORPORATE BOND ETF,ISHARES 3-7 YEAR TREASURY BOND ETF,ISHARES GNMA BOND ETF,POWERSHARES LADDERRITE 0-5 YEAR CORPORATE BOND PORTFOLIO,ISHARES CORE 1-5 YEAR USD BOND ETF}: 2016:0%,2017:0%,2018:-1% 3,{EATON VANCE HIGH INCOME 2021 TARGET TERM TRUST,ISHARES INTERMEDIATE CREDIT BOND ETF,ETF SERIES SOLUTIONS TRUST VIDENT CORE U.S. BOND STRATEGY FUND}: 2016:0%,2017:0%,2018:-2% 2,{SYPRIS SOLUTIONS, INC.},VIRTUS LIFESCI BIOTECH CLINICAL TRIALS ETF}: 2016:-33%,2017:53%,2018:14% 2,{KAYNE ANDERSON ENERGY DEVELOPMENT COMPANY,"COMSTOCK RESOURCES, INC."}: 2016:10%,2017:-11%,2018:3% 2,{STANTEC INC,NUVEEN DIVERSIFIED DIVIDEND AND INCOME FUND}: 2016:4%,2017:9%,2018:-9% 2,{SPDR DORSEY WRIGHT FIXED INCOME ALLOCATION ETF,INVESCO TRUST FOR INVESTMENT GRADE MUNICIPALS}: 2016:-4%,2017:3%,2018:-7% 2,{LOCKHEED MARTIN CORPORATION,BOOKING HOLDINGS INC.}: 2016:17%,2017:26%,2018:0% 2,{PRICESMART, INC.},NUVEN MORTGAGE OPPORTUNITY TERM FUND 2}: 2016:3%,2017:2%,2018:-2% 2,{CENTERPOINT ENERGY, INC.},SCORPIO BULKERS INC.}: 2016:34%,2017:15%,2018:0%
HIVE EX3
10,{EATON VANCE NEXTSHARES TRUST,FIRST TRUST ENHANCED SHORT MATURITY ETF,ISHARES 0-5 YEAR INVESTMENT GRADE CORPORATE BOND ETF,ISHARES 1-3 YEAR CREDIT BOND ETF,ISHARES 1-3 YEAR TREASURY BOND ETF,ISHARES SHORT TREASURY BOND ETF,IVY NEXTSHARES,POWERSHARES VARIABLE RATE INVESTMENT GRADE PORTFOLIO,VANGUARD SHORT-TERM TREASURY ETF,WISDOMTREE BARCLAYS INTEREST RATE HEDGED U.S. AGGREGATE BOND F},2016:0%,2017:0%,2018:0% 7,{ISHARES 3-7 YEAR TREASURY BOND ETF,ISHARES CORE 1-5 YEAR USD BOND ETF,ISHARES GNMA BOND ETF,POWERSHARES LADDERRITE 0-5 YEAR CORPORATE BOND PORTFOLIO,VANGUARD INTERMEDIATE-TERM TREASURY ETF,VANGUARD MORTGAGE-BACKED SECURITIES ETF,VANGUARD SHORT-TERM CORPORATE BOND ETF},2016:0%,2017:0%,2018:-1% 3,{EATON VANCE HIGH INCOME 2021 TARGET TERM TRUST,ETF SERIES SOLUTIONS TRUST VIDENT CORE U.S. BOND STRATEGY FUND,ISHARES INTERMEDIATE CREDIT BOND ETF},2016:0%,2017:0%,2018:-2% 2,{SYPRIS SOLUTIONS, INC.},VIRTUS LIFESCI BIOTECH CLINICAL TRIALS ETF}: 2016:-33%,2017:53%,2018:14% 2,{KAYNE ANDERSON ENERGY DEVELOPMENT COMPANY,"COMSTOCK RESOURCES, INC."}: 2016:10%,2017:-11%,2018:3% 2,{STANTEC INC,NUVEEN DIVERSIFIED DIVIDEND AND INCOME FUND}: 2016:4%,2017:9%,2018:-9% 2,{SPDR DORSEY WRIGHT FIXED INCOME ALLOCATION ETF,INVESCO TRUST FOR INVESTMENT GRADE MUNICIPALS}: 2016:-4%,2017:3%,2018:-7% 2,{LOCKHEED MARTIN CORPORATION,BOOKING HOLDINGS INC.}: 2016:17%,2017:26%,2018:0% 2,{PRICESMART, INC.},NUVEN MORTGAGE OPPORTUNITY TERM FUND 2}: 2016:3%,2017:2%,2018:-2% 2,{CENTERPOINT ENERGY, INC.},SCORPIO BULKERS INC.}: 2016:34%,2017:15%,2018:0%
SPARK EX3
n. aziende trend comune,(AZIENDE_CON_TREND_COMUNE):,2016: VAR_ANN_%,2017: VAR_ANN_%,2018: VAR_ANN_% 10,{EATON VANCE NEXTSHARES TRUST,VANGUARD SHORT-TERM TREASURY ETF,ISHARES SHORT TREASURY BOND ETF,IVY NEXTSHARES,ISHARES 1-3 YEAR CREDIT BOND ETF,POWERSHARES VARIABLE RATE INVESTMENT GRADE PORTFOLIO,ISHARES 1-3 YEAR TREASURY BOND ETF,ISHARES 0-5 YEAR INVESTMENT GRADE CORPORATE BOND ETF,WISDOMTREE BARCLAYS INTEREST RATE HEDGED U.S. AGGREGATE BOND F,FIRST TRUST ENHANCED SHORT MATURITY ETF}: ,2016: 0%,2017: 0%,2018: 0% 7,{ISHARES 3-7 YEAR TREASURY BOND ETF,ISHARES CORE 1-5 YEAR USD BOND ETF,ISHARES GNMA BOND ETF,VANGUARD MORTGAGE-BACKED SECURITIES ETF,POWERSHARES LADDERRITE 0-5 YEAR CORPORATE BOND PORTFOLIO,VANGUARD INTERMEDIATE-TERM TREASURY ETF,VANGUARD SHORT-TERM CORPORATE BOND ETF}, ,2016: 0%,2017: 0%,2018: -1% 3,{ISHARES INTERMEDIATE CREDIT BOND ETF,EATON VANCE HIGH INCOME 2021 TARGET TERM TRUST,ETF SERIES SOLUTIONS TRUST VIDENT CORE U.S. BOND STRATEGY FUND}, ,2016: 0%,2017: 0%,2018: -2% 2,{COMSTOCK RESOURCES, INC.},KAYNE ANDERSON ENERGY DEVELOPMENT COMPANY}: ,2016: 10%,2017: -11%,2018: 3% 2,{JAZZ INC.,"WESTERN GAS PARTNERS, LP"}}, ,2016: 21%,2017: -20%,2018: 5% 2,{NUVEEN MORTGAGE OPPORTUNITY TERM FUND 2,"PRICESMART, INC."}: ,2016: 3%,2017: 2%,2018: -2% 2,{SYPRIS SOLUTIONS, INC.},VIRTUS LIFESCI BIOTECH CLINICAL TRIALS ETF}: ,2016: -33%,2017: 53%,2018: 14% 2,{FIRST TRUST INTERNATIONAL MULTI-ASSET DIVERSIFIED INCOME INDEX,FIRST TRUST RIVERFRONT DYNAMIC ASIA PACIFIC ETF}: ,2016: -1%,2017: 16%,2018: -8% 2,{STORE CAPITAL CORPORATION,SEMPRA ENERGY}: ,2016: 8%,2017: 5%,2018: 10% 2,{TEMPLETON GLOBAL INCOME FUND, INC.},REGENCY CENTERS CORPORATION}: ,2016: 2%,2017: 0%,2018: -5%

Figure 15: Esercizio 3 in Hadoop, Hive, Spark

HIVE EX3 STANDARD
10,{EATON VANCE NEXTSHARES TRUST,FIRST TRUST ENHANCED SHORT MATURITY ETF,ISHARES 0-5 YEAR INVESTMENT GRADE CORPORATE BOND ETF,ISHARES 1-3 YEAR CREDIT BOND ETF,ISHARES 1-3 YEAR TREASURY BOND ETF,ISHARES SHORT TREASURY BOND ETF,IVY NEXTSHARES,POWERSHARES VARIABLE RATE INVESTMENT GRADE PORTFOLIO,VANGUARD SHORT-TERM TREASURY ETF,WISDOMTREE BARCLAYS INTEREST RATE HEDGED U.S. AGGREGATE BOND F},2016:0%,2017:0%,2018:0% 7,{ISHARES 3-7 YEAR TREASURY BOND ETF,ISHARES CORE 1-5 YEAR USD BOND ETF,ISHARES GNMA BOND ETF,POWERSHARES LADDERRITE 0-5 YEAR CORPORATE BOND PORTFOLIO,VANGUARD INTERMEDIATE-TERM TREASURY ETF,VANGUARD MORTGAGE-BACKED SECURITIES ETF,VANGUARD SHORT-TERM CORPORATE BOND ETF},2016:0%,2017:0%,2018:-1% 3,{EATON VANCE HIGH INCOME 2021 TARGET TERM TRUST,ETF SERIES SOLUTIONS TRUST VIDENT CORE U.S. BOND STRATEGY FUND,ISHARES INTERMEDIATE CREDIT BOND ETF},2016:0%,2017:0%,2018:-2% 2,{SYPRIS SOLUTIONS, INC.},VIRTUS LIFESCI BIOTECH CLINICAL TRIALS ETF}: 2016:-33%,2017:53%,2018:14% 2,{KAYNE ANDERSON ENERGY DEVELOPMENT COMPANY,"COMSTOCK RESOURCES, INC."}: 2016:10%,2017:-11%,2018:3% 2,{STANTEC INC,NUVEEN DIVERSIFIED DIVIDEND AND INCOME FUND}: 2016:4%,2017:9%,2018:-9% 2,{SPDR DORSEY WRIGHT FIXED INCOME ALLOCATION ETF,INVESCO TRUST FOR INVESTMENT GRADE MUNICIPALS}: 2016:-4%,2017:3%,2018:-7% 2,{LOCKHEED MARTIN CORPORATION,BOOKING HOLDINGS INC.}: 2016:17%,2017:26%,2018:0% 2,{PRICESMART, INC.},NUVEN MORTGAGE OPPORTUNITY TERM FUND 2}: 2016:3%,2017:2%,2018:-2% 2,{CENTERPOINT ENERGY, INC.},SCORPIO BULKERS INC.}: 2016:34%,2017:15%,2018:0%
HIVE EX3 FACOLTATIVO
{ADVENT/CLAYMORE ENHANCED GROWTH & INCOME FUND;BLACKROCK MUNIYIELD INVESTMENT QUALITYFUND},2016:0%;2017:0%;2018:-7% {ALLIANTZG CONVERTIBLE & INCOME FUND II;TPG SPECIALTY LENDING, INC.},2016:13%;2017:7%;2018:0% {AMERICA FIRST MULTIFAMILY INVESTORS, L.P.;FLEXSHARES REAL ASSETS ALLOCATION INDEX FUND},2016:6%;2017:12%;2018:-2% ... (all results with same trends as always) ... {{ACACIA COMMUNICATIONS, INC.},{OASIS PETROLEUM INC.},{MINERVA NEUROSCIENCES, INC.},{BIOTELEMETRY, INC.},{RIVERNORTH/DOUBLELINE STRATEGIC OPPORTUNITY FUND, INC.}},36681 {{ACORN INTERNATIONAL, INC.},{TAKE-TWO INTERACTIVE SOFTWARE, INC.},{CASA SYSTEMS, INC.}},19510 {{ADESTO TECHNOLOGIES CORPORATION},{CASA SYSTEMS, INC.}},19411 {{ADVANCED MICRO DEVICES, INC.},{LINE CORPORATION}},112957 {{ADVANCED MICRO DEVICES, INC.},{RIVERNORTH/DOUBLELINE STRATEGIC OPPORTUNITY FUND, INC.}},36471 {{ADVANCED MICRO DEVICES, INC.},{THE ORGANICS ETF}},30216

Figure 16: Esercizio 3 Standard e Facoltativo in Hive

7 Test Performance

Sono stati eseguiti molteplici test di performance sui job richiesti.

Tra le versioni implementate degli esercizi e sopra discusse, sono state considerate per la fase di testing solo quelle che presentano differenze rilevanti a livello implementativo, quali il numero di job MapReduce in chain o la presenza o meno del combiner: versioni invece che si distinguessero solo per il formato in output non sono state soggette a test distinti poiché considerati non rilevanti per il lavoro in questione.

La modalità con la quale essi sono stati eseguiti è attraverso il comando "time" della shell Unix, riportando il valore associato a "real", ovvero il tempo trascorso dall'inizio alla fine della chiamata del comando. Ogni esecuzione è stata ripetuta più volte ed è stata effettuata una media dei valori rilevati. In tutti i test di seguito presentati, il tempo è espresso in secondi.

Si rimanda al seguente link per i tempi precisi di ciascun test qui descritto:

<https://docs.google.com/spreadsheets/d/14P0k3KyA4DZzq8ee5VYf8z8aSvbKepmsuG7xBNNZC7E/edit?usp=sharing>.

I test effettuati mirano alla verifica delle performance e della scalabilità delle tecnologie utilizzate, ovvero:

- Hadoop 3.2.1
- Hive 2.3.6 e 3.1.2
- Spark 3.0.0

A tal fine si sono utilizzati diversi ambienti di esecuzione quali:

- VM con buon livello di hardware dedicato: 4 processori, 8GB di RAM, 30GB HDD.
- VM con scarso livello di hardware dedicato: 2 processori, 4GB di RAM, 30GB HDD.
- 3 istanze di nodi m5.xlarge su cluster AWS.

Al fine di verificare la scalabilità dei job creati, sono stati replicati 3 volte i dati di input originari, per poter osservare il nuovo comportamento. I valori riferiti a questo test di scalabilità si riferiscono ad esecuzioni effettuate sulla VM con buon livello di hardware dedicato.

A seguito dell'esecuzione dei test, sono stati creati dei grafici, riportati nell'ultima pagina di questo documento, per poter visualizzare e confrontare diversi aspetti.

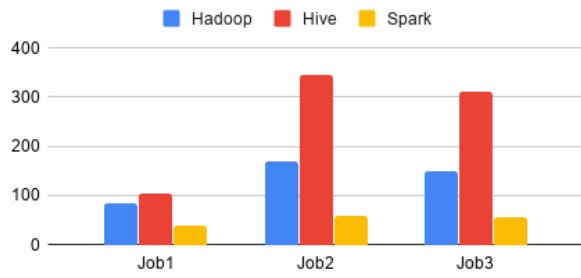
- (a);(b);(c); Confronto dei tempi di esecuzione dei vari job proposti, al variare della tecnologia utilizzata e al variare dell'ambiente di esecuzione sul quale la tecnologia viene utilizzata, per l'esecuzione dello stesso job. Questo permette di osservare che:
 - Il Job2 è quello che richiede un'elaborazione maggiore, seguito dal Job3, mentre il Job1 è risultato il più rapido.
 - La tecnologia più efficiente sia Spark, a prescindere dalla tipologia del job e della macchina utilizzata, almeno per quanto riguarda le macchine da noi utilizzate ed i job sviluppati.
- (d); Confronto dei tempi di esecuzione in hadoop del Job 2 se si utilizzino o meno dei Combiner per aumentare la sua efficienza. È interessante osservare che le prestazioni siano quasi invariate, questo è dovuto alla natura del dataset ed alla tipologia di elaborazioni richieste dal job stesso.
- (e); Confronto dei tempi di esecuzione tra le implementazioni Basic e Complex del job2 in Hadoop e Hive. Prevedibilmente, la versione più semplice, che richiede uno step Map-Reduce in meno rispetto alla sua versione complessa, presenta tempi inferiori.
- (f); Confronto dei tempi di esecuzione tra le implementazioni SingleRow e DistributedRow di Spark. È possibile osservare come, su un input normale, gestire dei risultati intermedi porti a dei benefici in termini di tempo, mentre su input di grandi dimensioni Spark riesca a gestire efficientemente l'esecuzione, anche con prestazioni leggermente migliori, qualora le funzioni fossero utilizzate in un'unica riga.
- (g); Confronto dei tempi di esecuzione in Hadoop al variare della dimensione dei dati in input o della macchina sulla quale viene eseguito il Job. Da questi risultati è possibile effettuare diverse considerazioni:
 - L'assenza del dato per il Job1 eseguito con un input grande è dovuto al continuo fallimento dello stesso in quanto richiedeva più memoria di quanta ne fosse disponibile nella VM utilizzata e, anche ottimizzando al massimo gli spazi a disposizione, non si è riuscito ad ottenere un'elaborazione completa, che si bloccava sempre in fase di merge finale dei risultati.
 - Per un input normale, al variare delle prestazioni della VM, non è percepibile una differenza di tempistiche, questo implica che Hadoop non ottimizzi automaticamente l'esecuzione sulla base delle risorse disponibili, e che queste debbano essere gestite manualmente tramite un file di configurazione.
 - Le performance in termini di tempi di esecuzione su cluster, per l'istanza selezionata, sono risultate superiori rispetto all'esecuzione su singola VM, anche se con prestazioni limitate.
 - Come ci si poteva aspettare, essendo l'elaborazione del job3 più semplice rispetto a quella del Job2, l'incremento di tempistiche dovuto a dimensioni di input crescente per il job3 è inferiore rispetto a quelle per il job2.

- (h); Confronto dei tempi di esecuzione in Hive al variare della dimensione dei dati in input o della macchina sul quale viene eseguito il Job. Per Hive è possibile effettuare le stesse considerazioni fatte su Hadoop, con alcune differenze:
 - Il Job1 si è rilevato essere più efficiente su Cluster.
 - Il job1, al variare della dimensione dell'input, ha subito un incremento di prestazioni superiore, dovuto alla particolarità del Job stesso.
- (i); Confronto dei tempi di esecuzione in Spark al variare della dimensione dei dati in input o della macchina sul quale viene eseguito il Job. Le considerazioni che è possibile effettuare su Spark sono:
 - Il Job1 e Job3 hanno risposto efficientemente al variare della dimensione in input e della macchina sul quale viene eseguito.
 - Il Job2 ha subito un forte incremento delle tempistiche al crescere dell'input, dovuto alla complessità del job stesso, che richiede più fasi di elaborazione e quindi un corrispettivo overhead di comunicazione.
 - La differenza di prestazioni ottenuta tra VM con un buon livello di hardware dedicato e VM con livello di hardware inferiore è più apprezzabile rispetto alle precedenti. Questo implica che Spark sia più efficiente quando abbia a disposizione più RAM e processori, e che sia in grado di sfruttare tale differenza. Da notare che anche Spark permetta di gestire manualmente, tramite file di configurazione, la quantità di memoria RAM utilizzabile.
- (j); Confronto nell'esecuzione in Hive delle versioni dei Job 1 e Job 3 con poche e con molte tabelle temporanee, al fine di comprendere l'impatto di queste ultime sulle prestazioni. Dai risultati si può osservare che, fatta eccezione per il test del Job 3 con input incrementato, la presenza di tabelle temporanee non abbia mostrato conseguenze in termini di tempo impiegato per l'esecuzione.

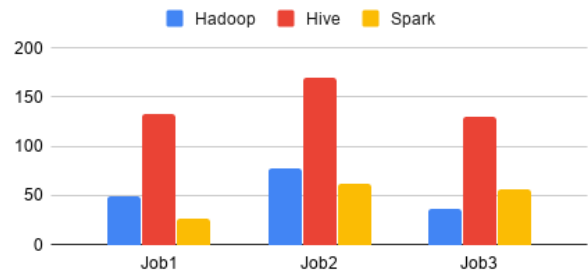
8 Conclusioni

Questa relazione ha descritto le varie fasi del lavoro svolto come primo progetto del corso "Big Data". L'obiettivo di questa attività è stato l'impiego delle tecnologie Hadoop, Hive e Spark per lo svolgimento di job MapReduce applicati a scenari. Inoltre è stata caratterizzata da una consistente implementazione in varie versioni, sulla base delle specifiche fornite e delle successive discussioni con il docente, così come da una intensa attività di testing di queste ultime, al fine di comprendere vantaggi e svantaggi dei vari approcci. Dall'esperienza fornitaci dallo svolgimento di questo progetto, è possibile sottolineare i seguenti aspetti:

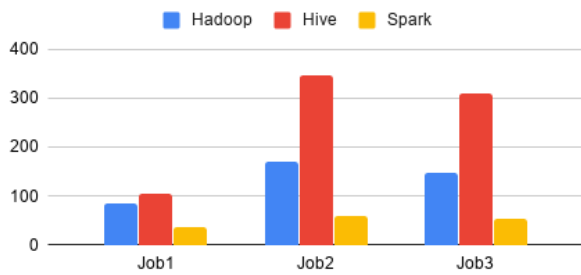
- **Spark**, con riferimento ai risultati ottenuti durante la fase di testing, risulta che Spark sia la tecnologia più efficiente. Essa spicca anche per la facilità e velocità di scrittura del codice, della documentazione e della leggibilità del codice prodotto.
- **Hadoop**, presenta l'ottimo paradigma Map Reduce, permette di ottenere buoni risultati in termini di performance e risulta abbastanza semplice scrivere e documentare del codice.
- **Hive**, si è rivelata essere la peggiore, paragonata alle altre due tecnologie sopracitate, sia in termini di performance, sia in termini di facilità di scrittura del codice in quanto, anche se SQL è un linguaggio estremamente familiare, risulta meno intuitivo scrivere delle interrogazioni per elaborazioni complesse. Di contro presenta il vantaggio dell'estrema sinteticità, oltre alla già citata familiarità con lo standard SQL ed alla facilità di documentare le interrogazioni prodotte.



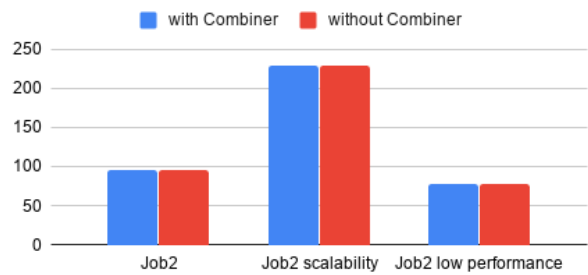
(a) Confronto tecnologie - VM prestante



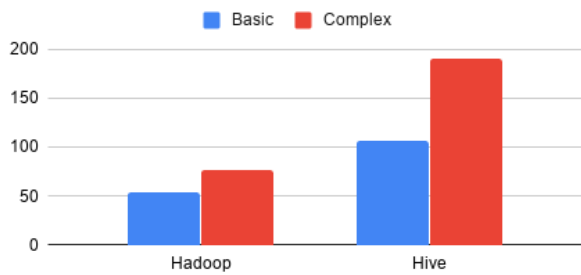
(b) Confronto tecnologie - VM low performance



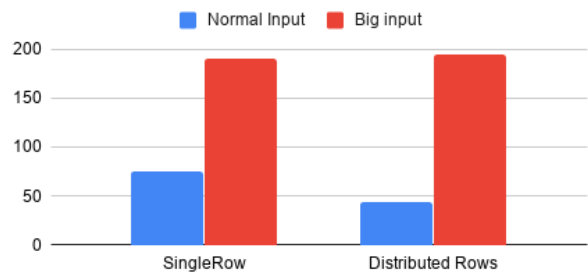
(c) Confronto tecnologie - Cluster AWS



(d) Job2 con e senza Combiners



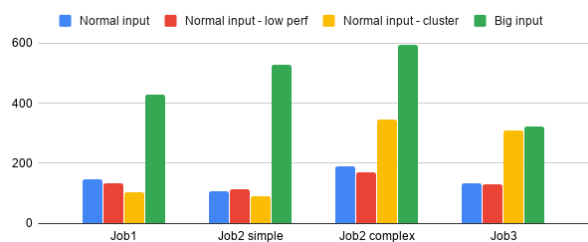
(e) Job2 versione Basic vs Complex



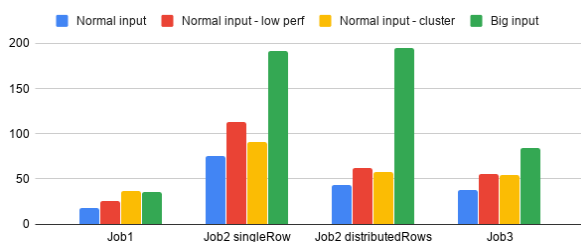
(f) Job2 Spark versione SingleRow



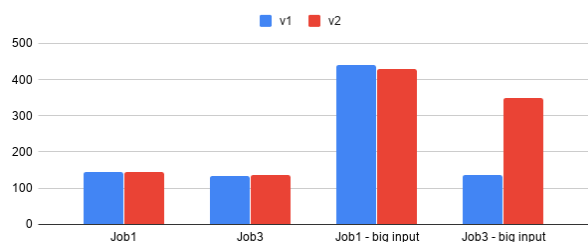
(g) Confronto esecuzione Hadoop al variare di input e macchina



(h) Confronto esecuzione Hive al variare di input e macchina



(i) Confronto esecuzione Spark al variare di input e macchina



(j) Confronto esecuzione Hive con poche o molte tabelle temporanee