Documentation

Meer Shah Hamza Shah

Hadoop MapReduce

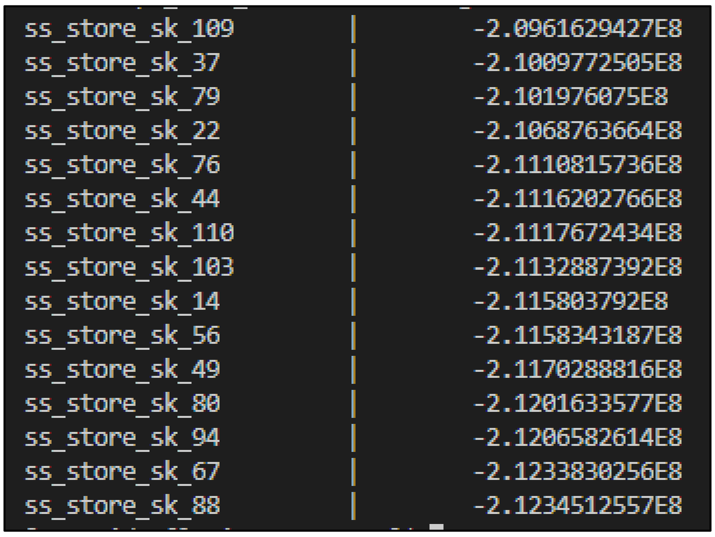
Query 1a

```
hadoop com.sun.tools.javac.Main TreeMap1a/Query.java
```

```
jar cf 1a.jar TreeMap1a/Query*.class
```

```
hadoop jar 1a.jar TreeMap1a/Query 15 2451420 2451620
input/40G/store_sales output/1a
```

```
hdfs dfs -cat output/1a/out2/*
```



```
ss_store_sk_109        |        -2.0961629427E8
ss_store_sk_37         |        -2.1009772505E8
ss_store_sk_79         |        -2.101976075E8
ss_store_sk_22         |        -2.1068763664E8
ss_store_sk_76         |        -2.1110815736E8
ss_store_sk_44         |        -2.1116202766E8
ss_store_sk_110        |        -2.1117672434E8
ss_store_sk_103        |        -2.1132887392E8
ss_store_sk_14         |        -2.115803792E8
ss_store_sk_56         |        -2.1158343187E8
ss_store_sk_49         |        -2.1170288816E8
ss_store_sk_80         |        -2.1201633577E8
ss_store_sk_94         |        -2.1206582614E8
ss_store_sk_67         |        -2.1233830256E8
ss_store_sk_88         |        -2.1234512557E8
```

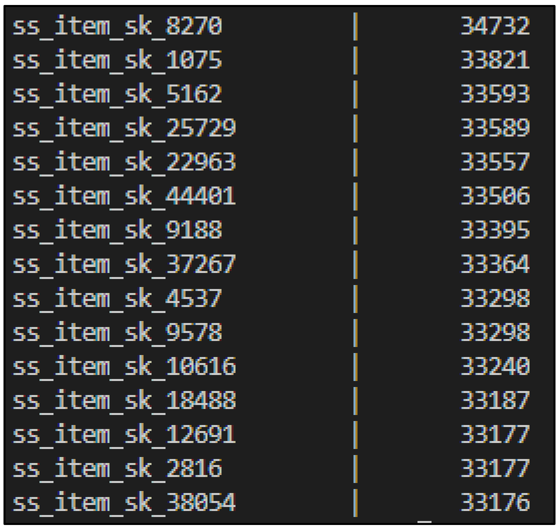
Query 1b

```
hadoop com.sun.tools.javac.Main TreeMap1b/Query.java
```

```
jar cf 1b.jar TreeMap1b/Query*.class
```

```
hadoop jar 1b.jar TreeMap1b/Query 15 2451420 2451620
input/40G/store_sales output/1b
```

```
hdfs dfs -cat output/1b/out2/*
```



```
ss_item_sk_8270        |        34732
ss_item_sk_1075        |        33821
ss_item_sk_5162        |        33593
ss_item_sk_25729       |        33589
ss_item_sk_22963       |        33557
ss_item_sk_44401       |        33506
ss_item_sk_9188        |        33395
ss_item_sk_37267       |        33364
ss_item_sk_4537        |        33298
ss_item_sk_9578        |        33298
ss_item_sk_10616       |        33240
ss_item_sk_18488       |        33187
ss_item_sk_12691       |        33177
ss_item_sk_2816        |        33177
ss_item_sk_38054       |        33176
```

Documentation

Meer Shah Hamza Shah

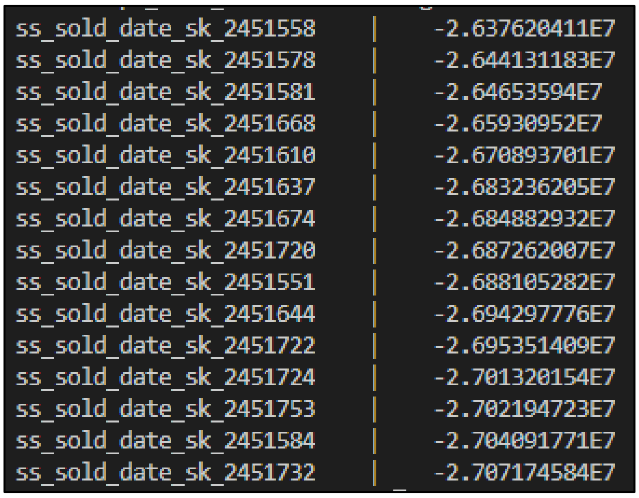
Query 1c

```
hadoop com.sun.tools.javac.Main TreeMap1c/Query.java
```

```
jar cf 1c.jar TreeMap1c/Query*.class
```

```
hadoop jar 1c.jar TreeMap1c/Query 15 2451520 2451771
input/40G/store_sales output/1c
```

```
hdfs dfs -cat output/1c/out2/*
```



```
ss_sold_date_sk_2451558   |   -2.637620411E7
ss_sold_date_sk_2451578   |   -2.644131183E7
ss_sold_date_sk_2451581   |   -2.64653594E7
ss_sold_date_sk_2451668   |   -2.65930952E7
ss_sold_date_sk_2451610   |   -2.670893701E7
ss_sold_date_sk_2451637   |   -2.683236205E7
ss_sold_date_sk_2451674   |   -2.684882932E7
ss_sold_date_sk_2451720   |   -2.687262007E7
ss_sold_date_sk_2451551   |   -2.688105282E7
ss_sold_date_sk_2451644   |   -2.694297776E7
ss_sold_date_sk_2451722   |   -2.695351409E7
ss_sold_date_sk_2451724   |   -2.701320154E7
ss_sold_date_sk_2451753   |   -2.702194723E7
ss_sold_date_sk_2451584   |   -2.704091771E7
ss_sold_date_sk_2451732   |   -2.707174584E7
```

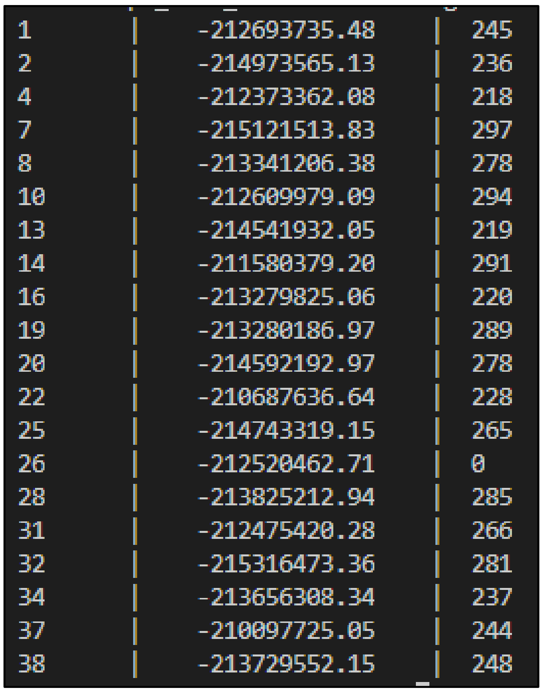
Query 2

```
hadoop com.sun.tools.javac.Main TreeMap2/Query.java
```

```
jar cf 2.jar TreeMap2/Query*.class
```

```
hadoop jar 2.jar TreeMap2/Query 20 2451420 2451620
input/40G/store_sales input/40G/store output/2
```

```
hdfs dfs -cat output/2/out2/*
```



```
1     |    -212693735.48   |  245
2     |    -214973565.13   |  236
4     |    -212373362.08   |  218
7     |    -215121513.83   |  297
8     |    -213341206.38   |  278
10    |    -212609979.09   |  294
13    |    -214541932.05   |  219
14    |    -211580379.20   |  291
16    |    -213279825.06   |  220
19    |    -213280186.97   |  289
20    |    -214592192.97   |  278
22    |    -210687636.64   |  228
25    |    -214743319.15   |  265
26    |    -212520462.71   |  0
28    |    -213825212.94   |  285
31    |    -212475420.28   |  266
32    |    -215316473.36   |  281
34    |    -213656308.34   |  237
37    |    -210097725.05   |  244
38    |    -213729552.15   |  248
```

Documentation

Meer Shah Hamza Shah

Apache Hive

Table Creation

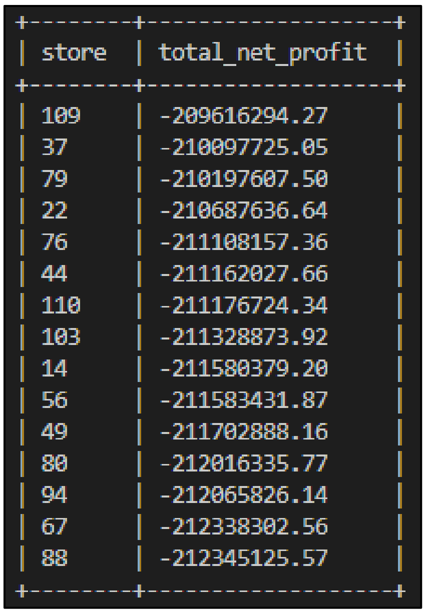
```
$HIVE_HOME/bin/beeline -u jdbc:hive2://
```

```
CREATE EXTERNAL TABLE store_sales_40 (
ss_sold_dates_sk_ss_sold_time_sk int, ss_sold_time_sk int,
ss_item_sk int, ss_customer_sk int, ss_cdemo_sk int, ss_hdemo_sk
int, ss_addr_sk int, ss_store_sk int, ss_promo_sk int,
ss_ticket_number int, ss_quantity int, ss_wholesale_cost
decimal(10,2), ss_list_price decimal(10,2), ss_sales_price
decimal(10,2), ss_ext_discount_amt decimal(10,2), ss_ext_sales_price
decimal(10,2), ss_ext_wholesale_cost decimal(10,2),
ss_ext_list_price decimal(10,2), ss_ext_tax decimal(10,2),
ss_coupon_amt decimal(10,2), ss_net_paid decimal(10,2),
ss_net_paid_inc_tax decimal(10,2), ss_net_profit decimal(10,2) ) ROW
FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION
'input/40G/store_sales';
```

```
CREATE EXTERNAL TABLE store_40 (s_store_sk int, s_store_id
varchar(16), s_rec_start_date date, s_rec_end_date date,
s_closed_date_sk int, s_store_name varchar(32), s_number_employees
int, s_floor_space int, s_hours varchar(16), s_manager varchar(64),
s_market_id int, s_geography_class varchar(16), s_market_dsc
varchar(256), s_market_manager varchar(64), s_division_id int,
s_division_name varchar(16), s_company_id int, s_company_name
varchar(16), s_street_number int) ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' LOCATION 'input/40G/store';
```

Query 1a

```
SELECT ss_store_sk AS store, SUM(ss_net_profit) AS total_net_profit
FROM store_sales_40 WHERE ss_sold_dates_sk_ss_sold_time_sk BETWEEN
2451420 AND 2451620 AND ss_store_sk IS NOT NULL GROUP BY ss_store_sk
ORDER BY total_net_profit DESC LIMIT 15;
```



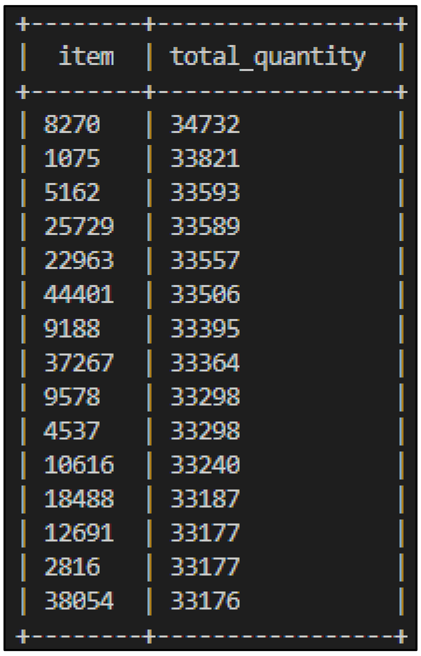
```
+--------+-------------------+
| store  | total_net_profit  |
+--------+-------------------+
| 109    | -209616294.27     |
| 37     | -210097725.05     |
| 79     | -210197607.50     |
| 22     | -210687636.64     |
| 76     | -211108157.36     |
| 44     | -211162027.66     |
| 110    | -211176724.34     |
| 103    | -211328873.92     |
| 14     | -211580379.20     |
| 56     | -211583431.87     |
| 49     | -211702888.16     |
| 80     | -212016335.77     |
| 94     | -212065826.14     |
| 67     | -212338302.56     |
| 88     | -212345125.57     |
+--------+-------------------+
```

Documentation

Meer Shah Hamza Shah

Query 1b

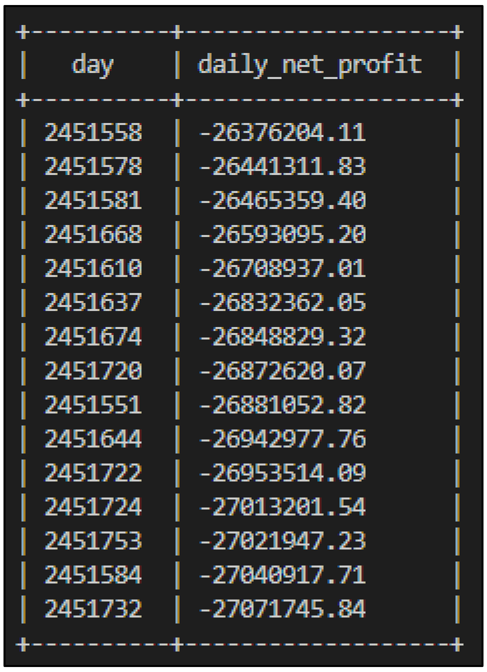
```
SELECT ss_item_sk AS item, SUM(ss_quantity) AS total_quantity FROM
store_sales_40 WHERE ss_sold_dates_sk_ss_sold_time_sk BETWEEN
2451420 AND 2451620 GROUP BY ss_item_sk ORDER BY total_quantity DESC
LIMIT 15;
```



```
+--------+-----------------+
|  item  | total_quantity  |
+--------+-----------------+
| 8270   | 34732           |
| 1075   | 33821           |
| 5162   | 33593           |
| 25729  | 33589           |
| 22963  | 33557           |
| 44401  | 33506           |
| 9188   | 33395           |
| 37267  | 33364           |
| 9578   | 33298           |
| 4537   | 33298           |
| 10616  | 33240           |
| 18488  | 33187           |
| 12691  | 33177           |
| 2816   | 33177           |
| 38054  | 33176           |
+--------+-----------------+
```

Query 1c

```
SELECT ss_sold_dates_sk_ss_sold_time_sk AS day, SUM(ss_net_profit)
AS daily_net_profit FROM store_sales_40 WHERE
ss_sold_dates_sk_ss_sold_time_sk BETWEEN 2451520 AND 2451771 GROUP
BY ss_sold_dates_sk_ss_sold_time_sk ORDER BY daily_net_profit  DESC
LIMIT 15;
```



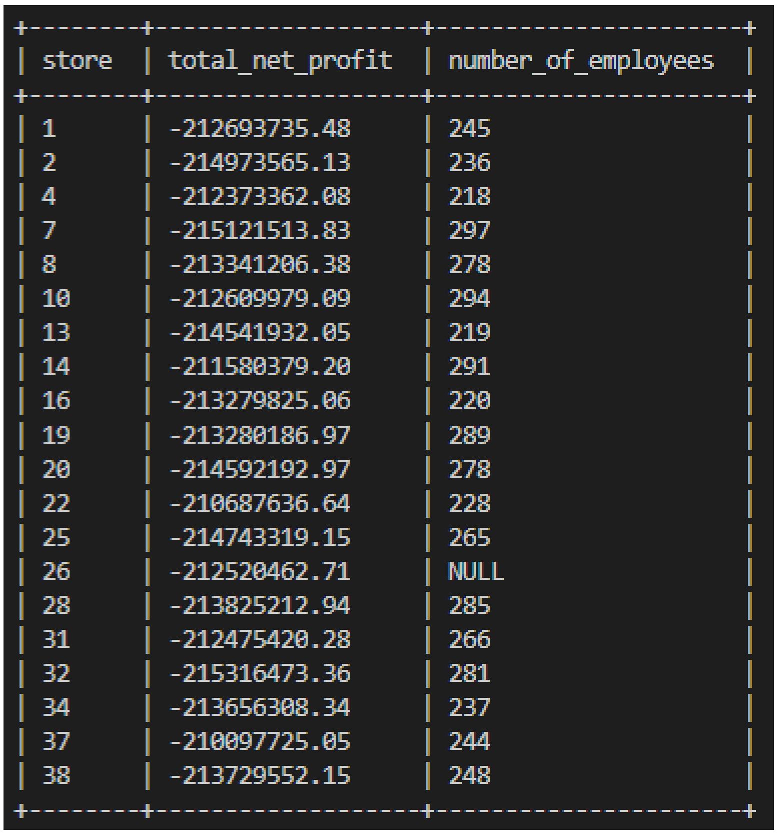
```
+----------+-------------------+
|   day    | daily_net_profit  |
+----------+-------------------+
| 2451558  | -26376204.11      |
| 2451578  | -26441311.83      |
| 2451581  | -26465359.40      |
| 2451668  | -26593095.20      |
| 2451610  | -26708937.01      |
| 2451637  | -26832362.05      |
| 2451674  | -26848829.32      |
| 2451720  | -26872620.07      |
| 2451551  | -26881052.82      |
| 2451644  | -26942977.76      |
| 2451722  | -26953514.09      |
| 2451724  | -27013201.54      |
| 2451753  | -27021947.23      |
| 2451584  | -27040917.71      |
| 2451732  | -27071745.84      |
+----------+-------------------+
```

Documentation

Meer Shah Hamza Shah

Query 2

```
SELECT ss.ss_store_sk AS store, SUM(ss.ss_net_profit) AS
total_net_profit, st.s_number_employees AS number_of_employees FROM
store_sales_40 ss INNER JOIN store_40 st ON ss.ss_store_sk =
st.s_store_sk WHERE ss.ss_sold_dates_sk_ss_sold_time_sk BETWEEN
2451420 AND 2451620 GROUP BY ss.ss_store_sk, st.s_number_employees
ORDER BY ss.ss_store_sk ASC LIMIT 20;
```



```
+--------+-------------------+----------------------+
| store  | total_net_profit  | number_of_employees  |
+--------+-------------------+----------------------+
| 1      | -212693735.48     | 245                  |
| 2      | -214973565.13     | 236                  |
| 4      | -212373362.08     | 218                  |
| 7      | -215121513.83     | 297                  |
| 8      | -213341206.38     | 278                  |
| 10     | -212609979.09     | 294                  |
| 13     | -214541932.05     | 219                  |
| 14     | -211580379.20     | 291                  |
| 16     | -213279825.06     | 220                  |
| 19     | -213280186.97     | 289                  |
| 20     | -214592192.97     | 278                  |
| 22     | -210687636.64     | 228                  |
| 25     | -214743319.15     | 265                  |
| 26     | -212520462.71     | NULL                 |
| 28     | -213825212.94     | 285                  |
| 31     | -212475420.28     | 266                  |
| 32     | -215316473.36     | 281                  |
| 34     | -213656308.34     | 237                  |
| 37     | -210097725.05     | 244                  |
| 38     | -213729552.15     | 248                  |
+--------+-------------------+----------------------+
```

Documentation

Meer Shah Hamza Shah

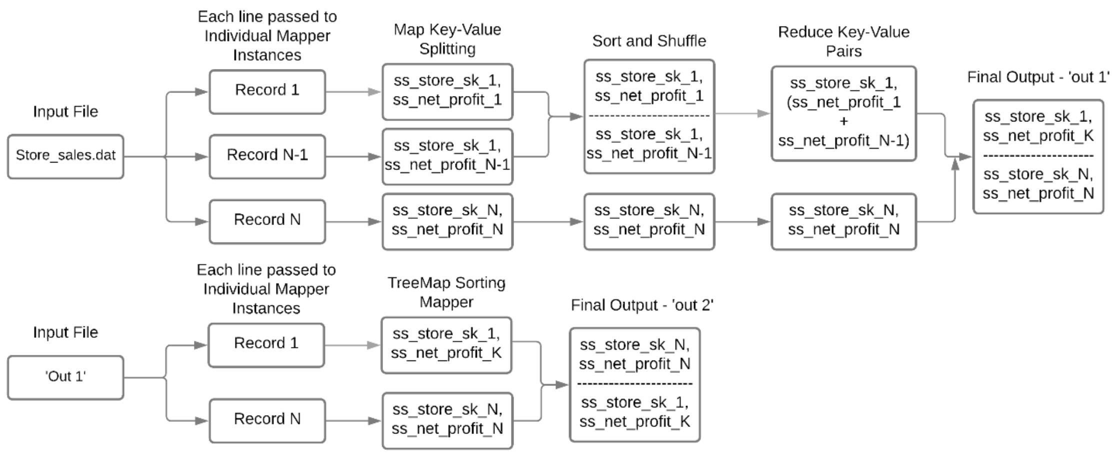
Implementation

From the Group Report

Hadoop MapReduce - Query 1

Our MapReduce solution in the context of Query 1a uses a full MapReduce job and a final map operation. Our firstmapper selects the rows in the store_sales table which match the date constraints. It does this by iterating through the rows assigned to that mapper. We create a string array of the fields in that row, importantly including its empty fields. This is important as it allows us to check the row against the constraints efficiently using indexing. The mapper returns a map with the store as key and an array of the corresponding net profits for that single store. The consequent reducer, after the automatic shuffle and sort, receives groups of these maps andaggregates the net profits. This is achieved by iterating over the profit array for every corresponding storename and calculating net profit over all dates for the store. These Key-Value pairs are then passed to the final job that consists of a singular mapper.

The next job attempts to sort the aggregated key value pairs by net profits. It does this using a JavaTreeMap. First the Key-Value Pairs will be reversed to Value-Key i.e., the key would be net profit and the value the store name. This is done to allow the TreeMap to sort the net profits in descending order. Once all Key-Pair values have been reversed and input into the TreeMap to be sorted, only the Top-K rows specified are retrieved from the TreeMap and correspond to the Top-K store names that have the largest net profit over the dates specified. These store names and profits are output from the TreeMap in the form<NetProfit, StoreName> but are reversed before being finally output from the mapper to <StoreName,NetProfit>. This step is not necessary but ensures the output is aligned to the sample output which is also more readable.

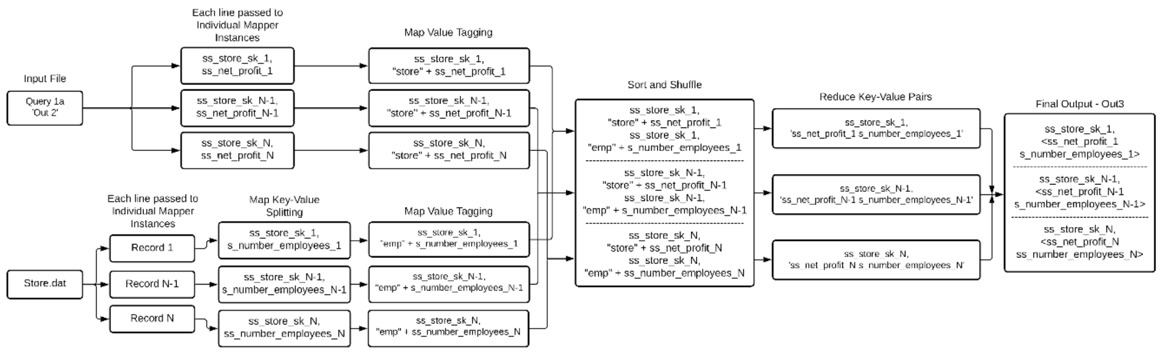

Input File
Store_sales.dat
Each line passed to Individual Mapper Instances
Record 1
Record N-1
Record N
Map Key-Value Splitting
ss_store_sk_1, ss_net_profit_1
ss_store_sk_1, ss_net_profit_N-1
ss_store_sk_N, ss_net_profit_N
Sort and Shuffle
ss_store_sk_1, ss_net_profit_1
ss_store_sk_1, ss_net_profit_N-1
ss_store_sk_N, ss_net_profit_N
Reduce Key-Value Pairs
ss_store_sk_1, (ss_net_profit_1 + ss_net_profit_N-1)
ss_store_sk_N, ss_net_profit_N
Final Output - 'out 1'
ss_store_sk_1, ss_net_profit_K
ss_store_sk_N, ss_net_profit_N
Input File
'Out 1'
Each line passed to Individual Mapper Instances
Record 1
Record N
TreeMap Sorting Mapper
ss_store_sk_1, ss_net_profit_K
ss_store_sk_N, ss_net_profit_N
Final Output - 'out 2'
ss_store_sk_N, ss_net_profit_N
ss_store_sk_1, ss_net_profit_K

Documentation

Meer Shah Hamza Shah

Hadoop MapReduce - Query 2

The foundation of Query 2 is built upon Query 1a. It also contains two additional mappers and a final reducer. The first mapper takes the output of the Mapper from Query 1a as input. The mapper will then get all the Key-Value Pairs from the input and tag them to be used by the reducer for the join. As previously mentioned, Query 2 also consists of another mapper that takes input from'store.dat'. This mapper will select the columns corresponding to the Key-Value pair 'StoreName' and 'Number of Employees'. Again, these will be tagged for the reduce-side join. Finally, the reducer will take the input of both the mappers. For every key i.e., StoreName in both inputs, the reducer will merge the values of both mappers essentially joining the two columns, 'Number of Employees'and 'Net Profit' on the key, 'Store Name'. The K parameter from the command line is used to limit the output to the Top-K 'Store Name' which are sorted in ascending order. Lastly before the reducer outputs the joined data, the output is formatted by converting numbers to 2 decimal places for consistency and adding pipes for the separation of data increasing readability.


Input File
Query 1a 'Out 2'
Each line passed to Individual Mapper Instances
ss_store_sk_1, ss_net_profit_1
ss_store_sk_N-1, ss_net_profit_N-1
ss_store_sk_N, ss_net_profit_N
Map Value Tagging
ss_store_sk_1, "store" + ss_net_profit_1
ss_store_sk_N-1, "store" + ss_net_profit_N-1
ss_store_sk_N, "store" + ss_net_profit_N
Store.dat
Each line passed to Individual Mapper Instances
Record 1
Record N-1
Record N
Map Key-Value Splitting
ss_store_sk_1, s_number_employees_1
ss_store_sk_N-1, s_number_employees_N-1
ss_store_sk_N, ss_number_employees_N
Map Value Tagging
ss_store_sk_1, "emp" + s_number_employees_1
ss_store_sk_N-1, "emp" + s_number_employees_N-1
ss_store_sk_N, "emp" + ss_number_employees_N
Sort and Shuffle
ss_store_sk_1, "store" + ss_net_profit_1 ss_store_sk_1, "emp" + s_number_employees_1
ss_store_sk_N-1, "store" + ss_net_profit_N-1 ss_store_sk_N-1, "emp" + s_number_employees_N-1
ss_store_sk_N, "store" + ss_net_profit_N ss_store_sk_N, "emp" + ss_number_employees_N
Reduce Key-Value Pairs
ss_store_sk_1, 'ss_net_profit_1 s_number_employees_1'
ss_store_sk_N-1, 'ss_net_profit_N-1 s_number_employees_N-1'
ss_store_sk_N, 'ss_net_profit_N s_number_employees_N'
Final Output - Out3
ss_store_sk_1, <ss_net_profit_1 s_number_employees_1>
ss_store_sk_N-1, <ss_net_profit_N-1 s_number_employees_N-1>
ss_store_sk_N, <ss_net_profit_N ss_number_employees_N>

Considerations

An important consideration when dealing with MapReduce jobs was to overcome the unstructured formatting of the data. Since, some rows had missing values for certain columns, these had to be accounted for by considering them as empty strings when parsing the rows in the databases. This allows for consistency when indexing the array created from each row.

Another important consideration was dealing with negative and positive net profits simultaneously. Since the required output of net profit must be in descending order, aggregated net profits had to be multiplied by -1 before and after processing. This essentially allowed for descending sort rather than the original ascending sort.

Documentation

Meer Shah Hamza Shah

Apache Hive - Query 1

Consider query 1a. Following the output requirements of this query, the ss_store_sk and ss_net_profit projection attributes are selected from the store_sales table. The aggregate function SUM is applied toss_net_profit attributes since a store's net profits can be spread across multiple rows and thus the SUM operation gives the cumulative net profit for each store. Correspondingly, the GROUP BY function is used onthe ss_store_sk attribute. The date interval, in the queries provided, is arbitrarily chosen and the constraints are met in the WHERE clause. The BETWEEN operation ensures the ss_sold_dates_sk_ss_sold_time_sk attribute in the row under consideration is between the dates chosen. This is chosen over the alternative, comparison operators, as it is more readable. Aliases were also used to make the query and output tablemore readable. NULL fields occur in the store_sales table and do not contribute to any store's total netprofit. To ensure the 'NULL-store' fields are not outputted, an extra constraint is included in the WHERE clause. Finally, the ORDER BY operation ensures the output is ordered by the total net profits and the LIMIT operation corresponds to the 'top k' constraint in the question. In this example, k=10. A similar approach is taken for queries 1a and 1b.

Apache Hive - Query 2

Query 2 requires a JOIN operation since data is needed from both store_sales and store. Specifically, we select the store and the SUM of the net profits from store_sales, and the number of employees from the store table. We JOIN on the store attribute since it is present in both tables. Similar to queries 1a, 1b and 1c, we use the BETWEEN, GROUP BY, ORDER BY and LIMIT operators as needed. We again make use of aliases to make the output readable. In this query, using table aliases makes the SQL statement concise and easier to follow.