CLOUD COMPUTING PROJECT -2 REPORT

BIGDATA ANALYSIS PROJECT – WEATHER

SUBMITTED BY:

VARUN RAJ, RAYABARAPU

M#:M12467636

SUBMISSION DATE:

April 26, 2018

TABLE OF CONTENTS

1. INTRODUCTION:

This project report solves all the assigned tasks, on the give weather data from https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/, they are:

1. Average TMIN, TMAX for each year excluding abnormalities or missing data.
2. Maximum TMAX, Minimum TMIN for each year excluding abnormalities or missing data.
3. 5 hottest, 5 coldest weather stations for each year excluding abnormalities or missing data.
4. Hottest and coldest day and corresponding weather stations in the entire dataset.

Bonus tasks:

1. Median TMIN, TMAX for each year and corresponding weather stations.
2. Median TMIN, TMAX for the entire dataset

The project is mainly done using HiveQL from the AWS HUE Cluster Provided.

IMPORTANT NOTICE: ALL THE TEMPERATURES ARE REPRESENTED IN TENTHS OF DEGREE CELSIUS (DISCUSSED IN 1.1), IT MUST NOT BE MISUNDERSTOOD WITH CELSIUS.

1.1 WHAT IS TENTHS OF DEGREE CELSIUS?

For instance, if the tenths of degree Celsius representation is 189, then the actual value in Celsius is 189/10 degrees Celsius, that is, 18.9 degrees Celsius. The rest of the repost contains the temperatures in tenths of degree Celsius which should not be misunderstood with degree Celsius.

189 tenths of degrees Celsius = 18.9 degrees Celsius

2. SOLVING TASKS:

Firstly, a temporary table is created in the HiveQL as follows:

```
CREATE EXTERNAL TABLE IF NOT EXISTS weather_rayabavj (
station STRING,
date_id INT,
measurement STRING,
VALUE INT,
MFLAG STRING,
QFLAG STRING,
SFLAG STRING,
OBSTIME STRING)
COMMENT 'Data about weather'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
location '/user/cloudcomputing/weather'
```

For all the tasks, this table is used for analysis.

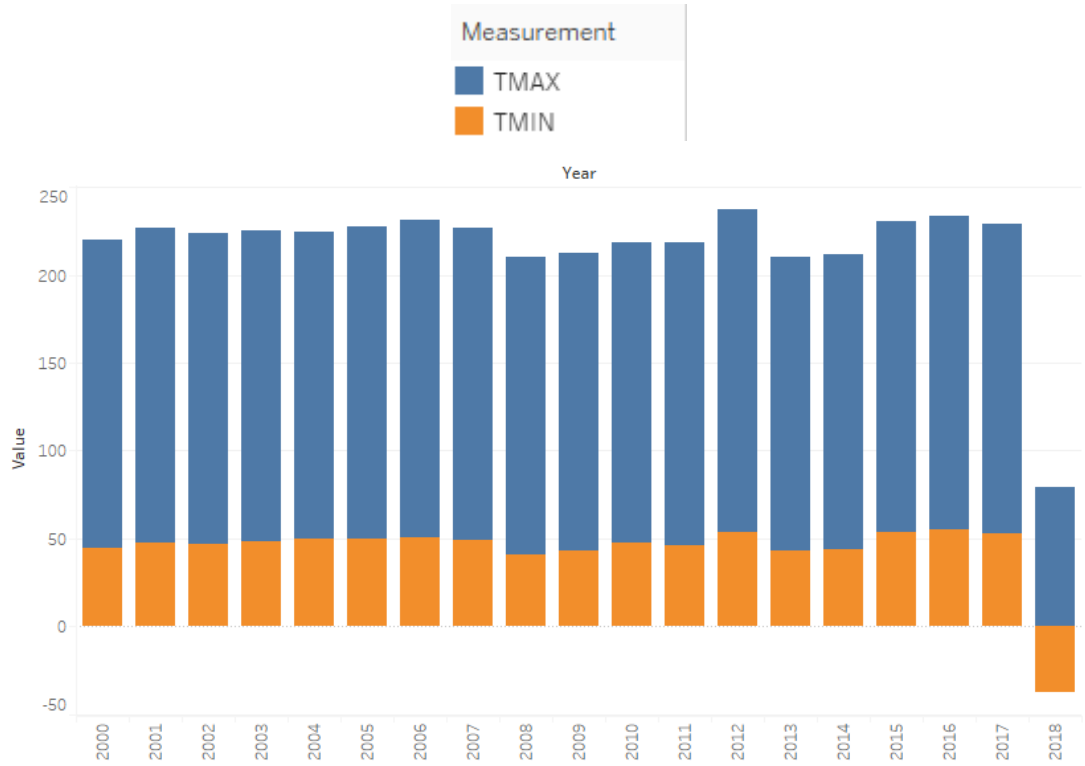
2.1 TASK 1: Average TMIN, TMAX for each year excluding abnormalities or missing data.
SOLVED USING: HIVEQL
QUERY:

```
SELECT substr(cast(date_id as string),1,4) year , measurement , avg(value) value
FROM weather_rayabavj
WHERE qflag=''
GROUP BY substr(cast(date_id as string),1,4), measurement
ORDER BY year
```

Results:

Year	Measurement	Avg.Value (tenths of °C)
2000	TMAX	175.588598
2000	TMIN	44.3120519
2001	TMIN	47.9266813
2001	TMAX	178.750314
2002	TMAX	177.231495
2002	TMIN	46.5532524
2003	TMAX	177.094666
2003	TMIN	48.6280615
2004	TMAX	174.902617
2004	TMIN	49.4991206
2005	TMIN	49.8068024
2005	TMAX	177.662085
2006	TMAX	180.736368
2006	TMIN	50.8600557
2007	TMIN	49.0043438
2007	TMAX	178.136984
2008	TMAX	169.998781
2008	TMIN	40.6528865
2009	TMIN	43.4429799
2009	TMAX	168.807402
2010	TMIN	47.2333299
2010	TMAX	171.433361
2011	TMIN	46.0174038
2011	TMAX	172.380835
2012	TMIN	53.440854
2012	TMAX	183.820525
2013	TMAX	167.31983
2013	TMIN	43.0705551
2014	TMIN	43.7839876
2014	TMAX	168.409385
2015	TMIN	53.4254955
2015	TMAX	177.115037
2016	TMIN	55.0031959
2016	TMAX	178.990128
2017	TMIN	52.6928672
2017	TMAX	176.566693
2018	TMIN	-37.3588068
2018	TMAX	78.8660958

Plots:



Short Analysis:

The average TMAX and TMIN follow the same trend, that is, when TMAX increases the TMIN also increases or remains same for the next year, but when TMAX decreases TMIN decreases for the following year. The low TMAX and TMIN for the 2018 are because of first cold months data is available.

2.2 TASK2: Maximum TMAX, Minimum TMIN for each year excluding abnormalities or missing data.

SOLVED USING: HiveQL

QUERY:

For Maximum TMAX temperatures per year:

```
SELECT substr(cast(date_id as string),1,4) year , measurement , max(value) value
FROM weather_rayabavj
WHERE qflag='' AND measurement='TMAX'
GROUP BY substr(cast(date_id as string),1,4), measurement
ORDER BY year
```

For Minimum TMIN temperatures per year:

```
SELECT substr(cast(date_id as string),1,4) year , measurement , min(value) value
FROM weather_rayabavj
WHERE qflag='' AND measurement='TMIN'
GROUP BY substr(cast(date_id as string),1,4), measurement
ORDER BY year
```

Results:

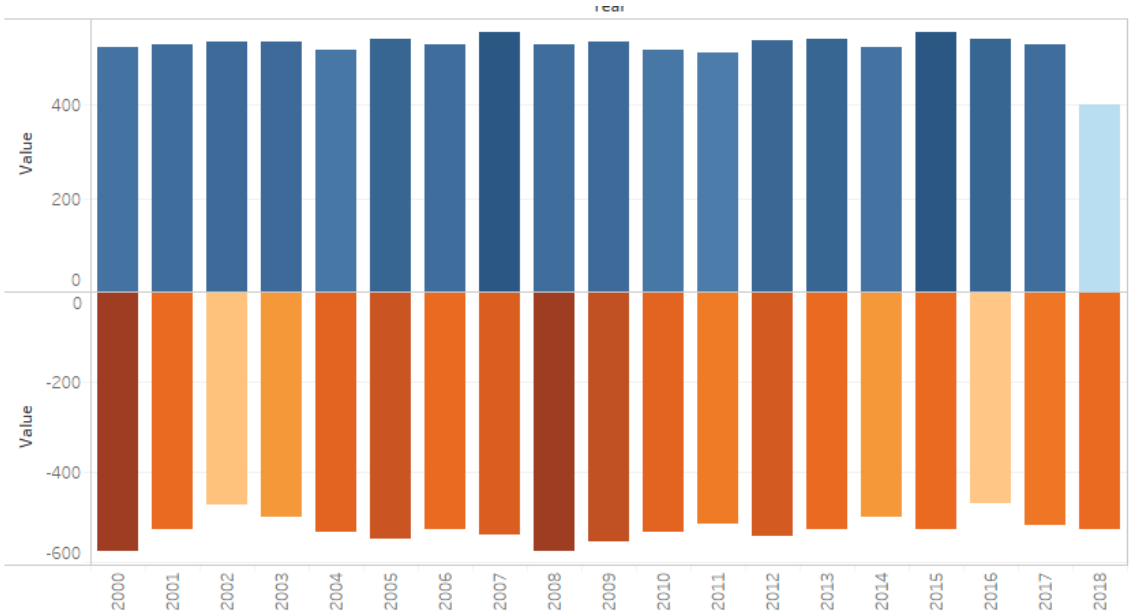
Year	Measurement	Max Value (tenths of °C)
2000	TMAX	522
2001	TMAX	528
2002	TMAX	533
2003	TMAX	533
2004	TMAX	517
2005	TMAX	539
2006	TMAX	528
2007	TMAX	556
2008	TMAX	528
2009	TMAX	533
2010	TMAX	517
2011	TMAX	511
2012	TMAX	537
2013	TMAX	539
2014	TMAX	522
2015	TMAX	556
2016	TMAX	539
2017	TMAX	528
2018	TMAX	400

Year	Measurement	Min Value (tenths of °C)
2000	TMIN	-578
2001	TMIN	-528
2002	TMIN	-472
2003	TMIN	-500
2004	TMIN	-533
2005	TMIN	-550
2006	TMIN	-528
2007	TMIN	-539
2008	TMIN	-578
2009	TMIN	-556
2010	TMIN	-533
2011	TMIN	-517
2012	TMIN	-544
2013	TMIN	-528
2014	TMIN	-500
2015	TMIN	-528
2016	TMIN	-469
2017	TMIN	-520
2018	TMIN	-528

Plots:

TMAX:

TMIN:



Short Analysis:
2000 and 2008 are the coldest years from the given data, whereas the 2007 and 2015 are the hottest years.

2.3 TASK3: 5 hottest, 5 coldest weather stations for each year excluding abnormalities or missing data.

SOLVED USING: HiveQL
QUERY:
For 5 Hottest Weather Stations per year:
Code Analysis:
The query consists of four SELECT statements, one within the other. The innermost SELECT statement cleans the data from abnormalities by checking qflag='', partitions the data according to year, orders them in descending order of values column and finally, dense_ranks the rows within the partitions. Another SELECT statement outside the innermost one, selects the rows whose ranks are less than 15 (Top 15 hottest temperatures recorded) from the partitions. The result of this SELECT statement is grouped again according to year, station & value (To eliminate repeated stations records) and ordered by descending value to select only the maximum of these within of each these groups and give them a row number. Finally, the outermost SELECT statement will

pick only five rows from the row number which are grouped by year, station & value to give five rows per year and corresponding stations.

Query:

```
SELECT year, station, val
FROM(
  SELECT year, station, val,
  row_number() OVER (PARTITION BY year ORDER BY val DESC) row_number
  FROM(
  SELECT year, station, max(value) val
  FROM (
  SELECT year, measurement, value, station, dense_rank
  FROM (
  SELECT
    substr(cast(date_id as STRING) ,1,4) year,
    measurement,
    value, station,
    dense_rank() OVER
      (PARTITION BY substr(cast(date_id as STRING) ,1,4) ORDER BY value DESC)
dense_rank
  FROM weather_rayabavj
  WHERE measurement="TMAX" and qflag="
  ) t
  WHERE dense_rank < 15 ORDER BY year, dense_rank
  ) x
  GROUP BY year, station
  ORDER BY year, val DESC
  )y
  GROUP BY year, station,val
  ORDER BY year, val DESC
  )z
WHERE row_number<6
GROUP BY year, station,val
ORDER BY year, val DESC
```

For 5 Coldest Weather Stations per year:

Code Analysis:

The query consists of four SELECT statements, one within the other. The innermost SELECT statement cleans the data from abnormalities by checking qflag='', partitions the data according to year, orders them in ascending order of values column and finally, dense_ranks the rows within the partitions. Another SELECT statement outside the innermost one, selects the rows whose ranks are less than 15 (Top 15 coldest temperatures recorded) from the partitions. The result of this SELECT statement is grouped again according to year, station & value (To eliminate repeated stations records) and ordered by ascending value to select only the minimum of these within of each these groups and give them a row number. Finally, the outermost SELECT statement will pick only five rows from the row number which are grouped by year, station & value to give five rows per year and corresponding stations.

Query:

```
SELECT year, station, val
FROM(
  SELECT year, station, val,
  row_number() OVER (PARTITION BY year ORDER BY val ASC) row_number
  FROM(
  SELECT year, station, min(value) val
  FROM (
  SELECT year, measurement, value, station, dense_rank
  FROM (
  SELECT
    substr(cast(date_id as STRING) ,1,4) year,
    measurement,
    value, station,
    dense_rank() OVER
      (PARTITION BY substr(cast(date_id as STRING) ,1,4) ORDER BY value ASC)
dense_rank
  FROM weather_rayabavj
  WHERE measurement="TMIN" and qflag=''
  ) t
  WHERE dense_rank < 15 ORDER BY year, dense_rank
  ) x
  GROUP BY year, station
  ORDER BY year, val ASC
  )y
  GROUP BY year, station,val
  ORDER BY year, val ASC
  )z
WHERE row_number<6
GROUP BY year, station,val
ORDER BY year, val ASC
```

Results:

For 5 Hottest stations per year:

Year	Station_ID	Max Temp. (tenths of °C)
2000	USC00042319	522
2000	USC00024761	494
2000	USR0000AHAV	494
2000	USC00021050	494
2000	USC00029376	489
2001	USC00042319	528
2001	USR0000CMOJ	506
2001	USC00021050	500
2001	USC00024829	500
2001	USC00265085	500
2002	USC00042319	533
2002	USR0000CMOJ	500

2002	USC00265846	500
2002	USR0000CCOR	489
2002	USC00044297	489
2003	USC00042319	533
2003	USR0000CMEA	528
2003	USR0000AHAV	522
2003	USC00021050	511
2003	USC00029376	506
2004	USC00042319	517
2004	USC00024761	511
2004	USW00053139	495
2004	USC00021050	489
2004	USC00261371	489
2005	USC00042319	539
2005	USW00053139	519
2005	USR0000CMOJ	517
2005	USW00023179	517
2005	USR0000AHAV	511
2006	USC00042319	528
2006	USW00053139	511
2006	USC00029376	500
2006	USC00044259	500
2006	USC00045502	500
2007	USR0000CSAW	556
2007	USC00042319	539
2007	USW00053139	523
2007	USR0000CMOJ	517
2007	USR0000AHAV	506
2008	USC00042319	528
2008	USC00044297	528
2008	USC00024761	511
2008	USW00053139	508
2008	USC00261371	500
2009	USC00042319	533
2009	USW00053139	509
2009	USR0000CBUU	494
2009	USR0000CSQU	494
2009	USR0000AHAV	494
2010	USC00042319	517
2010	USR0000AHAV	511
2010	USW00053139	504
2010	USC00021050	494
2010	USR0000CBUU	494
2011	USC00042319	511
2011	USR0000CCAA	500
2011	USC00022782	494
2011	USR0000CBUU	494
2011	USW00053139	490
2012	USS0005N23S	537
2012	USC00042319	533
2012	USW00053139	512
2012	USC00020672	494

2012	USC00265846	494
2013	USC00042319	539
2013	USW00004134	533
2013	USC00044297	528
2013	USW00053139	520
2013	USR0000AHAV	506
2014	USC00042319	522
2014	USW00053139	511
2014	USC00028396	494
2014	USC00020672	489
2014	USW00003125	489
2015	USR0000HKAU	556
2015	USC00042319	517
2015	USW00053139	499
2015	USR0000CBUU	489
2015	USW00023158	483
2016	USR0000CBEV	539
2016	USC00042319	528
2016	USC00040924	522
2016	USW00023179	517
2016	USW00053139	517
2017	USC00042319	528
2017	USC00021050	522
2017	USW00053139	518
2017	USW00023179	517
2017	USC00046386	511
2018	USC00417624	400
2018	USC00411524	394
2018	USR0000TFAL	394
2018	USR0000CCAA	389
2018	USW00003104	383

For 5 Coldest stations per year:

Year	Station_ID	Min Temp. (tenths of °C)
2000	USC00505644	-578
2000	USC00501684	-578
2000	USC00508140	-539
2000	USW00026440	-517
2000	USC00502350	-511
2001	USW00026508	-528
2001	USR0000ABCA	-511
2001	USS0051R01S	-480
2001	USC00501492	-461
2001	USC00504567	-461
2002	USR0000AKAI	-472
2002	USS0051R01S	-470
2002	USS0050S01S	-470
2002	USR0000ABEV	-467
2002	USC00503212	-461
2003	USC00501492	-500

2003	USS0051R01S	-490
2003	USW00026533	-483
2003	USS0050S01S	-480
2003	USC00509869	-467
2004	USC00501684	-533
2004	USC00502568	-506
2004	USC00502350	-500
2004	USS0045O10S	-500
2004	USW00026412	-500
2005	USC00501684	-550
2005	USC00509313	-528
2005	USC00509869	-517
2005	USC00502568	-511
2005	USW00026412	-511
2006	USR0000ASEL	-528
2006	USC00501492	-522
2006	USR0000ABEV	-511
2006	USR0000ACHL	-506
2006	USC00501684	-500
2007	USC00501684	-539
2007	USS0045R01S	-501
2007	USW00026422	-493
2007	USC00502607	-489
2007	USR0000ACHL	-489
2008	USC00501684	-578
2008	USC00501492	-522
2008	USR0000ACHL	-522
2008	USS0045R01S	-520
2008	USR0000ABEV	-511
2009	USC00502101	-556
2009	USC00501684	-556
2009	USC00509313	-528
2009	USR0000ACHL	-522
2009	USC00501492	-517
2010	USC00501684	-533
2010	USC00502101	-533
2010	USS0051R01S	-519
2010	USR0000ACHL	-517
2010	USS0045R01S	-515
2011	USC00509869	-517
2011	USS0045R01S	-507
2011	USS0051R01S	-502
2011	USC00501684	-500
2011	USR0000AFYK	-494
2012	USC00503165	-544
2012	USC00503212	-539
2012	USS0051R01S	-536
2012	USC00504210	-533
2012	USR0000AFYK	-533
2013	USC00502339	-528
2013	USC00501684	-522
2013	USS0051R01S	-505

2013	USS0045R01S	-502
2013	USR0000ABCK	-500
2014	USC00501684	-500
2014	USS0045R01S	-479
2014	USR0000ABCK	-472
2014	USR0000AFYK	-472
2014	USR0000ACOA	-467
2015	USC00502339	-528
2015	USC00501684	-506
2015	USS0041P07S	-491
2015	USC00509314	-489
2015	USW00026533	-488
2016	USS0051R01S	-469
2016	USR0000ACHL	-467
2016	USC00501684	-467
2016	USS0045R01S	-456
2016	USS0041P07S	-443
2017	USS0051R01S	-520
2017	USR0000ASLC	-506
2017	USW00026529	-506
2017	USR0000ALIV	-494
2017	USS0045O04S	-491
2018	USC00502339	-528
2018	USC00501684	-478
2018	USR0000AKAV	-478
2018	USR0000ANOR	-478
2018	USW00096406	-475

2.4 TASK4: Hottest and coldest day and corresponding weather stations in the entire dataset.

SOLVED USING: HiveQL

QUERY:

For Hottest Day and Corresponding Station:

```
SELECT * FROM weather_rayabavj
WHERE qflag='' AND measurement='TMAX'
ORDER BY value DESC
LIMIT 10
```

This Query will result in top 10 hottest days, it is so because same maximum value can be recorded multiple times at different stations and different dates.

For Coldest Day and Corresponding Station:

```
SELECT * FROM weather_rayabavj
WHERE qflag='' AND measurement='TMIN'
ORDER BY value ASC
LIMIT 10
```

This Query will result in top 10 coldest days, it is so because same minimum value can be recorded multiple times at different stations and different dates.

Results:

For Hottest Day and Corresponding Station:

Station_ID	Date	Measurement	Value (tenths of °C)	MFlag	QFlag	SFlag	OBSTime
USR0000HKAU	20150213	TMAX	556	H		U	
USR0000HKAU	20150215	TMAX	556	H		U	
USR0000CSAW	20071013	TMAX	556	H		U	
USC00042319	20070707	TMAX	539			0	0800
USC00042319	20050720	TMAX	539			0	0800
USC00042319	20130701	TMAX	539			7	0800
USR0000HKAU	20150212	TMAX	539	H		U	
USR0000CBEV	20160926	TMAX	539	H		U	
USS0005N23S	20120704	TMAX	537			T	
USC00042319	20120712	TMAX	533			7	0800

From the above table it is evident that top three entries have same highest TMAX value, which will be the hottest days from the data.

For Coldest Day and Corresponding Station:

Station_ID	Date	Measurement	Value (tenths of °C)	MFlag	QFlag	SFlag	OBSTime
USC00501684	20080208	TMIN	-578			0	0800
USC00501684	20080207	TMIN	-578			0	0800
USC00505644	20000102	TMIN	-578			0	1800
USC00501684	20000101	TMIN	-578			0	0800
USC00501684	20080209	TMIN	-561			0	0800
USC00501684	20090108	TMIN	-556			0	0800
USC00502101	20090110	TMIN	-556			0	0800
USC00502101	20090109	TMIN	-550			0	0800
USC00505644	20000101	TMIN	-550			0	1800
USC00501684	20000102	TMIN	-550			0	0800

From the above table it is evident that top four entries have same lowest TMIN value, which will be the coldest days from the data.

2.5 BONUS TASK 1: Median TMIN, TMAX for each year and corresponding weather stations.

SOLVED USING: HiveQL and pySpark

HiveQL Query will result in Median TMIN and TMAX for each year, and for the corresponding stations pySpark is used. The code for pySpaark is in Appendix ii. This code will give the corresponding median value recorded first in that year.

QUERY:

```
SELECT year, measurement, percentile(cast(value AS BIGINT),0.5) Median FROM
(
   SELECT *, substr(cast(date_id as STRING) ,1,4) year FROM
   weather_rayabavj
   WHERE measurement='TMAX' AND qflag=''
) z1
GROUP BY year,measurement
ORDER BY year
UNION ALL
SELECT year, measurement, percentile(cast(value AS BIGINT),0.5) Median FROM
(
   SELECT *, substr(cast(date_id as STRING) ,1,4) year FROM
   weather_rayabavj
   WHERE measurement='TMIN' AND qflag=''
) z2
GROUP BY year,measurement
ORDER BY year
```

Result of Query(for Query Results Consider only first three columns) and pySpark(for pyspark results look Appendix ii):

Year	Measurement	Median Value (tenths of °C)	Station_Id
2000	TMAX	189	USC00385946
2001	TMAX	194	USC00029376
2002	TMAX	183	USC00046144
2003	TMAX	194	USC00419361
2004	TMAX	189	USC00440993
2005	TMAX	189	USC00093578
2006	TMAX	189	USC00110187
2007	TMAX	194	USC00013575
2008	TMAX	183	USC00099027
2009	TMAX	183	USC00264341
2010	TMAX	183	USW00012876
2011	TMAX	178	USC00110187
2012	TMAX	194	USC00036768
2013	TMAX	183	USC00227128
2014	TMAX	183	USC00051121
2015	TMAX	194	USC00082944
2016	TMAX	194	USC00020288
2017	TMAX	193	USW00053182
2018	TMAX	72	USC00414382
2000	TMIN	47	USS0011R06S
2001	TMIN	50	USC00046144
2002	TMIN	44	USR0000WCOU
2003	TMIN	52	USS0003E05S
2004	TMIN	56	USC00029376
2005	TMIN	50	USC00091266
2006	TMIN	50	USW00024229

2007	TMIN	56	USR0000CCAS
2008	TMIN	44	USC00419361
2009	TMIN	50	USW00024229
2010	TMIN	49	USS0022D02S
2011	TMIN	50	USC00093060
2012	TMIN	56	USC00042147
2013	TMIN	44	USC00416733
2014	TMIN	50	USR0000MMAR
2015	TMIN	61	USR0000CCOO
2016	TMIN	56	USR0000TSBS
2017	TMIN	56	USC00385633
2018	TMIN	-33	USC00458715

2.6 BONUS TASK 2: Median TMIN, TMAX for entire dataset.

SOLVED USING: HiveQL

QUERY:

```
SELECT percentile(cast(value as BIGINT),0.5),measurement
FROM weather_rayabavj
WHERE qflag='' AND measurement='TMAX'
UNION ALL
SELECT percentile(cast(value as BIGINT),0.5), measurement
FROM weather_rayabavj
WHERE qflag='' AND measurement='TMIN'
```

Result:

Measurement	**Median Value (tenths of °C)**
TMAX	**189**
TMIN	**50**

3. Why I used HiveQL, not Spark or other tools?

In this section, I am going to discuss about why I used HiveQL and not Spark or other tools. Firstly, the SQL like interface made it much easier to write queries specific to the task. Furthermore, it optimizes the query in the background which eventually get converted in to map-reduce jobs. However, the speed is by far better than other tools like pySpark. I have tried to solve the Task 3 and Bonus Tasks using pySpark initially (Code is included in Appendix i), but the computation took almost 20 min whereas the simple hive query gave results in 2 min. Hence, the computation speed is a boon to HiveQL. Finally, the aggregate functions like **avg(), min()** and **max()** are readily available in Hive QL which is further facilitated query language with the GROUP BY and ORDER BY statements. And also, it noteworthy to mention the **dense_rank(), row_number()** and **percentile()** functions which are used in Task 3 and Bonus Tasks condensed the chunks of code in Spark (Code is included in Appendix i) into single line.

4. APPENDIX

i. Spark code for the Task 3:

In this appendix i Spark Code, which is used to compare with HiveQL is discussed.

```
from __future__ import print_function
import sys
from operator import add
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: TASK3 <file>", file=sys.stderr)
        exit(-1)

    sc = SparkContext(appName= "TASK3")
    sqlContext = SQLContext(sc)
    lines = sc.textFile(sys.argv[1])\
    .map(lambda l: l.split(","))\
    .map(lambda
p:Row(Station_ID=p[0],dte=p[1],element=p[2],value=int(p[3]),MFlag=p[4],QFlag=p[5],
SFlag=p[6],OBS_time=p[7]))
    df = sqlContext.createDataFrame(lines)
    sqlContext.registerDataFrameAsTable(df,"table1")
    df_n1=sqlContext.sql("SELECT Station_ID, dte, element, value FROM table1
WHERE QFlag='' AND element='TMAX' ORDER BY value DESC LIMIT 30")
    print ("Hottest Stations\n")
    temp1=[]; temp2=[]; i=0; j=0
    for p in df_n1.collect():
        if p.Station_ID not in temp1:
            i+=1
            temp1.append(p.Station_ID)
            print (p.Station_ID,"\t",p.dte,"\t",p.element,"\t",p.value)
        if i==5:
           break
    df_n2=sqlContext.sql("SELECT Station_ID, dte, element, value FROM table1
WHERE QFlag='' AND element='TMIN' ORDER BY value ASC LIMIT 30")
    print ("Coldest Stations\n")
    for p in df_n2.collect():
        if p.Station_ID not in temp2:
            j+=1
            temp2.append(p.Station_ID)
            print (p.Station_ID,"\t",p.dte,"\t",p.element,"\t",p.value)
        if j==5:
            break
    sc.stop()
```

The above code is executed on the data files per year individually using shell script as follows:

```
>for i in `seq 2000 2018`
>do
>echo “For the year: $i” >> Result.txt
>spark-submit –master yarn-client code.py /user/tatavag/PIIweather/$i.csv >> Result.txt
>done
```

This has taken almost **20min** to compute result on the other hand **HiveQL** executed in **2 min** only.

ii. Spark code to get Corresponding Stations for the Median temperature recorded per year:

```
from __future__ import print_function
import csv
import sys
from operator import add
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: Median_TMAX <ResultFileCSV> <DataFile>", file=sys.stderr)
        exit(-1)
    rows=[]
    sc = SparkContext(appName= "Median_TMAX")
    sqlContext = SQLContext(sc)
    with open(sys.argv[1]) as csvfile:
        csvreader = csv.reader(csvfile)
        for row in csvreader:
            rows.append(row)
    lines = sc.textFile(sys.argv[2])\
    .map(lambda l: l.split(","))\
    .map(lambda
p:Row(Station_ID=p[0],dte=p[1],element=p[2],value=int(p[3]),MFlag=p[4],QFlag=p[5],
SFlag=p[6],OBS_time=p[7],year=p[1][0:4]))
    df = sqlContext.createDataFrame(lines)
    for i in rows:

x=df.where(df.year==i[0]).where(df.element==i[1]).where(df.QFlag==").where(df.valu
e==int(i[2])).limit(1).collect()

print("Year:",x[0]['year'],"Date:",x[0]['dte'],"Station_ID:",x[0]['Station_ID'],"Measureme
nt:",x[0]['element'],"Median Value:",x[0]['value'],"\n")
        x=None
    sc.stop()
```

This code is executed as follows:

$ spark-submit –master yarn-client code.py <Query_ResultFile_AS_CSV> /user/tatavag/PIIweather > Result.txt

The Query_ResultFile_AS_CSV is the CSV format file of the result obtained from the Hive Query of the Bonus Task 1, is give as first input to the code and the data stored in Hadoop as second input file. The results are collected in Result.txt file.

Result.txt:

Year: 2000 Date: 20000101 Station_ID: USC00385946 Measurement: TMAX Median Value: 189

Year: 2001 Date: 20010101 Station_ID: USC00029376 Measurement: TMAX Median Value: 194

Year: 2002 Date: 20020101 Station_ID: USC00046144 Measurement: TMAX Median Value: 183

Year: 2003 Date: 20030101 Station_ID: USC00419361 Measurement: TMAX Median Value: 194

Year: 2004 Date: 20040101 Station_ID: USC00440993 Measurement: TMAX Median Value: 189

Year: 2005 Date: 20050101 Station_ID: USC00093578 Measurement: TMAX Median Value: 189

Year: 2006 Date: 20060101 Station_ID: USC00110187 Measurement: TMAX Median Value: 189

Year: 2007 Date: 20070101 Station_ID: USC00013575 Measurement: TMAX Median Value: 194

Year: 2008 Date: 20080101 Station_ID: USC00099027 Measurement: TMAX Median Value: 183

Year: 2009 Date: 20090101 Station_ID: USC00264341 Measurement: TMAX Median Value: 183

Year: 2010 Date: 20100101 Station_ID: USW00012876 Measurement: TMAX Median Value: 183

Year: 2011 Date: 20110101 Station_ID: USC00110187 Measurement: TMAX Median Value: 178

Year: 2012 Date: 20120101 Station_ID: USC00036768 Measurement: TMAX Median Value: 194

Year: 2013 Date: 20130101 Station_ID: USC00227128 Measurement: TMAX Median Value: 183

Year: 2014 Date: 20140101 Station_ID: USC00051121 Measurement: TMAX Median Value: 183

Year: 2015 Date: 20150101 Station_ID: USC00082944 Measurement: TMAX Median Value: 194

Year: 2016 Date: 20160101 Station_ID: USC00020288 Measurement: TMAX Median Value: 194

Year: 2017 Date: 20170109 Station_ID: USW00053182 Measurement: TMAX Median Value: 193

Year: 2018 Date: 20180101 Station_ID: USC00414382 Measurement: TMAX Median Value: 72

Year: 2000 Date: 20000119 Station_ID: USS0011R06S Measurement: TMIN Median Value: 47

Year: 2001 Date: 20010101 Station_ID: USC00046144 Measurement: TMIN Median Value: 50

Year: 2002 Date: 20020101 Station_ID: USR0000WCOU Measurement: TMIN Median Value: 44

Year: 2003 Date: 20030103 Station_ID: USS0003E05S Measurement: TMIN Median Value: 52

Year: 2004 Date: 20040101 Station_ID: USC00029376 Measurement: TMIN Median Value: 56

Year: 2005 Date: 20050101 Station_ID: USC00091266 Measurement: TMIN Median Value: 50

Year: 2006 Date: 20060101 Station_ID: USW00024229 Measurement: TMIN Median Value: 50

Year: 2007 Date: 20070101 Station_ID: USR0000CCAS Measurement: TMIN Median Value: 56

Year: 2008 Date: 20080101 Station_ID: USC00419361 Measurement: TMIN Median Value: 44

Year: 2009 Date: 20090101 Station_ID: USW00024229 Measurement: TMIN Median Value: 50

Year: 2010 Date: 20100105 Station_ID: USS0022D02S Measurement: TMIN Median Value: 49

Year: 2011 Date: 20110101 Station_ID: USC00093060 Measurement: TMIN Median Value: 50

Year: 2012 Date: 20120101 Station_ID: USC00042147 Measurement: TMIN Median Value: 56

Year: 2013 Date: 20130101 Station_ID: USC00416733 Measurement: TMIN Median Value: 44

Year: 2014 Date: 20140101 Station_ID: USR0000MMAR Measurement: TMIN Median Value: 50

Year: 2015 Date: 20150101 Station_ID: USR0000CCOO Measurement: TMIN Median Value: 61

Year: 2016 Date: 20160101 Station_ID: USR0000TSBS Measurement: TMIN Median Value: 56

Year: 2017 Date: 20170101 Station_ID: USC00385633 Measurement: TMIN Median Value: 56

Year: 2018 Date: 20180101 Station_ID: USC00458715 Measurement: TMIN Median Value: -33