Ex .1 - Reminders on database

1. A **join** clause in SQL – corresponding to a join operation in relational algebra – combines columns from one or more tables into a new table. ANSI SQL specifies five types of `JOIN`: `INNER`, `LEFT OUTER`, `RIGHT OUTER`, `FULL OUTER` and `CROSS`.
2. **Aggregations** operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.
3. We can find the meaning of each column in `2017.csv` from the format file provided on the official website.

```
ID = 11 character station identification code
YEAR/MONTH/DAY = 8 character date YYYYMMDD
ELEMENT = 4 character indicator of element type
DATA VALUE = 5 character data value for ELEMENT
```

In this homework, I run drill on Docker for simplicity.

Then we can write the following 3 nested queries.

```
select * from (select columns[0] as ID, columns[1] as dates,
columns[2] as c2, columns[3] as c3, columns[4] as c4,
columns[5] as c5, columns[6] as c6, columns[7] as c7 from
dfs.`/home/hadoop/Desktop/h4/weather/2017.csv`) limit 10;

select dates, count(*) as counts from (select columns[0] as ID,
columns[1] as dates from
dfs.`/home/hadoop/Desktop/h4/weather/2017.csv`) GROUP BY dates
limit 10;

select dates, ID from (select columns[0] as ID, columns[1] as
dates from dfs.`/home/hadoop/Desktop/h4/weather/2017.csv`)
ORDER BY dates limit 10;
```

Ex. 2 - Holidays!

1. I assume that I will carry out an one-week holiday during 8.10-8.16. The perfect weather is defined as days which have a precipitation under 30%, average temperature under 25 $^\circ\mathrm{C}$, min temperature above 15 $^\circ\mathrm{C}$ and max temperature under 30 $^\circ\mathrm{C}$.
2. We first find the columns representing precipitation, average temperature and daily temperature in `2017.csv`.

   ```
   PRCP: Precipitation
   MNPN: Min temperature
   MXPN: Max temperature
   TAVG: Average temperature
   ```

   First, select the week from 8.10-8.16 and create a temp table called `one-week`.

   ```
   CREATE TABLE dfs.tmp.`one-week` AS SELECT * FROM (select
   columns[0] as ID, columns[1] as day, columns[2] as c2,
   columns[3] as c3, columns[4] as c4, columns[5] as c5,
   columns[6] as c6, columns[7] as c7 from
   dfs.`/home/hadoop/Desktop/h4/weather/2017.csv`) where day in
   ('20170810','20170811','20170812','20170813','20170814','201708
   15','20170816');
   ```

   Secondly, we select average temperature and create a temp table called `TAVG`.

   ```
   create table dfs.tmp.`TAVG` as select * from dfs.tmp.`one-week`
   where c2 = 'TAVG';
   ```

   Thirdly, we select the precipitation of the week and create a temp table called `PRCP_SUM`

   ```
   create table dfs.tmp.`PRCP_SUM` as select ID, sum(cast(c3 as
   int)) as sum from dfs.tmp.`one-week` where c2 = 'PRCP' group by
   ID;
   ```

   Next, we consider the max and min temperature and calculate the temperature amplitude.

   ```
   create table dfs.tmp.`min` as select ID,CAST(AVG(CAST(c3 as
   INT)) as INT) as tmin from dfs.tmp.`one-week` where c2 = 'MNPN'
   group by ID;
   ```

   ```
   create table dfs.tmp.`max` as select ID,CAST(AVG(CAST(c3 as
   INT)) as INT) as tmax from dfs.tmp.`one-week` where c2 = 'MXPN'
   group by ID;
   ```

```
create table dfs.tmp.`amp` as select tmin.ID, tmin.tmin,
tmax.tmax
from dfs.tmp.`min` as tmin
join dfs.tmp.`max` as tmax on tmin.ID = tmax.ID;
```

Combine all the above records and create a temp table called `all`.

```
create table dfs.tmp.`all` as
select t1.ID, t1.tmin, t1.tmax, t2.sum as prcp from
dfs.tmp.`amp` as t1
join
(select ID, sum from dfs.tmp.`PRCP_SUM`) as t2
on t1.ID = t2.ID;
```

Then, we check these records that fit the previous definition of "perfect weather",

```
# temperature are in tenths.
select * from dfs.tmp.`all` where prcp < 30 and tmin > 150 and
tmax < 300;
```

Result:

```
'ID','tmin','tmax','prcp'
'USC00247560','161','298','5'
'USC00146333','261','261','0'
'USC00356550','156','289','15'
'USC00049026','164','164','0'
'USC00457941','155','280','5'
'USC00450587','152','276','5'
'USC00193276','182','259','18'
7 rows selected (0.278 seconds)
```

Finally, let us check these ID's average temperature.

```
select * from dfs.tmp.`TAVG` where ID in
('USC00247560','USC00146333','USC00356550','USC00049026','USC00
457941','USC00450587','USC00193276');
```

However, there are no record in `TAVG` with the above IDs. Anyway, we can check these ID's corresponding location since we already have the min and max temperature and the average temperature is not that important after all.

ID	Location
USC00247560	SIDNEY
USC00146333	PERRY LAKE
USC00356550	PENDLETON WFO
USC00049026	TRINITY RVR HATCHERY
USC00457941	SPOKANE WFO
USC00450587	BELLINGHAM
USC00193276	GROVELAND

Ex. 3 - Data visualization

1. First, we select all the continent information and create a table called `continent`

   ```
   create table dfs.tmp.`continent` as select * from (select
   columns[0] as name, columns[1] as continent, columns[2] as abre
   from
   dfs.`/home/hadoop/Desktop/h4/weather/country_continent.csv`);
   ```

   Since the first 2 bits of the `ID` stand for the abbreviation of this ID's location, we could extract only the first 2 bits of the ID as well as the average temperature from `2017.csv`.

   ```
   create table dfs.tmp.`continent_TAVG` as select substr(ID,1,2)
   as abre, c2, c3 from (select columns[0] as ID, columns[2] as
   c2, columns[3] as c3 from
   dfs.`/home/hadoop/Desktop/h4/weather/2017.csv`) where c2 =
   'TAVG';
   ```

   We join these two tables.

   ```
   create table dfs.tmp.`TAVG_CONTINENT` as select t2.*,
   t1.continent from dfs.tmp.`continent` as t1
   join dfs.tmp.`continent_TAVG` as t2
   on t1.abre = t2.abre;
   ```

   Finally, we calculate the variance of average temperature.

   ```
   select continent, cast(variance(cast(c3 as INT)) as
   decimal(10,2)) as var from dfs.tmp.`TAVG_CONTINENT` group by
   continent;
   ```

The result is:

```
'continent','var'
'AS','14995.22'
'AF','4537.85'
'EU','16386.39'
'OC','5172.56'
'SA','5177.65'
'AN','37091.48'
'NA','13692.09'
7 rows selected (1.428 seconds)
```

2. We calculate the average temperature of continents.

```sql
select continent, cast(AVG(cast(c3 as INT)) as decimal(10,2))
as AVRG from dfs.tmp.`TAVG_CONTINENT` group by continent;
```

The result is:

```
'continent','AVRG'
'AS','180.71'
'AF','236.75'
'OC','203.96'
'EU','54.58'
'AN','-146.73'
'NA','79.70'
'SA','212.94'
7 rows selected (1.759 seconds)
```

Check the `result` folder for plots. All the units are in **tenths** of degree.