**Problem 1 - Data Modeling**

**-------------------------**

**Choose a database to use for this coding exercise (SQLite, Postgres, etc.). Design a data model to represent the weather data records. If you use an ORM, your answer should be in the form of that ORM's data definition format. If you use pure SQL, your answer should be in the form of DDL statements.**

I have chosen a MySQL database for storing the raw data. I have created two tables

‘weather\_data’ - to store the weather data

‘weather\_stats’ – to store the weather stats data

**Weather Data Model :**

class Weather(models.Model):

    station\_id = models.CharField(primary\_key=True,max\_length=20)

    date = models.DateField()

    max\_temp = models.IntegerField() #maximum temperature

    min\_temp = models.IntegerField() #minimum temperature

    rain = models.IntegerField()

**Weather Table Schema :**

CREATE TABLE `weather\_data` (

`station\_id` char(20) NOT NULL,

`date` date NOT NULL,

`max\_temp` float DEFAULT NULL,

`min\_temp` float DEFAULT NULL,

`rain` float DEFAULT NULL,

PRIMARY KEY (`station\_id`,`date`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

**Weather Stats Data Model :**

class WeatherStats(models.Model):

    station\_id = models.CharField(primary\_key=True, max\_length=20)

    year = models.IntegerField()

    avg\_min\_temp = models.DecimalField(decimal\_places=2,max\_digits=8) #Average minimum temperature

    avg\_max\_temp = models.DecimalField(decimal\_places=2,max\_digits=8) #Average maximum temperature

    total\_rain = models.DecimalField(decimal\_places=2,max\_digits=8)

**Weather Stats Table DDL :**

CREATE TABLE `weather\_stats` (

`station\_id` char(20) NOT NULL,

`year` int NOT NULL,

`avg\_min\_temp` float DEFAULT NULL,

`avg\_max\_temp` float DEFAULT NULL,

`total\_rain` float DEFAULT NULL,

PRIMARY KEY (`station\_id`,`year`)

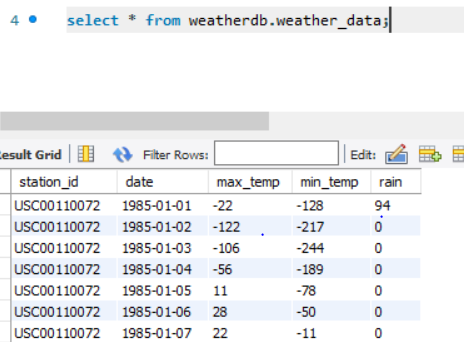
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci

**Problem 2 - Ingestion**

**---------------------**

**Write code to ingest the weather data from the raw text files supplied into your database, using the model you designed. Check for duplicates: if your code is run twice, you should not end up with multiple rows with the same data in your database. Your code should also produce log output indicating start and end times and number of records ingested.**

I have created the python code to ingest the weather data from the raw text files supplied into the “weatherdb” database and table “weather\_data” . File name considered as “station id” field and remaining three fields (max\_temp, min\_temp,amount of precipitation) exists in file on that data dumped into weather\_data table.



**Weather\_data Table**

Script file : datadump.py

In the above script file, this is the method to create the table and insert those data records into that table

**create\_weatherdb()**

**Problem 3 - Data Analysis**

**-------------------------**

**For every year, for every weather station, calculate:**

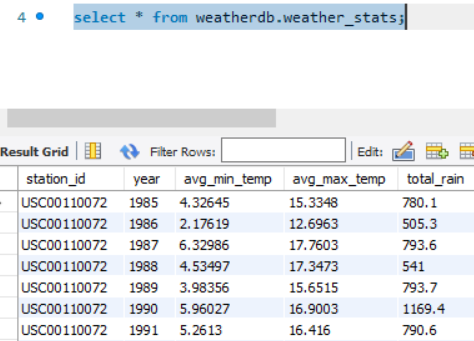
**\* Average maximum temperature (in degrees Celsius)**

**\* Average minimum temperature (in degrees Celsius)**

**\* Total accumulated precipitation (in centimeters)**

For Weather Stats Data, calculated the average maximum temperature, average minimum temperature,

Total accumulated precipitation based on weather data. Updated the fields with Null values if value is -9999



**Weather\_stats Table**

Script file : datadump.py

In the above scripts, I have created a method “create\_weather\_stats()” to calculate the avg max temperature, avg min temperature and Total accumulated Precipitation. After the calculation, the data is ingested into this table “weather\_stats”

**Problem 4 - REST API**

**--------------------**

**Choose a web framework (e.g. Flask, Django REST Framework). Create a REST API with the following GET endpoints:**

**/api/weather**

**/api/weather/stats**

**Both endpoints should return a JSON-formatted response with a representation of the ingested/calculated data in your database. Allow clients to filter the response by date and station ID (where present) using the query string. Data should be paginated.**

I have created two rest api end points.

* /api/weather – to fetch the weather related records
* /api/weather/stats – to fetch the weather stats records.

Each request should return response based on the filter by station id and date using Django filter Backend. I have created custom pagination with page size 10 . If you want to increase the page size, make changes in settings file for a variable “PAGE\_SIZE”.