Corteva

Code Challenge   
Response Document

Date 04/02/2023



Moser Consulting, Incorporated

6220 Castleway West Drive

Indianapolis, IN 46250

Phone: 317-596-8022

Table of Contents

[Problem 1 – Data modeling 1](#_Toc131350694)

[Analysis of Problem 1](#_Toc131350695)

[Database Platform 1](#_Toc131350696)

[Data Mode 1](#_Toc131350697)

[Database Scripts 1](#_Toc131350698)

[Problem 2 – Ingestion 2](#_Toc131350699)

[Analysis of Problem 2](#_Toc131350700)

[Ingestion Code 2](#_Toc131350701)

[Problem 3 – Data Analysis 3](#_Toc131350702)

[Analysis of Problem 3](#_Toc131350703)

[Data Model 3](#_Toc131350704)

[Data Ingestion 3](#_Toc131350705)

[Problem 4 – REST API 3](#_Toc131350706)

[Analysis of Problem 3](#_Toc131350707)

[Code Base 3](#_Toc131350708)

[AWS Architecture 3](#_Toc131350709)

[Analysis of Problem 3](#_Toc131350710)

[Proposed Architecture 4](#_Toc131350711)

# Problem 1 – Data modeling

## Analysis of Problem

The challenge contains two primary data sets; one for crop yields by year and the second weather data records from various weather stations by date.

The problem is to choose a database to place the data, design a model and to present to code required to establish the model in whatever form that is decided.

## Database Platform

I choose to use MySQL as the database platform. This is an open-source database that is readily available and easy to install and run locally on nearly any machine. In addition, MySQL can be migrated to most cloud vendors without much effort. PostgreSQL was considered, as it also meets these same basic attributes. The deciding factor was simply that I had a MySQL database already established and running on my local machine.

## Data Mode

The data provided is in a relatively simple model and it would have been possible to model tables directly after the file structure. I decided instead to apply a standard dimensional model to the problem. As the data is primarily for analysis by placing the data into a dimensional model it allows for expansion later in the lifecycle as needed. To establish the model, I applied the standard process of identifying the fields that are dimensional and measurable in nature. Those that are dimensional I pulled into tables and those that are measurable went into a fact table. Here is the basic model I ended up with.

Diagram

Description automatically generated

The Yield and Weather facts were kept separated based on the fact that weather is daily and related to stations, while yield is yearly and has no other relationship.

## Database Scripts

I considered using SQLAlchemy in order to produce the tables, but based on the number of tables, complexity of the design and ease that a consumer might have setting up an environment, I decided to simply provide SQL scripts. These can be found in src folder under DatabaseScripts.

# Problem 2 – Ingestion

## Analysis of Problem

The challenge asked to load the weather data into the model from the first problem. At first, I limited my code to just this one part of the data. However, as I got deeper into the challenge, I found that the dimension from the crop yield data and the analysis problem was shared. Based on this I went ahead and loaded both the weather date and the yield data into the model. Having both data allowed for the tie between the yearly weather rollup data and the yearly yield data.

Code is to log output including start and finish times along with number of records.

## Ingestion Code

I decided that Python was a good fit to load the data from the flat files provided into the data model. The solution floe is outlined below. In addition, the code is setup to log information to a log file. An example can be found in the log directory. It contains start and end comments for the main job and for each step function in the code block. Number of records for each step is also included.

Diagram

Description automatically generated

I decided to write and execute the code using Visual Studio and Juypter Notebook. This allowed for each step to be explained to any developer that might take on the code base in the future. In addition, a Pandas data frame was maintained to capture details about the job for validation pruposes.

# Problem 3 – Data Analysis

## Analysis of Problem

The challenge is to establish a rollup of the weather data for every year and every weather staion to provide the following:

Average maximum temperature (in degrees Celsius)

Average minimum temperature (in degrees Celsius)

Total accumulated precipitation (in centimeters)

Ignore missing data and store the results in a new model. Use null for statistics that cannot be calculated.

## Data Model

To store the results, I added a new fact table and leveraged the existing year dimension table from the yield solution. This resulted in adding a new fact weather rollup table. The script for the table is included with the database scripts.

## Data Ingestion

The code to load the table is included in the Juypter Notebook code provided. The solution leveraged utilizing the power of the RDBMS engine to calculate the results and store them in the table.

# Problem 4 – REST API

## Analysis of Problem

The problem is to provide a RESTful API for weather and weather stats from the fact weather table and fact weather rollup table respectfully.

The API need to be setup to respond on /api/weather and /api/weather/stats.

API is to return JSON responses and allow query by date and weather station. Results should be paginated.

## Code Base

To establish a local API to meet the requirements I utilized Flask and SQL Alchemy. Swagger documents were created using flasgger and YAML configuration files.

# AWS Architecture

## Analysis of Problem

To purpose how to take the solution and port to AWS cloud and identify the tools needed to do that. The ingestion needs to be a scheduled version of the code.

## Proposed Architecture

The easiest way to deploy this on AWS would be to utilize RDS for the database and migrate the database tables to the cloud. The ingestion Python code could be moved to AWS Glue and S3 used for inbound and archive file locations. AWS Glue allows for scheduling the runs on the needed intervals. AWS API Gateway would be ideal to place the API onto