**This coding exercise will help us understand how you approach some of the common problems we see in data engineering. Please approach it as you would a work assignment: ask questions if things are unclear, use best practices and common software patterns, and feel free to go the extra mile to show off your skills.**

**You will be asked to ingest some weather and crop yield data (provided), design a database schema for it, and expose the data through a REST API. You may use whatever software tools you would like to answer the problems below, but keep in mind the skills required for the position you are applying for and how best to demonstrate them. Read through all the problems before beginning, as later problems may inform your approach to earlier problems.**

**You can retrieve the data required for this exercise by cloning this repository:**

**https://github.com/corteva/code-challenge-template**

**Weather Data Description**

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

**The wx\_data directory has files containing weather data records from 1985-01-01 to 2014-12-31. Each file corresponds to a particular weather station from Nebraska, Iowa, Illinois, Indiana, or Ohio.**

**Each line in the file contains 4 records separated by tabs:**

**1. The date (YYYYMMDD format)**

**2. The maximum temperature for that day (in tenths of a degree Celsius)**

**3. The minimum temperature for that day (in tenths of a degree Celsius)**

**4. The amount of precipitation for that day (in tenths of a millimeter)**

**Missing values are indicated by the value -9999.**

**Yield Data Description**

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

**The yld\_data directory has a single file, US\_corn\_grain\_yield.txt, containing a table of the total harvested corn grain yield in the United States measured in 1000s of megatons for the years 1985-2014.**

**---**

**Problem 1 - Data Modeling**

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

**Choose a database to use for this coding exercise (SQLite, Postgres, etc.). Design two data models: one to represent the weather data records, and one to represent the yield 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.**

**Solution: Created two models one is Weather and another one is Yield**

**These are Django ORM models**

**Problem 2 - Ingestion**

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

**Write code to ingest the weather and yield data from the raw text files supplied into your database, using the models 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.**

**Solution: The data is ingested using a custom migration script which will take the data from the text file and insert all the data into the database.**

**Need to run following commands to load the data into database:**

**python3 manage.py load\_weather\_data**

**python3 manage.py load\_yield\_data**

**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)**

**Solution: Year wise average maximum, minimum temperature and total accumulated precipitation is calculated and shown through api urls:**

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

**Done. Link: localhost:8000/USC00110072/avg\_max\_temp/2008**

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

**Done. Link: http://127.0.0.1:8001/USC00110072/avg\_min\_temp/2008**

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

**Done. Link:http://127.0.0.1:8001/USC00110072/avg\_precipitation/2008**

**The logic used Avg Max temperature is total max temperature divided by number of temperatures**

**Avg Min temperature is total max temperature divided by number of temperatures**

**Avg Min temperature is total precipitation divided by number of precipitation**

**Ignore missing data when calculating these statistics.**

**Missing data is handled within the api end**

**Design a new data model to store the results. Use NULL for statistics that cannot be calculated.**

**Your answer should include the new model definition as well as the code used to calculate the new values and store them in the database.**

**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/yield**

**/api/weather/stats**

**Each endpoint should return a JSON-formatted response with a representation of the ingested 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.**

**Your answer should include all files necessary to run your API locally, along with any unit tests.**

**Solution: The restful apis are created and unit tested and api urls are:**

**Build the api:**

**http://127.0.0.1:8000/api/yield/2000**

**http://127.0.0.1:8000/api/weather/USC00110072/19850109**

**The data is paginated response and used django lint and code formatter and include all the test cases.**

**---**

**Submitting Your Answers**

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

**Consider using a linter, code formatter, and including tests and code comments. Anything that helps us understand your thought process is helpful!**

**Please provide us with a link to your Git repository, hosted on GitHub/GitLab, containing your answers and code.**