Project 2: Data Exploration and Cleaning
CSC 6605 ML Production Systems

Overview

In this part of the project, you're going to access main data from the database, explore it, and determine what needs to be done to clean it. You will also prepare and upload complementary data sets to the database.

Instructions

Part I: Create Project Repository on GitHub

Your source code will be stored in a private repository on GitHub. Your instructor will have access to this repository to be able to review your code when grading each part of the project.

1. Create a new, **private** GitHub repository.

2. Add your group members and instructor (rnowling) as collaborators to the repository. Ensure that your group members have write access.

In my repository, I have a directory for each component which correspond 1-to-1 with Docker images.

```
$ ls -w 4
ansible-playbooks
cleaning-job
cleaning-job-setup
cleaning-job-test
complementary-data-ingestion
complementary-data-setup
data
docs
evaluation-pipeline
joining-job
joining-job-setup
joining-job-test
LICENSE
notebooks
prediction-service
README.md
simulator
simulator-setup
simulator-test
```

The main components you will implement will be a cleaning and enrichment batch job, a prediction service, an evaluation pipeline, and some Jupyter notebooks. A Docker image is created for each component (except the notebooks), and each component is run in a Docker container. I chose to automate starting and stopping components using Ansible playbooks. (I won't cover Ansible in this class, but you can certainly look if you are interested.) I also automated one-off tasks such as creating database tables (the components ending in -setup). These are also packaged as Docker images and run inside of containers. The notebooks directory contains notebooks I used for EDA and model development. Lastly, I include some documentation such as instructions for running the system in the README.md file and a brief summary of the architecture with a diagram (saved as an image file in the docs directory).

Part I: Create a local Python Virtual Environment

1. On your local machine, create a Python virtual environment and install dependencies:

```
$ python3 -m venv csc6605
$ source csc6605/bin/activate
$ pip install -U pip wheel build
$ pip install jupyter pandas seaborn scikit-learn "psycopg[binary]" marshmallow
```

2. Run Jupyter.

```
$ jupyter notebook
```

Part II: Download the Sales Events

1. SSH into your group's VM and forward port 5432 to your local machine:

```
$ ssh -L 5432:localhost:5432 -p 22GG student@dh-arm.hpc.msoe.edu
```

2. Check out your GitHub repo to your local machine.

3. Create a notebook directory in your GitHub repo.

4. Create a Jupyter notebook in the notebook directory, connect to the database, and download all of the records:

```
from matplotlib import pyplot
import seaborn as sns
import pandas as pd

import psycopg
from psycopg.rows import dict_row

user = "postgres"
password = "psql-password"
host = "127.0.0.1"

conn_string                                                    =
"postgresql://{}:{}@{}:5432/house_price_prediction_service".format(user,
password, host)
with psycopg.connect(conn_string, row_factory=dict_row) as conn:
      with conn.cursor() as cur:
            cur.execute("SELECT      id,      event_date,      data      FROM
            raw_home_sale_events;")

            # rows will be a list of dicts with the keys "id", "event_date",
            and "data"
            rows = cur.fetchall()
```

5. Create a Pandas DataFrame of the records

```
# Add all the ids to the inner data dictionaries and extract them
json_events = []
for row in rows:
      event = row["data"]
      event["id"] = row["id"]
      json_events.append(event)

df = pd.DataFrame.from_records(json_events)
```

Part III: Explore and Clean the Data

Explore the home sale event data using Pandas and Seaborn. The dataset contains records for home sales in King County, Washington from May 2014 to May 20215. The data were downloaded from Kaggle. Here is some information about the variables in the data:

- **id -** Unique id for each home sold
- **date -** Date of the home sale
- **price -** Price of each home sold
- **bedrooms -** Number of bedrooms
- **bathrooms -** Number of bathrooms
- **sqft_living -** Squared footage of the apartments interior living space
- **sqft_lot -** Squared footage of the land space
- **floors -** Number of floors
- **waterfront -** A dummy variable for whether the apartment was overlooking the waterfront or not
- **view -** An index from 0 to 4 of how good the view of the property was (higher is better)
- condition - an index from 1 to 5 on the condition of the apartment
- **grade -** An index from 1 to 13 where 1-3 falls short of building construction and design, 7 has an average level of construction and design , and 11-13 have a high quality level of construction and design
- **sqft_above -** the square footage of the interior housing space that is above ground level
- **sqft_basement -** the square footage of the interior housing space that is below ground level
- **yr_built -** The year of the house was initially built
- **yr_renovated -** The year of the house's last renovation
- **zipcode -** What zipcode area the house is in
- **lat -** Latitude
- **long -** Longitude
- **sqft_living15 -** The square footage of interior housing living space for the 15 nearest neighbors
- **sqft_lot15 -** The square footage of the land lots of the 15 nearest neighbors

1. Come up with a one sentence description for each column describing its purpose based on its name and values.

2. Identify the appropriate types (e.g., int, float, Boolean, string, date, categorical, etc.) for each column.

3. Check the range of values for each numerical field using the min / max values from df.describe() and distribution of values using Seaborn's histplot(). Do all of the values look reasonable to you? Are any values unexpected? Are any values (e.g., 0's) likely being used as placeholders for missing data?

4. Identify records with unexpected values. Unexpected values can from typos or outliers (e.g., very large properties with few examples). Look at the values of other fields. It is likely that more than one field will have an unexpected value.

5. Determine ranges of expected values for each numerical column. At a later point, you will drop rows that have values (but not null values) outside of these ranges.

Part IV: Prepare and Upload Complementary Data

You are provided with two complementary data sets that you will join with the home sale events in a later stage. You need to prepare and upload these data sets to the database.

1. Create the appropriate tables.

```
CREATE TABLE cleaned_zipcode_populations (
                zipcode CHAR(5) UNIQUE,
                population INTEGER NOT NULL);

CREATE TABLE cleaned_zipcode_public_schools (
                zipcode CHAR(5) UNIQUE,
                high_schools INTEGER NOT NULL,
                middle_schools INTEGER NOT NULL,
                primary_schools INTEGER NOT NULL,
                other_schools INTEGER NOT NULL,
                unknown_schools INTEGER NOT NULL,
                total_schools INTEGER NOT NULL);
```

You will also need to create indices on the zipcode columns for fast retrieval.

2. In a Jupyter notebook, load the data. When you load the data sets, make sure that you specify that the zipcode columns should be interpreted as strings, not integers. You can accomplish this by passing the type to the read_csv() method:

```
raw_population_df = pd.read_csv("population_by_zip_2010.csv",
                                 usecols=["population", "zipcode"],
                                 dtype={ "zipcode" : str })

raw_school_df = pd.read_csv(flname,
                            usecols=["ZIP", "LEVEL_"],
                            dtype= { "ZIP" : str })
```

3. Aggregate the data to get population counts and the number of schools of each level by zip code. Calculate the total number of schools.

4. Insert the data into the database:

```
with psycopg.connect(conn_string) as conn:
        with conn.cursor() as cur:
            for row in population_df.to_dict(orient="records"):
                cur.execute("""INSERT INTO cleaned_zipcode_populations
                               (zipcode, population)
                               VALUES
                               (%(zipcode)s, %(population)s);""",
                            row)
            conn.commit()
```

The to_dict(orient="records") method convers the DataFrame to a list of dictionaries with an entry for each column. The cur.execute() method will expect row to be a dictionary with the keys "zipcode" and "population".

Repeat this for the public school data.

Demonstration and Submission

Create a 2-3 page (or longer if needed) report describing the data, your interpretations of the columns, and any data issues you found. Include a table with the column names, descriptions, types, whether you found nulls and their placeholder values (if found), and cutoffs for reasonable minimum and maximum values. Include a SQL CREATE TABLE statement for the home sale events with appropriate column names, types, and nullability modifiers for each variable. (You do not need to create the home sale event table in the database. You are using the statement to communicate your inferred schema.)

Submit your report as PDF and notebooks as HTML files to Canvas. Your instructor will check that the complementary data were uploaded correctly into PostgreSQL.

Rubric

Description	Percentage
Written Report • Report is written in a professional manner using proper grammar and spelling. • Report is a useful standalone document that can be shared with a business partner.	15%
Plots • Appropriate types of plots were chosen for each analysis. • Axes are properly labeled. • Used legends if appropriate. • Chose appropriate axis limits to make plots readable and avoid misleading interpretations. • Font sizes are legible. • Figures are saved at high resolutions.	15%
Data Exploration and Cleaning • Home sale event data set were evaluated to identify data problems. • Appropriate descriptions have been written for each field.	30%

• Appropriate types have been determined. • Fields have been appropriately identified as nullable / not nullable. • Appropriate value ranges have been determined for each field. • Criteria for fixing or filtering out records with data problems have been determined. • Provided a SQL CREATE statement specifying the data model for the cleaned data.	
Complementary Data • Population and school data are cleaned and aggregated by zip code. • Appropriate tables have been created in PostgreSQL. • The data have been uploaded into PostgreSQL. • Collections have been indexed by zip code for fast retrieval.	30%
GitHub Repository • Created private repository. • Gave the instructor access to the repository. • Repository has a notebook directory with the two notebooks created for this assignment.	10%