Q1 Pandas/NumPy

20 Points

For Q1 and Q2, you are required to use Pandas and NumPy, and the least amount of other Python code as may be absolutely needed.

For Q1 and Q2, assume the following imports:

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
```

You are given a dataframe **df** of daily new cases of Covid-19 infections in a certain number of the states of the U.S. Each row holds the new cases for one day, the columns are the states. The dataframe is indexed starting at 0, where 0 is the first day for which data was recorded. Days are in sequence without jumps, i.e. the first row is for the first day, second row for the second day, third row for the third day, etc.

Perform the following actions on **df**, in the given sequence.

Q1.1

5 Points

Replace all missing values (NaNs) in **df** with the data for the previous day for that state. If there is a succession of days for which values are missing, they should all get the value for the last (latest) day in the past for which there was available data. Assume that there are no missing values for the first day.

```
df = df.fillna(method='ffill')
```

Q1.2
10 Points

For each of the states in the updated **df** after Q1.1, get the day that recorded the biggest change (increase or decrease) from the previous day. For instance, if for some state X, the largest change was from day (d-1) to day d, then the result is day d. Store the result in an ndarray named **max_jumps**.

Q1.3
5 Points

To the updated **df** after Q1.1, add a column that holds in each row the sum of all new cases for that day.

```
df['Sum_Cases'] = df.sum(axis = 1, skipna = True)
```

Q2 Pandas/NumPy
43 Points

A dataframe **df** stores sales information for sedans sold in the US in 2020. The columns of **df** are 'Make', 'Model', 'Price', and 'Category', in that order. 'Price' is the only numerical column, the others are all non-numerical. The 'Category' column holds one of four values: 'HP' (high performance), 'LX' (luxury), 'MR' (mid-range), and 'EC' (economy).

Perform the following sequence of actions on **df**

Q2.1
8 Points

Remove rows from **df** that have missing values for *both* 'Price' and 'Category'.

```
df = df.dropna(  how = "all" , subset = ["Price", "Category"] )
```

Q2.2
13 Points

On the updated **df** after Q2.1, replace missing (NaN) values in the 'Price' column with the average price of cars in the same category. For instance, if there is a missing price in the category 'LX', replace it with the average price of cars in the 'LX' category.

```
df["Price"] = df['Price'].fillna(df.groupby('Category')
['Price'].transform('mean'))
```

Q2.3
22 Points

On the updated **df** after Q2.2, replace missing category values (NaNs) by basing category on price range. Assume that the maximum possible price for a category is the maximum of all prices currently listed in that category. You may assume that no two categories overlap in price range.
(Hint: You can check if an isolated non-numerical cell value is NaN with the Python function `isinstance(value,str)`, which returns

`False` if value is NaN, `True` otherwise.)

```
def update_category(df):
    max_HP = df.groupby('Category')["Price"].max()["HP"]
    max_LX = df.groupby('Category')["Price"].max()["LX"]
    max_MR = df.groupby('Category')["Price"].max()["MR"]
    max_EC = df.groupby('Category')["Price"].max()["EC"]

    for index in df.index:
        category = str(df.loc[index, "Category"])
        if category == "nan" or "NaN":
            price = float(str(df.loc[index, "Price"]))
            if price <= max_EC:
                df.loc[index, "Category"] = "EC"
            elif price <= max_MR:
                df.loc[index, "Category"] = "MR"
            elif price <= max_LX:
                df.loc[index, "Category"] ="LX"
            elif price <= max_HP:
                df.loc[index, "Category"] = "HP"
    return df
```

Q3 SQL Database Schema

28 Points

Design the schema (tables) for a bookstore database described below.

Each book has a title, one or more authors, a unique ISBN (International Standard Book Number, exactly 13 digits long), a publication date, a format (which could be either hardcover, paperback, ebook, or audiobook), a price, and number of copies sold. There could be more than one book with the same title.

The original format of any book is hardcover. In other words, a book is always available in hardcover, but it may or may not have been published in other formats. Each format of a book has a different ISBN, and the publication date, price, and copies sold would vary by format. The title and authors for a book are the same across all formats.

Each author has a name (first+last combined), unique numeric id (assigned by bookstore for accounting purposes), and email.

Each ISBN is published by a publishing company (uniquely named). Different formats (ISBNs) of the same book may have different publishers.

Write the **create table** statements for all the tables in your schema. Be sure to choose the most appropriate/precise data type for each column, and whether it can be null or not. Be sure to specify primary key and unique columns as appropriate.

Your schema should be minimally redundant in storage of data, and should allow for effective querying.

```
CREATE TABLE BookAuthor(
    Authors VARCHAR(100) NOT NULL,
    title VARCHAR(100) NOT NULL,
    PRIMARY KEY (Authors, title)
);

CREATE TABLE pub_comp (
    pub_comp_name VARCHAR(200) NOT NULL,
    isbn_num INT(13) NOT NULL,
    PRIMARY KEY (isbn_num, pub_comp_name)
    FOREIGN KEY (isbn_num) REFERENCES
BookAuthor(isbn_num),
);
```

```
CREATE TABLE Books (
    num_sold INT(20) NOT NULL,
    title VARCHAR(100) NOT NULL,
    Authors VARCHAR(100) NOT NULL,
    b_type VARCHAR(100) NOT NULL,
    p_date DATE NOT NULL,
    price float(20) NOT NULL,
    isbn_num INT(13) NOT NULL,
    PRIMARY KEY (isbn_num),
    foreign key (Authors, title) REFERENCES
BookAuthor(Authors, title)
);

CREATE TABLE WriterTable(
    Authors VARCHAR(200) NOT NULL,
    isbn_num INT(13) NOT NULL,
title VARCHAR(200) NOT NULL,
    e_mail VARCHAR(200) NOT NULL,
    numeric_id INT(15) NOT NULL,
    PRIMARY KEY (numeric_id),
    FOREIGN KEY (isbn_num) REFERENCES Books(isbn_num),
    FOREIGN KEY (Authors, title) REFERENCES
BookAuthor(Authors, title)
);
```

Q4 SQL Database Queries

59 Points

You are given the following database schema of tables:

Hotel (HotelNo, HotelChainName, City, Country)
- HotelNo (hotel number) is the primary key
- HotelChainName is unique

- A hotel chain has one or more hotel locations identified by city and country

Room (RoomNo, HotelNo, Type, Price)
- RoomNo is the room number
- HotelNo references HotelNo of the Hotel table
- The combination of RoomNo + HotelNo is unique

Guest(GuestNo, GuestName, GuestCity, GuestCountry)
- GuestNo (guest number) is the primary key
- GuestCity and GuestCountry identify where the guest is from

Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)
- Each row stores a guest booking for a room in a hotel
- The RoomNo+HotelNo combination for any row is present in the Room table
- GuestNo references GuestNo of the Guest table
- The combination of HotelNo, GuestNo, DateFrom, and RoomNo is unique
- Assume DateFrom and DateTo are within the same month, and they are each stored using the `date` type

Assume that all bookings have resulted in a stay for the entire duration of the booking (in other words, there were no cancellations or incomplete stays).

Note: Dates can be compared numerically for chronological ordering using the arithmetic relational operators such as <, <=, >, >=, and =. Also, you can use the SQL function `CURRENT_DATE()` to get the current date.

Using this schema, write SQL queries for each of the following.

Note: For partial credit, make sure that the foundational structure of your query is correct: choose the correct table(s), and if a join is

needed, use the correct joining conditions for cross-table columns. If the foundation is incorrect, partial credit will be significantly impacted.

Q4.1

5 Points

What is the average price of a hotel room in London, England?

```
SELECT AVG(price) as "Average_Price_Room"
FROM Room table_r_1
INNER JOIN Hotel table_h_1
ON (table_r_1.HotelNo = table_h_1.HotelNo)
where table_h_1.Country = 'England' and table_h_1.City =
'London'
GROUP BY table_h_1.HotelNo;
```

Q4.2

7 Points

How many different guests from New York, USA, have made bookings for the month of August for any year in any hotel?

```
SELECT COUNT(table_g.guestNo)
FROM Guest table_g
INNER JOIN Booking table_book
ON table_book.GuestNo = table_g.GuestNo
WHERE table_g.GuestCity = "New York" AND
table_g.GuestCountry = "USA" AND
MONTH(table_book.DateFrom) = "August"
```

Q4.3

12 Points

Get all rooms with corresponding hotel numbers that are currently unoccupied at the Hilton chain of hotels.

```
SELECT table_room.RoomNo, table_room.HotelNo
FROM Room table_room
INNER JOIN join Hotel table_hotel
ON table_hotel.HotelNo = table_room.HotelNo
WHERE t2.HotelChainName = "Hilton" AND
table_room.RoomNo NOT IN
(
SELECT RoomNo IN Booking
);
```

Q4.4

9 Points

Get the country-wise revenue for each hotel chain for the year 2000. The result should have columns named 'HotelChain', 'Country' and 'Revenue'

```
SELECT hotel_inner.HotelChainName, hotel_inner.Country, ,
SUM(table_book.price) as "Revenue"
FROM Room table_room, Booking table_book
INNER JOIN Hotel hotel_inner
ON (hotel_inner 1.HotelNo = table_room.HotelNo)
```

Q4.5

14 Points

Get the guest numbers of guests who have booked every type of room listed in the database.