SIT220/731 2022.T3: Task 5D

Working with *pandas* Data Frames Part 2

Last updated: 2022-12-12

Contents

1 Task

This task is related to Section 4.6 (see the *Learning Resources* on the unit site; see also Chapter 13 of our book). In case of any problems/questions, do hot hesitate to attend our on-campus/online classes. The hard deadline is Week 11 (Friday).

We will study the dataset called nycflights13. It gives information about all 336,776 flights that departed in 2013 from the three New York (in the US) airports (EWR, JFK, and LGA) to destinations in the United States, Puerto Rico, and the American Virgin Islands.

Download the following data files from our unit site (*Learning Resources → Data*):

- nycflights13_flights.csv.gz – flights information,
- nycflights13_airlines.csv.gz – decodes two letter carrier codes,
- nycflights13_airports.csv.gz – airport data,
- nycflights13_planes.csv.gz – plane data,
- nycflights13_weather.csv.gz – hourly meteorological data for LGA, JFK, and EWR.

Refer to the comment lines in the CSV files (note that they are gzipped) for more details about each column.

Just like in the 6th part of Module 4 (4.6. Database Access), our aim is to use *pandas* to come up with results equivalent to those that correspond to example SQL queries.

Create a single Jupyter/IPython notebook (see the *Artefacts* section below for all the requirements), where you perform what follows.

1. Establish a connection with a new SQLite database on your disk.
2. Export all the CSV files to the said database.

3. For each of the SQL queries below (each query in a separate section), write the code that yields equivalent results using *pandas* only and explain – in your own words – what it does.

```python
task1_sql = pd.read_sql_query("""
    ...an SQL statement...
""", conn)
task1_my = (
    ...your solution using pandas... – without SQL
)
pd.testing.assert_frame_equal(task1_sql, task1_my)  # we expect no error here
```

Important. Sometimes, the results generated by *pandas* will be the same up to the reordering of rows. In such a case, before calling `assert_frame_equal`, we should `sort_values` on both data frames to sort them with respect to 1 or 2 chosen columns.

Here are the SQL queries:

1.
   ```sql
   SELECT DISTINCT engine FROM planes
   ```

2.
   ```sql
   SELECT DISTINCT type, engine FROM planes
   ```

3.
   ```sql
   SELECT COUNT(*), engine FROM planes GROUP BY engine
   ```

4.
   ```sql
   SELECT COUNT(*), engine, type FROM planes
   GROUP BY engine, type
   ```

5.
   ```sql
   SELECT MIN(year), AVG(year), MAX(year), engine, manufacturer
   FROM planes
   GROUP BY engine, manufacturer
   ```

6.
   ```sql
   SELECT * FROM planes WHERE speed IS NOT NULL
   ```

7.
   ```sql
   SELECT tailnum FROM planes
   WHERE seats BETWEEN 150 AND 190 AND year >= 2012
   ```

8.
   ```sql
   SELECT tailnum, manufacturer, seats FROM planes
   WHERE manufacturer IN ("BOEING", "AIRBUS", "EMBRAER") AND seats>390
   ```

9.
   ```sql
   SELECT DISTINCT year, seats  FROM planes
   WHERE year >= 2012 ORDER BY year ASC, seats DESC
   ```

10.
    ```sql
    SELECT DISTINCT year, seats  FROM planes
    WHERE year >= 2012 ORDER BY seats DESC, year ASC
    ```

11.
    ```sql
    SELECT manufacturer, COUNT(*) FROM planes
    WHERE seats > 200 GROUP BY manufacturer
    ```

12.
    ```sql
    SELECT manufacturer, COUNT(*) FROM planes
    GROUP BY manufacturer HAVING COUNT(*) > 10
    ```

13.
    ```sql
    SELECT manufacturer, COUNT(*) FROM planes
    ```

```
WHERE seats > 200 GROUP BY manufacturer HAVING COUNT(*) > 10
```

14.
    ```
    SELECT manufacturer, COUNT(*) AS howmany
    FROM planes
    GROUP BY manufacturer
    ORDER BY howmany DESC LIMIT 5
    ```

15.
    ```
    SELECT
        flights.*,
        planes.year  AS plane_year,
        planes.speed AS plane_speed,
        planes.seats AS plane_seats
    FROM flights LEFT JOIN planes ON flights.tailnum=planes.tailnum
    ```

16.
    ```
    SELECT planes.*, airlines.* FROM
    (SELECT DISTINCT carrier, tailnum FROM flights) AS cartail
    INNER JOIN planes ON cartail.tailnum=planes.tailnum
    INNER JOIN airlines ON cartail.carrier=airlines.carrier
    ```

Do not include full outputs of the SQL queries in the report!

2 Additional Tasks for Postgraduate (SIT731) Students (*)

Postgraduate students, apart from the above tasks, are additionally **required** to solve/address/discuss what follows.

17. An additional SQL query to implement:

    ```
    SELECT
        flights2.*,
        atemp,
        ahumid
    FROM (
        SELECT * FROM flights WHERE origin='EWR'
    ) AS flights2
    LEFT JOIN (
        SELECT
            year, month, day,
            AVG(temp) AS atemp,
            AVG(humid) AS ahumid
        FROM weather
        WHERE origin='EWR'
        GROUP BY year, month, day
    ) AS weather2
    ON flights2.year=weather2.year
    AND flights2.month=weather2.month
    AND flights2.day=weather2.day
    ```

3 Optional Features (**)

The following suggestions are not part of the requirements for a pass grade, therefore you can skip them. Nevertheless, you might still want to tackle them, as only practice makes perfect.

1. Use the `timeit` module to compare the run-times of SQLite3 (through *pandas*) vs pure *pandas* solutions.
2. If there is one more way to implement a given operation, do not hesitate to provide the alternative.

4 Artefacts

The solution to the task must be included in a single Jupyter/IPython notebook (an .ipynb file) running against a Python 3 kernel.

Make sure that your notebook has a **readable structure**; in particular, that it is divided into sections. Use rich Markdown formatting (text in dedicated Markdown chunks – not just Python comments).

Imagine it is a report that you would like to show to your manager or clients — you certainly want to make a good impression. Check your spelling and grammar. Also, use formal language.

At the start of the notebook, you need to provide: the **title** of the report (e.g., *Task 42: How Much I Love This Unit*), your **name**, **student number**, **email address**, and whether you are an **undergraduate (SIT220) or postgraduate (SIT731)** student.

Then, add 1-2 introductory paragraphs (an introduction/abstract – what the task is about).

Before each nontrivial code chunk, briefly **explain** what its purpose is. After each code chunk, **summarise and discuss the obtained results** (in a few sentences).

Conclude the report with 1-2 paragraphs (summary/discussion/possible extensions of the analysis/etc.).

Submit one file via OnTrack:

1. the version of the Jupyter/IPython notebook converted to a PDF file (e.g., via *File → Export Notebook As → PDF* or convert to HTML and from that to PDF with your web browser; any method will do).

You do not need to submit the .ipynb file via OnTrack, but you must store it for further reference – a marking tutor might ask for it later, e.g., at the end of the trimester.

5 Intended Learning Outcomes

ULO	Is Related?
ULO1 (Data Processing/Wrangling)	YES
ULO2 (Data Discovery/Extraction)	YES
ULO3 (Requirement Analysis/Data Sources)	YES
ULO4 (Exploratory Data Analysis)	
ULO5 (Data Privacy and Ethics)	