For both tasks, I utilized Google Colab for my testing purposes. It gave me the ability to utilize any python libraries I might need to test for the dataset provided. It also allowed me to develop using clean cell views and receiving my test results back in a structured manner.

I uploaded the given dataset into my colab directory, since it was a publicly sourced dataset, from the Bureau of Transportation Statistics.

Task 1:

1. I first read in the excel file from my drive folder, skipping 7 rows, since the file contained metadata before the Headers
2. For the next step I imported sqlite3 to mimic using a actual Postgres database, as was specified in the pdf.
3. I then configured a connection that would serve as an “in-memory” database, and inserted the file into a table called my\_table

I then took the following steps to verify I was retrieving the correct data:

1. Created a common table expression (CTE) called RankedFlights
2. Selected the flight details:
   * flightkey, flightnum, flight\_dt, orig\_arpt, dest\_arpt, flightstatus, lastupdt.
3. Ranked flights using ROW\_NUMBER()**:**
   * Partitioned by flight date, flight number, origin, and destination.
   * Ordered by last update (lastupdt) descending, so the most recent update comes first.
   * Used flightkey ASC as a tie-breaker for deterministic results.
   * Assigned a row number (rn) to each flight within its partition.

I then performed a final SELECT: Retrieving all columns from RankedFlights.

1. Filtering to latest flight status: WHERE rn = 1 keeping only the most recently updated row per flight (per date, number, origin, destination).

Task 2:

1. For this part of the task, I first created a function header called get\_latest\_flight\_status, which takes in a filepath (and assumes its an excel file as was provided in email).
2. The function then does a pandas read\_excel function, with a header=None parameter.
3. This is done so that the header can be determined, minding the metadata rows presented.
4. The function looks for the first row contain ing the word ‘flightkey’, as it is the unique identifier for the entries.
5. Then the row is assigned to be the header row, and column names are assigned by parsing it.
6. The metadata information is then dropped, for the purposes of this function
7. Carrier code column is dropped as to reduce memory usage, since data only contains “DL” entry, and it is not needed for output
8. Combining the flight date with lastupdt to create unique combinations and ensure same date being used for the flightkey. Being done for accurate timestamp sorting
9. Sorting so that the most recent update is being kept first
10. Groups by flightkey and keeps the first row (most recent update) in each group.
11. Returns a DataFrame with one row per flightkey, representing the most recent flight status

The python function was then tested using a standard main method to test for functionality, calling the function name, with the filepath inside it.

Following these 2 tasks, I went into the codes and added thorough in-line commenting to make any newcomers to my code be able to read and understand the entire process.

**Reflecting:**

If I had more time to work on this function, I’d approach it a bit differently. I’d make the header detection smarter so it could handle typos or small variations in column names instead of just looking for “flightkey.”

I’d also spend more time adding checks for missing or messy data, so that errors wouldn’t just be silently ignored. Instead of automatically dropping the carrier\_code column, I’d make it optional, in case that information becomes useful later.

For flights with multiple updates, I’d try to combine the data in a smarter way rather than just taking the first row, so no important details get lost.

I’d also think about performance improvements for really large Excel files, like reading only the necessary columns or processing the data in chunks.

Finally, I’d break the function into smaller, more manageable pieces and add tests to make sure everything works reliably. Overall, these changes would make the function more flexible, robust, and easier to maintain.