THE UNIVERSITY OF SYDNEY

School of Computer Science
Uwe Roehm and Alan Fekete

DATA3404: Scalable Data Management **1.Sem./2021**

Big Data Analysis Assignment

Group Assignment (15%) **12.05.2021**

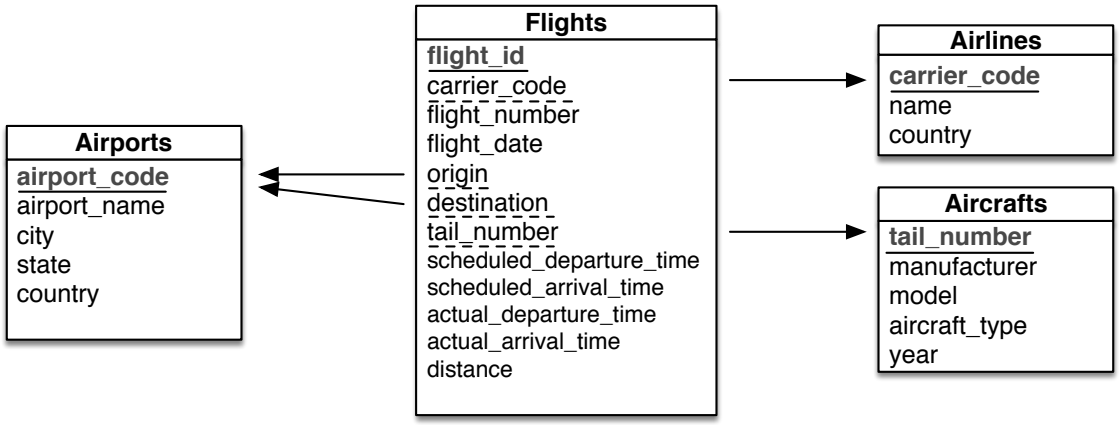
Introduction

This is the practical assignment of DATA3404 in which you have to write a series of Apache Spark programs to analyze an *air traffic* data set and then optimise your programs for scalability on increasing data volumes. We provide you with the schema and dataset. Your task is to implement the three given data analysis tasks, to evaluate their performance, and to decide on which optimisations are best suited to improve the task's performance.

You find links to online documentation, data, and hints on tools and schema needed for this assignment in the 'Assignments' section in Canvas.

Data Set Description and Preparation

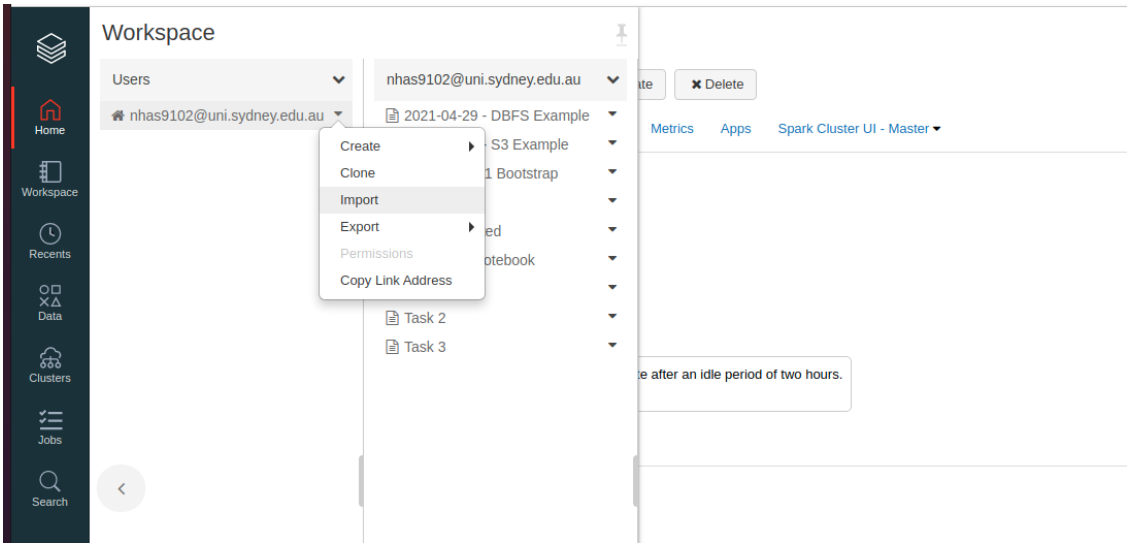
This assignment is based on an Aviation On-time data set which includes information about *airports*, *airlines*, *aircrafts*, and *flights*. This data set has the following structure:



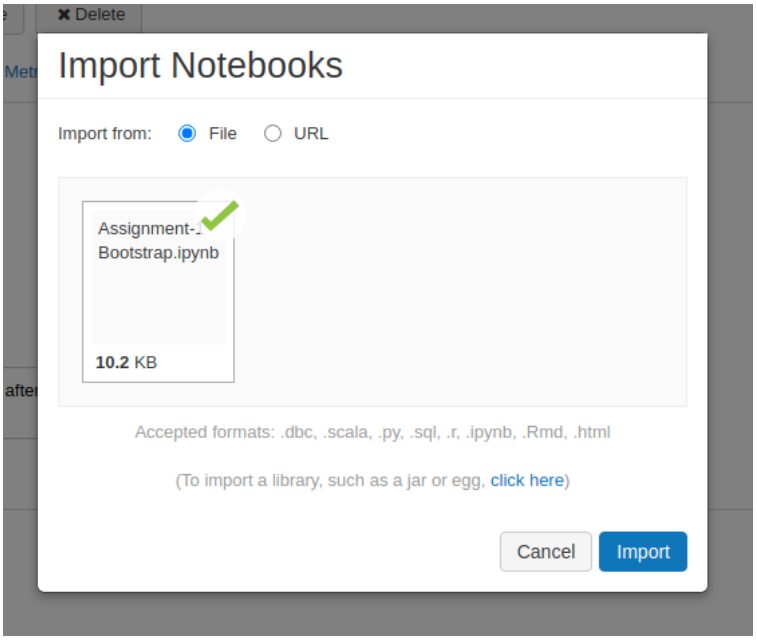
You will be provided with scaffold Jupyter notebook files for setting up your Databricks workspace. Follow the steps provided in the Power Point document to set up your Databricks Community Edition subscription in order to access your workspace.

1. Download the linked scaffold archives from the course website and unpack them.
2. In your Databricks console, press the Clusters tab on the left hand side of your screen. Follow the instructions in this video to create your cluster with Databricks runtime 7.6. **Note:** This video also shows a segment to run the Demo Task that is included with the scaffold.

3. Once your cluster is provisioned (including the SparkMeasure package as explained in the video), import the notebooks provided in the scaffold to your workspace. You can perform this step by clicking on the `Home` tab on the left hand side of your screen.



4. Repeat these steps for `Task 1`, `Task 2`, `Task 3` and `Demo Task`.



5. To download your datasets, open the "`Assignment Bootstrap`" notebook and run all the cells up-to the large dataset (**Tip: leave the download of the largest dataset until later as this will take some while and is only needed for the final scalability evaluation**).
You should only perform this step once as this is only meant to download the data sets required to complete your assignment. Running the entire notebook might take some time (typically 30 to 40 minutes).

6. **Optional:** It is worth inspecting the steps performed in the bootstrap notebook to understand the underlying architecture of Databricks. To understand these steps, you should familiarise yourself with the concepts of a `cluster`, `notebook` and the `Databricks File System (DBFS)`. A cluster is comprised of one or more physical machines that are provisioned in the Databricks cloud platform. A notebook is a generic Jupyter notebook with the additional caveat of having to `attach` a notebook to a running cluster. `DBFS` is a distributed storage platform similar to Hadoop File System (HDFS), which is provisioned for your account. The bootstrap notebook downloads the data sets from a public Google Drive directory, stores them in temporary storage and copies them into DBFS. This step is important as files stored in DBFS will persist as opposed to local storage, which will be immediately destroyed when you terminate your cluster. This is why this notebook should only be run once. By default, the Databricks Spark Context reads and writes from DBFS when a path without a file system format is provided to any I/O operation.

7. By default, a Databricks cluster will be terminated after two hours of inactivity. In such a case, you should **re-create your cluster**, **re-install the SparkMeasure library and its PyPI binding**, and then re-attach your existing notebooks to your cluster before you can work again.

Question 1: Data Analysis with Apache Spark

You shall implement three different analysis tasks of the given data set using plain Apache Spark (using the Apache Spark's RDD API or Dataframe API in Python):

1. **Task 1: Top-3 Cessna Models**
   Write an Apache Spark program that determines the top-3 Cessna aircraft models with regard to the number of flights, listed in descending order of number of flights. Output the Cessna models in the form "Cessna 123" as one string with only the initial 'C' capitalised and the model number having just its three digits. The output file should have the following tab-delimited format, ordered by number of flights in descending order:
   `Cessna XYZ \t numberOfDepartingFlights`
2. **Task 2: Average Departure Delay**
   In the second task, write a Apache Spark program that determines the average, min and max delay (in minutes) of flights by US airlines in a given year (user-specified year). Only consider delayed flights, i.e. a flight whose `actual_departure_time` is after its `scheduled_departure_time`, and ignore any canceled flights. The output file should have the following tab-delimited format (ordered alphabetically by `airline_name`):

   ```
   airline_name \t num_delays \t average_delay \t min_delay \t max_delay
   ```
3. **Task 3: Most Popular Aircraft Types**
   In the third task, you shall write an Apache Spark program that lists per airline of a given country (user-specified) the five most-used aircraft types (manufacturer, model). List the airlines in alphabetical order, and show the five most-used aircraft in descending order of the number of flights as a single, comma-separated string that is enclosed in '[' and ']' (indicating a list). Format the name of an aircraft type as follows: MANUFACTURER ' ' MODEL (for example, "Boeing 787" or "Airbus A350").

   The output should have the following tab-delimited format (alphabetically by `airline_name`):

   ```
   airline_name \t [aircraft_type1, aircraft_type2,  ... , aircraft_type5]
   ```

General Coding Requirements

1. You should solve this assignment with the Apache Spark version 2.4 as installed in Databricks. You will need a free Databricks account for this.
2. If you use any code fragments or code cliches from third-party sources (which you should not need for these tasks...), you must reference those properly. Include a statement on which parts of your submission are from yourself.
3. Always test your code using a small data set before running it on any larger data set.

Question 2: Performance Evaluation and Tuning

a) Conduct a performance evaluation of your implementations for each task on varying dataset sizes. We will provide you with four different data sizes. You should execute your code on each data size and record the execution times and the sizes of the intermediate results (communication efforts).

b) Suggest some optimisations to the the analysis task implementations such that the performance of your task(s) improve. Show that it works. Alternatively, implement a task in both RDD and Dataframe API and compare the performance of your two implementations.

Question 3: Documentation of Implementation and Tuning Decisions

Write a text document (plain text or Word document or PDF file, no more than 6 pages plus optional Appendix) in which you document your implementation and your performance evaluation. Your document should contain the following:

1. **Job Design Documentation**
   In your document, describe the Apache Spark jobs you use to implement Tasks 1 to 3. For each job, briefly describe the different transformation functions. If you use any user-defined functions, classes or operators, please describe those too.
2. **Justification of any tuning decisions or optimisations**; document the changes in the execution plans and the estimated execution costs for each individual analysis tasks before and after your optimisations using the DAG Visualizations of Apache Spark.
3. **Briefly justify each tuning decision**.
4. **Performance Evaluation**: Include a chart and a table with the **average execution times of your tasks** for different data sets.

Milestones

Have the first task ready in the Week 11 tutorials for the tutors to review and to give feedback.

Deliverables and Submission Details

There are three deliverables: **source code**, **output files**, a brief **program design and performance documentation** (up to 6 pages, as of content description above), and a **demo in Week 12** via Zoom.

All deliverables are due in Week 12, no later than **8 pm, Friday 28 May 2021**. Late submission penalty: -20%/day late of the awarded marks. Marking rubric is linked in Canvas.

Please submit the source code and a soft copy of design documentation as a zip or tar file electronically in Canvas, one per each group. Name your zip archive after your group number X with the following name pattern: **data3404_assignment2021s1-group*X*.zip**

Demo: A few points of the marking scheme will be given to any submission which can be demoed successfully on our own cluster.

Students must retain electronic copies of their submitted assignment files and databases, as the unit coordinator may request to inspect these files before marking of an assignment is completed. If these assignment files are not made available to the unit coordinator when requested, the marking of this assignment may not proceed.

All the best!

Group member participation

This is a group assignment. The mark awarded for your assignment is conditional on you being able to explain any of your answers to your tutor or the subject coordinator if asked.

If members of your group do not contribute sufficiently you should alert your tutor as soon as possible. The tutor has the discretion to scale the group's mark for each member as follows, based on the outcome of the group's demo in Weeks 12 or 13:

Level of contribution	**Proportion of final grade received**
No participation.	0%
Passive member, but full understanding of the submitted work.	50%
Minor contributor to the group's submission.	75%
Major contributor to the group's submission.	100%