1. **Job Design Documentation**

For Task 3, three CSV files are accessed: *flights*, *airlines* and *aircrafts*.

From these files, the following initial data sets are created:

Airlines: A set of tuples with three fields – *carrier\_code, name,* and *country.*

Flights: A set of tuples with two fields – *carrier\_code* and *tail\_number.*

Aircraft: A set of tuples with three fields – *tailnum, manufacturer* and *model.*

The next step is to reduce the *airlines* data set into one which contains only US airlines. This is achieved using a custom class named *USAirlineReducer*, which implements Flink’s *GroupReduceFunction* interface. Within this class there is the *reduce* function, which takes an iterable set of Tuples (in this case, the *airlines* data set), iterates over each Tuple and adds each Tuple containing a US airline to the new data set which is output as *usAirline.*

The *usAirline* data set is then joined (using Flink’s *join* transformation) to the *flights* data set using their *carrier\_code* fields as the join condition. A new data set is then created by the class *JoinALF*, which implements Flink’s *JoinFunction* interface, taking both data sets and outputting the joined data set *airlineTailNumbers*, which contains Tuples with the fields *name* from the *usAirline* data set and *tail\_number* from the *flights* data set.

After this, the newly created *airlineTailNumbers* data set is joined to the *aircraft* data set, using the same approach as the previous step. The join condition in this case is the *tail\_number* field. The joined data set is input into the class *JoinALC* which is again similar to the previous step in that it creates a new data set *aircraftDetails*, this time containing Tuples with the fields *name* and *tail\_number* from the *airlineTailNumbers* data set, as well as *manufacturer* and *model* from the *aircraft* data set.

The data set *aircraftDetails* then undergoes the following operations:

1. The Tuples are grouped by *tail­­\_number*,
2. The groups are reduced using Flink’s *reduceGroup* function, along with a custom class *AircraftCounter*, which implements the *GroupReduceFunction* interface. This class contains an overriding *reduce* function, in which each Tuple is iterated over and a counter is created for each unique tail number. This counter is then included in a new data set containing Tuples with the same four fields found in *aircraftDetails* as well as an additional field containing the count for each tail number. The decision was made to exclude null tail numbers.
3. The resultant data set is sorted alphabetically by airline name (ascending) using Flink’s *sortPartition* function,
4. The data set is then sorted in descending order by tail number count, again using *sortPartition*.

The result of these operations is a data set *countResult*, which is sorted by airline, and for each airline sorted from most used to least used tail number.

The *countResult* data set is then reduced using Flink’s *reduceGroup* function along with a custom class *AirlineReducer*. This class implements Flink’s *GroupReduceFunction* interface and iterates over *countResult,* first creating a Tuple containing a string field for the airline name, and a new ArrayList. It then adds to the ArrayList the *manufacturer* and *model* fields of the first five Tuples (five most used tail numbers) from the given airline. Once five Tuples have been added, the Tuple containing the airline name and the ArrayList is added to the output data set and a new Tuple is created for the next airline, continuing until all the airlines have been iterated over. The final data set is output as *reduceResult.*

Finally, *reduceResult* is converted to a List and input into the custom function *saveResultsToFile*. This function uses methods from the java.io and FileUtils libraries to create an output file and write to it the data from *reduceResult*, formatting it according to the assignment brief.