# Project 1: Data Warehousing (Descriptive analytics)

September 2025

## 1 Statement

Given the ACME-Flying Use Case, the purpose of this project is to design a Data Warehouse (DW) that allows to perform descriptive analysis (i.e., OLAP). The project consists of three parts:

1. Create a multidimensional database design (i.e., multidimensional schema) that complies with the requirements (and allows the correct execution of the queries) defined in the statement (see Section 1.1). In this step, you can use any graphical editor to draw the multidimensional schema (i.e., a conceptual/class diagram).
2. Develop an ETL process that allows to extract, transform and load the data from existing databases (i.e., AMOS and AIMS) to the database developed in the first step (see Section 1.2). Here you will use the "pygrametl" Python library.[1]
3. Compare the execution time of the queries over the data sources and over the multidimensional schemas in the data warehouse implemented in DuckDB[2] (see Section 1.3).

### 1.1 Multidimensional Design

You are asked to create a multidimensional schema, potentially consisting of different stars/snowflakes, that allows to easily retrieve the KPIs about aircraft utilization (namely `FH, TO, ADIS, ADOS, ADOSS, ADOSU, DU, DC, DYR, CNR, TDR` and `ADD`) as well as logbook reporting (namely RRh, RRc, PRRh, PRRc, MRRh and MRRc). All these metrics should be obtained from the available data in the databases of `AIMS` and `AMOS`. These data may need to be complemented with the following sources:

- A file containing for every aircraft registration code, its manufacturer registration code, the aircraft model and manufacturer.
- Another file containing the maintenance personnel, their identifier and the code of the airport where they work (any other employee not in this file is assumed to be a pilot).

Finest temporal granule for `FH` and `TO` is the day, while the `ADIS` and `ADOS` are calculated per month or year, like the `DYR, CNR, TDR, ADD`. Thus, required queries are:

a) Give me `FH` and `TO` per aircraft (also per model and manufacturer) per day (also per month and per year).

b) Give me `ADIS, ADOS, ADOSS, ADOSU, DYR, CNR, TDR, ADD` per aircraft (also per model and manufacturer) per month (also per year).

[1] `https://pygrametl.org`

[2] `https://duckdb.org`

c) Give me the RRh, RRc, PRRh, PRRc, MRRh and MRRc per aircraft (also per model and manufacturer) per month (also per year).

d) Give me the MRRh and MRRc per airport of the reporting person per aircraft (also per model and manufacturer).

#### 1.1.1 Deliverables

1. "dw.py" containing the tables creation and pygrametl multidimensional schema declarations.
2. PDF file (one single A4 page, 2.5cm margins, font size 12, inline space 1.15) containing:
    - The conceptual design of the multidimensional schema (i.e., boxes representing facts and dimensions together with their hierarchies and relationships).
    - All the assumptions and justifications of the decisions made (if any).

### 1.2 Extract-Transform-Load (ETL) Process Design

You are asked to create an Extract-Transform-Load (ETL) process that can be executed in order to **extract** data from the `AIMS` and `AMOS` operational databases and additionally provided data sources, **transform** these data to conform to the multidimensional schema previously defined (see Section 1.2), and **load** the data into that schema.

The designed ETL process should adhere to the following instructions:

#### Extraction

- Connect to the source databases `AIMS` and `AMOS` for extracting the raw operational data.
- You will also need to use additional data sources (i.e., `aircraft-manufaturerinfo-lookup.csv` and `maintenance_personnel.csv`) and thus will need to extract their data.

#### Transformation

- Integrate data coming from `AIMS` and `AMOS` data sources. You should consider integrating these two sources having in mind the common attributes that they share.
- Complement the operational data coming from `AIMS` and `AMOS` by means of performing a lookup to the external data sources about:
    - **Aircraft manufacturer information** (`aircraft-manufacturerinfo-lookup.csv`) such that with each aircraft registration code, your ETL also provides its manufacturer registration code, the aircraft model and manufacturer.
    - **Maintenance personnel employment place** (`maintenance_personnel.csv`) such that for each person from the maintenance personnel (i.e., reporteurID from table `PostFlightReports`), your ETL also provides information at which airport this person works.
- Derive additional attributes, by means of, but not limited to value conversion and formula calculation, in order to enable the calculation of the requested KPIs . For example, to calculate `Flight Hours (FH)` you should subtract `actualDeparture` from `actualArrival` times, and for `Flight cycles (TO)` you should count only the *non-cancelled flights* in table `Flights`.

- Improve the quality of the sources data by checking and fixing only the following three business rules (those attached as an appendix to this document are merely informative to facilitate the interpretation of the attributes and query results):

    1. In a Flight, actualArrival is posterior to actualDeparture, related to BR-23 (Fix: Swap their values)
    2. Two non-cancelled Flights of the same aircraft cannot overlap, related to BR-21 (Fix: Ignore the first flight, but record the row in a log file)
    3. The aircraft registration in a post flight report must be an aircraft (Fix: Ignore the report, but record the row in a log file)

**Loading**

- Create the DW in a DuckDB database.
- Load dimension tables of your multidimensional schema, paying special attention to keep the different aggregation levels.
    - For example, `aircraft dimension` table with information about the corresponding `aircraft model`.
- Load fact tables of your multidimensional schema, allowing the calculation of all the metrics needed to generate the required KPIs.

### 1.2.1 Deliverables

1. "extract.py", "transform.py" and "load.py" containing the implementation of the different ETL tasks.
    - Indicate clearly where each business rule is checked/acted upon (i.e., put a comment in the Python code).
2. "etl_control_flow" containing the control flow of the ETL.
3. PDF file (one single A4 page, 2.5cm margins, font size 11, inline space 1.15) with:
    - The conceptual design of the ETL (just boxes and arrows).
    - All the assumptions and justifications of the decisions you made (if any).

## 1.3 Comparison of the queries over the Data Warehouse and the sources

Check the given queries over the raw data in PostgreSQL and reimplement them over the DW in DuckDB using "query_test.py". Without any cleaning, the results should be exactly the same.

### 1.3.1 Deliverables

1. "dw.py" containing the three queries over the multidimensional schema in the corresponding place.
2. PDF file (one single A4 page, 2.5cm margins, font size 11, inline space 1.15), containing:
    - The assumptions and justifications of the decisions you made (if any).

## 2 Assessment criteria

a) Python code

  i) Correctness of the code (i.e., the project needs to be executable and without errors, without any changes by the lecturer)

  ii) Understandability (i.e., variable names need to be readable and complex lines/blocks need to be commented)

  iii) Efficiency (i.e., execution time)

  iv) Not hardcoding data

b) PDF files

  i) Conciseness of explanations

  ii) Coherence

  iii) Soundness

## 3 Evaluation

- 60% Deliverables (i.e., Python code and PDF file)
- 40% Exercises related to the project done individually in the classroom the day of the partial exam

## A Appendix

## Business Rules

Below you can find the business rules that one would expect to be true in the data. Nevertheless, neither the processes nor the DBMS enforced them. Thus, they may have been violated, giving rise to quality problems.

### `AMOS` database

#### Identifiers

*BR-1* `WorkPackageID` is an identifier of `WorkPackage`.

*BR-2* `workOrderID` is an identifier of `WorkOrders/ForecastedOrders/TechnicalLogBookOrders`.

*BR-3* `maintenanceID` is an identifier of `MaintenanceEvents/OperationInterruption`.

*BR-4* `file` is an identifier of `Attachments`.

#### References

*BR-5* `event` of an `Attachement` is a reference to `maintenanceID` of `MaintenanceEvents`.

#### Datatypes/Domains

*BR-6* `subsystem` of `MaintenanceEvents` should be a 4 digits ATA code[3]

*BR-7* `delayCode` in `OperationInterruption` should be a 2 digits IATA code[4]

*BR-8* `ReportKind` values “PIREP” and “MAREP” refer to pilot and maintenance personnel as reporters, respectively.

*BR-9* `MELCathegory` values `A,B,C,D` refer to `3,10,30,120` days of allowed delay in the repairing of the problem in the aircraft, respectively.

*BR-10* `airport` in `MaintenanceEvents` must have a value.

#### Other business rules

*BR-11* In `OperationInterruption`, departure must coincide with the date of the `FlightID` (see below how it is composed).

*BR-12* The `Flight` registered in `OperationInterruption`, must exist in the `Flights` of `AIMS` database, and be marked as “delayed” (i.e., `delayCode` is not null) with the same IATA delay code.

*BR-13* In `MaintenanceEvents`, the events of kind `Maintenance` that correspond to a `Revision`, are those of the same aircraft whose interval is completely included in that of the `Revision`. For all of them, the airport must be the same.

- In `MaintenanceEvents`, the events of kind `Maintenance` cannot partially intersect that of a `Revision` of the same aircraft.

*BR-14* In `MaintenanceEvents`, maintenance duration must have the expected length according to the kind of maintenance (Delay – minutes, Safety – undetermined/unlimited, `AircraftOnGround` - hours, `Maintenance` – hours to max 1 day, `Revision` – days to 1 month).

[3] ATA codes for commercial aircrafts: https://en.wikipedia.org/wiki/ATA_100

[4] IATA delay codes: https://en.wikipedia.org/wiki/IATA_delay_codes

## `AIMS` database

### Identifiers

*BR-15* `FlightID` is an identifier of `Flights`.

### Datatypes/Domains

*BR-16* `FlightID` is derived by concatenating the following values:
`Date-Origin-Destination-FlightNumber-AircraftRegistration`
(lengths: 6+1+3+1+3+1+4+1+6=26).

*BR-17* `delayCode` in `OperationInterruption` is a 2 digits IATA code[5]

### Other business rules

*BR-18* In a `Slot`, `scheduledArrival` must be posterior to the `scheduledDeparture`.

*BR-19* A `Flight` is not longer than 24 hours.

*BR-20* All the hours of a `Flight` are imputed to the date of its `scheduledDeparture`.

*BR-21* Two `Slots` of the same aircraft cannot overlap.

*BR-22* In `Flights`, departure and arrival airports must be those in the `FlightID` (unless this `Flight` has been diverted).

*BR-23* In a `Flight`, `actualArrival` is posterior to `actualDeparture`.

*BR-24* In a `Maintenance`, the corresponding events must exist in `AMOS` inside the corresponding time interval.

[5] IATA delay codes: https://en.wikipedia.org/wiki/IATA_delay_codes