Univ.-Prof. Dr.-Ing. Matthias Boehm
Technische Universität Berlin
Faculty IV - Electrical Engineering and Computer Science
Berlin Institute for the Foundations of Learning and Data (BIFOLD)
Big Data Engineering (DAMS Lab) Group

DIA WiSe2025: Exercise – Berlin Public Transport Data Analysis

Published: Oct 13, 2025
Deadline: Jan 30, 2025; 11.59pm

This exercise is an alternative to the DIA programming projects and provides practical experience in the development of ETL pipelines for data integration and analytics. The task of this semester is to ingest and analyze data from Berlin's public transport system, extracted via the Deutsche Bahn (DB) API marketplace[1]. We collected real-world information for 133 Berlin stations, including train connections and disruptions, from Sep 02, 2025 through Oct 15, 2025. The objectives are to design a schema capable of accommodating this data and to implement queries that demonstrate proficiency in data integration and large-scale analysis. The final submission is a zip archive named `DIA_Exercise_<student ID>.zip` (max 5MB), containing: (1) the source code used to solve the individual sub-tasks, as well as (2) a PDF report of up to 8 pages (10pt), including the names of all team members, a summary of how to run the code, and a description of the solutions to the individual sub-tasks.

Data Source: The dataset consists of three main components:

- `Stations`: A `.json` file with metadata for the 133 Berlin stations.
- `Timetables`: `.xml` files containing planned train movements (arrival/departure times, platforms, lines, numbers, routes), collected once per hour at `HH:01`.
- `Timetable Changes`: `.xml` files containing disruptions (delays, cancellations, modification messages), collected every 15 minutes at `HH:01`, `HH:16`, `HH:31`, and `HH:46`.

We also provide a `schema.json` file describing all fields. All timetable files are stored as: `/timetables/{date_hour_00}/{station_name}_timetable.xml`, where `date_hour_00` encodes the download time, e.g. `2509051100` for Sep 05, 2025 at 11:00. Each file covers one hour of data. Furthermore, all timetable change files are stored as: `/timetable_changes/{date_hour_minute}/{station_name}_change.xml`, where `date_hour_minute` encodes the download time, e.g. `2509051116` for Sep 05, 2025 at 11:16. Each file covers 15 minutes of data.

Grading: This exercise can be pursued in teams of 1 to 3 persons (one submission, scaled quality expectations). The overall grading is a pass/fail for the entire team. Exercises with $\geq$ 50/100 points are a pass, and with $\geq$ 90/100 points receive 5 extra points for each team member in the exam.

1 Schema Design and Data Ingestion (25/100 points)

Design a star schema to represent station metadata, timetables, and timetable changes, and implement a pipeline that loads the provided data into a relational database.

- **Task 1.1 (10 points):** Design a star schema with one fact table for train movements (planned arrivals, departures, delays, cancellations) and suitable dimension tables (e.g., stations, trains, time). Clearly specify all primary and foreign keys.

[1] https://developers.deutschebahn.com/db-api-marketplace/apis/

- **Task 1.2 (15 points):** Implement a data ingestion pipeline that parses the `.json` and `.xml` files and loads the data into your schema in PostgreSQL. Ensure consistent mapping of station identifiers and correct interpretation of timestamps from the folder structure.

Expected Results: A SQL script (with comments) for creating your star schema (tables, keys, relationships) and a working ETL pipeline that ingests the provided data into PostgreSQL.

2 Data Analysis (25/100 points)

Implement SQL queries on the ingested dataset. The queries should be written in standard SQL, be executable in PostgreSQL, and produce correct expected results.

- **Task 2.1 (5 points):** Given a station name, return its coordinates and identifier.
- **Task 2.2 (5 points):** Given latitude/longitude, return the name of the closest station.
- **Task 2.3 (5 points):** Given a time snapshot (`date_hour`), return the total number of canceled trains over all 133 stations in Berlin.
- **Task 2.4 (10 points):** Given a station name, return the average train delay in that station.

Expected Results: A collection of SQL queries (one per task), accompanied by brief explanations.

3 Large-Scale ETL and Analysis in Spark (35/100 points)

Implement an end-to-end ETL pipeline in **Apache Spark** over the provided `.xml` files. The goal is to produce a Parquet dataset partitioned by time snapshots and to use it for analytical queries.

- **Task 3.1 (15 points):** Implement a Spark ETL job that parses the timetable and change files, extracts the relevant fields, and stores the result as a Parquet dataset.
- **Task 3.2 (10 points):** Using the Parquet dataset, write a Spark query that, for a given station, computes the average daily delay over the period that the data was collected.
- **Task 3.3 (10 points):** Return the average number of train departures per station during peak hours (07:00 to 09:00 and 17:00 to 19:00).

Expected Results: A description of your ETL pipeline and the resulting schema, together with the Spark code that runs the required query.

4 Graph-Based Analytics (20/100 points)

Finally, work with the train network as a graph and implement routing algorithms of increasing complexity.

- **Task 4.1 (10 points):** Model the stations as nodes and direct train connections as edges. Implement an algorithm that, given two stations, computes the shortest path between them in terms of the number of transfers (graph hops), without considering schedules or delays.
- **Task 4.2 (10 points):** Extend the approach by taking timetable information into account. Implement the *earliest arrival time* algorithm: given a source station, a target station, and a departure time, compute the earliest possible arrival time considering departures, arrivals, and transfers. You may assume all transfers are possible whenever arrival and departure times at a station are consistent.

Expected Results: Code implementing both algorithms and a short explanation of the logic and data structures used.