**Data Engineering Position – Coding Challenge**

Documentation of the general approach for this exercise:

1. Get Started:

* Open the raw csv file with Unicode UTF-8
* Identify the correct separator and convert text in columns.
* Save it locally as a .CSV file

1. Get an overview of the file:

* Use of Pandas library in Jupyter Notebook
* In total:
  + Rows: 35613
  + Columns: 38
  + 11 columns with missing values (NaN)

1. Data Cleaning:

* Get rid of NaNs, negative Values, inconsistent values
* Script for data cleaning was created:
  + Datacleaning.py

1. Data Modeling:

* First open draw.io
  + Look for ‘\_id’ in the .CSV file 🡪 indicator for a table
  + Think of possible relations
  + Normalize tables to avoid redundancy
* Design a star schema
* Design a snowflake schema with more normalized tables
* Export the Data Models as a .PDF file
* Export the Data Models as a .XML file (needed for the script later)

1. Data Modeling Documentation

* In a separate file ‘Data Modeling Documentation.docx’ the following topics were documented for both schemas:
  + Facts Table (PKs, FKs, Relationship)
  + Dimension Tables (PKs, FKs, Relationship, Normalization, Historicization)
  + Performance Implications (Pros and Cons)

1. Data Dictionary

* Based on the data model a data dictionary was created for the Facts and Dimension Tables and its attributes
* Python script ‘xml2datadict\_starschema.py’ / ‘xml2datadict\_snowflakeschema.py’ converts the .XML file of the respective data model into a .CSV file, which represents the data dictionary.
* This data dictionary contains the columns (Tabellenname, Spaltenname, Spaltentyp, Zeichenlaenge, PK, FK, AI, NN, and Beschreibung)
* The columns Spaltentyp, Zeichenlaenge and Beschreibung were manually entered

1. SQL Query

* Next, a SQL script was generated to create SQL querys based on the updated .CSV file (data dictionary)
* The python script ‘datadict2sqlcreate\_starschema.py’ / ‘datadict2sqlcreate\_snowflakeschema.py’ generates a .SQL file with all its required SQL queries to create tables

1. Database deployment

* Download and install DB Browser for SQLite
* Open a new database (e.g. ‘munich\_sales\_starschema’)
* Insert the SQL queries from the previous generated .SQL file for the respective schema
* Run the Query statements to see if the SQL queries work

1. Data Pipeline implementation

* Write an ETL script for both schemas
* Extract the data from .CSV file
* Use the sql scripts for the tables
* Transform the denormalized data into normalized tables
* Load the data into the database