Project - HR Analytics Proof of Concept

Purpose

The project aims to implement the modern data stack to solve a real-world problem. In addition to using a certain number of techniques, there is a focus on working together in a data team. This includes both agile development and being able to use Git and GitHub in a team.

Scenario

Imagine you are a data engineer for a HR agency. Here's an overview of the business model of this agency:

Talent acquisition specialists work with different occupation fields. According to the opening job ads on Arbetsförmedlingen, they will:

- search and contact potential candidates from LinkedIn
- contact and market those potential candidates to corresponding employers

Therefore, they constantly analyze job ads in order to understand which types of candidates they should approach. Currently, every begining of the week, they manually browse the homepage of Arbetsförmedlingen and download a list of opening job ads to guide their work over the week. However, they are not able to draw insights from these job ads as:

- the information are messy
- they have spent too much time to manually collect and clean data so that they do not have much time to analyze the data, which is important to improve the efficiency of their work

Now, you are given a task to create a data pipeline for the team of talent acquisition specialists to:

- automate the data extraction from Jobtech API of Arbetsförmedlingen
- transform and structure data according to a dimensional model
- design a dashboard for talent acquisition specialists to analyse numbers of vacancies by city, by occupation and by employment types etc, for each of the occupation fields
- etc... details of all requirements will be described in the tasks below

Mandatory tasks

Task 1 - Setup

GitHub Repository

One person sets up a public GitHub repository and invites the others. Also, set up GitHub Project to have a Kanban board where you can see which tasks/issues need to be done and who is working on which task/issue.

Remember to work in your own branch and pull from the main branch before making a pull request to main. This way, any merge conflicts are resolved in your own branch. Also, name your branch in a meaningful way, for instance, `dbt_test`, if your branch is adding `dbt test` files.

Make many commits, don't wait to make a few large commits. This way, the code is better version-controlled, you get a backup, and the codebase develops over time.

UV Virtual Environment

One person installs the necessary packages and creates a requirements file that is pushed to GitHub:
`uv pip freeze > requirements.txt`
Other people install the dependencies listed in the requirements file:
`uv pip install -r requirements.txt`

dbt

Here, one person sets up dbt, and others do a git pull. However, remember to have the same content in profiles.yml pointing to this project. Run dbt deps to install packages for those who have pulled the code.

Task 2 - Data ingestion with dlt to duckdb

Use dlt to load data from JOBTECH API. Each group will be loading data for three occupation fields specified on your group list.

This means that different groups will load data for different occupation fields.

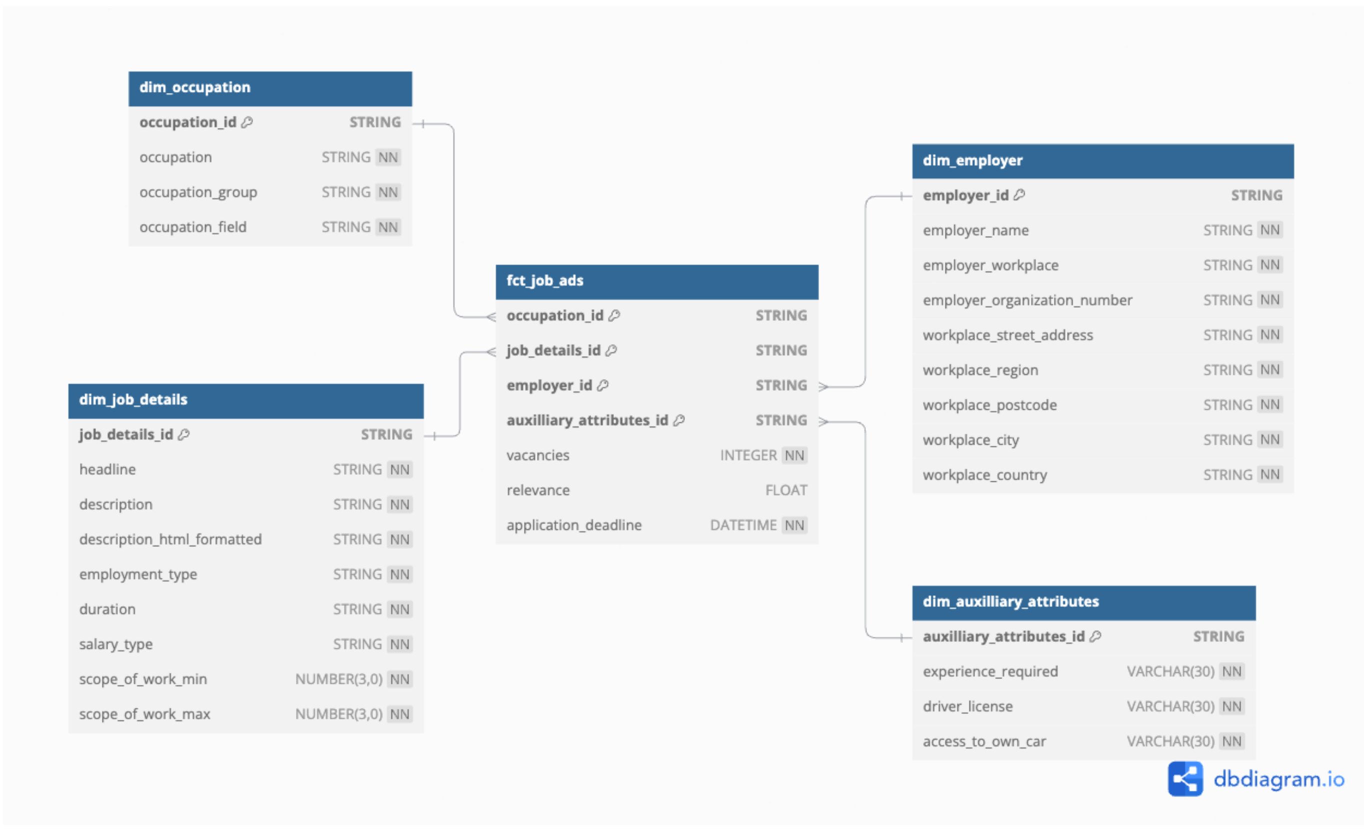
You can refer to the example python script used in lecture 6 for loading job ads by filtering occupation fields. You will be using this API url: `https://jobsearch.api.jobtechdev.se/search`
and this query parameter:
`occupation-field`

With this python script, you will be able to load more job ads for your downstream streamlit dashboard.

Task 4 - Data Transformation with dbt

Use dbt to transform data loaded to the staging schema by dlt, to data warehouse and mart schemas.

Data in the data warehouse schema should follow the structure specified in the dimensional model below:


dim_occupation
occupation_id STRING
occupation STRING NN
occupation_group STRING NN
occupation_field STRING NN
fct_job_ads
occupation_id STRING
job_details_id STRING
employer_id STRING
auxilliary_attributes_id STRING
vacancies INTEGER NN
relevance FLOAT
application_deadline DATETIME NN
dim_employer
employer_id STRING
employer_name STRING NN
employer_workplace STRING NN
employer_organization_number STRING NN
workplace_street_address STRING NN
workplace_region STRING NN
workplace_postcode STRING NN
workplace_city STRING NN
workplace_country STRING NN
dim_job_details
job_details_id STRING
headline STRING NN
description STRING NN
description_html_formatted STRING NN
employment_type STRING NN
duration STRING NN
salary_type STRING NN
scope_of_work_min NUMBER(3,0) NN
scope_of_work_max NUMBER(3,0) NN
dim_auxilliary_attributes
auxilliary_attributes_id STRING
experience_required VARCHAR(30) NN
driver_license VARCHAR(30) NN
access_to_own_car VARCHAR(30) NN
dbdiagram.io

While in the mart schema, create three views, one for each occupation field, that are data that can be consumed directly by the downstream dashboard for analyzing job ads.

Task 5 - Dashboard with streamlit/python

Create a streamlit dashboard to consume data **from the mart schema**. Design the dashboard in a way to help talent acquisition specialists working for specific occupation field your group is working on. For instance, try to produce visualizations to answer these questions:

- for a specific occupation field (i.e. Data/IT), which occupation (i.e. data engineer) has a higher number of vacanies?
- which cities has a higher number of vacancies?
- etc

You should include at least four meaningful KPI/metrics and visualizations on your dashboard that are able to improve efficiency of the work of talent acquisition specialists in this HR agency.

Bonus tasks

Task 6 - dbt test and documentation

In the data transformation part, add meaningful dbt tests that validate the quality of your dbt models. Also, create dbt documentation to help the company understand data dependency in the data warehouse.

Make use of a README.md file to document highlights of your dbt tests and documentations.

Task 7 - Apply LLM on dashboard

The HR agency would like to create a graph showing major qualities employers in a certain occupation field is looking for from job applicants. It wants to post the graph to LinkedIn and attract applicants with these qualities to contact the agency.

There are several text columns in the job ads data, for instance, `description`. Use LLM to analyze the these columns and produce analysis. Then, add a visualization using the analysis on your streamlit dashboard.

Task 8 - Orchestration with Dagster

Orchestrate different parts of your data pipeline with Dagster, then, try to run the pipeline for at least twice and check if you can find trend from the data.

Submission and Grading

There are group and individual submissions for this project. Each submission will recieve the grade of IG, G and VG. In order to get VG in this course, VG is required for both submissions. In order to get G in this course, both submissions should be at least G. Below are requirements for each submission:

Group

You'll submit your shared code base in GitHub and the project Kanban board. The commit history of the code base and the Kanban board should show what each person has done during the project. Each group will present for 10 minutes, including:

- illustrate how the dashboard assists decision making of talent acquisition specialists
- share how you worked with Git and GitHub in the team
- briefly present how data flowed from upstream to downstream

You data pipeline should include all components for the mandatory tasks, while for VG, you should also include components for the bonus tasks. Include a README.md for your repo to document your project which can guide others to understand your project. Use functions and modules to well-structure your scripts.

Individual

Two things will be accessed for individual grading:

- the commit histories of the repo should show that you have made relevant commits for the project codebase. To obtain VG, the commit histories should show that you have worked on the VG components for the data pipeline.
- write a report of 1-2 pages describing 1) reflection of team work and 2) the technical steps in the project. Write in English and use suitable technical terminology.

For the reflection of team work, you can share:

- have everyone worked on the same tasks together
- or else, how the labour division looks like?
- how your team benefit from Github collaboration features?

For technical steps, here're some example points you can include:

- how has data flowed; describe data lineage here?

- describe how you have transformed the data
- what trade-offs have you made in the dashboard?
- how does the division of the different data layers look, i.e., the different schemas you've created?
- what is the purpose behind the different data layers?
- what technology have you used and for what purposes in this project?
- feel free to describe more points, but keep the report to a maximum of 2 pages.