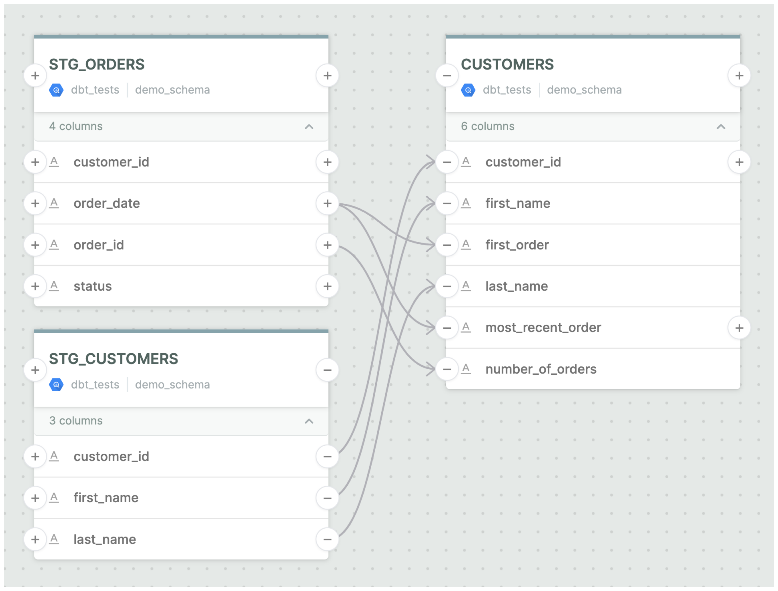
R&D Backend Exercise

Hi there! Welcome to our coding exercise. Over the next two hours, you're going to build some lineage processing features.
Lineage is the inheritance relationship between entities. In our case - the entities are columns in SQL tables, so lineage can look like this:



You'll start simple, then build from there.
Since this is an exercise, you will **not** be required to parse raw SQLs. We have some lovely code that transforms SQL queries into edges that connect source columns to destinations. You'll be working with the output of that component.

Input format

For cross-language purposes, you'll receive the inputs as a JSON file with tables and edges.

Example 1:

<u>Query</u>

```
INSERT INTO A SELECT a from B;
```

<u>JSON</u>

```
{
  "tables": [
    {
      "table_name": "A",
      "columns": [
        { "col_name": "a", "type": "varchar" }
      ]
    },
    {
      "table_name": "B",
      "columns": [
        { "col_name": "a", "type": "varchar" }
      ]
    }
  ],
  "edges": [
    {
      "src_column": {
        "table_name": "B",
        "col_name": "a"
      },
      "dst_column": {
        "table_name": "A",
        "col_name": "a"
      },
      "confidence": 1
    }
  ]
}
```

Example 2:

Query (column’s origin is uncertain)

```
INSERT INTO A SELECT a from B, C;
```

JSON

```
{
  "tables": [
    {
      "table_name": "A",
      "columns": [
        { "col_name": "a", "type": "varchar" }
      ]
    },
    {
      "table_name": "B",
      "columns": [
        { "col_name": "a", "type": "varchar" }
      ]
    },
    {
      "table_name": "C",
      "columns": [
        { "col_name": "b", "type": "varchar" }
      ]
    }
  ],
  "edges": [
    {
      "src_column": {
        "table_name": "B",
        "col_name": "a"
      },
      "dst_column": {
        "table_name": "A",
        "col_name": "a"
      },
      "confidence": 0.5
    },
    {
      "src_column": {
        "table_name": "C",
        "col_name": "a"
      },
      "dst_column": {
        "table_name": "A",
        "col_name": "a"
      },
      "confidence": 0.5
    }
  ]
}
```

Guiding Principles

- **Perfect is the enemy of good.** Trade-offs are an integral part of software engineering. Make sure you finish. You can always improve later, if you have the time.

- **Write code for people, not computers.** Meaningful variable names and function names go a long way, as does a clean, legible style of coding.
- **Ensure correctness.** Your code should be robust and work in various scenarios.

Guidelines

Hand in your code, along with the output for each level.
The levels are built in increasing complexity, it's ok and advisable to use the solutions to previous levels.

Tasks

To check your code, navigate to https://coding-exercise.foundational.io/ and paste your output JSON (in its original format).

Level 1: Warming up

As you can see in Example 2, sometimes you can't know for sure whether certain edges are correct.
Read the input at ***base_sample.json*** and filter any edge for which the confidence <= 0.5.

Level 2: House cleaning

During Level 1, you removed some of the edges that contained lineage information. Using the available tables' schema (under the "tables" key), you can now cross-reference the edges with the schema and find out which one is correct!
Go back to ***base_sample.json***, and this time, try to keep as many edges as possible for each group of edges leading to the same destination. If possible, single out the correct edge and remove the rest.
e.g., in Example 2, you would keep the edge from B.a -> A.a and remove the one from C.a -> A.a, because the source column C.a doesn't exist.

Level 3: Getting wild

Oh noes! Our analyst started running wildcard queries:
`INSERT INTO A (SELECT * from B)`
You'll find these in ***wildcard_sample.json**,* marked in the edge as "*" in the *col_name* field of both the *src_column* and *dst_column*. You'll need to take care of those by expanding this single edge to multiple ones, one for each column.

Level 4: Down to business

Time to take off your training wheels. In reality, you won't usually get the full schema – just the schema for the leftmost tables. But don't let that stop you! Check out ***advanced_sample.json***

and <u>deduce the schema for the other tables</u> by going over the connecting edges in the right order.
Oh, by the way – our analyst is getting pretty obsessed with the SELECT * syntax, so don’t blame us if you start seeing it everywhere.