Documentation

1. Clients Table

- **Table Name**: `clients_table`
- **Primary Key**: `client_id`
- **Attributes**:
  - `client_name`: Name of the client.
  - `country`: Geographical location of the client.

2. Suppliers Table

- **Table Name**: `suppliers_table`
- **Primary Key**: `supplier_id`
- **Attributes**:
  - `supplier_name`: Name of the supplier.
  - `country`: Country where the supplier is located.

3. Sonar Runs Table

- **Table Name**: `sonar_runs`
- **Primary Key**: `sonar_run_id`
- **Attributes**:
  - `client_id`: A foreign key that references the `client_id` in the `clients` table, establishing a relationship that associates each sonar run with a specific client.
  - `date`: The date when the sonar run was conducted.
  - `status`: Current status of the sonar run (e.g., completed, in progress).

Associative Table: Sonar Run Suppliers

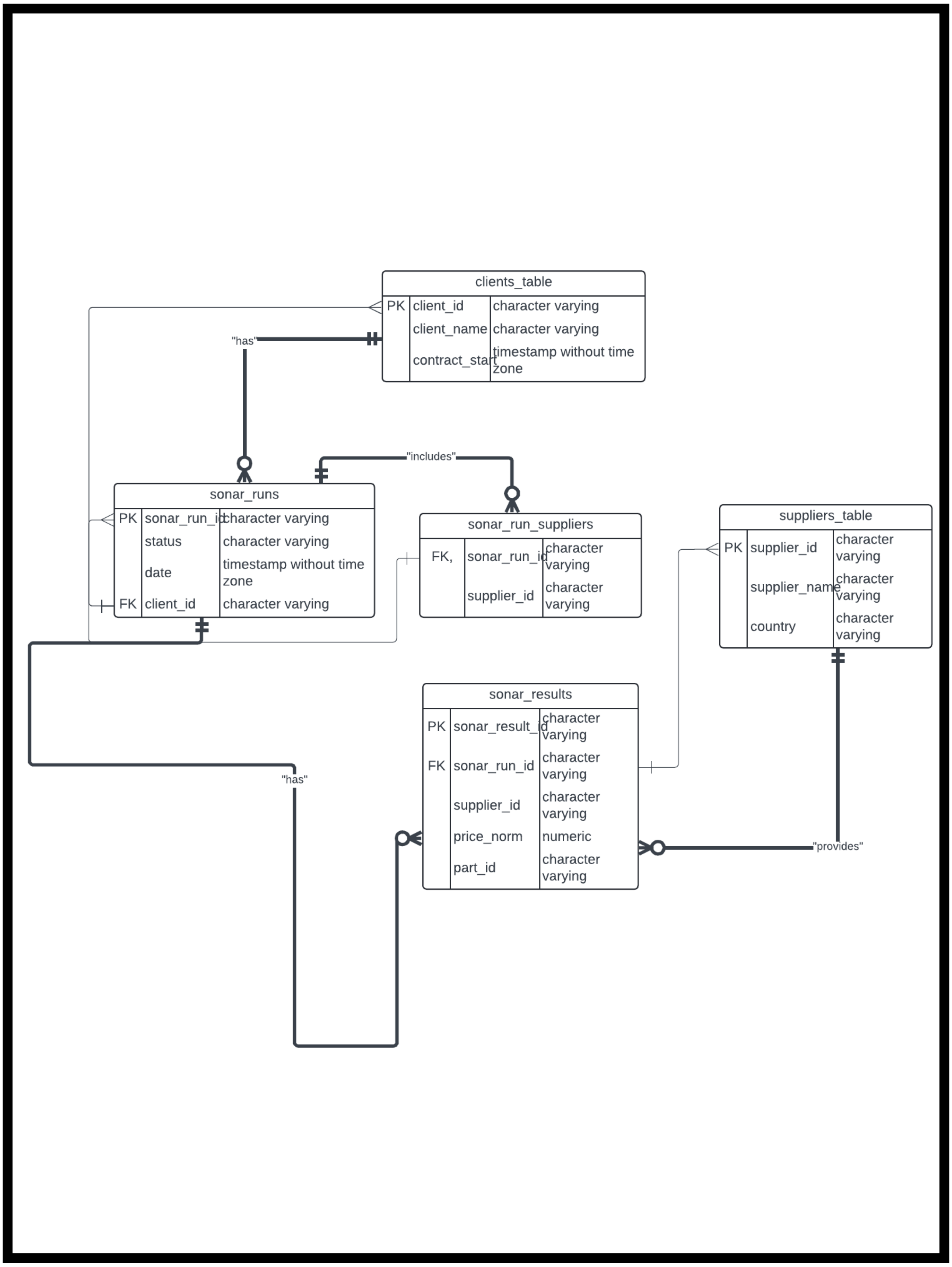
- **Table Name**: `sonar_run_suppliers`
- **Foreign Keys**:
  - `sonar_run_id`: A foreign key referencing `sonar_runs.sonar_run_id`, linking each supplier to a specific sonar run.
  - `supplier_id`: A foreign key referencing `suppliers.supplier_id`, indicating which supplier is associated with the sonar run.
- **Purpose**: This associative table is designed to manage the many-to-many relationship between sonar runs and suppliers. A single sonar run can involve multiple suppliers, and a single supplier can be associated with multiple sonar runs.

4. Sonar Results Table

- **Table Name**: `sonar_results`
- **Primary Key**: `sonar_result_id`
- **Foreign Keys**:
  - `sonar_run_id`: A foreign key referencing `sonar_runs.sonar_run_id`, which connects each sonar result to its corresponding sonar run.
  - `supplier_id`: A foreign key referencing `suppliers.supplier_id`, linking each result to the supplier that provided it.
- **Attributes**:
  - `part_number`: Identifies the specific part related to the sonar result.

- `price`: The price associated with the part in the sonar result

- Below is the ER-Diagram for the relations:


clients_table
PK client_id character varying
client_name character varying
contract_start timestamp without time zone
"has"
"includes"
sonar_runs
PK sonar_run_id character varying
status character varying
date timestamp without time zone
FK client_id character varying
sonar_run_suppliers
FK, sonar_run_id character varying
supplier_id character varying
suppliers_table
PK supplier_id character varying
supplier_name character varying
country character varying
sonar_results
PK sonar_result_id character varying
FK sonar_run_id character varying
supplier_id character varying
price_norm numeric
part_id character varying
"has"
"provides"

Relationships Summary

- `Clients and Sonar Runs`: A one-to-many relationship exists between the clients table and the sonar_runs table, where a single client can have multiple sonar runs associated with them, but each sonar run is linked to only one client.

- `Suppliers and Sonar Runs`: The sonar_run_suppliers associative table establishes a many-to-many relationship between sonar_runs and suppliers, indicating that each sonar run can have multiple suppliers involved, and each supplier can be linked to multiple sonar runs.

- `Suppliers and Sonar Results`: Each sonar result is also associated with one supplier through a foreign key, creating a one-to-many relationship where a supplier can provide multiple results across different sonar runs.

Assumption: Here I have assumed that all the suppliers from each sonar run is already included in the suppliers table.But this assumption can be changed as per the business use case. In case the business use case requires that there can be additional suppliers apart from the suppliers list, the code can be modified to first insert all the suppliers that are not in suppliers table and then have a foreign key reference from the sonar_runs_suppliers table to the supplier table.

High-Level Overview of Transformations

1. **Data Loading from Source to DataFrame**:
    - o Data from MongoDB collections (clients, suppliers, sonar runs, and sonar results) has been extracted and loaded into pandas DataFrames for processing and analysis.
2. **Data Validation and Preparation**:
    - o Validation checks are performed on data types and structures (e.g., ensuring `supplier_ids` is a list).
    - o Handling of potential null or missing values, especially in the `supplier_ids` column.

5. **Establishing Relationships**:
    - o Inserting relationships between each sonar run and its associated suppliers into the `sonar_run_suppliers` table, allowing for many-to-many relationships.
    - o Again utilizing the `ON CONFLICT` clause to manage potential duplicates.
6. **Error Handling**:
    - o Implementing try-except blocks to catch and report errors during the insertion process, ensuring the ETL process can handle exceptions gracefully.
7. **Data Commit**:
    - o Committing the transaction to the PostgreSQL database to ensure all changes are saved and persisted after successful insertions.
8. **Logging**:
    - o Printing out key information and progress updates during the ETL process, providing visibility into the operations being performed.

Scope for improvement

- The code can be further improved to process the dates like extract the month, day and year.
- We can use Python's logging library instead of print statements for better logging of events and errors. This would help in tracking the flow of the ETL process.
- For better performance, especially with larger datasets, we can execute_batch from psycopg2.extras to perform batch inserts.
- We can also organize code into separate modules or classes for better readability and maintenance. For instance, separate the extraction, transformation, and loading logic into different files or classes.

Note:

There are two .py files

- ETL_pipeline_mongo- Here the entire pipeline is taken into consideration starting from connecting to MongoDB and extracting the json files till the loading of data into Postgres. (more realistic approach for realtime)
- ETL_pipeline_mongo.py- Here we assume we have the extracted the MongoDB collections and we simply use the collection to build the pipeline