**ETL Pipeline for Cancer Clinical Trials from clinicaltrials.gov**

**Objective:**

The project aims to create a comprehensive database of cancer-related clinical trials by developing an **ETL pipeline** that extracts data from the **ClinicalTrials.gov API**, transforms it into a structured format, and loads it into a relational database. The objective is to enable complex queries on trials, conditions, interventions, and sponsors, supporting efficient data analysis and decision-making in clinical research.

**Database Schema:**

To effectively organize the extracted data, the following tables are created to represent key entities and their relationships in the database:

* **Trials:** Basic trial information.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| trial\_id | INT | Primary Key | Unique identifier for each trial. |
| nct\_id | VARCHAR (50) | NOT NULL | The NCTXXXXXXXX identifier for the trial |
| title | VARCHAR (255) | NOT NULL | The title of the clinical trial. |
| status | VARCHAR (50) |  | The status of the trial (e.g., Active, Completed). |
| phase | VARCHAR (50) |  | The phase of the trial (e.g., Phase 1, Phase 2). |
| sponsor\_id | INT | Foreign Key (references sponsors.sponsor\_id) | The sponsor of the trial (e.g., pharmaceutical company). |

* **Conditions:** List of cancer conditions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| condition\_id | INT | Primary Key | Unique identifier for each condition. |
| name | VARCHAR (255) | UNIQUE, NOT NULL | Name of the medical condition (e.g., "Lung Cancer"). |

* **Interventions:** List of interventions (drugs, therapies).

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| intervention\_id | INT | Primary Key | Unique identifier for each intervention. |
| name | VARCHAR (255) | UNIQUE, NOT NULL | Name of the intervention (e.g., "Chemotherapy"). |

* **Trial\_conditions:** many-to-many relationship between trials and conditions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| trial\_id | INT | Foreign Key (references trials.trial\_id) | The ID of the trial. |
| condition\_id | INT | Foreign Key (references conditions.condition\_id) | The ID of the condition. |
| Primary Key | (trial\_id, condition\_id) | Composite Primary Key | Ensures each combination is unique. |

* **Trial\_interventions:** many-to-many relationship between trials and interventions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| trial\_id | INT | Foreign Key (references trials.trial\_id) | The ID of the trial. |
| intervention\_id | INT | Foreign Key (references interventions.intervention\_id) | The ID of the intervention. |
| Primary Key | (trial\_id, intervention\_id) | Composite Primary Key | Ensures each combination is unique. |

* **Sponsors:** Information about sponsors.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| sponsor\_id | INT (or SERIAL) | Primary Key | Unique identifier for each sponsor. |
| name | VARCHAR (255) | UNIQUE, NOT NULL | Name of the sponsor (e.g., "Pfizer"). |

* **Locations:** Where the trials are conducted.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| location\_id | INT | Primary Key | Unique identifier for each location. |
| trial\_id | INT | Foreign Key (references trials.trial\_id) | The ID of the trial is being conducted. |
| name | VARCHAR (100) | NOT NULL | The name (e.g., Hospital) |
| address | VARCHAR (255) | NOT NULL | The address (e.g., City, Hospital) |

* **Log\_pages:** Token used for pagination.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Constraints** | **Description** |
| nextPageToken | VARCHAR (255) | NOT NULL | Unique token identifying each page. |

**Implementing the ETL pipeline:**

1. **Extract**:

The objective of this step is to gather raw data from an external data source. In this case, the clinicaltrials.gov API, so that it can be transformed into a usable format for your database.

* 1. **API Interaction:** You will interact with the clinicaltrials.gov API to retrieve clinical trial data. The API provides trial details in JSON format.
  2. **Selecting Relevant Data:** The clinical trials API returns various fields about each clinical trial, including trial identifiers, conditions being studied, interventions (e.g., drugs, therapies), sponsors, locations, and status. You’ll need to extract:
     + nct\_id
     + title
     + status
     + phase
     + conditions (medical conditions being studied)
     + interventions (treatments, drugs, or procedures)
     + sponsor (pharmaceutical company or organization)
     + location (where the trial is being conducted)
  3. **Handling Pagination:** Clinicaltrials.gov may return large datasets with multiple pages of results. The extraction process must handle pagination by sending multiple requests, if necessary, until all relevant trial data has been retrieved.
  4. **Output:** The raw data from the API is typically in JSON format and will contain information such as trial metadata (e.g., trial name, status) and linked details about conditions, interventions, and sponsors.

1. **Transform:**

This step involves cleaning, restructuring, and mapping the extracted data into the format that matches the relational schema of PostgreSQL database. This ensures the data is ready for insertion into the database tables.

* 1. **Mapping Data to Database Schema**: The extracted data must be mapped to the appropriate tables in your database schema, such as:
  + **Trials Table**: This will store trial metadata (trial ID, title, status, phase, and sponsor).
  + **Conditions Table**: This stores the list of cancer conditions (e.g., "Lung Cancer").
  + **Interventions Table**: This stores the list of interventions (e.g., "Chemotherapy").
  + **Sponsors Table**: This stores information about the sponsors (e.g., pharmaceutical companies).
  + **Trial\_Conditions and Trial\_Interventions Tables**: These tables will store the many-to-many relationships between trials and conditions, as well as trials and interventions.
  1. **Data Transformation Logic**:
  + **Trial Information:** Extract the NCTId not as trial\_id, use the BriefTitle or OfficialTitle as the trial title, and map status and phase from the relevant fields.
  + **Conditions and Interventions:** Loop through the conditions and interventions associated with each trial, ensuring each one has a unique condition\_id or intervention\_id. This may involve checking if the condition or intervention already exists in the database to avoid duplicates.
  + **Sponsor Information:** Extract the sponsor’s name and ensure it is mapped to the corresponding sponsor on the sponsors table. If the sponsor is new, it needs to be inserted into the sponsors table.
  + **Handling Missing Data:** Ensure that missing or incomplete data (e.g., missing condition names or phases) is handled gracefully, either by skipping the record or assigning a default value.
  1. **Duplicate Verification:**

To ensure data integrity and avoid conflicts, the last IDs and existing names of entities should be retrieved during the transformation process. This will allow the system to check if a new entry already exists before insertion, ensuring that there are no duplicates.

* + - **Trials Table:** Retrieve the last trial\_id and any existing trial titles to ensure no new trial entries duplicate existing trials based on the title or other unique identifiers.
    - **Conditions Table:** Retrieve the last condition\_id and a list of existing condition names (e.g., "Lung Cancer") to check if the condition already exists in the database before insertion.
    - **Interventions Table:** Retrieve the last intervention\_id and the existing names of interventions (e.g., "Chemotherapy") to avoid duplicating existing intervention entries.
    - **Sponsors Table:** Retrieve the last sponsor\_id and the existing sponsor names to check if the sponsor already exists in the database, and if so, map the sponsor accordingly. If not, insert a new record.
    - **Locations Table:** If location information is part of the database schema, retrieve the last location\_id and any existing location names (e.g., "New York") to avoid duplication and ensure proper mapping of locations to trials.
  1. **Output**: This step will produce data that is now ready to be inserted into the database. This includes:
     + A list of trials with all necessary fields for insertion into the trials table.
     + A list of conditions and interventions with IDs for use in the relationship tables.
     + Many-to-many relationship data between trials and conditions, as well as trials and interventions.

1. **Load**:

This step involves inserting the transformed data into the destination PostgreSQL database. This is where the actual database population happens.

* 1. **Database Connection**:
  + Connect to your PostgreSQL database using a Python database adapter like **psycopg2**. You will need to provide the database connection parameters, including host, database name, username, and password.
  1. **Inserting Data**:
     + - **Trials Table**: Insert each trial's data (e.g., trial\_id, title, status, phase, and sponsor\_id).
       - **Conditions and Interventions Tables**: Insert the unique conditions and interventions into their respective tables. You may need to check if a condition or intervention already exists before inserting to avoid duplicates.
       - **Many-to-Many Relationship Tables**: Insert the corresponding trial-condition and trial-intervention relationships into the trial\_conditions and trial\_interventions tables.

**3.3 Output**:

* + - * After this step, the data will be stored in the PostgreSQL database, and your schema will reflect the latest information about trials, conditions, interventions, sponsors, and locations.

**Technical Stack:**

* **Programming Language**: Python (using libraries like requests for API calls, pandas for data manipulation)
* **Database**: PostgreSQL for storing the transformed data
* **API**: clinicaltrials.gov API for data extraction

**Expected Outcomes:**

* A fully functional ETL pipeline that runs on a scheduled basis (e.g., daily, weekly) to keep the database updated with the latest data from clinicaltrials.gov database.
* A well-structured relational database containing rich datasets that can be queried for insights in cancer clinical trials.
* Documentation detailing the ETL process, schema design, and usage instructions for team members.

**Benefits:**

* Enhanced accessibility to cancer clinical trials data for researchers.
* Streamlined workflow for data retrieval and analysis, reducing manual efforts.
* Improved data quality and consistency for better research outcomes.