Documentation for TICTAC

This documentation outlines the complete workflow, capturing all transformations and the relationship between the input and output files.

Documentation for `step1`

Script Name: `step1.py`

Purpose:

This script processes a dataset to generate disease-target associations. It filters, transforms, aggregates, and writes the processed data to an output file in Parquet format.

Input File:

- **Name**: `sddt_links.parquet`
- **Description**: This is a Parquet file containing raw data about disease terms, drug names, genes, and clinical study identifiers.

Output File:

- **Name**: `step1_output.parquet/`
- **Description**: The output is a partitioned Parquet file containing the processed dataset. It includes new columns: `doid_uniprot`, `nStud`, and `nct_ids`.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Sets up a Spark session optimized for processing large datasets. Memory allocation, dynamic allocation, and shuffle partitions are configured for efficiency.
2	Set Logging Level	Reduces Spark's logging verbosity to "ERROR" to minimize noise during execution.
3	Load Data	Reads the input file (`sddt_links.parquet`) into a DataFrame and repartitions the data by `doid` for efficient processing.
4	Filter Data	Removes rows with missing values in critical columns: `disease_term`, `drug_name`, and `gene_symbol`.
5	Persist DataFrame	Caches the filtered DataFrame in memory to optimize its reuse in subsequent steps.
6	Create `doid_uniprot` Column	Adds a new column (`doid_uniprot`) by concatenating `doid` and `uniprot` values with an underscore.
7	Aggregate Data	Groups the data by `doid_uniprot` and computes: - **`nStud`**: Count of unique `nct_id` values.

Step	Operation	Description
		- **`nct_ids`**: A collected set of unique `nct_id` values.
8	Join Aggregated Data	Merges the aggregated results (`nStud` and `nct_ids`) back into the original DataFrame using `doid_uniprot` as the key.
9	Select Final Columns	Extracts relevant columns for the final dataset, including `disease_term`, `doid`, `drug_name`, `gene_symbol`, and the new fields (`doid_uniprot`, `nStud`, `nct_ids`).
10	Save Output	Writes the processed DataFrame to `step1_output.parquet/`, partitioned by `doid`.

Generated Columns in Output:

Column Name	Description
`doid_uniprot`	A unique identifier combining `doid` and `uniprot`.
`nStud`	Count of distinct `nct_id` values for each `doid_uniprot`.
`nct_ids`	A list of unique `nct_id` values associated with each `doid_uniprot`.
Other Fields	`disease_term`, `doid`, `drug_name`, `gene_symbol`, `uniprot`, and other fields from the input.

Usage:

To run the script: `python step1.py`

Output Summary:

- The processed file is saved as `step1_output.parquet/`, partitioned by the `doid` column.
- This output serves as input for subsequent analytical steps or machine learning workflows.

Documentation for `step2`

Script Name: `step2.py`

Purpose:

This script processes the dataset produced by `step1.py` to generate additional aggregations, including counts of drugs and diseases per `doid_uniprot`, while retaining study-level information.

Input File:

- **Name**: `step1_output.parquet`
- **Description**: The output from `step1.py`, this file contains the dataset enriched with fields such as:

- `nStud`: Count of unique clinical study IDs (`nct_id`) per `doid_uniprot`.
- `nct_ids`: A list of unique `nct_id` values associated with each `doid_uniprot`.

Output File:

- **Name**: `step2_output.parquet`
- **Description**: The output is a Parquet file that includes:
  - Aggregated counts of drugs (`nDrug`) and diseases (`nDiseases`) for each `doid_uniprot`.
  - The fields `nStud` and `nct_ids` retained from the input dataset.
  - Scores (`semantic_score` and `frequency_score`) rounded to 4 decimal places.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Sets up a Spark session with configurations for memory, cores, and shuffle partitions.
2	Load Input Dataset	Reads the `step1_output.parquet` file into a Spark DataFrame.
3	Round Scores	Rounds `semantic_score` and `frequency_score` to 4 decimal places for consistency.
4	Group and Aggregate	Performs aggregations grouped by `disease_term`, `doid`, `gene_symbol`, `uniprot`, and `doid_uniprot`:
	- **`nDrug`**	Count of unique drugs (`drug_name`) per `doid_uniprot`.
	- **`nDiseases`**	Count of unique diseases (`doid`) per `doid_uniprot`.
	- **`nStud`**	Retains the value of `nStud` from the input dataset.
	- **`nct_ids`**	Retains the list of unique `nct_ids` from the input dataset.
	- **Scores**	Retains the first non-null values of `semantic_score` and `frequency_score` for each group.
5	Save Output Dataset	Writes the aggregated DataFrame to `step2_output.parquet` in overwrite mode.

Generated Columns in Output:

Column Name	Description
`disease_term`	Disease name/term.
`doid`	Disease ontology ID.
`gene_symbol`	Gene symbol associated with the disease-target interaction.
`uniprot`	UniProt ID for the target protein.
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`nDrug`	Count of unique drugs (`drug_name`) per `doid_uniprot`.
`nDiseases`	Count of unique diseases (`doid`) per `doid_uniprot`.
`nStud`	Count of distinct clinical study IDs (`nct_id`) retained from `step1_output.parquet`.
`nct_ids`	List of unique clinical study IDs (`nct_id`) associated with each `doid_uniprot`.
`semantic_score`	Semantic score for the disease-target interaction, rounded to 4 decimal places.

Column Name	Description
`frequency_score`	Frequency score for the disease-target interaction, rounded to 4 decimal places.

Usage:

To run the script: python `step2.py`

Output Summary:

- **File Name**: `step2_output.parquet`
- **Description**: A processed and aggregated dataset with new fields (`nDrug`, `nDiseases`) .
- **Partitioning**: Data is stored as a single Parquet file without additional partitioning.

Documentation for `step3`

Script Name: `step3.py`

Purpose:

This script processes the dataset from **`step2_output.parquet`** to associate clinical study IDs (`nct_ids`) with publication references. It computes the number of unique publications (`nPub`) and collects the associated PMIDs for each `doid_uniprot`, producing a dataset enriched with publication data.

Input Files:

1. **`step2_output.parquet`**
    - **Description**: The output from `step2.py`, containing aggregated data with fields such as:
        - `nStud`: Count of unique clinical study IDs for each `doid_uniprot`.
        - `nct_ids`: List of unique `nct_id` values associated with each `doid_uniprot`.
2. **`aact_study_refs_trans_corrected.parquet`**
    - **Description**: A dataset mapping clinical study IDs (`nct_id`) to publication IDs (`pmid`), which is used to associate studies with their corresponding publications.

Output File:

- **Name**: `step3_output.parquet`
- **Description**: The output is a Parquet file containing enriched data that includes:
    - The count of unique publications (`nPub`) associated with each `doid_uniprot`.
    - The list of unique PMIDs (`pmids`) for each `doid_uniprot`.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Sets up a Spark session with appropriate memory configurations.
2	Load Input Files	Reads `step2_output.parquet` and `aact_study_refs_trans_corrected.parquet` into Spark DataFrames.
3	Explode `nct_ids` Column	Expands the list of `nct_ids` into individual rows to allow for mapping with publication data.
4	Join with Publication Data	Performs a left join with `aact_study_refs_trans_corrected.parquet` to associate each `nct_id` with its corresponding `pmid`.
5	Aggregate by `doid_uniprot`	Groups the data by `doid_uniprot` to calculate: - **`nPub`**: Count of distinct PMIDs for each `doid_uniprot`. - **`pmids`**: List of unique PMIDs for each `doid_uniprot`.
6	Join Aggregated Data	Merges the aggregated publication data back into the original dataset from `step2_output.parquet`.
7	Save Output Dataset	Writes the resulting dataset to `step3_output.parquet` in overwrite mode.

Generated Columns in Output:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`gene_symbol`	Gene symbol associated with the disease-target interaction.
`uniprot`	UniProt ID for the target protein.
`nDiseases`	Count of unique diseases (`doid`) per `doid_uniprot`.
`doid`	Disease ontology ID.
`nDrug`	Count of unique drugs (`drug_name`) per `doid_uniprot`.
`nStud`	Count of distinct clinical study IDs (`nct_id`).
`nct_ids`	List of unique clinical study IDs (`nct_id`) associated with each `doid_uniprot`.
`semantic_score`	Semantic score for the disease-target interaction.
`frequency_score`	Frequency score for the disease-target interaction.
`nPub`	Count of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`pmids`	List of unique PMIDs associated with clinical studies for each `doid_uniprot`.

Usage:

To run the script: python `step3.py`

Output Summary:

- **File Name**: `step3_output.parquet`
- **Description**: A dataset enriched with publication-level data (`nPub`, `pmids`) for each `doid_uniprot`. This data allows for detailed analysis of publications associated with disease-target interactions.

Documentation for `step4`

Script Name: `step4.py`

Purpose:

This script enriches the dataset from **`step3_output.parquet`** by calculating a metric called `nStudyNewness`, which uses an exponential decay function to measure the recency and importance of clinical studies associated with each `doid_uniprot`.

Input Files:

1. **`step3_output.parquet`**
    - **Description**: The output from `step3.py`, containing aggregated disease-target data, publication information (`nPub`, `pmids`), and clinical study IDs (`nct_ids`).
2. **`aact_study_refs_trans_corrected.parquet`**
    - **Description**: A dataset mapping clinical study IDs (`nct_id`) to publication years, used to calculate the `nStudyNewness` metric.

Output File:

- **Name**: `step4_output.parquet`
- **Description**: A Parquet file enriched with the `nStudyNewness` metric, which quantifies the recency and weighted significance of associated clinical studies.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Sets up a Spark session with appropriate memory configurations.
2	Load Input Files	Reads `step3_output.parquet` and `aact_study_refs_trans_corrected.parquet` into Spark DataFrames.
3	Explode `nct_ids`	Expands the list of `nct_ids` into individual rows to allow mapping with study years.
4	Join Study Years	Joins the exploded dataset with the reference dataset to associate `nct_id` with its corresponding year.
5	Calculate Study Age	Computes the age of each study relative to 2024, assigning an age of 1 for studies from 2024.
6	Apply Exponential Decay	Uses an exponential decay function to compute the weighted significance of each study's age.
7	Aggregate `nStudyNewness`	Sums the weighted ages for each `doid_uniprot` to calculate the `nStudyNewness` metric.
8	Save Enriched Dataset	Writes the enriched dataset to `step4_output.parquet` in overwrite mode.

Generated Columns in Output:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`gene_symbol`	Gene symbol associated with the disease-target interaction.
`uniprot`	UniProt ID for the target protein.
`nDiseases`	Count of unique diseases (`doid`) per `doid_uniprot`.
`doid`	Disease ontology ID.
`nDrug`	Count of unique drugs (`drug_name`) per `doid_uniprot`.
`nStud`	Count of distinct clinical study IDs (`nct_id`).
`nct_ids`	List of unique clinical study IDs (`nct_id`) associated with each `doid_uniprot`.
`semantic_score`	Semantic score for the disease-target interaction.
`frequency_score`	Frequency score for the disease-target interaction.
`nPub`	Count of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`pmids`	List of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`nStudyNewness`	Weighted sum of study ages, using exponential decay to prioritize newer studies.

Usage:

To run the script: python `step4.py`

Output Summary:

- **File Name**: `step4_output.parquet`
- **Description**: The dataset includes all fields from `step3_output.parquet` and adds the `nStudyNewness` metric, which quantifies the recency and relevance of associated studies.

- **Partitioning**: Data is stored as a single Parquet file without additional partitioning.

Documentation for `step5`

Script Name: `step5.py`

Purpose:

This script enriches the dataset from **`step4_output.parquet`** by calculating a weighted publication metric (`nPublicationWeighted`) for each `doid_uniprot`. The metric is computed based on weights assigned to different publication types (result, background, derived) and aggregated for each `doid_uniprot`.

Input Files:

1. **`step4_output.parquet`**
    - **Description**: The output from `step4.py`, containing fields such as:
        - `nStudyNewness`: A metric quantifying the recency and relevance of clinical studies.
        - `nPub` and `pmids`: Information about the number of publications and their IDs.
        - `nct_ids`: A list of clinical study IDs associated with each `doid_uniprot`.
2. **`aact_study_refs_trans_corrected.parquet`**
    - **Description**: A reference dataset mapping clinical study IDs (`nct_id`) to reference types (`reference_type`), which are used to calculate weighted publications.

Output File:

- **Name**: `step5_output.parquet`
- **Description**: The output is a Parquet file enriched with the `nPublicationWeighted` metric, which aggregates weighted publication counts for each `doid_uniprot`.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Sets up a Spark session with appropriate memory configurations.
2	Load Input Files	Reads `step4_output.parquet` and

Step	Operation	Description
		`aact_study_refs_trans_corrected.parquet` into Spark DataFrames.
3	Explode `nct_ids`	Expands the list of `nct_ids` into individual rows for mapping with reference types.
4	Join with Reference Data	Joins the exploded dataset with the reference dataset to associate each `nct_id` with its `reference_type`.
5	Define Weight Mapping - `0.0`: Result publications (weight = 1.0). - `1.0`: Background publications (weight = 0.5). - `2.0`: Derived publications (weight = 0.25).	Maps `reference_type` to predefined weights:
6	Add Weighted Values	Adds a column (`weighted_type`) that assigns weights to each row based on its `reference_type`.
7	Aggregate `nPublicationWeighted`	Groups data by `doid_uniprot` and sums up the weighted values to compute `nPublicationWeighted`.
8	Join Back to Dataset	Merges the `nPublicationWeighted` metric back into the original dataset from `step4_output.parquet`.
9	Save Enriched Dataset	Writes the enriched dataset to `step5_output.parquet` in overwrite mode.

Generated Columns in Output:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`gene_symbol`	Gene symbol associated with the disease-target interaction.
`uniprot`	UniProt ID for the target protein.
`nDiseases`	Count of unique diseases (`doid`) per `doid_uniprot`.
`doid`	Disease ontology ID.
`nDrug`	Count of unique drugs (`drug_name`) per `doid_uniprot`.
`nStud`	Count of distinct clinical study IDs (`nct_id`).
`nct_ids`	List of unique clinical study IDs (`nct_id`) associated with each `doid_uniprot`.
`semantic_score`	Semantic score for the disease-target interaction.
`frequency_score`	Frequency score for the disease-target interaction.
`nPub`	Count of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`pmids`	List of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`nStudyNewness`	Weighted sum of study ages, using exponential decay to prioritize newer studies.
`nPublicationWeighted`	Sum of weighted publications, where weights are based on `reference_type` values.

Usage:

To run the script: python `step5.py`

Output Summary:

- **File Name**: `step5_output.parquet`
- **Description**: A dataset enriched with the `nPublicationWeighted` metric, which quantifies the weighted significance of publications associated with each `doid_uniprot`. This metric complements existing information such as `nStudyNewness` and `nPub`.

Documentation for `step6.py`

Script Name: `step6.py`

`This script calculates comprehensive rankings and generates two output tables for further analysis of disease-target associations.`

Purpose:

This script processes the dataset from **`step5_output.parquet`** to compute rankings, calculate mean rank scores, and generate two output datasets:

1. **`disease_target_association.parquet`**: Contains detailed association metrics for each disease-target pair.
2. **`ranking_info.parquet`**: Contains ranking information for numerical metrics used in the analysis.

Input File:

- **`step5_output.parquet`**
    - **Description**: The output from `step5.py`, containing fields such as:
        - `nDiseases`, `nDrug`, `nStud`, `nPub`: Numerical metrics representing different aspects of disease-target associations.
        - `nStudyNewness`: A metric quantifying the recency and relevance of clinical studies.
        - `nPublicationWeighted`: A metric reflecting the weighted significance of associated publications.

Output Files:

1. **`disease_target_association.parquet`**
    - **Description**: Contains association metrics and a calculated rank score for each disease-target pair, with fields:
        - `doid_uniprot`, `gene_symbol`, `nDiseases`, `nDrug`, `nStud`, `nPub`, `nStudyNewness`, `nPublicationWeighted`, `meanRankScore`, `nct_ids`, `pmids`.
2. **`ranking_info.parquet`**
    - **Description**: Contains detailed ranking information for numerical metrics and their computed mean rank scores, with fields:
        - `doid_uniprot`, `meanRank`, `percentile_meanRank`, `meanRankScore`, and ranks for each numerical metric (`nDiseases_rank`, `nDrug_rank`, `nStud_rank`, `nPub_rank`, `nStudyNewness_rank`, `nPublicationWeighted_rank`).

Steps in the Script:

Step	Operation	Description
1	Load Input Dataset	Reads `step5_output.parquet` into a Spark DataFrame.
2	Compute Ranks for Variables	Ranks each numerical metric (`nDiseases`, `nDrug`, `nStud`, `nPub`, `nStudyNewness`, `nPublicationWeighted`) in descending order and creates a new column for each rank.
3	Calculate `meanRank`	Computes the average rank across all numerical metrics for each `doid_uniprot`.
4	Calculate Percentile and `meanRankScore` - **Percentile**: Calculates the percentile rank for `meanRank` across the dataset. - **`meanRankScore`**: Converts percentile rank into a score (higher is better) scaled between 0 and 100.	
5	Create Disease-Target Association Table	Selects relevant columns to generate `disease_target_association.parquet`.
6	Create Ranking Information Table	Selects relevant columns to generate `ranking_info.parquet`.
7	Save Output Datasets	Saves the two output tables as Parquet files.

Generated Columns in Output Files:

`disease_target_association.parquet`

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.

Column Name	Description
gene_symbol	Gene symbol associated with the disease-target interaction.
nDiseases	Count of unique diseases (doid) per doid_uniprot.
nDrug	Count of unique drugs (drug_name) per doid_uniprot.
nStud	Count of distinct clinical study IDs (nct_id).
nPub	Count of unique PMIDs associated with clinical studies for each doid_uniprot.
nStudyNewness	Weighted sum of study ages, using exponential decay to prioritize newer studies.
nPublicationWeighted	Sum of weighted publications, where weights are based on reference_type values.
meanRankScore	A score representing the overall rank of the association, scaled between 0 and 100.
nct_ids	List of unique clinical study IDs (nct_id) associated with each doid_uniprot.
pmids	List of unique PMIDs associated with clinical studies for each doid_uniprot.

ranking_info.parquet

Column Name	Description
doid_uniprot	Unique identifier combining doid and uniprot.
meanRank	Average rank across all numerical metrics for each doid_uniprot.
percentile_meanRank	Percentile rank of the mean rank across the dataset.
meanRankScore	Rank score derived from the percentile rank, scaled between 0 and 100.
nDiseases_rank	Rank of nDiseases metric in descending order.
nDrug_rank	Rank of nDrug metric in descending order.
nStud_rank	Rank of nStud metric in descending order.
nPub_rank	Rank of nPub metric in descending order.
nStudyNewness_rank	Rank of nStudyNewness metric in descending order.
nPublicationWeighted_rank	Rank of nPublicationWeighted metric in descending order.

Usage:

To run the script: python `step6.py`

Output Summary:

1. **disease_target_association.parquet**:
   - Provides detailed association metrics for each disease-target pair.
   - Includes scores and lists of clinical studies and publications.
2. **ranking_info.parquet**:
   - Contains ranking information for each disease-target pair, including ranks for individual metrics and the overall mean rank score.

Documentation for step7.py

Script Name: `step7.py`

Purpose:

This script merges disease-gene associations (from **`disease_gene_association.parquet`**) with additional drug-target interaction data (from **`sddt_links2.parquet`**) on the `doid_uniprot` column. The resulting dataset re-integrate drug names, disease terms and idgTDL columns that were originally present

---

Input Files:

1. **`disease_gene_association.parquet`**
    - **Description**: This dataset contains metrics and associations for disease-gene pairs, generated from the previous pipeline steps.
        - Key columns include:
            - `doid_uniprot`: Unique identifier for disease-gene pairs.
            - Metrics such as `nDiseases`, `nDrug`, `nStud`, `nPub`, `nStudyNewness`, `nPublicationWeighted`.
2. **`sddt_links2.parquet`**
    - **Description**: This dataset includes drug-target interaction data with the following fields:
        - `doid_uniprot`: Unique identifier for disease-gene pairs.
        - `disease_term`: Description of the disease.
        - `drug_name`: Name of the associated drug.
        - `idgTDL`: Target Development Level (TDL) classification for the drug (e.g., `Tchem`, `Tclin`).

Output File:

- **Name**: `merged_disease_drug_association.parquet`
- **Description**: The output is a Parquet file containing merged data that combines disease-gene associations with drug-target interaction information, partitioned by the `gene_symbol` column to optimize querying.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Configures a Spark session with optimized memory and shuffle partition settings for large-scale data processing.
2	Load Input Datasets	Loads **`disease_gene_association.parquet`** and **`sddt_links2.parquet`** into

Step	Operation	Description
		Spark DataFrames, persisting them in memory to optimize processing.
3	Merge Datasets	Performs an inner join on the `doid_uniprot` column to merge disease-gene associations with drug-target interaction data.
4	Broadcast Join (Optional)	Uses Spark's broadcast join optimization for the smaller dataset to improve join performance.
5	Save Merged Dataset	Writes the merged dataset to **`merged_disease_drug_association.parquet`** in overwrite mode, partitioned by the `gene_symbol` column.

Generated Columns in Output File:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`gene_symbol`	Gene symbol associated with the disease-target interaction.
`nDiseases`	Count of unique diseases (`doid`) per `doid_uniprot`.
`nDrug`	Count of unique drugs (`drug_name`) per `doid_uniprot`.
`nStud`	Count of distinct clinical study IDs (`nct_id`).
`nPub`	Count of unique PMIDs associated with clinical studies for each `doid_uniprot`.
`nStudyNewness`	Weighted sum of study ages, using exponential decay to prioritize newer studies.
`nPublicationWeighted`	Sum of weighted publications, where weights are based on `reference_type` values.
`disease_term`	Disease description from `sddt_links2.parquet`.
`drug_name`	Name of the associated drug from `sddt_links2.parquet`.
`idgTDL`	Target Development Level (TDL) classification for the drug (e.g., `Tchem`, `Tclin`).

Usage:

To run the script: `python step7.py`

Output Summary:

- **File Name**: `merged_disease_drug_association.parquet`
- **Description**: A dataset combining disease-gene associations with drug-target interaction data for comprehensive analysis. The output is partitioned by `gene_symbol` for optimized querying and storage.

Documentation for `step8.py`

Script Name: `step8.py`

Purpose:

This script transforms and integrates study reference data from **`aact_study_refs.tsv`** with disease-drug associations from **`merged_disease_drug_association.parquet`**. The output dataset links study references, including publication metadata, with the corresponding disease-gene-drug associations.

Input Files:

1. **`aact_study_refs.tsv`**
    - **Description**: A tab-separated file containing study reference information, including:
        - `nct_id`: Clinical study ID.
        - `reference_type`: Type of the reference (e.g., `BACKGROUND`, `RESULT`).
        - `pmid`: PubMed ID (optional).
        - `citation`: Full citation text for the reference.
2. **`merged_disease_drug_association.parquet`**
    - **Description**: A Parquet file containing merged disease, gene, and drug associations, including:
        - `doid_uniprot`: Unique identifier for disease-gene pairs.
        - `nct_ids`: List of clinical study IDs associated with each `doid_uniprot`.

Output File:

- **Name**: `study_ref_redundant.parquet`
- **Description**: The output dataset links study references to disease-gene-drug associations, containing:
    - `doid_uniprot`: Unique identifier for disease-gene pairs.
    - `nct_id`: Clinical study ID.
    - `reference_type`: Type of the reference (e.g., `BACKGROUND`, `RESULT`).
    - `pmid`: PubMed ID (optional).
    - `citation`: Full citation text.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Configures a Spark session with memory and shuffle settings optimized for large-

Step	Operation	Description
		scale data processing.
2	Load Study References	Reads `aact_study_refs.tsv` into a Spark DataFrame, selecting relevant columns (`nct_id`, `reference_type`, `pmid`, `citation`).
3	Load Merged Disease-Drug Associations	Reads `merged_disease_drug_association.parquet`, selecting `doid_uniprot` and `nct_ids`.
4	Explode `nct_ids`	Expands the list of `nct_ids` into individual rows, creating one row per `nct_id` for each `doid_uniprot`.
5	Join Datasets on `nct_id`	Performs an inner join between exploded disease-drug associations and study references on the `nct_id` column.
6	Select Relevant Columns	Filters the joined dataset to retain only the required fields: `doid_uniprot`, `nct_id`, `reference_type`, `pmid`, `citation`.
7	Save Transformed Data	Writes the resulting dataset to **`study_ref_redundant.parquet`** in Parquet format.

Generated Columns in Output File:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`nct_id`	Clinical study ID.
`reference_type`	Type of the reference (e.g., `BACKGROUND`, `RESULT`).
`pmid`	PubMed ID for the reference (optional).
`citation`	Full citation text for the reference.

Usage:

To run the script: `spark-submit step8.py`

Output Summary:

- **File Name**: `study_ref_redundant.parquet`
- **Description**: A dataset linking study references to disease-gene-drug associations. This provides a comprehensive view of the studies and their associated references, which can be used for further analysis.

Documentation for `step9.py`

Script Name: `step9.py`

Purpose:

This script deduplicates the study references dataset generated in the previous step (**`study_ref_redundant.parquet`**) based on key columns. It optimizes performance through checkpointing and repartitioning, ensuring efficient processing and storage. The output is saved as **`study_refs.parquet`**.

Input File:

- **`study_ref_redundant.parquet`**
  - **Description**: A Parquet file linking study references to disease-gene-drug associations. This dataset contains:
    - `doid_uniprot`: Unique identifier for disease-gene pairs.
    - `nct_id`: Clinical study ID.
    - `reference_type`: Type of the reference (e.g., `BACKGROUND`, `RESULT`).
    - `pmid`: PubMed ID (optional).
    - `citation`: Full citation text.

Output File:

- **Name**: `study_refs.parquet`
- **Description**: The deduplicated dataset that ensures unique records for each combination of `doid_uniprot`, `nct_id`, `reference_type`, and `pmid`. The output is partitioned by `doid_uniprot` for optimized storage and querying.

Steps in the Script:

Step	Operation	Description
1	Initialize Spark Session	Configures a Spark session with optimized memory, shuffle, and checkpointing settings for large-scale data processing.
2	Load Input Dataset	Reads **`study_ref_redundant.parquet`** into a Spark DataFrame.
3	Deduplicate Data	Removes duplicate rows based on the specified subset of columns: `doid_uniprot`, `nct_id`, `reference_type`, and `pmid`.
4	Set Checkpointing	Configures a local checkpoint directory to enable intermediate result storage for fault tolerance.
5	Repartition Data	Reorganizes the data into smaller partitions and partitions the output by `doid_uniprot`.
6	Save Deduplicated Dataset	Writes the deduplicated dataset to **`study_refs.parquet`** in Parquet format.

Generated Columns in Output File:

Column Name	Description
`doid_uniprot`	Unique identifier combining `doid` and `uniprot`.
`nct_id`	Clinical study ID.
`reference_type`	Type of the reference (e.g., `BACKGROUND`, `RESULT`).
`pmid`	PubMed ID for the reference (optional).
`citation`	Full citation text for the reference.

Usage:

To run the script: python `step9.py`

Output Summary:

- **File Name**: `study_refs.parquet`
- **Description**: A deduplicated and optimized dataset for study references, partitioned by `doid_uniprot` to facilitate efficient querying and storage.

This documentation provides a clear overview of the deduplication and optimization process performed by `step9.py`.