Unveiling Patterns of Self-Harm and Suicide:

A Data-Driven Analytical Approach

IE 7945 Group B Final Report

By Desiree Reid, Priska Mohunsingh, Karnaz Obaidullah, Peter Mink, Vy Hoang

Table of Contents

EXECUTIVE SUMMARY

This project set out to systematically uncover hidden patterns, disparities, and research gaps in the academic literature on suicide and self-harm using SafetyLit as our primary source. Our central goal was not only to analyze existing research but also to construct a complete and reliable dataset that could support downstream discovery and modeling efforts with confidence. By scraping over 29,000 bibliographic records from SafetyLit and integrating metadata from sources such as PubMed and CrossRef, we developed a robust and scalable pipeline that allowed us to recover missing identifiers (e.g., DOIs, PMIDs), affiliations, categories, and other crucial metadata fields.

The most significant challenge—and arguably the most impactful contribution—was addressing the incompleteness and inconsistency of academic metadata. Titles were inconsistently formatted across platforms, affiliation data was embedded in noisy text, and key fields like grant information or suicide prevention strategies were often missing or variably recorded. Through iterative development, we applied fuzzy matching, regular expression-based cleaning, and NLP pipelines to fill in these gaps. This process was essential for transitioning from raw, fragmented records to a trustworthy and queryable research corpus.

Armed with this enriched dataset, we entered the "discovery" phase of the project, where we were able to reveal latent structures in the literature. For instance, our modeling identified systemic bias in gender-focused suicide research: nearly all models performed well on male-labeled abstracts but struggled to detect female-focused studies. Topic modeling and text clustering on adolescent suicide articles revealed the dominance of themes like depression, anxiety, and self-harm, suggesting an underexploration of external or environmental factors. Time series analyses further showed consistent growth in suicide-related publications, with a notable spike around 2014 that warrants further contextual investigation.

In short, this project demonstrates how completeness of data is a prerequisite for discovery. Without resolving missing values, normalizing relationships, and understanding the source-specific rules of scraped content, meaningful insights would have remained buried. Our work serves not only as a technical case study in data recovery and validation but also as a replicable foundation for future research aiming to identify disparities and trends in the suicide prevention literature.

ABSTRACT

In the shadows of rising despair, where numbers often obscure the pain of lives lived and lost, this research endeavors to cast light upon the intricate tapestry of suicide and self-harm. Using bibliographic data from SafetyLit articles spanning January 1, 2005, to December 31, 2014, this study analyzes the nuanced and multilayered factors—psychological, socioeconomic, cultural, and beyond—that intersect to shape the narratives of suicide in the United States.

Leveraging Python for bibliographic data collection and processing, unstructured BibTeX files were transformed into a structured and clean CSV dataset. The automation of data extraction and transformation was facilitated using a range of Python libraries, including os, time, re, urllib.parse, requests, BeautifulSoup for web scraping, and selenium with webdriver_manager for automating interactions with web pages. Specifically, Selenium's WebDriver was used to navigate and extract structured data from SafetyLit, employing WebDriverWait, Select, and expected_conditions for handling dynamic content.

This analysis seeks to uncover trends, correlations, and potential causations behind critical questions, such as why suicide rates remain alarmingly high in the U.S. compared to other regions and why men consistently have higher suicide rates than women.

Each data point tells a story—of struggle, resilience, or loss. By bridging statistical rigor with compassionate inquiry, this research aims to contribute meaningful insights that might one day inform more effective prevention strategies. Behind every number lies a human life and understanding the "why" is a step toward saving the "who."

Through this project, we aim to better understand the social, psychological, and demographic factors driving self-harm and suicide. By using this data-driven approach, we hope to provide insights that may inform and improve prevention efforts.

Research Questions

We aim for this project to explore key solution-oriented questions on self-harm and suicide:

1. What terms or topics have emerged as most distinctive in suicide prevention research over time?
2. Why is suicide more prevalent among males in US than females?
3. Which mental health-related topics co-occur most often with discussions of adolescent suicide in research papers?
4. Identify the most effective suicide prevention measures across studies.
5. Predict how long after exposure to risk factors an individual may attempt suicide.
6. How do suicide rates and interventions change over time?

By addressing these questions, the research aims to uncover critical insights to improve prevention strategies and save lives.

Hypothesis

Our hypothesis is that societal pressures, stigma around seeking mental health support, and barriers to accessing care contribute to the higher suicide rates among men**.** We predict that our dataset will reveal evidence of these factors through trends in the literature, including discussions on social isolation, economic hardship, and differences in coping mechanisms compared to women.

Progress Summary

In the initial phase of this project, we focused on establishing a solid foundation for data collection and quality assurance. Over the first three weeks, we made sure to:

1. Familiarize ourselves with SafetyLit as the primary data source.
2. Implement web scraping techniques using Beautiful Soup, Habanero, and Zotero for managing bibliographic metadata.
3. Conduct rigorous quality control, ensuring extracted data is consistent, complete, and clean for subsequent analysis.
4. Finalize web scraping using Selenium, Beautiful Soup, and ChromeDriverManager, without the usage of Zotero.

Data Source

- **Website:** SafetyLit
- **Scope**: This project specifically focuses on articles related to suicide and selfharm to ensure a detailed and relevant dataset.
- **Time Frame:** Articles from 1/1/2005 to 12/31/2014.

WEB SCRAPING

To improve the accuracy and reliability of collecting article URLs from SafetyLit, we updated our original scraping script to better handle inconsistencies in title formatting. Initially, a straightforward string match wasn't sufficient. Titles from our input dataset often didn't align exactly with how they appeared on the SafetyLit site with URLs. Variations in punctuation, casing, or word order led to mismatches. To address this, we introduced fuzzy matching using the RapidFuzz library. We normalized all titles (i.e. stripping punctuation, collapsing whitespace, and converting everything to lowercase), and used token_sort_ratio to compare each title against the scraped titles from SafetyLit. We chose an 85% similarity threshold to filter out poor matches while still capturing the correct articles. If a title fell below this threshold, we logged the top match suggestions to review later.

To get better visibility into what was happening during runtime, we enhanced our logging. Each log entry shows whether a match was made (along with the similarity score and resulting URL) or if a title didn't have a strong enough match, along with top alternatives. This helped not only with transparency but also made debugging more straightforward and allowed us to flag articles for manual review.
Since scraping hundreds of pages sequentially proved inefficient, we restructured the script to run in parallel using Python's multiprocessing module. We divided the total number of pages into batches (in our case, 10 pages per batch) and assigned each batch to a separate worker process. Each process ran its own instance of Selenium in headless mode, allowing for simultaneous scraping without collisions. After all batches completed, we combined the results into a single mapping of article titles and their corresponding URLs.

Overall, fuzzy matching for better accuracy, stronger logging for observability, and parallel processing for scalability, resulted in a much more robust and efficient scraping pipeline. Upon validation, we were able to confirm the full accuracy of these improvements in the quality of our matched data, which gave us the confidence to ensure the pipeline can scale to larger datasets or future expansions of the SafetyLit archive.

Process and Methodology

The scraping workflow consists of the following key steps:

1. **Website Navigation** – Automates access to SafetyLit's archives and performs an advanced search for relevant articles based on specified keywords, publication type, and date range.
2. **Article Identification & Extraction** – Retrieves article metadata, including title, authors, publication date, abstract, and, when available, full text or direct links.
3. **Data Processing & Transformation** –
   - ○ Converts BibTeX references into CSV format.
   - ○ Scrapes PMIDs and author affiliations. ○ Extracts article categories.
4. **Data Storage & Enhancement** – Stores the extracted data in structured formats (CSV/JSON) for further analysis. Additionally, NLP techniques are applied to enrich the dataset by extracting keywords, research methodologies, prevention measures, and funding information.

Script Execution Order

1. **1-BibTeX-automation.py**
   Purpose: Automates the download or generation of .bib files.

2. **2-BibTeX-to-CSV.py**
   Purpose: Converts the .bib files to a structured CSV, which will be the input for downstream scripts.

3. **3-PMID-affiliations-scraping.py**
   Purpose: Scrapes affiliations using PMIDs from the structured data.

4. **4-Category-scraping.py**
   Purpose: Scrapes article categories or labels from external sources (journal classifications or metadata pages).

5. **7-Grant-completeness.py**
   Purpose: Likely checks for or scrapes grant-related metadata, verifying funding completeness.

6. **8-Fill-missing-PMIDs.py**
   Purpose: Backfills PMIDs for entries that might be missing them using title/author lookups.

7. **9-Fill-missing-DOIs.py**
   Purpose: Finds DOIs for entries that don't have them (e.g., via CrossRef or other APIs).

8. **10-Fill-missing-urls.py**
   Purpose: Appends direct article URLs (publisher links or open-access links) if not present.

Post-Scraping Cleanup & NLP

9. **6-Data-cleaning-script.py**
   Purpose: Cleans the merged and scraped dataset (deduplication, standardization, etc.).

10. **5-NLP-script.py**
    Purpose: Performs downstream NLP tasks like keyword extraction, sentiment analysis, or text classification. Utilizes NLP for keyword extraction, research methodology classification, and suicide prevention strategy identification

This automated pipeline enables efficient data collection, transformation, and enrichment, supporting further analysis of research trends in suicide and self-harm literature.

Challenges Encountered

While working on this project, we faced several challenges that required iterative problem-solving and adaptive techniques. These challenges primarily revolved around data extraction, standardization, and API integration, as detailed below:

1. **Navigating Pagination** – SafetyLit's input-based pagination required precise scripting to ensure all pages were traversed without skipping or revisiting sections. Additionally, dynamic content loading complicated the process, necessitating iterative debugging and script adjustments.
2. **Metadata Inconsistencies** – Certain fields, such as author names and publication years, were inconsistently structured across records. This required frequent modifications to our scraping logic to properly extract and standardize the metadata.
3. **HTML Variability** – Differences in HTML structure across article pages posed challenges in extracting nuanced fields like keywords, affiliations, and abstracts.

Custom selectors had to be frequently updated to accommodate these variations.

4. **Special Character Issues** – The presence of HTML-encoded entities (e.g., " for quotation marks) led to formatting inconsistencies. Preprocessing steps were implemented to clean these characters and standardize the dataset.
5. **Habanero Integration Challenges** – Queries to CrossRef using Habanero occasionally returned ambiguous results, particularly for articles with common titles or missing author names. To improve accuracy, we introduced filtering logic that prioritized matches based on title similarity and publication year.
6. **API Rate Limits** – Both Zotero and Habanero APIs imposed rate limits, slowing data collection and requiring the implementation of throttling mechanisms to prevent disruptions.
7. **Error Propagation** – Refinements to our scraping and preprocessing logic occasionally introduced issues that affected previously functional components. Continuous testing and debugging were necessary to ensure script reliability.
8. **Field Completeness & Cross-Referencing** – Ensuring that we captured all required fields from SafetyLit and additional sources posed a challenge. Cross-referencing the collected data against the provided documentation and validating the number of extracted articles with external sources was essential to ensure completeness.
9. **Retrieving Additional Metadata** – Some required fields were not available directly through SafetyLit, necessitating integration with PubMed and CrossRef APIs. These additional steps introduced complexity in data merging and cleaning.

Despite these challenges, collaborative problem-solving and continuous refinement of our scripts enabled us to develop a robust and scalable web scraping process. The final dataset successfully captured key metadata fields while maintaining accuracy and consistency.

Data Preprocessing and Validation

These were our critical steps in ensuring the dataset's accuracy and reliability. We focused on cleaning, standardizing, and verifying extracted information in the data to address inconsistencies and prepare it for analysis. By implementing a series of targeted quality control measures, we ensured that the dataset was comprehensive and free from redundant or inaccurate entries.

- **Cleaning Special Characters**: Converted HTML entities into proper text for consistency.
- **Handling Missing Fields**: Replaced missing values (e.g., DOI, abstract) with "N/A" to maintain dataset uniformity.
- **Formatting Author Names**: Standardized multi-author entries, separating names with semicolons.
- **Removing Redundant Fields**: Eliminated irrelevant fields like affiliation, PMID, and copyright.

- **Deduplication**: Ensured uniqueness by removing duplicate entries based on a combination of title and DOI.

To ensure the integrity of the extracted data, we carried out several validation checks. First, we made sure that all required fields, such as title, journal, and year, were properly populated. This step was crucial in maintaining the completeness of the dataset. We also checked for duplicates to avoid any redundancy that could skew analysis or create unnecessary complications. Afterwards, we focused on maintaining consistent formatting for multi-author names by separating them with semicolons, ensuring the dataset was clean and standardized for further processing.

In our preprocessing step, we focused on cleaning and standardizing the data to ensure it was ready for analysis. First, we addressed issues with special characters and HTML entities, such as ", by converting them into proper text. Missing fields, including DOI, abstract, and authors, were flagged and replaced with a default value of "N/A" to maintain consistency across the dataset. We also standardized the formatting of author names by ensuring they were combined into a single field and separated by semicolons for uniformity. We furthermore removed unnecessary fields like affiliation, PMID, and copyright, which were consistently empty and not relevant to our analysis. To avoid duplication, we implemented a step that checked for repeated entries by comparing unique identifiers, such as the combination of the title and DOI. These steps helped streamline the dataset and ensured that it was clean, consistent, and ready for further use.

CONCEPTUAL DATABASE MODELING

Database Schema and Entity Relationships

1. **Article:** articleID (Primary Key), title, publicationYear, doi (Digital Object Identifier), url, volume, number, pages, abstract, journalID (Foreign Key, references Journal), languageID (Foreign Key, references Language), PMID (PubMed ID)
2. **Journal:** journalID (Primary Key), journalName
3. **Author:** authorID (Primary Key), authorName
4. **Affiliation:** affiliationID (Primary Key), institutionName, department, location
5. **Language:** languageID (Primary Key), abbreviation, languageName
6. **Category:** categoryID (Primary Key), categoryName
7. **Keyword:** keywordID (Primary Key), keyword
8. **Prevention Measures:** preventionID (Primary Key), preventionMeasure
9. **Grant Information:** grantID (Primary Key), grantDetails
10. **Research Methodology:** methodologyID (Primary Key), methodologyType
11. **Data Source:** sourceID (Primary Key), sourceName

Entity Relationships and Cardinality in the Database Schema

Article – Journal

- Relationship: "published in"
- Rationale: Each article is published in one journal, but a journal can contain many articles.
- Cardinality: *Many (Article) → One (Journal)*

Article – Author

- Relationship: "written by"
- Rationale: Each article can be written by multiple authors, and each author can write multiple articles.
- Cardinality: *Many (Article) → Many (Author)*
- Implementation: Requires a junction table (ArticleAuthor), storing articleID and authorID.

Article – Language

- Relationship: "written in"
- Rationale: Each article is written in one language, but multiple articles can be written in the same language.
- Cardinality: *Many (Article) → One (Language)*

Article – Affiliation

- Relationship: "has"
- Rationale: Some articles are associated with an institution through the authors' affiliations.
- Cardinality: *Many (Article) → Many (Affiliation) via AuthorAffiliation*

Article – Category

- Relationship: "categorized as"
- Rationale: Each article can belong to multiple categories, and each category can contain multiple articles.
- Cardinality: *Many (Article) → Many (Category)*
- Implementation: Requires a junction table (ArticleCategory), storing articleID and categoryID.

Article – Keyword

- Relationship: "tagged with"
- Rationale: Articles can have multiple keywords, and each keyword can be used in multiple articles.
- Cardinality: *Many (Article) → Many (Keyword)*
- Implementation: Requires a junction table (ArticleKeyword), storing articleID and keywordID.

Article – Prevention Measures

- Relationship: "linked to"
- Rationale: Some articles suggest prevention measures, and some measures are linked to multiple articles.
- Cardinality: *Many (Article) → Many (Prevention Measures)*

- Implementation: Requires a junction table (ArticlePrevention), storing articleID and preventionID.

Article – Grant Information

- Relationship: "has"
- Rationale: Some articles are linked to grant funding, while a grant can support multiple articles.
- Cardinality: *Many (Article) → Many (Grant Information)*
- Implementation: Requires a junction table (ArticleGrant), storing articleID and grantID.

Article – Research Methodology

- Relationship: "comes from"
- Rationale: Each article is based on a specific research methodology, and methodologies apply to multiple articles.
- Cardinality: *Many (Article) → One (Research Methodology)*

Article – Data Source

- Relationship: "published in"
- Rationale: Articles originate from a specific data source, but multiple articles can come from the same source.
- Cardinality: *Many (Article) → One (Data Source)*

Table 1.1: Database Schema for SafetyLit DB and Their Relationships

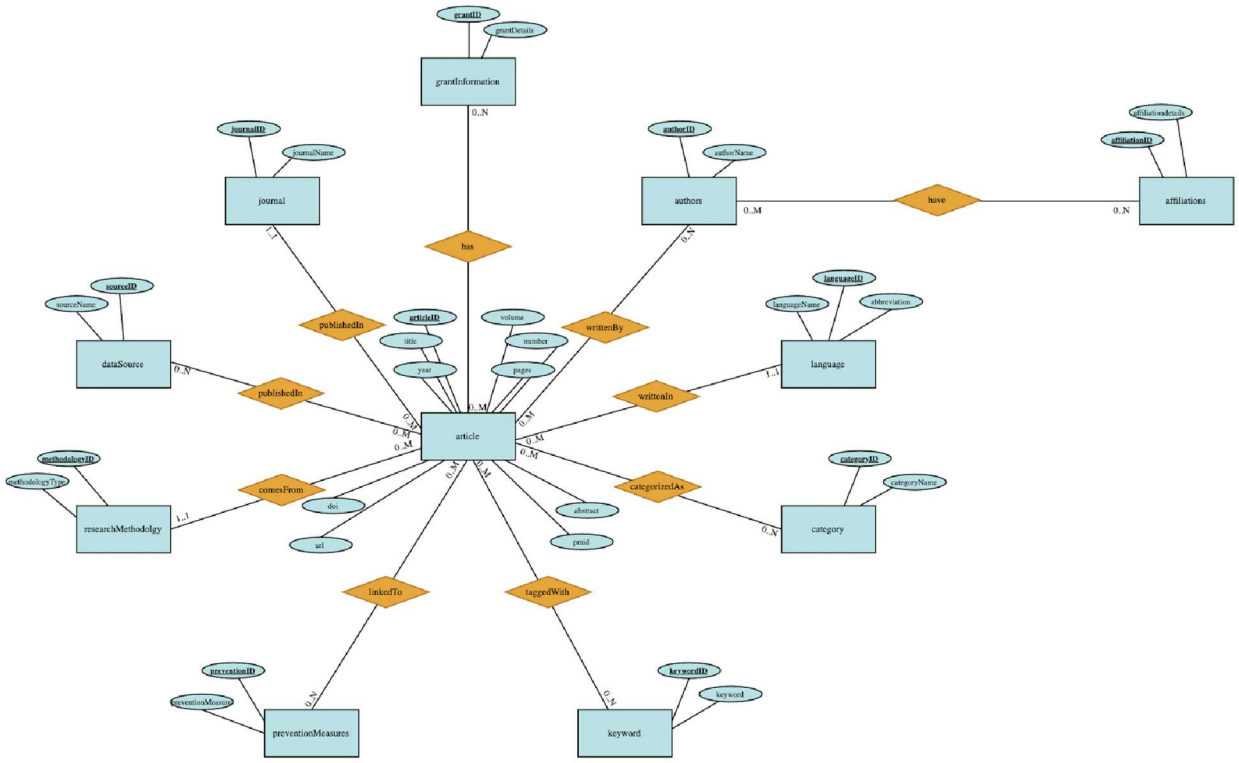
Entity	**Key Attributes**	**Relationships**

article	articleID, title, publicationYear, doi, url, volume, number, pages, abstract, pmid, languageID, journalID, researchMethodologyID, dataSourceID, suggestedPreventionID, grantID	📌 written by author (M:N) Via articleAuthor 📌 published in journal (M:1) 📌 written in language (M:1) 📌 categorized under category (M:N) via articleCategory 📌 tagged with keyword (M:N) via articleKeyword 📌 associated with affiliation (M:N) via articleAffliation 📌 funded by grant (M:1) 📌 uses research methodology (M:1) sourced from data source (M:1) includes suggested prevention measures (M:1) 📌 affiliated with institution via authorAffiliation (M:N)
authorAffiliation	authorID, affiliationID	📌 linked author and affiliation (for M:N relationship)
journal	journalID, journalName	📌 publishes article (1:M)
author(s)	authorID, authorName	📌 writes article (M:N) via articleAuthor
articleAuthor	articleID, authorID	📌 links article and author (for M:N relationship)
language	languageID, abbreviation, languageName	📌 defines the language of an article (1:M)
category	categoryID, categoryName	📌 categorizes article (M:N) via articleCategory
articleCategory	articleID, categoryID	📌 links article and category (for M:N relationship)
keyword	keywordID, keyword	📌 tags article (M:N) via articleKeyword

articleKeyword	articleID, keywordID	📌links article and keyword (for M:N relationship)
affiliation	affiliationID, affiliationName	📌 associates article with affiliation (M:N) via authorAffiliation
grant	grantID, grantDetails	📌 funds article (M:1)
researchMethodology	researchMethodologyID, methodologyName	📌 describes research methodology of article (M:1)
dataSource	dataSourceID, sourceName	📌 identifies source of article (M:1)
preventionMeasures	measureID, measureDetails	📌 suggests prevention measures for article (M:1)

Conceptual ER Model Diagram

Fig 1.0: Conceptual ERD



RELATIONAL DATABASE SCHEMA DESIGN

Relational Model

The relational model is as follows:

1. **Journal** (JournalID, JournalName)

2. **Language** (LanguageID, LanguageName, Abbreviation)
3. **Article** (ArticleID, Title, Year, DOI, URL, Volume, Number, Pages, Abstract, Affiliation, PMID, *JournalID*, *LanguageID*)
    - *JournalID* is a Foreign Key referencing JournalID in Journal and NOT NULL
    - *LanguageID* is a Foreign Key referencing LanguageID in Language and NOT NULL
4. **Author** (AuthorID, AuthorName)
5. **AuthorArticle** (*AuthorID*, *ArticleID*)
    - *AuthorID* is a Foreign Key referencing AuthorID in Author and NOT NULL
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
6. **Category** (CategoryID, CategoryName)
7. **ArticleCategory** (*ArticleID*, *CategoryID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *CategoryID* is a Foreign Key referencing CategoryID in Category and NOT NULL
8. **DataSource** (SourceID, SourceName)
9. **ArticleDataSource** (*ArticleID*, *SourceID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *SourceID* is a Foreign Key referencing SourceID in DataSource and NOT NULL
10. **ResearchMethodology** (MethodologyID, MethodologyType)
11. **ArticleMethodology** (*ArticleID*, *MethodologyID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *MethodologyID* is a Foreign Key referencing MethodologyID in ResearchMethodology and NOT NULL
12. **GrantInformation** (GrantID, GrantDetails)
13. **ArticleGrant** (*ArticleID*, *GrantID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *GrantID* is a Foreign Key referencing GrantID in GrantInformation and NOT NULL
14. **PreventionMeasures** (PreventionID, PreventionMeasure)
15. **ArticlePrevention** (*ArticleID*, *PreventionID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *PreventionID* is a Foreign Key referencing PreventionID in PreventionMeasures and NOT NULL
16. **Keyword** (KeywordID, Keyword)
17. **ArticleKeyword** (*ArticleID*, *KeywordID*)
    - *ArticleID* is a Foreign Key referencing ArticleID in Article and NOT NULL
    - *KeywordID* is a Foreign Key referencing KeywordID in Keyword and NOT NULL

DATABASE IMPLEMENTATION

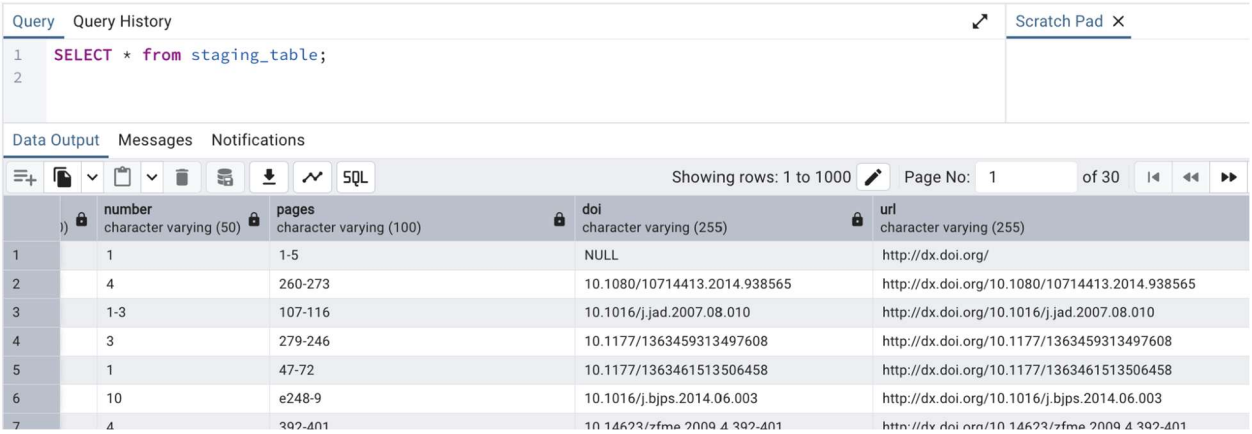
To implement the database in postgres we executed the following queries in SQL for each table (entity):

Step 1: Create staging_table and Load Data from CSV

The raw CSV data is first loaded into the **staging_table** without constraints, ensuring that all columns match the structure of the CSV file.

```
CREATE DATABASE IF NOT EXISTS SafetylitDB;
USE SafetylitDB;

-- create staging table to load raw csv data
CREATE TABLE staging_table(
    Title TEXT,
    Journal VARCHAR(255),
    Year INT,
    Volume VARCHAR(50),
    PMID VARCHAR(50),
    Number VARCHAR(50),
    Pages VARCHAR(100),
    DOI VARCHAR(255),
    URL VARCHAR(255),
    Abstract TEXT,
    Authors TEXT,
    Language VARCHAR(50),
    Affiliation TEXT,
    Categories TEXT,
    Keywords TEXT,
    Suggested_Suicide_Prevention_Measures TEXT,
    Grant_Information TEXT,
    Research_Methodology VARCHAR(100),
```

```
    Data_Source VARCHAR(100)
);
-- Import into staging
COPY staging_table
FROM '/Users/kobaid/Documents/Northeastern/IE 7945/ProjectDB/new/cleaned-latestdata-articles.csv'
WITH CSV HEADER ENCODING 'UTF8';
```



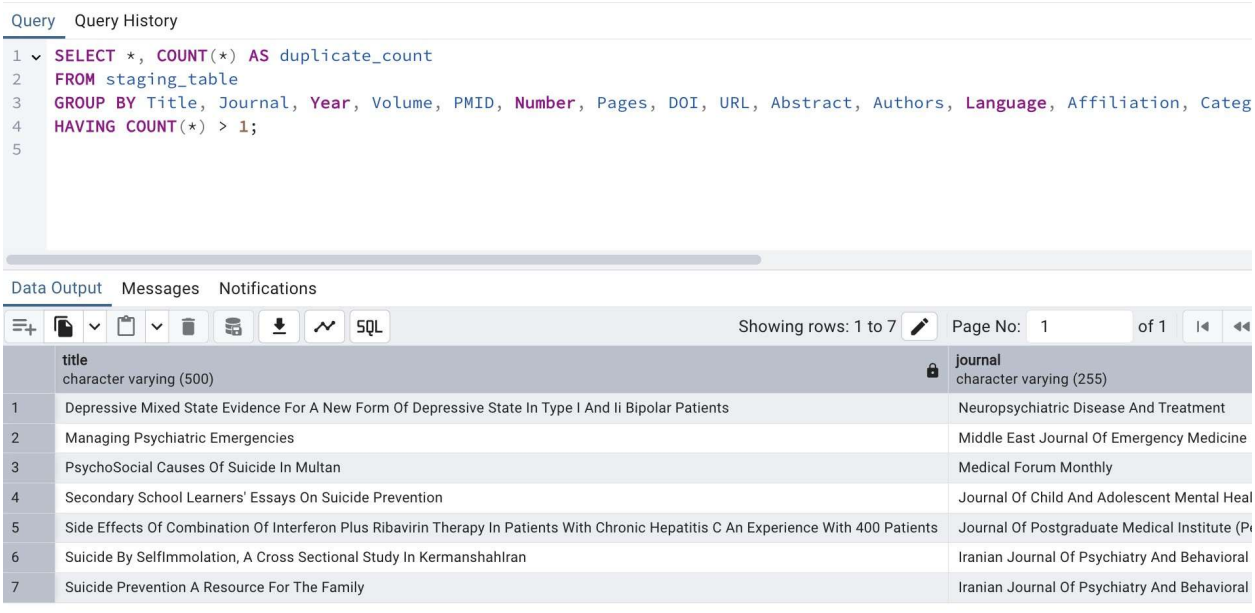
```
SELECT * from staging_table;
```

	number character varying (50)	pages character varying (100)	doi character varying (255)	url character varying (255)
1	1	1-5	NULL	http://dx.doi.org/
2	4	260-273	10.1080/10714413.2014.938565	http://dx.doi.org/10.1080/10714413.2014.938565
3	1-3	107-116	10.1016/j.jad.2007.08.010	http://dx.doi.org/10.1016/j.jad.2007.08.010
4	3	279-246	10.1177/1363459313497608	http://dx.doi.org/10.1177/1363459313497608
5	1	47-72	10.1177/1363461513506458	http://dx.doi.org/10.1177/1363461513506458
6	10	e248-9	10.1016/j.bjps.2014.06.003	http://dx.doi.org/10.1016/j.bjps.2014.06.003
7	4	392-401	10.14623/zfme.2009.4.392-401	http://dx.doi.org/10.14623/zfme.2009.4.392-401

- This staging table serves as an **intermediate layer** before normalization. ● The data is stored in **denormalized form** to facilitate transformation.

Step 2: Data Validation Checks Before Normalization

1. Find Duplicate Rows



```
SELECT *, COUNT(*) AS duplicate_count
FROM staging_table
GROUP BY Title, Journal, Year, Volume, PMID, Number, Pages, DOI, URL, Abstract, Authors, Language, Affiliation, Categ
HAVING COUNT(*) > 1;
```

	title character varying (500)	journal character varying (255)
1	Depressive Mixed State Evidence For A New Form Of Depressive State In Type I And Ii Bipolar Patients	Neuropsychiatric Disease And Treatment
2	Managing Psychiatric Emergencies	Middle East Journal Of Emergency Medicine
3	PsychoSocial Causes Of Suicide In Multan	Medical Forum Monthly
4	Secondary School Learners' Essays On Suicide Prevention	Journal Of Child And Adolescent Mental Heal
5	Side Effects Of Combination Of Interferon Plus Ribavirin Therapy In Patients With Chronic Hepatitis C An Experience With 400 Patients	Journal Of Postgraduate Medical Institute (P
6	Suicide By SelfImmolation, A Cross Sectional Study In KermanshahIran	Iranian Journal Of Psychiatry And Behavioral
7	Suicide Prevention A Resource For The Family	Iranian Journal Of Psychiatry And Behavioral

- Duplicate records detected: The query identified multiple rows with the same Title, Journal, Year, and other attributes, indicating potential duplicate entries in the staging_table.
- Standard practice in this instance would be to remove/drop duplicates; but, we're keeping them based on instruction provided by Eric.
-

Step 3: Normalize Data into Relational Tables

The **staging table** is then used to populate the normalized tables following the **relational model**:

3.1 Create tables in DB

```
-- Create main tables without foreign keys first
CREATE TABLE Journal (
    JournalID SERIAL PRIMARY KEY,
    JournalName VARCHAR(255) NOT NULL
);

CREATE TABLE Language (
    LanguageID SERIAL PRIMARY KEY,
    LanguageName VARCHAR(100) NOT NULL,
    Abbreviation VARCHAR(10)
);

CREATE TABLE Author (
    AuthorID SERIAL PRIMARY KEY,
    AuthorName TEXT NOT NULL
);

CREATE TABLE Category (
    CategoryID SERIAL PRIMARY KEY,
    CategoryName TEXT NOT NULL
);

CREATE TABLE DataSource (
    SourceID SERIAL PRIMARY KEY,
    SourceName VARCHAR(255) NOT NULL
);

CREATE TABLE ResearchMethodology (
    MethodologyID SERIAL PRIMARY KEY,
    MethodologyType VARCHAR(255) NOT NULL
);
```

```
CREATE TABLE GrantInformation (
    GrantID SERIAL PRIMARY KEY,
    GrantDetails TEXT NOT NULL
);

CREATE TABLE PreventionMeasures (
    PreventionID SERIAL PRIMARY KEY,
    PreventionMeasure TEXT NOT NULL
);

CREATE TABLE Keyword (
    KeywordID SERIAL PRIMARY KEY,
    Keyword VARCHAR(255) NOT NULL
);

-- Create table with foreign keys
CREATE TABLE Article (
    ArticleID SERIAL PRIMARY KEY,
    Title TEXT NOT NULL,
    Year INT NOT NULL,
    DOI VARCHAR(255),
    URL VARCHAR(255),
    Volume VARCHAR(50),
    Number VARCHAR(50),
    Pages VARCHAR(100),
    Abstract TEXT,
    Affiliation TEXT,
    PMID VARCHAR(50),
    JournalID INT NOT NULL,
    LanguageID INT NOT NULL,
    CONSTRAINT fk_article_journal FOREIGN KEY (JournalID) REFERENCES
Journal(JournalID) ON DELETE CASCADE,
    CONSTRAINT fk_article_language FOREIGN KEY (LanguageID) REFERENCES
Language(LanguageID) ON DELETE CASCADE
);

-- Create many-to-many relationship tables
CREATE TABLE AuthorArticle (
    AuthorID INT NOT NULL,
    ArticleID INT NOT NULL,
    PRIMARY KEY (AuthorID, ArticleID),
    CONSTRAINT fk_authorarticle_author FOREIGN KEY (AuthorID) REFERENCES
Author(AuthorID) ON DELETE CASCADE,
    CONSTRAINT fk_authorarticle_article FOREIGN KEY (ArticleID) REFERENCES
```

```
Article(ArticleID) ON DELETE CASCADE
);

CREATE TABLE ArticleCategory (
   ArticleID INT NOT NULL,
   CategoryID INT NOT NULL,
   PRIMARY KEY (ArticleID, CategoryID),
   CONSTRAINT fk_articlecategory_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
   CONSTRAINT fk_articlecategory_category FOREIGN KEY (CategoryID) REFERENCES
Category(CategoryID) ON DELETE CASCADE
);

CREATE TABLE ArticleDataSource (
   ArticleID INT NOT NULL,
   SourceID INT NOT NULL,
   PRIMARY KEY (ArticleID, SourceID),
   CONSTRAINT fk_articledatasource_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
   CONSTRAINT fk_articledatasource_source FOREIGN KEY (SourceID) REFERENCES
DataSource(SourceID) ON DELETE CASCADE
);

CREATE TABLE ArticleMethodology (
   ArticleID INT NOT NULL,
   MethodologyID INT NOT NULL,
   PRIMARY KEY (ArticleID, MethodologyID),
   CONSTRAINT fk_articlemethodology_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
   CONSTRAINT fk_articlemethodology_method FOREIGN KEY (MethodologyID)
REFERENCES ResearchMethodology(MethodologyID) ON DELETE CASCADE
);

CREATE TABLE ArticleGrant (
   ArticleID INT NOT NULL,
   GrantID INT NOT NULL,
   PRIMARY KEY (ArticleID, GrantID),
   CONSTRAINT fk_articlegrant_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
   CONSTRAINT fk_articlegrant_grant FOREIGN KEY (GrantID) REFERENCES
GrantInformation(GrantID) ON DELETE CASCADE
);

CREATE TABLE ArticlePrevention (
```

```
    ArticleID INT NOT NULL,
    PreventionID INT NOT NULL,
    PRIMARY KEY (ArticleID, PreventionID),
    CONSTRAINT fk_articleprevention_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
    CONSTRAINT fk_articleprevention_prevention FOREIGN KEY (PreventionID)
REFERENCES PreventionMeasures(PreventionID) ON DELETE CASCADE
);

CREATE TABLE ArticleKeyword (
    ArticleID INT NOT NULL,
    KeywordID INT NOT NULL,
    PRIMARY KEY (ArticleID, KeywordID),
    CONSTRAINT fk_articlekeyword_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
    CONSTRAINT fk_articlekeyword_keyword FOREIGN KEY (KeywordID) REFERENCES
Keyword(KeywordID) ON DELETE CASCADE
);
```

3.2 Write ALTER Statements

```
ALTER TABLE Language ALTER COLUMN LanguageName DROP NOT NULL;
ALTER TABLE Article ALTER COLUMN LanguageID DROP NOT NULL;
```

3.3 Populate Main Data tables

```
-- Populate Journal Table
INSERT INTO Journal (JournalName)
SELECT DISTINCT Journal FROM staging_table WHERE Journal IS NOT NULL;
```

Query Query History

```
1  -- Populate Journal Table
2  INSERT INTO Journal (JournalName)
3  SELECT DISTINCT Journal FROM staging_table WHERE Journal IS NOT NULL;
4
5  Select * from Journal;
```

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No:

	journalid [PK] integer	journalname character varying (255)
1	1	Journal Of Zahedan University Of Medical Sciences And Health Services
2	2	Journal Of Healthcare Protection Management
3	3	Contemporary Review Of The Middle East
4	4	Chinese Journal Of Radiological Medicine And Protection
5	5	Journal Of Pediatric Nursing
6	6	Journal Of Astrobiology And Outreach

Everything looks good in this table so **no Data Validation needed.**

-- Populate Language Table
INSERT INTO Language (Abbreviation)
SELECT DISTINCT Language FROM staging_table WHERE Language IS NOT NULL;

Query Query History

```
1  -- Populate Language Table
2  INSERT INTO Language (Abbreviation)
3  SELECT DISTINCT Language FROM staging_table WHERE Language IS NOT NULL;
4
5  SELECT * FROM Language;
```

Data Output Messages Notifications

Showing rows: 1 to 74

	languageid [PK] integer	languagename character varying (100)	abbreviation character varying (10)
1	2	[null]	bs
2	3	[null]	sr
3	4	[null]	fr
4	5	[null]	sk
5	6	[null]	tr

3.3.1 Data Validation Check: Update LanguageName Based on Known Abbreviations

```
-- Trim any leading/trailing spaces from abbreviations
UPDATE Language
SET Abbreviation = TRIM(Abbreviation);

-- Update LanguageName based on abbreviation
UPDATE Language
SET LanguageName = CASE
    WHEN Abbreviation = 'bs' THEN 'Bosnian'
    WHEN Abbreviation = 'sr' THEN 'Serbian'
    WHEN Abbreviation = 'fr' THEN 'French'
    WHEN Abbreviation = 'sk' THEN 'Slovak'
    WHEN Abbreviation = 'tr' THEN 'Turkish'
    WHEN Abbreviation = 'en' THEN 'English'
    WHEN Abbreviation = 'uk' THEN 'Ukrainian'
    WHEN Abbreviation = 'fi' THEN 'Finnish'
    WHEN Abbreviation = 'bg' THEN 'Bulgarian'
    WHEN Abbreviation = 'ru' THEN 'Russian'
    WHEN Abbreviation = 'he' THEN 'Hebrew'
    WHEN Abbreviation = 'vi' THEN 'Vietnamese'
    WHEN Abbreviation = 'fa' THEN 'Persian'
    WHEN Abbreviation = 'hr' THEN 'Croatian'
    WHEN Abbreviation = 'is' THEN 'Icelandic'
    WHEN Abbreviation = 'ja' THEN 'Japanese'
    WHEN Abbreviation = 'cs' THEN 'Czech'
    WHEN Abbreviation = 'lt' THEN 'Lithuanian'
    WHEN Abbreviation = 'sv' THEN 'Swedish'
    WHEN Abbreviation = 'ro' THEN 'Romanian'
    WHEN Abbreviation = 'de' THEN 'German'
    WHEN Abbreviation = 'nl' THEN 'Dutch'
    WHEN Abbreviation = 'et' THEN 'Estonian'
    WHEN Abbreviation = 'ar' THEN 'Arabic'
    WHEN Abbreviation = 'th' THEN 'Thai'
    WHEN Abbreviation = 'no' THEN 'Norwegian'
    WHEN Abbreviation = 'ca' THEN 'Catalan'
    WHEN Abbreviation = 'pt' THEN 'Portuguese'
    WHEN Abbreviation = 'hu' THEN 'Hungarian'
    WHEN Abbreviation = 'pl' THEN 'Polish'
    WHEN Abbreviation = 'ko' THEN 'Korean'
    WHEN Abbreviation = 'el' THEN 'Greek'
    WHEN Abbreviation = 'zh' THEN 'Chinese'
```

```
    WHEN Abbreviation = 'it' THEN 'Italian'
    WHEN Abbreviation = 'es' THEN 'Spanish'
    WHEN Abbreviation = 'da' THEN 'Danish'
    WHEN Abbreviation = 'NA' THEN 'Unknown'
    ELSE 'Unknown'
END
WHERE LanguageName IS NULL OR LanguageName = 'Unknown';

-- Verify the update
SELECT * FROM Language ORDER BY Abbreviation;
```

```
49  -- Verify the update
50  SELECT * FROM Language ORDER BY Abbreviation;
51
```

Data Output | Messages | Notifications

	languageid [PK] integer	languagename character varying (100)	abbreviation character varying (10)
2	25	Arabic	ar
3	9	Bulgarian	bg
4	1	Bosnian	bs
5	28	Catalan	ca
6	17	Czech	cs
7	37	Danish	da
8	21	German	de
9	33	Greek	el
10	6	English	en
11	36	Spanish	es
12	24	Estonian	et
13	13	Persian	fa
14	8	Finnish	fi
15	3	French	fr
16	11	Hebrew	he
17	14	Croatian	hr
18	30	Hungarian	hu
19	15	Icelandic	is
20	35	Italian	it
21	16	Japanese	ja

```
-- Populate Author Table
INSERT INTO Author (AuthorName)
SELECT DISTINCT Authors FROM staging_table WHERE Authors IS NOT NULL;
```

Query | Query History

```
1  SELECT * FROM Author;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page No: 1 of 27

	authorid [PK] integer	authorname text
1	26850	Kwan, Y. K. and Ip, Wai Cheong
2	26851	Friend, H.
3	26852	Sela-Shayovitz, Revital
4	26853	Takahashi, Kohei and Morimura, Naoto and Sakamoto, Tetsuya and Nagashima, Hiroshi and Hirata, Masafumi
5	26854	Marneros, Andreas
6	26855	Zimmerman, G.M.
7	26856	McGuffin, P. and Perroud, N. and Uher, R. and Butler, A. and Aitchison, K. J. and Craig, Ian W. and Lewis, C. and Farmer, A.
8	26857	Stephen, RE

3.3.2 Data Validation Check: Split Authorname in Author table to get individual Authors

Create a New Table for Individual Authors. Since the Author table currently has **multiple authors in a single row**, we need to **normalize** the data. Split the AuthorName Column and Insert Individual Authors. Verify the data. Check if the authors are now individual entries:

```
-- Step 1: Create a New Table for Individual Authors
CREATE TABLE AuthorNormalized (
    AuthorID SERIAL PRIMARY KEY,
    AuthorName TEXT UNIQUE
);

-- Step 2: Split the AuthorName Column and Insert Individual Authors
INSERT INTO AuthorNormalized (AuthorName)
SELECT DISTINCT TRIM(unnest(string_to_array(AuthorName, ' and ')))
FROM Author;

-- Step 3: Verify the Data - Check if authors are now stored individually
SELECT * FROM AuthorNormalized ORDER BY AuthorName;
```

Query | Query History

```
1  -- Step 1: Create a New Table for Individual Authors
2  CREATE TABLE AuthorNormalized (
3      AuthorID SERIAL PRIMARY KEY,
4      AuthorName TEXT UNIQUE
5  );
6
7  -- Step 2: Split the AuthorName Column and Insert Individual Authors
8  INSERT INTO AuthorNormalized (AuthorName)
9  SELECT DISTINCT TRIM(unnest(string_to_array(AuthorName, ' and ')))
10 FROM Author;
11
12 -- Step 3: Verify the Data - Check if authors are now stored individually
13 SELECT * FROM AuthorNormalized ORDER BY AuthorName;
14
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000

	authorid [PK] integer	authorname text
1	43904	44th Congress of the International Psychoanalytical Association,
2	48091	A, Raharivelo
3	36078	A., Abdoli
4	59936	A., Ahmadi
5	61909	A., Ahmadvand
6	12203	A., Akbari
7	60286	A., Al Ansari
8	61723	A., Asif
9	56327	A., Charkazi

Now that we've verified that we have individual Authors like we want we can drop the Original Author table and rename this to

DROP TABLE IF EXISTS Author;

-- Rename AuthorNormalized to Author
ALTER TABLE AuthorNormalized RENAME TO Author;

SELECT * FROM Author;

Query Query History

```
1  SELECT * FROM Author;
```

Data Output Messages Notifications

	authorid [PK] integer	authorname text
1	1	Aydın, Irfan
2	2	Mello, Sueli Moreira
3	3	Hudson, V.M.
4	4	Fekete, Sandor
5	5	Friend, H.
6	6	Sela-Shayovitz, Revital
7	7	Karagöz, Hatice
8	8	Schartz, Anne
9	9	Hishinuma, Earl S.

-- Populate Category Table
INSERT INTO Category (CategoryName)
SELECT DISTINCT Categories FROM staging_table WHERE Categories IS NOT NULL;

Query Query History

```
1  SELECT * FROM Category;
```

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No: 1 of 3

	categoryid [PK] integer	categoryname text
1	1	Economics of Injury and Safety, PTSD, Injury Outcomes, Ergonomics, Human Factors, Anthropometrics, Physiology, Recreational and Sports Issues, TBI
2	2	Economics of Injury and Safety, PTSD, Injury Outcomes, Recreational and Sports Issues
3	3	Economics of Injury and Safety, PTSD, Injury Outcomes, Recreational and Sports Issues, Suicide and Self-Harm, Violence and Weapons Issues
4	4	Occupational Issues, Suicide and Self-Harm, Transportation Issues, Violence and Weapons Issues
5	5	Program and Other Evaluations, Effectiveness Studies, Risk Factor Prevalence, Injury Occurrence, Social Etiologies and Disparities, Suicide and Self-Harm

3.3.3 Data Validation Check: Split CategoryName into Individual Categories

First extract distinct categories from the **staging table**, then temporarily stores split values before **clearing and repopulating the Category table** with unique entries. Finally, the temporary table is removed, leaving a **clean, structured dataset** ready for further relational mapping.

```
-- Data Validation - Category table
-- Step 1: Create a Temporary Table to Store Individual Categories
CREATE TEMP TABLE TempCategory (
    CategoryName TEXT UNIQUE
);

-- Step 2: Extract & Insert Individual Categories into TempCategory
INSERT INTO TempCategory (CategoryName)
SELECT DISTINCT TRIM(unnest(string_to_array(CategoryName, ',')))
FROM Category;

-- Step 3: Clear the Existing Category Table
TRUNCATE TABLE Category RESTART IDENTITY;

-- Step 4: Insert Normalized Categories Back into Category Table
INSERT INTO Category (CategoryName)
SELECT CategoryName FROM TempCategory;

-- Step 5: Drop Temporary Table
DROP TABLE TempCategory;

-- Verify That Categories Are Now Individual Entries
SELECT * FROM Category ORDER BY CategoryName;
```

Query Query History

```
1  -- Verify That Categories Are Now Individual Entries
2  SELECT * FROM Category ORDER BY CategoryName;
```

Data Output Messages Notifications

SQL

	categoryid [PK] integer	categoryname text
1	67	Age: Adolescents
2	18	Age: Elder Adults
3	56	Age: Infants and Children
4	69	Age: Young Adults
5	25	Alcohol and Other Drugs
6	17	Anthropometrics
7	41	Burns
8	57	Chronobiology
9	43	Climate
10	8	Commentary
11	53	Community-Based Prevention

-- Populate Keyword Table
INSERT INTO Keyword (Keyword)
SELECT DISTINCT Keywords FROM staging_table WHERE Keywords IS NOT NULL;

```
6   SELECT * FROM Keyword;
7
```

Data Output | Messages | Notifications

	keywordid [PK] integer	keyword character varying (255)
1	1	help, self, %, harm, students
2	2	death, deaths, cause, time, 1980s
3	3	people, self-harm, article, focus, groups
4	4	risk, treatment, patients, behaviour, borderline
5	5	health, care, illness, transitions, article
6	6	weaving, course, silk, people, training
7	7	suicide, risk, trauma, article, literature
8	8	suicide, ska2, variation, cohorts, gene

3.3.4 Data Validation Check: Split Keyword into Individual Keywords

Since keywords in the Keyword table are separated by commas, this process involves extracting distinct keyword values, splitting them into separate rows, and normalizing the data. A temporary table is used to store unique keywords before repopulating the main Keyword table. This step enhances data integrity and consistency, making it easier to query and analyze individual keywords efficiently.

```
-- Data Validation - Keyword table
-- Step 1: Create a Temporary Table to Store Individual Keywords
CREATE TEMP TABLE TempKeyword (
    Keyword TEXT UNIQUE
);

-- Step 2: Extract & Insert Individual Keywords into TempKeyword
INSERT INTO TempKeyword (Keyword)
SELECT DISTINCT TRIM(unnest(string_to_array(Keyword, ',')))
FROM Keyword;

-- Step 3: Clear the Existing Keyword Table
TRUNCATE TABLE Keyword RESTART IDENTITY;
```

-- Step 4: Insert Normalized Keywords Back into Keyword Table
INSERT INTO Keyword (Keyword)
SELECT Keyword FROM TempKeyword;

-- Step 5: Drop Temporary Table
DROP TABLE TempKeyword;

-- Verify That Keywords Are Now Individual Entries
SELECT * FROM Keyword ORDER BY Keyword;

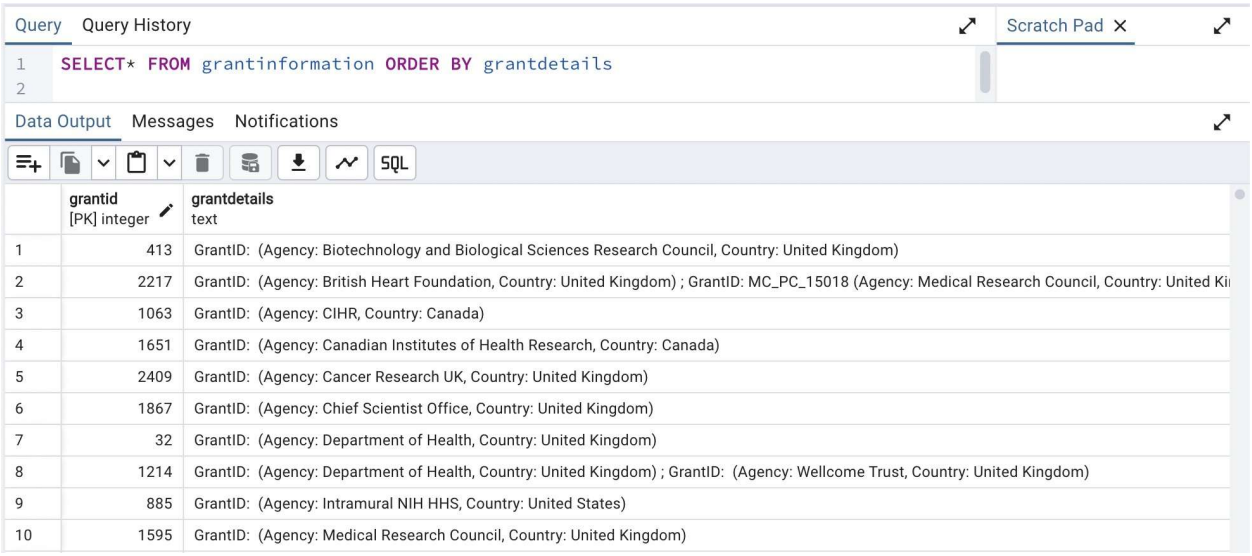
```
26  -- Verify That Keywords Are Now Individual Entries
27  SELECT * FROM Keyword ORDER BY Keyword;
28
```

Data Output | Messages | Notifications

SQL | Sh

	keywordid [PK] integer	keyword character varying (255)
311	8004	ageing
312	6424	agencies
313	3177	agency
314	123	agenda
315	8180	agenesis
316	5153	agent
317	8816	agent(s
318	9150	agents
319	9290	ages
320	1546	agg
321	1752	aggregation

-- Populate GrantInformation Table
INSERT INTO GrantInformation (GrantDetails)
SELECT DISTINCT Grant_Information FROM staging_table WHERE Grant_Information IS NOT NULL;



Query | Query History | Scratch Pad

```
SELECT* FROM grantinformation ORDER BY grantdetails
```

Data Output | Messages | Notifications

	grantid [PK] integer	grantdetails text
1	413	GrantID: (Agency: Biotechnology and Biological Sciences Research Council, Country: United Kingdom)
2	2217	GrantID: (Agency: British Heart Foundation, Country: United Kingdom) ; GrantID: MC_PC_15018 (Agency: Medical Research Council, Country: United Ki
3	1063	GrantID: (Agency: CIHR, Country: Canada)
4	1651	GrantID: (Agency: Canadian Institutes of Health Research, Country: Canada)
5	2409	GrantID: (Agency: Cancer Research UK, Country: United Kingdom)
6	1867	GrantID: (Agency: Chief Scientist Office, Country: United Kingdom)
7	32	GrantID: (Agency: Department of Health, Country: United Kingdom)
8	1214	GrantID: (Agency: Department of Health, Country: United Kingdom) ; GrantID: (Agency: Wellcome Trust, Country: United Kingdom)
9	885	GrantID: (Agency: Intramural NIH HHS, Country: United States)
10	1595	GrantID: (Agency: Medical Research Council, Country: United Kingdom)

3.3.5 Data Validation: Remove Duplicates in Grant Information table

```
-- Data Validation - GrantInformation table
-- Step 1: Create a Temporary Table to Store Unique Grant Details
CREATE TEMP TABLE TempGrantInformation (
    GrantDetails TEXT UNIQUE
);

-- Step 2: Insert Distinct Grant Details into Temp Table
INSERT INTO TempGrantInformation (GrantDetails)
SELECT DISTINCT GrantDetails FROM GrantInformation;

-- Step 3: Truncate the Existing GrantInformation Table
TRUNCATE TABLE GrantInformation RESTART IDENTITY;

-- Step 4: Insert Normalized Grant Details Back into the Table
INSERT INTO GrantInformation (GrantDetails)
SELECT GrantDetails FROM TempGrantInformation;

-- Step 5: Drop Temporary Table
DROP TABLE TempGrantInformation;
```

```
-- Populate ResearchMethodology Table
INSERT INTO ResearchMethodology (MethodologyType)
SELECT DISTINCT Research_Methodology FROM staging_table WHERE
Research_Methodology IS NOT NULL;
```

```
5   SELECT * FROM ResearchMethodology;
```

Data Output | Messages | Notifications

	methodologyid [PK] integer	methodologytype character varying (255)
1	1	Quantitative
2	2	Qualitative

No data validation required.

-- Populate PreventionMeasures Table
INSERT INTO PreventionMeasures (PreventionMeasure)
SELECT DISTINCT Suggested_Suicide_Prevention_Measures FROM staging_table WHERE Suggested_Suicide_Prevention_Measures IS NOT NULL;

```
6   SELECT * FROM PreventionMeasures;
```

Data Output | Messages | Notifications

	preventionid [PK] integer	preventionmeasure text
1	1	Crisis intervention, Counseling, Awareness programs
2	2	Counseling, Awareness programs
3	3	Counseling
4	4	Mental health services, Support groups, Counseling
5	5	Crisis intervention, Therapy, Awareness programs
6	6	Mental health services, Crisis intervention, Counseling, Awareness programs
7	7	Therapy, Counseling, Crisis intervention
8	8	Crisis intervention, Awareness programs
9	9	Mental health services, Crisis intervention, Therapy, Awareness programs

3.3.6 Data Validation Check: Split Prevention Measures

```
-- Data Validation - Prevention Measures
-- Step 1: Create a Temporary Table to Store Individual Prevention Measures
CREATE TEMP TABLE TempPreventionMeasures (
    PreventionMeasure TEXT UNIQUE
);

-- Step 2: Extract & Insert Individual Prevention Measures
INSERT INTO TempPreventionMeasures (PreventionMeasure)
SELECT DISTINCT TRIM(unnest(string_to_array(PreventionMeasure, ',')))
FROM PreventionMeasures;

-- Step 3: Clear the Existing PreventionMeasures Table
TRUNCATE TABLE PreventionMeasures RESTART IDENTITY;

-- Step 4: Insert Normalized Prevention Measures Back into the Table
INSERT INTO PreventionMeasures (PreventionMeasure)
SELECT PreventionMeasure FROM TempPreventionMeasures;

-- Step 5: Drop Temporary Table
DROP TABLE TempPreventionMeasures;
```

```
21  -- Verify That Prevention Measures Are Now Individual Entries
22  SELECT * FROM PreventionMeasures ORDER BY PreventionMeasure;
23
24
```

Data Output | Messages | Notifications

Showing row

	preventionid [PK] integer	preventionmeasure text
1	11	Awareness programs
2	4	Counseling
3	8	Crisis intervention
4	9	Hotline support
5	1	Implement comprehensive suicide prevention programs including crisis intervention
6	7	Mental health services
7	6	Support groups
8	3	Therapy
9	2	and public awareness campaigns
10	5	counseling
11	10	mental health support

-- Populate DataSource Table
INSERT INTO DataSource (SourceName)
SELECT DISTINCT Data_Source FROM staging_table WHERE Data_Source IS NOT NULL;

```
5   SELECT * FROM DataSource;
```

Data Output | Messages | Notifications

	sourceid [PK] integer	sourcename character varying (255)
1	1	Online Publication
2	2	Academic Journal
3	3	News Media
4	4	Multiple Sources
5	5	Conference Proceedings
6	6	Online Publication
7	7	Academic Journal
8	8	News Media
9	9	Multiple Sources
10	10	Conference Proceedings

There are some duplicates so we need to fix those for data validation.

3.3.7 Data Validation Check: Remove Duplicates from Data Sources Table

```
-- Step 1: Create a Temporary Table to Store Unique Data Sources
CREATE TEMP TABLE TempDataSource (
    SourceName TEXT UNIQUE
);

-- Step 2: Insert Unique Source Names into the Temporary Table
INSERT INTO TempDataSource (SourceName)
SELECT DISTINCT SourceName FROM DataSource;

-- Step 3: Clear the Existing DataSource Table
```

TRUNCATE TABLE DataSource RESTART IDENTITY;

-- Step 4: Insert Normalized Data Sources Back into DataSource Table
INSERT INTO DataSource (SourceName)
SELECT SourceName FROM TempDataSource;

-- Step 5: Drop Temporary Table
DROP TABLE TempDataSource;

```
20  -- Verify That Data Sources Are Now Unique
21  SELECT * FROM DataSource ORDER BY SourceName;
22
```

Data Output | Messages | Notifications

	sourceid [PK] integer	sourcename character varying (255)
1	5	Academic Journal
2	2	Conference Proceedings
3	3	Multiple Sources
4	1	News Media
5	4	Online Publication

-- Populate Article Table
INSERT INTO Article (Title, Year, DOI, URL, Volume, Number, Pages, Abstract, Affiliation, PMID, JournalID, LanguageID)
SELECT s.Title, s.Year, s.DOI, s.URL, s.Volume, s.Number, s.Pages, s.Abstract, s.Affiliation, s.PMID,
    j.JournalID, l.LanguageID
FROM staging_table s
LEFT JOIN Journal j ON s.Journal = j.JournalName
LEFT JOIN Language l ON s.Language = l.LanguageName;

year integer	doi character varying (255)	url character varying (255)	volume character varying (50)	number character varying (50)	p c
2013	10.1016/j.medleg.2012.10.001	http://dx.doi.org/10.1016/j.medleg.2012.10.001	4	1	3
2011	10.1037/a0022231	http://dx.doi.org/10.1037/a0022231	31	3	1
2013	NA	http://dx.doi.org/	33	1	4
2010	NA	http://dx.doi.org/	38	1	1
2011	10.1177/1363461510383178	http://dx.doi.org/10.1177/1363461510383178	48	02-Jan	3
2014	10.1017/S0032247413000041	http://dx.doi.org/10.1017/S0032247413000041	50	2	1
2006	10.1097/01.JGP.0000196631.65375.62	http://dx.doi.org/10.1097/01.JGP.0000196631.65375.62	14	4	3
2012	10.1080/03066150.2011.653344	http://dx.doi.org/10.1080/03066150.2011.653344	39	5	1
2012	10.1136/injuryprev-2012-040590n.7	http://dx.doi.org/10.1136/injuryprev-2012-040590n.7	18	Suppl 1	A

It looks like there might be **some data that needs to be cleaned** but this is something we're looking to do **in Python** in this case as we will need to dig deeper for this table.

3.4 Populate Many to Many Relationship Tables

```
-- Populate Many-to-Many Relationship Tables
INSERT INTO AuthorArticle (AuthorID, ArticleID)
SELECT DISTINCT a.AuthorID, art.ArticleID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN Author a ON a.AuthorName = s.Authors
WHERE s.Authors IS NOT NULL;

INSERT INTO ArticleCategory (ArticleID, CategoryID)
SELECT DISTINCT art.ArticleID, c.CategoryID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN Category c ON c.CategoryName = s.Categories
WHERE s.Categories IS NOT NULL;

INSERT INTO ArticleKeyword (ArticleID, KeywordID)
SELECT DISTINCT art.ArticleID, k.KeywordID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN Keyword k ON k.Keyword = s.Keywords
WHERE s.Keywords IS NOT NULL;

INSERT INTO ArticleGrant (ArticleID, GrantID)
SELECT DISTINCT art.ArticleID, g.GrantID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN GrantInformation g ON g.GrantDetails = s.Grant_Information
WHERE s.Grant_Information IS NOT NULL;
```

```
INSERT INTO ArticleMethodology (ArticleID, MethodologyID)
SELECT DISTINCT art.ArticleID, rm.MethodologyID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN ResearchMethodology rm ON rm.MethodologyType = s.Research_Methodology
WHERE s.Research_Methodology IS NOT NULL;

INSERT INTO ArticlePrevention (ArticleID, PreventionID)
SELECT DISTINCT art.ArticleID, pm.PreventionID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN PreventionMeasures pm ON pm.PreventionMeasure =
s.Suggested_Suicide_Prevention_Measures
WHERE s.Suggested_Suicide_Prevention_Measures IS NOT NULL;

INSERT INTO ArticleDataSource (ArticleID, SourceID)
SELECT DISTINCT art.ArticleID, ds.SourceID
FROM staging_table s
JOIN Article art ON s.Title = art.Title
JOIN DataSource ds ON ds.SourceName = s.Data_Source
WHERE s.Data_Source IS NOT NULL;
```

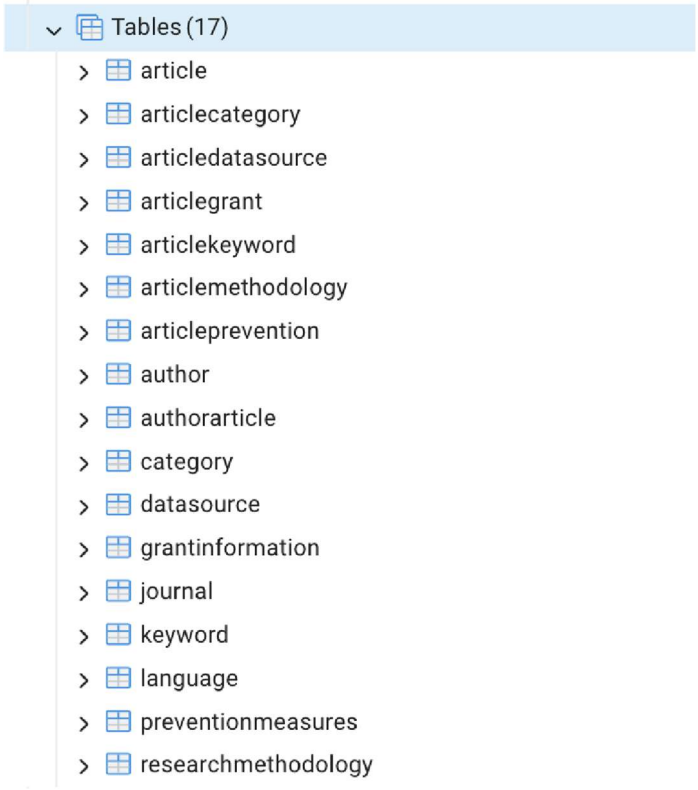
3.5 Add Foreign Keys

```
-- Add Foreign Key Constraints
ALTER TABLE Article
ADD CONSTRAINT fk_article_journal FOREIGN KEY (JournalID) REFERENCES
Journal(JournalID) ON DELETE CASCADE,
ADD CONSTRAINT fk_article_language FOREIGN KEY (LanguageID) REFERENCES
Language(LanguageID) ON DELETE CASCADE;

ALTER TABLE AuthorArticle
ADD CONSTRAINT fk_authorarticle_author FOREIGN KEY (AuthorID) REFERENCES
Author(AuthorID) ON DELETE CASCADE,
ADD CONSTRAINT fk_authorarticle_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE;

ALTER TABLE ArticleCategory
ADD CONSTRAINT fk_articlecategory_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articlecategory_category FOREIGN KEY (CategoryID) REFERENCES
Category(CategoryID) ON DELETE CASCADE;
```

```
ALTER TABLE ArticleDataSource
ADD CONSTRAINT fk_articledatasource_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articledatasource_source FOREIGN KEY (SourceID) REFERENCES
DataSource(SourceID) ON DELETE CASCADE;

ALTER TABLE ArticleMethodology
ADD CONSTRAINT fk_articlemethodology_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articlemethodology_method FOREIGN KEY (MethodologyID)
REFERENCES ResearchMethodology(MethodologyID) ON DELETE CASCADE;

ALTER TABLE ArticleGrant
ADD CONSTRAINT fk_articlegrant_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articlegrant_grant FOREIGN KEY (GrantID) REFERENCES
GrantInformation(GrantID) ON DELETE CASCADE;

ALTER TABLE ArticlePrevention
ADD CONSTRAINT fk_articleprevention_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articleprevention_prevention FOREIGN KEY (PreventionID)
REFERENCES PreventionMeasures(PreventionID) ON DELETE CASCADE;

ALTER TABLE ArticleKeyword
ADD CONSTRAINT fk_articlekeyword_article FOREIGN KEY (ArticleID) REFERENCES
Article(ArticleID) ON DELETE CASCADE,
ADD CONSTRAINT fk_articlekeyword_keyword FOREIGN KEY (KeywordID) REFERENCES
Keyword(KeywordID) ON DELETE CASCADE;
```

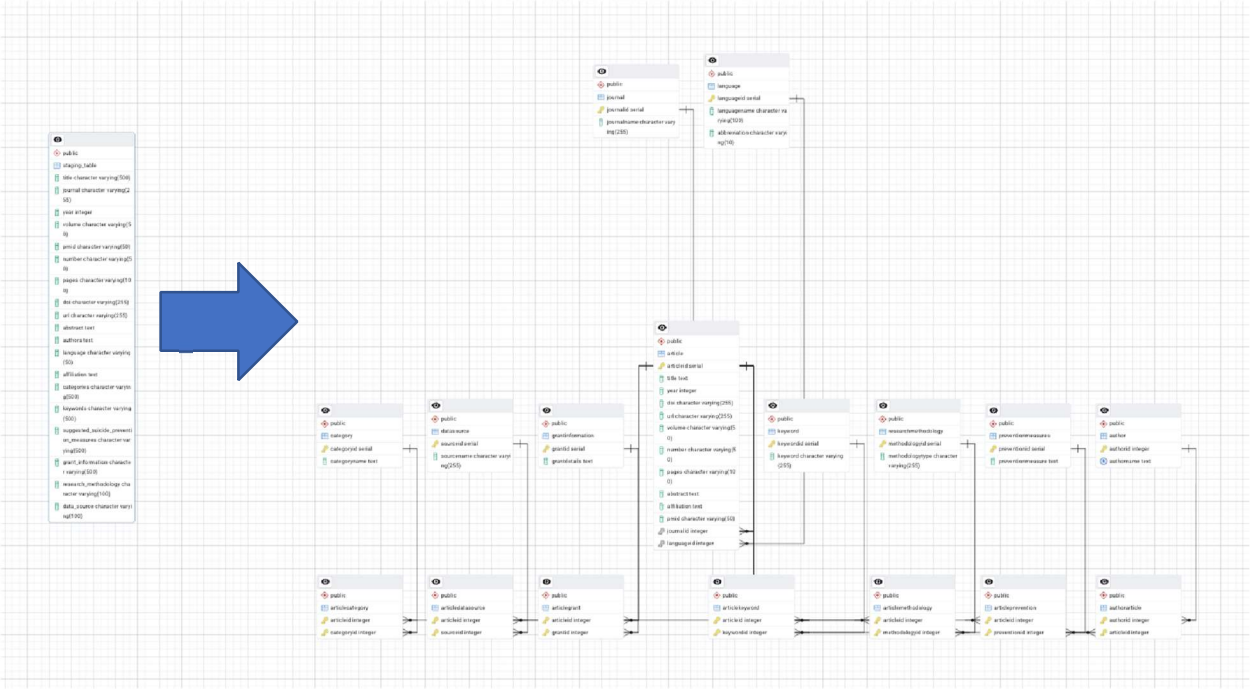
3.6 DROP Staging_table

```
DROP TABLE IF EXISTS staging_table;
```



All tables are loaded and validated successfully!

Fig2.1: Before and After Normalization ERD



Data Querying in SQL

Once the data is validated and cleaned, querying it for insights and relationships is essential.

1. **Show count of all records in all table in database**

Show count of all records in all tables in the database.

Query Query History

```
30  UNION ALL
31  SELECT 'articlemethodology', COUNT(*) FROM articlemethodology
```

Data Output Messages Notifications

	table_name text	count bigint
1	language	37
2	category	74
3	preventionmeasures	11
4	researchmethodology	2
5	datasource	5
6	grantinformation	2431
7	journal	4951
8	keyword	10820
9	affiliation	14405
10	articlegrant	29005
11	articledatasource	29031
12	articlemethodology	29043
13	articlecategory	65356
14	article	29005
15	author	67796
16	articleprevention	95733
17	articlekeyword	135241
18	authorarticle	111775
19	authoraffiliation	99301

2. Retrieve All Articles with Author Details

Displays articles along with their authors.

Query Query History

```
1  SELECT a.ArticleID, a.Title, au.AuthorName
2  FROM Article a
3  JOIN AuthorArticle aa ON a.ArticleID = aa.ArticleID
4  JOIN Author au ON aa.AuthorID = au.AuthorID
5  ORDER BY a.Title;
6
```

Data Output Messages Notifications

Showing rows: 1 to 1000 Page No

	articleid integer	title text
1	22900	I Have Always Thought About Death, A Death That I Might Give Myself
2	27042	"MurderSuicide A Review Of The Recent Literature' Comment' Reply
3	22970	"What Are You" A Recurring Question In A CrossCultural Psychiatrist'S Life And Career
4	22972	'A Life Wasted Making Dust' Affective Histories Of Dearth, Death, Debt And Farmers' Suicides In India
5	12763	'A Long Journey Starts With A Single Step'Waimakariri'S Journey Into Suicide Prevention
6	22973	'A Man Can Be Destroyed But Not Defeated' Ernest Hemingway'S NearDeath Experience And Declining Health

3. Find the Most Common Keywords in Articles

Identifies frequently occurring keywords.

Query Query History

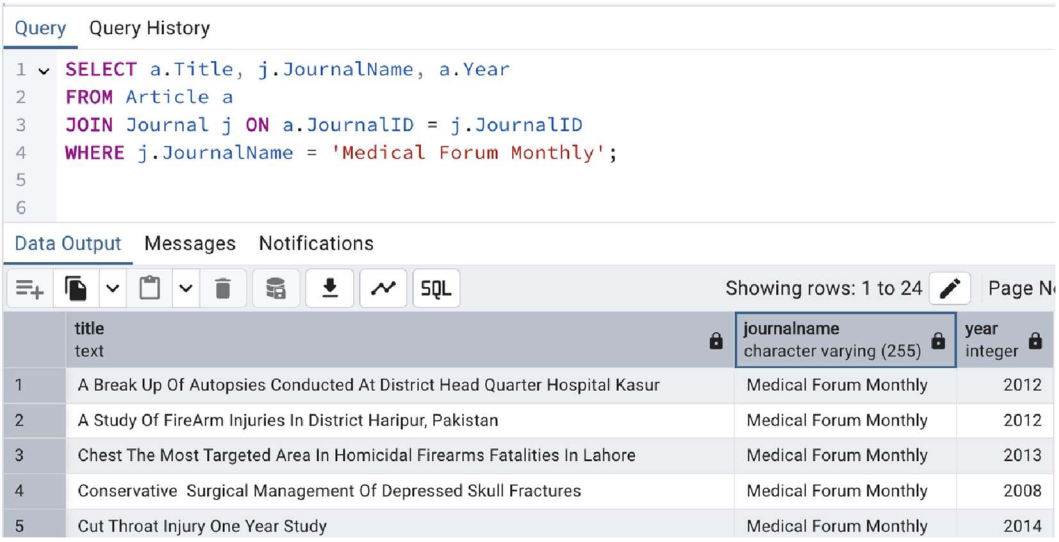
```
1  SELECT
2      k.Keyword,
3      COUNT(ak.ArticleID) AS frequency
4  FROM
5      ArticleKeyword ak
6  JOIN
7      Keyword k ON ak.KeywordID = k.KeywordID
8  GROUP BY
9      k.Keyword
10 ORDER BY
11     frequency DESC;
```

Data Output Messages Notifications

	keyword character varying (255)	frequency bigint
1	suicide	9209
2	language	4238
3	p>[abstract	4166
4	%	3786
5	patients	2887
6	risk	2391
7	self	1909
8	unavailable]</p	1818
9	health	1616
10	depression	1541
11	study	1336
12	treatment	1185

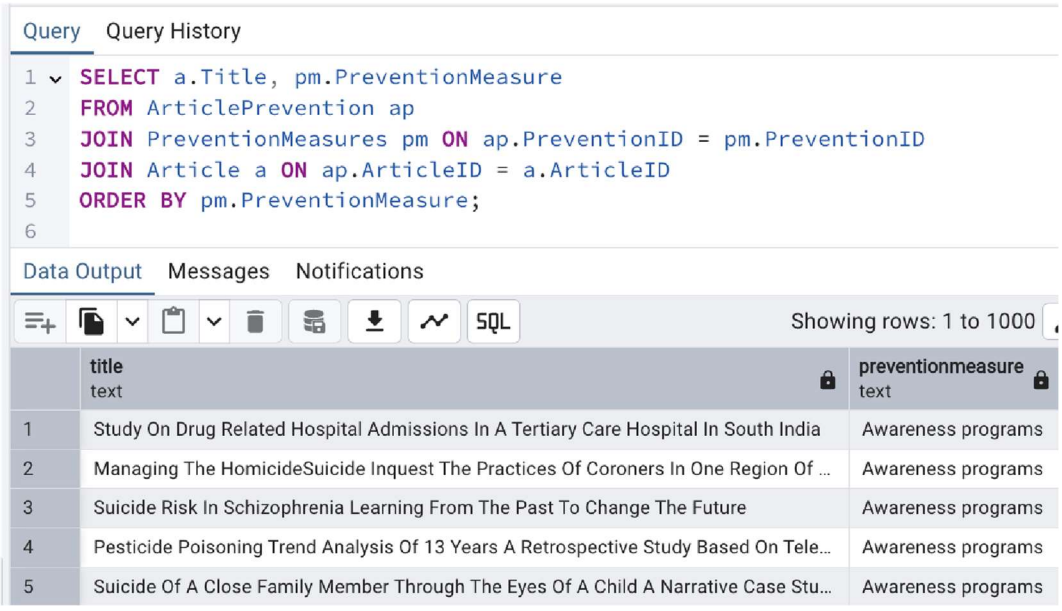
4. Get Articles Published in a Specific Journal

Finds all articles published in a given journal.



5. List Articles by Prevention Measures

Retrieves articles categorized under a given prevention measure.



6. Find Authors Who Have Published the Most Articles

Ranks authors based on the number of articles they have contributed to.

```sql
SELECT
    au.AuthorName,
    COUNT(aa.ArticleID) AS total_articles
FROM
    AuthorArticle aa
JOIN
    Author au ON aa.AuthorID = au.AuthorID
GROUP BY
    au.AuthorName
ORDER BY
    total_articles DESC;
```

Data Output | Messages | Notifications

	authorname text	total_articles bigint
1	Lester, David	252
2	Hawton, Keith E.	185
3	Gunnell, David	152
4	Mann, J. John	138
5	Sher, Leo	135
6	Pompili, Maurizio	128
7	Oquendo, Maria A.	124
8	De Leo, Diego	121
9	Girardi, Paolo	113
10	Turecki, Gustavo	106
11	Kapur, Navneet	97
12	Joiner, Thomas E.	92

7. Get Articles Published Over Time

Displays how many articles were published per year.

Query Query History

```
1 SELECT Year, COUNT(*) AS total_articles
2 FROM Article
3 GROUP BY Year
4 ORDER BY Year;
5
```

Data Output Messages Notifications

SQL

	year integer	total_articles bigint
1	2005	2306
2	2006	2702
3	2007	2669
4	2008	2767
5	2009	2881
6	2010	3015
7	2011	3033
8	2012	2907
9	2013	2933
10	2014	3807

8. Find the Most Common Research Methodologies Used

Analyzes the types of research methodologies employed in studies.

Query Query History

```
2 FROM ArticleMethodology am
3 JOIN ResearchMethodology rm ON am.MethodologyID = rm.MethodologyID
4 GROUP BY rm.MethodologyType
5 ORDER BY frequency DESC;
6
```

Data Output Messages Notifications

SQL Showing rows: 1 t

	methodologytype character varying (255)	frequency bigint
1	Qualitative	25120
2	Quantitative	3943

9. Identify the Most Cited Articles

Retrieves the top 10 articles with the highest number of PubMed citations (PMID), assuming that articles with a PMID are more frequently cited.

Query | Query History

```
1  SELECT Title, PMID, Year, JournalID
2  FROM Article
3  WHERE PMID IS NOT NULL
4  ORDER BY PMID DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1000 | Page

	title text	pmid character varying (50)	year integer	journalid integer
332	Lilya 4Ever. PostSoviet Neoliberal Angels And Nordic Intellectual...	unavailable	2014	377
333	Controlling For Known Risk Factors For Suicide In Multivariate ...	unavailable	2010	1533
334	Controversy Over Fatal Monointoxication With Flunitrazepam	unavailable	2009	905
335	Lille Taking Up The Challenge Of The Quality Of Care	unavailable	2005	3987
336	Like Red Tulips At Springtime Understanding The Absence Of Fe...	unavailable	2010	1300

10. Find the Most Common Data Sources for Research

Shows the most frequently used data sources in articles.

Query | Query History

```
1  SELECT ds.SourceName, COUNT(ad.ArticleID) AS frequency
2  FROM ArticleDataSource ad
3  JOIN DataSource ds ON ad.SourceID = ds.SourceID
4  GROUP BY ds.SourceName
5  ORDER BY frequency DESC;
6
7
8
```

Data Output | Messages | Notifications

	sourcename character varying (255)	frequency bigint
1	Multiple Sources	28155
2	Academic Journal	414
3	News Media	231
4	Conference Proceedings	139
5	Online Publication	109

11. Identify the Most Popular Research Topics

Determines which research categories have the most articles.

Query Query History

```
1  SELECT c.CategoryName, COUNT(ac.ArticleID) AS article_count
2  FROM ArticleCategory ac
3  JOIN Category c ON ac.CategoryID = c.CategoryID
4  GROUP BY c.CategoryName
5  ORDER BY article_count DESC;
6
```

Data Output Messages Notifications

SQL Showin

	categoryname text	article_count bigint
1	Non-Categorized	12625
2	Suicide and Self-Harm	4693
3	Violence and Weapons Issues	189
4	Poisoning	145
5	Not Available	117
6	Alcohol and Other Drugs	30

12. Find the Relationship Between Suicide Prevention Measures and Research Methodologies

Shows how different suicide prevention measures are associated with various research methodologies.

Query Query History

```
1  SELECT pm.PreventionMeasure, rm.MethodologyType, COUNT(am.ArticleID) AS count
2  FROM ArticlePrevention ap
3  JOIN PreventionMeasures pm ON ap.PreventionID = pm.PreventionID
4  JOIN ArticleMethodology am ON ap.ArticleID = am.ArticleID
5  JOIN ResearchMethodology rm ON am.MethodologyID = rm.MethodologyID
6  GROUP BY pm.PreventionMeasure, rm.MethodologyType
7  ORDER BY count DESC;
8
```

Data Output Messages Notifications

SQL Showing rows: 1 to 14 Page

	preventionmeasure text	methodologytype character varying (255)	count bigint
1	Crisis intervention	Qualitative	1886
2	Mental health services	Qualitative	1612
3	Therapy	Qualitative	1107
4	Crisis intervention	Quantitative	465
5	Mental health services	Quantitative	428
6	Awareness programs	Qualitative	250
7	Therapy	Quantitative	119
8	Counseling	Qualitative	51
9	Awareness programs	Quantitative	46
10	Counseling	Quantitative	18
11	Support groups	Qualitative	17
12	Hotline support	Qualitative	6
13	Support groups	Quantitative	3
14	Hotline support	Quantitative	3

13. Determine the Top 10 Most Published Authors

Lists authors who have contributed to the most research articles.

Query Query History

```
SELECT a.AuthorName, COUNT(aa.ArticleID) AS total_publications
FROM AuthorArticle aa
JOIN Author a ON aa.AuthorID = a.AuthorID
GROUP BY a.AuthorName
ORDER BY total_publications DESC
LIMIT 10;
```

Data Output Messages Notifications

SQL Showing row

	authorname text	total_publications bigint
1	NA	252
2	Lester, David	68
3	Sher, Leo	47
4	Shah, Ajit	45
5	Voracek, Martin	30
6	Goldney, Robert D.	26
7	Rezaeian, Mohsen	22
8	Brent, David A.	12
9	Forrester, Mathias B.	11
10	Andriessen, Karl	10

DATA VALIDATION

Data Validation during Web Scraping

To ensure data integrity and maintain database compatibility, the following validation checks were applied to the scraped dataset before loading it into the database. These steps helped clean and standardize the data while preserving completeness and accuracy.

1. Handling Missing Values

- Converted empty cells to **NULL** to maintain SQL compatibility.
- Retained missing values instead of dropping them, ensuring that all records remain intact.
- Used **NULL** consistently for both text and numeric missing values to prevent inconsistencies.

2. Text Data Cleaning & Standardization

- **Trimmed whitespace:** Removed leading and trailing spaces from all text fields.
- **Applied title case formatting** to key columns such as **Title, Author, and Journal.**
- **Removed unnecessary special characters f**rom fields like **Title and Abstract,** keeping essential punctuation (, . ') intact.

3. Numeric Data Validation

- Ensured numeric fields (**Year, Citations, and Impact Factor)** were properly formatted as numbers.
- Converted invalid numeric values to **NULL** instead of dropping them.

4. Duplicate Detection (Without Removal)

- Identified duplicate rows in the dataset **without deleting them** to prevent accidental data loss.

5. Date Format Validation

- Converted **Publication Date** to a valid **datetime format** for consistency.
- Replaced invalid or missing dates with **NULL** instead of removing them.
- Ensured incorrect date values did not cause errors during data loading.

6. DOI Field Validation

- Replaced empty DOI values with **NULL** while preserving valid DOI entries.
- Ensured that missing DOI fields do not cause errors when inserted into the database.

7. Article URL Validation

- URLs containing only "http://dx.doi.org/" were replaced with **NULL** as they were incomplete.
- Retained all other valid URLs to ensure that critical reference links were not lost.

By applying these structured **data validation techniques,** we ensured that the dataset was properly cleaned and prepared for **database integration,** minimizing inconsistencies and potential errors.

Data Validation using SQL

Data Validation Using SQL Before Normalization

Before normalizing the data, we performed several validation checks to ensure data quality in the staging_table, which contained raw CSV data. The primary objectives were to identify and address missing values, duplicate records, and inconsistent formatting before loading the data into relational tables.

1. **Identifying and ~~Removing~~ Duplicates** ○ Detected duplicate records where Title, Year, and Journal were identical, indicating multiple instances of the same article.

Data Validation Using SQL After Normalization

After structuring the data into relational tables, additional validation checks were conducted to ensure the data was properly transformed and relationships were accurately established.

1. **Splitting Multi-Valued Fields into Individual Entries** ○ Authors were extracted and stored as individual records in the Author table instead of a single text field.
    - Categories and Keywords were normalized by splitting comma-separated values into distinct rows.
2. **Ensuring Referential Integrity** ○ Verified that all foreign key relationships were correctly established. ○ Checked if every Article, Author, and Category had valid references in their respective tables.
3. **Standardizing Language Abbreviations** ○ Verified that every abbreviation in the Language table was correctly mapped to its corresponding language name.
    - Fixed cases where unknown or null values were assigned.

4. **Cross-Referencing Data Sources and Grant Information** ○ Ensured each grant or source type was uniquely represented without redundancy.

5. **Validating Foreign Key Integrity Post-Population** ○ Checked that every ArticleID, AuthorID, CategoryID, and other entity IDs in relationship tables had valid references.
    - Ensured there were no orphaned records (entries in many-to-many tables referencing non-existent entities).

By executing these post-normalization validation steps, we ensured that the database structure was accurate, relationships were correctly defined, and data was efficiently organized for querying and analysis. ***Data Validation with Python***

Before proceeding with Exploratory Data Analysis (EDA), we need to perform data validation checks to ensure data integrity.

1. Connecting to Database and then checking for number of rows

```
import psycopg2
import pandas as pd

def connect():
   """Connect to the PostgreSQL database server."""
conn = None     params_dic = {
       "host": "127.0.0.1",
       "database": "SafetylitDB",
       "user": "#####",  # Replace with your PostgreSQL username
       "password": "####",  # Replace with your PostgreSQL password
       "port": 5431  # Default PostgreSQL port
   }
try:
       print('Connecting to the PostgreSQL database...')
conn = psycopg2.connect(**params_dic)
print("Connection successful")        return conn
except psycopg2.Error as error:
       print(f"Error connecting to PostgreSQL: {error}")
return None

def postgres_to_dataframe(conn, select_query):
   """Transform a SELECT query result into a pandas DataFrame."""
try:         with conn.cursor() as cursor:
           cursor.execute(select_query)
           column_names = [desc[0] for desc in cursor.description]  # Fetch column names
dynamically
           result = cursor.fetchall()
           return pd.DataFrame(result, columns=column_names)
except psycopg2.Error as error:
```

```
        print(f"Error executing query: {error}")
return None

# Connect to the database conn
= connect()

if conn:
    # Load each table into a DataFrame
tables = [
    "Article", "Author", "Journal", "Language", "Category", "Keyword",
    "GrantInformation", "ResearchMethodology", "PreventionMeasures", "DataSource",
    "AuthorArticle", "ArticleCategory", "ArticleKeyword", "ArticleGrant",
    "ArticleMethodology", "ArticlePrevention", "ArticleDataSource"
]
    dataframes = {table: postgres_to_dataframe(conn, f"SELECT * FROM {table};") for
table in tables}

    # Close the connection
conn.close()
    print("PostgreSQL connection is closed")

    # Display table information
print("\nData Summary:")     for
table, df in dataframes.items():
        if df is not None:
            print(f"{table} Table: {df.shape[0]} rows")

    # Display sample data from each table
for table, df in dataframes.items():
        if df is not None:              print(f"\nSample
Data from {table} Table:")              print(df.head())

else:
    print("Connection failed. Check your connection parameters.")
```

```
Connecting to the PostgreSQL database...
Connection successful
PostgreSQL connection is closed

Data Summary:
Article Table: 29020 rows
Author Table: 67795 rows
Journal Table: 4950 rows
Language Table: 37 rows
Category Table: 74 rows
Keyword Table: 10706 rows
GrantInformation Table: 2 rows
ResearchMethodology Table: 2 rows
PreventionMeasures Table: 11 rows
DataSource Table: 5 rows
AuthorArticle Table: 6704 rows
ArticleCategory Table: 17852 rows
```

Below is a full snapshot of tables imported into Python

Data Summary:
Article Table: 29020 rows
Author Table: 67795 rows
Journal Table: 4950 rows
Language Table: 37 rows
Category Table: 74 rows
Keyword Table: 10706 rows
GrantInformation Table: 2 rows
ResearchMethodology Table: 2 rows
PreventionMeasures Table: 11 rows
DataSource Table: 5 rows
AuthorArticle Table: 6704 rows
ArticleCategory Table: 17852 rows
ArticleKeyword Table: 127 rows
ArticleGrant Table: 29022 rows
ArticleMethodology Table: 29063 rows
ArticlePrevention Table: 5998 rows
ArticleDataSource Table: 29048 rows

The above summary is consistent with our results in SQL

2. Check for duplicate records

Before checking for duplicates we need to make the id columns the index of the dataframes.

Find all table names then corresponding dataframes:

```
[13]:
dataframes.keys()
```

```
[13]:
dict_keys(['Article', 'Author', 'Journal', 'Language', 'Category', 'Keyword',
'GrantInformation', 'ResearchMethodology', 'PreventionMeasures', 'DataSourc
e', 'AuthorArticle', 'ArticleCategory', 'ArticleKeyword', 'ArticleGrant', 'Ar
ticleMethodology', 'ArticlePrevention', 'ArticleDataSource'])
```

```
[14]:
df_article = dataframes["Article"].copy()
df_author = dataframes["Author"].copy()
df_journal = dataframes["Journal"].copy()
df_language = dataframes["Language"].copy()
df_category = dataframes["Category"].copy()
df_keyword = dataframes["Keyword"].copy()
df_grant_information = dataframes["GrantInformation"].copy()
df_research_methodology = dataframes["ResearchMethodology"].copy()
df_prevention_measures = dataframes["PreventionMeasures"].copy()
df_data_source = dataframes["DataSource"].copy()
df_author_article = dataframes["AuthorArticle"].copy()
df_article_category = dataframes["ArticleCategory"].copy()
df_article_keyword = dataframes["ArticleKeyword"].copy()
df_article_grant = dataframes["ArticleGrant"].copy()
df_article_methodology = dataframes["ArticleMethodology"].copy()
df_article_prevention = dataframes["ArticlePrevention"].copy()
df_article_data_source = dataframes["ArticleDataSource"].copy()
```

Setting index for all main data tables. Leaving out the many-to-many relationship tables as they have foreign keys as primary keys:

[52]:

```
# Setting ID columns as index for all tables
df_article.set_index("articleid", inplace=True)
df_author.set_index("authorid", inplace=True)
df_journal.set_index("journalid", inplace=True)
df_language.set_index("languageid", inplace=True)
df_category.set_index("categoryid", inplace=True)
df_keyword.set_index("keywordid", inplace=True)
df_grant_information.set_index("grantid", inplace=True)
df_research_methodology.set_index("methodologyid", inplace=True)
df_prevention_measures.set_index("preventionid", inplace=True)
df_data_source.set_index("sourceid", inplace=True)
```

Show Duplicate rows to double check:

[54]:

```
# Find and display duplicate rows in df_article
duplicates_df_article = df_article[df_article.duplicated()]
display(duplicates_df_article)
```

articleid	title	year	doi	url	volume	number	pages	abstract	affiliation	pmid	journalid	languageid
22399	Depressed Youth, Suicidality And Antidepressants	2005	NA	http://dx.doi.org/	183	5	275; author reply 276-275; author reply 276	pAbstract unavailablep pLanguage enp	Not Available	16247910	2632	None
2349	Depressive Mixed State Evidence For A New Form...	2007	NA	http://dx.doi.org/	3	6	899-902	BACKGROUND A high proportion of unipolar and b...	EA4139 Laboratoire de psychologie, Université ...	19300625	1392	None
3271	Firearm Legislation And GunRelated Fatalities	2013	NA	http://dx.doi.org/	173	21	e2011-e2011	pAbstract unavailableppLanguage enp	Not Available	unavailable	1516	None
4830	Managing Psychiatric Emergencies	2007	NA	http://dx.doi.org/	7	1	09-Mar	Behavioral emergencies are common goals of the...	Not Available	unavailable	1049	None
6530	PsychoSocial Causes Of Suicide In Multan	2007	NA	http://dx.doi.org/	18	8	15-Aug	Suicide occurs when the present the future is...	Not Available	unavailable	2592	None
7279	Secondary School Learners' Essays On Suicide P...	2007	NA	http://dx.doi.org/	19	2	131-136	OBJECTIVES Considering the extent of the probl...	Not Available	25865445	3317	None
7494	Side Effects Of Combination Of Interferon Plus...	2007	NA	http://dx.doi.org/	21	3	187-191	OBJECTIVE To assess the side effects of combin...	Not Available	unavailable	4162	None
8206	Suicide By SelfImmolation, A Cross Sectional S...	2007	NA	http://dx.doi.org/	1	2	15-Nov	OBJECTIVE Although in international studies, i...	Not Available	unavailable	2630	None
8335	Suicide Prevention A Resource For The Family	2007	NA	http://dx.doi.org/	1	2	10-Apr	The family can play an important role in the p...	Not Available	unavailable	2630	None
9216	The Prevalence Of Substance Use And Its Associ...	2011	NA	http://dx.doi.org/	53	1	83-90	PURPOSE In South Africa there has been an incr...	Not Available	unavailable	1626	None
16276	Suicide And Documentation A MustRead	2006	NA	http://dx.doi.org/	44	12	08-Aug	pAbstract unavailablep pLanguage enp	Not Available	17201034	774	None

3. Validate Data Types

```
[57]: # Validate Data Types for all DataFrames
      for name, df in dataframes.items():
          print(f"\n{name} - Data Types:\n{df.dtypes}")
```

```
Article - Data Types:
articleid       int64
title          object
year            int64
doi            object
url            object
volume         object
number         object
pages          object
abstract       object
affiliation    object
pmid           object
journalid       int64
languageid     object
dtype: object

Author - Data Types:
authorid       int64
authorname    object
dtype: object

Journal - Data Types:
journalid       int64
journalname    object
dtype: object

Language - Data Types:
languageid       int64
languagename    object
abbreviation    object
dtype: object
```

The data types are consistent with the dataset.

4. Check Text columns for Inconsistencies

Text columns can have inconsistencies like:

- Leading/trailing spaces
- Special characters or typos
- Empty strings instead of NULL values

Show text columns in all dataframes:

[71]:
```
# Identify object (text) columns in all DataFrames
text_columns = {name: df.select_dtypes(include=['object']).columns for
name, df in dataframes.items()}

# Display text columns for each DataFrame
for name, cols in text_columns.items():
    print(f"{name} - Text Columns: {list(cols)}")
```

```
Article - Text Columns: ['title', 'doi', 'url', 'volume', 'number', 'page
s', 'abstract', 'affiliation', 'pmid', 'languageid']
Author - Text Columns: ['authorname']
Journal - Text Columns: ['journalname']
Language - Text Columns: ['languagename', 'abbreviation']
Category - Text Columns: ['categoryname']
Keyword - Text Columns: ['keyword']
GrantInformation - Text Columns: ['grantdetails']
ResearchMethodology - Text Columns: ['methodologytype']
PreventionMeasures - Text Columns: ['preventionmeasure']
DataSource - Text Columns: ['sourcename']
AuthorArticle - Text Columns: []
ArticleCategory - Text Columns: []
ArticleKeyword - Text Columns: []
ArticleGrant - Text Columns: []
ArticleMethodology - Text Columns: []
ArticlePrevention - Text Columns: []
ArticleDataSource - Text Columns: []
```

No empty strings found the columns in the dataframes:

[76]:
```
def check_empty_strings(df, df_name):
    """Check for empty strings in text columns."""
    for col in df.select_dtypes(include=['object']).columns:
        empty_count = (df[col] == "").sum()
        if empty_count > 0:
            print(f"\n{df_name} - Empty Strings Found in Column: {col} ->
{empty_count} occurrences")
        else:
            print("No Empty Strings Found")

# Run check on all tables
for name, df in dataframes.items():
    check_empty_strings(df, name)
```

```
No Empty Strings Found
No Empty Strings Found
```

Check for columns with special characters:

[77]:
```python
import re

def check_special_characters(df, df_name):
    """Check for special characters in text columns and list affected columns."""
    special_char_pattern = re.compile(r"[^a-zA-Z0-9\s.,'-]")  # Allow common characters

    affected_columns = []
    for col in df.select_dtypes(include=['object']).columns:
        if df[col].str.contains(special_char_pattern, na=False).any():
            affected_columns.append(col)

    if affected_columns:
        print(f"\n{df_name} - Columns with Special Characters: {affected_columns}")

# Run check on all tables
for name, df in dataframes.items():
    check_special_characters(df, name)
```

```
Article - Columns with Special Characters: ['doi', 'url', 'volume', 'numbe
r', 'pages', 'affiliation']

Author - Columns with Special Characters: ['authorname']

Journal - Columns with Special Characters: ['journalname']

Category - Columns with Special Characters: ['categoryname']

Keyword - Columns with Special Characters: ['keyword']
```

The output shows the columns in each DataFrame that contain special characters beyond standard alphanumeric values, spaces, commas, periods, hyphens, and apostrophes. These columns may require cleaning to ensure consistency and proper formatting for further data analysis.

5. Validate DOI and URL fields

In this step, we will validate the **DOI (Digital Object Identifier) and URL fields** in the dataset to ensure they follow the correct format and structure.

The DOI is a standardized identifier for academic publications, and it should begin with "10." followed by a valid alphanumeric sequence. Similarly, the URL field should contain properly formatted web addresses starting with "http://" or "https://".

We will check for missing values, detect incorrectly formatted DOIs and URLs, and identify any anomalies that might require cleaning. Below is the Python code to perform these validations.

[78]:
```
import re

def validate_doi_url(df, df_name):
    """Check DOI and URL fields for missing values and format validity."""

    if 'doi' in df.columns:
        doi_pattern = re.compile(r"^10\.\d{4,9}/[-._;()/:A-Za-z0-9]+$")
        invalid_doi = df[~df['doi'].isna() &
~df['doi'].str.match(doi_pattern, na=False)]
        print(f"\n{df_name} - Invalid DOIs:\n",
invalid_doi[['doi']].head())

    if 'url' in df.columns:
        url_pattern = re.compile(r"^(http|https)://[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}(/.*)?$")
        invalid_url = df[~df['url'].isna() &
~df['url'].str.match(url_pattern, na=False)]
        print(f"\n{df_name} - Invalid URLs:\n",
invalid_url[['url']].head())

# Run validation on the Article table (where DOI and URL exist)
validate_doi_url(df_article, "Article")
```

```
Article - Invalid DOIs:
            doi
articleid
1           NA
2           NA
3           NA
4           NA
5           NA

Article - Invalid URLs:
            url
articleid
17574       NA
17591       NA
17592       NA
17593       NA
17579       NA
```

The output shows the results of DOI and URL validation in the **Article** table. It indicates that the first few DOI values are missing (represented as **NA**), which suggests missing or improperly formatted DOIs. Similarly, several URLs are also flagged as invalid, meaning they either have incorrect formatting or are missing entirely.

EXPLORATORY DATA ANALYSIS

The exploratory data analysis (EDA) phase provides a comprehensive overview of the dataset, highlighting patterns, relationships, and potential data quality issues. This section presents the key findings from data cleaning, visualization, and text processing efforts.

The dataset was first inspected for missing values, inconsistencies, and outliers. Preliminary analysis included loading the data and summarizing its structure using descriptive statistics. Initial findings showed that some columns contained null values, necessitating appropriate data imputation techniques.

Data cleaning involved handling missing values through imputation or removal where necessary, standardizing textual data by converting to lowercase and removing special characters, eliminating stopwords to improve text analysis accuracy, and tokenizing, lemmatizing, and stemming textual data for better feature extraction.

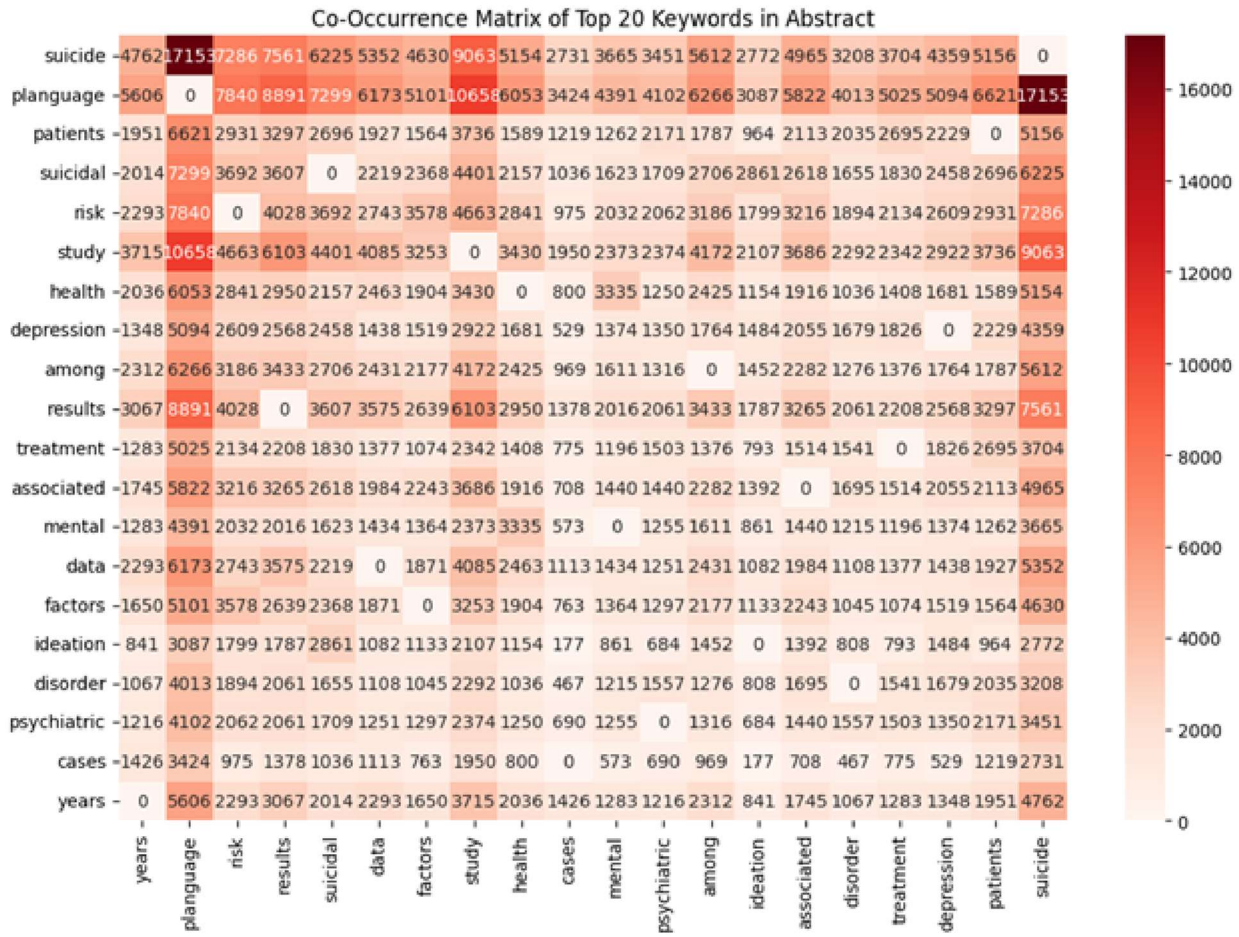
Several visualizations were conducted to uncover insights, including:

- **Distribution Analysis**: Histograms and boxplots were used to examine the distribution of key numerical variables, identifying potential skewness and outliers.
- **Keyword Frequency Analysis**: A frequency distribution of extracted keywords from article titles revealed common themes and topics.
- **Trend Analysis**: Time series plots demonstrated patterns overtime, highlighting fluctuations in data volume and keyword trends.
- **Correlation Analysis:** A heatmap was generated to identify relationships between keywords, guiding further analysis.

To extract meaningful insights from textual data, keyword extraction and word frequency analysis were conducted. The top 20 keywords were identified, shedding light on the most prevalent topics within the dataset. This step helped in understanding the thematic focus and potential areas for deeper analysis.

Data Visualizations

Co-Occurrence Matrix of Top 20 Keywords in Abstracts



This heatmap visualizes the co-occurrence of the top 20 keywords in abstracts from suicide-related research articles sourced from SafetyLit. The matrix quantifies the frequency with which pairs of keywords appear together in the same abstracts, with darker red shades indicating higher co-occurrence values.

Key Insights:

1. **High-Impact Keyword Relationships:**
    - The most frequently co-occurring terms include:
        - "suicide" and "risk"; indicating that suicide risk assessment is a central theme in the literature.
        - "suicidal" and "ideation"; reflecting a common research focus on suicidal thoughts.

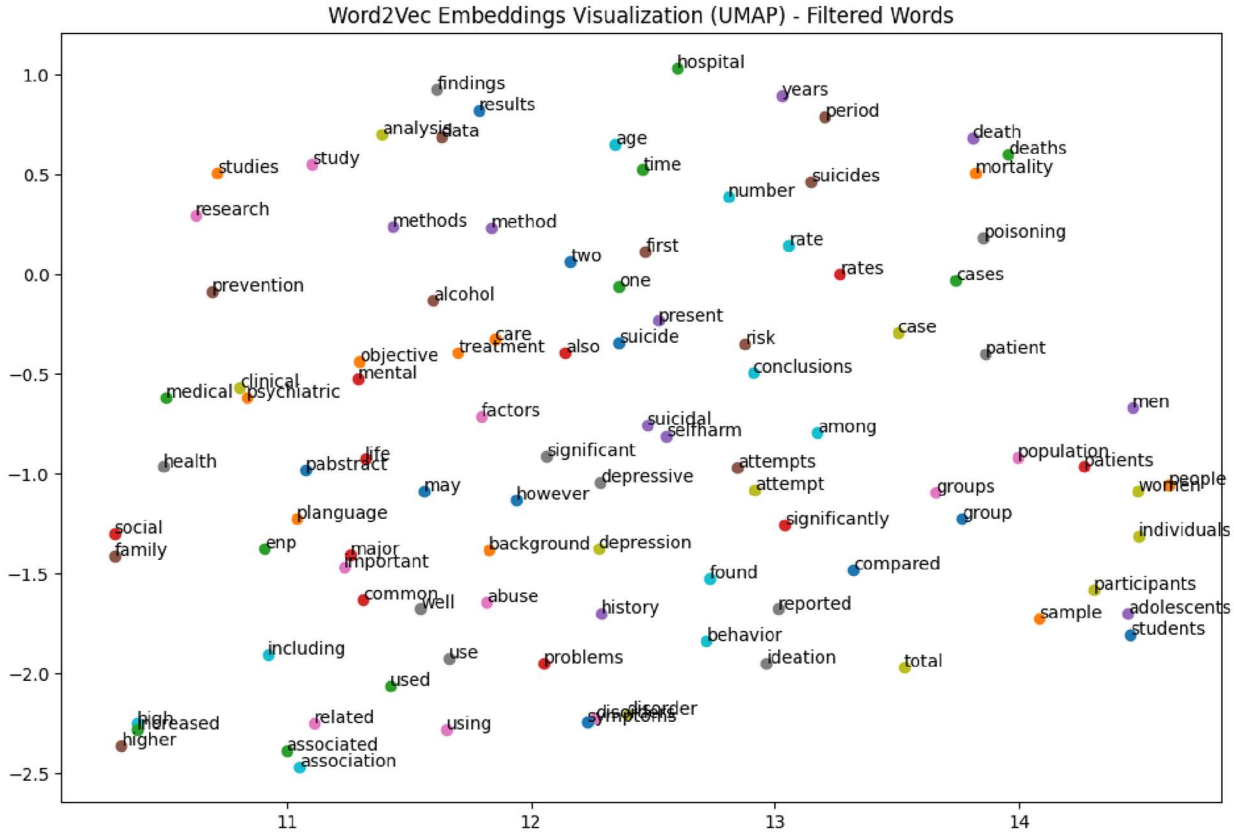
        - "patients" and "treatment"; highlighting the clinical aspect of suicide prevention.
        - "depression" and "mental"; reinforcing the connection between mental health conditions and suicide.
    - The strong correlation between "language" and "suicide" suggests a focus on suicide-related language analysis, possibly in social media, interviews, or clinical settings.

2. **Research Trends and Thematic Clusters:**
    - The clustering of terms like "study," "results," and "data" suggests a methodological emphasis on data-driven research.
    - Terms such as "disorder," "psychiatric," and "associated" appear frequently together, indicating discussions around psychiatric disorders and their link to suicidality.
3. **Clinical and Social Implications:**
    - The presence of terms such as "health," "treatment," and "associated" suggests a strong public health and intervention-based perspective.
    - The co-occurrence of "cases," "years," and "factors" points to studies analyzing long-term trends, risk factors, and case studies.

Conclusion:

This co-occurrence matrix provides valuable insight into the thematic structure of suicide research. The high-frequency associations indicate that contemporary studies largely focus on risk assessment, psychiatric disorders, treatment, and suicide ideation. This visualization helps identify gaps and potential areas for further exploration, such as underrepresented factors influencing suicide beyond clinical settings.

Word2Vec Embeddings Visualization



This visualization represents word embeddings using Word2Vec, with dimensionality reduction performed by UMAP. The plot highlights how words related to self-harm and suicide are grouped based on semantic similarity.

Key Insights:

1. **Semantic Relationships and Word Proximity**
   - Frequently co-occurring words appear in close proximity, reflecting meaningful semantic relationships.
   - Terms such as "suicide," "risk," and "self-harm" cluster together, reinforcing their strong association in the dataset.

2. **Model Refinements and Enhancements**
   - Compared to previous iterations, this model produces smoother results with less separation, while this may seem like a set backwards, smoothness is expected given the dataset's thematic focus, and the separation that we

were getting was a result of lack of tune and a mismatch of visualization techniques.

- Earlier skewness in clustering was likely due to suboptimal preprocessing and unoptimized visualization parameters.
- Improved filtering has enhanced the clarity of word groupings, though further refinements remain necessary.

3. ~~Areas for Further Improvement~~

- ~~The stopword list requires additional refinement to ensure only nonmeaningful words are removed.~~
- ~~Lemmatization should be incorporated to consolidate variations of the same word (e.g., "suicide" vs. "suicidal") when they do not carry distinct semantic weight.~~
- ~~UMAP parameter tuning (n_neighbors, min_dist) can further enhance the separation of word groups, improving the interpretability of clusters.~~

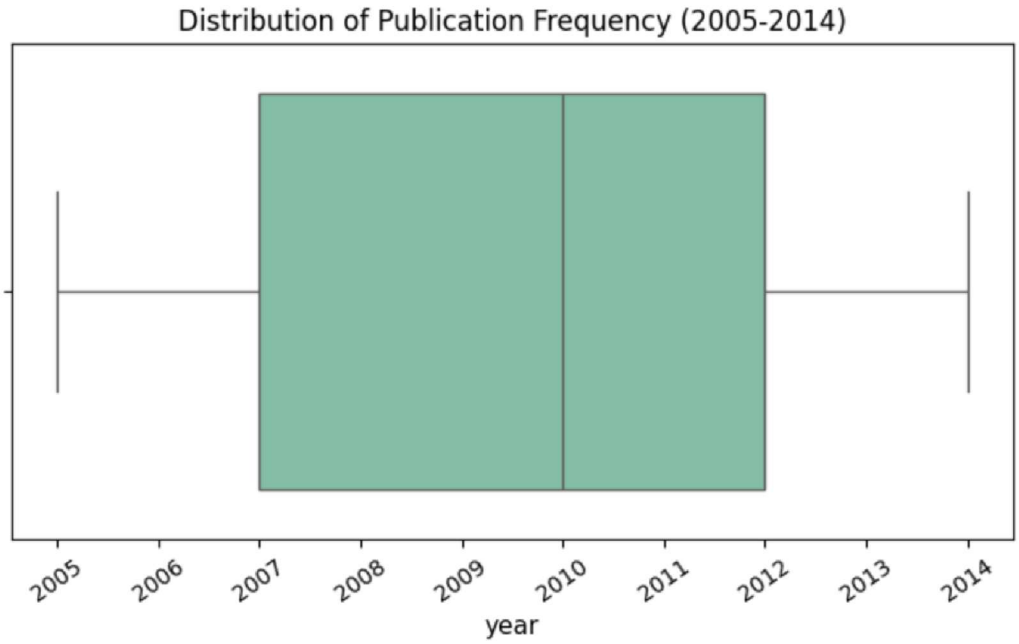
4. ~~Future Optimization Steps~~

- ~~Although Word2Vec is a relatively simple model, fine-tuning hyperparameters can lead to more accurate embeddings.~~
- ~~With improved stopword handling and lemmatization, the model can achieve tighter clustering while avoiding overfitting.~~
- ~~These findings emphasize the critical role of meticulous data preprocessing in ensuring robust, meaningful representations of text data.~~

Conclusion:

This analysis demonstrates that Word2Vec embeddings can effectively capture semantic relationships in self-harm-related text data. While the model has improved in smoothness and clustering accuracy, further refinements in stopword removal, lemmatization, and hyperparameter tuning can enhance interpretability. Optimizing these preprocessing steps is essential for extracting clear, actionable insights from textbased datasets.

Analysis of Suicide Research Publication Trends (2005–2014)



The box plot visualizes the distribution of suicide-related research publications from 2005 to 2014, based SafetyLit data. This statistical summary provides insights into publication trends over this decade.

Key Insights:

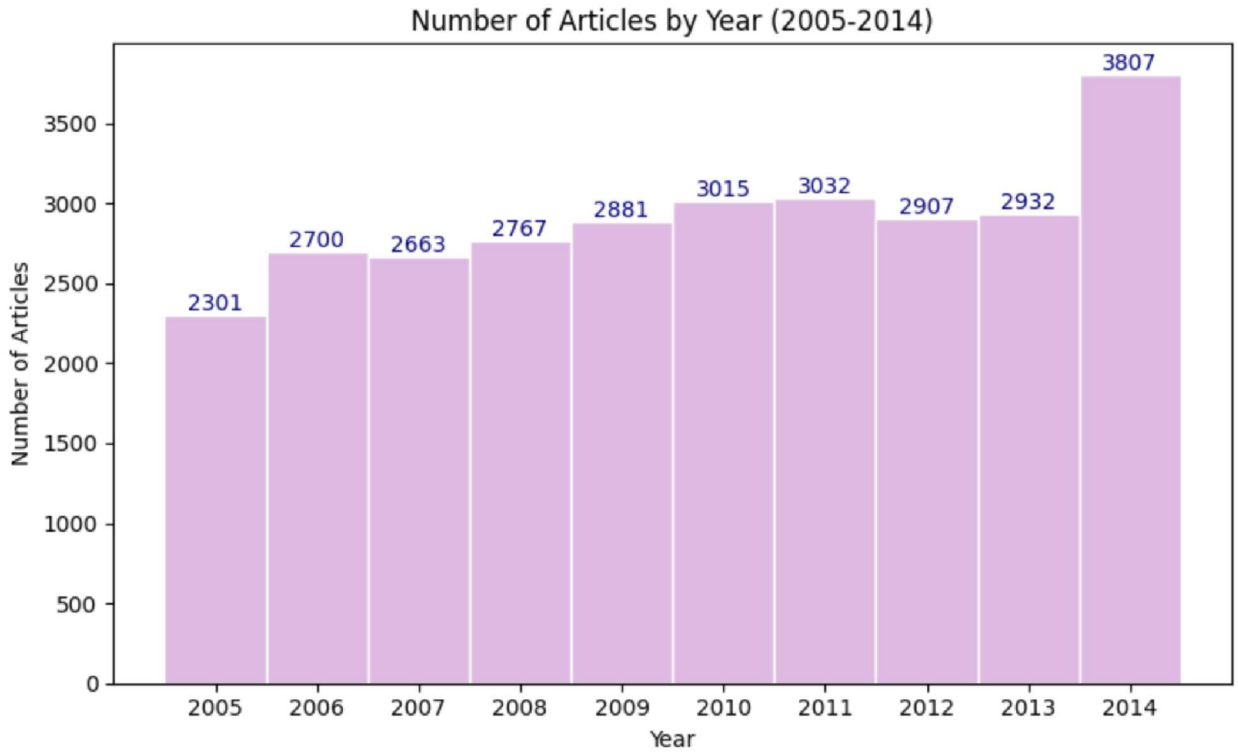
1. **Publication Density and Central Trend:**
    - The interquartile range (IQR) spans from 2007 to 2012, indicating that the middle 50% of suicide research studies were published within this period.
    - The median publication year is 2010, meaning half of the studies were published before this year and the other half after.

    Early and Late Research Distribution:
    - The whiskers extend from 2005 to 2014, representing the full range of publication years for this dataset.
    - 25% of the research was conducted before 2007 and the other 25% was conducted post 2012.
2. **Overall Trend and Implications:**
    - The increase in publications from 2007 onward suggests a growing academic and public health interest in suicide research.

Conclusion:

This analysis highlights a significant increase in suicide-related research publications from 2007 to 2012. The findings suggest that this period saw increased academic and clinical focus on suicide prevention, risk assessment, and mental health interventions. Understanding these publication trends can inform future research directions and funding allocations.

Suicide Research Publication Trends (2005–2014)



This histogram shows the number of suicide-related research articles published annually from 2005 to 2014, based on SafetyLit data.

Key Insights:

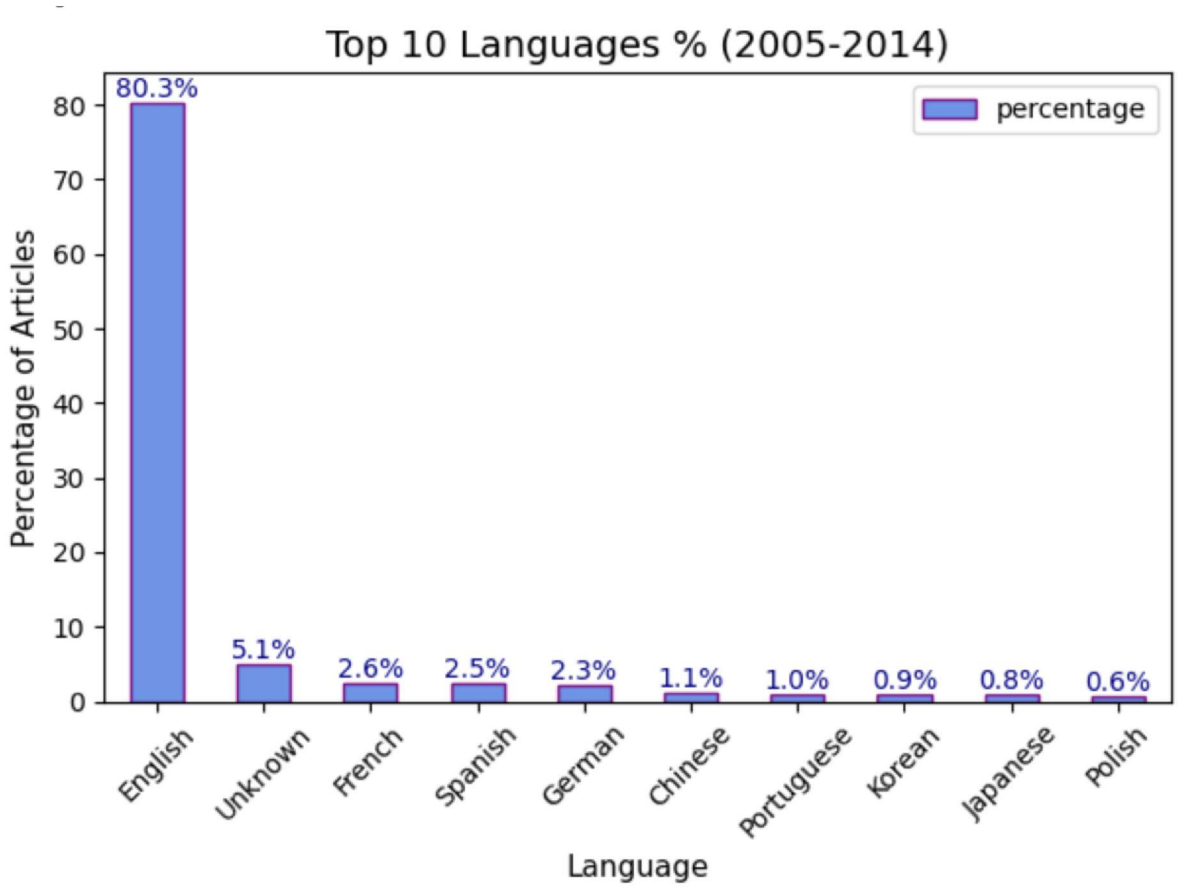
1. **Steady Growth:**
    - Publications increased from 2,301 in 2005 to 3,807 in 2014, which is about a 65% rise, reflecting growing academic and public health interest.
    - The number of articles consistently increased until 2010–2011, when it peaked above 3,000 per year.
2. **Fluctuations & Key Trends:**
    - The peak in the late 2000s to early 2010s may correspond with heightened awareness or advancements in suicide prevention strategies.

- A slight dip in 2012 was followed by recovery in 2013.
- 2014 saw a sharp increase in publications, possibly due to reasons like policy changes, increased funding, or emerging research trends.

Conclusion:

The steady rise in suicide research publication suggests greater awareness, improved prevention strategies, and evolving methodologies. The spike in 2014 may indicate renewed focus and funding in this critical area, possibly reflecting the impact of significant historical events during this time period.

Language Distribution in Suicide Research (2005–2014)



This bar chart shows the percentage of suicide-related research articles published in the top 10 languages from 2005 to 2014, based on SafetyLit data. There are 36 languages in total in this dataset. "Unknown" represents articles, where language is not defined in Safetylit data.

Key Insights:

1. **English as the Primary Language:**

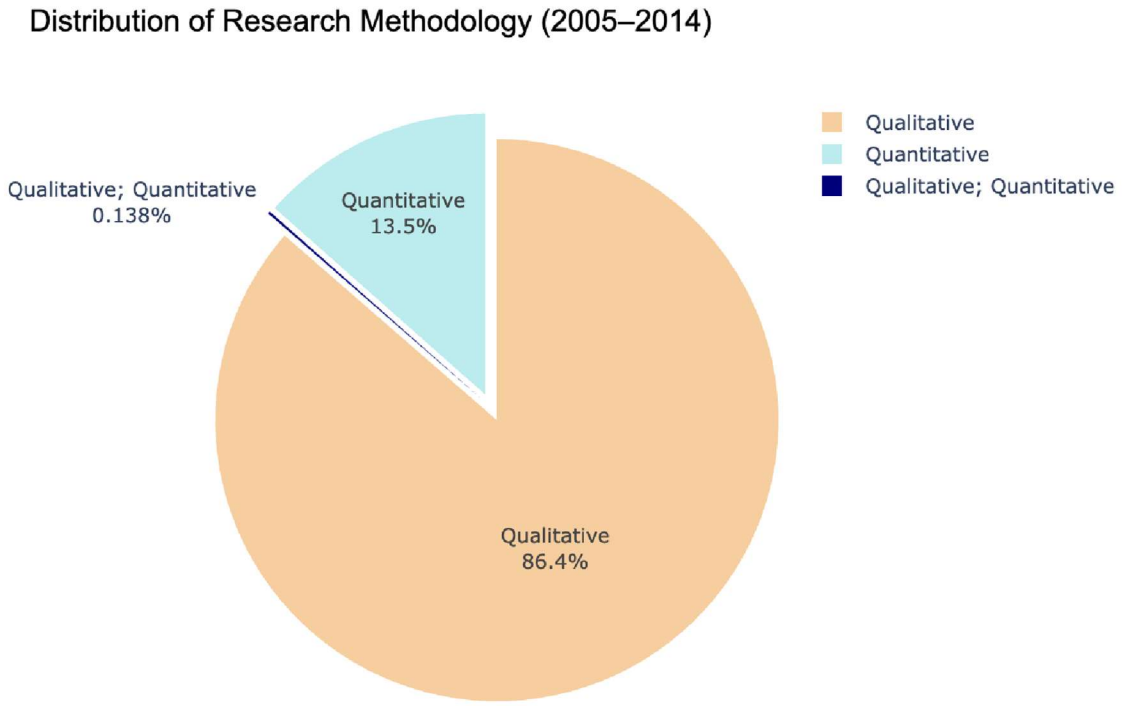
- 80.3% of articles are published in English, making it the primary language of suicide research in this timeframe.
- This suggests that most studies are accessible to English-speaking researchers, limiting reach in non-English-speaking regions.

2. **Representation of Other Languages:**
   - French (2.6%), Spanish (2.5%), and German (2.3%) follow, but each accounts for only a small fraction compared to English.
   - Chinese (1.1%) and Korean (0.9%) have even lower representation, despite high suicide rates in some regions.

Conclusion:

The predominance of English in suicide research reflects global academic publishing norms. However, the presence of other languages suggests ongoing efforts to diversify accessibility. Further expanding multilingual research publications may help reach more regional audiences and enhance global collaboration in suicide prevention studies.

Research Methodology Distribution in Suicide Studies (2005–2014)



This pie chart illustrates the proportion of qualitative and quantitative research methodologies used in suicide-related studies from 2005 to 2014, based on SafetyLit data. Further explanations for the qualitative: quantitative portion is noted in the discovery section of this report.

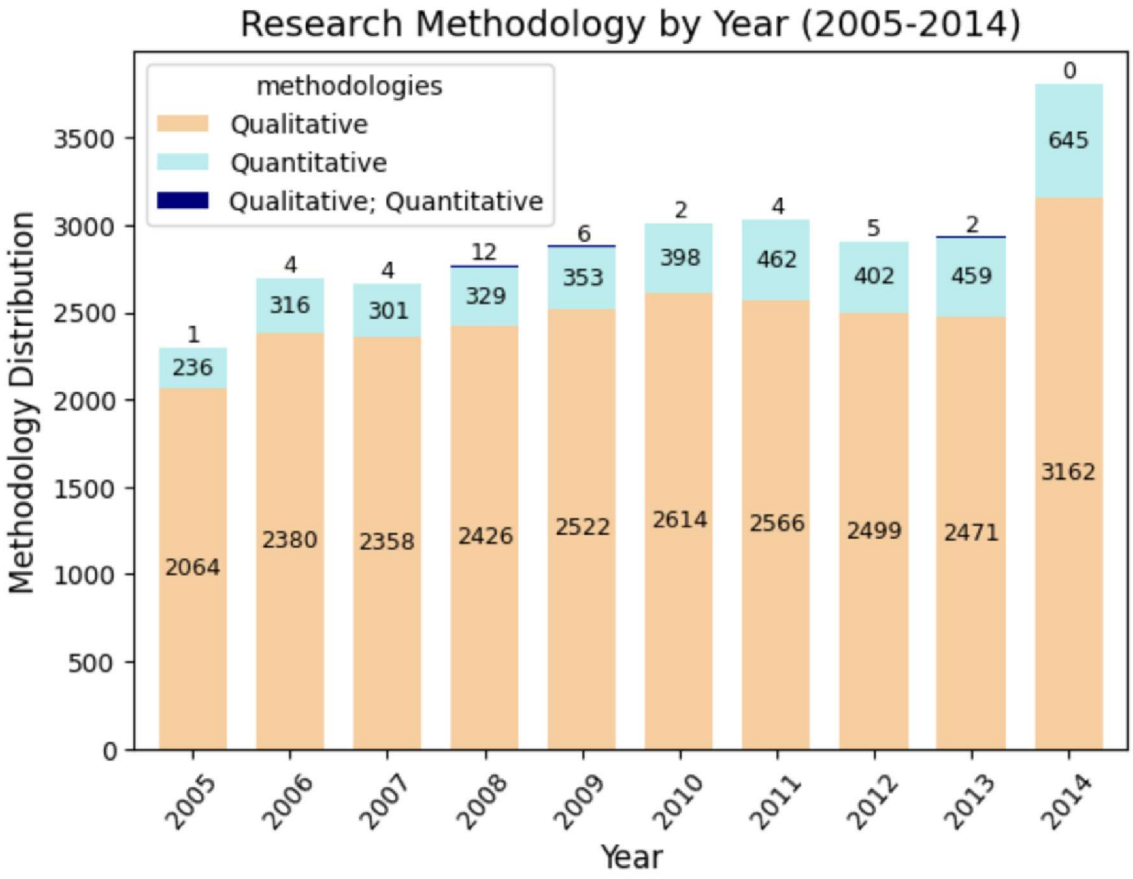
Key Insights:

- The high prevalence of qualitative research (86.4%) highlights a strong commitment to understanding personal experiences, social factors, and psychological contexts related to suicide.
- The presence of quantitative studies (13.5%) demonstrates the use of data-driven approaches, allowing for statistical validation and broader trend analysis.
- The combination of both methodologies ensures a comprehensive exploration of suicide, integrating personal narratives with empirical evidence.

Conclusion:

This balanced research approach enhances the depth and accuracy of suicide studies, supporting both individual case insights and large-scale trend analysis. Expanding these methodologies further can strengthen prevention strategies and improve mental health interventions.

Growth Trends in Research Methodologies for Suicide Studies (2005–2014)



This stacked bar chart illustrates the annual trends in qualitative and quantitative research methodologies used in suicide studies from 2005 to 2014, based on SafetyLit data. Further explanations for the qualitative: quantitative portion is noted in the discovery section of this report.

Key Insights:

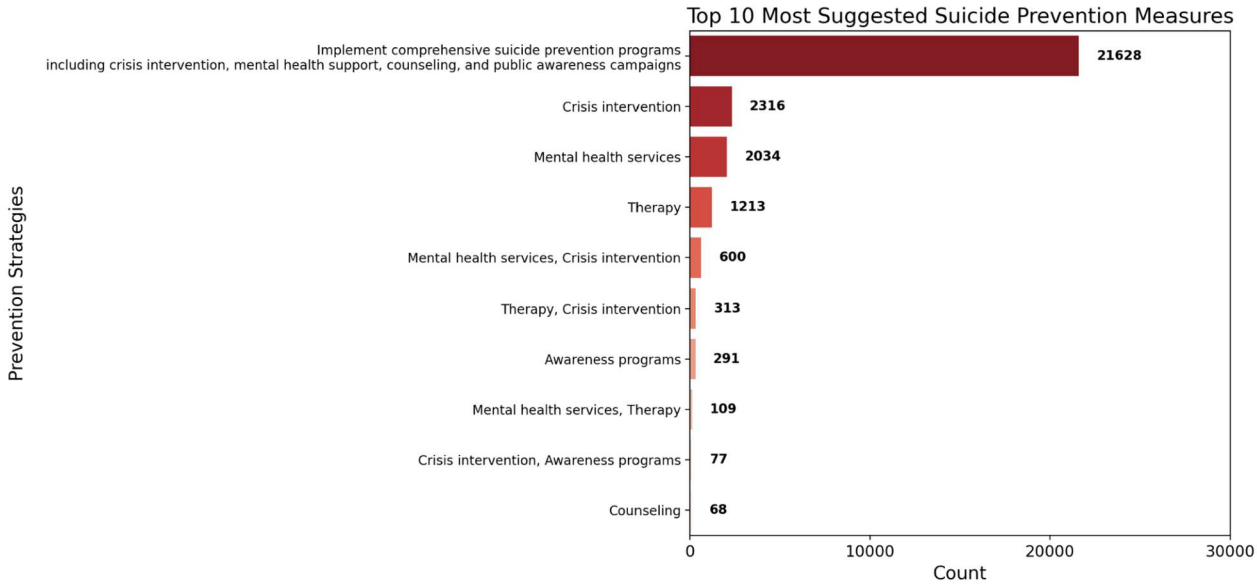
1. **Qualitative Research (Consistently High but Stable Growth)**
    - Qualitative studies dominate each year, showing steady but slow growth over the decade.
    - The number of qualitative studies remained relatively stable from 2010 to 2013, with minor fluctuations.
    - In 2014, qualitative research saw a significant increase, reaching its highest recorded value (3,162 studies), suggesting renewed interest or additional funding in qualitative approaches.
2. **Quantitative Research (Gradual but Noticeable Increase)**
    - Although a smaller portion of studies are quantitative, they show a gradual upward trend over the years.

- The number of quantitative studies increased from 236 in 2005 to 645 in 2014, indicating a shift toward more data-driven and statistical analyses in suicide research.
- The sharpest rise in quantitative studies occurred in 2014, possibly reflecting advancements in data collection, mental health analytics, or increased adoption of computational methods.

Conclusion:

While qualitative research remains dominant, quantitative studies have steadily increased over the years, particularly in 2014. This suggests an evolving research landscape where both methodologies are being leveraged, with qualitative methods providing in-depth context and quantitative methods enabling statistical validation and trend analysis.

Top Suggested Suicide Prevention Measures



This horizontal bar chart presents the 10 most recommended suicide prevention strategies based on SafetyLit data.

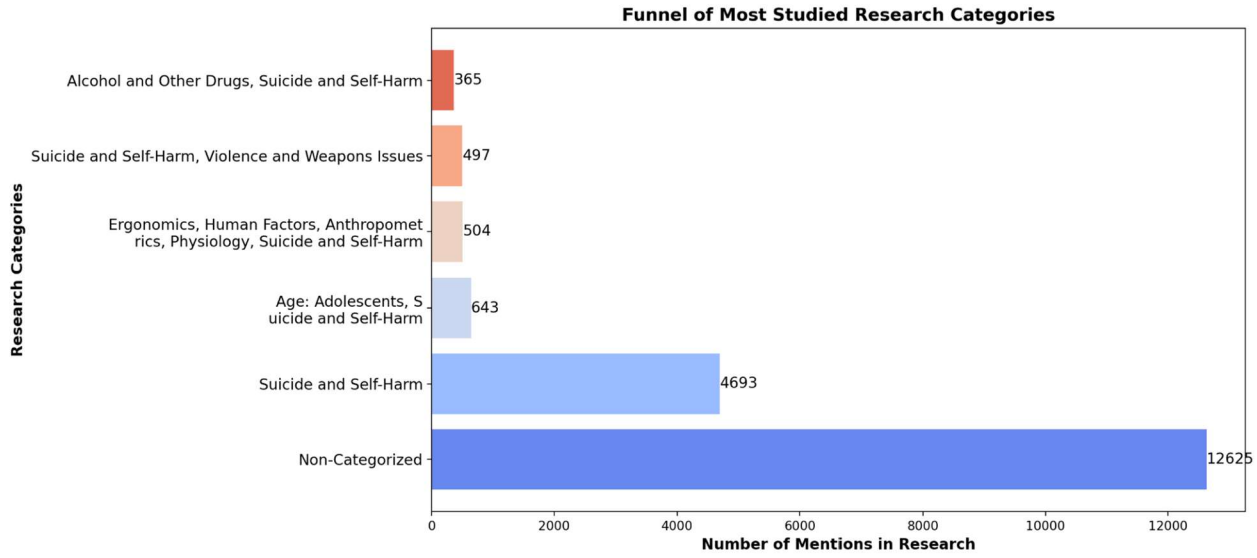
Key Insights:

- The most frequently suggested measure is comprehensive suicide prevention programs (21,628 mentions), which include crisis intervention, mental health support, counseling, and public awareness campaigns.
- Crisis intervention (2,316 mentions) and mental health services (2,034 mentions) are also widely emphasized, highlighting their critical role in immediate suicide prevention efforts.
- Therapy (1,213 mentions) is another key recommendation, often combined with mental health services or crisis intervention.
- Other strategies, such as awareness programs (291 mentions) and counseling (68 mentions), are less frequently mentioned but still contribute to prevention efforts.

Conclusion:

The data suggests a strong preference for multi-faceted prevention approaches that integrate crisis response, mental health services, and therapy. These findings highlight the importance of both immediate intervention and long-term mental health support in suicide prevention efforts.

Most Studied Research Categories in Suicide Studies



This bar chart displays the most frequently studied research categories related to suicide and self-harm, based on mentions in SafetyLit data.

Key Insights:

- The largest category is Non-Categorized (12,625 mentions), indicating many studies do not fall into specific predefined topics.
- "Suicide and Self-Harm" (4,693 mentions) is the most explicitly studied category, showing a strong focus on direct suicide-related factors.
- Other notable areas include:
    - Adolescents and Suicide (643 mentions); reflecting concern about youth suicide prevention.
    - Violence, Weapons, and Suicide (497 mentions); highlighting links between self-harm and external harm.
    - Alcohol, Drugs, and Suicide (365 mentions); showing the connection between substance use and suicide risks.

Conclusion:

The focus on suicide and self-harm demonstrates a priority in understanding direct risk factors. The presence of substance use, violence, and adolescent-related studies suggests ongoing efforts to examine external influences on suicidal behavior. Expanding research into underrepresented areas could provide a more holistic understanding of suicide prevention.

Alternative Research Fields in Suicide Studies

Field	Top Mentions	Key Insight
Research Methodology	Qualitative (86.4%) vs Quantitative (13.5%)	Qualitative research dominates, but quantitative studies are increasing.
Research Language	English (80.3%), French Spanish (2.6%)	English is the primary language in (2.7%), research, much less multilingual studies. Many studies are reprinted in English skewing this..
Prevention Strategies	Comprehensive Programs (21,628), Crisis (2,316)	Programs integrating mental health services and crisis intervention are Intervention widely recommended.

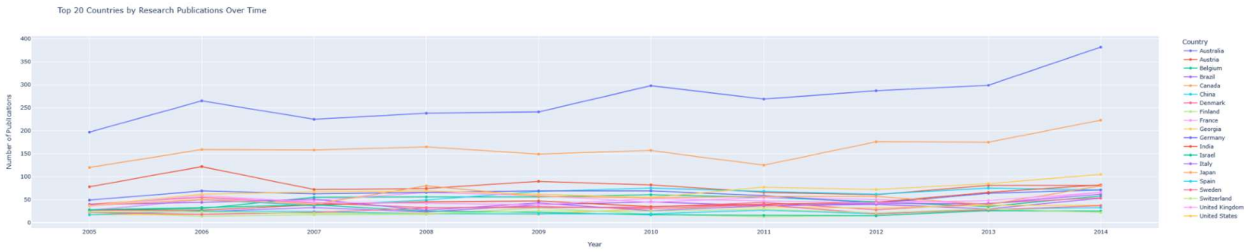
High-Risk Groups	Adolescents, Military Personnel, LGBTQ+	Certain populations are more vulnerable and require targeted prevention efforts.
Most Studied Risk Factors	Depression, Substance Abuse, Social Isolation	Mental health conditions and social factors are key contributors to suicide risk.

This table highlights key research fields in suicide studies from SafetyLit data, focusing on methodology, language, prevention strategies, high-risk groups, and risk factors.

Conclusion:

This data suggests that suicide research is largely qualitative, English-based, and focused on high-risk populations and mental health risk factors. Expanding multilingual studies and quantitative approaches could improve global suicide prevention efforts.

Top 20 Countries by Research Publications Overtime



This line graph depicts the top 20 countries with the most publications during 2005 to 2014, based on Safetylit data.

Key Insights:

- **Research Language:** As expected, English speaking countries tend to publish the most articles. Given that 80.3% of our dataset is in English, this is not surprising.
- **Top Three:** Australia and then Canada are by far the most prolific countries followed by the U.S. in terms of number of publications.
- **What do these three have in common?:** If we compare this to WHO reporting on suicide by country, it is less suprising as suicide is more common in Australia,

Canada and the U.S. and those countries have the resources to pay for research.

Conclusion:

Australia, Canada and the U.S. contributed the most suicide-related research in this time frame. This makes intuitive sense as mentioned above. An interesting correlation is that in these countries, opioid usage became more widely talked about as Canada and Australia spiked in the last year. It is also interesting that the other countries on the chart have stayed more or less constant.

Addressing Feedback

To address the feedback received from previous Milestones and incorporate key insights from our discussions with Eric, we have refined our approach to improve clarity, efficiency, and scalability in our data collection and analysis.

1. **Clarifying Challenges and Solutions**
    - We refined our explanations of the challenges encountered during web scraping, ensuring that each issue is clearly articulated alongside the corresponding solution.
    - Emphasis was placed on explaining the technical decisions made to handle pagination, metadata inconsistencies, and dynamic content extraction more effectively.
2. **Optimizing the Web Scraper**
    - To improve efficiency, we enhanced the scraper to handle dynamic content more effectively and streamlined API calls to minimize processing time.
    - Additional refinements were made to reduce redundancy in data retrieval and ensure consistency across extracted fields.
3. **Capturing All Required Fields**
    - During our call, Eric emphasized the importance of ensuring that all fields listed in the provided document were captured.
    - To achieve this, we modified our web scraper to extract all necessary metadata and validated its completeness.
    - Additionally, we cross-checked the number of captured articles against the visual Eric provided, which outlined the expected number of articles per year. This verification step ensured our data was as accurate as possible and aligned with the reference dataset.

4. **Handling OCR for PDFs**
    - Since some abstracts were missing, we explored the integration of OCRbased extraction methods (e.g., Tesseract OCR or Adobe PDF extraction tools) as a potential solution.
    - A feasibility assessment was conducted to determine whether OCR processing was necessary for a significant portion of missing abstracts.
5. **Preserving All Data and Flagging Inconsistencies**
    - Rather than removing records or deduplicating aggressively, we chose to retain all entries and instead flag issues like semantic duplication, metadata mismatch, and formatting inconsistencies.
    - This approach ensures maximum transparency and data retention, enabling future analysts to make informed decisions about inclusion/exclusion.
    - It also supports deeper insights into the structure and limitations of the SafetyLit dataset, as many inconsistencies were not clearly detectable through automated methods alone.

FEATURE ENGINEERING

Across all modeling notebooks, extensive feature engineering was conducted to transform raw academic publication data into structured features. These included:

- Text Features:
    - Abstracts, Titles, and Journal names were combined and vectorized using TF-IDF or tokenized for embedding layers in deep learning models.

- Text columns like keywords, categories, and authors were cleaned, tokenized, and used in multi-hot encoding or embeddings. ○ After stemming and lemmatization for columns: title, journal and abstract, POS tagging was applied as a precondition for Named Entity Recognition

- Categorical Features:
  - Features like language, journal, and data_source were either one-hot encoded or passed directly into model pipelines depending on their nature. ○ Keywords and categories were extracted, cleaned of special characters, and one-hot encoded using pd.get_dummies().
- Grant Parsing:
  - The grant_info column was parsed using regular expressions to extract GrantID, Agency, and Country — each stored in separate fields.
- Affiliation Parsing:
  - Custom NLP pipelines using spaCy were developed to extract institutional information, departments, email addresses, and locations from unstructured affiliation data. Benchmarked edge cases were hard-coded for robustness.

MODEL SELECTION & CROSS-VALIDATION

Across all modeling workflows, model selection was guided by relevance to each research question and the nature of the prediction task. For classification problems, we explored both traditional machine learning models (Logistic Regression, Random Forest, Support Vector Machine) and deep learning approaches (LSTM, BERT + Logistic Regression), while regression-oriented tasks like publication year prediction, we included models capable of handling multiclass outputs. Stratified train-validation-test splits were used to preserve class distribution across datasets, and k-fold cross-validation (typically

4- or 5-fold) was employed within hyperparameter tuning pipelines to ensure robustness and minimize overfitting. Model comparison was primarily based on validation F1-score or accuracy, depending on the task, and best-performing configurations were carried forward for final evaluation on test data.

HYPERPARAMETER TUNING

Grid search was the primary approach that was utilized in tuning our models. It is a powerful approach since it thoroughly searches through all specified parameters. Therefore, this ensures that all options are explored for the best combination. Grid search guarantees that any potential optimal combination is not overlooked. Although an exhaustive approach, it applied well to our relatively small dataset size. If this was a larger dataset, then an alternative approach would be random search, which requires less computational resources and proves to be more efficient. However, a disadvantage in using random search is that it may not provide the optimal combination. Grid search is especially useful for models with multiple hyperparameters as it performs a detailed inspection of the search space. Overall, grid search provides a reliable way to improve model accuracy.

MODEL DEVELOPMENT & EVALUATION

Before building our predictive and analytical models, we focused on designing a consistent and interpretable feature extraction pipeline. Given the nature of our dataset (text-heavy metadata and abstracts from suicide prevention studies), we

chose Term Frequency-Inverse Document Frequency (TF-IDF) as our primary vectorization method. TF-IDF was selected for its balance between simplicity and effectiveness: it not only captures the importance of terms within individual documents but also discounts commonly occurring words across the corpus, which is particularly valuable in a domain where general terms like "suicide" or "intervention" appear frequently. This approach allowed our models to focus on more nuanced, contextspecific language, improving performance across the time series forecasting, metaanalysis, and survival analysis components of the project.

For most of our Research Questions we used 2014 as our chosen year for modeling as 2014 was the year we fully ensured data completion for.

RQ1: Machine Learning for Suicide Research Trends (Year Prediction)[1]

To investigate whether abstract text could predict publication year , we developed a series of machine learning models using lemmatized abstract tokens and TF-IDF features. This approach was chosen for its simplicity, interpretability, and efficiency in handling high-dimensional sparse text data. We tested three traditional models—Logistic Regression, Random Forest, and Support Vector Machine—to assess which best captured temporal patterns in the literature. Among them, Random Forest performed best, achieving 26% accuracy on the test set, notably outperforming Logistic Regression and SVM (both ~20%). However, even the top model struggled to generalize beyond detecting recent publication years, likely due to overlapping language across time. While not highly predictive, these results suggest subtle temporal signals exist in abstract phrasing, warranting future refinement.

RQ1: Deep Learning for Suicide Research Trends (Year Prediction) [2]

To explore how the language in suicide-related research has evolved over time, we developed deep learning models to predict the publication year of each article based on its abstract. We compared two architectures: BERT embeddings paired with Logistic Regression, and an LSTM-based neural network trained directly on tokenized text. BERT embeddings were extracted using the [CLS] token representation and fed into a logistic regression model, achieving a best cross-validated accuracy of 29% using the saga solver and L2 regularization. This could be due to the complexity of the text data, which

1 *Note: RQ1 is redundant when focusing on 2014 as our chosen year but we opted to keep section below as it pertains to our previous research on data from all years.*

2 *Note: RQ1 is redundant when focusing on 2014 as our chosen year but we opted to keep section below as it pertains to our previous research on data from all years.*

includes long abstracts. While performance was modest, the model showed potential for capturing temporal language shifts in the corpus. The LSTM model, trained for five epochs, showed steadily improving training accuracy (up to 57.7%) but low validation accuracy (25.4%), likely due to overfitting and limited training size. Both models illustrate

the challenge of learning subtle year-specific signals in academic text, but they provide a starting point for modeling temporal trends using deep learning methods. Hyperparameters, model outputs, and embeddings were stored for reproducibility.

```
Best parameters: {'logreg__C': 0.1, 'logreg__max_iter': 100, 'logreg__penalty': 'l2', 'logreg__solver': 'saga'}
Best score: 0.29000000000000004
```

	precision	recall	f1-score	support
2005	0.0000	0.0000	0.0000	12
2006	0.0000	0.0000	0.0000	10
2007	0.0000	0.0000	0.0000	6
2008	0.1667	0.1667	0.1667	12
2009	0.1429	0.0909	0.1111	11
2010	0.1000	0.1429	0.1176	7
2011	0.0000	0.0000	0.0000	6
2012	0.1429	0.1000	0.1176	10
2013	0.1667	0.3636	0.2286	11
2014	0.1892	0.4667	0.2692	15
accuracy			0.1600	100
macro avg	0.0908	0.1331	0.1011	100
weighted avg	0.1037	0.1600	0.1177	100

Figure: Logistic Regression with BERT Embeddings

Classification report shows low scores of <0.50 across all metrics.

```
Epoch 1/5
726/726 ━━━━━━━━━━━━━━━━━━━━ 167s 226ms/step - accuracy: 0.1666 - loss: 2.2214 - val_accuracy: 0.2099 - val_loss: 2.0907
Epoch 2/5
726/726 ━━━━━━━━━━━━━━━━━━━━ 200s 224ms/step - accuracy: 0.2700 - loss: 1.9836 - val_accuracy: 0.2581 - val_loss: 2.0125
Epoch 3/5
726/726 ━━━━━━━━━━━━━━━━━━━━ 204s 226ms/step - accuracy: 0.3840 - loss: 1.7347 - val_accuracy: 0.2436 - val_loss: 2.1108
Epoch 4/5
726/726 ━━━━━━━━━━━━━━━━━━━━ 202s 226ms/step - accuracy: 0.4885 - loss: 1.4687 - val_accuracy: 0.2471 - val_loss: 2.2915
Epoch 5/5
726/726 ━━━━━━━━━━━━━━━━━━━━ 203s 227ms/step - accuracy: 0.5769 - loss: 1.2199 - val_accuracy: 0.2543 - val_loss: 2.5214
```

Figure: LSTM Training vs. Validation Accuracy over Epochs

Visually shows an overfitting trend — strong training gains, but flat or declining validation accuracy.

RQ2. Gender Focus in Suicide Research (DL)

Across all six models tested for classifying gender focus in suicide-related research, a consistent trend emerges: strong performance on male-focused content and significant difficulty in identifying female-focused studies. The RNN-based models all demonstrated high accuracy and strong F1-scores for classifying male-oriented texts. However, they struggled to detect female-focused abstracts, with recall scores as low as 0.05–0.07 for the female class. Overfitting was evident, particularly in the GRU and LSTM models, where validation accuracy plateaued or declined while training accuracy continued to rise. This suggests that the linguistic features or frequency of male-focused studies in the dataset dominated the models' learning process, making them less effective at generalizing across underrepresented or more variable female-focused content.