Cancer Atlas
Software Documentation

Itay Zahor, Shmuel Granot.

Table Of Contents

Project Idea:

The Cancer Atlas project aims to analyze the relationships between socioeconomic status, environmental factors, risk factors, and cancer rates across U.S. states. By integrating diverse datasets and performing in-depth analyses, the project provides insights that help scientists and policymakers make data-driven decisions.

Key Features:

- Analysis of over 45 million cancer cases (incidence and mortality).
- Advanced visualizations, such as heatmaps, to explore trends and disparities.
- Comprehensive filtering options based on demographics, socioeconomic data, and environmental metrics.
- Server-side processing for efficient data handling and complex calculations.

Project Goals:

- Provide a robust platform for understanding the factors influencing cancer outcomes.
- Enable the exploration of patterns within health, environmental, and economic data.
- Support research and policymaking through actionable insights and user-friendly tools.

GitHub Repository:

The entire project, including its source code and documentation, is hosted on GitHub:

https://github.com/itayzahor/Cancer_Atlas_app

About The Data

The data utilized in the Cancer Atlas app is sourced from reliable government and academic organizations. Below is an overview of the datasets and their significance:

Cancer Data

Source: U.S. Cancer Statistics Data Visualizations Tool, provided by the CDC.
https://www.cdc.gov/united-states-cancer-statistics/dataviz/data-tables.html
The "By Area" table was used, which includes fields such as area, count, event type, race, sex, site, and year.
Data covers over 98% of the U.S. population from 1999 to 2021.

Purpose:
This dataset serves as a foundation for analyzing geographic disparities and understanding cancer incidence and mortality trends. It is the basis of our site and where we draw all the cancer cases from.

Data Handling:
To ensure the Cancer Atlas app uses accurate and clean data, we performed the following data preprocessing steps on the cancer dataset:

Column Removal:
Removed unnecessary columns that were not relevant to our analysis or visualizations.

Row Cleaning:

Deleted rows where the count field contained invalid values such as 0, -, ~, or +, as these typically represent low or unreliable case counts.
Excluded rows where SITE was labeled as "All Cancer Sites Combined," RACE was "ALL RACES," or other rows that presented aggregated data.
Removed rows where YEAR represented aggregated time ranges, such as "2018–2022" or "2017–2021," to focus on annual data.
Removed rows for Washington D.C., as the dataset already included data for the District of Columbia.
Combined the data for Washington D.C. with the District of Columbia to avoid duplication.
Note: This adjustment may explain why the District of Columbia appears unusual in some visualizations.

Field Mapping:

Standardized column values to ensure consistency:
AREA was replaced with state_id.
RACE was replaced with race_id.
SITE was replaced with cancer_type_id.
EVENT_TYPE was replaced with numeric values:
0 for Mortality.
1 for Incidence.
SEX was replaced with numeric values:
0 for Male.
1 for Female.

Races Table

Created a races table by extracting unique values from the RACE column in the cancer dataset.
Corrected race names to make them more understandable and user-friendly. For example:
Changed "Non-Hispanic White" to "White (Non-Hispanic)."
Each race was assigned a unique ID for consistent referencing.

Cancer Types

Created a cancer_types table by extracting unique values from the SITE column in the cancer dataset.
Assigned a unique ID to each cancer type for use as a foreign key in other tables.

States

Sources: GitHub repository
https://gist.github.com/CollectionOfAtoms/a68304810fa6c331fa997f9f08312408
U.S. States with Latitude and Longitude.
This dataset includes:
State Names (e.g., "Alabama")
State Abbreviations (e.g., "AL")
Latitude and Longitude coordinates for each state

Purpose:
The states table serves two main purposes:

State Abbreviations, Latitude, and Longitude:
These variables are used for accurate geographic visualizations in the app, ensuring that data is mapped correctly to U.S. states in the heatmap.

State IDs and Names:
These form the basis for state-level analysis, supporting consistent and efficient data organization throughout the Cancer Atlas app.

Data Handling:
Extracting Distinct State Names from the cancer_data table to ensure compatibility with the cancer dataset.

Each state was assigned a unique ID, which serves as a primary key for integration with other tables in the database.

Joined the dataset from GitHub with the distinct state names extracted from cancer_data, using the state name as the matching field.

Socioeconomic Data

Sources: U.S. Census Bureau datasets, including:
https://data.census.gov/table/ACSST1Y2023.S2301?q=Employment%20&g=010XX00US$0400000&moe=false
Employment Data (2023 ACS): Provides unemployment rates for population over 16 years for each state.

Health Insurance Data (2023 ACS):
https://data.census.gov/table/ACSST1Y2023.S2701?t=Health%20Insurance&g=010XX00US$0400000&moe=false&tp=false
Reports the percentage of insured individuals by state.

Median Income Data (2023 ACS):
https://data.census.gov/table/ACSST1Y2023.S1901?q=median%20income%20by%20state&g=010XX00US$0400000&moe=false
Offers median income statistics for each state.

Purpose:
Combining unemployment rates, insurance coverage, and income levels allows for the exploration of socioeconomic factors influencing cancer outcomes.

Data Handling:
Column Transformation

Transformed specific columns into standardized formats. For instance, the column Alabama!!Unemployment rate!!Estimate was mapped to its corresponding state_id to align with the database schema.

Data Integration

The cleaned and transformed data was then inserted into the socioeconomic_data table in the database. This ensured the data was readily accessible for querying and analysis.

Risk Factors Data

Source: CDC datasets, including:
Cigarette Usage Among Adults (2019):
https://www.cdc.gov/statesystem/cigaretteuseadult.html
State-level estimates of adult smoking rates.
Note: This dataset is less robust as it is drawn from relatively small sample sizes.

Physical Inactivity Prevalence (2017–2020):
https://www.cdc.gov/physical-activity/php/data/inactivity-maps.html
Prevalence rates of physical inactivity by state.
Prevalence is defined as the percentage of adults who report not engaging in leisure-time physical activity.

Purpose:
Analyzing these behavioral risk factors provides insights into how lifestyle choices influence cancer incidence and disparities. Specifically:

Smoking Rates: Help identify correlations between smoking prevalence and cancer outcomes.
Physical Inactivity Rates: Offer insights into health risks related to lifestyle and their potential link to cancer incidence.

Data Handling:

Smoking Rates:
Used the Data_Value field (e.g., "20.2" for Alabama) as the smoking rate for each state.
Deleted other rows, including confidence limits and sample size, to focus on the core data.
Assigned a unique state_id to each smoking rate by joining the smoking data with the states table using the state name.

Handling Missing Data:
New Jersey was missing from the dataset, so its smoking rate was assigned the average value of the other states.

Physical Inactivity:
Assigned a unique state_id by joining the inactivity data with the states table using the state name.

Eventually, We combined the tables.

Environmental Data

Source: https://www.epa.gov/outdoor-air-quality-data
Air Quality Index (AQI)(2024): EPA-provided data on daily AQI readings, averaged at the state level.

CO_2 Emissions (2022): https://www.eia.gov/environment/emissions/state/
State-level metric tons of energy-related carbon dioxide (CO_2) emissions per capita from the U.S. Energy Information Administration.

Purpose:
These metrics help investigate environmental quality and its potential links to cancer incidence.
AQI: Represents air quality conditions, highlighting states with higher pollution levels.
CO_2 Emissions: Reflects energy consumption patterns and environmental footprints, offering insights into long-term health risks.

Data handling:

Air Quality Index (AQI)
Focused on the Median AQI field, which provides a robust representation of typical air quality while minimizing the influence of outliers.
Calculated the average of the Median AQI values for all counties within each state to derive state-level AQI data.
Assigned a unique state_id to each state by joining the AQI data with the states table using the state name.

CO_2 Emissions
Assigned a unique state_id to each state by joining the CO_2 emissions data with the states table using the state name.

Eventually, We combined the tables.

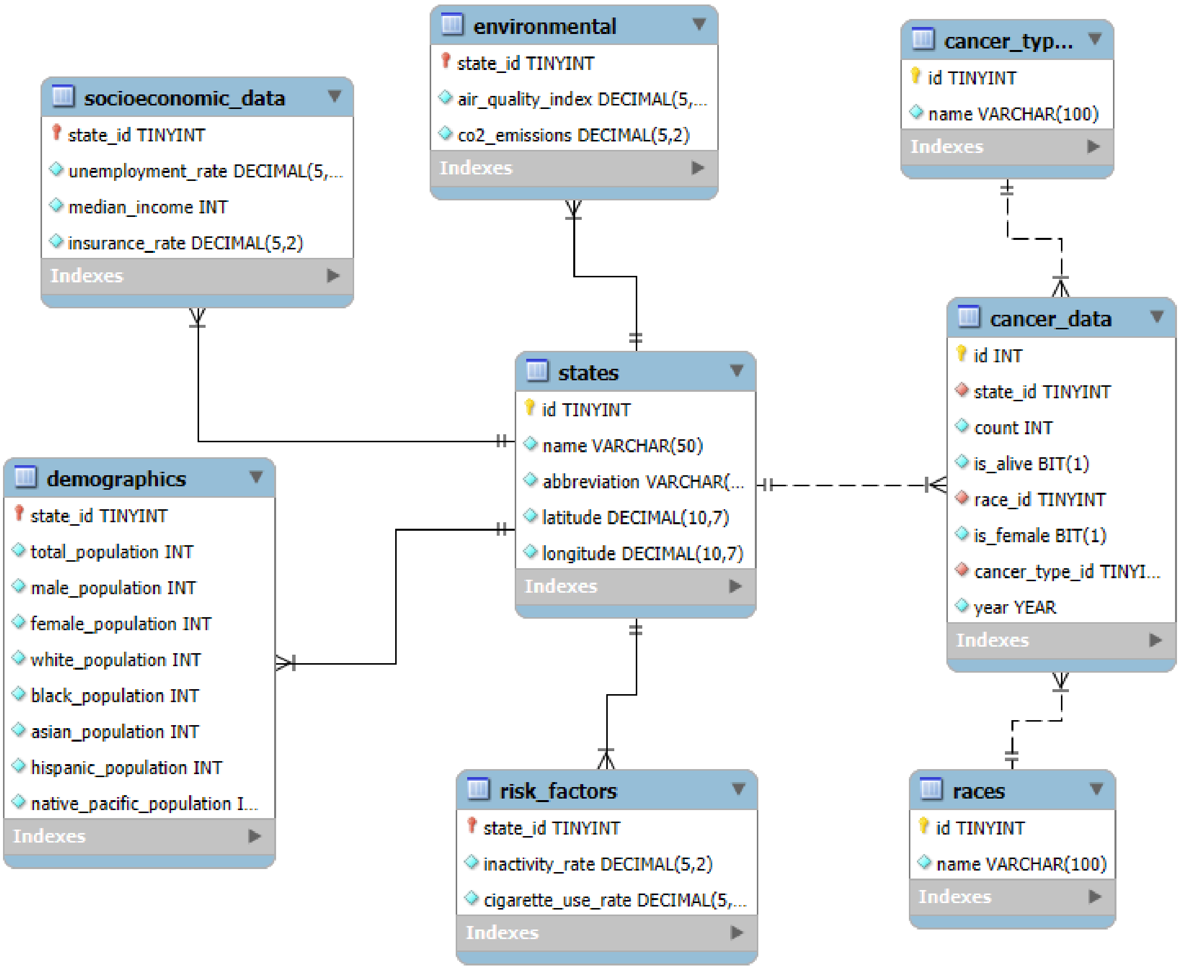
Demographics Data

Source: Demographics Table (2023 ACS).
https://data.census.gov/table/ACSDP1Y2023.DP05?q=race%20and%20gender%20rates%20by%20state&t=Populations%20and%20People
Includes state-level statistics on:
Total population
Male and female populations
Race-specific population counts

Purpose:
Demographics data ensures accurate rate calculations by aligning cancer data with population filters (e.g., race or gender). This supports more precise analysis of disparities and trends in cancer outcomes.

Data Handling:
Included only the races available in the cancer_data table for consistency with the cancer dataset.
Retained fields for: Total population, Male and female populations and Race-specific populations (e.g., White, Black or African American, Asian, Hispanic)
Processed labels such as "Alabama!!Estimate" into a clean format by extracting the state name and assigning a corresponding state_id using the states table.

Scheme



environmental
state_id TINYINT
air_quality_index DECIMAL(5,...
co2_emissions DECIMAL(5,2)
Indexes
cancer_typ...
id TINYINT
name VARCHAR(100)
Indexes
socioeconomic_data
state_id TINYINT
unemployment_rate DECIMAL(5,...
median_income INT
insurance_rate DECIMAL(5,2)
Indexes
cancer_data
id INT
state_id TINYINT
count INT
is_alive BIT(1)
race_id TINYINT
is_female BIT(1)
cancer_type_id TINYI...
year YEAR
Indexes
states
id TINYINT
name VARCHAR(50)
abbreviation VARCHAR(...
latitude DECIMAL(10,7)
longitude DECIMAL(10,7)
Indexes
demographics
state_id TINYINT
total_population INT
male_population INT
female_population INT
white_population INT
black_population INT
asian_population INT
hispanic_population INT
native_pacific_population I...
Indexes
risk_factors
state_id TINYINT
inactivity_rate DECIMAL(5,2)
cigarette_use_rate DECIMAL(5,...
Indexes
races
id TINYINT
name VARCHAR(100)
Indexes

Schema Documentation

cancer_data

Primary Key (PK):
id: A unique identifier for each record.

Foreign Keys (FKs):
state_id: → states.id.
race_id:→ races.id.
cancer_type_id: → cancer_types.id.

Fields:
id (int unsigned): A unique identifier for the record (auto-incremented, not nullable, unique).
state_id (tinyint unsigned): The ID of the state (foreign key, not nullable).
count (int unsigned): The number of cases or deaths recorded (not nullable).
is_alive (bit(1)): Indicates the event type (0 = Mortality, 1 = Incidence; not nullable).
race_id (tinyint unsigned): The ID of the race (foreign key, not nullable).
is_female (bit(1)): Indicates gender (0 = Male, 1 = Female; not nullable).
cancer_type_id (tinyint unsigned): The ID of the cancer type (foreign key, not nullable).
year (year): The year the record corresponds to (not nullable).

races

Primary Key (PK):
id: A unique identifier for each race.

Fields:
id (tinyint unsigned): A unique identifier for the race (auto-incremented, not nullable, unique).
name (varchar(100)): The name of the race (unique, not nullable).

cancer_types

Primary Key (PK):
id: A unique identifier for each cancer type.

Fields:
id (tinyint unsigned): A unique identifier for the cancer type (auto-incremented,unique , not nullable).
name (varchar(100)): The name of the cancer type (unique, not nullable).

states

Primary Key (PK):
id: A unique identifier for each state.

Fields:
id (tinyint unsigned): A unique identifier for the state (auto-incremented, not nullable, unique).
name (varchar(50)): The full name of the state (unique, not nullable).
abbreviation (varchar(2)): The state abbreviation (unique, not nullable).
latitude (decimal(10,7)): The latitude coordinate of the state (not nullable).
longitude (decimal(10,7)): The longitude coordinate of the state (not nullable).

socioeconomic_data

Primary Key (PK):
state_id: A unique identifier for each state.

Foreign Keys (FKs):
state_id: → states.id.

Fields:
state_id (tinyint unsigned): The ID of the state (foreign key, not nullable, unique).
unemployment_rate (decimal(5,2) unsigned): The unemployment rate in the state for people over 16 (not nullable).
median_income (int unsigned): The median income in the state (not nullable).
insurance_rate (decimal(5,2) unsigned): The percentage of insured individuals in the state (not nullable).

risk_factors

Primary Key (PK):
state_id: A unique identifier for each state.

Foreign Keys (FKs):
state_id: → states.id.

Fields:
state_id (tinyint unsigned): The ID of the state (foreign key, not nullable, unique).
inactivity_rate (decimal(5,2) unsigned): The percentage of adults reporting no leisure-time physical activity (not nullable).
cigarette_use_rate (decimal(5,2) unsigned): The percentage of adults reporting cigarette use (not nullable).

environmental

Primary Key (PK):
state_id: A unique identifier for each state.

Foreign Keys (FKs):
state_id: → states.id.

Fields:
state_id (tinyint unsigned): The ID of the state (foreign key, not nullable, unique).
air_quality_index (decimal(5,2) unsigned): The median AQI (Air quality index) value for the state (not nullable).
co2_emissions (decimal(5,2) unsigned): metric tons of energy-related carbon dioxide (CO2) emissions per capita (not nullable).

demographics

Primary Key (PK):
state_id: A unique identifier for each state.

Foreign Keys (FKs):
state_id: → states.id.

Fields:
state_id (tinyint unsigned): The ID of the state (foreign key, not nullable, unique).
total_population (int unsigned): Total population in the state (not nullable).
male_population (int unsigned): Male population in the state (not nullable).
female_population (int unsigned): Female population in the state (not nullable).
white_population (int unsigned): White population in the state (not nullable).
black_population (int unsigned): Black or African American population in the state (not nullable).
asian_population (int unsigned): Asian population in the state (not nullable).
hispanic_population (int unsigned): Hispanic population in the state (not nullable).
native_pacific_population (int unsigned): Native Hawaiian or Pacific Islander population in the state (not nullable).

Indexes

Automatically Generated Indexes:

These indexes are created by MySQL automatically when defining primary keys, foreign keys, and unique constraints. Their purpose is to:

Primary Keys (PRIMARY): Ensure each record in the table is uniquely identifiable and enable fast lookups.
Foreign Keys (fk_...): Support relationships between tables by indexing the referencing columns.
Unique Constraints (..._UNIQUE): Prevent duplicate entries in columns that must contain unique values.

cancer_data:
PRIMARY: Index on id.
fk_cancer_races: Index on race_id.
fk_cancer_types: Index on cancer_type_id.
fk_states: Index on state_id.

races
PRIMARY: Index on id.

states
PRIMARY: Index on id.

cancer_types:
PRIMARY: Index on id.

demographics:
PRIMARY: Index on state_id (links to the states table).

environmental
PRIMARY: Index on state_id (links to the states table).

risk_factors
PRIMARY: Index on state_id (links to the states table).

socioeconomic_data
PRIMARY: Index on state_id (links to the states table).

Manually Created Indexes

These indexes were specifically added to optimize query performance for frequently used filters or joins. By targeting key query patterns, these indexes enhance the efficiency of data retrieval operations.

idx_cancer_data_composite
Type: Composite Index
Columns: is_alive, cancer_type_id, state_id
Purpose:
Optimizes the performance of queries used in key analyses within the app, Such as
Socio Economic Impact Analysis, Environmental Impact Analysis, Risk Factors Impact Analysis.
Because, it first filters by the is_alive status (e.g., mortality or incidence) used in all of them.
Then filters by cancer_type_id when a specific cancer type is selected.
And in the end, assists in the GROUP BY state_id to complete the analysis.

idx_cancer_data_year
Type: Single-Column Index
Column: year
Purpose:
Improves the performance of queries fetching distinct years from the cancer data, such as populating the years dropdown for filtering or reports.

Queries

The application utilizes three simple queries and four complex query templates. The simple queries handle straightforward data retrieval tasks, while the complex query templates are designed for flexibility and reusability, enabling the creation of numerous dynamic queries on the website.

By using templates for complex queries, we avoid duplicating code and ensure that the queries can be easily adjusted to meet different analytical needs. This approach balances maintainability with the ability to handle diverse user requirements efficiently.

Hopefully it fits your requirement.

Simple Queries

Fetch Cancer Types

```
“””
SELECT id, name
FROM cancer_types
ORDER BY name;
“””
```

Purpose:
This query retrieves all the available cancer types from the cancer_types table, sorted alphabetically.
It is used to populate dropdown menus in the app where users can filter or select specific cancer types.

Fetch Races

```
“””
SELECT id, name
FROM races
ORDER BY name;
“””
```

Purpose:
This query retrieves all the race categories from the races table, sorted alphabetically.
It is used to populate dropdown menus in the app, allowing users to filter or analyze data based on specific racial groups.

Fetch Years

"""
SELECT DISTINCT year
FROM cancer_data
ORDER BY year;
"""

Purpose:
This query retrieves all the distinct years from the cancer_data table, sorted in ascending order.
It is used to populate dropdown menus in the app, enabling users to filter data or analyses by specific years.

Complex queries

Construct Heatmap

This query is intentionally complex because we prioritized performing calculations on the server rather than in the application code. While this approach makes the query more challenging to read, it ensures that all rates and normalized rates are calculated directly within the database, reducing the processing load on the application.

Originally envisioned as a simple query, its complexity grew as we integrated rate calculations and normalization directly into the query logic. This design decision enhances performance and centralizes the computational logic, even if it makes the query more intricate.

Input:

Filters: default is “-” for the All option.
cancer_type: (int) ID of the cancer type or “-” for all cancer types.
year: (int) Year to filter data.
is_female: (string) Gender filter (1 for Female, 0 for Male, or "-" for all genders).
is_alive: (string) Filter for incidence or mortality (1 for Incidence, 0 for Mortality, or "-" for both).
race_id: (int) Race ID or “-” for all races.

Advanced filters: in case of None received then there is no limit.
unemployement_min, unemployement_max: (float) Range for unemployment rate.
median_min, median_max: (int) Range for median income.
insurance_min, insurance_max: (float) Range for insurance rate.
inactivity_min, inactivity_max: (float) Range for inactivity rate.
cigarette_min, cigarette_max: (float) Range for cigarette use rate.
aqi_min, aqi_max: (float) Range for air quality index.
co2_min, co2_max: (float) Range for CO2 emissions.

Output:

name: (string) Name of the state.
abbreviation: (string) State abbreviation (used for the map).
latitude: (float) Latitude of the state (used for the map).
longitude: (float) Longitude of the state (used for the map).
total_count: (int) Total cases or deaths in the state.
filtered_population: (float) Subset of the population based on filters.
rate: (float) Percentage of total cases relative to the filtered population.
normalized_rate: (float) Standardized rate (between 0 and 1) for visual comparison in the heatmap.

Template Query:

```
WITH base_data AS (
SELECT
    s.name,
    s.abbreviation,
    s.latitude,
    s.longitude,
    COALESCE(SUM(c.count), 0) AS total_count,
    {filtered_population} AS filtered_population
FROM states s
LEFT JOIN cancer_data c ON s.id = c.state_id
LEFT JOIN demographics d ON s.id = d.state_id
LEFT JOIN socioeconomic_data sd ON s.id = sd.state_id
LEFT JOIN risk_factors rf ON s.id = rf.state_id
LEFT JOIN environmental e ON s.id = e.state_id
WHERE 1=1
    {filter_conditions}
GROUP BY s.id
),
rate_data AS (
SELECT
    base_data.*,
    ROUND(
        CASE
            WHEN total_count > 0 AND filtered_population > 0 THEN
                (total_count / filtered_population) * 100
            ELSE 0
        END, 2
    ) AS rate
FROM base_data
),
final_data AS (
SELECT
    rate_data.*,
    CASE
        WHEN rate_bounds.max_rate > rate_bounds.min_rate THEN
            (rate_data.rate - rate_bounds.min_rate) / (rate_bounds.max_rate -
rate_bounds.min_rate)
        ELSE 0
    END AS normalized_rate
FROM rate_data,
    (SELECT MIN(rate) AS min_rate, MAX(rate) AS max_rate FROM rate_data) AS
rate_bounds
)
SELECT * FROM final_data;
```

Example Query

“””

```
WITH base_data AS (
          SELECT
             s.name,
             s.abbreviation,
             s.latitude,
             s.longitude,
             COALESCE(SUM(c.count), 0) AS total_count,
             d.white_population * (d.female_population / NULLIF(d.total_population, 0)) AS
filtered_population
          FROM states s
          LEFT JOIN cancer_data c ON s.id = c.state_id
    AND c.cancer_type_id = 6
AND c.year = 2020
AND c.is_female = 1
AND c.is_alive = 1
AND c.race_id = 5
       LEFT JOIN demographics d ON s.id = d.state_id
       LEFT JOIN socioeconomic_data sd ON s.id = sd.state_id
       LEFT JOIN risk_factors rf ON s.id = rf.state_id
       LEFT JOIN environmental e ON s.id = e.state_id
       WHERE 1=1
    AND sd.unemployment_rate >= 3.0 AND sd.unemployment_rate <= 10.0
AND sd.median_income >= 50000 AND sd.median_income <= 100000
AND sd.insurance_rate >= 80.0 AND sd.insurance_rate <= 100.0
AND rf.inactivity_rate >= 15.0 AND rf.inactivity_rate <= 50.0
AND rf.cigarette_use_rate >= 10.0 AND rf.cigarette_use_rate <= 50.0
AND e.air_quality_index >= 10.0 AND e.air_quality_index <= 50.0
AND e.co2_emissions >= 20.0 AND e.co2_emissions <= 50.0
       GROUP BY
          s.id
       ),

       rate_data AS (
       SELECT
          base_data.*,
          ROUND(
             CASE
                WHEN total_count > 0 AND filtered_population > 0 THEN
                   (total_count / filtered_population) * 100
                ELSE 0
             END, 2
          ) AS rate
       FROM base_data
   ),
```

```
    final_data AS (
      SELECT
        rate_data.*,
        CASE
          WHEN rate_bounds.max_rate > rate_bounds.min_rate THEN
            (rate_data.rate - rate_bounds.min_rate) / (rate_bounds.max_rate -
rate_bounds.min_rate)
          ELSE 0
        END AS normalized_rate
      FROM rate_data,
        (SELECT MIN(rate) AS min_rate, MAX(rate) AS max_rate FROM rate_data) AS
rate_bounds
    )
    SELECT * FROM final_data;

“””
```

Explanation

base_data CTE:

Purpose:
This is the first step of the query, where we gather raw data by combining states with other relevant tables (cancer_data, demographics, socioeconomic_data, risk_factors, and environmental).
It calculates:
total_count: The total number of cancer cases or deaths for each state based on the filters provided.
filtered_population: A dynamically calculated subset of the state population, based on race and gender filters.

Joins:
states is joined with:
cancer_data (to get case counts).
demographics (to calculate population by race and gender).
socioeconomic_data (for unemployment, income, and insurance).
risk_factors (for inactivity and smoking rates).
environmental (for air quality and CO2 emissions).

Filters:
The query applies filters for:
Cancer Data: Filters like cancer type, year, gender, survival status, and race.
Socioeconomic Factors: E.g., unemployment rate, median income, insurance rate.
Risk Factors: E.g., inactivity and cigarette use rates.
Environmental Factors: E.g., air quality index (AQI) and CO2 emissions.
Calculations:
filtered_population:
Formula:
{race_population} * (d.gender_population / NULLIF(d.total_population, 0))
Explanation:
race_population is selected dynamically based on the race filter.
If a gender filter is applied (is_female), the population is further adjusted based on the gender ratio.
NULLIF ensures no division by zero.

Results:
This step aggregates and prepares the foundational data, which will later be used to calculate cancer rates and normalized values.

rate_data CTE:

Purpose:
This CTE calculates the cancer rate for each state.
The cancer rate represents the percentage of cancer cases relative to the filtered population.

Rate Formula:
```
ROUND(
  CASE
    WHEN total_count > 0 AND filtered_population > 0 THEN
      (total_count / filtered_population) * 100
    ELSE 0
  END, 2
) AS rate
```

Explanation:
total_count: The total number of cancer cases or deaths in the state.
filtered_population: The relevant subset of the population (adjusted by race and gender).
The rate is expressed as a percentage.
If either total_count or filtered_population is 0, the rate defaults to 0 to prevent invalid calculations.

Results:
This step derives the cancer rate for each state, providing a core metric for the heatmap.

final_data CTE:

Purpose:
This CTE normalizes the cancer rate for each state, scaling it to a range of 0 to 1 for better comparison.

Normalization Formula:

```
CASE
   WHEN rate_bounds.max_rate > rate_bounds.min_rate THEN
      (rate_data.rate - rate_bounds.min_rate) / (rate_bounds.max_rate - rate_bounds.min_rate)
   ELSE 0
END AS normalized_rate
```

Explanation:
rate_bounds.max_rate and rate_bounds.min_rate are the highest and lowest cancer rates across all states.
Normalization ensures all rates are scaled between 0 and 1, enabling intuitive visual representation on the heatmap.
If all rates are equal (max_rate = min_rate), the normalized value defaults to 0.

Results:
This step prepares the data for visualization, ensuring consistent scaling across different filters.

Summary:
base_data: Gathers and aggregates raw data, calculates filtered population.
rate_data: Calculates the cancer rate as a percentage of the filtered population.
final_data: Normalizes the cancer rate for visualization.

The query is so complex because we wanted the calculations to be on the server and not
in the code itself. That is why this query is a nightmare to read.

Socioeconomic_vs_mortality

Input:
cancer_type: The type of cancer to filter by. If "-" is provided, no filter is applied, and all cancer types are included.
factor: A column name from the socioeconomic_data table (e.g., median_income, unemployment_rate). Determines which socioeconomic factor to analyze in relation to cancer mortality rates.

Output:
Fields Returned:
state: The name of the state (from states.name).
socioeconomic_factor: The value of the specified socioeconomic factor for the state (from socioeconomic_data).
mortality_rate: The calculated mortality rate per 100 people in the state.

Template Query:

```
SELECT
    s.name AS state,
    se.{factor} AS socioeconomic_factor,
    mortality.mortality_rate AS mortality_rate
FROM
    (SELECT
        c.state_id,
        SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS mortality_rate
     FROM
        cancer_data c
     JOIN
        demographics d ON c.state_id = d.state_id
     WHERE
        c.is_alive = 0  -- Only mortality data
        {cancer_type_filter}
     GROUP BY c.state_id
    ) AS mortality
JOIN
    states s ON mortality.state_id = s.id
JOIN
    socioeconomic_data se ON mortality.state_id = se.state_id
GROUP BY
    s.name, se.{factor}, mortality.mortality_rate;
```

Example Query:

For cancer_type = 3 (e.g., Lung Cancer) and factor = "median_income":

```
SELECT
   s.name AS state,
   se.median_income AS socioeconomic_factor,
   mortality.mortality_rate AS mortality_rate
FROM
   (SELECT
      c.state_id,
      SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS mortality_rate
    FROM
      cancer_data c
    JOIN
      demographics d ON c.state_id = d.state_id
    WHERE
      c.is_alive = 0  -- Only mortality data
      AND c.cancer_type_id = 3
    GROUP BY c.state_id
   ) AS mortality
JOIN
   states s ON mortality.state_id = s.id
JOIN
   socioeconomic_data se ON mortality.state_id = se.state_id
GROUP BY
   s.name, se.median_income, mortality.mortality_rate;
```

Explanation:

Inner Query (mortality):
Calculates the cancer mortality rate for each state.
Filters data to include only records where is_alive = 0 (mortality data).
Optionally filters by cancer_type (if provided).

mortality_rate:
SUM(c.count) / NULLIF(d.total_population, 0) * 100
Numerator (SUM(c.count)): Total cancer deaths for the state.
Denominator (d.total_population): Total state population.
NULLIF: Prevents division by zero if total_population is 0.
The result is multiplied by 100 to express the rate as a percentage.
Groups results by c.state_id to calculate mortality rates per state.

Outer Query:
Joins the mortality data with:
states (to get state names).
socioeconomic_data (to get the specified socioeconomic factor).
Groups results by state, socioeconomic_factor, and mortality_rate to ensure unique rows.

Purpose:
Returns a dataset that links the state's socioeconomic factor to its cancer mortality rate.

risk_factors_vs_incidence

Input:
cancer_type: The type of cancer to filter by. If "-" is provided, all cancer types are included.
factor: A column name from the risk_factors table (e.g., inactivity_rate, cigarette_use_rate). Determines the specific risk factor to analyze in relation to cancer incidence rates.

Output:
state: The name of the state (states.name).
risk_factor: The value of the specified risk factor for the state (risk_factors.{factor}).
cancer_incidence_rate: The calculated cancer incidence rate per 100 people in the state.

Template Query:

```
SELECT
   s.name AS state,
   rf.{factor} AS risk_factor,
   incidence.incidence_rate AS cancer_incidence_rate
FROM
   (SELECT
      c.state_id,
      SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS incidence_rate
    FROM
      cancer_data c
    JOIN
      demographics d ON c.state_id = d.state_id
    WHERE
      c.is_alive = 1  -- Only incidence data
      {cancer_type_filter}
    GROUP BY c.state_id
   ) AS incidence
JOIN
   states s ON incidence.state_id = s.id
JOIN
   risk_factors rf ON incidence.state_id = rf.state_id
GROUP BY
   s.name, rf.{factor}, incidence.incidence_rate;
```

Example Query:

For cancer_type = 2 and factor = "inactivity_rate":

```
SELECT
   s.name AS state,
   rf.inactivity_rate AS risk_factor,
   incidence.incidence_rate AS cancer_incidence_rate
FROM
   (SELECT
      c.state_id,
      SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS incidence_rate
    FROM
      cancer_data c
    JOIN
      demographics d ON c.state_id = d.state_id
    WHERE
      c.is_alive = 1  -- Only incidence data
      AND c.cancer_type_id = 2
    GROUP BY c.state_id
   ) AS incidence
JOIN
   states s ON incidence.state_id = s.id
JOIN
   risk_factors rf ON incidence.state_id = rf.state_id
GROUP BY
   s.name, rf.inactivity_rate, incidence.incidence_rate;
```

Explanation:

Inner Query (incidence):
Calculates the cancer incidence rate for each state.
Filters for cancer cases where is_alive = 1 (indicating incidence data, not mortality).
Optionally filters by a specific cancer type if cancer_type is provided.

Incidence Rate Formula:
SUM(c.count) / NULLIF(d.total_population, 0) * 100
Numerator (SUM(c.count)): Total cancer cases for the state.
Denominator (d.total_population): Total state population.
NULLIF: Prevents division by zero if total_population is 0.
The result is expressed as a percentage by multiplying by 100.
Groups results by c.state_id to calculate incidence rates per state.

Outer Query:
Joins the incidence data with:
states (to get state names).
risk_factors (to get the specified risk factor value for the state).
Groups results by state, risk_factor, and incidence_rate to ensure unique rows.

Purpose:
Returns a dataset that links the state's risk factor values to its cancer incidence rates.
Dynamic Filters:

environmental_vs_incidence

Input:
cancer_type: Specifies the type of cancer to filter by. If "-" is provided, the query includes all cancer types.
factor: A column name from the environmental table (e.g., air_quality_index, co2_emissions).Determines the environmental factor to analyze in relation to cancer incidence rates.

Output:
state: The name of the state (states.name).
environmental_factor: The value of the specified environmental factor for the state (environmental.{factor}).
cancer_incidence_rate: The calculated cancer incidence rate per 100 people in the state.

Template Query:

```
SELECT
   s.name AS state,
   e.{factor} AS environmental_factor,
   incidence.incidence_rate AS cancer_incidence_rate
FROM
   (SELECT
      c.state_id,
      SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS incidence_rate
    FROM
      cancer_data c
    JOIN
      demographics d ON c.state_id = d.state_id
    WHERE
      c.is_alive = 1  -- Only incidence data
      {cancer_type_filter}
    GROUP BY c.state_id
   ) AS incidence
JOIN
   states s ON incidence.state_id = s.id
JOIN
   environmental e ON incidence.state_id = e.state_id
GROUP BY
   s.name, e.{factor}, incidence.incidence_rate;
```

Example Query

For cancer_type = 1 and factor = "air_quality_index":

```
SELECT
   s.name AS state,
   e.air_quality_index AS environmental_factor,
   incidence.incidence_rate AS cancer_incidence_rate
FROM
   (SELECT
      c.state_id,
      SUM(c.count) / NULLIF(d.total_population, 0) * 100 AS incidence_rate
    FROM
      cancer_data c
    JOIN
      demographics d ON c.state_id = d.state_id
    WHERE
      c.is_alive = 1  -- Only incidence data
      AND c.cancer_type_id = 1
    GROUP BY c.state_id
   ) AS incidence
JOIN
   states s ON incidence.state_id = s.id
JOIN
   environmental e ON incidence.state_id = e.state_id
GROUP BY
   s.name, e.air_quality_index, incidence.incidence_rate;
```

Explanation:

Inner Query (incidence):
Calculates the cancer incidence rate for each state.
Filters data for cases where is_alive = 1 (indicating incidence data, not mortality).
Optionally filters by a specific cancer type (cancer_type), if provided.

Incidence Rate Formula:
SUM(c.count) / NULLIF(d.total_population, 0) * 100
Numerator (SUM(c.count)): Total cancer cases for the state.
Denominator (d.total_population): Total state population.
NULLIF: Prevents division by zero if total_population is 0.
The result is multiplied by 100 to express as a percentage.
Groups results by c.state_id to calculate incidence rates for each state.

Outer Query:
Joins the incidence data with:
states (to get state names).
environmental (to get the value of the specified environmental factor for each state).
Groups the final results by state, environmental_factor, and incidence_rate to ensure unique rows.

Purpose:
Returns a clean dataset that links the state's environmental factor values to its cancer incidence rates.

Code Documentation

This section provides an overview of the project's structure, describing the purpose and responsibilities of folders and files.

db Folder

The db folder contains the database-related code responsible for connecting to the database, executing queries, and managing data operations. Below is a detailed explanation of the files and their roles:

db_connector.py
Handles the connection to the database and executes queries.
Responsible for all direct interactions with the database.

db_operations.py
Provides utility functions to perform database operations like fetching and querying data.
Calls the execute_query function from db_connector.py and executes queries defined in the query files.

queries.py
Stores reusable SQL queries for general application functionality, including the heatmap query.

insights_queries.py
Contains SQL queries designed for generating insights and analysis.
Includes queries that analyze relationships between socioeconomic factors, risk factors, environmental conditions, and cancer data.

gui Folder

The gui folder is responsible for managing user interactions and presenting visualizations of the data. Each file corresponds to a specific analytical feature, handling user filters and rendering the appropriate analysis or visualization. Below is a brief explanation of the files and their roles:

environmental_analysis.py
Handles user filters and visualizations related to environmental factors, such as air quality and CO2 emissions.

heatmap.py
Generates the heatmap visualization by processing user-selected filters and executing the heatmap query.

risk_factors_analysis.py

Analyzes the impact of behavioral risk factors (e.g., smoking, physical inactivity) on cancer data and presents the results visually.

socioeconomic_analysis.py
Examines socioeconomic disparities in cancer data based on user-provided filters like income, unemployment, and insurance rates.

Other files

__init__.py

Initializes the Flask application, configures session handling, and registers Blueprints for different app modules.
Configures server-side session storage using Flask-Session. Sessions are stored in the filesystem and expire when the browser is closed.

home.py

Defines the route for the app's homepage using the /cancer_atlas endpoint.
Renders the home.html template as the app's landing page.
Implements a Flask Blueprint (home_bp) for modular route organization.

utilities.py

Provides shared utility routes for the app.
Allows users to download session-stored data as a CSV file.
Dynamically generates the CSV file from session data identified by a data_key query parameter.
Handles errors if the data_key is missing or no data is available.

run.py

Serves as the entry point for running the Flask application.

templates Folder

Contains the HTML templates used for rendering the web pages in the app. Each template corresponds to a specific feature or page, such as the heatmap, socioeconomic analysis, or risk calculator. Templates follow Flask’s templating system and are dynamically populated with data from the backend.

static Folder

Holds static assets like CSS files for styling the app's pages. The folder includes shared styles (shared.css), feature-specific styles (e.g., insights.css), and the general layout style (style.css).

Interaction With The Database

The application follows a modular approach for database interactions, ensuring clarity and maintainability. The responsibilities for building and executing queries, as well as handling results, are divided across specific files:

Building Queries: The queries are built in queries.py and insights_queries.py. These files store reusable SQL query functions, which dynamically construct queries based on parameters.

Executing Queries: Query execution is centralized in db_connector.py.
It provides two main functions:
get_db_connection: Establishes a connection to the database and returns a connection object and cursor.
execute_query: Executes SQL queries passed as functions from the query files and fetches the results.

Handling Results:
The db_operations.py file is responsible for processing the raw results returned by execute_query.
It contains utility functions like get_cancer_types, which:
Call query functions from queries.py.
Use execute_query to run the query and fetch results.
Format or augment the data before passing it to the application.
Integration with the GUI:
The GUI modules call the utility functions in db_operations.py to retrieve data for visualizations. While most data handling is done in db_operations.py, the GUI is responsible for ensuring data validity. Specifically, the GUI checks if the retrieved data is None to handle errors gracefully, as this is critical for applications relying on dynamic visualizations.

This structure ensures a clear separation of responsibilities: query construction is centralized, execution is abstracted, and results are formatted before use. It simplifies code maintenance and ensures reliability in database interactions.