Data Cleaning and Preparation (SQL)

This file provides an overview of the data cleaning and preparation steps, including table creation, data import, handling of approximate location data, addressing missing values, data transformation and normalization.

1. Table Creation:
   A table named "**healthcare_data**" was created in the database to import the data. The table was designed with multiple columns to accommodate the dataset.

```sql
-- created a table in the database to import the data
CREATE TABLE healthcare_data(
    metric_item_label text,
    metric_cat_label text,
    metric_subcat_label text,
    metric_item_label_subtitle text,
    metric_cat_item_yaxis_label text,
    metric_source_desc_label_fn text,
    metric_source_desc_label_url_fn text,
    geo_label_city text,
    geo_label_state text,
    geo_label_citystate text,
    geo_fips_code numeric,
    value numeric,
    date_label text,
    geo_label_proxy_or_real text,
    geo_label_proxy_footnote text,
    geo_fips_desc text,
    date_label_proxy_or_real text,
    date_label_proxy_footnote text,
    value_ci_flag_yesno text,
    value_95_ci_low numeric,
    value_95_ci_high numeric,
    value_90_ci_low numeric,
    value_90_ci_high numeric,
    geo_strata_region text,
    geo_strata_poverty text,
    geo_strata_Population text,
    geo_strata_PopDensity text,
    strata_race_label text,
    strata_sex_label text,
    strata_race_sex_label text
);
```

2. Data Import:
   The data was imported into the "**healthcare_data**" table from a CSV file using the COPY command.

```sql
-- Imported the data into the table
COPY healthcare_data FROM 'path_to_your_csv_file.csv' DELIMITER ',' CSV HEADER;
```

3. Handling Approximate Location Data:
   Rows with approximate location data were removed from the table by deleting records where **geo_label_proxy_or_real** is set to "proxy".

```sql
-- Removed the rows form the table where location data is approximate
DELETE FROM healthcare_data
WHERE geo_label_proxy_or_real = 'proxy';
```

4. Missing Value Analysis:
   A thorough analysis was performed to identify missing values in the dataset. SQL queries were used to count the number of missing values in specific columns.

```sql
-- Checking for missing values in the data
SELECT
    SUM(CASE WHEN metric_item_label IS NULL THEN 1 ELSE 0 END) AS metric_item_label,
    SUM(CASE WHEN metric_cat_label IS NULL THEN 1 ELSE 0 END) AS metric_cat_label,
    SUM(CASE WHEN geo_label_city IS NULL THEN 1 ELSE 0 END) AS geo_label_city,
    SUM(CASE WHEN geo_label_state IS NULL THEN 1 ELSE 0 END) AS geo_label_state,
        SUM(CASE WHEN date_label IS NULL THEN 1 ELSE 0 END) AS date_label,
        SUM(CASE WHEN value_95_ci_high IS NULL THEN 1 ELSE 0 END) AS value_95_ci_high,
        SUM(CASE WHEN value_95_ci_low IS NULL THEN 1 ELSE 0 END) AS value_95_ci_low,
        SUM(CASE WHEN value IS NULL THEN 1 ELSE 0 END) AS value,
    SUM(CASE WHEN geo_strata_poverty IS NULL THEN 1 ELSE 0 END) AS geo_strata_poverty,
    SUM(CASE WHEN geo_strata_population IS NULL THEN 1 ELSE 0 END) AS geo_strata_population,
    SUM(CASE WHEN geo_strata_popdensity IS NULL THEN 1 ELSE 0 END) AS geo_strata_popdensity
FROM healthcare_data;
```

5. Missing Value Handling:
   Rows with missing values in the **value** column, **geo_strata_poverty** column, and **value_ci_flag_yesno** column were deleted from the table.

```sql
-- Removed rows with missing data
DELETE FROM healthcare_data
WHERE value IS NULL OR geo_strata_poverty IS NULL;

DELETE FROM healthcare_data
where value_ci_flag_yesno = 'no';

DELETE FROM healthcare_data
where value_ci_flag_yesno = 'yes' and value_95_ci_low is null;
```

6. Table Transformation:
   A new table named `**healthcare_data_clean**` was created. The column names in this table were simplified and made more readable. Data from the original table was inserted into this transformed table.

```sql
-- Created a new table with columns required for the analysis, the column names are renamed into readable way
CREATE TABLE healthcare_data_clean (
    metric text,
    category text,
    category_label text,
    city text,
    "state" text,
    date_label numeric,
    value numeric,
    value_95_ci_high numeric,
    value_95_ci_low numeric,
    poverty text,
    population text,
    population_density text
);

INSERT INTO healthcare_data_clean
SELECT
    metric_item_label,
    metric_cat_label,
    metric_cat_item_yaxis_label,
    geo_label_city,
    geo_label_state,
    date_label::numeric,  -- Cast to numeric
    value,
    value_95_ci_high,
    value_95_ci_low,
    geo_strata_poverty,
    geo_strata_Population,
    geo_strata_PopDensity
FROM healthcare_data;
```

7. Categorization:
   Values in the `**poverty**` column were categorized as 'bpl' (below poverty level) and 'apl' (above poverty level). Similarly, values in the `population_density` column were categorized as 'highest' and 'lower.'

```sql
-- Set values in poverty column as bpl(below poverty line) and apl(above poverty line)
UPDATE healthcare_data_clean
SET poverty = CASE
    WHEN poverty = 'Less poor cities (<20% poor)' THEN 'bpl'
    WHEN poverty = 'Poorest cities (20%+ poor)' THEN 'apl'
    ELSE poverty
END;

-- updated values in pop_density column into a simpler way
UPDATE healthcare_data_clean
SET population_density = CASE
    WHEN population_density = 'Highest pop. density (>10k per sq mi)' THEN 'highest'
    WHEN population_density = 'Lower pop. density (<10k per sq mi)' THEN 'lower'
    ELSE population_density
END;
```

8. State Abbreviation to Full Name:
   State abbreviations were replaced with their full names for better readability and analysis.

```sql
-- updating the state names
UPDATE healthcare_data_clean
SET state =
    CASE
        WHEN state = 'MN' THEN 'Minnesota'
        WHEN state = 'PA' THEN 'Pennsylvania'
        WHEN state = 'CA' THEN 'California'
        WHEN state = 'MD' THEN 'Maryland'
        WHEN state = 'OR' THEN 'Oregon'
        WHEN state = 'TX' THEN 'Texas'
        WHEN state = 'IL' THEN 'Illinois'
        WHEN state = 'NV' THEN 'Nevada'
        WHEN state = 'TN' THEN 'Tennessee'
        WHEN state = 'KY' THEN 'Kentucky'
        WHEN state = 'OH' THEN 'Ohio'
        WHEN state = 'NY' THEN 'New York'
        WHEN state = 'MI' THEN 'Michigan'
        WHEN state = 'WA' THEN 'Washington'
        WHEN state = 'MA' THEN 'Massachusetts'
        WHEN state = 'IN' THEN 'Indiana'
        WHEN state = 'DC' THEN 'District of Columbia'
        WHEN state = 'MO' THEN 'Missouri'
        WHEN state = 'CO' THEN 'Colorado'
        WHEN state = 'NC' THEN 'North Carolina'
        WHEN state = 'WI' THEN 'Wisconsin'
        WHEN state = 'AZ' THEN 'Arizona'
        WHEN state = 'OK' THEN 'Oklahoma'
        ELSE state
    END;
```

9. Normalization:
   The `**normalized_value**` column was added to the table. Metrics in percentage were updated directly in this column, while metrics with values per 100,000 and 1,000 were adjusted to represent percentages.

```sql
-- Create the normalized_value column with default value 0.
ALTER TABLE healthcare_data_clean ADD COLUMN normalized_value numeric DEFAULT 0;

-- Update normalized_value for metrics in percentage.
UPDATE healthcare_data_clean
SET normalized_value = value
WHERE category_label = 'Percent';

-- Update normalized_value for metrics with values per 100k.
UPDATE healthcare_data_clean
SET normalized_value = (value / 1000) -- Divide by 1000 to convert to percentage
WHERE category_label like '%100,000%';

-- Update normalized_value for metrics with values per 1k.
UPDATE healthcare_data_clean
SET normalized_value = (value / 10) -- divide by 10 to get percentages
WHERE category_label like '%1,000%';
```

The SQL queries listed were executed to perform each of these data preparation tasks.

A clean and well-structured dataset in the `healthcare_data_clean` table, ready for further analysis.