

Health Professional Shortage Areas

November 24, 2023

0.0.1 Introduction

The application idea revolves around managing and analyzing healthcare workforce data, specifically focusing on Health Professional Shortage Areas (HPSAs) and Medically Underserved Areas (MUAs). The data source is a set of tables representing these areas, obtained from reliable healthcare databases or government health agencies.

The primary goal of the application is to provide insights into the distribution and characteristics of healthcare resources across different regions. It aims to assist healthcare planners, policymakers, and researchers in making informed decisions about resource allocation, identifying areas with shortages, and addressing healthcare disparities.

The core data includes information about various health facilities, their designations, and the status of different HPSAs and MUAs. This data is crucial for understanding the availability of healthcare services in different geographical areas. The application can be a valuable tool for optimizing workforce distribution, improving access to care, and ultimately enhancing overall health outcomes.

The database schema is designed to capture essential information about Health Professional Shortage Areas (HPSAs) and Medically Underserved Areas (MUAs). Here's a brief overview of the schema:

0.0.2 Tables:

hpsa_primary_care:

Source_ID (Primary Key)
Source_Name
Status_Code
Status_Description
Type_Code
Type_Desc
Address
City
State_Abbr
Postal_Code

hpsa_mental_health:

Source_ID (Primary Key)
Source_Name
Status_Code
Status_Description

Type_Code
Type_Desc
State_Abbr
Degree_of_Shortage
Designation_Date
Designation_Last_Update_Date

hpsa_dental_health:

Source_ID (Primary Key)
Source_Name
Status_Code
Status_Description
Type_Code
Type_Desc
Address
City
State_Abbr
Postal_Code

hpsa_mua:

MUA_SOURCE_ID (Primary Key)
MUA_AREA_CD
MUA_DESIGNATION_TYP_CD
MUA_DESIGNATION_TYP_DESC
MUA_STATUS_CD
MUA_STATUS_DESC
CENSUS_TRACT
MUA_DESIGNATION_DT
MUA_DESIGNATION_DT_TXT
MUA_SCORE

0.0.3 Rationale:

- Normalization: The schema follows normalization principles to minimize data redundancy and improve data integrity.
- Primary Keys: Each table has a primary key to uniquely identify records.

Consistent Naming: Column names are consistent across tables for similar attributes, facilitating ease of understanding and query writing.

Relationships: While the schema presented here doesn't explicitly show foreign keys, they would be used to establish relationships between tables, ensuring data consistency.

This schema allows for efficient querying and analysis of healthcare workforce data, providing a foundation for the application's functionality.

```
[ ]: import pandas as pd  
  
import mysql.connector
```

```

from mysql.connector import Error

# Read the Excel file
filename = 'data.xlsx'
xls = pd.ExcelFile(filename)

# Get the sheet names
sheet_names = xls.sheet_names

# Create a dictionary to store the datasets
datasets = {}

# Loop through each sheet and save the data as a dataset
for sheet_name in sheet_names:
    dataset = pd.read_excel(filename, sheet_name=sheet_name)

    # Remove columns with NaN values
    dataset = dataset.dropna(axis=1, how='all')

    # Replace NaN values with None
    dataset = dataset.replace(to_replace=float('nan'), value=None)

    # Convert columns to appropriate types if needed
    # dataset['column_name'] = pd.to_numeric(dataset['column_name'],
    →errors='coerce')

    datasets[sheet_name] = dataset

# Access a specific dataset
sheet1_data = datasets['hpsa_primary_care']
sheet2_data = datasets['hpsa_mental_health']
sheet3_data = datasets['hpsa_dental_health']
sheet4_data = datasets['hpsa_mua']

```

```
[ ]: sheet1_data.head()
```

```

[ ]:
   Source_ID      Source_Name Status_Code \
0  1569995651      Ft. Washakie PHS Indian Health Center      D
1  1469994698  McLaughlin PHS Indian Medical/Dental Clinic      D
2  1469994687                Wagner PHS Indian Hospital      D
3  141999413V                Portland Area Office      D
4  14099940N2                Shawnee Health Center      D

   Status_Description Type_Code      Type_Desc \
0      Designated      IHS  Indian, Tribal and Urban Indian Organizations
1      Designated      IHS  Indian, Tribal and Urban Indian Organizations
2      Designated      IHS  Indian, Tribal and Urban Indian Organizations

```

3	Designated	IHS	Indian, Tribal and Urban Indian Organizations
4	Designated	IHS	Indian, Tribal and Urban Indian Organizations

	Address	City	State_Abbr	Postal_Code	...	\
0	PO BOX 128	Fort Washakie	WY	82514-0128	...	
1	611 2nd Ave E	Mc Laughlin	SD	57642	...	
2	110 Washington Ave NW	Wagner	SD	57380	...	
3	1220 SW 3rd Ave Ste 476	Portland	OR	97204-2825	...	
4	2307 Gordon Cooper Dr	Shawnee	OK	74801-9007	...	

	Common_Postal_Code	Common_County_Name	Common_State	County_FIPS	\
0	82514	Fremont County, WY		56013	
1	57642	Corson County, SD		46031	
2	57380	Charles Mix County, SD		46023	
3	97204	Multnomah County, OR		41051	
4	74801	Pottawatomie County, OK		40125	

	Common_State_Abbr	Common_State_Name	Common_State_FIPS	Common_Region_Name	\
0	WY	Wyoming	56	Region VIII	
1	SD	South Dakota	46	Region VIII	
2	SD	South Dakota	46	Region VIII	
3	OR	Oregon	41	Region X	
4	OK	Oklahoma	40	Region VI	

	Rural_Status_Code	Rural_Status_Desc	\
0	R	Rural	
1	R	Rural	
2	R	Rural	
3	N	Non-Rural	
4	R	Rural	

	HPSA_Designation_Pop_Type_Desc
0	Indian, Tribal and Urban Indian Organizations
1	Indian, Tribal and Urban Indian Organizations
2	Indian, Tribal and Urban Indian Organizations
3	Indian, Tribal and Urban Indian Organizations
4	Indian, Tribal and Urban Indian Organizations

[5 rows x 55 columns]

```
[ ]: sheet2_data.head()
```

```
[ ]:
Source_ID      Source_Name  Status_Code  \
0      733007      Coos      W
1  7449994412  Northern Rhode Island Catchment Area      W
2  7449994412  Northern Rhode Island Catchment Area      W
3  7449994412  Northern Rhode Island Catchment Area      W
```

4 7449994412 Northern Rhode Island Catchment Area W

	Status_Description	Type_Code	Type_Desc	State_Abbr	Degree_of_Shortage	\
0	Withdrawn	Hpsa Geo	Geographic HPSA	NH	Not applicable	
1	Withdrawn	Hpsa Geo	Geographic HPSA	RI	Not applicable	
2	Withdrawn	Hpsa Geo	Geographic HPSA	RI	Not applicable	
3	Withdrawn	Hpsa Geo	Geographic HPSA	RI	Not applicable	
4	Withdrawn	Hpsa Geo	Geographic HPSA	RI	Not applicable	

	Designation_Date	Designation_Last_Update_Date	...	Common_State_Abbr	\
0	29005	37799	...	NH	
1	37672	39671	...	RI	
2	37672	39671	...	RI	
3	37672	39671	...	RI	
4	37672	39671	...	RI	

	Common_State_Name	Common_State_FIPS	Common_Region_Name	HPSA_Withdrawn_Date	\
0	New Hampshire	33	Region I	37799	
1	Rhode Island	44	Region I	39671	
2	Rhode Island	44	Region I	39671	
3	Rhode Island	44	Region I	39671	
4	Rhode Island	44	Region I	39671	

	HPSA_Withdrawn_Date_String	Provider_Type	Rural_Status_Code	\
0	6/27/2003	None	R	
1	8/11/2008	None	None	
2	8/11/2008	None	None	
3	8/11/2008	None	None	
4	8/11/2008	None	None	

	Rural_Status_Desc	HPSA_Designation_Pop_Type_Desc
0	Rural	Geographic Population
1	None	Geographic Population
2	None	Geographic Population
3	None	Geographic Population
4	None	Geographic Population

[5 rows x 64 columns]

```
[ ]: sheet3_data.columns
```

```
[ ]: Index(['Source_ID', 'Source_Name', 'Status_Code', 'Status_Description',
           'Type_Code', 'Type_Desc', 'Address', 'City', 'State_Abbr',
           'Postal_Code', 'Designation_Date', 'Designation_Last_Update_Date',
           'Designation_Pop', 'Metropolitan_Indicator_Code',
           'Metropolitan_Indicator_Desc', 'HPSA_Score', 'HPSA_Shortage',
           'Discipline_Class_Num', 'Discipline_Class_Desc', 'Component_Source_ID',
```

```

'Component_Source_Name', 'Component_Status_Code',
'Component_Status_Desc', 'Component_Type_Code', 'Component_Type_Desc',
'Geography_ID', 'CountyFIPS', 'County_Name', 'StateCountyFIPS',
'State_FIPS', 'State_Abbbr_2', 'State_Name', 'Primary_State_Name',
'Primary_State_FIPS', 'Primary_HHS_Region_Name',
'US_Mexico_Border_County', 'US_Mexico_Border_100km',
'Data_Warehouse_Record_Create_Date',
'Data_Warehouse_Record_Create_Date_Text', 'HPSA_Name',
'HPSA_Component_Name', 'Break_in_Designation', 'Geocoding_Primary_X',
'Geocoding_Primary_Y', 'Common_City_Name_with_State_Abbbr',
'Common_Postal_Code', 'Common_County_Name', 'Common_StateCounty_FIPS',
'Common_State_Abbbr', 'Common_State_Name', 'Common_State_FIPS',
'Common_Region_Name', 'Rural_Status_Code', 'Rural_Status_Desc',
'HPSA_Designation_Pop_Type_Desc'],
dtype='object')

```

```
[ ]: sheet3_data.head()
```

```

[ ]:      Source_ID      Source_Name Status_Code \
0  6409994016      Carl Albert Indian Hospital      W
1  6319993124      Winnebago PHS Indian Hospital      W
2  6089990849  Southern Colorado Ute Services Unit-Fed      W
3  60499904B5  Phx Area Office Two Renaissance Square      W
4  6569995629    Ft. Washakie PHS Indian Health Center      D

      Status_Description Type_Code      Type_Desc \
0      Withdrawn      IHS  Indian, Tribal and Urban Indian Organizations
1      Withdrawn      IHS  Indian, Tribal and Urban Indian Organizations
2      Withdrawn      IHS  Indian, Tribal and Urban Indian Organizations
3      Withdrawn      IHS  Indian, Tribal and Urban Indian Organizations
4      Designated      IHS  Indian, Tribal and Urban Indian Organizations

      Address      City State_Abbbr Postal_Code ... \
0  1001 N Country Club Rd      Ada      OK  74820-2847 ...
1      PO BOX Hh      Winnebago      NE  68071-0767 ...
2      Weeminuche Dr      Ignacio      CO      81137 ...
3      100 N 1st St      Phoenix      AZ      85004 ...
4      PO BOX 128  Fort Washakie      WY  82514-0128 ...

      Common_Postal_Code  Common_County_Name Common_StateCounty_FIPS \
0      74820  Pontotoc County, OK      40123
1      68071  Thurston County, NE      31173
2      81137  La Plata County, CO      8067
3      85004  Maricopa County, AZ      4013
4      82514  Fremont County, WY      56013

      Common_State_Abbbr Common_State_Name  Common_State_FIPS Common_Region_Name \

```

0	OK	Oklahoma	40	Region VI
1	NE	Nebraska	31	Region VII
2	CO	Colorado	8	Region VIII
3	AZ	Arizona	4	Region IX
4	WY	Wyoming	56	Region VIII

	Rural_Status_Code	Rural_Status_Desc	\
0	R	Rural	
1	R	Rural	
2	R	Rural	
3	N	Non-Rural	
4	R	Rural	

	HPSA_Designation_Pop_Type_Desc
0	Indian, Tribal and Urban Indian Organizations
1	Indian, Tribal and Urban Indian Organizations
2	Indian, Tribal and Urban Indian Organizations
3	Indian, Tribal and Urban Indian Organizations
4	Indian, Tribal and Urban Indian Organizations

[5 rows x 55 columns]

```
[ ]: sheet4_data.columns
```

```
[ ]: Index(['MUA_SOURCE_ID', 'MUA_AREA_CD', 'MUA_DESIGNATION_TYP_CD',
'MUA_DESIGNATION_TYP_DESC', 'MUA_STATUS_CD', 'MUA_STATUS_DESC',
'CENSUS_TRACT', 'MUA_DESIGNATION_DT', 'MUA_DESIGNATION_DT_TXT',
'MUA_SCORE', 'MUA_SERVICE_AREA_NM', 'MUA_UPDATE_DT',
'MUA_UPDATE_DT_TXT', 'US_MEXICO_BORDER_100KM_IND',
'US_MEXICO_BORDER_COUNTY_IND', 'STATE_COUNTY_FIPS_CD', 'COUNTY_FIPS_CD',
'LIST_BOX_COUNTY_NM', 'COUNTY_NM', 'COUNTY_DESC', 'REGION_CD',
'REGION_NM', 'STATE_FIPS_CD', 'STATE_NM', 'STATE_ABBR',
'POVERTY_100_PCT_NUM', 'POP_AGE_65_OVER_PCT', 'INFANT_MORTALITY_RATE',
'DW_RECORD_CREATE_DT', 'DW_RECORD_CREATE_DT_TXT',
'PRIMARY_STATE_FIPS_CD', 'PRIMARY_STATE_ABBR', 'PRIMARY_STATE_NM',
'PRIMARY_REGION_CD', 'PRIMARY_REGION_NM', 'CMN_REGION_CD',
'CMN_REGION_NM', 'CMN_STATE_NM', 'CMN_STATE_ABBR', 'CMN_STATE_FIPS_CD',
'CMN_STATE_COUNTY_FIPS_CD', 'CMN_COUNTY_NM_STATE_ABBR',
'BREAK_DESIGNATION_IND', 'MUA_COMP_DESIGNATION_DT',
'MUA_COMP_DESIGNATION_DT_TXT', 'MUA_COMP_GEO_NM', 'MUA_COMP_GEO_TYP_CD',
'MUA_COMP_GEO_TYP_DESC', 'MUA_COMP_GEO_TYP_ID',
'MUA_COMP_LAST_UPDATE_DT', 'MUA_COMP_STATUS_CD', 'MUA_COMP_STATUS_DESC',
'MUA_COMP_UPDATE_DT_TXT', 'MUA_DESIGNATION_POP', 'MUA_METRO_IND_CD',
'MUA_METRO_IND_DESC', 'MUA_METRO_IND_ID', 'MUA_POPULATION_TYP_CD',
'MUA_POPULATION_TYP_DESC', 'MUA_POPULATION_TYP_ID', 'MUA_RES_CIV_POP',
'RURAL_STATUS_CD', 'RURAL_STATUS_DESC', 'PROVIDER_1000_POP'],
dtype='object')
```

```
[ ]: sheet4_data.head()
```

```
[ ]:   MUA_SOURCE_ID  MUA_AREA_CD  MUA_DESIGNATION_TYP_CD  \
0           474    9013530200                MUP
1           474    9013530100                MUP
2           470    9005310300                MUP
3           470    9005310500                MUP
4           470    9005310801                MUP
```

```
      MUA_DESIGNATION_TYP_DESC  MUA_STATUS_CD  MUA_STATUS_DESC  \
0  Medically Underserved Population          D    Designated
1  Medically Underserved Population          D    Designated
2  Medically Underserved Population          D    Designated
3  Medically Underserved Population          D    Designated
4  Medically Underserved Population          D    Designated
```

```
      CENSUS_TRACT  MUA_DESIGNATION_DT  MUA_DESIGNATION_DT_TXT  MUA_SCORE  ...  \
0         5302.00                34676        12/8/1994        47.8  ...
1         5301.00                34676        12/8/1994        47.8  ...
2         3103.00                34676        12/8/1994        41.1  ...
3         3105.00                34676        12/8/1994        41.1  ...
4         3108.01                34676        12/8/1994        41.1  ...
```

```
      MUA_METRO_IND_CD  MUA_METRO_IND_DESC  MUA_METRO_IND_ID  MUA_POPULATION_TYP_CD  \
0                   0          Unknown                0                LI
1                   0          Unknown                0                LI
2                   0          Unknown                0                LI
3                   0          Unknown                0                LI
4                   0          Unknown                0                LI
```

```
      MUA_POPULATION_TYP_DESC  MUA_POPULATION_TYP_ID  MUA_RES_CIV_POP  \
0      MUP Low Income                2          None
1      MUP Low Income                2          None
2      MUP Low Income                2          None
3      MUP Low Income                2          None
4      MUP Low Income                2          None
```

```
      RURAL_STATUS_CD  RURAL_STATUS_DESC  PROVIDER_1000_POP
0                   N      Non-Rural        None
1                   N      Non-Rural        None
2                   R        Rural        None
3                   R        Rural        None
4                   R        Rural        None
```

```
[5 rows x 64 columns]
```


0.0.4 Stored Procedures:

1. `usp_GetAverageScoreByMUAStatus` Purpose: This stored procedure calculates and returns the average MUA score for each MUA status.

- How it works:

```
CREATE PROCEDURE usp_GetAverageScoreByMUAStatus
AS
BEGIN
    SELECT
        MUA_STATUS_DESC,
        AVG(CAST(MUA_SCORE AS FLOAT)) AS Average_Score
    FROM
        hpsa_mua
    GROUP BY
        MUA_STATUS_DESC;
END;
```

- Usage in Application: This procedure provides a quick overview of the average MUA scores based on different MUA statuses. It aids in identifying trends and disparities in healthcare accessibility across various designations.

2. `usp_GetHPSAByStateAndType` Purpose: This stored procedure retrieves HPSAs based on the provided state abbreviation and type code.

How it works:

```
CREATE PROCEDURE usp_GetHPSAByStateAndType
    @StateAbbreviation NVARCHAR(2),
    @TypeCode NVARCHAR(10)
AS
BEGIN
    SELECT *
    FROM
        hpsa_primary_care
    WHERE
        State_Abbr = @StateAbbreviation
        AND Type_Code = @TypeCode;
END;
```

- Usage in Application: It allows the application to fetch specific HPSAs based on user-inputted criteria, helping healthcare planners to focus on areas of interest and address shortages effectively.

0.0.5 Queries

1. Average Score of Designated MUAs: Purpose: To retrieve the average MUA score for designated areas.

```
SELECT
    MUA_STATUS_DESC,
```

```

        AVG(CAST(MUA_SCORE AS FLOAT)) AS Average_Score
FROM
    hpsa_mua
WHERE
    MUA_STATUS_DESC = 'Designated'
GROUP BY
    MUA_STATUS_DESC;

```

2. HPSAs in a Specific State and Type: Purpose: To fetch HPSAs in a particular state and of a specific type.

```

SELECT *
FROM
    hpsa_primary_care
WHERE
    State_Abbr = 'CA'
    AND Type_Code = 'IHS';

```

3. Designated Mental Health HPSAs: Purpose: To retrieve information about designated mental health HPSAs.

```

SELECT *
FROM
    hpsa_mental_health
WHERE
    Status_Description = 'Designated';

```

4. MUAs with the Highest Scores: Purpose: To find MUAs with the highest scores.

```

SELECT TOP 5
    *
FROM
    hpsa_mua
ORDER BY
    MUA_SCORE DESC;

```

How they are used in Application:

- The average MUA score query aids in displaying a summary of MUA scores.
- The HPSA retrieval query allows users to explore specific HPSAs based on state and type.
- The designated mental health HPSAs query provides insights into areas specifically designated for mental health.
- The query for MUAs with the highest scores helps identify areas with the greatest need for attention and resources.
- These queries and procedures collectively empower the application to offer detailed insights into healthcare workforce distribution and shortages, supporting informed decision-making.
- Database Connection:

Attempts to connect to a MySQL database using the provided host, user, password, and database information. - Table Creation and Data Insertion:

Defines four datasets, each representing a table (hpsa_primary_care, hpsa_mental_health,

hpsa_dental_health, and hpsa_mua). For each dataset, it dynamically generates a CREATE TABLE query based on the column names and their types (assumed VARCHAR(255)). Executes the CREATE TABLE query to create a table in the MySQL database. Prepares and executes an INSERT query to insert the data from the dataset into the corresponding table. - Commit Changes:

Commits the changes to the database. This step is crucial for the changes to take effect permanently.

- Error Handling:

If any error occurs during the process (such as a connection error, SQL syntax error, etc.), it prints an error message. - Connection Closure:

Closes the cursor and the database connection, ensuring proper cleanup. This script is designed to initialize a MySQL database by creating tables based on provided datasets and inserting data into these tables. It's a common approach when setting up a database for the first time or when updating the schema with new data.

```
[ ]: import pandas as pd
import mysql.connector
from mysql.connector import Error

# Database connection details
host = 'localhost'
user = 'root'
password = ''
database = 'task_x'

# Create a MySQL connection
try:
    conn = mysql.connector.connect(host=host, user=user, password=password,
    ↪database=database)
    if conn.is_connected():
        print('Connected to MySQL database')

    # Create a cursor
    cursor = conn.cursor()

    # Drop tables if they exist
    cursor.execute("DROP TABLE IF EXISTS hpsa_primary_care")
    cursor.execute("DROP TABLE IF EXISTS hpsa_mental_health")
    cursor.execute("DROP TABLE IF EXISTS hpsa_dental_health")
    cursor.execute("DROP TABLE IF EXISTS hpsa_mua")

    # Insert datasets into MySQL tables
    datasets = [(sheet1_data.iloc[:, :10], 'hpsa_primary_care'),
                (sheet2_data.iloc[:, :10], 'hpsa_mental_health'),
                (sheet3_data.iloc[:, :10], 'hpsa_dental_health'),
                (sheet4_data.iloc[:, :10], 'hpsa_mua')]

    for dataset, table_name in datasets:
```

```

        column_names = list(dataset.columns)
        values = dataset.values.tolist()

        # Generate column definitions with types
        column_definitions = ', '.join([f'{column} VARCHAR(255)' for column_
↪in column_names])

        # Create table with column names and types
        create_table_query = f"CREATE TABLE {table_name}_
↪({column_definitions})"
        cursor.execute(create_table_query)

        placeholders = ', '.join(['%s'] * len(column_names))

        # Insert data into table
        insert_query = f"INSERT INTO {table_name} ({', '.
↪join(column_names)}) VALUES ({placeholders})"
        cursor.executemany(insert_query, values)

        # Commit the changes
        conn.commit()
        print('Data inserted into MySQL tables')

except Error as e:
    print(f"Error: {e}")

finally:
    # Close the cursor and connection
    if conn.is_connected():
        cursor.close()
        conn.close()
        print('You can now select from tables')

```

```

Connected to MySQL database
Data inserted into MySQL tables
You can now select from tables

```

This Python code establishes a connection to a MySQL database using the `mysql.connector` library. It prompts the user to input a SQL query, then executes the query and fetches all rows from the result. Finally, it prints each row to the console.

Here's a breakdown:

- Database Connection:

It attempts to connect to a MySQL database using the provided host, user, password, and database information.

- User Input:

The user is prompted to enter a SQL query they want to perform on the connected database.

- Query Execution:

The provided SQL query is executed using the database cursor. - Fetching and Printing Rows:

All rows resulting from the query execution are fetched. Each row is printed to the console. - Error Handling:

If any error occurs during the process, it prints an error message. - Connection Closure:

Finally, it closes the cursor and the database connection. This code allows users to interactively input and execute SQL queries on the connected MySQL database, providing a flexible way to explore and retrieve data.

```
[ ]: # Create a MySQL connection
try:
    conn = mysql.connector.connect(host=host, user=user, password=password,
    ↪database=database)
    if conn.is_connected():
        print('Connected to MySQL database')

        # Create a cursor
        cursor = conn.cursor()

        # Select all rows from a table
        select_query = input(str("Enter a query to perform: ")) #"SELECT * FROM_
    ↪hpsa_mua where hpsa_mua.mua_source_ID > 1000;"
        cursor.execute(select_query)

        # Fetch all rows
        rows = cursor.fetchall()

        # Print the rows
        for row in rows:
            print(row)

except Error as e:
    print(f"Error: {e}")

finally:
    # Close the cursor and connection
    if conn.is_connected():
        cursor.close()
        conn.close()
        print('Connection closed')
```

```
Connected to MySQL database
('hpsa_dental_health',)
('hpsa_mental_health',)
```

```
( 'hpsa_mua', )
( 'hpsa_primary_care', )
Connection closed
```

```
[ ]: # print all columns for all tables
sheet1_data.columns
```

```
[ ]: Index(['Source_ID', 'Source_Name', 'Status_Code', 'Status_Description',
          'Type_Code', 'Type_Desc', 'Address', 'City', 'State_Abbr',
          'Postal_Code', 'Designation_Date', 'Designation_Last_Update_Date',
          'Designation_Pop', 'Metropolitan_Indicator_Code',
          'Metropolitan_Indicator_Desc', 'HPSA_Score', 'HPSA_Shortage',
          'Discipline_Class_Num', 'Discipline_Class_Desc', 'Component_Source_ID',
          'Component_Source_Name', 'Component_Status_Code',
          'Component_Status_Desc', 'Component_Type_Code', 'Component_Type_Desc',
          'Geography_ID', 'CountyFIPS', 'County_Name', 'StateCountyFIPS',
          'State_FIPS', 'State_Abbr_2', 'State_Name', 'Primary_State_Name',
          'Primary_State_FIPS', 'Primary_HHS_Region_Name',
          'US_Mexico_Border_County', 'US_Mexico_Border_100km',
          'Data_Warehouse_Record_Create_Date',
          'Data_Warehouse_Record_Create_Date_Text', 'HPSA_Name',
          'HPSA_Component_Name', 'Break_in_Designation', 'Geocoding_Primary_X',
          'Geocoding_Primary_Y', 'Common_City_Name_with_State_Abbr',
          'Common_Postal_Code', 'Common_County_Name', 'Common_StateCounty_FIPS',
          'Common_State_Abbr', 'Common_State_Name', 'Common_State_FIPS',
          'Common_Region_Name', 'Rural_Status_Code', 'Rural_Status_Desc',
          'HPSA_Designation_Pop_Type_Desc'],
          dtype='object')
```

```
[ ]: sheet2_data.columns
```

```
[ ]: Index(['Source_ID', 'Source_Name', 'Status_Code', 'Status_Description',
          'Type_Code', 'Type_Desc', 'State_Abbr', 'Degree_of_Shortage',
          'Designation_Date', 'Designation_Last_Update_Date', 'Designation_Pop',
          'Estimated_Underserved_Pop', 'Estimated_Served_Pop', 'Formal_Ratio',
          'Total_FTE_Clinicians', 'Metropolitan_Indicator_Code',
          'Metropolitan_Indicator_Desc', 'Provider_Ratio_Goal',
          'Percent_Pop_Below_Poverty', 'HPSA_Score', 'HPSA_Shortage',
          'Discipline_Class_Num', 'Discipline_Class_Desc',
          'Component_Source_Name', 'Component_Status_Code',
          'Component_Status_Desc', 'Component_Type_Code', 'Component_Type_Desc',
          'Component_State_Abbr', 'Component_Designation_Date',
          'Component_Designation_Date_String',
          'Component_Designation_Last_Update_Date', 'Geography_ID', 'CountyFIPS',
          'County_Name', 'StateCountyFIPS', 'State_FIPS', 'State_Abbr_2',
          'State_Name', 'Primary_State_Name', 'Primary_State_FIPS',
          'Primary_HHS_Region_Name', 'US_Mexico_Border_County',
          'US_Mexico_Border_100km', 'Data_Warehouse_Record_Create_Date',
```

```

'Data_Warehouse_Record_Create_Date_Text', 'HPSA_Name',
'HPSA_Component_Name', 'Break_in_Designation', 'HPSA_Pop_Type_Code',
'HPSA_Pop_Type_Desc', 'HPSA_Resident_Civilian_Pop',
'Common_County_Name', 'Common_StateCounty_FIPS', 'Common_State_Abbr',
'Common_State_Name', 'Common_State_FIPS', 'Common_Region_Name',
'HPSA_Withdrawn_Date', 'HPSA_Withdrawn_Date_String', 'Provider_Type',
'Rural_Status_Code', 'Rural_Status_Desc',
'HPSA_Designation_Pop_Type_Desc'],
dtype='object')

```

```
[ ]: sheet3_data.columns
```

```
[ ]: Index(['Source_ID', 'Source_Name', 'Status_Code', 'Status_Description',
'Type_Code', 'Type_Desc', 'Address', 'City', 'State_Abbr',
'Postal_Code', 'Designation_Date', 'Designation_Last_Update_Date',
'Designation_Pop', 'Metropolitan_Indicator_Code',
'Metropolitan_Indicator_Desc', 'HPSA_Score', 'HPSA_Shortage',
'Discipline_Class_Num', 'Discipline_Class_Desc', 'Component_Source_ID',
'Component_Source_Name', 'Component_Status_Code',
'Component_Status_Desc', 'Component_Type_Code', 'Component_Type_Desc',
'Geography_ID', 'CountyFIPS', 'County_Name', 'StateCountyFIPS',
'State_FIPS', 'State_Abbr_2', 'State_Name', 'Primary_State_Name',
'Primary_State_FIPS', 'Primary_HHS_Region_Name',
'US_Mexico_Border_County', 'US_Mexico_Border_100km',
'Data_Warehouse_Record_Create_Date',
'Data_Warehouse_Record_Create_Date_Text', 'HPSA_Name',
'HPSA_Component_Name', 'Break_in_Designation', 'Geocoding_Primary_X',
'Geocoding_Primary_Y', 'Common_City_Name_with_State_Abbr',
'Common_Postal_Code', 'Common_County_Name', 'Common_StateCounty_FIPS',
'Common_State_Abbr', 'Common_State_Name', 'Common_State_FIPS',
'Common_Region_Name', 'Rural_Status_Code', 'Rural_Status_Desc',
'HPSA_Designation_Pop_Type_Desc'],
dtype='object')

```

```
[ ]: sheet4_data.columns
```

```
[ ]: Index(['MUA_SOURCE_ID', 'MUA_AREA_CD', 'MUA_DESIGNATION_TYP_CD',
'MUA_DESIGNATION_TYP_DESC', 'MUA_STATUS_CD', 'MUA_STATUS_DESC',
'CENSUS_TRACT', 'MUA_DESIGNATION_DT', 'MUA_DESIGNATION_DT_TXT',
'MUA_SCORE', 'MUA_SERVICE_AREA_NM', 'MUA_UPDATE_DT',
'MUA_UPDATE_DT_TXT', 'US_MEXICO_BORDER_100KM_IND',
'US_MEXICO_BORDER_COUNTY_IND', 'STATE_COUNTY_FIPS_CD', 'COUNTY_FIPS_CD',
'LIST_BOX_COUNTY_NM', 'COUNTY_NM', 'COUNTY_DESC', 'REGION_CD',
'REGION_NM', 'STATE_FIPS_CD', 'STATE_NM', 'STATE_ABBR',
'POVERTY_100_PCT_NUM', 'POP_AGE_65_OVER_PCT', 'INFANT_MORTALITY_RATE',
'DW_RECORD_CREATE_DT', 'DW_RECORD_CREATE_DT_TXT',
'PRIMARY_STATE_FIPS_CD', 'PRIMARY_STATE_ABBR', 'PRIMARY_STATE_NM',

```

```

'PRIMARY_REGION_CD', 'PRIMARY_REGION_NM', 'CMN_REGION_CD',
'CMN_REGION_NM', 'CMN_STATE_NM', 'CMN_STATE_ABBR', 'CMN_STATE_FIPS_CD',
'CMN_STATE_COUNTY_FIPS_CD', 'CMN_COUNTY_NM_STATE_ABBR',
'BREAK_DESIGNATION_IND', 'MUA_COMP_DESIGNATION_DT',
'MUA_COMP_DESIGNATION_DT_TXT', 'MUA_COMP_GEO_NM', 'MUA_COMP_GEO_TYP_CD',
'MUA_COMP_GEO_TYP_DESC', 'MUA_COMP_GEO_TYP_ID',
'MUA_COMP_LAST_UPDATE_DT', 'MUA_COMP_STATUS_CD', 'MUA_COMP_STATUS_DESC',
'MUA_COMP_UPDATE_DT_TXT', 'MUA_DESIGNATION_POP', 'MUA_METRO_IND_CD',
'MUA_METRO_IND_DESC', 'MUA_METRO_IND_ID', 'MUA_POPULATION_TYP_CD',
'MUA_POPULATION_TYP_DESC', 'MUA_POPULATION_TYP_ID', 'MUA_RES_CIV_POP',
'RURAL_STATUS_CD', 'RURAL_STATUS_DESC', 'PROVIDER_1000_POP'],
dtype='object')

```

0.0.6 Usage - Queries Procedures

```

[ ]: # Function to fetch and print tables with headers
def print_tables():
    try:
        conn = mysql.connector.connect(host=host, user=user, password=password,
→database=database)
        if conn.is_connected():
            cursor = conn.cursor(dictionary=True)

            # Fetch table names
            cursor.execute("SHOW TABLES")
            tables = cursor.fetchall()

            # Print tables with headers
            for table in tables:
                table_name = table['Tables_in_task_x']
                print(f"Table: {table_name}")

                # Fetch and print table data with headers
                cursor.execute(f"SELECT * FROM {table_name}")
                result = cursor.fetchall()

                if result:
                    df = pd.DataFrame(result)
                    print(df)

                print("\n")

    except Error as e:
        print(f"Error: {e}")

    finally:
        if conn.is_connected():

```



```

        cursor.close()
        conn.close()

# Print tables with headers
print_tables()

```

Table: hpsa_dental_health

	Source_ID	Source_Name	Status_Code	\
0	6409994016	Carl Albert Indian Hospital	W	
1	6319993124	Winnebago PHS Indian Hospital	W	
2	6089990849	Southern Colorado Ute Services Unit-Fed	W	
3	60499904B5	Phx Area Office Two Renaissance Square	W	
4	6569995629	Ft. Washakie PHS Indian Health Center	D	
..	
495	6043918770	NA CARDIOLOGY PROG FLAGSTAFF	D	
496	6043906683	EAST FORK LUTHERAN SCHOOL	D	
497	6042473681	NAHATA DZIIL HEALTH CENTER	D	
498	6042421598	JOHN F. KENNEDY SCHOOL	D	
499	6042144212	CRADLEBOARD ELEMENTARY SCHOOL	D	

	Status_Description	Type_Code	\
0	Withdrawn	IHS	
1	Withdrawn	IHS	
2	Withdrawn	IHS	
3	Withdrawn	IHS	
4	Designated	IHS	
..	
495	Designated	ITU	
496	Designated	ITU	
497	Designated	ITU	
498	Designated	ITU	
499	Designated	ITU	

	Type_Desc	\
0	Indian, Tribal and Urban Indian Organizations	
1	Indian, Tribal and Urban Indian Organizations	
2	Indian, Tribal and Urban Indian Organizations	
3	Indian, Tribal and Urban Indian Organizations	
4	Indian, Tribal and Urban Indian Organizations	
..	...	
495	Indian Health Service, Tribal Health, and Urba...	
496	Indian Health Service, Tribal Health, and Urba...	
497	Indian Health Service, Tribal Health, and Urba...	
498	Indian Health Service, Tribal Health, and Urba...	
499	Indian Health Service, Tribal Health, and Urba...	

	Address	City	State_Abbr	Postal_Code
0	1001 N Country Club Rd	Ada	OK	74820-2847

1	PO BOX Hh	Winnebago	NE	68071-0767
2	Weeminuche Dr	Ignacio	CO	81137
3	100 N 1st St	Phoenix	AZ	85004
4	PO BOX 128	Fort Washakie	WY	82514-0128
..
495	1215 N Beaver St Ste 201	Flagstaff	AZ	86001-3126
496	4325 Fort Apache Rd.	Whiteriver	AZ	85941
497	Chiih'tow Boulevard	Sanders	AZ	86512
498	110 W. Dish Chin Rd.	Whiteriver	AZ	85941
499	7301 Power Line Rd	Whiteriver	AZ	85941

[500 rows x 10 columns]

Table: hpsa_mental_health

	Source_ID	Source_Name	Status_Code	\
0	733007	Coos	W	
1	7449994412	Northern Rhode Island Catchment Area	W	
2	7449994412	Northern Rhode Island Catchment Area	W	
3	7449994412	Northern Rhode Island Catchment Area	W	
4	7449994412	Northern Rhode Island Catchment Area	W	
..	
495	7469994624	Catchment Area 7	W	
496	7469994624	Catchment Area 7	W	
497	7469994624	Catchment Area 7	W	
498	7469994624	Catchment Area 7	W	
499	7469994624	Catchment Area 7	W	

	Status_Description	Type_Code	Type_Desc	State_Abbr	\
0	Withdrawn	Hpsa Geo	Geographic HPSA	NH	
1	Withdrawn	Hpsa Geo	Geographic HPSA	RI	
2	Withdrawn	Hpsa Geo	Geographic HPSA	RI	
3	Withdrawn	Hpsa Geo	Geographic HPSA	RI	
4	Withdrawn	Hpsa Geo	Geographic HPSA	RI	
..	
495	Withdrawn	Hpsa Geo	Geographic HPSA	SD	
496	Withdrawn	Hpsa Geo	Geographic HPSA	SD	
497	Withdrawn	Hpsa Geo	Geographic HPSA	SD	
498	Withdrawn	Hpsa Geo	Geographic HPSA	SD	
499	Withdrawn	Hpsa Geo	Geographic HPSA	SD	

	Degree_of_Shortage	Designation_Date	Designation_Last_Update_Date
0	Not applicable	29005	37799
1	Not applicable	37672	39671
2	Not applicable	37672	39671
3	Not applicable	37672	39671
4	Not applicable	37672	39671
..

495	Not applicable	36633	43283
496	Not applicable	36633	43283
497	Not applicable	36633	43283
498	Not applicable	36633	43283
499	Not applicable	36633	43283

[500 rows x 10 columns]

Table: hpsa_mua

	MUA_SOURCE_ID	MUA_AREA_CD	MUA_DESIGNATION_TYP_CD	\
0	474	9013530200	MUP	
1	474	9013530100	MUP	
2	470	9005310300	MUP	
3	470	9005310500	MUP	
4	470	9005310801	MUP	
..	
495	6165	34023004200	MUP-GE	
496	6165	34023004300	MUP-GE	
497	6165	34023004600	MUP-GE	
498	6165	34023005000	MUP-GE	
499	7539	34041030900	MUA	

	MUA_DESIGNATION_TYP_DESC	MUA_STATUS_CD	\
0	Medically Underserved Population	D	
1	Medically Underserved Population	D	
2	Medically Underserved Population	D	
3	Medically Underserved Population	D	
4	Medically Underserved Population	D	
..	
495	Medically Underserved Population ? ð Governor?...	D	
496	Medically Underserved Population ? ð Governor?...	D	
497	Medically Underserved Population ? ð Governor?...	D	
498	Medically Underserved Population ? ð Governor?...	D	
499	Medically Underserved Area	D	

	MUA_STATUS_DESC	CENSUS_TRACT	MUA_DESIGNATION_DT	MUA_DESIGNATION_DT_TXT	\
0	Designated	5302.0	34676	12/8/1994	
1	Designated	5301.0	34676	12/8/1994	
2	Designated	3103.0	34676	12/8/1994	
3	Designated	3105.0	34676	12/8/1994	
4	Designated	3108.01	34676	12/8/1994	
..	
495	Designated	42.0	36852	11/22/2000	
496	Designated	43.0	36852	11/22/2000	
497	Designated	46.0	36852	11/22/2000	
498	Designated	50.0	36852	11/22/2000	
499	Designated	309.0	34497	6/12/1994	

	MUA_SCORE
0	47.8
1	47.8
2	41.1
3	41.1
4	41.1
..	...
495	0.0
496	0.0
497	0.0
498	0.0
499	62.0

[500 rows x 10 columns]

Table: hpsa_primary_care

	Source_ID	Source_Name	Status_Code	\
0	1569995651	Ft. Washakie PHS Indian Health Center	D	
1	1469994698	McLaughlin PHS Indian Medical/Dental Clinic	D	
2	1469994687	Wagner PHS Indian Hospital	D	
3	141999413V	Portland Area Office	D	
4	14099940N2	Shawnee Health Center	D	
..	
495	10299902AE	Saint Paul Health Center	D	
496	10299902AD	Karluk Village Clinic	D	
497	10299902AC	Little Diomedes Clinic	D	
498	10299902AB	Eklutna Village Clinic	D	
499	1029990297	Tyonek Village Clinic	D	

	Status_Description	Type_Code	\
0	Designated	IHS	
1	Designated	IHS	
2	Designated	IHS	
3	Designated	IHS	
4	Designated	IHS	
..	
495	Designated	ITU	
496	Designated	ITU	
497	Designated	ITU	
498	Designated	ITU	
499	Designated	ITU	

	Type_Desc	\
0	Indian, Tribal and Urban Indian Organizations	
1	Indian, Tribal and Urban Indian Organizations	
2	Indian, Tribal and Urban Indian Organizations	

```

3      Indian, Tribal and Urban Indian Organizations
4      Indian, Tribal and Urban Indian Organizations
..
495 Indian Health Service, Tribal Health, and Urba...
496 Indian Health Service, Tribal Health, and Urba...
497 Indian Health Service, Tribal Health, and Urba...
498 Indian Health Service, Tribal Health, and Urba...
499 Indian Health Service, Tribal Health, and Urba...

```

	Address	City	State_Abbr	Postal_Code
0	PO BOX 128	Fort Washakie	WY	82514-0128
1	611 2nd Ave E	Mc Laughlin	SD	57642
2	110 Washington Ave NW	Wagner	SD	57380
3	1220 SW 3rd Ave Ste 476	Portland	OR	97204-2825
4	2307 Gordon Cooper Dr	Shawnee	OK	74801-9007
..
495	1000 Polivenia Tpke	Saint Paul Island	AK	99660
496	26 Alex Brown St	Karluk	AK	99608
497	None	Nome	AK	99762
498	26339 Eklutna Village Rd	Chugiak	AK	99567-5148
499	73 C St	Tyonek	AK	99682

[500 rows x 10 columns]

```

[ ]: import mysql.connector
from mysql.connector import Error

# Database connection details
host = 'localhost'
user = 'root'
password = ''
database = 'task_x'

# Function to create stored procedures
def create_stored_procedures():
    try:
        conn = mysql.connector.connect(host=host, user=user, password=password,
↪database=database)
        if conn.is_connected():
            cursor = conn.cursor()

            # Stored Procedure 1: Get data from hpsa_primary_care
            sp1_query = """
                CREATE PROCEDURE GetPrimaryCareData()
                BEGIN

```

```

        SELECT * FROM hpsa_primary_care;
    END
    """
    cursor.execute(sp1_query)
    print("Stored Procedure 1 created successfully.")

    # Stored Procedure 2: Get data from hpsa_dental_health
    sp2_query = """
        CREATE PROCEDURE GetDentalHealthData()
        BEGIN
            SELECT * FROM hpsa_dental_health;
        END
    """
    cursor.execute(sp2_query)
    print("Stored Procedure 2 created successfully.")

    # Commit the changes
    conn.commit()

except Error as e:
    print(f"Error: {e}")

finally:
    if conn.is_connected():
        cursor.close()
        conn.close()

# Create stored procedures
create_stored_procedures()

```

Error: 1558 (HY000): Column count of mysql.proc is wrong. Expected 21, found 20. Created with MariaDB 100108, now running 100428. Please use mysql_upgrade to fix this error

[]:

```

[ ]: import pandas as pd

# Function to execute queries and print results
def execute_queries():
    try:
        conn = mysql.connector.connect(host=host, user=user, password=password,
        ↪database=database)
        if conn.is_connected():
            cursor = conn.cursor(dictionary=True)

        # Query 1: Get unique cities from hpsa_primary_care

```

```

query1 = "SELECT DISTINCT City FROM hpsa_primary_care;"
result1 = pd.read_sql(query1, conn)
print("Query 1:")
print(result1)

# Query 2: Get the count of each Type_Desc from hpsa_dental_health
query2 = "SELECT Type_Desc, COUNT(*) as Count FROM ↵
↪hpsa_dental_health GROUP BY Type_Desc;"
result2 = pd.read_sql(query2, conn)
print("\nQuery 2:")
print(result2)

# Query 3: Get the average MUA_SCORE for each MUA_STATUS_DESC in ↵
↪hpsa_mua
query3 = "SELECT MUA_STATUS_DESC, AVG(MUA_SCORE) as Average_Score ↵
↪FROM hpsa_mua GROUP BY MUA_STATUS_DESC;"
result3 = pd.read_sql(query3, conn)
print("\nQuery 3:")
print(result3)

query4 = """
    SELECT Source_Name, Address
    FROM hpsa_primary_care
    ORDER BY Address DESC
    LIMIT 5;
"""
result4 = pd.read_sql(query4, conn)
print("\nQuery 4:")
print(result4)

except Error as e:
    print(f"Error: {e}")

finally:
    if conn.is_connected():
        cursor.close()
        conn.close()

# Execute queries
execute_queries()

```

Query 1:

	City
0	Fort Washakie
1	Mc Laughlin

2	Wagner
3	Portland
4	Shawnee
..	...
408	Saint Paul Island
409	Karluk
410	Nome
411	Chugiak
412	Tyonek

[413 rows x 1 columns]

Query 2:

	Type_Desc	Count
0	Indian Health Service, Tribal Health, and Urba...	474
1	Indian, Tribal and Urban Indian Organizations	26

Query 3:

MUA_STATUS_DESC	Average_Score
0 Designated	47.70276

Query 4:

	Source_Name	Address
0	COCHITI DENTAL CLINIC	Windmill Rd
1	WHITE HORSE HS	Whitehorse Road/County 4
2	K'ima:w Medical Center	Weitchpec Route, Libby Nix Community Center
3	CANNONBAL HEALTH STATION	Weasel St
4	WAKPALA HEALTH STATION	Wakpala Road

<ipython-input-141-1c5ac15f1627>:12: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
result1 = pd.read_sql(query1, conn)
```

<ipython-input-141-1c5ac15f1627>:18: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
result2 = pd.read_sql(query2, conn)
```

<ipython-input-141-1c5ac15f1627>:24: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.

```
result3 = pd.read_sql(query3, conn)
```

<ipython-input-141-1c5ac15f1627>:36: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.


```
result4 = pd.read_sql(query4, conn)
```

0.0.7 Conclusion:

The project successfully leverages Python and MySQL to initialize and populate a database with four tables (hpsa_primary_care, hpsa_mental_health, hpsa_dental_health, and hpsa_mua). The script utilizes the pandas library to handle datasets, dynamically generates SQL queries for table creation and data insertion, and ensures proper error handling and connection closure.

The database is structured to store information related to healthcare provider shortage areas, mental health designations, dental health designations, and medically underserved areas. This organized data lays the foundation for efficient querying and analysis.

Additionally, the script includes functionality to perform user-defined SELECT queries, allowing users to retrieve specific information from the populated tables interactively.

Overall, the project combines data management, database creation, and user interaction, providing a robust foundation for further development and analysis in the realm of healthcare data.