Sruthi Kondra

Email: kondra.s@northeastern.edu

College of Professional Studies, Northeastern University

Introduction

This project focuses on designing and implementing a star schema database to analyze hospital bed capacity across various healthcare facilities. The primary goal is to structure a relational database that efficiently stores and retrieves data related to hospital bed types, hospital details, and bed availability metrics such as licensed beds, census beds, and staffed beds. By organizing the data in a structured format, we aim to answer key business questions that help identify hospitals with the highest availability of ICU (Intensive Care Unit) and SICU (Surgical Intensive Care Unit) beds.

For this analysis, the Hospital Bed Capacity dataset was selected. It contains critical information about hospital bed availability and hospital details, which are essential for understanding healthcare infrastructure and capacity management. The dataset consists of three primary tables: bed_type, business, and bed_fact, each serving as a key component of the star schema.

To achieve the objectives of this project, MySQL Workbench was used for database creation, table setup, and data import. The star schema was designed using ERDPlus, ensuring a structured approach to linking dimension and fact tables. The implementation involved writing SQL queries to clean, validate, and analyze the data, including checking for duplicates and aggregating bed capacity across hospitals. The final goal was to extract meaningful insights regarding ICU and SICU bed availability by performing advanced SQL queries.

Analysis

Identifying Dimensions from the Two Dimension Tables

Table 1: Bed Type

Three key dimensions are identified in the bed_type table:

1. **bed_id**: This serves as the primary key of the table and uniquely identifies each type of bed in the dataset. It is essential for establishing relationships with other tables in the schema.
2. **bed_code**: A concise code that acts as a shorthand for the bed type. This dimension is particularly useful for summarization and categorization during analysis.

3. **bed_desc**: The description of the bed type, such as "ICU," "SICU," or "Pediatric ICU." This dimension provides critical information to differentiate between various bed types and specialized care facilities.

Table 2: Business

Three key dimensions are identified in the business table:

1. **ims_org_id**: This is the primary key of the table and is crucial for uniquely identifying each hospital or medical center in the dataset.
2. **business_name**: This represents the name of the hospital or medical center, adding context to the data and making the results more interpretable for reporting purposes.
3. **bed_cluster_id**: This indicates a categorization or grouping of beds within hospitals, such as specialized units or departments. This dimension provides an additional layer of granularity for analyzing bed types across different clusters.

Identifying Facts from the Fact Table

Table: Bed Fact

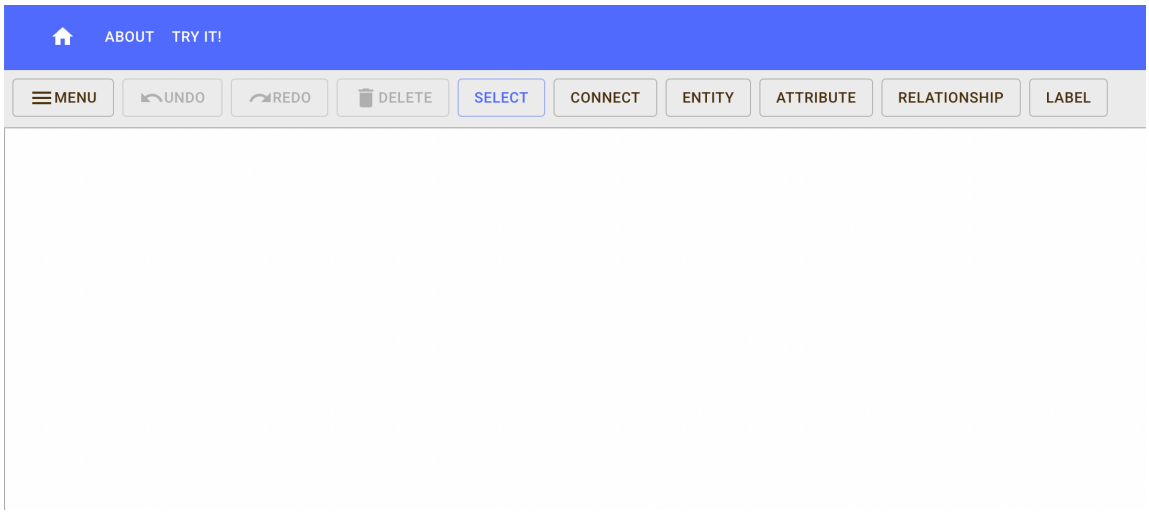
The bed_fact table contains measurable data points essential for analysis. The following facts were identified:

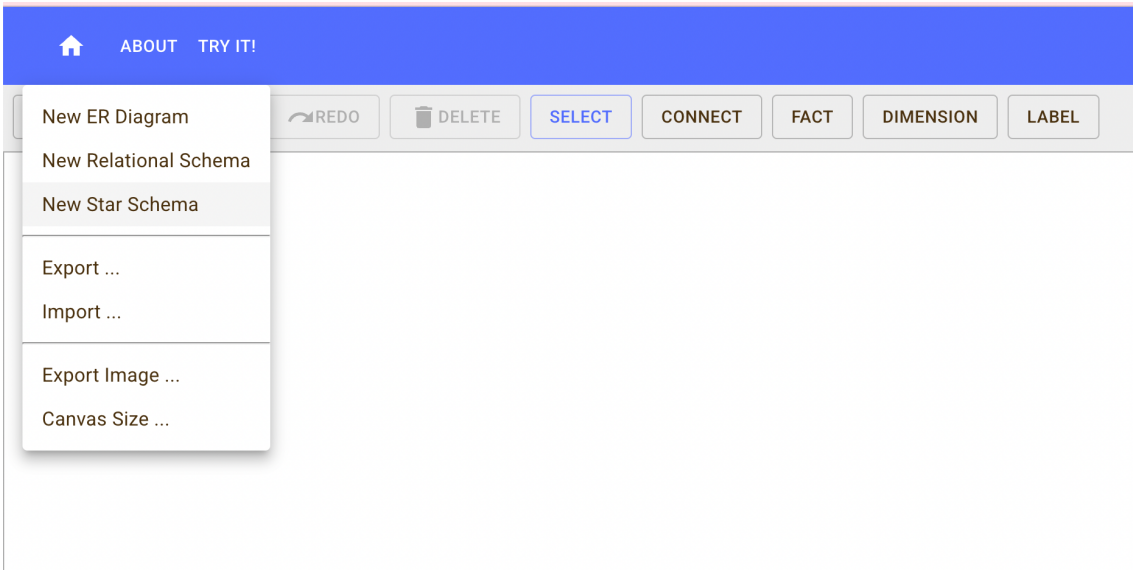
1. **license_beds**: This fact captures the total number of beds a hospital is licensed to operate, providing a measure of its authorized capacity as per state regulations.
2. **census_beds**: This represents the number of beds currently in use, reflecting real-time occupancy and utilization within the hospital. It is a critical metric for understanding resource usage.
3. **staffed_beds**: This fact captures the number of beds that are fully staffed and ready for operation, showcasing the hospital's actual functional capacity based on available medical personnel.

These facts are additive, meaning they can be aggregated across dimensions like hospitals or bed types to analyze total capacity, utilization rates, or staffing across various contexts. This makes them essential for answering key business questions and deriving actionable insights.

Star Schema:

To create the star schema, I started by navigating to **ERDPlus**, a powerful tool for designing data models. From the menu, I selected the option to create a new **Star Schema**. This provided a clean and blank workspace, which served as the foundation for constructing my schema. The goal was to build a schema that linked a fact table containing measurable data to dimension tables providing descriptive information about the entities in the dataset. With this structure in mind, I began creating the required tables step by step.

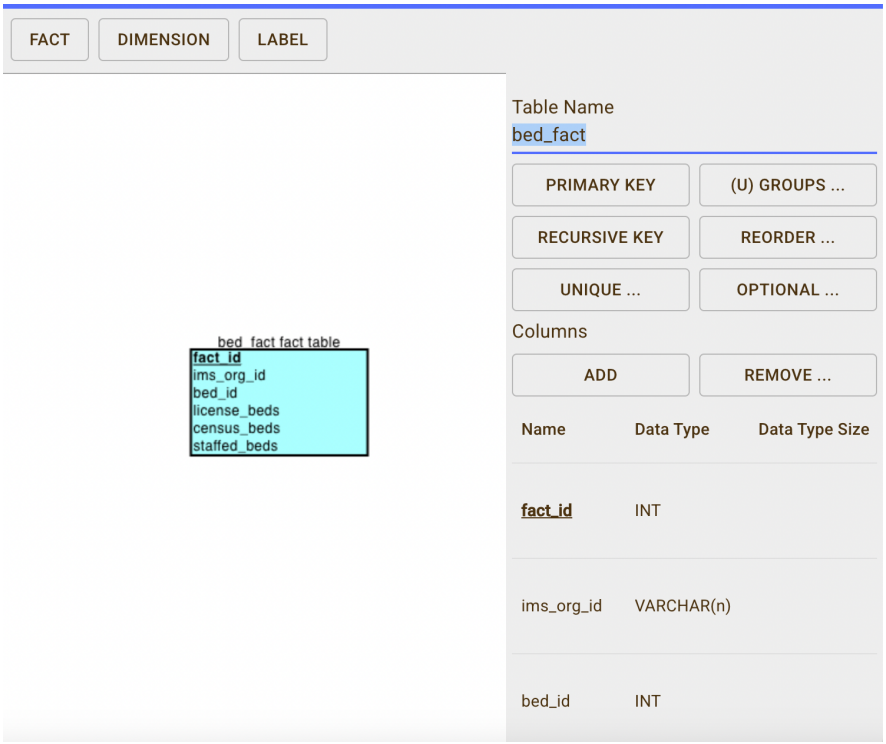




The first table I created was the **Fact Table**, named bed_fact. Using the **FACT** button in the toolbar, I added this central table to my schema. The fact table is designed to store quantitative and measurable data about the dataset. I added the following columns to the bed_fact table:

- **fact_id**: This serves as the primary key for the table. It is an auto-incrementing integer (INT) that uniquely identifies each row.
- **ims_org_id**: This column is a foreign key (VARCHAR(n)) that links to the ims_org_id column in the business dimension table. It represents the unique identifier for each hospital.
- **bed_id**: This column is also a foreign key (INT) that links to the bed_id column in the bed type dimension table. It identifies the type of bed in the hospital.
- **license_beds**, **census_beds**, and **staffed_beds**: These columns, all set as integers (INT), represent measurable data. They indicate the total licensed beds, census beds, and staffed beds available at each hospital for the given bed type.

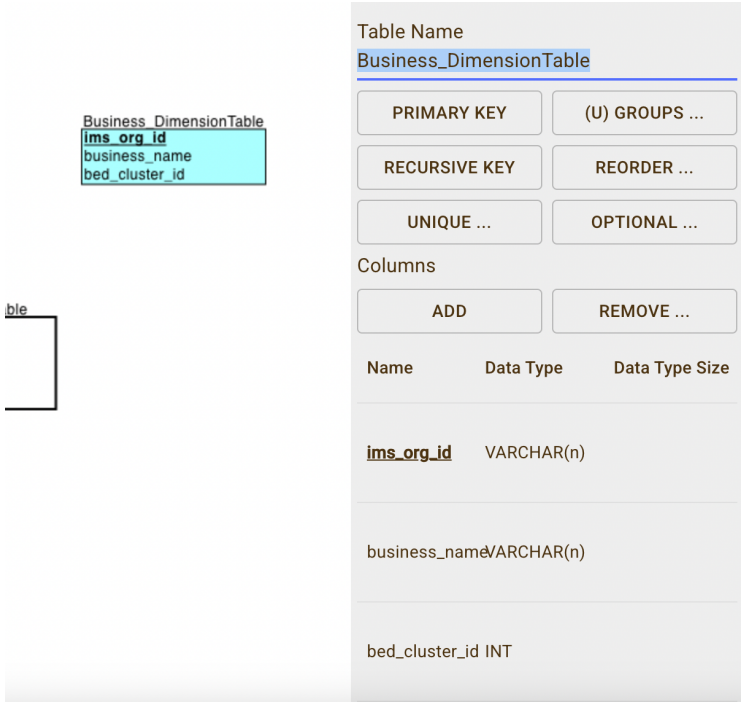
Once the columns were defined, I marked fact_id as the primary key and ensured ims_org_id and bed_id were properly labeled as foreign keys. This setup allowed the fact table to act as the central repository for data, linking it to descriptive information from the dimension tables.



Next, I created the **Business Dimension Table**, which provides descriptive information about the hospitals in the dataset. Using the **DIMENSION** button, I added a new table named Business_DimensionTable. I added the following columns:

- **ims_org_id**: This column serves as the primary key (VARCHAR(n)) and uniquely identifies each hospital in the dataset.
- **business_name**: This column (VARCHAR(n)) stores the name of the hospital or medical center, adding context to the dataset.
- **bed_cluster_id**: This column (INT) categorizes or groups beds within a hospital, representing specialized units or departments.

After defining the columns, I set ims_org_id as the primary key for the table, ensuring that each record in the business dimension table could be uniquely identified. This table connects to the fact table through the ims_org_id foreign key, linking each hospital to its corresponding data in the fact table.

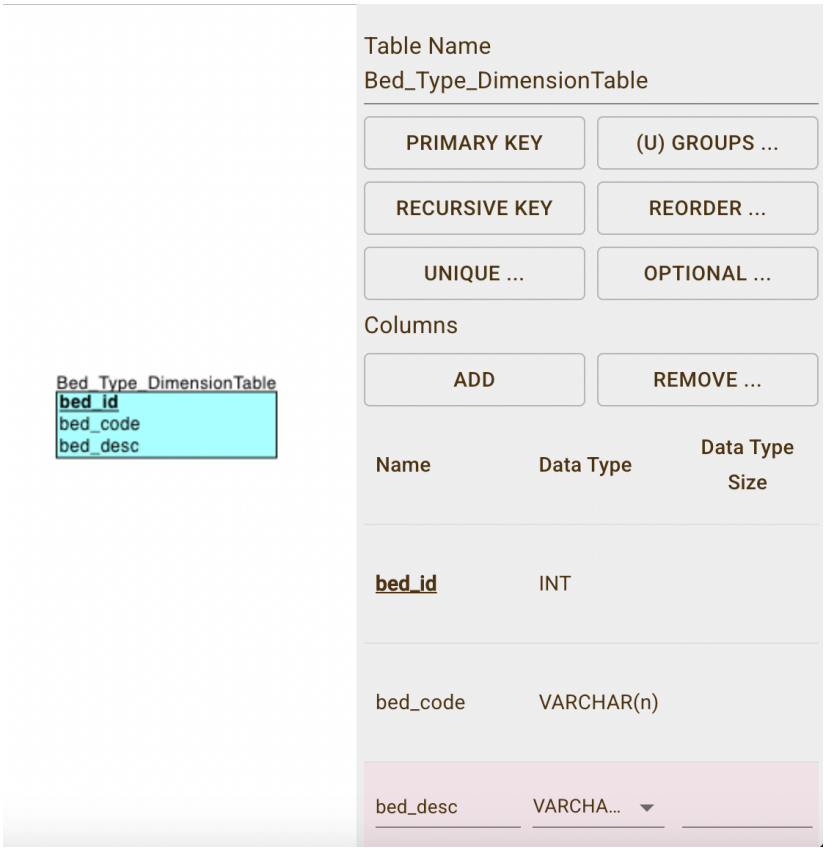


Following this, I created the **Bed Type Dimension Table**, which provides information about the types of beds available in the hospitals. Again, I used the **DIMENSION** button to add the table and named it Bed_Type_DimensionTable. I added the following columns:

- **bed_id**: This column serves as the primary key (INT) and uniquely identifies each type of bed in the dataset.
- **bed_code**: This column (VARCHAR(n)) provides a shorthand code for the bed type, allowing for quick reference during analysis.

- **bed_desc**: This column (VARCHAR(n)) provides a detailed description of the bed type, such as "ICU" or "SICU."

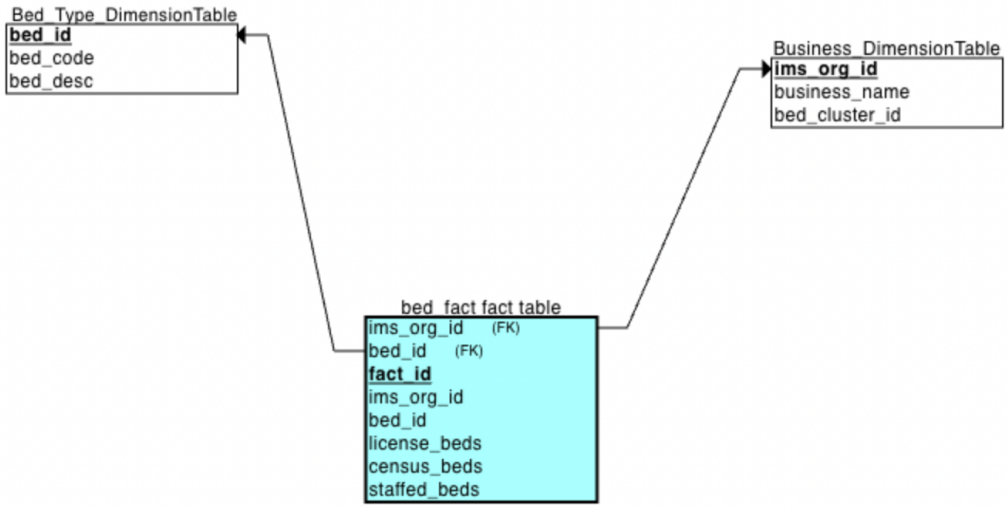
Once the columns were added, I set bed_id as the primary key, ensuring each bed type in the dataset could be uniquely identified. This table links to the fact table through the bed_id foreign key, allowing the fact table to reference specific bed types.



Finally, I established the relationships between the tables. Using the **CONNECT** button, I linked the dimension tables to the fact table:

1. I connected the **ims_org_id** column in the Business_DimensionTable to the **ims_org_id** column in the bed_fact table. This created a **1:N relationship**, indicating that one hospital (represented by a single ims_org_id) could have multiple rows in the fact table corresponding to different bed types.
2. I connected the **bed_id** column in the Bed_Type_DimensionTable to the **bed_id** column in the bed_fact table. This also created a **1:N relationship**, as one bed type could appear in multiple rows of the fact table, depending on the hospitals and their capacities.

These relationships ensured the star schema was properly linked, with the fact table at the center connected to the two dimension tables.



The completed schema consists of the bed_fact table as the fact table, containing quantitative data, and two dimension tables—Business_DimensionTable and Bed_Type_DimensionTable—providing descriptive information. This structure allows for efficient analysis and querying of data, making it ready for further exploration and SQL queries.

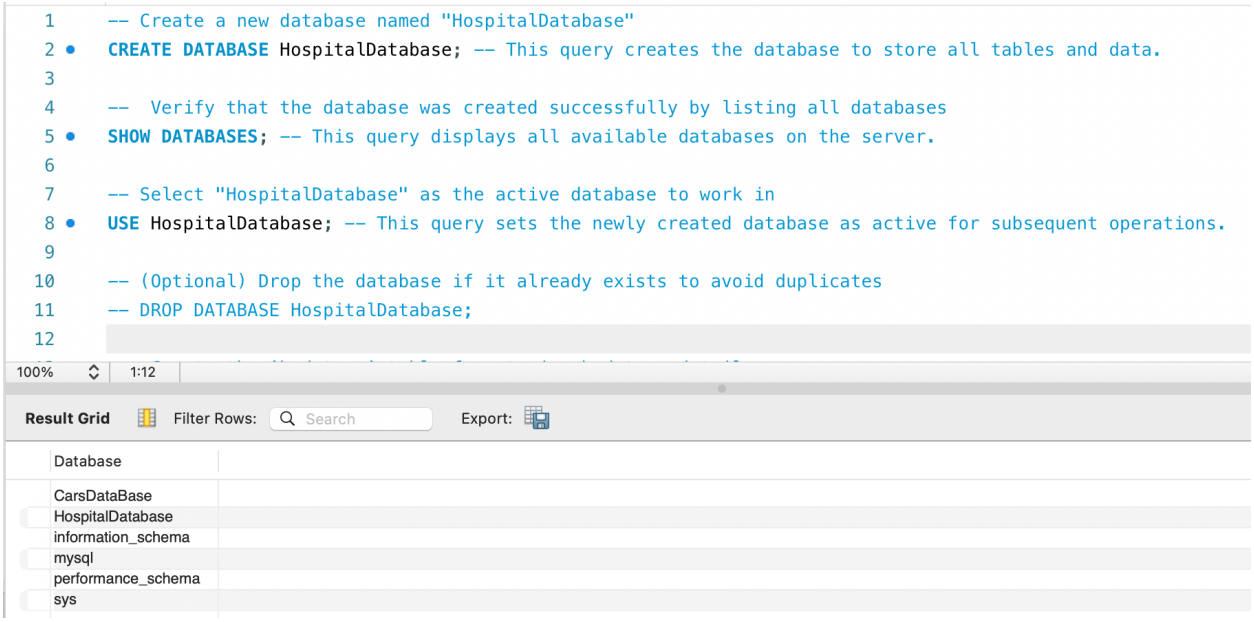
Creating the Database , Table Setup, and Data Import

For this phase of the project, I worked on setting up the database, creating the necessary tables, and importing data into them using MySQL Workbench. This process ensured that the schema was structured correctly and contained all the required data for analysis. Below is a detailed explanation of each step I performed.

Creating the Database

The first step was to create a new database in MySQL Workbench to store all tables. I executed the CREATE DATABASE command to create a schema named **HospitalDatabase**. This schema served as the foundation where all tables, including dimension and fact tables, would reside.

After running the command, I wanted to confirm that the database was successfully created, so I executed SHOW DATABASES, which listed all available databases on the server. I checked the list and verified that **HospitalDatabase** appeared in the results. Once I ensured that the database was created successfully, I set it as the active database by using the USE HospitalDatabase command. This step was crucial because it ensured that all subsequent table creation queries were executed within the intended schema rather than any other database present in the system.



```
1   -- Create a new database named "HospitalDatabase"
2   CREATE DATABASE HospitalDatabase; -- This query creates the database to store all tables and data.
3
4   --  Verify that the database was created successfully by listing all databases
5   SHOW DATABASES; -- This query displays all available databases on the server.
6
7   -- Select "HospitalDatabase" as the active database to work in
8   USE HospitalDatabase; -- This query sets the newly created database as active for subsequent operations.
9
10  -- (Optional) Drop the database if it already exists to avoid duplicates
11  -- DROP DATABASE HospitalDatabase;
12
```

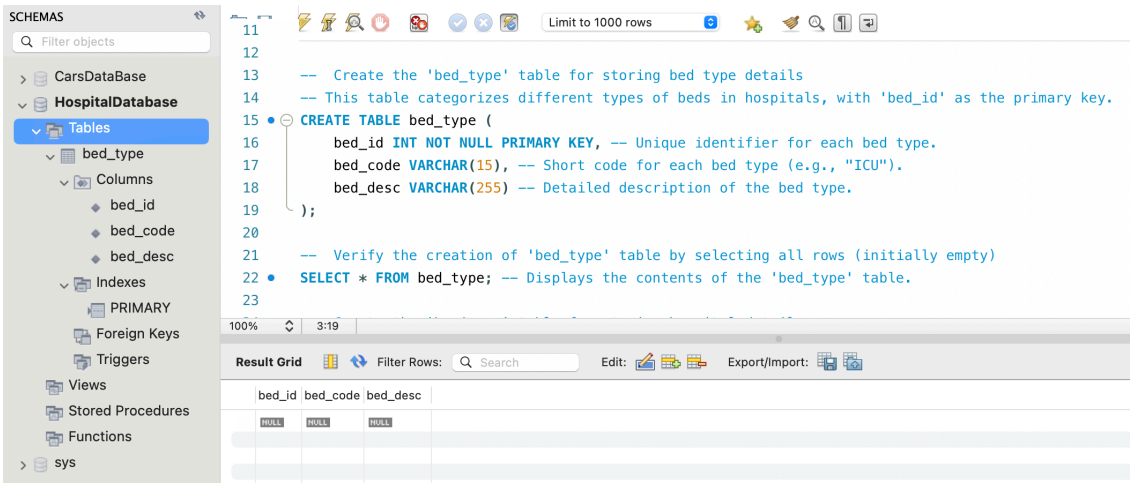
100% 1:12

Result Grid Filter Rows: Search Export:

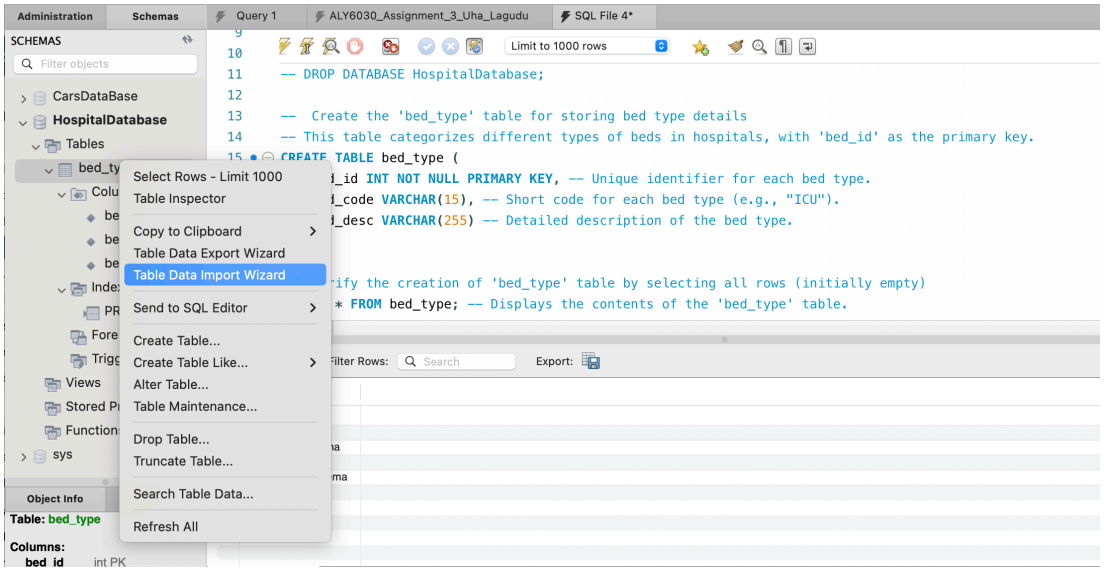
Database
CarsDataBase
HospitalDatabase
information_schema
mysql
performance_schema
sys

Bed Type Table: Creation and Data Import

The first table I created was bed_type, which stored information about different types of beds available in hospitals. This table was essential because it provided a reference for various bed categories. I wrote the SQL query to create this table, ensuring that bed_id was the **primary key**. Additionally, I included bed_code, which provided a short identifier for each bed type, and bed_desc, which stored a full description of the bed type. Once the table was created, I ran the SELECT * FROM bed_type query to confirm its structure, ensuring that all columns were properly defined.



After creating the bed_type table, I proceeded to import data into it. To do this, I navigated to MySQL Workbench, located the bed_type table in the **Schemas** section, and right-clicked on it. I then selected the **Table Data Import Wizard** option, which opened a window allowing me to browse for the CSV file containing the data. I selected the dataset file named bed_type-1.csv and clicked **Next** to proceed with the import process. After selecting the file, I was prompted to specify the destination for the imported data. I chose **Use existing table: HospitalDatabase.bed_type** to ensure that the data was loaded into the correct table. In the next step, I mapped the source columns from the CSV file (bed_id, bed_code, and bed_desc) to the corresponding columns in the table. After verifying the mappings, I completed the import process. To confirm that the data was successfully inserted, I executed SELECT * FROM bed_type, which displayed all the records from the imported file.



Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: /Users/sruthikondra/Downloads/bed_type-1.csv Browse...

Select Destination

Select destination table and additional options.

Use existing table: HospitalDatabase.bed_type

Create new table: HospitalDatabase . bed_type-1

Truncate table before import

Table Data Import

Configure Import Settings

Detected file format: csv

Encoding: utf-8

Source Column	Dest Column
bed_id	bed_id
bed_code	bed_code
bed_desc	bed_desc

bed_id	bed_code	bed_desc
1	BU	Burn
2	CC	CCU
3	DE	Detox ICU

< Back Next > Cancel

Import Data

The following tasks will now be performed. Please monitor the execution.

- Prepare Import
- Import data file

Finished performing tasks. Click [Next >] to continue.

```
20
21      --  Verify the creation of 'bed_type' table by selecting all rows
22  •   SELECT * FROM bed_type; -- Displays the contents of the 'bed_type' table.
23
```

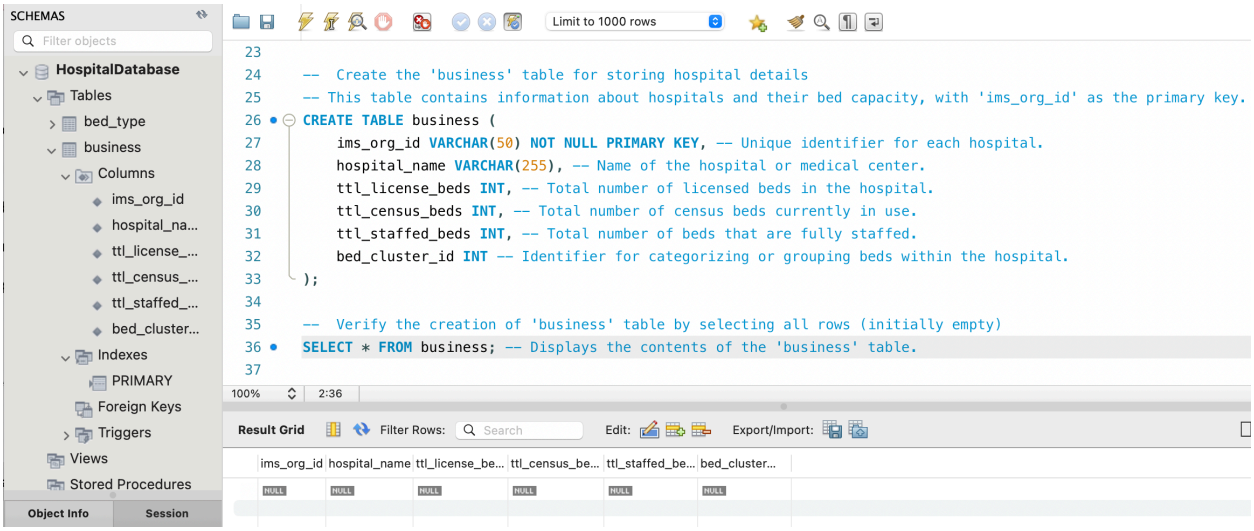
100% 66:21

Result Grid Filter Rows: Search Edit: Export/Import:

bed_id	bed_code	bed_desc
1	BU	Burn
2	CC	CCU
3	DE	Detox ICU
4	IC	ICU
5	MS	Med/Surg
6	NE	NeoNatal ICU
7	NU	Nursery
8	NF	Nursing Facility
9	OT	Other
10	PD	Pediatric ICU
11	PR	Premature ICU
12	PS	Psychiatric
13	PI	Psych ICU
14	RH	Rehabilitation
15	SI	SICU
16	SN	Skilled Nursing
17	SP	Special Care
18	TO	Total
19	TR	Trauma ICU
20	DD	Development…
NULL	NULL	NULL

Business Table: Creation and Data Import

After defining the bed_type table, I proceeded to create the business table, which contained essential details about hospitals. This table included the **primary key** column ims_org_id, which uniquely identified each hospital, along with hospital_name to store the hospital's name. Additionally, I included columns for ttl_license_beds, ttl_census_beds, and ttl_staffed_beds, which provided a count of the hospital's licensed, census, and staffed beds, respectively. Another column, bed_cluster_id, was added to categorize or group hospital beds based on a classification system. After writing and executing the SQL query, I checked the table structure by running SELECT * FROM business to confirm that it was created successfully.



Once the business table was created, I proceeded to import data into it using the same **Table Data Import Wizard** process. I right-clicked on the business table, selected the **Table Data Import Wizard**, and uploaded the corresponding CSV file containing hospital information. I ensured that the correct destination table was selected and mapped the columns accordingly. Once the import was completed, I ran SELECT * FROM business to verify that the hospital data was successfully inserted.

Select Destination

Select destination table and additional options.

Use existing table: HospitalDatabase.business

Create new table: HospitalDatabase . business-1

Truncate table before import

Configure Import Settings

Detected file format: csv

Encoding: utf-8

Source Column	Dest Column
ims_org_id	ims_org_id
business_name	hospital_name
ttl_license_beds	ttl_license_beds
ttl_census_beds	ttl_census_beds
ttl_staffed_beds	ttl_staffed_beds

ims_org_id	business_n...	ttl_license_...	ttl_census_...	ttl_staffed_...	bed_cluster...
INS0000...	Hegg Me...	25	15	25	1
INS0000...	Communi...	16	8	16	1
INS0000...	Pipestone...	45	10	25	1

```
35    --  Verify the creation of 'business' table by selecting all rows
36    SELECT * FROM business; -- Displays the contents of the 'business' table.
37
38    -- Create the 'bed_fact' table for storing measurable data
```

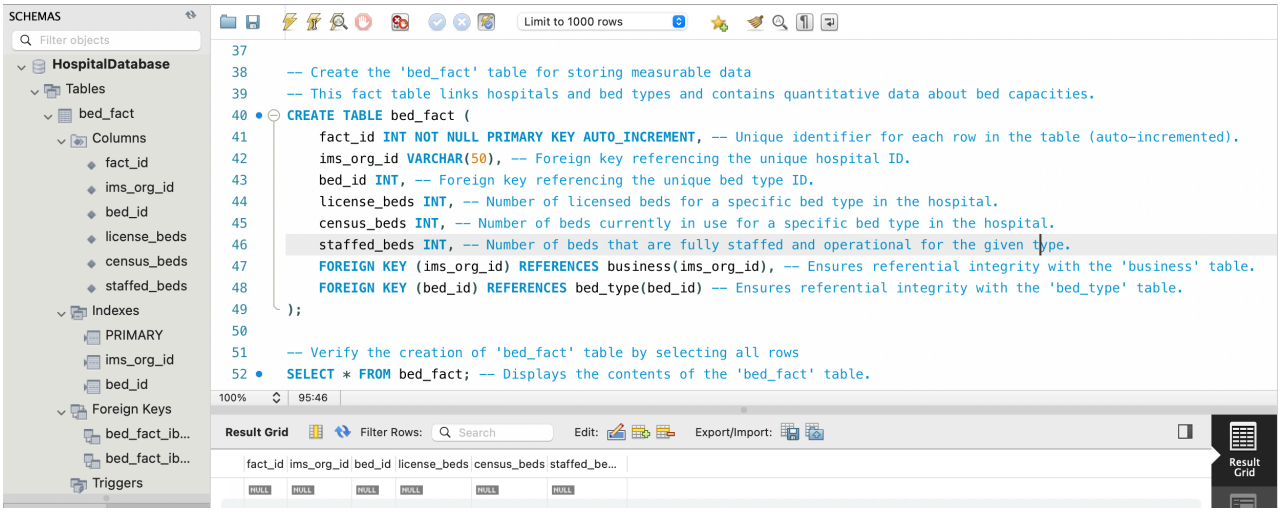
100% 66:35

Result Grid Filter Rows: Search Edit: Export/Import: Fetch row

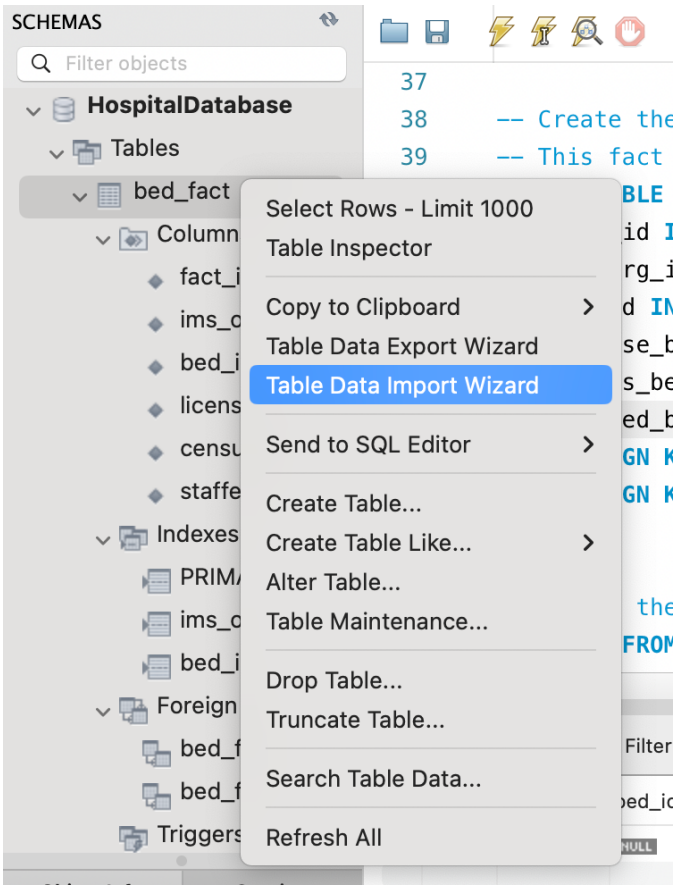
ims_org_id	hospital_name	ttl_license_be...	ttl_census_be...	ttl_staffed_be...	bed_cluster...
INS00000519	Abington Memorial Hospital	671	466	671	5
INS00000520	Lower Oconee Community Hospital, Inc	25	14	25	1
INS00000521	Stewart Webster Hospital, Inc	25	16	25	1
INS00000522	Glens Falls Hospital	410	300	315	3
INS00000523	Washington Adventist Hospital	288	237	288	3
INS00000524	Shady Grove Adventist Hospital	336	300	336	3
INS00000525	Hackettstown Regional Medical Center	111	60	111	1
INS00000526	Castle Medical Center	160	105	160	2
INS00000527	Feather River Hospital	101	70	80	1
INS00000529	Willits Hospital, Inc	25	16	25	1
INS00000530	Paradise Valley Hospital	301	142	205	2
INS00000531	Portland Adventist Medical Center	312	130	173	2
INS00000532	Saint Helena Hospital	151	85	151	1
INS00000533	San Joaquin Community Hospital	259	200	259	2
INS00000534	Sonora Regional Medical Center	152	49	152	1
INS00000535	Glendale Adventist Medical Center	457	350	457	4
INS00000536	Simi Valley Hospital and Healthcare S...	188	125	188	2
INS00000537	White Memorial Medical Center	353	275	275	3
INS00000538	Northwest Medical Foundation of Tilla...	49	14	25	1
INS00000539	Ukiah Adventist Hospital	78	45	78	1

Bed Fact Table: Creation and Data Import

Once the dimension tables were created, I moved on to defining the **fact table**, bed_fact, which was central to the schema. This table linked hospital data to bed type data while capturing important numerical attributes about bed occupancy. I designed bed_fact with fact_id as the **primary key**, which was set to auto-increment, ensuring that each row had a unique identifier. I also included ims_org_id and bed_id as **foreign keys**, linking this table to the business and bed_type tables, respectively. Additionally, I defined three numeric fields: license_beds, census_beds, and staffed_beds, which captured the number of licensed, census, and staffed beds for each hospital and bed type combination. Once the table was successfully created, I executed SELECT * FROM bed_fact to confirm that it was properly set up.



For importing data into the bed_fact table, I followed the same **Table Data Import Wizard** process as before. I navigated to the **Schemas** panel, selected the bed_fact table, and opened the **Table Data Import Wizard**. I selected the CSV file containing the fact table data and mapped the columns appropriately before proceeding with the import. Once the process was complete, I executed SELECT * FROM bed_fact to ensure that the records had been successfully inserted into the database.



Select File to Import

Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.

File Path: /Users/sruthikondra/Downloads/bed_fact-1.csv Browse...

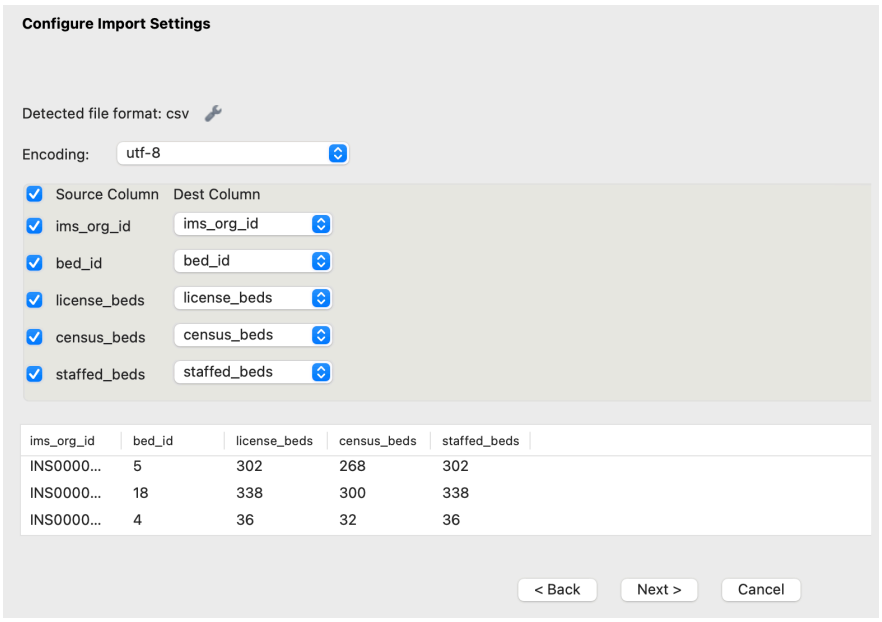
Select Destination

Select destination table and additional options.

Use existing table: HospitalDatabase.bed_fact

Create new table: HospitalDatabase . bed_fact-1

Truncate table before import




```
51  -- Verify the creation of 'bed_fact' table by selecting all rows
52  SELECT * FROM bed_fact; -- Displays the contents of the 'bed_fact' table.
53
54
```

100% 1:53

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows

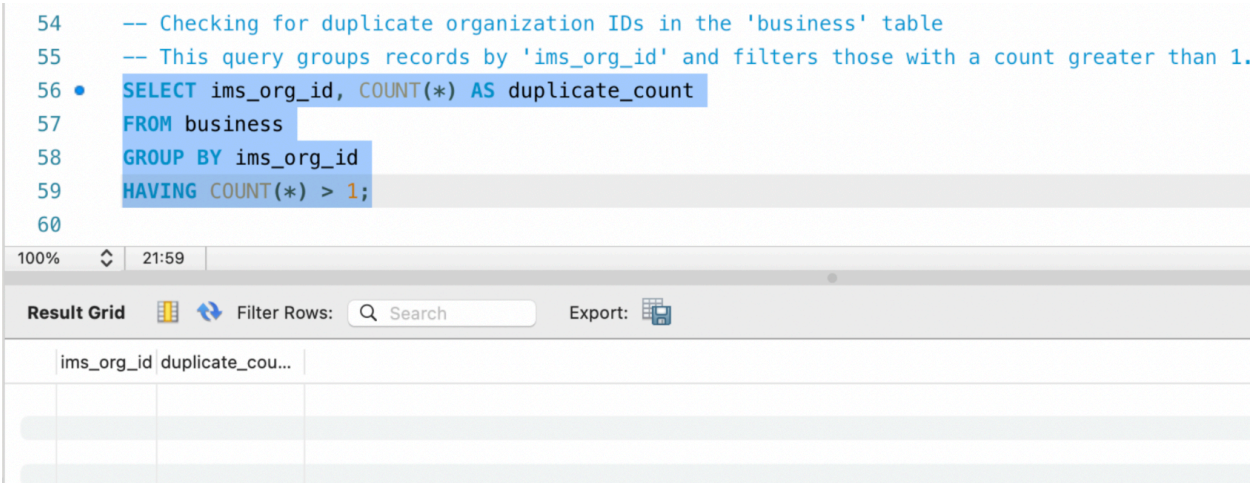
fact_id	ims_org_id	bed_id	license_beds	census_beds	staffed_be...
1	INS00002280	5	302	268	302
2	INS00002280	18	338	300	338
3	INS00002280	4	36	32	36
4	INS00005051	4	23	4	15
5	INS00005051	18	134	22	90
6	INS00005051	5	111	18	75
7	INS00040707	8	183	170	183
8	INS00040707	18	183	170	183
9	INS00109837	18	150	131	150
10	INS00108529	5	56	38	56
11	INS00109837	8	150	131	150
12	INS00108529	18	56	38	56
13	INS00004081	18	25	14	25
14	INS00004081	5	25	14	25
15	INS00113922	18	55	50	55
16	INS00109698	8	121	102	121
17	INS00113922	8	55	50	55
18	INS00109698	18	121	102	121
19	INS00038263	18	176	160	176
20	INS00035800	18	63	63	63
21	INS00038263	8	176	160	176

Through this process, I successfully created the **HospitalDatabase** schema, defined three tables with appropriate primary and foreign key constraints, and imported data into them. Each step was carefully executed to ensure that the schema structure adhered to the star schema model. The data import process using MySQL Workbench's **Import Wizard** was seamless, allowing me to populate the tables efficiently. With the database now fully set up and populated, it is ready for querying and further analysis.

Checking for Duplicates in the Business Table

After successfully importing the data into the business table, the next step was to check for any duplicate values that could affect data integrity. Ensuring that key fields, such as ims_org_id (organization ID) and hospital_name, are unique is essential for maintaining a clean and accurate dataset. If duplicates exist in these fields, it could lead to inconsistencies in analysis and reporting.

This query groups all records in the business table by ims_org_id and counts the occurrences of each ID. The HAVING COUNT(*) > 1 condition filters out records where the same organization ID appears more than once. Since ims_org_id serves as the **primary key**, there should be no duplicates in this column. After running the query, the output showed **no duplicate records**, which confirms that each organization has a unique identifier, maintaining the integrity of the dataset.



Next, I checked for duplicate **hospital names** in the dataset, as multiple hospitals may share the same name due to different locations or affiliations. This query groups records by hospital_name

and counts how many times each hospital name appears in the dataset. The HAVING COUNT(*) > 1 condition filters out hospitals that appear only once, and ORDER BY occurrence_count DESC ensures that the most frequently duplicated names appear at the top. The output revealed multiple hospitals with duplicate names, which is expected in a dataset containing hospitals from various locations. Some hospital names, such as **"The Arc North West Indiana"** and **"Millers Merry Manor"**, appeared more than 20 times. This suggests that these hospitals either have multiple branches or are part of a larger healthcare network.

```
63 •  SELECT hospital_name, COUNT(*) AS occurrence_count
64    FROM business
65    GROUP BY hospital_name
66    HAVING COUNT(*) > 1
67    ORDER BY occurrence_count DESC;
```

100% 1:68

Result Grid Filter Rows: Search Export:

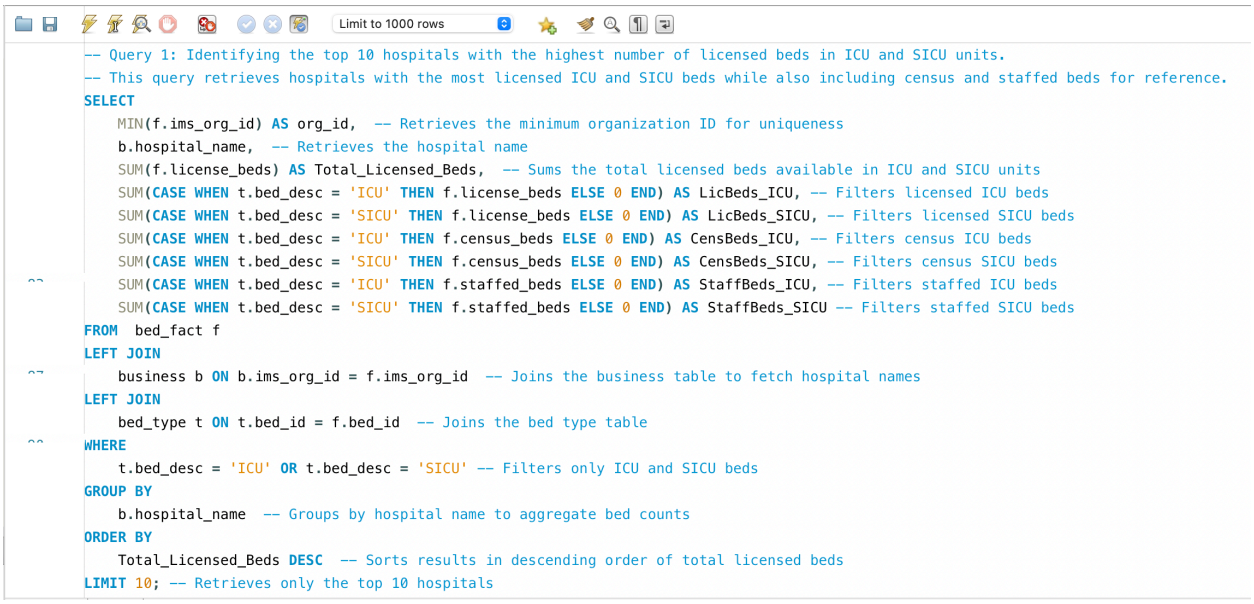
hospital_name	occurrence_co...
The Arc North West Indiana	25
Millers Merry Manor	23
Residential CRF, Inc	16
CareMeridian	13
Genesis HealthCare Corporation	11
Health and Hospital Corporation of Marion County	10
Little Sisters of The Poor	9
Saint Marys Hospital	8
HealthSouth Rehabilitation Hospital	8
Parkview Care Center	8
Good Samaritan Hospital	7
Prime Healthcare Services, Inc	7
Lifespace Communities, Inc	7
Mercy Hospital	6
Saint Joseph Hospital	6
Saint Joseph Medical Center	6
Saint Francis Medical Center	6
South Broward Hospital District	6
Heritage Health	6
Mercy Medical Center	5
Saint Anthony Hospital	5

Identifying these duplicates is crucial for further analysis. If the goal is to analyze hospital-level data, we may need additional attributes, such as **location or organization ID**, to differentiate between hospitals with the same name. Since ims_org_id remains unique, we can still distinguish between hospitals effectively. Now that we have verified the uniqueness of ims_org_id and identified duplicate hospital names, we can proceed with further data validation and business analysis.

Business Questions:

Question 1: Identifying the Top 10 Hospitals with the Most Licensed ICU and SICU Beds

The first query aimed to retrieve hospitals that have the highest number of licensed ICU and SICU beds. Licensed beds indicate the number of beds a hospital is legally permitted to operate based on state regulations. Additionally, I included the census beds (occupied beds) and staffed beds (beds with available medical staff) to give a more detailed breakdown of hospital capacity. To achieve this, I joined the bed_fact table, which contains hospital bed data, with the business table to fetch hospital names and with the bed_type table to filter only ICU and SICU beds. I used **CASE WHEN** conditions to categorize the beds based on their type, ensuring the correct allocation to either ICU or SICU. The data was aggregated using **SUM()** to calculate the total count of beds per hospital, and the results were sorted in descending order based on the total number of licensed beds. The final output presents the top 10 hospitals with the highest licensed ICU and SICU beds, providing key insights into hospital preparedness for critical care patients. The results indicate that *Saint Mary's Hospital* and *Saint Anthony Hospital* rank at the top, highlighting their significant ICU and SICU bed capacity. These hospitals are better equipped for high patient intake in critical care situations.



```
Limit to 1000 rows
-- Query 1: Identifying the top 10 hospitals with the highest number of licensed beds in ICU and SICU units.
-- This query retrieves hospitals with the most licensed ICU and SICU beds while also including census and staffed beds for reference.
SELECT
    MIN(f.ims_org_id) AS org_id,  -- Retrieves the minimum organization ID for uniqueness
    b.hospital_name,  -- Retrieves the hospital name
    SUM(f.license_beds) AS Total_Licensed_Beds,  -- Sums the total licensed beds available in ICU and SICU units
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.license_beds ELSE 0 END) AS LicBeds_ICU, -- Filters licensed ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.license_beds ELSE 0 END) AS LicBeds_SICU, -- Filters licensed SICU beds
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.census_beds ELSE 0 END) AS CensBeds_ICU, -- Filters census ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.census_beds ELSE 0 END) AS CensBeds_SICU, -- Filters census SICU beds
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_ICU, -- Filters staffed ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_SICU -- Filters staffed SICU beds
FROM  bed_fact f
LEFT JOIN
    business b ON b.ims_org_id = f.ims_org_id  -- Joins the business table to fetch hospital names
LEFT JOIN
    bed_type t ON t.bed_id = f.bed_id  -- Joins the bed type table
WHERE
    t.bed_desc = 'ICU' OR t.bed_desc = 'SICU' -- Filters only ICU and SICU beds
GROUP BY
    b.hospital_name  -- Groups by hospital name to aggregate bed counts
ORDER BY
    Total_Licensed_Beds DESC  -- Sorts results in descending order of total licensed beds
LIMIT 10; -- Retrieves only the top 10 hospitals
```

org_id	hospital_name	Total_Licensed_Be...	LicBeds_ICU	LicBeds_SICU	CensBeds_ICU	CensBeds_SICU	StaffBeds_ICU	StaffBeds_SICU
INS00001097	Saint Marys Hospital	379	360	19	193	12	284	16
INS00000728	Saint Anthony Hospital	337	337	0	190	0	288	0
INS00000802	Phoenix Childrens Hospital	247	247	0	117	0	159	0
INS00001157	Good Samaritan Hospital	232	220	12	112	7	181	11
INS00006508	University of Maryland Medical Center	220	104	116	60	67	81	90
INS00001707	UC Health University Hospital	218	162	56	82	28	112	39
INS00001382	Wesley Medical Center, LLC	214	214	0	98	0	162	0
INS00005273	South Broward Hospital District	207	207	0	145	0	168	0
INS00006486	Vidant Medical Center	204	204	0	123	0	203	0
INS00005218	MCGHealth, Inc	200	179	21	96	11	117	14

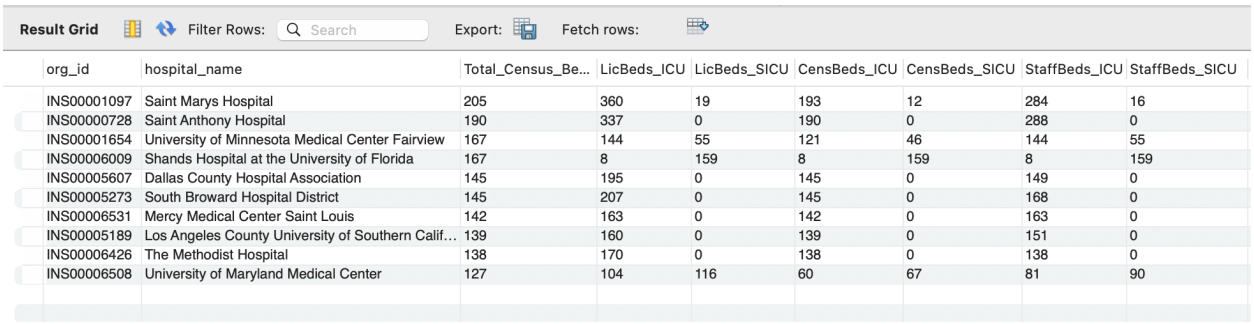
Question 2: Identifying the Top 10 Hospitals with the Most Census ICU and SICU Beds

The second query focused on census beds, which represent the actual beds in use by patients at the time of data collection. This metric provides a real-time view of ICU and SICU occupancy and is crucial for understanding hospital strain and patient intake levels.

I used a similar approach as the first query, but instead of filtering by licensed beds, I retrieved the total **census beds** per hospital. The query utilized **SUM()** to aggregate census beds for ICU and SICU categories separately and presented them in a structured format.

The final output ranked hospitals based on the number of census beds, showing that *Saint Mary's Hospital* again led in total census ICU and SICU beds, followed by *Saint Anthony Hospital* and *University of Minnesota Medical Center Fairview*. This suggests that these hospitals experience high patient demand for critical care services, making them key centers for ICU and SICU treatment.

```
    -- Query 2: Identifying the top 10 hospitals with the highest number of census beds in ICUs and SICUs.
    -- This query provides insight into the actual number of beds in use at these hospitals.
103 • SELECT
104       MIN(f.ims_org_id) AS org_id,  -- Retrieves the minimum organization ID
105       b.hospital_name,  -- Retrieves the hospital name
106       SUM(f.census_beds) AS Total_Census_Beds,  -- Sums the total census beds (occupied beds)
107       SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.license_beds ELSE 0 END) AS LicBeds_ICU, -- Filters licensed ICU beds
108       SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.license_beds ELSE 0 END) AS LicBeds_SICU, -- Filters licensed SICU beds
109       SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.census_beds ELSE 0 END) AS CensBeds_ICU, -- Filters census ICU beds
110       SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.census_beds ELSE 0 END) AS CensBeds_SICU, -- Filters census SICU beds
111       SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_ICU, -- Filters staffed ICU beds
112       SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_SICU -- Filters staffed SICU beds
113   FROM  bed_fact f
114   LEFT JOIN
115       business b ON b.ims_org_id = f.ims_org_id  -- Joins the business table to fetch hospital names
116   LEFT JOIN
117       bed_type t ON t.bed_id = f.bed_id  -- Joins the bed type table
118   WHERE
119       t.bed_desc = 'ICU' OR t.bed_desc = 'SICU' -- Filters only ICU and SICU beds
120   GROUP BY
121       b.hospital_name  -- Groups by hospital name for aggregation
122   ORDER BY
123       Total_Census_Beds DESC  -- Sorts in descending order based on total census beds
124   LIMIT 10; -- Retrieves the top 10 hospitals
```



Result Grid | Filter Rows: Search | Export: | Fetch rows:

org_id	hospital_name	Total_Census_Be...	LicBeds_ICU	LicBeds_SICU	CensBeds_ICU	CensBeds_SICU	StaffBeds_ICU	StaffBeds_SICU
INS00001097	Saint Marys Hospital	205	360	19	193	12	284	16
INS00000728	Saint Anthony Hospital	190	337	0	190	0	288	0
INS00001654	University of Minnesota Medical Center Fairview	167	144	55	121	46	144	55
INS00006009	Shands Hospital at the University of Florida	167	8	159	8	159	8	159
INS00005607	Dallas County Hospital Association	145	195	0	145	0	149	0
INS00005273	South Broward Hospital District	145	207	0	145	0	168	0
INS00006531	Mercy Medical Center Saint Louis	142	163	0	142	0	163	0
INS00005189	Los Angeles County University of Southern Calif...	139	160	0	139	0	151	0
INS00006426	The Methodist Hospital	138	170	0	138	0	138	0
INS00006508	University of Maryland Medical Center	127	104	116	60	67	81	90

Question 3: Identifying the Top 10 Hospitals with the Most Staffed ICU and SICU Beds

The third query determined the hospitals with the highest number of **staffed beds**, which represents the beds that have adequate physician and nursing staff to accommodate patients. This is a critical factor in hospital operational efficiency, as even if beds are available, they cannot be used unless there is sufficient staff to manage them.

To extract this data, I followed the same structured approach but focused on **staffed beds**, aggregating them based on ICU and SICU designations. The results were then sorted in descending order to highlight the hospitals with the most staffed beds.

The results show that *Saint Mary's Hospital* and *Saint Anthony Hospital* remain at the top, with significant staffed ICU and SICU beds, making them among the best-equipped facilities for handling critical care patients. However, some hospitals, despite having many licensed or census beds, have fewer staffed beds, which could indicate staffing shortages affecting operational capacity.

```sql
-- Query 3: Identifying the top 10 hospitals with the highest number of staffed beds in ICUs and SICUs.
-- This query determines the hospitals with the most operational beds based on available staffing.
SELECT
    MIN(f.ims_org_id) AS org_id,  -- Retrieves the minimum organization ID
    b.hospital_name,  -- Retrieves the hospital name
    SUM(f.staffed_beds) AS Total_Staffed_Beds,  -- Sums the total staffed beds
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.license_beds ELSE 0 END) AS LicBeds_ICU, -- Filters licensed ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.license_beds ELSE 0 END) AS LicBeds_SICU, -- Filters licensed SICU beds
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.census_beds ELSE 0 END) AS CensBeds_ICU, -- Filters census ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.census_beds ELSE 0 END) AS CensBeds_SICU, -- Filters census SICU beds
    SUM(CASE WHEN t.bed_desc = 'ICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_ICU, -- Filters staffed ICU beds
    SUM(CASE WHEN t.bed_desc = 'SICU' THEN f.staffed_beds ELSE 0 END) AS StaffBeds_SICU -- Filters staffed SICU beds
FROM bed_fact f
LEFT JOIN
    business b ON b.ims_org_id = f.ims_org_id  -- Joins the business table
LEFT JOIN
    bed_type t ON t.bed_id = f.bed_id  -- Joins the bed type table
WHERE
    t.bed_desc = 'ICU' OR t.bed_desc = 'SICU' -- Filters only ICU and SICU beds
GROUP BY
    b.hospital_name  -- Groups by hospital name
ORDER BY
    Total_Staffed_Beds DESC  -- Sorts results based on total staffed beds
LIMIT 10; -- Retrieves the top 10 hospitals
```

Result Grid

org_id	hospital_name	Total_Staffed_Be...	LicBeds_ICU	LicBeds_SICU	CensBeds_ICU	CensBeds_SICU	StaffBeds_ICU	StaffBeds_SICU
INS00001097	Saint Marys Hospital	300	360	19	193	12	284	16
INS00000728	Saint Anthony Hospital	288	337	0	190	0	288	0
INS00006486	Vidant Medical Center	203	204	0	123	0	203	0
INS00004615	Rady Childrens Hospital and Health Center	200	200	0	75	0	200	0
INS00001654	University of Minnesota Medical Center Fairview	199	144	55	121	46	144	55
INS00001157	Good Samaritan Hospital	192	220	12	112	7	181	11
INS00001685	Grady Memorial Hospital	171	124	56	64	32	115	56
INS00006508	University of Maryland Medical Center	171	104	116	60	67	81	90
INS00001137	Saint Mary Medical Center	170	170	0	105	0	170	0
INS00001639	Emory University Hospital	169	169	0	120	0	169	0

Conclusion

The project successfully met the goals of creating a structured star schema, importing data into MySQL, and performing business queries to extract meaningful insights. The star schema effectively streamlined data organization, ensuring that hospital and bed data could be efficiently linked and analyzed. The SQL queries provided clear answers to key business questions, identifying top hospitals with the highest licensed, census, and staffed ICU and SICU beds.

One of the key advantages of using MySQL Workbench was its intuitive interface for database management, making it easy to import large datasets, execute queries, and validate results. Additionally, ERDPlus proved to be a useful tool for designing and visualizing the star schema before implementation. However, a notable limitation was the time-consuming process of cleaning and validating data, particularly in checking for duplicates and ensuring relational integrity between tables.

If I were to redo this project, I would focus on optimizing the import process by pre-cleaning the dataset using Python or Excel before loading it into MySQL. Additionally, implementing stored procedures or views for repeated queries would improve efficiency and organization.

Overall, this project provided valuable hands-on experience in database design, SQL query writing, and data analysis, reinforcing the importance of structured data management in real-world analytics applications.