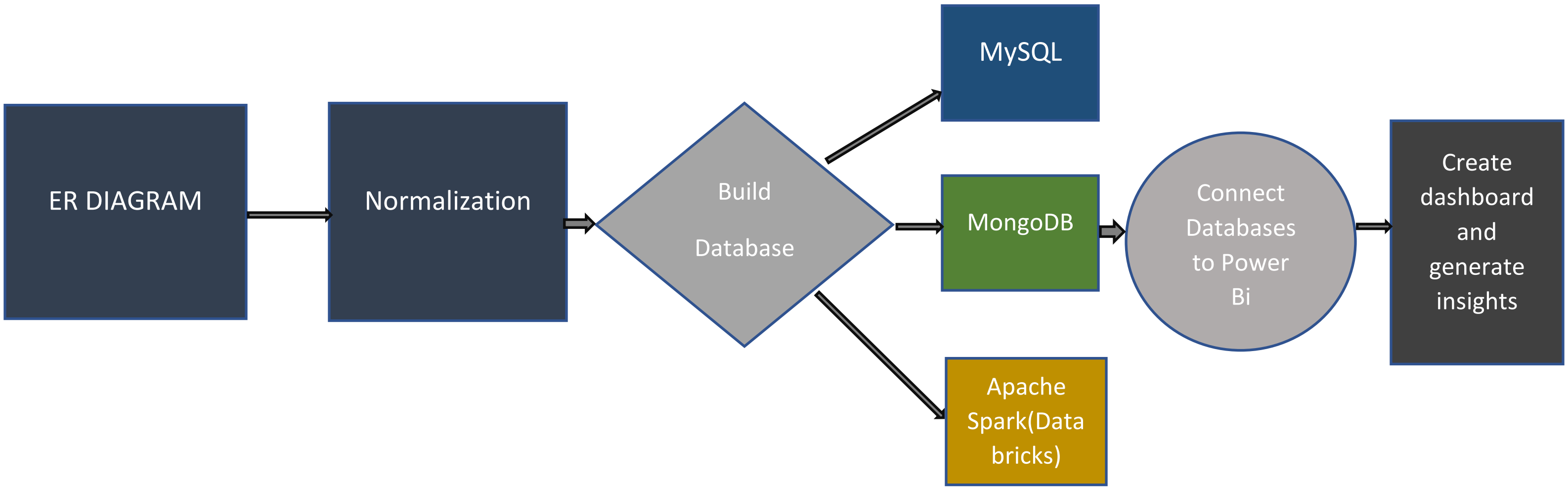
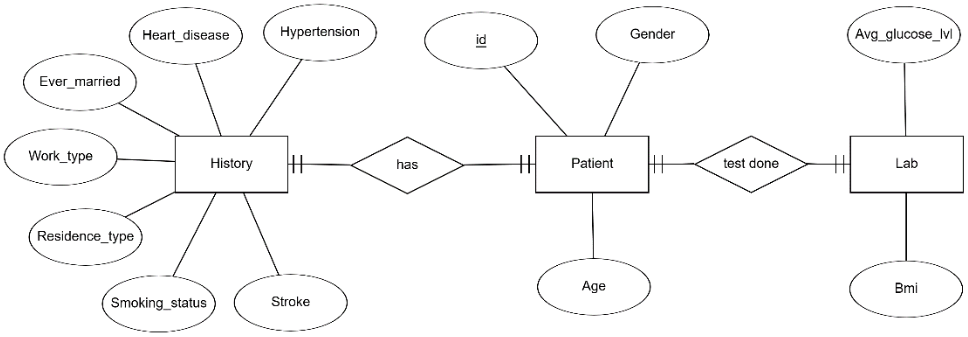
SYTEM ARCHITECTURE


ER DIAGRAM
Normalization
Build
Database
MySQL
MongoDB
Apache Spark(Data bricks)
Connect Databases to Power Bi
Create dashboard and generate insights

Conceptual Modeling with ER Diagram

The ER diagram would help us understand the entities, their attributes, and the relationships between them.



Summary

In the ER diagram:

- Patient table contains personal attributes of an individual.
- Lifestyle table contains lifestyle-related attributes.
- Health table contains Health-related attributes.
- id is the primary key (PK) in the Patient table and a foreign key (FK) in the History table, and Lab table, linking these tables together.
- A Person entity has a one-to-one relationship with a history entity and a hospital entity, meaning that each person has one History and one Lab test results, and vice versa.

Normalization

I normalized the data to **1NF**, **2NF**, and **3NF**.

First Normal Form(1NF):

- To achieve the first normal form, we need to make sure that each cell contains only a single value, each column has a unique name, and there are no duplicate rows. We also need to identify a primary key for the table. In this case, we can use the `id` column as the primary key since it uniquely identifies each row.
- No multivalued attributes are present.
- Based on the dataset, we already have a tabular representation, which satisfies 1NF.

Second Normal Form (2NF):

- To achieve the second normal form, we need to make sure that each non-key attribute is fully dependent on the primary key. This means that there should be no partial dependencies, where a column depends on only part of the primary key. In this case, since the primary key is a single column (id), there are no partial dependencies
- Therefore, the table already satisfies 2NF.

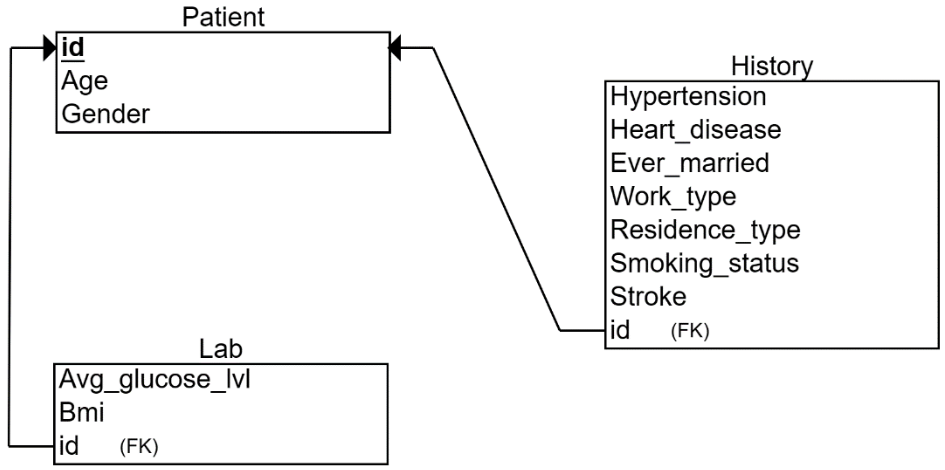
Third Normal Form (3NF):

- To achieve the third normal form, we need to make sure that all non-key attributes are independent of each other. This means there should be no transitive dependencies, where a column depends on another column not the primary key.
- In summary, the existing table is already in 1NF and 2NF and 3NF

Relational Schema

I translated the conceptual model into a relational schema, which includes defining tables, columns, primary keys, foreign keys, and other constraints. This step involves making decisions about how the data will be stored in a relational database.

The Relational Schema for the ER diagram is below:



Database Development using MySQL, MongoDB, and Apache Spark

MySQL

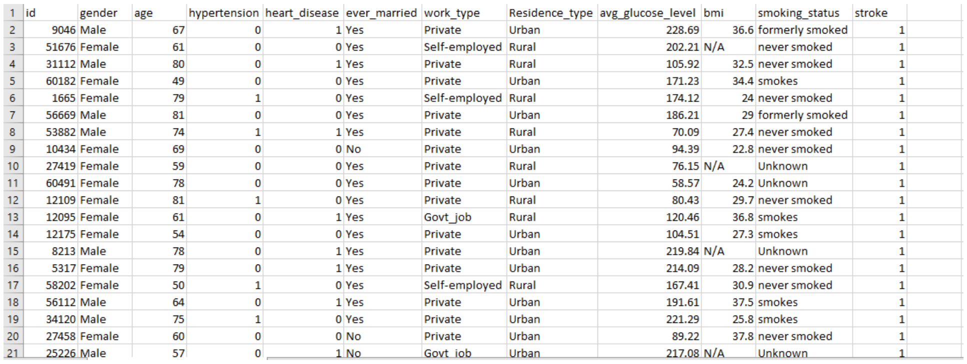
Objective: Develop a database using MySQL

Database Implementation

Using the Relational Schema. I partitioned the dataset into three distinct entities: Patient, History, and Lab. I employed MySQL to create three tables corresponding to these entities. Firstly, I created a new database ‘health_stroke_data. I populated these tables by importing the relevant attributes into them from the dataset in .csv. Keeping them in separate tables would enhance the clarity and readability of the database design.

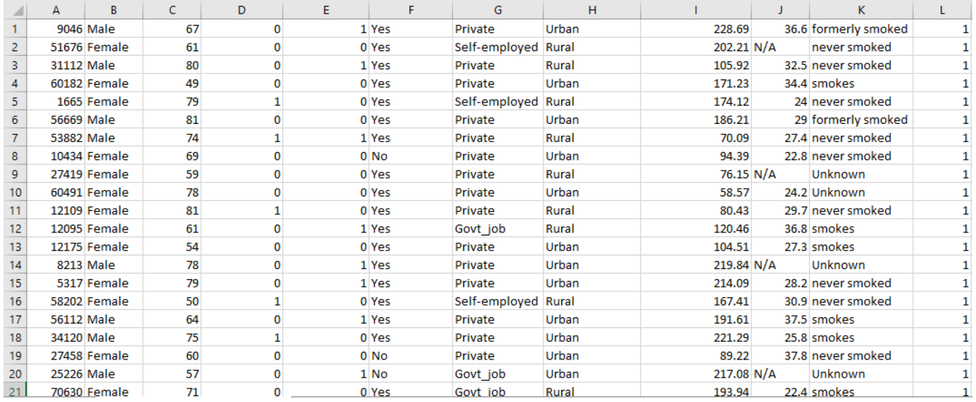
Create Database

- Get Csv file



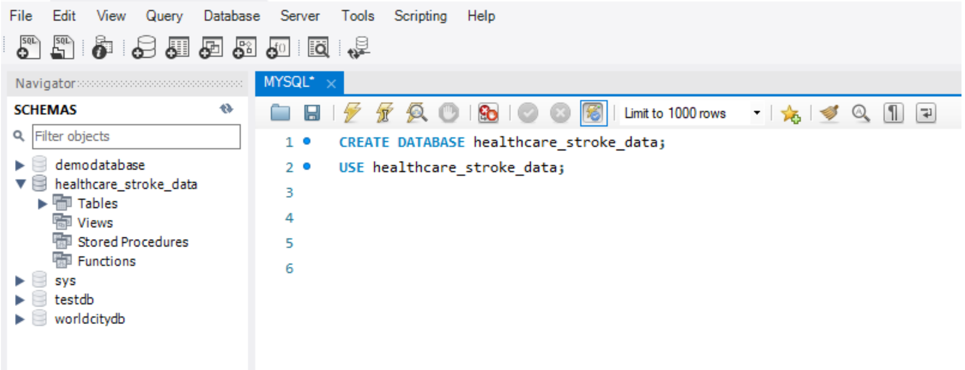
1	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
2	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
3	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
4	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
5	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
6	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
7	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
8	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
9	10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
10	27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
11	60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
12	12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
13	12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
14	12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
15	8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
16	5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1
17	58202	Female	50	1	0	Yes	Self-employed	Rural	167.41	30.9	never smoked	1
18	56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
19	34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
20	27458	Female	60	0	0	No	Private	Urban	89.22	37.8	never smoked	1
21	25226	Male	57	0	1	No	Govt_job	Urban	217.08	N/A	Unknown	1

- ➢ Create a new version of the .csv file without fieldnames (only data rows) By deleting the filed names



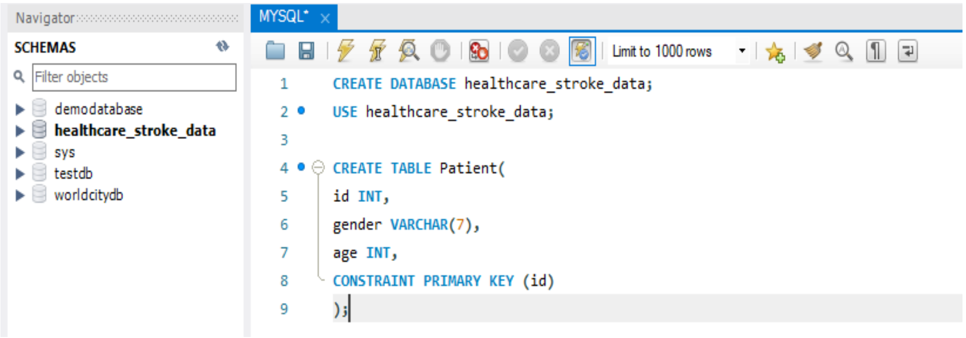
	A	B	C	D	E	F	G	H	I	J	K	L
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
2	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked	1
3	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
4	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
5	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked	1
6	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
7	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
8	10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
9	27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
10	60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
11	12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
12	12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
13	12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
14	8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
15	5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1
16	58202	Female	50	1	0	Yes	Self-employed	Rural	167.41	30.9	never smoked	1
17	56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
18	34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1
19	27458	Female	60	0	0	No	Private	Urban	89.22	37.8	never smoked	1
20	25226	Male	57	0	1	No	Govt_job	Urban	217.08	N/A	Unknown	1
21	70630	Female	71	0	0	Yes	Govt_job	Rural	193.94	22.4	smokes	1

- ➢ Create database and switch to new database



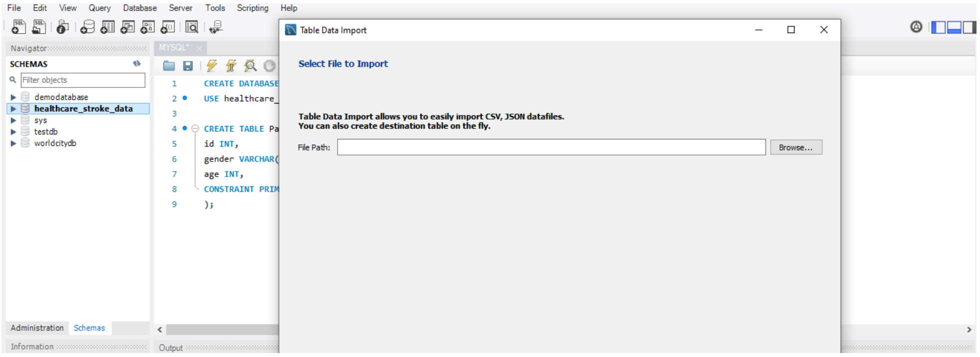
```
CREATE DATABASE healthcare_stroke_data;
USE healthcare_stroke_data;
```

- ➢ Create table “Patient”



```
CREATE DATABASE healthcare_stroke_data;
USE healthcare_stroke_data;

CREATE TABLE Patient(
id INT,
gender VARCHAR(7),
age INT,
CONSTRAINT PRIMARY KEY (id)
);
```

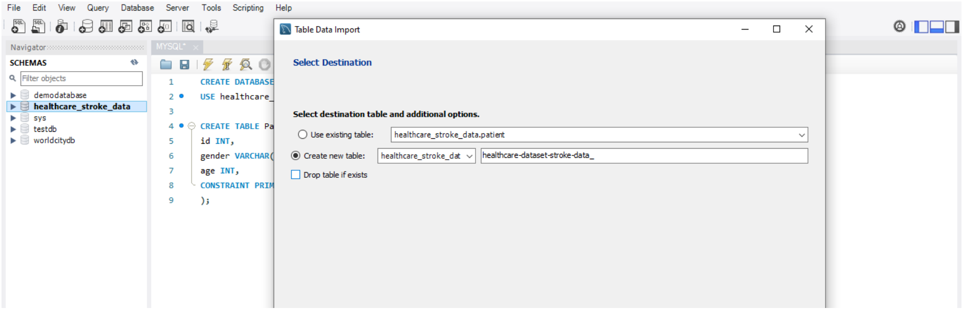
- Imported Csv file to the table



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:



```
CREATE DATABASE
USE healthcare_

CREATE TABLE Pa
id INT,
gender VARCHAR(
age INT,
CONSTRAINT PRIM
);
```

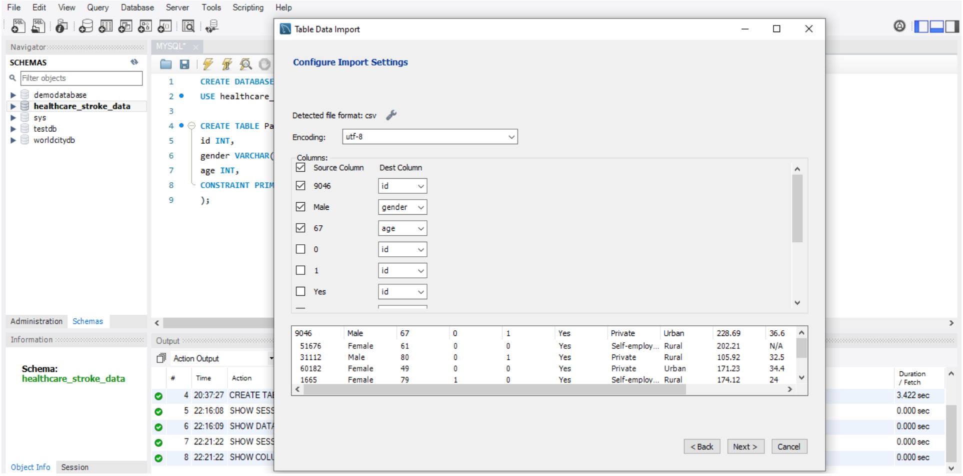
Select Destination

Select destination table and additional options.

○ Use existing table: healthcare_stroke_data.patient

● Create new table: healthcare_stroke_dat healthcare-dataset-stroke-data_

□ Drop table if exists



```
CREATE DATABASE
USE healthcare_

CREATE TABLE Pa
id INT,
gender VARCHAR(
age INT,
CONSTRAINT PRIM
);
```

Schema: healthcare_stroke_data

```
   #  Time      Action         Duration / Fetch
   4  20:37:27  CREATE TA      3.422 sec
   5  22:16:08  SHOW SESS      0.000 sec
   6  22:16:09  SHOW DATA      0.000 sec
   7  22:21:22  SHOW SESS      0.000 sec
   8  22:21:22  SHOW COLU      0.000 sec
```

Configure Import Settings

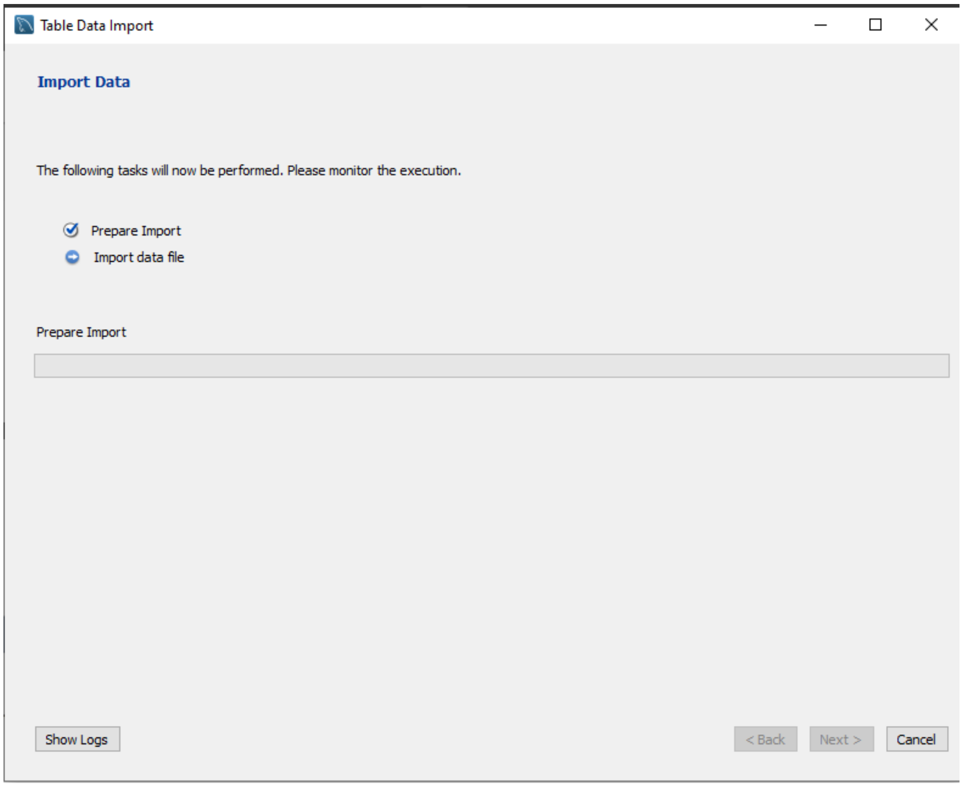
Detected file format: csv

Encoding: utf-8

Columns:

☑ Source Column	Dest Column
☑ 9046	id
☑ Male	gender
☑ 67	age
□ 0	id
□ 1	id
□ Yes	id

9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6
51676	Female	61	0	0	Yes	Self-employ...	Rural	202.21	N/A
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4
1665	Female	79	1	0	Yes	Self-employ...	Rural	174.12	24



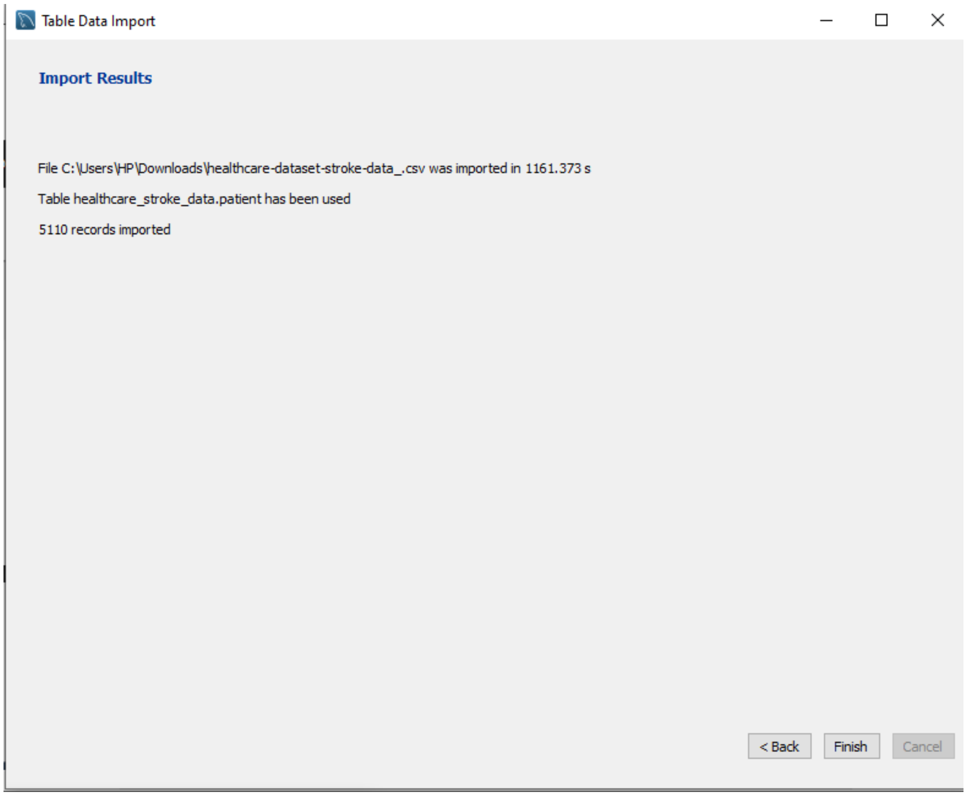
Import Data

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

✓ Prepare Import

● Import data file

Prepare Import



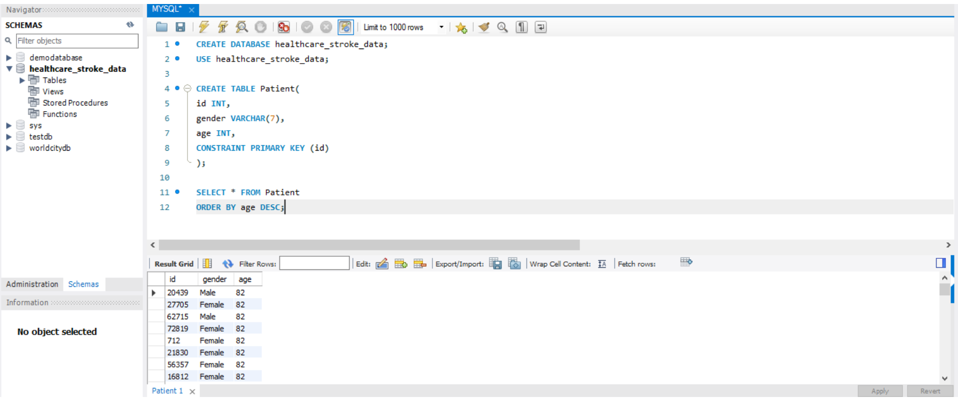
Import Results

File C:\Users\HP\Downloads\healthcare-dataset-stroke-data_.csv was imported in 1161.373 s

Table healthcare_stroke_data.patient has been used

5110 records imported

- Queried the data

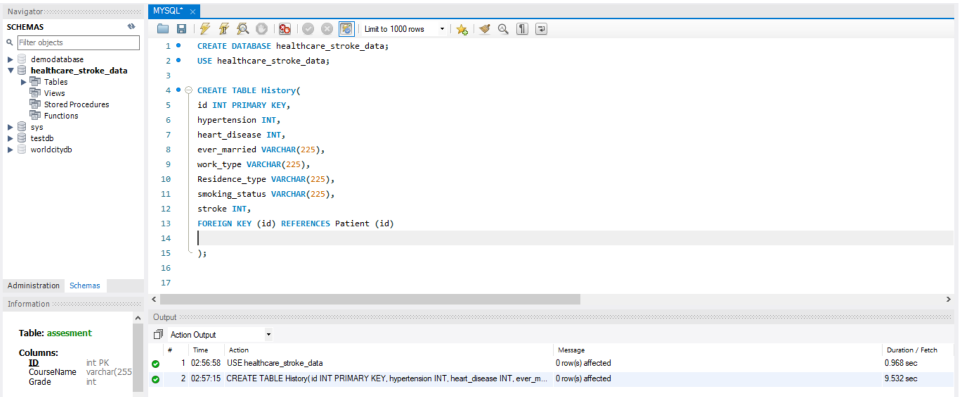


```
CREATE DATABASE healthcare_stroke_data;
USE healthcare_stroke_data;

CREATE TABLE Patient(
id INT,
gender VARCHAR(7),
age INT,
CONSTRAINT PRIMARY KEY (id)
);

SELECT * FROM Patient
ORDER BY age DESC;
```

id	gender	age
20439	Male	82
27705	Female	82
62715	Male	82
72819	Female	82
712	Female	82
21830	Female	82
56357	Female	82
16812	Female	82

I repeated the process for other tables like "History" and "Lab" Entities

- **For History**



```
CREATE DATABASE healthcare_stroke_data;
USE healthcare_stroke_data;

CREATE TABLE History(
id INT PRIMARY KEY,
hypertension INT,
heart_disease INT,
ever_married VARCHAR(225),
work_type VARCHAR(225),
Residence_type VARCHAR(225),
smoking_status VARCHAR(225),
stroke INT,
FOREIGN KEY (id) REFERENCES Patient (id)

);
```

Table: assesment

Columns:
ID int PK
CourseName varchar(255
Grade int

```
#  Time      Action                                                                            Message             Duration / Fetch
1  02:56:58  USE healthcare_stroke_data                                                        0 row(s) affected   0.968 sec
2  02:57:15  CREATE TABLE History( id INT PRIMARY KEY, hypertension INT, heart_disease INT, ever_m...   0 row(s) affected   9.532 sec
```

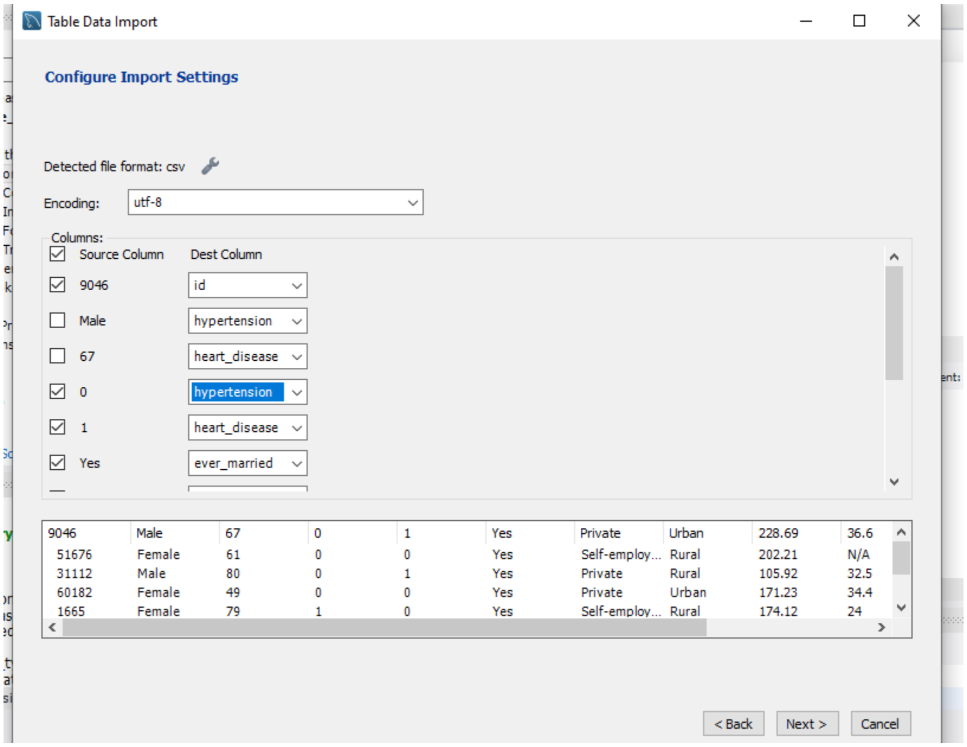


Table Data Import

Configure Import Settings

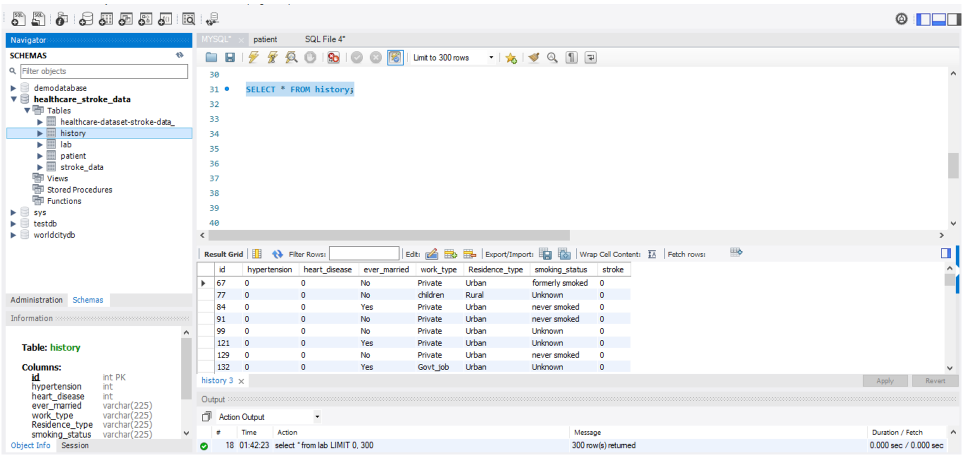
Detected file format: csv

Encoding: utf-8

Columns:

	Source Column	Dest Column
☑	9046	id
□	Male	hypertension
□	67	heart_disease
☑	0	hypertension
☑	1	heart_disease
☑	Yes	ever_married

9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6
51676	Female	61	0	0	Yes	Self-employ...	Rural	202.21	N/A
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4
1665	Female	79	1	0	Yes	Self-employ...	Rural	174.12	24



```
SELECT * FROM history;
```

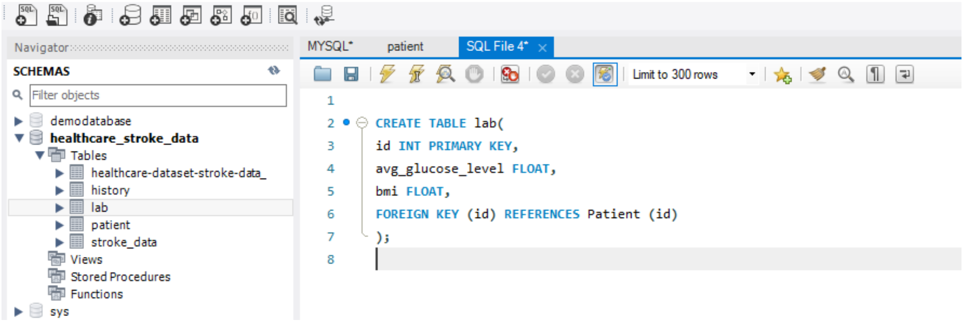
id	hypertension	heart_disease	ever_married	work_type	Residence_type	smoking_status	stroke
67	0	0	No	Private	Urban	formerly smoked	0
77	0	0	No	children	Rural	Unknown	0
84	0	0	Yes	Private	Urban	never smoked	0
91	0	0	No	Private	Urban	never smoked	0
99	0	0	No	Private	Urban	Unknown	0
121	0	0	Yes	Private	Urban	Unknown	0
129	0	0	No	Private	Urban	never smoked	0
132	0	0	Yes	Govt_job	Urban	Unknown	0

Table: history

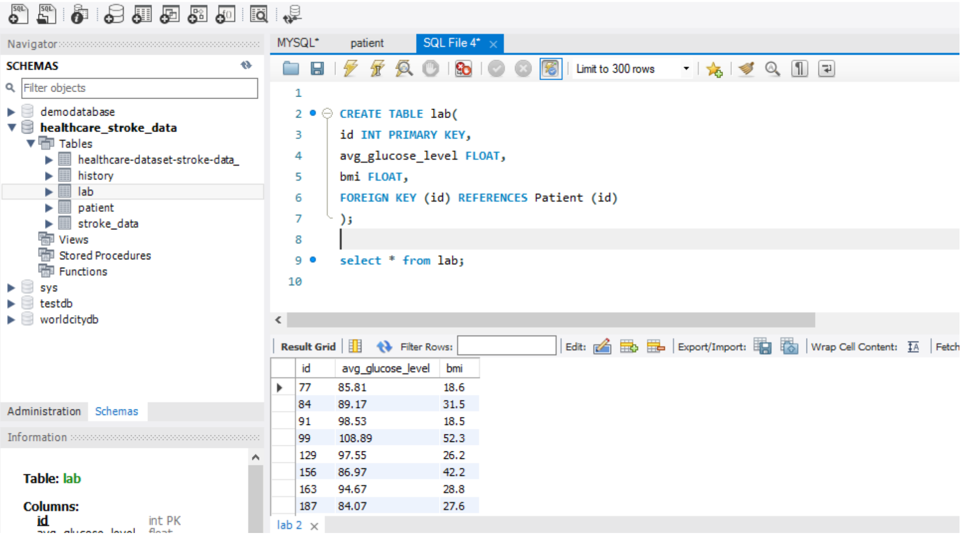
Columns:
id int PK
hypertension int
heart_disease int
ever_married varchar(225)
work_type varchar(225)
Residence_type varchar(225)
smoking_status varchar(225)

```
#   Time      Action                          Message              Duration / Fetch
18  01:42:23  select * from lab LIMIT 0, 300  300 row(s) returned  0.000 sec / 0.000 sec
```

- **For Lab**



```sql
CREATE TABLE lab(
id INT PRIMARY KEY,
avg_glucose_level FLOAT,
bmi FLOAT,
FOREIGN KEY (id) REFERENCES Patient (id)
);
```



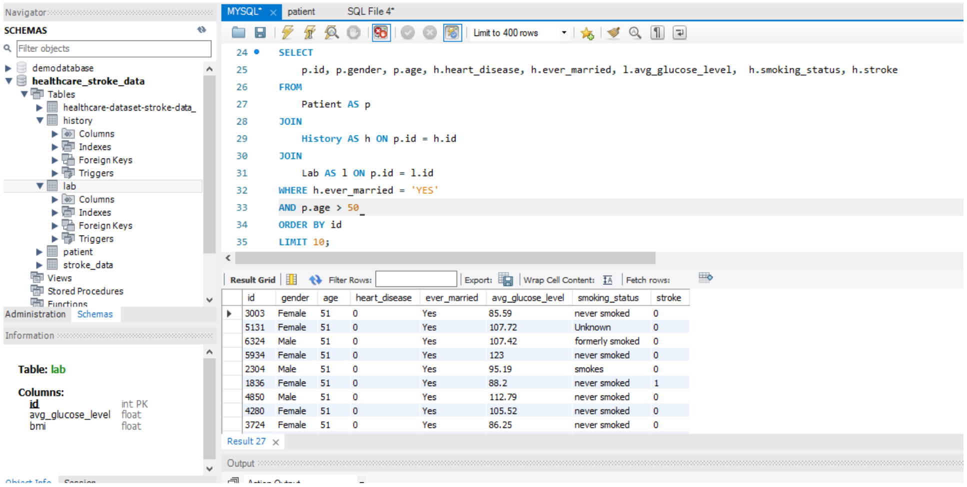
```sql
CREATE TABLE lab(
id INT PRIMARY KEY,
avg_glucose_level FLOAT,
bmi FLOAT,
FOREIGN KEY (id) REFERENCES Patient (id)
);

select * from lab;
```

id	avg_glucose_level	bmi
77	85.81	18.6
84	89.17	31.5
91	98.53	18.5
99	108.89	52.3
129	97.55	26.2
156	86.97	42.2
163	94.67	28.8
187	84.07	27.6

The three entities have been successfully portioned into three tables: Patient, History and Lab.

Testing and Optimization

Query optimization is a crucial aspect of database management aimed at improving the performance and efficiency of SQL queries.



```sql
SELECT
    p.id, p.gender, p.age, h.heart_disease, h.ever_married, l.avg_glucose_level,  h.smoking_status, h.stroke
FROM
    Patient AS p
JOIN
    History AS h ON p.id = h.id
JOIN
    Lab AS l ON p.id = l.id
WHERE h.ever_married = 'YES'
AND p.age > 50
ORDER BY id
LIMIT 10;
```

id	gender	age	heart_disease	ever_married	avg_glucose_level	smoking_status	stroke
3003	Female	51	0	Yes	85.59	never smoked	0
5131	Female	51	0	Yes	107.72	Unknown	0
6324	Male	51	0	Yes	107.42	formerly smoked	0
5934	Female	51	0	Yes	123	never smoked	0
2304	Male	51	0	Yes	95.19	smokes	0
1836	Female	51	0	Yes	88.2	never smoked	1
4850	Male	51	0	Yes	112.79	never smoked	0
4280	Female	51	0	Yes	105.52	never smoked	0
3724	Female	51	0	Yes	86.25	never smoked	0

In this query:

- Not all columns are selected: Instead of selecting all columns using SELECT *, explicitly specify only the required columns in the SELECT statement. This reduces the amount of data transferred and can improve query performance.
- JOIN conditions are efficient and utilize indexes
- WHERE clauses filter data effectively
- The result set is limited to 10 rows: Use the LIMIT clause to restrict the number of rows returned by a query, especially for queries that return large result sets.

By following these optimization techniques and continuously fine-tuning queries based on performance analysis, we can significantly improve the efficiency and responsiveness of our database.

NoSQL

Objective: Develop a database using MongoDB

To develop a MongoDB database. The following steps will be taken.

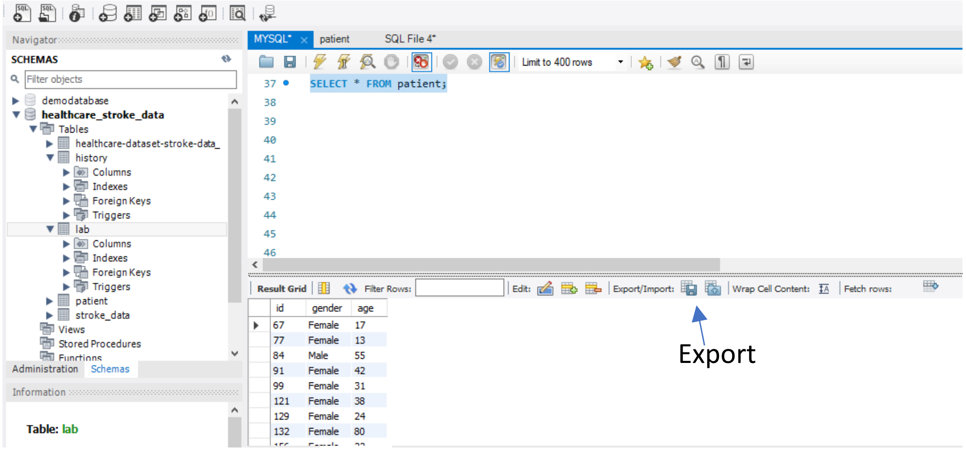
- **Collections**

From the ER diagram, the dataset structure and relationships was known. With this I could define the collections which are: Patients, History and Lab.

Create Database

I migrated the dataset from MySQL to MongoDB

- Imported data from MySQL Workbench

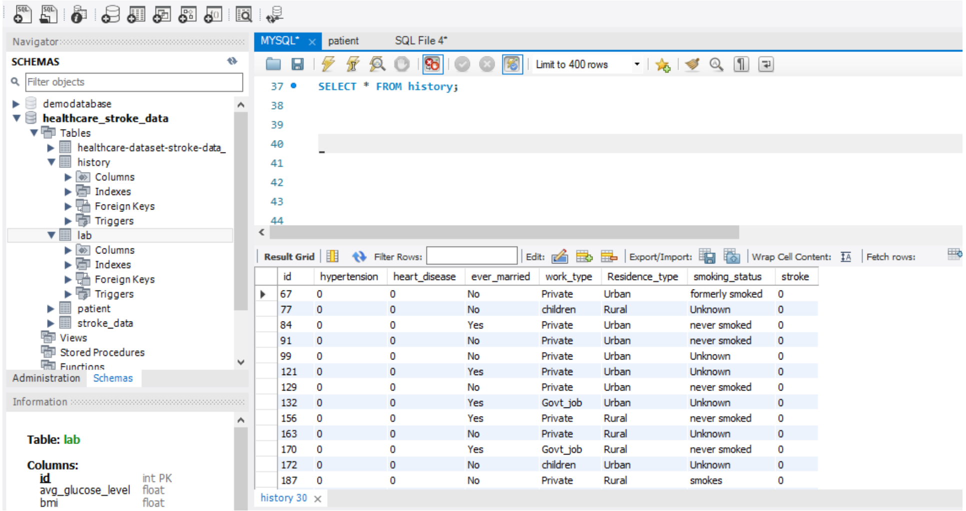


```
SELECT * FROM patient;
```

id	gender	age
67	Female	17
77	Female	13
84	Male	55
91	Female	42
99	Female	31
121	Female	38
129	Female	24
132	Female	80

Export

- Did the same for the other two entities



```
SELECT * FROM history;
```

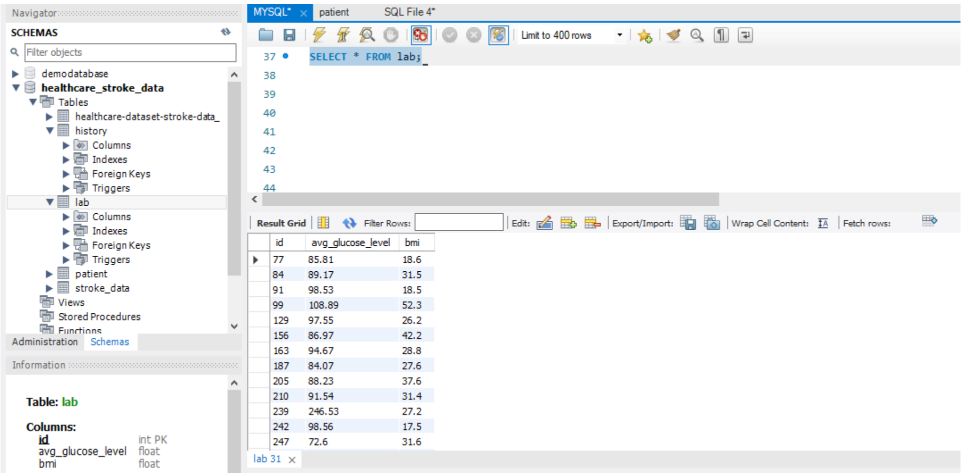
id	hypertension	heart_disease	ever_married	work_type	Residence_type	smoking_status	stroke
67	0	0	No	Private	Urban	formerly smoked	0
77	0	0	No	children	Rural	Unknown	0
84	0	0	Yes	Private	Urban	never smoked	0
91	0	0	No	Private	Urban	never smoked	0
99	0	0	No	Private	Urban	Unknown	0
121	0	0	Yes	Private	Urban	Unknown	0
129	0	0	No	Private	Urban	never smoked	0
132	0	0	Yes	Govt_job	Urban	Unknown	0
156	0	0	Yes	Private	Rural	never smoked	0
163	0	0	No	Private	Rural	Unknown	0
170	0	0	Yes	Govt_job	Rural	never smoked	0
172	0	0	No	children	Urban	Unknown	0
187	0	0	No	Private	Rural	smokes	0

Table: lab

Columns:

id	int PK
avg_glucose_level	float
bmi	float

- ➢ Opened MondoDB Compass and created a database named healthcare_stroke_data with three collections (Patient, History and Lab)



```
SELECT * FROM lab;
```

id	avg_glucose_level	bmi
77	85.81	18.6
84	89.17	31.5
91	98.53	18.5
99	108.89	52.3
129	97.55	26.2
156	86.97	42.2
163	94.67	28.8
187	84.07	27.6
205	88.23	37.6
210	91.54	31.4
239	246.53	27.2
242	98.56	17.5
247	72.6	31.6

Table: lab

Columns:

id	int PK
avg_glucose_level	float
bmi	float

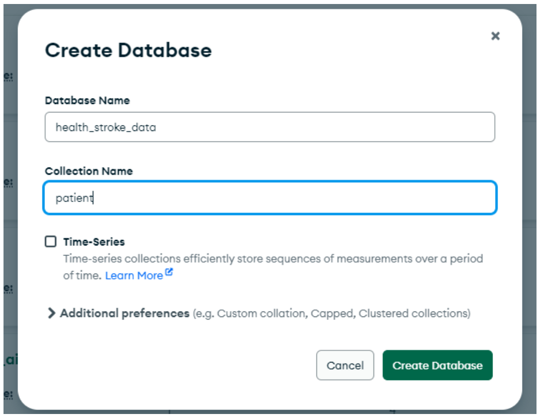
history.csv

patient.csv

lab.csv

The data was imported as .csv successfully.

- ➢ Created a new database “health_stroke_data ”and three collections: Patient, History and Lab.



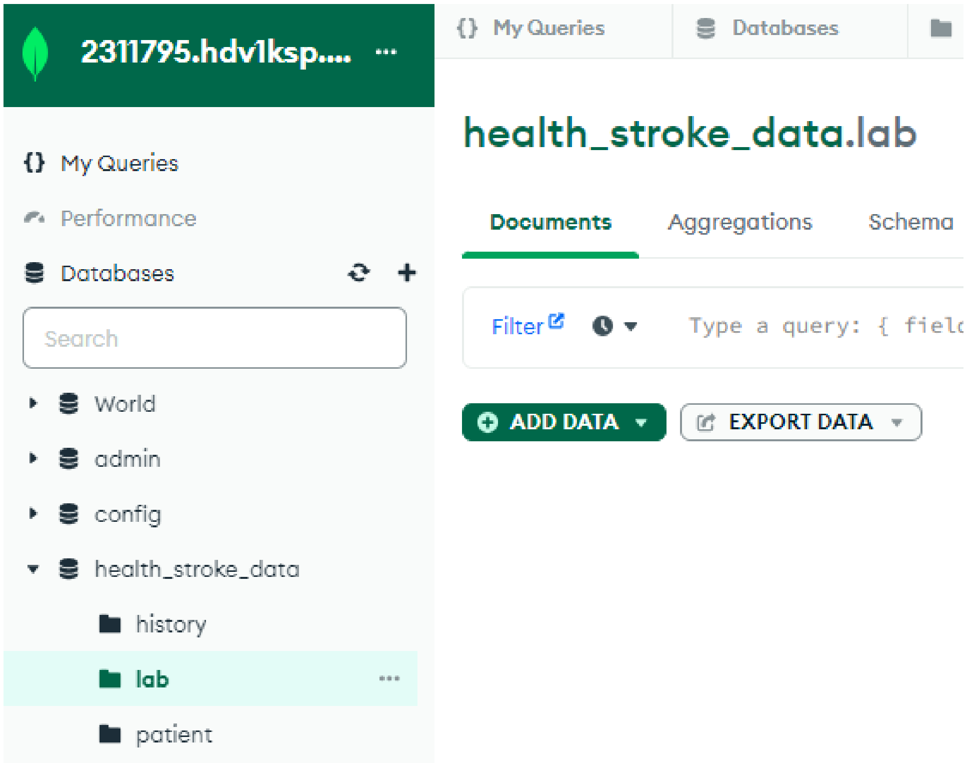
Create Database

Database Name: health_stroke_data

Collection Name: patient

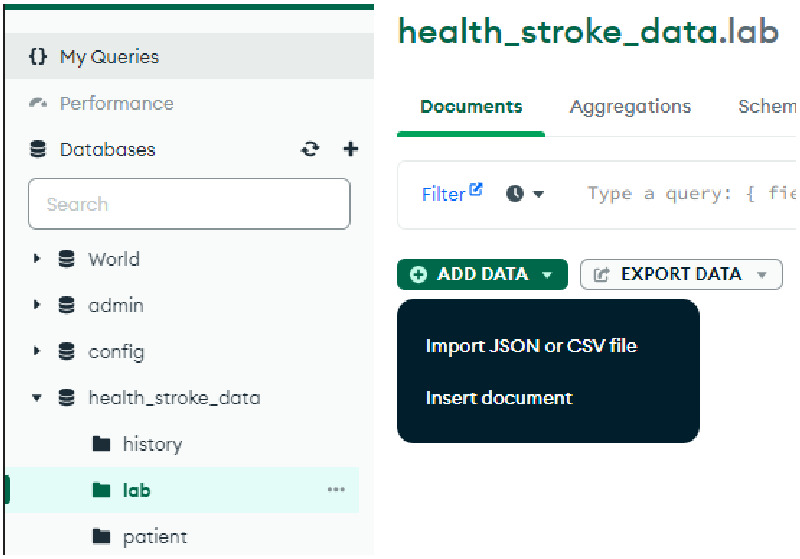
□ Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. Learn More

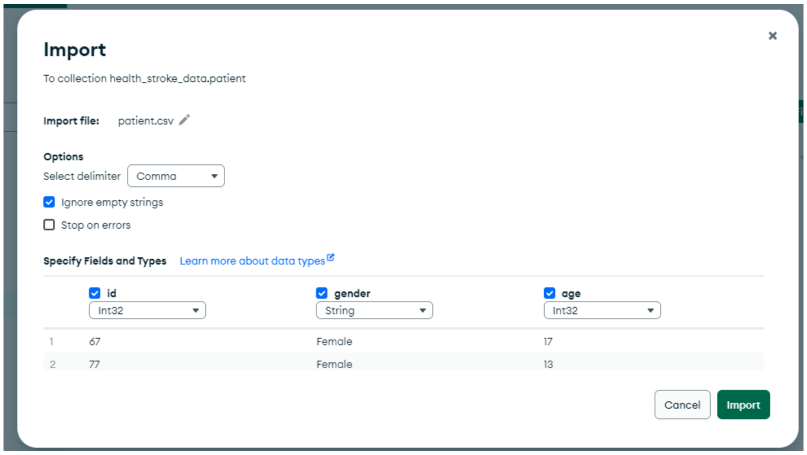
×
> Additional preferences (e.g. Custom collation, Capped, Clustered collections)
Cancel
Create Database



- ➢ Inserted Documents or Add Data

Added data to the collections by importing csv file from fig





Import

To collection health_stroke_data.patient

Import file: patient.csv

Options

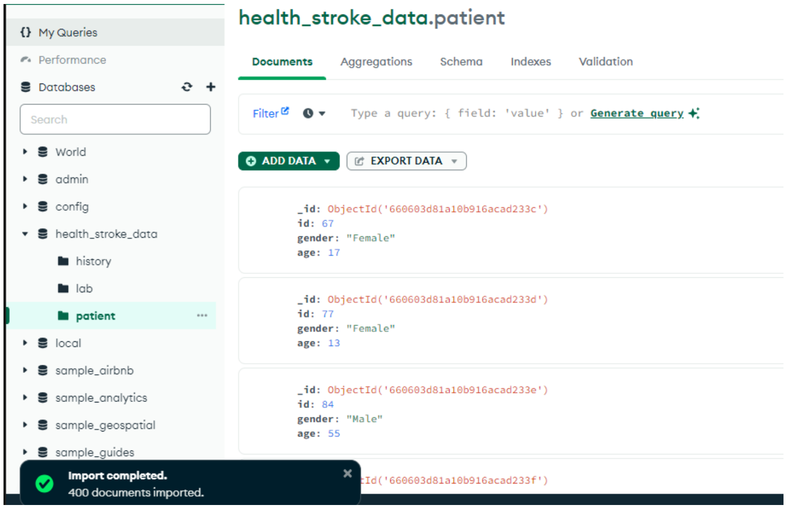
Select delimiter: Comma

☑ Ignore empty strings

□ Stop on errors

Specify Fields and Types Learn more about data types

	☑ id Int32	☑ gender String	☑ age Int32
1	67	Female	17
2	77	Female	13



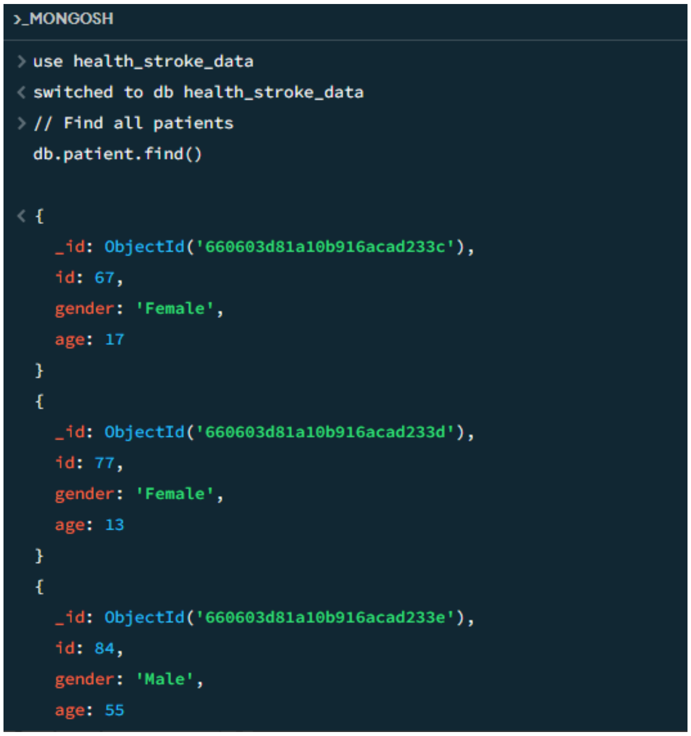
```
_id: ObjectId('660603d81a10b916acad233c')
id: 67
gender: "Female"
age: 17

_id: ObjectId('660603d81a10b916acad233d')
id: 77
gender: "Female"
age: 13

_id: ObjectId('660603d81a10b916acad233e')
id: 84
gender: "Male"
age: 55

tId('660603d81a10b916acad233f')
```

Import completed.
400 documents imported.

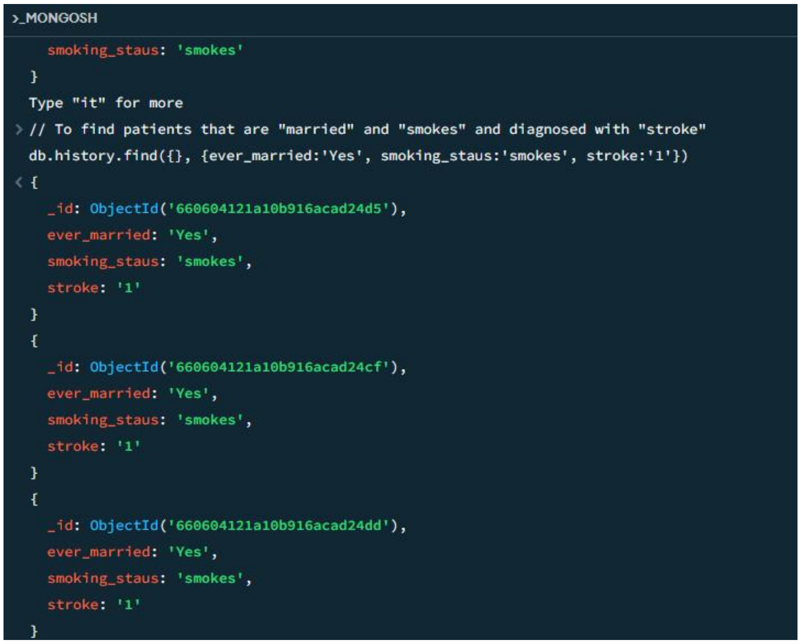
Data was successfully imported to the patient collection. I followed the same steps for the other two collections.

- ➢ Queried the Data

I used MongoDB's query language to retrieve data from the database.



```
>_MONGOSH
> use health_stroke_data
< switched to db health_stroke_data
> // Find all patients
  db.patient.find()

< {
    _id: ObjectId('660603d81a10b916acad233c'),
    id: 67,
    gender: 'Female',
    age: 17
  }
  {
    _id: ObjectId('660603d81a10b916acad233d'),
    id: 77,
    gender: 'Female',
    age: 13
  }
  {
    _id: ObjectId('660603d81a10b916acad233e'),
    id: 84,
    gender: 'Male',
    age: 55
```



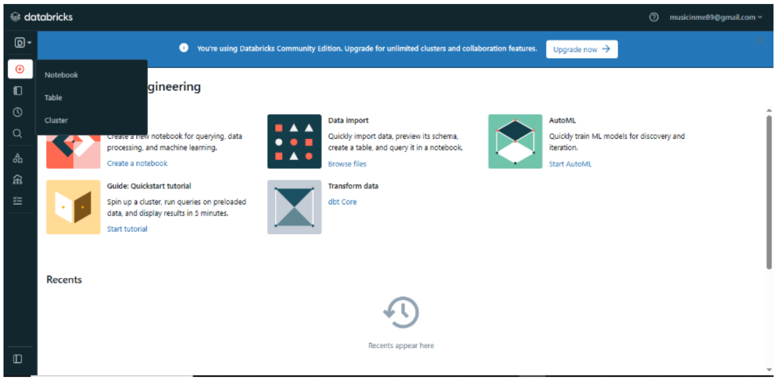
```
>_MONGOSH
    smoking_staus: 'smokes'
  }
  Type "it" for more
> // To find patients that are "married" and "smokes" and diagnosed with "stroke"
  db.history.find({}, {ever_married:'Yes', smoking_staus:'smokes', stroke:'1'})
< {
    _id: ObjectId('660604121a10b916acad24d5'),
    ever_married: 'Yes',
    smoking_staus: 'smokes',
    stroke: '1'
  }
  {
    _id: ObjectId('660604121a10b916acad24cf'),
    ever_married: 'Yes',
    smoking_staus: 'smokes',
    stroke: '1'
  }
  {
    _id: ObjectId('660604121a10b916acad24dd'),
    ever_married: 'Yes',
    smoking_staus: 'smokes',
    stroke: '1'
  }
```

Apache Spark

Objective: Develop a database using Data Bricks

To develop a MongoDB database. The following steps will be taken.

- ➢ Uploaded the csv file



You're using Databricks Community Edition. Upgrade for unlimited clusters and collaboration features.

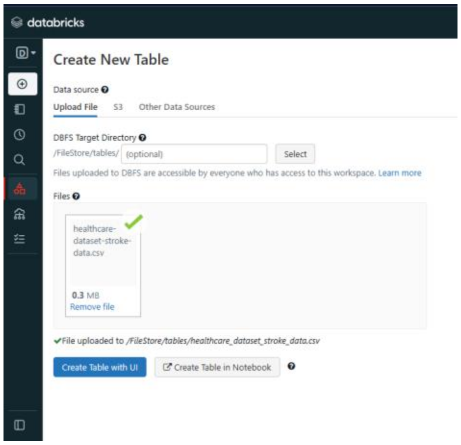
Create a new notebook for querying, data processing, and machine learning.

Data import
Quickly import data, preview its schema, create a table, and query it in a notebook.

AutoML
Quickly train ML models for discovery and iteration.

Guide: Quickstart tutorial
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Transform data



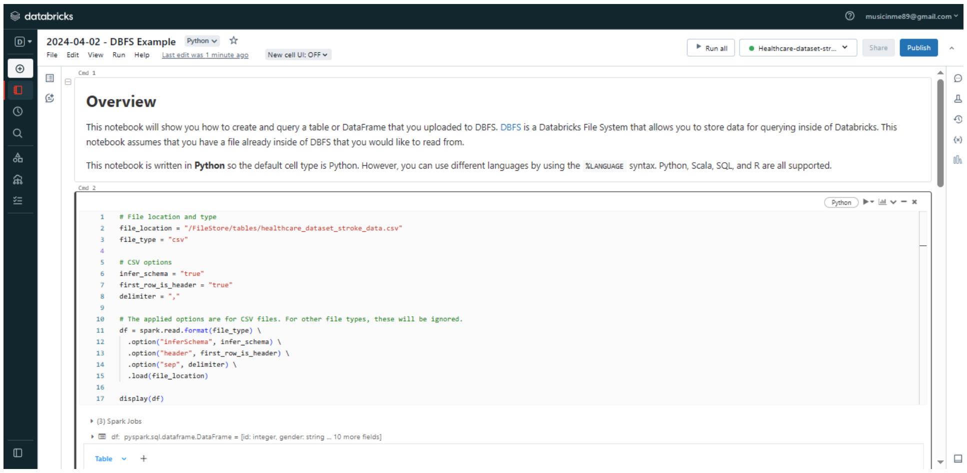
Create New Table

Data source

DBFS Target Directory
/FileStore/tables/ (optional)
Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files
healthcare-dataset-stroke-data.csv
0.3 MB

✓File uploaded to */FileStore/tables/healthcare_dataset_stroke_data.csv*



2024-04-02 - DBFS Example

Overview

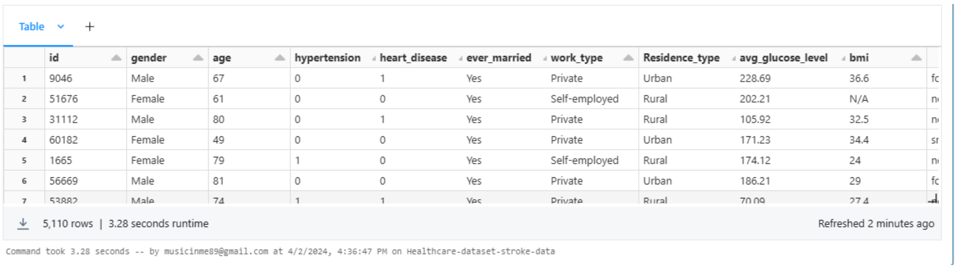
This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

```
# File location and type
file_location = "/FileStore/tables/healthcare_dataset_stroke_data.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

▸ (3) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [id: integer, gender: string ... 10 more fields]



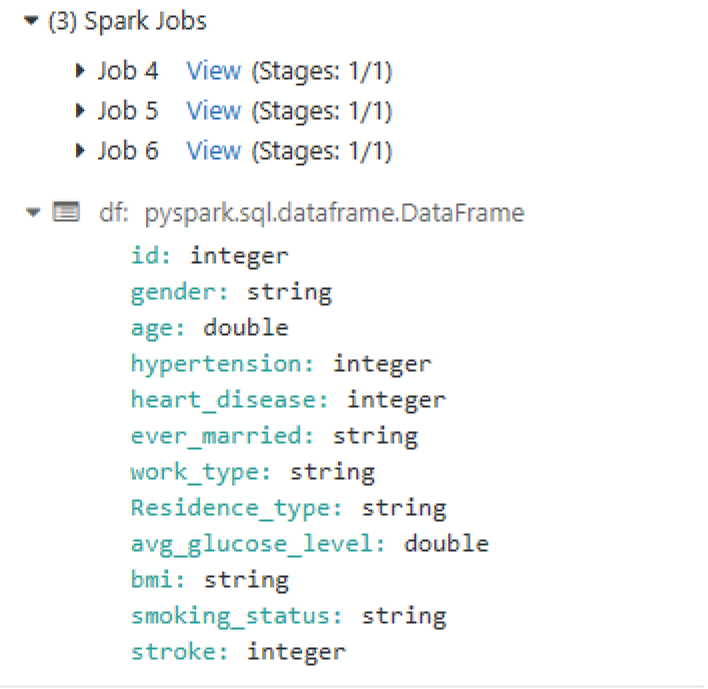
	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	fc
2	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	n
3	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	n
4	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	sr
5	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	n
6	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	fc
7	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	

5,110 rows | 3.28 seconds runtime

Refreshed 2 minutes ago

```
Command took 3.28 seconds -- by musicinme89@gmail.com at 4/2/2024, 4:36:47 PM on Healthcare-dataset-stroke-data
```

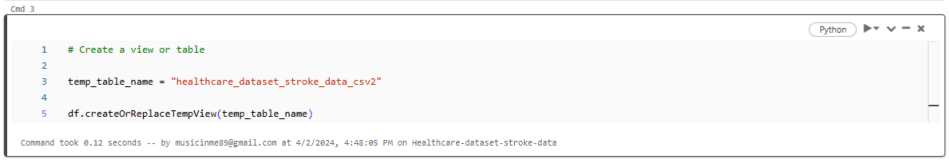
The csv file was uploaded successfully.



```
▾ (3) Spark Jobs
    ▸ Job 4  View (Stages: 1/1)
    ▸ Job 5  View (Stages: 1/1)
    ▸ Job 6  View (Stages: 1/1)
▾ df: pyspark.sql.dataframe.DataFrame
        id: integer
        gender: string
        age: double
        hypertension: integer
        heart_disease: integer
        ever_married: string
        work_type: string
        Residence_type: string
        avg_glucose_level: double
        bmi: string
        smoking_status: string
        stroke: integer
```

The image above shows the data frame and the datatypes of each column the dataset uploaded.

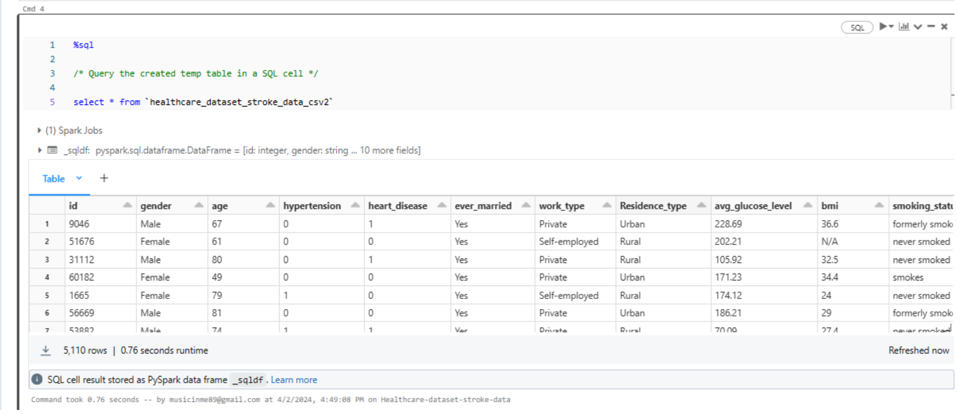
- ➢ Queried the data
- ➢ Created a temp table



```python
# Create a view or table

temp_table_name = "healthcare_dataset_stroke_data_csv2"

df.createOrReplaceTempView(temp_table_name)
```

```
Command took 0.12 seconds -- by musicinme89@gmail.com at 4/2/2024, 4:48:05 PM on Healthcare-dataset-stroke-data
```

Figure 1.1 1



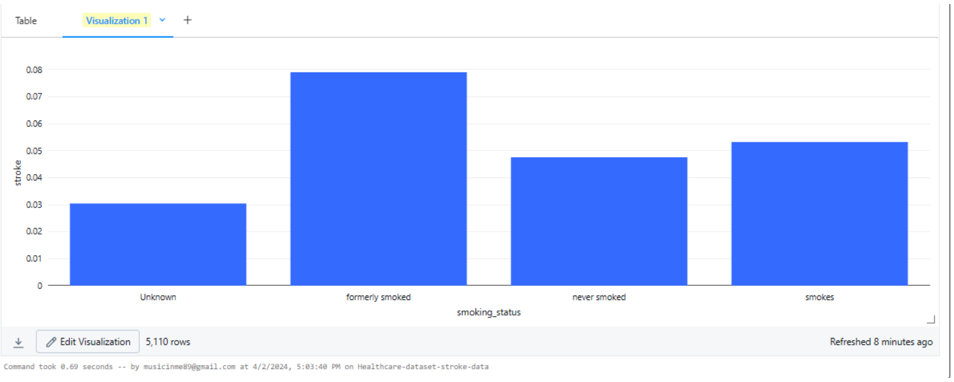
```sql
%sql

/* Query the created temp table in a SQL cell */

select * from `healthcare_dataset_stroke_data_csv2`
```

```
▸ (1) Spark Jobs
▸ _sqldf: pyspark.sql.dataframe.DataFrame = [id: integer, gender: string ... 10 more fields]
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_statu
1	9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smok
2	51676	Female	61	0	0	Yes	Self-employed	Rural	202.21	N/A	never smoked
3	31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked
4	60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes
5	1665	Female	79	1	0	Yes	Self-employed	Rural	174.12	24	never smoked
6	56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smok
7	53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked

```
5,110 rows | 0.76 seconds runtime                                Refreshed now
SQL cell result stored as PySpark data frame _sqldf . Learn more
Command took 0.76 seconds -- by musicinme89@gmail.com at 4/2/2024, 4:49:08 PM on Healthcare-dataset-stroke-data
```

- ➢ Visualized trends and insights.



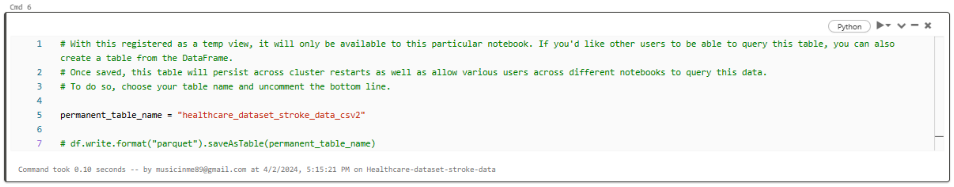
5,110 rows

Refreshed 8 minutes ago

```
Command took 0.69 seconds -- by musicinme89@gmail.com at 4/2/2024, 5:03:40 PM on Healthcare-dataset-stroke-data
```

From the fig above. People who previously smoked were at a higher risk of being diagnosed with stroke.

- Saved the temp table



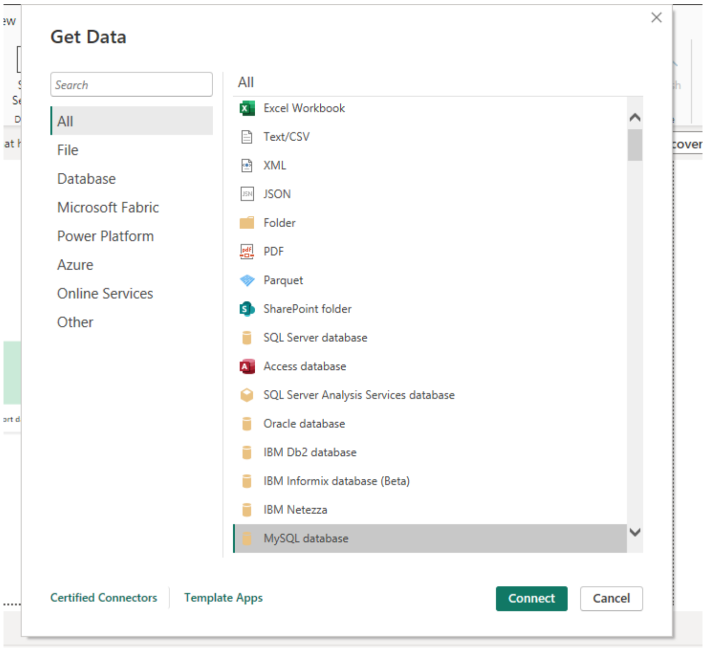
```python
# With this registered as a temp view, it will only be available to this particular notebook. If you'd like other users to be able to query this table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "healthcare_dataset_stroke_data_csv2"

# df.write.format("parquet").saveAsTable(permanent_table_name)
```

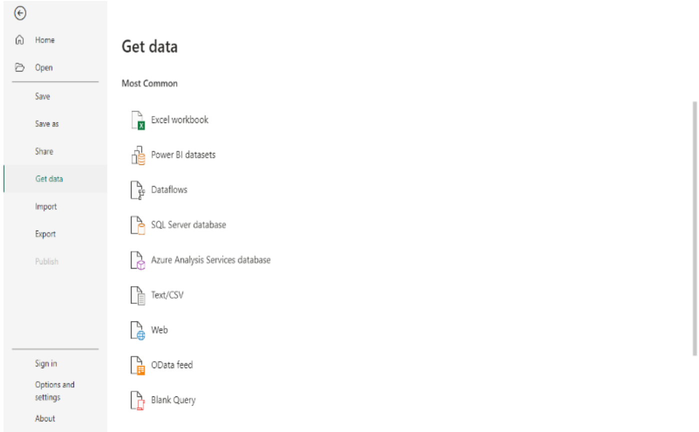
```
Command took 0.10 seconds -- by musicinme89@gmail.com at 4/2/2024, 5:15:21 PM on Healthcare-dataset-stroke-data
```

DATA VISUALIZATION ON POWER BI

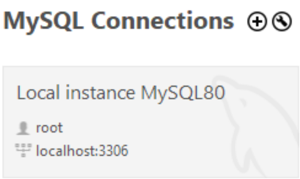
I integrated MySQL with Power BI to create an interactive dashboard and report to generate meaningful insights from the data.

- Connected Power BI to SQL Server database





- Got the server from MySQL

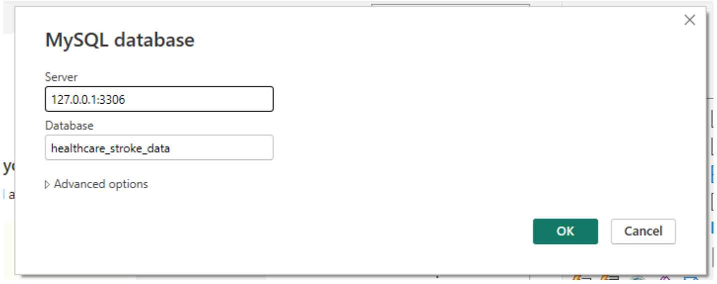


MySQL Connections

Local instance MySQL80

root

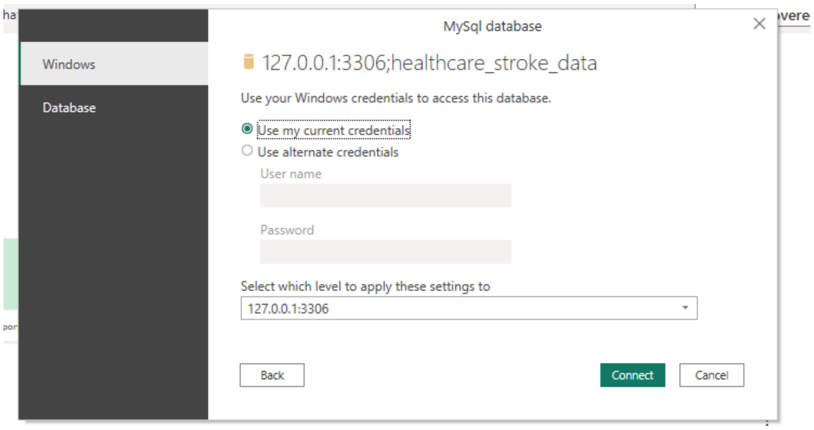
localhost:3306



MySQL database

Server: 127.0.0.1:3306

Database: healthcare_stroke_data



MySql database

127.0.0.1:3306;healthcare_stroke_data

Use your Windows credentials to access this database.

● Use my current credentials

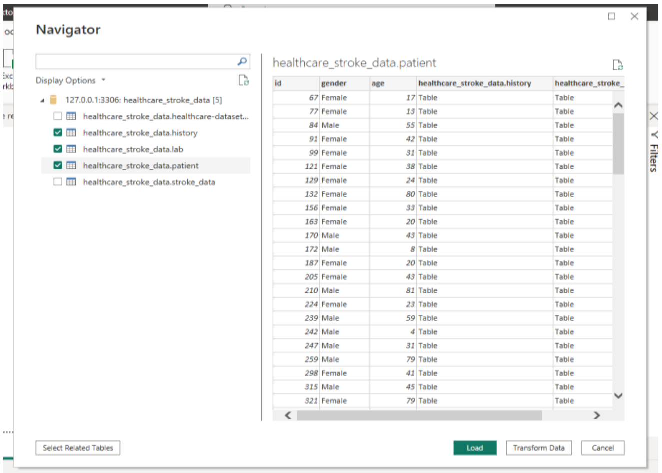
○ Use alternate credentials

User name

Password

Select which level to apply these settings to: 127.0.0.1:3306

- ➢ Selected the relevant tables from the database (Patient, History, Lab)



Navigator

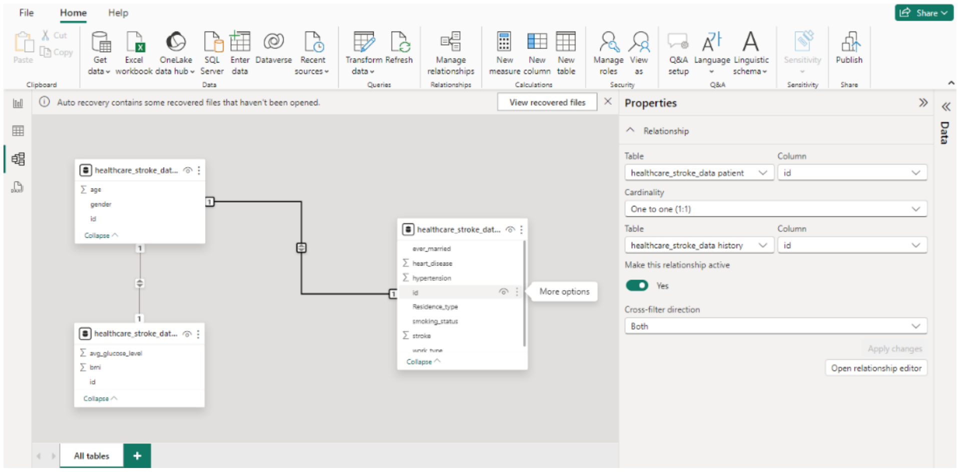
127.0.0.1:3306: healthcare_stroke_data [5]

- □ healthcare_stroke_data.healthcare-dataset...
- ☑ healthcare_stroke_data.history
- ☑ healthcare_stroke_data.lab
- ☑ healthcare_stroke_data.patient
- □ healthcare_stroke_data.stroke_data

healthcare_stroke_data.patient

id	gender	age	healthcare_stroke_data.history	healthcare_stroke_
67	Female	17	Table	Table
77	Female	13	Table	Table
84	Male	55	Table	Table
91	Female	42	Table	Table
99	Female	31	Table	Table
121	Female	38	Table	Table
129	Female	24	Table	Table
132	Female	80	Table	Table
156	Female	33	Table	Table
163	Female	20	Table	Table
170	Male	43	Table	Table
172	Male	8	Table	Table
187	Female	20	Table	Table
205	Female	43	Table	Table
210	Male	81	Table	Table
224	Female	23	Table	Table
239	Male	59	Table	Table
242	Male	4	Table	Table
247	Male	31	Table	Table
259	Male	79	Table	Table
298	Female	41	Table	Table
315	Male	45	Table	Table
321	Female	79	Table	Table

- ➢ Created Relationships



Auto recovery contains some recovered files that haven't been opened.

Properties

Relationship

Table: healthcare_stroke_data patient

Column: id

Cardinality: One to one (1:1)

Table: healthcare_stroke_data history

Column: id

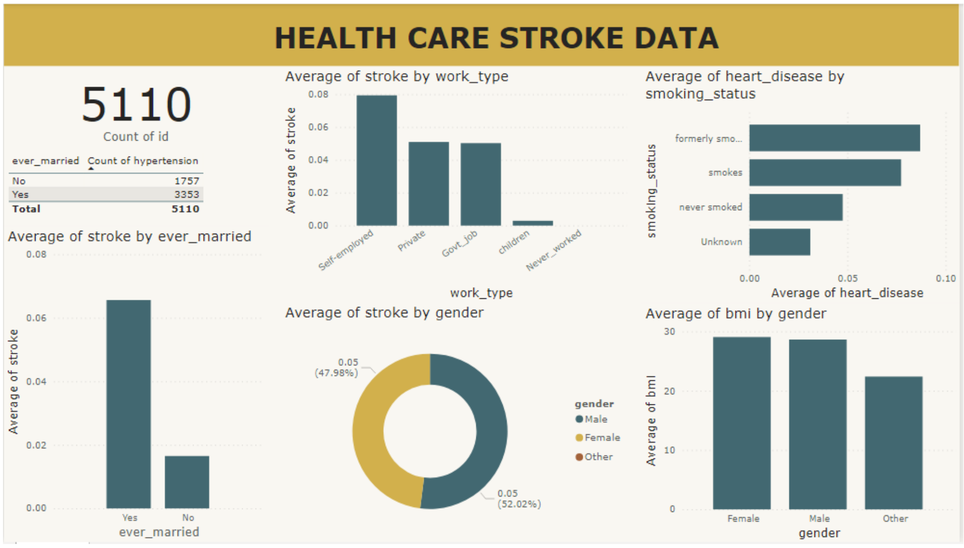
Make this relationship active: Yes

Cross-filter direction: Both

Apply changes

Open relationship editor

- ➢ Created an interactive dashboards and report



ever_married	Count of hypertension
No	1757
Yes	3353
Total	**5110**

Insights derived from the dashboard data:

- We have 5110 patients in our dataset.
- Patients with a history of marriage tend to have a higher incidence of hypertension.
- Self-employed individuals show a higher propensity for stroke compared to other occupational groups, while children were not diagnosed with stroke.
- Heart diseases are more prevalent among patients with a history of smoking.
- Patients with a marital history are more prone to being diagnosed with stroke.
- Men constitute 52.02% of stroke diagnoses, indicating a higher likelihood of stroke diagnosis in men compared to women that constitutes 47.98%.
- On average, women have a higher BMI than men.

RECOMMENDATION

Based on the insights provided by the dashboard data, several recommendations can be made:

1. Targeted Screening and Education: Focus on screening and education for hypertension among married individuals.

2. Stroke Prevention for Self-Employed: Develop strategies to prevent stroke among self-employed individuals, including stress management and awareness campaigns.

3. Smoking Cessation Programs: Implement smoking cessation initiatives to reduce heart disease among smokers.

4. Stroke Awareness Campaigns: Launch campaigns to raise awareness about stroke prevention and symptoms, particularly targeting married individuals.

5. Gender-Specific Health Initiatives: Develop health programs tailored to address the higher stroke risk in men and higher BMI in women.

6. Occupational Health Interventions: Provide occupational health support for self-employed individuals to mitigate stroke risk factors related to their work.