

Data Engineering Capstone Project- Data Architecture Design and Transition to AWS

Introduction

The goal of this project is to solve a business problem. We will design the data architecture for a small private clinic.

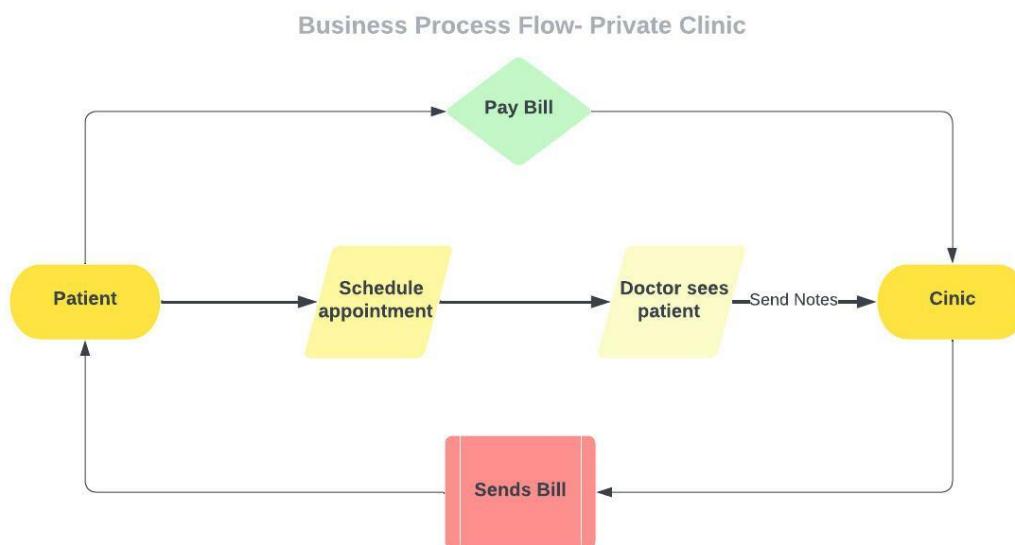
This is a fictional healthcare clinic.

- Name: UnicornHealth
- Scope: Local
- Patients: 500
- Employees: 25
- Kind of practice: Family Practice

In order to do design the data architecture, we need to understand the problem. After this, we will design a business process diagram to visualize the business model. Then, we will look at all relevant data and finally, we will understand how all the data relates. We will transition the business to a cloud environment to automate the data process.

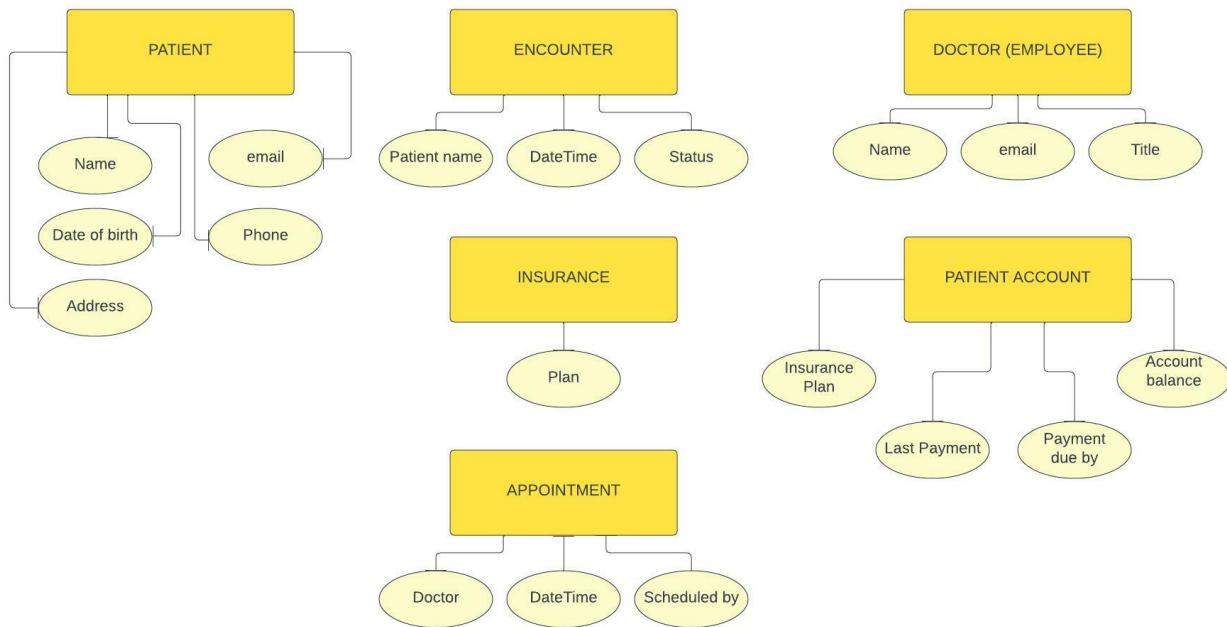
Lets visualize the basic business model:

We used [lucidchart](#) to make the business model diagram.



Schematic Presentation of Data Structure

The business model diagram, helped with the visualization of the basic business process of UnicornHealth. Next, we will determine the basic Data Structure by identifying the main elements, which are **patient**, **appointment**, **encounter**, **account**, **insurance** and **employee**. We also generated mock data related to billing and insurance . The attributes for each element are being also identified. Finally, we will proceed to identify the connections between the entities.



DATA CREATION PROCESS

Database creation

We used a [mock data generator](#) to generate fake but realistic data. Open [this file](#) to get an idea of the conditions we established to generate the data.

We loaded CSV files directly into our test environment which in our case was MySQL Workbench.

[account.csv file](#)

[employee.csv file](#)

[appointment.csv file](#)

[encounter.csv file](#)

[insurance.csv file](#)

[patient.csv file](#)

Determination of foreign and primary keys

We assigned the first column of each table as a primary key. Each table has a primary key, which is unique in the dataset.

The foreign are primary keys that appear in other tables. We used MySQL workbench which automatically identifies the foreign keys.

Data Definition Language (DDL)

We especified the storage groups to be used. We defined the data, primary keys and foreign keys in Mysql Workbench.

```
CREATE DATABASE unicornhealth !/40100 DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci !/80016 DEFAULT ENCRYPTION='N' ;

CREATE TABLE account ( accountId varchar(12) NOT NULL, member_id varchar(25)
DEFAULT NULL, last_payment decimal(10,0) DEFAULT NULL, payment_due
varchar(20) DEFAULT NULL, account_balance varchar(20) DEFAULT NULL, PRIMARY
KEY ( accountId ) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE appointment ( appId varchar(12) NOT NULL, physician varchar(25)
DEFAULT NULL, app_date varchar(10) DEFAULT NULL, app_time varchar(10) DEFAULT
NULL, scheduled_by varchar(20) DEFAULT NULL, PRIMARY KEY ( appId ) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE employee ( employee_id varchar(15) NOT NULL, first_name
varchar(20) DEFAULT NULL, last_name varchar(20) DEFAULT NULL, email varchar(40)
DEFAULT NULL, phone_number varchar(12) DEFAULT NULL, title varchar(20)
DEFAULT NULL, PRIMARY KEY ( employee_id ) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

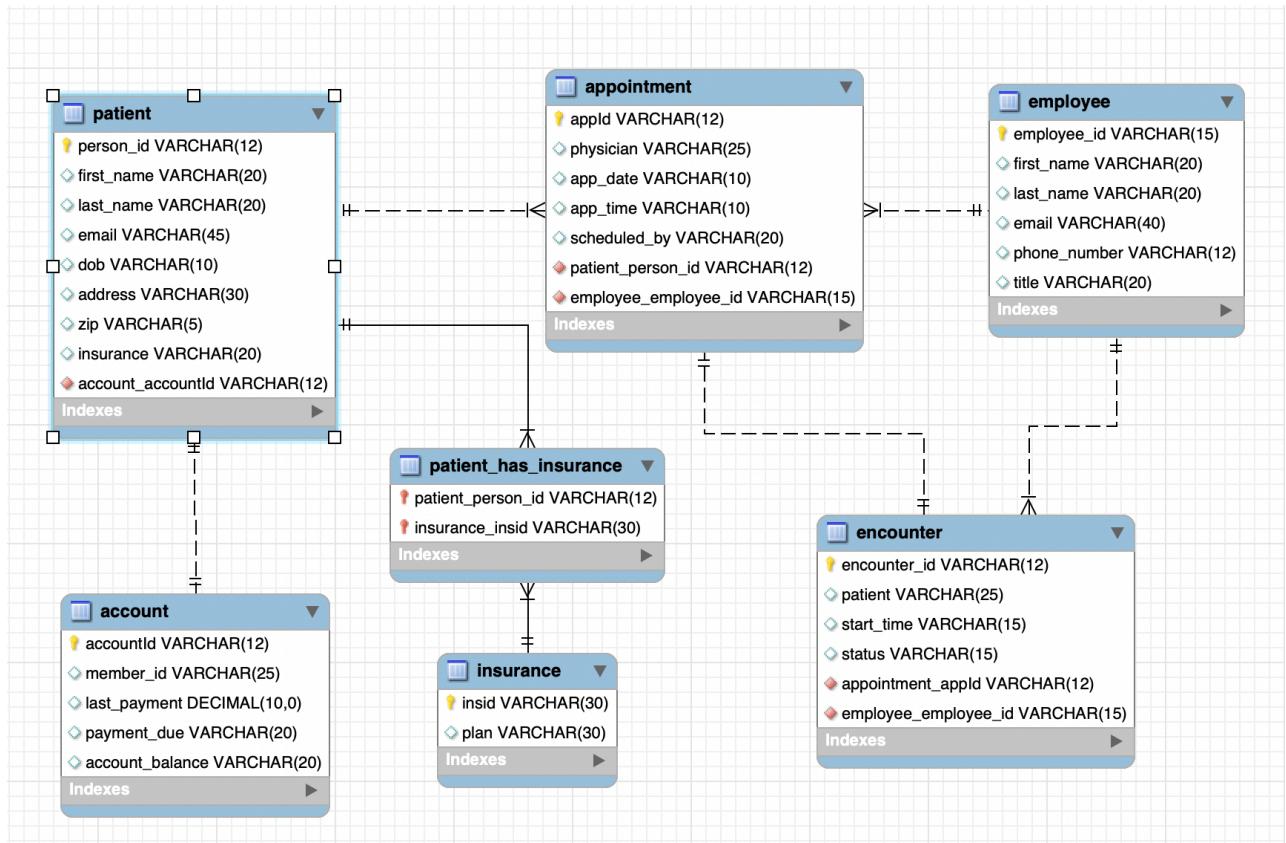
CREATE TABLE encounter ( encounter_id varchar(12) NOT NULL, patient
varchar(25) DEFAULT NULL, start_time varchar(15) DEFAULT NULL, status
varchar(15) DEFAULT NULL, PRIMARY KEY ( encounter_id ) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE insurance ( insid varchar(30) NOT NULL, plan varchar(30)
DEFAULT NULL, PRIMARY KEY ( insid ) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE patient ( person_id varchar(12) NOT NULL, first_name
varchar(20) DEFAULT NULL, last_name varchar(20) DEFAULT NULL, email varchar(45)
DEFAULT NULL, dob varchar(10) DEFAULT NULL, address varchar(30) DEFAULT NULL,
zip varchar(5) DEFAULT NULL, insurance varchar(20) DEFAULT NULL, PRIMARY KEY
( person_id ) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

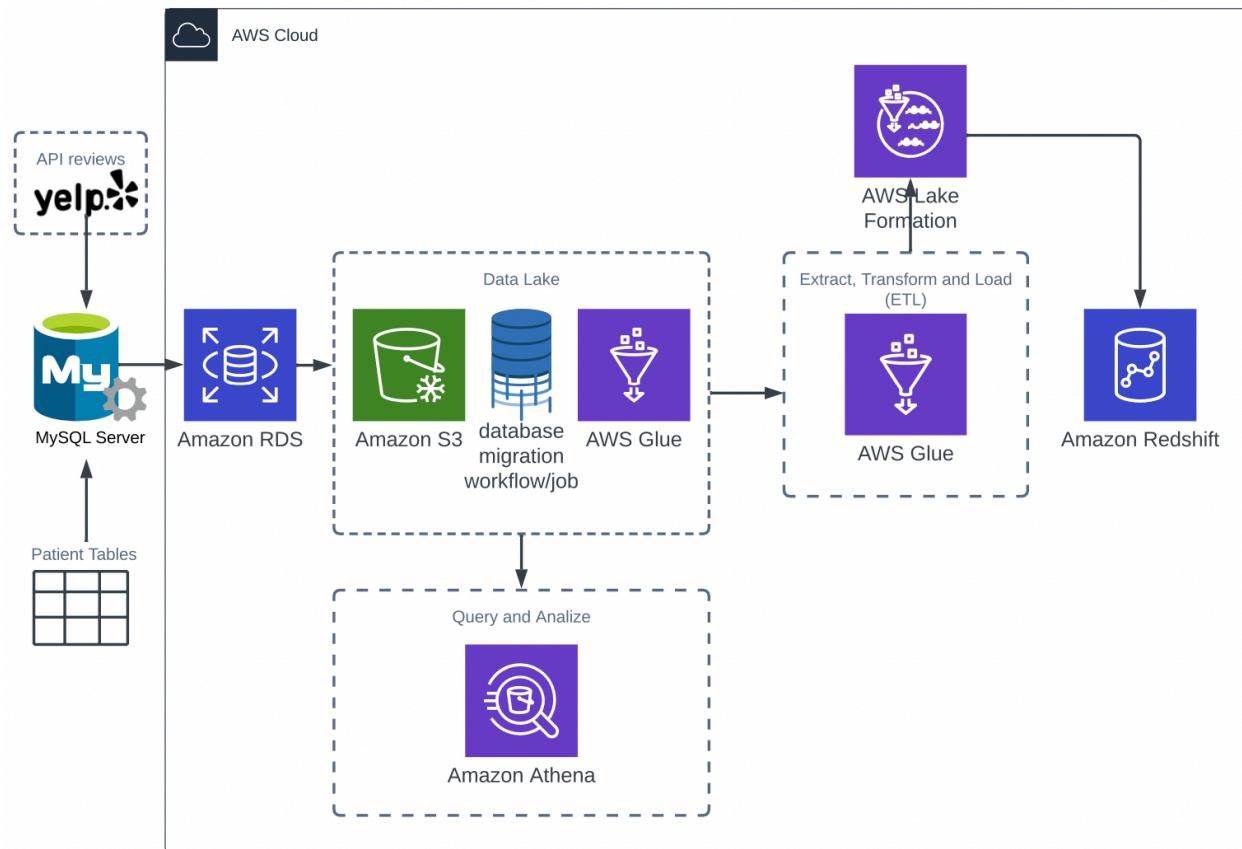
Enhanced entity-relationship (EER)

This model is helpful as a tool to plan the database and visualize the relationships among the tables.



Amazon Web Services (AWS) Architecture

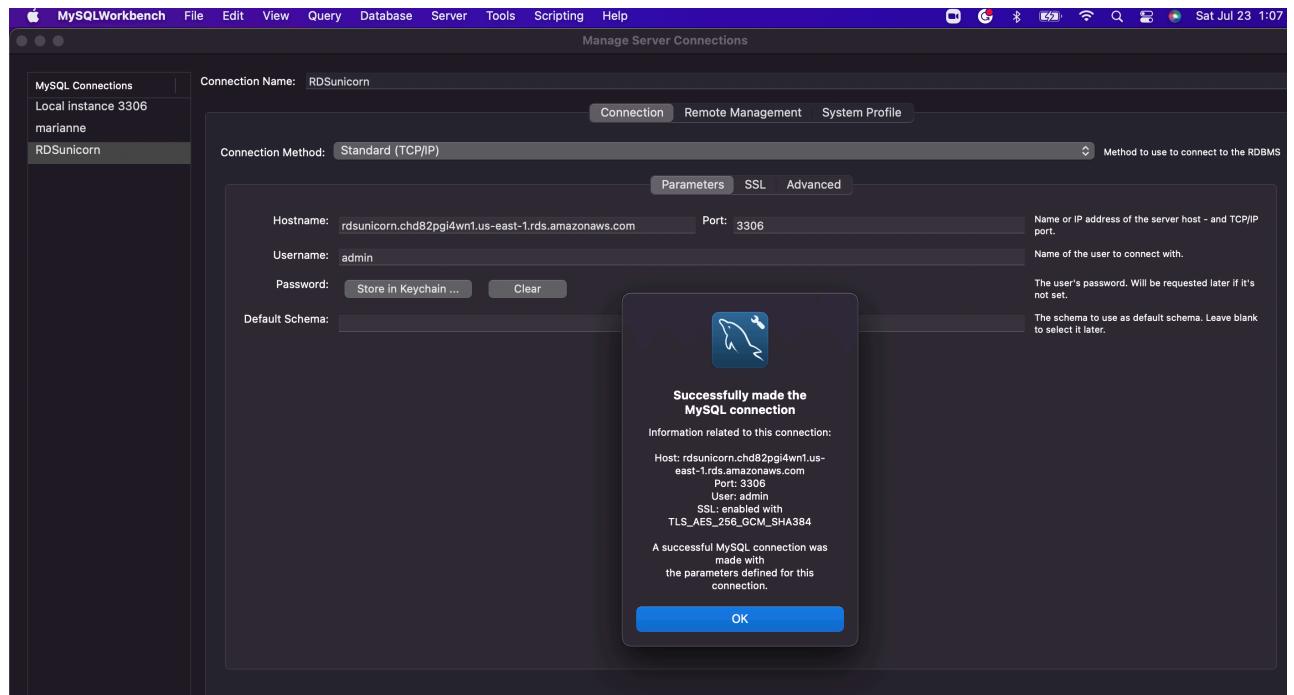
This is a visual guide to help us understand the flow of the data, as well as the microservices used.



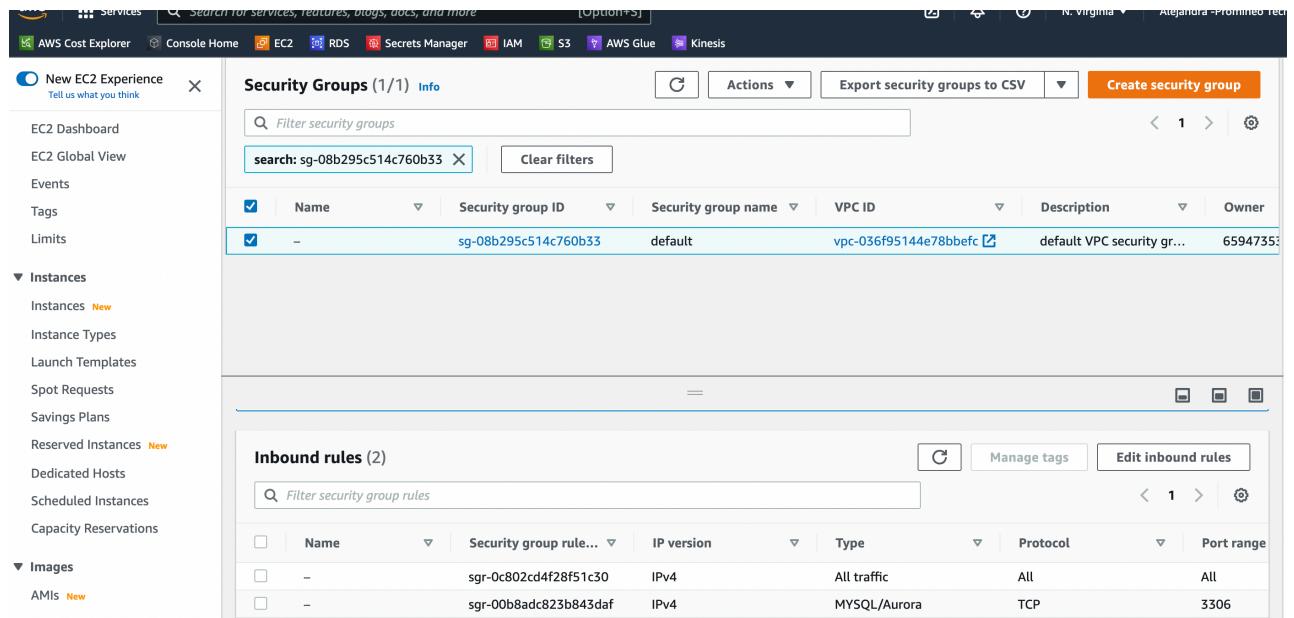
Database Migration from Client Server to AWS RDS

To migrate the database to AWS RDS we needed to, first create a database using Easy Create, MySQL, Free Tier engine in RDS.

Second, we went to MySQL workbench to establish a connection from local port to the endpoint of the AWS RDS database we just created



Third, we tested the connection, and it failed the first time. It was necessary to alter the inbound rules to the Security Group in our database AWS RDS to allow traffic from MySQL/Aurora port 3306 as shown in the image below



After, used Python to query the database from AWS as shown. Code source is in <https://www.w3schools.com/>

In []:

```
## Query data from MySQL database in Python

import mysql.connector ##--> or any mysql-python library

mydb = mysql.connector.connect(
    host="rdsunicorn.chd82pgi4wnl.us-east-1.rds.amazonaws.com", ##-->-endpoint
    user="admin", ##,<-- user name in AWS RDS database
    password="Capstone22", ##<-- database password
    database="unicornDB" ##<-- name of the database (NOT THE IDENTIFIER!)
)

mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM patient;")

myresult = mycursor.fetchall()

for x in myresult:
    print(x)
```

For the purpose of demonstration, we can reformat the query into a list of dictionaries for each row. In this case, we are showing the 'patient' table. Notice that indexing starts at 1, because we are using the indexing from SQL that starts in 1, not the indexing of Python which would start at 0.

In []:

```
## Querying MySQL RDS columns in a table into a list of dictionaries

import boto3
import mysql.connector
import json

cnx = mysql.connector.connect(user='admin', password='Capstone22', ##<--- us
                               host='rdsunicorn.chd82pgi4wn1.us-east-1.rds.amazonaws.com', ## <-- endpoint d
                               database='unicornDB') ##<--<database>

query = "SELECT * FROM patient"

cur = cnx.cursor()
results = cur.execute(query)
rows = cur.fetchall()

# Formatting the Query into a list of dictionaries for each row. In this case
# has 8 columns
Rows_lst = []
for i in rows:
    i_dict = {'person_id': i[1], 'first_name': i[2],
              'last_name': i[3], 'email': i[4],
              'dob': i[5], 'address': i[6], 'zip': i[7], 'insurance': i[8]}
    Rows_lst.append(i_dict)

print(i_dict)
```

Ingest data from RDS to S3

To move data from RDS to S3 we utilized Database Migration Service. First we created a Replication instance named unicornz.

Second we created a source and target endpoint, rdsunicorn and unicornt-target7 respectfully.

Finally, we created and sucessfully ran a database migration task called unicorn-herder. This sucessfully moved all of our data from RDS to S3.

The screenshot shows two main sections of the AWS DMS console:

Database migration tasks (1)

Identifier	Status	Progress	Type	Source	Target	Replication instance
unicorn-herder	Load complete	100%	Full load	rdsunicorn	unicornt-target7	unicornz

Endpoints (2)

Name	Type	Status	Engine	Server name
rdsunicorn	Source	Active	MySQL	rdsunicorn.chd82pgi4wn1.us-east-1.rds.amazonaws.com
unicornt-target7	Target	Active	Amazon S3	-

The screenshot shows the AWS Glue Console Dashboard. At the top, there are tabs for 'Action Service' and 'AWS Glue Console'. The URL in the address bar is <https://us-east-1.console.aws.amazon.com/dms/v2/home?region=us-east-1#dashboard>. A search bar at the top right contains the placeholder 'Search for services, features, blogs, docs, and more'.

The main area is titled 'Dashboard' and includes a sidebar with sections like 'Overview', 'Tasks', 'Jobs', 'Logs', and 'Metrics'. The 'Tasks' section is currently selected, showing the following data:

Database migration tasks	View all database migration tasks	Create database migration task
0 Active Info No active tasks.	0 Error Info No error tasks.	0 Failed No failed tasks.
1 Load completed unicorn-herder		View more load completed tasks

The 'Resources' section displays the following counts:

Database migration tasks	Replication instances	Endpoints
Max quota: 600 1	Max quota: 60 1	Max quota: 1000 2

Glue Crawler & Athena

Every table was crawled from S3 and listed in a glue database catalog. Data was verified by querying with Athena.

<https://console.aws.amazon.com/glue/home?region=us-east-1#catalog:tab=tables>

Services, features, blogs, docs, and more [Alt+S] N. Virginia ▾ Maria

Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Add tables ▾	Action ▾	Filter by attributes or search by keyword	Save view ▾	Showing: 1 - 10
Name	Database	Location	Classification	Last updated
patient_csv	unicorn-database-glue	s3://unicornhealth/patient...	csv	24 July 2022 3:33 AM
employee_csv	unicorn-database-glue	s3://unicornhealth/emplo...	csv	24 July 2022 3:33 AM
encounter_csv	unicorn-database-glue	s3://unicornhealth/encou...	csv	24 July 2022 3:33 AM
insurance_csv	unicorn-database-glue	s3://unicornhealth/insura...	csv	24 July 2022 3:33 AM
appointment	unicorn-database-glue	s3://unicornhealth/	csv	21 July 2022 8:02 AM
mock_data_1_csv	unicorn-database-glue	s3://unicornhealth/MOCK...	csv	24 July 2022 3:33 AM
yelpmerge1_csv	unicorn-database-glue	s3://unicornhealth/yelpm...	csv	24 July 2022 3:33 AM
account_csv	unicorn-database-glue	s3://unicornhealth/accou...	csv	24 July 2022 3:33 AM
capstoneer_png	unicorn-database-glue	s3://unicornhealth/Capst...	Unknown	24 July 2022 3:33 AM
appointment_csv	unicorn-database-glue	s3://unicornhealth/appoin...	csv	24 July 2022 3:33 AM

<https://console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/84c9585b-abe5-44cb-b053-17844f59910c>

Services, blogs, docs, and more [Alt+S] N. Virginia ▾ Maria

Run again Explain ▾ Cancel Save ▾ Clear Create ▾

Query results Query stats

Completed Time in queue: 232 ms Run time: 739 ms Data scanned: 7

Results (10) Copy Download results

Search rows < 1 >

#	col0	col1	col2	col3	col4	col5	col6
1	Daniela A.	"Manteca	0	1	5/20/2022	"The staff at	
2	Miriam B.	"Bell	0	5	5/27/2022	"Fast and rel	
3	Alice K.	"Los Angeles	94	75	5/27/2022	"I have come	
4	G V.	"Los Angeles	4	29	4/5/2022	Love the pec	
5	Stephanie W.	"Los Angeles	466	258	3/4/2022	"After postin	
6	Portida T.	"Los Angeles	172	342	1/7/2022	"No doctor is	

AWS Glue Console

aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/8b013acf-20b1-494f-8047-24c2eabdd1f5

Services, features, blogs, docs, and more [Alt+S]

N. Virginia ▾ Marian

SQL Ln 1, Col 1

Run again Explain Cancel Save Clear Create

Query results Query stats

Completed Time in queue: 123 ms Run time: 463 ms Data scanned

Results (10) Copy Download r

Search rows

#	col0	col1	col2	col3	col4	col5
1	0	APPO1	E00023	3/14/2022	10:08 AM	C
2	1	APPO2	E00020	1/16/2022	10:17 AM	M
3	2	APPO3	E00021	7/15/2021	2:14 PM	C
4	3	APPO4	E0007	7/15/2022	12:43 PM	C

Lake Formation Process

Administrative User

Per AWS Lake Formation Developer Guide, Administrators should not be Lake Formation Data Lake Administrators. To abide by the guidelines, we created a new user group in IAM "LakeAdministrators" and a separate users and roles to manage this microservice.

<https://us-east-1.console.aws.amazon.com/lakeformation/home?region=us-east-1#register-list>

The screenshot shows a table titled "Review permissions for s3://dmsunicorn" with the subtitle "Resources and their permissions in the selected Amazon S3 paths." The table has six columns: Resource type, Resource, Principal type, Principal, Permissions, and Grantable. There are four rows of data:

Resource type	Resource	Principal type	Principal	Permissions	Grantable
Database	unicorn-data-lake	IAM user	Marianne	Super, Alter, Create table, Describe, Drop	Super, Alter, Create table, Describe, Drop
Database	unicorn-data-lake	IAM user	Aleja	Super	Super
Table	appointment	IAM role	unicornglue2	Super, Alter, Delete, Describe, Drop, Insert	Super, Alter, Delete, Describe, Drop, Insert
Column	unicorn-from-dms.appoint	IAM role	unicornglue2	Select	Select

Redshift Cluster

We set up a serverless redshift cluster that connected to our Data Lake. All of our data was linked as an external table and queried successfully.

The screenshot shows the Amazon Redshift SQL Workbench interface. On the left, the sidebar displays the database structure under 'serverless: default'. It includes a 'public' folder, a 'unicorndb' folder containing 'account' and 'appointment' tables, and other folders like 'dev' and 'test'. In the main workspace, a green success message box is visible stating 'Schema "unicorndb" is created successfully.' Below it, a query editor window shows a single-line SELECT statement: 'SELECT * FROM "dev"."unicorndb"."account";'. The results pane, titled 'Result 1 (100)', displays a table with 10 rows of data. The columns are labeled 'col0', 'col1', 'col2', 'col3', and 'col4'. The data includes various account identifiers and their corresponding values. At the bottom right of the results pane, the text 'Elapsed time: 453ms' is shown. The top navigation bar shows several tabs: 'editor v2', 'copy command redshift - Se...', 'Amazon Redshift COPY Com...', 'COPY examples - Amazon Re...', and 'S3 Management Console'.

Type	NN	NN	CMP
string	NN	NN	

col0	col1	col2	col3	col4
ACC00010	INS2	+1.100000000000000e+01	6/13/2022	\$992.5
ACC000100	INS3	+2.000000000000000e+00	9/29/2021	\$1933.
ACC000101	INS1	+3.000000000000000e+00	7/21/2022	\$1297.
ACC000102	INS5	+2.000000000000000e+00	9/5/2022	\$443.8
ACC000103	INS3	+6.000000000000000e+00	1/24/2022	\$941.2
ACC000104	INS1	+1.000000000000000e+01	11/6/2022	\$1001.

The screenshot shows the Amazon SQL Workbench interface. On the left, there's a sidebar with a tree view of database resources under 'Serverless: default'. The 'dev' node is expanded, showing 'public', 'unicorndb', and 'sample_data_dev'. In the main area, a query editor window titled 'query editor v2' is open. It displays a success message: 'Schema "unicorndb" is created successfully.' Below this, a query is run: 'SELECT * FROM "dev"."unicorndb"."appointment";'. The results are shown in a table titled 'Result 1 (100)'. The table has columns: col0, col1, col2, col3, and col4. The data includes rows like ACC0001, INS4, +7.00000000000000e+00, 10/12/2021, \$2419. and ACC00010, INS2, +1.10000000000000e+01, 6/13/2022, \$992.5. An 'Elapsed time: 131' message is at the bottom right of the results table.

col0	col1	col2	col3	col4
ACC0001	INS4	+7.00000000000000e+00	10/12/2021	\$2419.
ACC00010	INS2	+1.10000000000000e+01	6/13/2022	\$992.5
ACC000100	INS3	+2.00000000000000e+00	9/29/2021	\$1933.
ACC000101	INS1	+3.00000000000000e+00	7/21/2022	\$1297.
ACC000102	INS5	+2.00000000000000e+00	9/5/2022	\$443.8
ACC000103	INS6	+2.00000000000000e+00	10/12/2021	\$111.2

Conclusion

We demonstrated the process of creating a database from a business flow diagram as a startpoint. We successfully achieved the migration of data to AWS to a data lake and, finally, we implemented a warehouse to be able to query and analyze data as needed.