# 

Abstract

In this assignment, we are designing and developing a database for library management system, which includes a conceptual model, and a logical model. The goal is to practice the skills needed to translate real-world scenarios into database designs.

**Database design & Implementation**

**INFT-1111: Case Study**

**Table of Contents**

[**Library Management System** 2](#_Toc184778612)

[**Business Use Case** 2](#_Toc184778613)

[**Problem Statement** 2](#_Toc184778614)

[**Objective** 2](#_Toc184778615)

[**Data Requirements** 2](#_Toc184778616)

[**Database Design** 2](#_Toc184778617)

[**Conceptual Model** 2](#_Toc184778618)

[**Entity Relationship Model - ER Diagram** 4](#_Toc184778619)

[**Logical Model: SQL Table Creation Scripts** 5](#_Toc184778620)

[**Database Deployment** 7](#_Toc184778621)

[**Amazon RDS Deployment - PostgreSQL** 7](#_Toc184778622)

[**DynamoDB Deployment** 8](#_Toc184778623)

[**Security and Backup Management** 9](#_Toc184778624)

[**Amazon RDS Settings** 9](#_Toc184778625)

[**DynamoDB Settings** 11](#_Toc184778626)

[**Rationale** 12](#_Toc184778627)

# **Library Management System**

## **Business Use Case**

### **Problem Statement**

Manual management of the library inventory, member records, and its borrowing/reservation system involves a lot of complexities and results in several errors. Operational inefficiencies, poor customer experience, and revenue loss arise because of inefficient tracking of availability of books, overdue items, and membership details.

**Objective**

Designing a database for Library has following objectives:

* To automate member and book inventories management.
* To provide an effective reservation and checkout system.
* To increase accessibility to catalog information and maintain accurate reporting.
* Maintain data integrity and security while providing a scalable system.

### **Data Requirements**

The system needs to keep track of the following:

* Member Information: Member ID, personal information, and type of membership.
* Inventory Details: Books and magazines with genres and authors.
* Transactions: Reservations and checkouts with dates and times and details of items.
* Relationships: Members to membership types, books to authors, and items to genres.

## **Database Design**

### **Conceptual Model**

#### **Business Definition Table**

|  |  |  |
| --- | --- | --- |
| Entity | Attributes | Description |
| Member | MemberID (PK), Name, Address, Phone, Email | Individuals who are registered to borrow items from the library. |
| MembershipType | MembershipTypeID (PK), TypeName, Benefits | Different types of memberships available (e.g., Adult, Child, Senior). |
| MemberMembership | MemberID (PK), MembershipTypeID (PK) | This table links Members to the Membership types that they own. |
| Book | BookID (PK), Title, ISBN, PublicationYear, GenreID (FK) | Books those are available in the library. |
| Magazine | MagazineID (PK), Title, IssueNumber, PublicationDate, GenreID (FK) | Magazines that are available in the library. |
| Genre | GenreID (PK), GenreName | Categories by which books and magazines are classified. |
| Author | AuthorID (PK), FirstName, LastName, Biography | Authors who have written books that are available in the library. |
| Reservation | ReservationID (PK), MemberID (FK), ItemID, ReservationDate | Records of items reserved by members. |
| Checkout | CheckoutID (PK), MemberID (FK), ItemID, CheckoutDate, DueDate | Records of items currently checked out by members. |
| Item | ItemID (PK), Title, Type (Book/Magazine), GenreID (FK) | General representation of library items (books and magazines). |
| BookAuthor | BookID (FK), AuthorID (FK) | Junction table representing the many-to-many relationship between books and authors. |

#### **Relationships**

* A Member can be related to many MembershipTypes.
* A member may be able to reserve many items, but he or she can only check out up to 5 items at a time.
* A Book has one Genre it belongs to and is written by several Authors.
* One Magazine is one Genre.
* Both Members and Items are associated with Reservations and Checkouts.

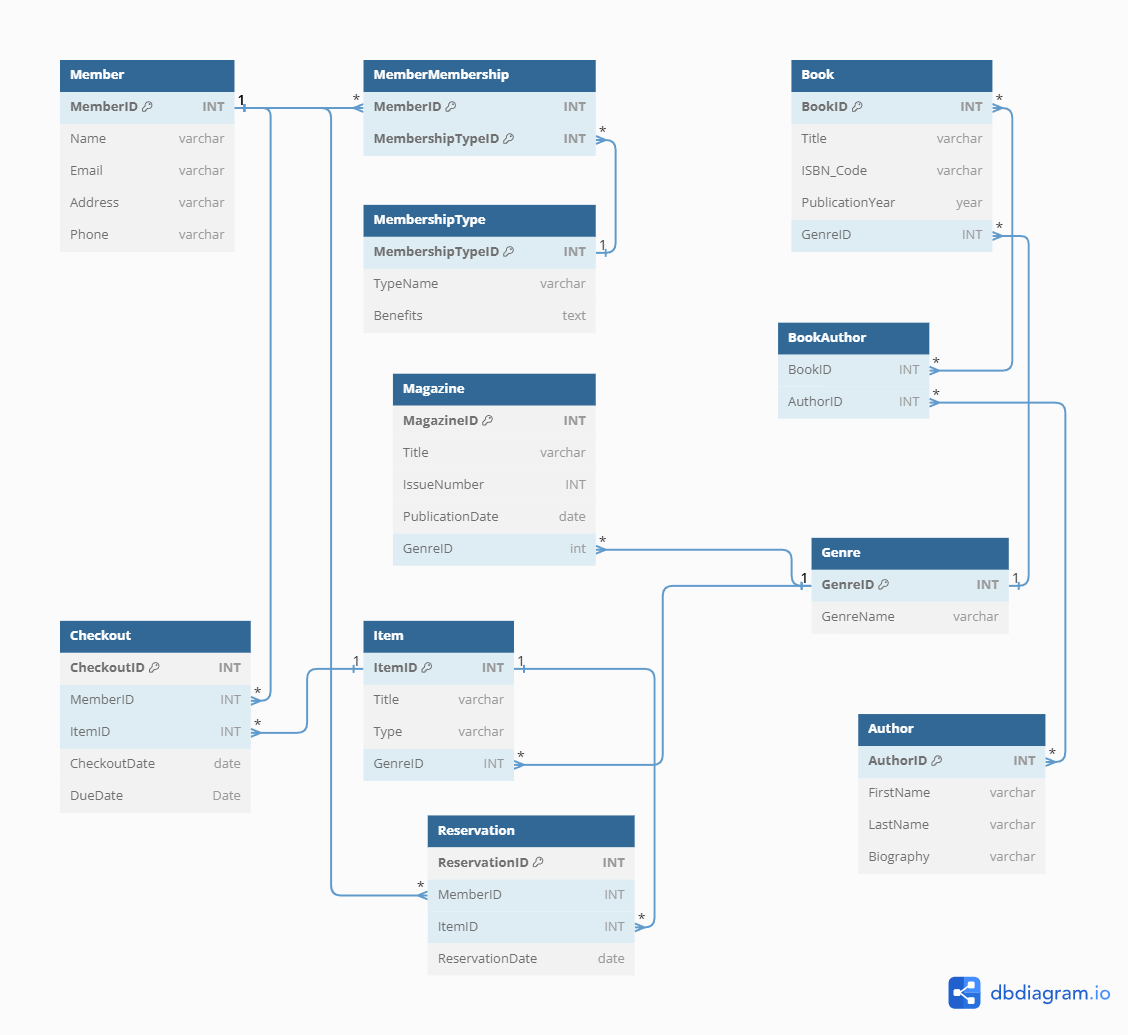
#### **Constraints**

* A Member can have a maximum of 5 active Checkouts.
* ItemType must be either 'Book' or 'Magazine'.

#### **Relationships and Cardinality**

* Member to MembershipType: One-to-Many (A member can have multiple membership types)
* Member to Reservation: One-to-Many (A member can have multiple reservations)
* Member to Checkout: One-to-Many (A member can have up to 5 checkouts)
* Book to Genre: Many-to-One (A book belongs to one genre)
* Magazine to Genre: Many-to-One (A magazine belongs to one genre)
* Book to Author: Many-to-Many via BookAuthor
* Reservation to Item: Many-to-One (A reservation is for one item)
* Checkout to Item: Many-to-One (A checkout is for one item)

### **Entity Relationship Model - ER Diagram**



### **Logical Model: SQL Table Creation Scripts**

**Table: Genre**

CREATE TABLE Genre (

GenreID INT **PRIMARY KEY**,

GenreName VARCHAR(100) NOT NULL

);

**Table: Member**

CREATE TABLE Member (

MemberID INT **PRIMARY KEY**,

Name VARCHAR(255) NOT NULL,

Address VARCHAR(255),

Phone VARCHAR(20),

Email VARCHAR(100) UNIQUE

);

**Table: MembershipType**

CREATE TABLE MembershipType (

MembershipTypeID INT **PRIMARY KEY,**

TypeName VARCHAR(50) NOT NULL,

Benefits TEXT

);

**Table: MemberMembership**

CREATE TABLE MemberMembership (

MemberID INT,

MembershipTypeID INT,

**PRIMARY KEY** (MemberID, MembershipTypeID),

**FOREIGN KEY** (MemberID) REFERENCES Member(MemberID),

**FOREIGN KEY** (MembershipTypeID) REFERENCES MembershipType(MembershipTypeID)

);

**Table: Item**

CREATE TABLE Item (

ItemID INT **PRIMARY KEY,**

Title VARCHAR(255) NOT NULL,

Type VARCHAR(20) CHECK (Type IN ('Book', 'Magazine')),

GenreID INT,

**FOREIGN KEY** (GenreID) REFERENCES Genre(GenreID)

);

**Table: Book**

CREATE TABLE Book (

BookID INT **PRIMARY KEY**,

Title VARCHAR(255) NOT NULL,

ISBN VARCHAR(20) UNIQUE,

PublicationYear YEAR,

GenreID INT,

**FOREIGN KEY** (GenreID) REFERENCES Genre(GenreID)

);

**Table: Magazine**

CREATE TABLE Magazine (

MagazineID INT **PRIMARY KEY**,

Title VARCHAR(255) NOT NULL,

IssueNumber INT,

PublicationDate DATE,

GenreID INT,

**FOREIGN KEY** (GenreID) REFERENCES Genre(GenreID)

);

**Table: Author**

CREATE TABLE Author (

AuthorID INT **PRIMARY KEY**,

FirstName VARCHAR(100),

LastName VARCHAR(100),

Biography TEXT

);

**Table: BookAuthor**

CREATE TABLE BookAuthor (

BookID INT,

AuthorID INT,

**PRIMARY KEY** (BookID, AuthorID),

**FOREIGN KEY** (BookID) REFERENCES Book(BookID),

**FOREIGN KEY** (AuthorID) REFERENCES Author(AuthorID)

);

**Table: Reservation**

CREATE TABLE Reservation (

ReservationID INT **PRIMARY KEY**,

MemberID INT,

ItemID INT,

ReservationDate DATE,

**FOREIGN KEY** (MemberID) REFERENCES Member(MemberID),

**FOREIGN KEY** (ItemID) REFERENCES Item(ItemID)

);

**Table: Checkout**

CREATE TABLE Checkout (

CheckoutID INT **PRIMARY KEY**,

MemberID INT,

ItemID INT,

CheckoutDate DATE,

DueDate DATE,

**FOREIGN KEY** (MemberID) REFERENCES Member(MemberID),

**FOREIGN KEY** (ItemID) REFERENCES Item(ItemID),

**CONSTRAINT chk\_max\_checkouts CHECK (**

**(SELECT COUNT(\*) FROM Checkout WHERE MemberID = Checkout.MemberID) <= 5**

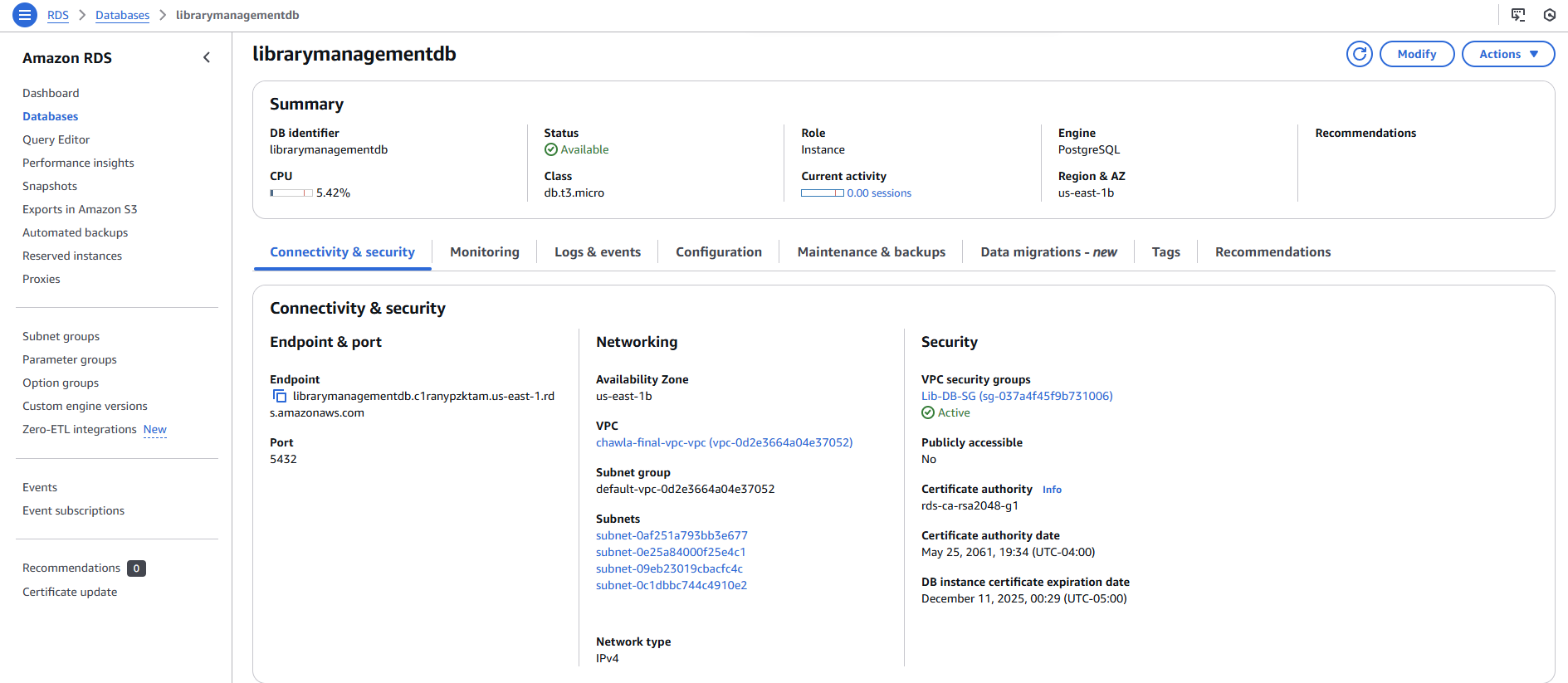
)

); --member can checkout a maximum of 5 items

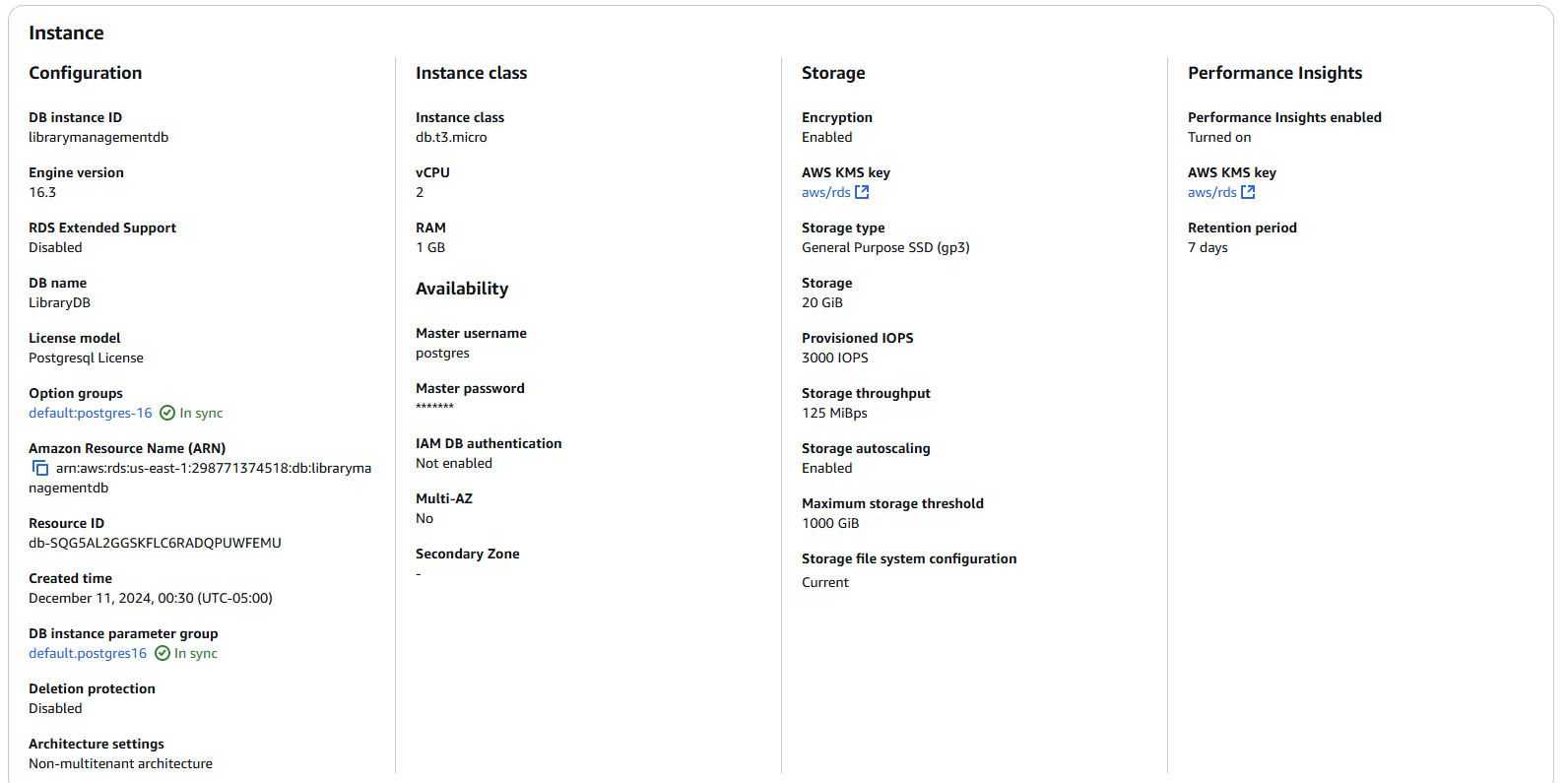
## **Database Deployment**

### **Amazon RDS Deployment - PostgreSQL**

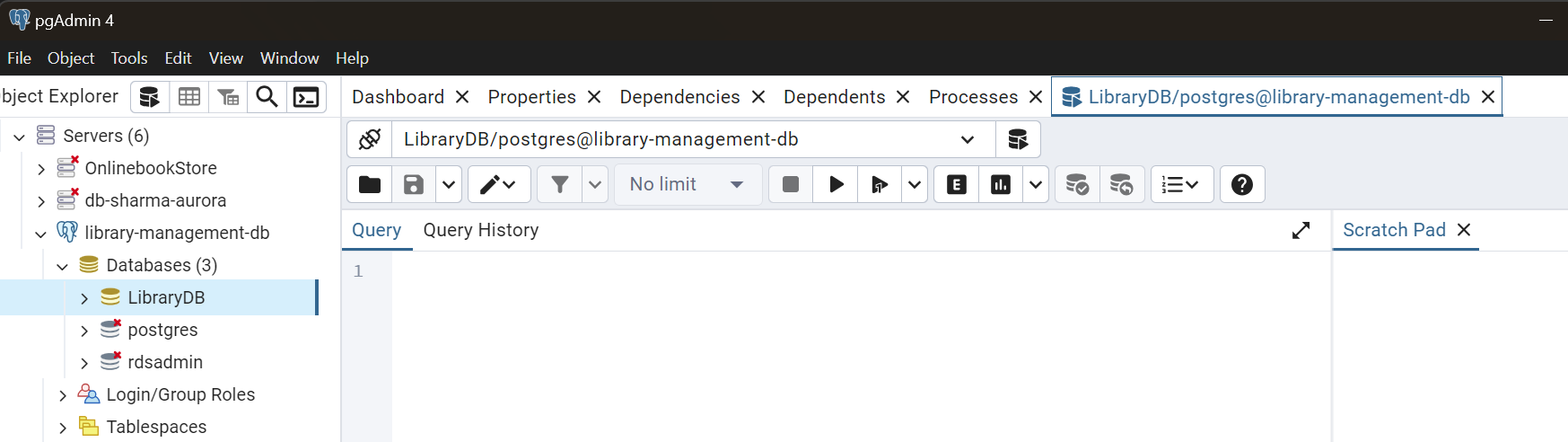
**Create DB**

****

**Configuration**

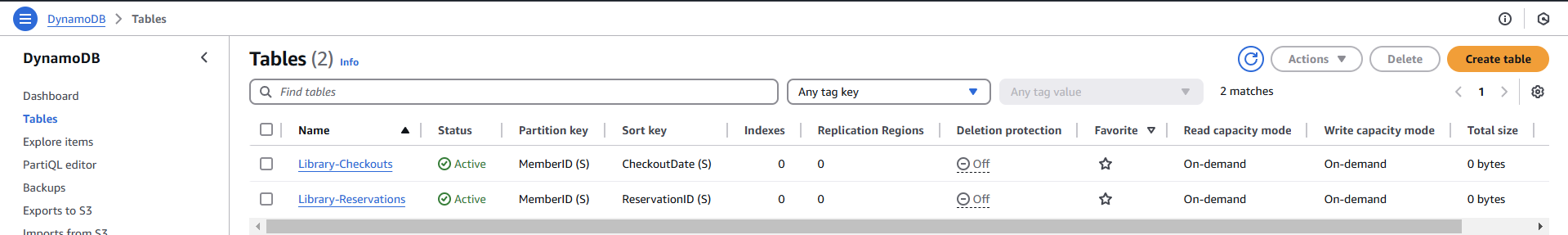
****

**RDS accessed using pgAdmin**

****

### **DynamoDB Deployment**

**Tables Created**

****

**Sample Record - Reservation**

**{**

**"MemberID": "M123",**

**"ReservationID": "R001",**

**"ItemID": "I456",**

**"ReservationDate": "2024-12-11"**

**}**

**Sample Record - Checkout**

**{**

**"MemberID": "M123",**

**"CheckoutDate": "2024-12-01",**

**"ItemID": "I456",**

**"DueDate": "2024-12-15",**

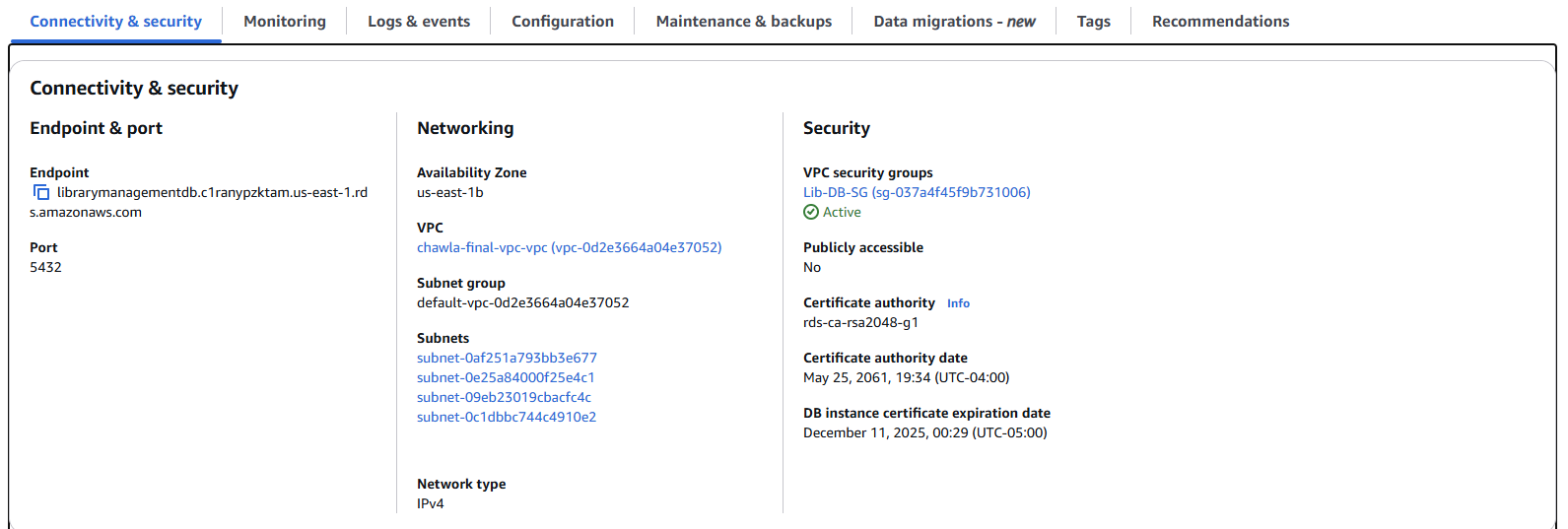
**"Title": "The Great Gatsby"**

**}**

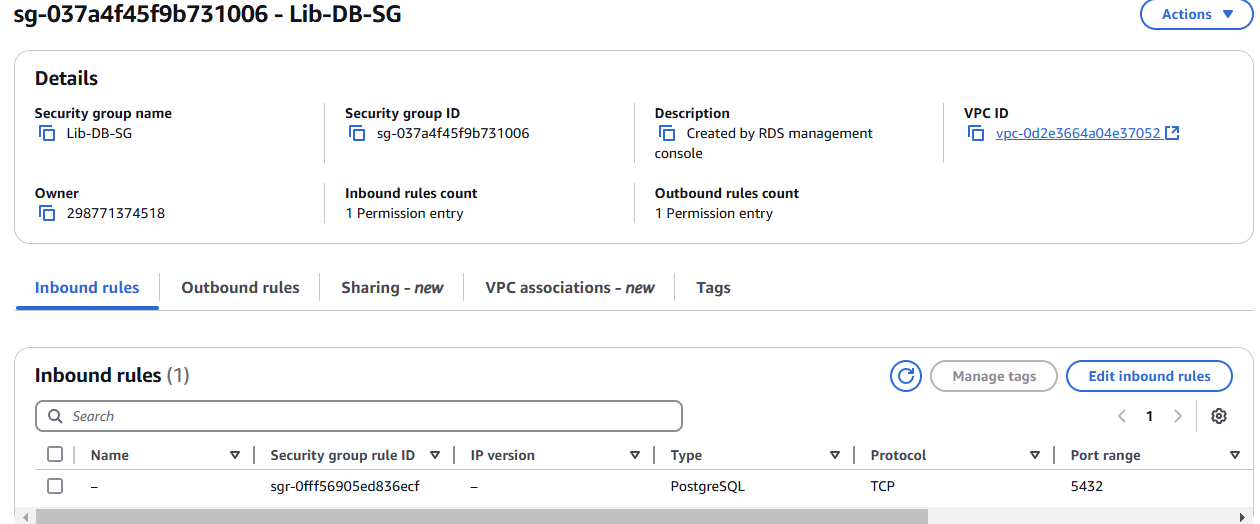
## **Security and Backup Management**

### **Amazon RDS Settings**

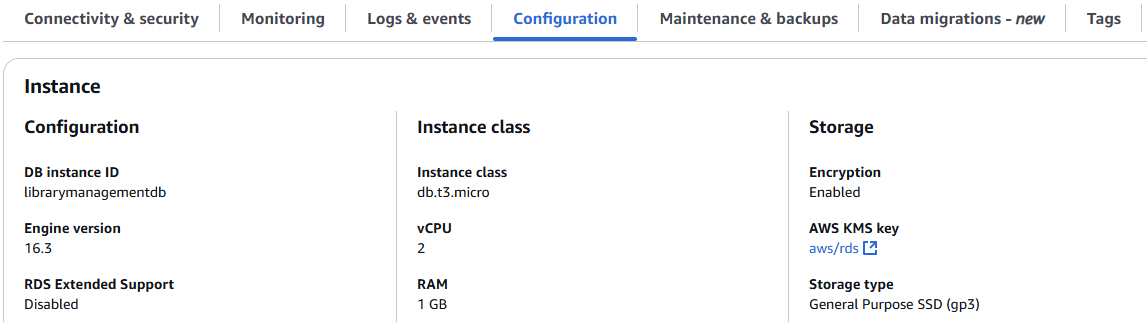
**Public Access – Not Allowed**

****

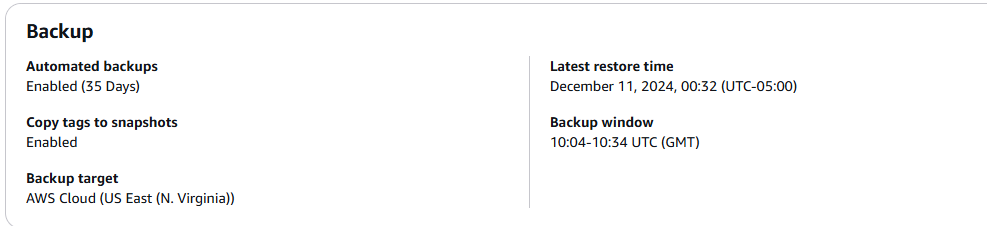
**Database Security Group – PostgreSQL access only to Bastion Host for SSH Tunnelling**

****

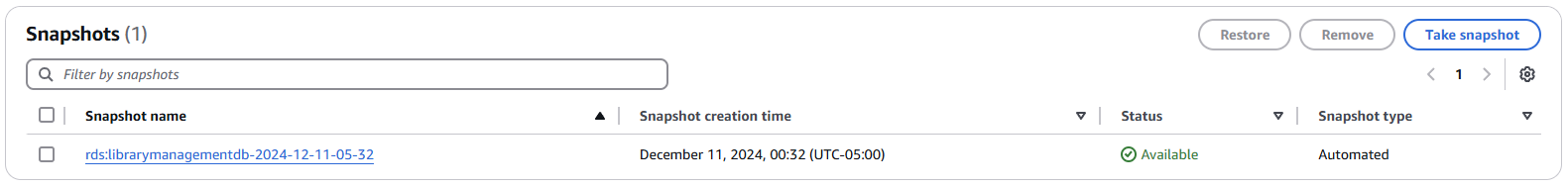
**Encryption - Enabled**

****

**Automated Backup Retention – 35 days**

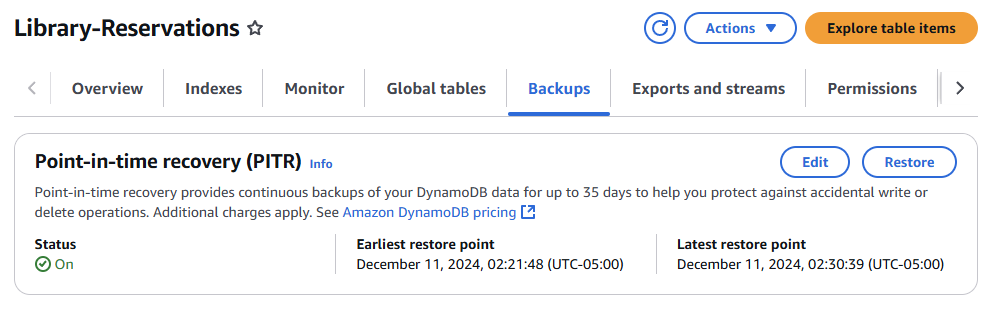
****

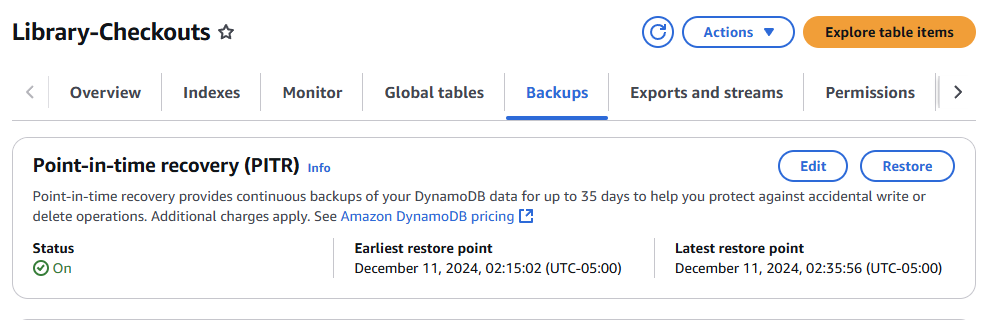
**Automated Snapshots**

****

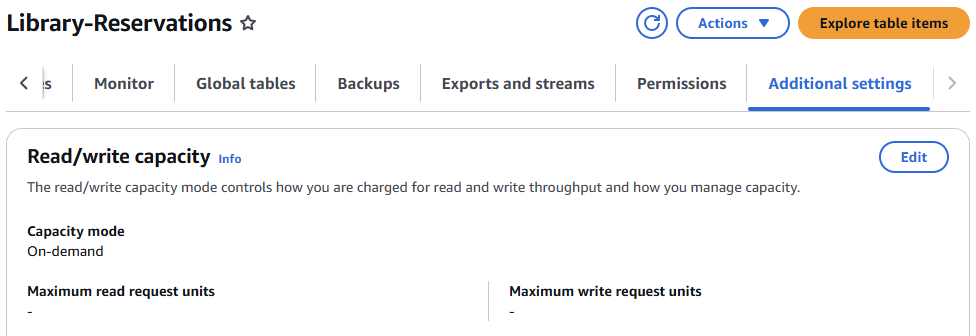
### **DynamoDB Settings**

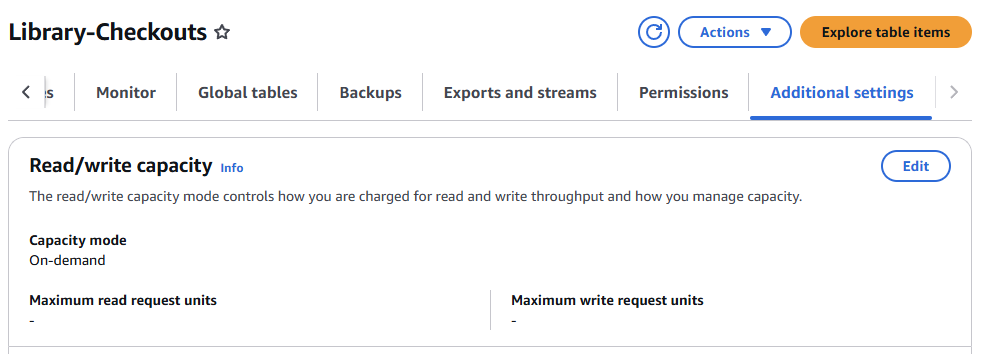
**Point in Time Recovery – Enabled**

****

****

**Capacity Mode**

****

****

### **Rationale**

#### **RDS Settings**

**Publicly Accessible: No**

By making it not publicly accessible, the database will not be directly exposed to the internet to avoid possible unauthorized access or attacks. It is only accessible by a specified IP or within a VPC for added security.

**DB Security Group: SSH tunnel**

Configure the DB security group to allow SSH tunneling for secure, encrypted access to the database via a bastion host. This will provide a controlled and monitored access without exposing the database to the wider network.

**Encryption: Yes**

By enabling encryption, the data at rest will be kept secure. AWS KMS will manage encryption keys, keeping sensitive information like user data and library records safe.

**Automated Backup Retention: 35 days**

Retaining automated backups for 35 days allows recovery from accidental deletions or corruption within a generous time frame. It supports compliance with data retention policies and ensures business continuity.

**Automated Snapshots**

Automated snapshots capture the database state at regular intervals, providing additional backup redundancy. Snapshots facilitate quick recovery in case of failure and allow point-in-time restoration of the database.

#### **DynamoDB Settings**

**Point-in-Time Recovery (PITR): Enabled**

Enabling PITR ensures restoration to any specific point in the last 35 days, thus minimizing the accidental data loss or corruption of a database, which then serves as a robust choice for holding business-critical data in terms of reservations and checkouts.

**Capacity Mode: On-Demand**

On-demand capacity mode automatically adjusts to workload demands, eliminating the need to manually manage throughput. It is cost-effective for unpredictable workloads, such as library transactions, which can have varying peaks (e.g., during membership renewals or holiday seasons).

These settings are designed to strike a balance between security, data availability, and scalability, ensuring the library management system is robust, reliable, and compliant with best practices for cloud-based deployments.