***Museum, Final Task description***

The business in question appears to be a museum management system. The primary goal of this system is to manage various entities such as items (artifacts, paintings, etc.), exhibitions, tickets, visitors, employees, storage facilities, and transactions related to the museum's operations. This system facilitates the tracking of exhibitions, the acquisition and management of items, ticketing, visitor information, employee records, and the management of storage facilities for the items.

* Step-by-Step Approach to Modeling:

I began by creating a database (museum\_db) and a dedicated schema (museum\_schema) within that database. The schema will hold all of our tables and functions, ensuring organization and proper separation of concerns.

CREATE DATABASE museum\_db;

CREATE SCHEMA IF NOT EXISTS museum\_schema;

SET search\_path TO museum\_schema;

* Define Entities and Relationships:

I defined the various entities involved in the museum system by creating tables for them. This includes items, exhibitions, visitors, employees, departments, storage facilities, and transactions.

1. Item Table: Contains details about each item (e.g., paintings, artifacts) in the museum.
2. Storage Facility Table: Defines the storage locations for items, tracking capacity and current occupancy.
3. Exhibition Table: Tracks exhibitions in the museum, including their start and end dates.
4. Ticket Table: Stores information about ticket purchases by visitors.
5. Visitor Table: Holds data about the museum's visitors.
6. Item-Exhibition Relationship (Many-to-Many): Items can be part of multiple exhibitions, and exhibitions can display multiple items.
7. Department and Employee Tables: Represents museum departments (manager, security…) and their employees.

Example for the Item table:

CREATE TABLE museum\_schema.item (

item\_id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

type VARCHAR(100) NOT NULL,

acquisition\_date DATE NOT NULL,

storage\_location\_id INT REFERENCES museum\_schema.storage\_facility(storage\_location\_id)

);

* Data integrity and constraints:

To ensure data integrity, constraints are applied:

1. Check constraints for ensuring valid values (non-negative prices, valid dates…).
2. Foreign keys to ensure relationships between tables are maintained.

Example of constraints:

ALTER TABLE museum\_schema.item

ADD CONSTRAINT chk\_item\_acquisition\_date CHECK (acquisition\_date > DATE '2024-01-01');

* Sample Data Insertion:

Sample data is inserted into the tables to simulate actual museum data. This helps test the schema.

INSERT INTO museum\_schema.item (name, acquisition\_date, description, type)

VALUES

('Mona Lisa', CURRENT\_DATE - 120, 'Famous painting by Leonardo da Vinci', 'Painting'),

('Ancient Vase', CURRENT\_DATE - 80, 'Clay vase from ancient Greece', 'Artifact');

* Dynamic Update Function:

A function was created to update the description of an item dynamically, where the column and value are passed as parameters.

CREATE OR REPLACE FUNCTION museum\_schema.update\_item\_data(

p\_item\_id INT,

p\_column\_name VARCHAR,

p\_new\_value VARCHAR

)

RETURNS VOID AS

$$

-- Function code

$$ LANGUAGE plpgsql;

* Transaction Management:

A function was created to add transactions to a museum\_transaction table, where each transaction represents an operation on a particular item (purchase or sale).

CREATE OR REPLACE FUNCTION museum\_schema.add\_transaction(

p\_item\_id INT,

p\_transaction\_type VARCHAR,

p\_quantity INT,

p\_total\_price DECIMAL(10, 2)

)

RETURNS VOID AS

$$

-- Function code

$$ LANGUAGE plpgsql;

* Recent Analytics View:

A view was created to show analytics for the most recent quarter. This helps the museum analyze recent activities related to items and exhibitions within a specific time period.

CREATE OR REPLACE VIEW museum\_schema.recent\_quarter\_analytics AS

WITH quarter\_dates AS (

SELECT

date\_trunc('quarter', CURRENT\_DATE) AS quarter\_start\_date,

date\_trunc('quarter', CURRENT\_DATE) + INTERVAL '3 months' - INTERVAL '1 day' AS quarter\_end\_date

),

SELECT

ri.item\_name, ri.item\_description, ri.item\_type, ri.item\_acquisition\_date,

ri.storage\_location, re.exhibition\_name, re.exhibition\_description,

re.exhibition\_start\_date, re.exhibition\_end\_date

FROM recent\_items ri

LEFT JOIN recent\_exhibitions re ON ri.item\_acquisition\_date <= re.exhibition\_start\_date

ORDER BY ri.item\_acquisition\_date DESC;

* Read-Only Manager Role:

A read-only role was created for the manager to grant SELECT privileges across all tables in the schema, allowing them to view data without modifying it.

CREATE ROLE manager\_read\_only LOGIN NOINHERIT;

GRANT USAGE ON SCHEMA museum\_schema TO manager\_read\_only;

GRANT SELECT ON ALL TABLES IN SCHEMA museum\_schema TO manager\_read\_only;

ALTER DEFAULT PRIVILEGES IN SCHEMA museum\_schema GRANT SELECT ON TABLES TO manager\_read\_only;

* Conclusion:

The modeling approach for the museum database system revolves around organizing the museum's operations in a relational structure, ensuring data integrity, and providing easy access to data analytics and management functionality. I have designed tables, relationships, functions, and views to meet the operational needs of the museum, all while ensuring flexibility and extensibility. The implementation also includes best practices for database security, such as the creation of read-only roles.