**Lab Exercises on Complex Data Types in MySQL**

**Exercise 1: Create a Table with ENUM**

**Task:**  
Create a table employees with a column role that only allows values: 'Manager', 'Developer', 'Analyst', 'Tester'.

**Solution:**

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

role ENUM('Manager', 'Developer', 'Analyst', 'Tester')

);

**Exercise 2: Insert Data into ENUM Column**

**Task:**  
Insert an employee named 'Alice' with the role 'Developer'.

**Solution:**

INSERT INTO employees (name, role) VALUES ('Alice', 'Developer');

**Exercise 3: Query ENUM Values**

**Task:**  
Retrieve all employees whose role is 'Manager'.

**Solution:**

SELECT \* FROM employees WHERE role = 'Manager';

**Exercise 4: Create a Table with SET**

**Task:**  
Create a table projects where the column technologies allows multiple values from 'Python', 'Java', 'SQL', 'C++'.

**Solution:**

CREATE TABLE projects (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

technologies SET('Python', 'Java', 'SQL', 'C++')

);

**Exercise 5: Insert Data into SET Column**

**Task:**  
Insert a project that uses both 'Python' and 'SQL'.

**Solution:**

INSERT INTO projects (name, technologies) VALUES ('Data Analysis', 'Python,SQL');

**Exercise 6: Query a Specific SET Value**

**Task:**  
Find all projects that use 'Java'.

**Solution:**

SELECT \* FROM projects WHERE FIND\_IN\_SET('Java', technologies);

**Exercise 7: Create a Table with JSON Column**

**Task:**  
Create a table orders with a details column storing JSON data.

**Solution:**

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100),

details JSON

);

**Exercise 8: Insert JSON Data**

**Task:**  
Insert an order for 'John Doe' with a JSON structure { "item": "Laptop", "price": 1200 }.

**Solution:**

INSERT INTO orders (customer\_name, details) VALUES

('John Doe', '{"item": "Laptop", "price": 1200}');

**Exercise 9: Extract Data from JSON**

**Task:**  
Retrieve the price of the item in each order.

**Solution:**

SELECT details->'$.price' AS price FROM orders;

**Exercise 10: Update JSON Data**

**Task:**  
Update the price of the 'Laptop' in 'John Doe's' order to 1300.

**Solution:**

UPDATE orders

SET details = JSON\_SET(details, '$.price', 1300)

WHERE customer\_name = 'John Doe';

**Exercise 11: Add a New Key to JSON**

**Task:**  
Add a new key "discount": 10 to all orders.

**Solution:**

UPDATE orders

SET details = JSON\_SET(details, '$.discount', 10);

**Exercise 12: Create a Table with XML Column**

**Task:**  
Create a table books with a column metadata that stores XML data.

**Solution:**

CREATE TABLE books (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

metadata XML

);

**Exercise 13: Insert XML Data**

**Task:**  
Insert an XML entry for a book with <author>Mark Twain</author>.

**Solution:**

INSERT INTO books (title, metadata)

VALUES ('The Adventures of Tom Sawyer', '<book><author>Mark Twain</author></book>');

**Exercise 14: Retrieve XML Data**

**Task:**  
Retrieve only the author field from the XML data.

**Solution:**

SELECT ExtractValue(metadata, '/book/author') AS author FROM books;

**Exercise 15: Create a Table with Spatial Data**

**Task:**  
Create a table locations with a coordinates column storing geographical points.

**Solution:**

CREATE TABLE locations (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

coordinates POINT NOT NULL,

SPATIAL INDEX(coordinates)

);

**Exercise 16: Insert Spatial Data**

**Task:**  
Insert a location named 'Central Park' with coordinates (40.785091, -73.968285).

**Solution:**

INSERT INTO locations (name, coordinates)

VALUES ('Central Park', ST\_GeomFromText('POINT(40.785091 -73.968285)'));

**Exercise 17: Find Locations Within a Distance**

**Task:**  
Find all locations within 10 km of a given point (40.785091, -73.968285).

**Solution:**

SELECT name FROM locations

WHERE ST\_Distance\_Sphere(coordinates, ST\_GeomFromText('POINT(40.785091 -73.968285)')) < 10000;

**Exercise 18: Create a Table with an Array in JSON**

**Task:**  
Modify the orders table to store multiple items in an array inside the JSON column.

**Solution:**

INSERT INTO orders (customer\_name, details) VALUES

('Jane Doe', '{"items": [{"name": "Tablet", "price": 300}, {"name": "Mouse", "price": 50}]}');

**Exercise 19: Retrieve Items from JSON Array**

**Task:**  
Retrieve the first item name from each order’s JSON array.

**Solution:**

SELECT JSON\_UNQUOTE(JSON\_EXTRACT(details, '$.items[0].name')) AS first\_item FROM orders;

**Exercise 20: Count JSON Array Elements**

**Task:**  
Count the number of items in each order.

**Solution:**

SELECT JSON\_LENGTH(details->'$.items') AS item\_count FROM orders;