1. Book Table, Queries, Insert, and Alteration (Analogy Included)

Code:

SQL

```sql
CREATE TABLE Book (
  S_No INT PRIMARY KEY,
  B_Name VARCHAR(255),
  Author VARCHAR(255),
  Price INT,
  Publisher VARCHAR(255)
);

-- Insert sample data (same as provided)
INSERT INTO Book (S_No, B_Name, Author, Price, Publisher)
VALUES (1, 'DBMS', 'Seema Kedhar', 250, 'Charulatha'),
       (2, 'TOC', 'John Martin', 400, 'Tata McGraw Hill'),
       (3, 'C Programming', 'E. Balagurusamy', 300, 'Technical');

-- Select all books
SELECT * FROM Book;

-- Select books priced above 300
SELECT * FROM Book WHERE Price > 300;

-- Insert a new book
INSERT INTO Book (S_No, B_Name, Author, Price, Publisher)
VALUES (4, 'Data Structures', 'Alfred V. Aho', 500, 'Addison-Wesley');

-- Drop the Author column (**Caution: This modification is
permanent!**)
ALTER TABLE Book DROP COLUMN Author;
```

Analogy:

Imagine a library. You're creating a table (Book) to organize book information like serial number (S_No), book name (B_Name), price, and publisher.

- **Creating the Table:** This is like designing a bookshelf with designated sections for each piece of information.
- **Selecting All Books:** This is like browsing the entire bookshelf and getting information about all the books.
- **Selecting Books Above 300:** This is like searching for specific books, in this case, those

priced higher than 300.
- **Inserting a New Book:** This is like adding a new book to the shelf and recording its details.
- **Dropping the Author Column:** This is like removing the author section from the bookshelf entirely (**Caution:** This is a permanent action in the database, similar to removing a section from a physical bookshelf).

2. Function for Customer Level based on Credit Limit (Analogy Included)

Code:

SQL

```
CREATE FUNCTION get_customer_level(credit_limit INT)
RETURNS VARCHAR(255)
BEGIN
  DECLARE level VARCHAR(255);

  IF credit_limit < 1000 THEN
    SET level = 'Low';
  ELSEIF credit_limit < 5000 THEN
    SET level = 'Medium';
  ELSE
    SET level = 'High';
  END IF;

  RETURN level;
END;

-- Example usage: Call the function with a credit limit of 2000
SELECT get_customer_level(2000);
```

Analogy:

Imagine a system that categorizes customer levels based on credit limits. This function acts like a decision-making tool.

- **credit_limit (parameter):** This represents the customer's credit limit amount.
- **level (variable):** This stores the customer level based on the IF conditions.
- **IF statements:** These define thresholds for credit limit levels (Low, Medium, High).
- **Function return:** This returns the determined customer level based on the credit limit.

3. Project Table, Group By Location, Order By Coordinators (Analogy Included)

Code:

SQL

```sql
CREATE TABLE Project (
  S_No INT PRIMARY KEY,
  P_Name VARCHAR(255),
  Coordinators VARCHAR(255),
  P_Price DECIMAL(10,2),
  Location VARCHAR(255)
);

-- Insert sample data (assuming you have project data)
-- ... Insert statements for your projects ...

-- Select projects grouped by location
SELECT Location, COUNT(*) AS project_count
FROM Project
GROUP BY Location;

-- Select projects ordered by coordinators
SELECT * FROM Project
ORDER BY Coordinators;
```

Analogy:

Imagine a project management system. You create a table named Project to store information like project ID (S_No), project name (P_Name), coordinator names, project budget (P_Price), and project location.

- **Creating the Table:** This is like setting up a project database to keep track of details.
- **Select with GROUP BY Location:** This is like grouping projects based on their location, similar to analyzing how many projects are happening in each location. Imagine creating a report that shows the number of projects happening in each city or region.
- **Select with ORDER BY Coordinators

Employee Table and Query with Analogy

Code:

SQL

```
CREATE TABLE Employee (
  S_No INT PRIMARY KEY,
  Name VARCHAR(255),
  Designation VARCHAR(255),
  Branch VARCHAR(255)
);

-- Insert sample data (assuming you have employee data)
-- ... Insert statements for your employees ...

-- Select managers working in Chennai
SELECT Name
FROM Employee
WHERE Designation = 'Manager' AND Branch = 'Chennai';
```

Analogy:

Imagine a company's Human Resources department. You're creating a table named `Employee` to store information like employee ID (S_No), name, designation (job title), and branch location.

- **Creating the Table:** This is like designing a database to record employee details.
- **Select Query:** This is like searching for specific employees based on criteria. In this case, you're looking for managers who work in the Chennai branch. Imagine filtering an employee list to find all managers stationed in the Chennai office.
    - **WHERE clause:** This defines the selection criteria. Here, you're specifying two conditions:
        - `Designation = 'Manager'`: This filters for employees whose

designation is 'Manager'.

    - `Branch = 'Chennai'`: This further filters for managers who are specifically working in the 'Chennai' branch.

- **Select Name:** This retrieves only the `Name` column for the matching employees. You can modify this to select other columns as needed.