Disaster Management System of India

Disaster management can be defined as the discipline of avoiding and dealing with risks. In other words, disaster management is a set of processes designed to be implemented before, during, and after disasters to prevent or mitigate their effects. This discipline involves preparing for disasters, responding to them, and finally supporting and rebuilding the society after initial disaster relief operations have ended. Because disasters pose a permanent threat, disaster management systems and practices should be continually monitored and improved.

The Disaster Management System (DMS) for India is designed to manage and track crucial disaster-related information, relief efforts, resources, and the people involved in disaster management. For this, we can create a database ‘Disaster’ consisting of the following tables:

1. **Disaster Table**

* **Purpose**: Tracks information about each disaster, including its type, severity, date of occurrence, and location. This helps the system keep a record of past and ongoing disasters, enabling better response planning and resource allocation.
* **Columns**:
  + **disaster\_id** (Primary Key): A unique identifier for each disaster.
  + **disaster\_type**: Specifies the type of disaster, such as Earthquake, Flood, or Cyclone. This classification helps in understanding the nature of the disaster.
  + **severity**: Indicates the severity level (Low, Medium, High) of the disaster. This helps prioritize response actions based on the severity.
  + **date\_occurred**: Records the date when the disaster occurred. It helps track historical events and calculate the duration of ongoing disasters.
  + **location\_id** (Foreign Key): Links to the Location table, identifying the specific area where the disaster occurred.

This table provides essential information for tracking and responding to disasters, helping in the quick identification of disaster type, location, and severity for coordinated response efforts.

**2. Location Table**

* **Purpose**: Stores details about the affected areas, including state, district, and city information. This geographic data is essential for disaster response teams to know exactly where relief is needed.
* **Columns**:
  + **location\_id** (Primary Key): Unique identifier for each location.
  + **state**: The state where the disaster has occurred. This is crucial for high-level geographic categorization.
  + **district**: The district affected within the state. This provides more detailed information on the disaster's location.
  + **city**: The specific city impacted by the disaster, helping pinpoint the exact area of focus.

This information helps responders quickly identify and categorize areas affected by disasters, enabling efficient and targeted disaster response. This data can be used to route resources and relief efforts to the precise locations most in need.

**3. Relief\_Center Table**

* **Purpose**: Manages information on relief centers, where people affected by disasters can receive aid. The data here assists in coordinating shelter and relief distribution.
* **Columns**:
  + **center\_id** (Primary Key): Unique identifier for each relief center.
  + **location\_id** (Foreign Key): Links to the **Location** table, specifying where the relief center is located.
  + **capacity**: Indicates the maximum number of people the relief center can accommodate. This helps in allocating resources and determining how many people can be sheltered.
  + **contact\_person**: Name of the contact person responsible for the relief center. Essential for coordinating with on-site personnel.
  + **contact\_number**: Contact number for reaching the relief center’s contact person.

This provides a centralized database of all available relief centers, their locations, and their capacities. This information is critical in managing shelter availability and ensuring effective relief distribution.

**4. Resources Table**

* **Purpose**: Tracks the types and quantities of resources available at each relief center. It helps ensure that essential supplies like food, water, and medical aid are adequately stocked and can be distributed when needed.
* **Columns**:
  + **resource\_id** (Primary Key): Unique identifier for each resource entry.
  + **resource\_type**: Specifies the type of resource (e.g., Food, Medicine).
  + **quantity**: Indicates the amount of each resource type available. This helps in inventory management and resource allocation.
  + **center\_id** (Foreign Key): Links to the **Relief\_Center** table, identifying where the resources are stored.

This facilitates efficient resource management by tracking available supplies across relief centres. The data helps disaster management teams monitor inventory and organize resource restocking as needed.

**5. Volunteers Table**

* **Purpose**: Contains information about volunteers, including contact details and the relief center they’re assigned to. This helps coordinate volunteer efforts across multiple locations.
* **Columns**:
  + **volunteer\_id** (Primary Key): Unique identifier for each volunteer.
  + **name**: Name of the volunteer, helping in personal identification.
  + **contact\_number**: Contact information for the volunteer.
  + **center\_id** (Foreign Key): Links to the **Relief\_Center** table, indicating where the volunteer is stationed.

This table tracks volunteer participation and assignments, ensuring sufficient manpower for disaster response and relief activities. The table helps management assign and reallocate volunteers as needed.

**6. Relief\_Operation Table**

* **Purpose**: Logs specific relief operations conducted for each disaster, including start and end dates, and the relief center involved. This helps monitor the progress and effectiveness of ongoing operations.
* **Columns**:
  + **operation\_id** (Primary Key): Unique identifier for each relief operation.
  + **disaster\_id** (Foreign Key): Links to the **Disaster** table, identifying the disaster associated with the operation.
  + **center\_id** (Foreign Key): Links to the **Relief\_Center** table, specifying where the operation is based.
  + **operation\_details**: Describes the relief operation, including key activities and goals.
  + **date\_started**: Start date of the operation.
  + **date\_ended**: End date of the operation, helping in calculating the duration of the relief operation.

It provides a detailed record of relief efforts and their associated disasters. This table is vital for assessing the scope and impact of each relief operation and helps in future planning for similar disaster responses.

**7. Donations Table**

* **Purpose**: Tracks donations received for disaster relief, including details about the donor and the amount donated. This information is important for managing funds and ensuring accountability.
* **Columns**:
  + **donation\_id** (Primary Key): Unique identifier for each donation record.
  + **amount**: Amount of money donated, essential for tracking the financial resources available for disaster relief.
  + **donor\_name**: Name of the donor, useful for record-keeping and potential future contact.
  + **donor\_contact**: Contact information for the donor, helping with communication and transparency.
  + **disaster\_id** (Foreign Key): Links to the **Disaster** table, indicating the specific disaster the donation is meant to support.

This helps the system keep track of funds collected for disaster relief and ties them to specific disasters. This ensures that donations are used effectively and can be accounted for in financial reporting.

These tables can be constructed in SQL using DDL commands as shown below:

**SYNTAX:**

mysql> create database disaster;

Query OK, 1 row affected (0.06 sec)

mysql> use disaster;

Database changed

mysql> CREATE TABLE Location (

-> location\_id INT PRIMARY KEY,

-> state VARCHAR(50),

-> district VARCHAR(50),

-> city VARCHAR(50)

-> );

Query OK, 0 rows affected (0.19 sec)

mysql> desc location;

+-------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------+-------------+------+-----+---------+-------+

| location\_id | int | NO | PRI | NULL | |

| state | varchar(50) | YES | | NULL | |

| district | varchar(50) | YES | | NULL | |

| city | varchar(50) | YES | | NULL | |

+-------------+-------------+------+-----+---------+-------+

4 rows in set (0.04 sec)

mysql> CREATE TABLE Disaster (

-> disaster\_id INT PRIMARY KEY,

-> disaster\_type VARCHAR(50),

-> severity VARCHAR(20),

-> date\_occurred DATE,

-> location\_id INT,

-> FOREIGN KEY (location\_id) REFERENCES Location(location\_id)

-> );

Query OK, 0 rows affected (0.09 sec)

mysql> desc disaster;

+---------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------------+-------------+------+-----+---------+-------+

| disaster\_id | int | NO | PRI | NULL | |

| disaster\_type | varchar(50) | YES | | NULL | |

| severity | varchar(20) | YES | | NULL | |

| date\_occurred | date | YES | | NULL | |

| location\_id | int | YES | MUL | NULL | |

+---------------+-------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

mysql> CREATE TABLE Relief\_Center (

-> center\_id INT PRIMARY KEY,

-> location\_id INT,

-> capacity INT,

-> contact\_person VARCHAR(50),

-> contact\_number VARCHAR(15),

-> FOREIGN KEY (location\_id) REFERENCES Location(location\_id)

-> );

Query OK, 0 rows affected (0.07 sec)

mysql> desc relief\_center;

+----------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+-------+

| center\_id | int | NO | PRI | NULL | |

| location\_id | int | YES | MUL | NULL | |

| capacity | int | YES | | NULL | |

| contact\_person | varchar(50) | YES | | NULL | |

| contact\_number | varchar(15) | YES | | NULL | |

+----------------+-------------+------+-----+---------+-------+

5 rows in set (0.00 sec)

mysql> CREATE TABLE Resources (

-> resource\_id INT PRIMARY KEY,

-> resource\_type VARCHAR(50),

-> quantity INT,

-> center\_id INT,

-> FOREIGN KEY (center\_id) REFERENCES Relief\_Center(center\_id)

-> );

Query OK, 0 rows affected (0.04 sec)

mysql> desc resources;

+---------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------------+-------------+------+-----+---------+-------+

| resource\_id | int | NO | PRI | NULL | |

| resource\_type | varchar(50) | YES | | NULL | |

| quantity | int | YES | | NULL | |

| center\_id | int | YES | MUL | NULL | |

+---------------+-------------+------+-----+---------+-------+

4 rows in set (0.00 sec)

mysql> CREATE TABLE Volunteers (

-> volunteer\_id INT PRIMARY KEY,

-> name VARCHAR(50),

-> contact\_number VARCHAR(15),

-> center\_id INT,

-> FOREIGN KEY (center\_id) REFERENCES Relief\_Center(center\_id)

-> );

Query OK, 0 rows affected (0.06 sec)

mysql> desc volunteers;

+----------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+-------+

| volunteer\_id | int | NO | PRI | NULL | |

| name | varchar(50) | YES | | NULL | |

| contact\_number | varchar(15) | YES | | NULL | |

| center\_id | int | YES | MUL | NULL | |

+----------------+-------------+------+-----+---------+-------+

4 rows in set (0.00 sec)

mysql> CREATE TABLE Relief\_Operation (

-> operation\_id INT PRIMARY KEY,

-> disaster\_id INT,

-> center\_id INT,

-> operation\_details TEXT,

-> date\_started DATE,

-> date\_ended DATE,

-> FOREIGN KEY (disaster\_id) REFERENCES Disaster(disaster\_id),

-> FOREIGN KEY (center\_id) REFERENCES Relief\_Center(center\_id)

-> );

Query OK, 0 rows affected (0.06 sec)

mysql> desc relief\_operation;

+-------------------+------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------------+------+------+-----+---------+-------+

| operation\_id | int | NO | PRI | NULL | |

| disaster\_id | int | YES | MUL | NULL | |

| center\_id | int | YES | MUL | NULL | |

| operation\_details | text | YES | | NULL | |

| date\_started | date | YES | | NULL | |

| date\_ended | date | YES | | NULL | |

+-------------------+------+------+-----+---------+-------+

6 rows in set (0.00 sec)

mysql> CREATE TABLE Donations (

-> donation\_id INT PRIMARY KEY,

-> amount DECIMAL(10, 2),

-> donor\_name VARCHAR(50),

-> donor\_contact VARCHAR(15),

-> disaster\_id INT,

-> FOREIGN KEY (disaster\_id) REFERENCES Disaster(disaster\_id)

-> );

Query OK, 0 rows affected (0.05 sec)

mysql> desc donations;

+---------------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------------+---------------+------+-----+---------+-------+

| donation\_id | int | NO | PRI | NULL | |

| amount | decimal(10,2) | YES | | NULL | |

| donor\_name | varchar(50) | YES | | NULL | |

| donor\_contact | varchar(15) | YES | | NULL | |

| disaster\_id | int | YES | MUL | NULL | |

+---------------+---------------+------+-----+---------+-------+

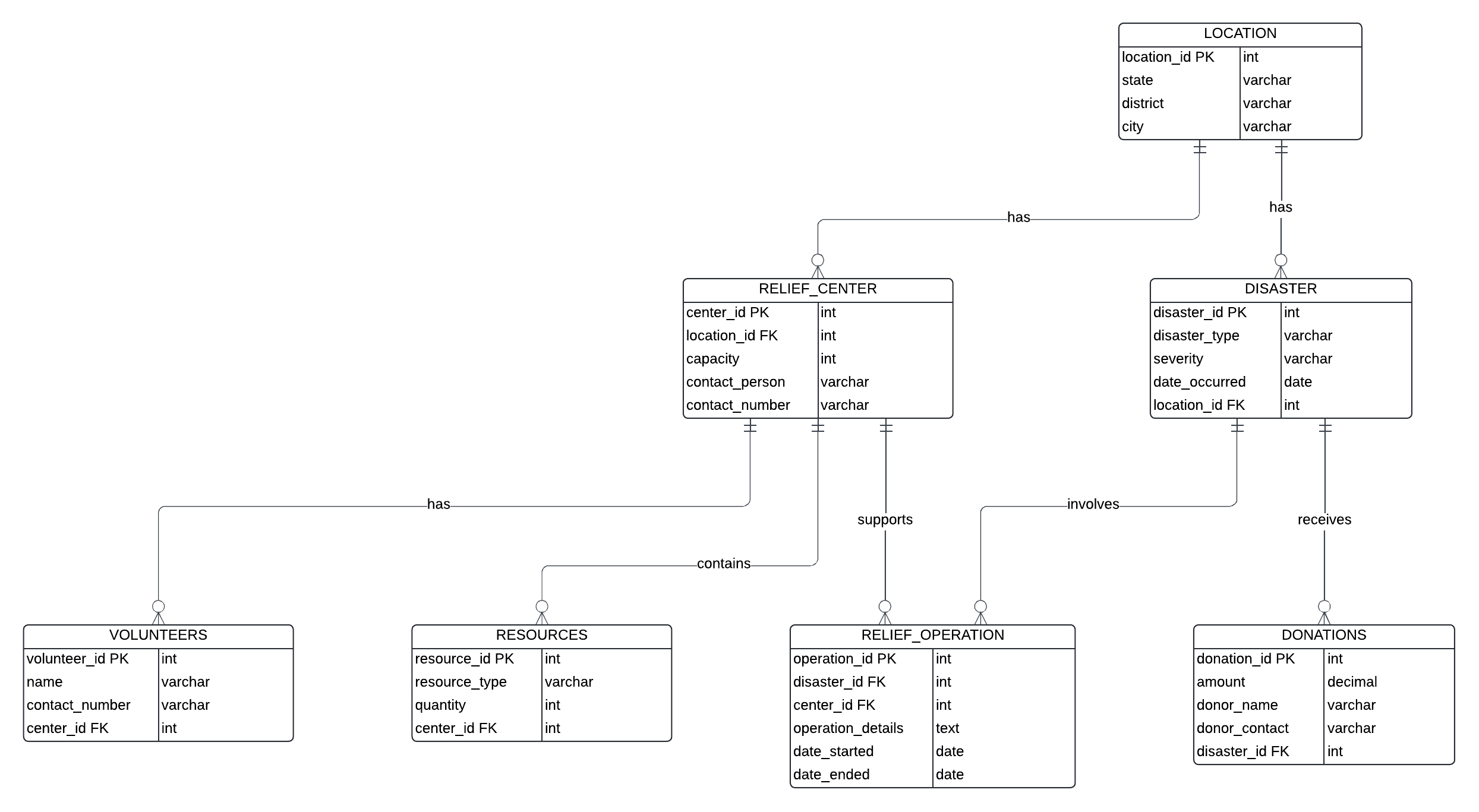
5 rows in set (0.01 sec)

Let's go through the DDL (Data Definition Language) commands we used to create the tables in the Disaster Management System database and how they function.

The CREATE TABLE command is used to define new tables and specify the structure for each table within the database. It includes defining the table's columns, data types, primary keys, and foreign keys.

A **primary key** uniquely identifies each record in a table, ensuring that each entry is distinct and cannot contain NULL values. It also improves the efficiency of data retrieval. Here, every \_id is a primary key.

A **foreign key** is used to link two tables together by establishing a relationship between them. It refers to the primary key of another table, ensuring that the data being referenced exists. Here, location\_id, center\_id and disaster\_id have been used as foreign key in more than 1 tables.

ER diagram of the given tables can be drawn as follows:

mysql> INSERT INTO Location (location\_id, state, district, city)

-> VALUES

-> (1, 'Maharashtra', 'Mumbai', 'Mumbai'),

-> (2, 'Bihar', 'Patna', 'Patna'),

-> (3, 'Odisha', 'Puri', 'Puri'),

-> (4, 'Assam', 'Kamrup', 'Guwahati');

-> (5, 'Uttar Pradesh', 'Varanasi', 'Varanasi'),

-> (6, 'West Bengal', 'Kolkata', 'Kolkata'),

-> (7, 'Tamil Nadu', 'Chennai', 'Chennai'),

-> (8, 'Kerala', 'Ernakulam', 'Kochi');

Query OK, 8 rows affected (0.00 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> select \* from location;

+-------------+---------------+-----------+----------+

| location\_id | state | district | city |

+-------------+---------------+-----------+----------+

| 1 | Maharashtra | Mumbai | Mumbai |

| 2 | Bihar | Patna | Patna |

| 3 | Odisha | Puri | Puri |

| 4 | Assam | Kamrup | Guwahati |

| 5 | Uttar Pradesh | Varanasi | Varanasi |

| 6 | West Bengal | Kolkata | Kolkata |

| 7 | Tamil Nadu | Chennai | Chennai |

| 8 | Kerala | Ernakulam | Kochi |

+-------------+---------------+-----------+----------+

8 rows in set (0.00 sec)

mysql> INSERT INTO Disaster (disaster\_id, disaster\_type, severity, date\_occurred, location\_id)

-> VALUES

-> (1, 'Flood', 'High', '2023-07-14', 1),

-> (2, 'Earthquake', 'Medium', '2021-05-26', 2),

-> (3, 'Cyclone', 'High', '2022-10-12', 3),

-> (4, 'Landslide', 'Low', '2020-08-22', 4);

-> (5, 'Flood', 'Medium', '2024-08-17', 5),

-> (6, 'Heat Wave', 'High', '2024-05-15', 6),

-> (7, 'Cyclone', 'High', '2023-11-20', 7),

-> (8, 'Flood', 'Low', '2023-07-25', 8),

-> (9, 'Earthquake', 'Low', '2024-01-10', 5);

Query OK, 9 rows affected (0.01 sec)

Records: 9 Duplicates: 0 Warnings: 0

mysql> select \* from disaster;

+-------------+---------------+----------+---------------+-------------+

| disaster\_id | disaster\_type | severity | date\_occurred | location\_id |

+-------------+---------------+----------+---------------+-------------+

| 1 | Flood | High | 2023-07-14 | 1 |

| 2 | Earthquake | Medium | 2021-05-26 | 2 |

| 3 | Cyclone | High | 2022-10-12 | 3 |

| 4 | Landslide | Low | 2020-08-22 | 4 |

| 5 | Flood | Medium | 2024-08-17 | 5 |

| 6 | Heat Wave | High | 2024-05-15 | 6 |

| 7 | Cyclone | High | 2023-11-20 | 7 |

| 8 | Flood | Low | 2023-07-25 | 8 |

| 9 | Earthquake | Low | 2024-01-10 | 5 |

+-------------+---------------+----------+---------------+-------------+

9 rows in set (0.00 sec)

mysql> INSERT INTO Relief\_Center (center\_id, location\_id, capacity, contact\_person, contact\_number)

-> VALUES

-> (1, 1, 500, 'Rajesh Kumar', '9123456789'),

-> (2, 2, 300, 'Anita Singh', '9198765432'),

-> (3, 3, 200, 'Kiran Reddy', '9132156784'),

-> (4, 4, 100, 'Pradeep Das', '9145267890');

-> (5, 5, 450, 'Ravi Singh', '9198234567'),

-> (6, 6, 600, 'Maya Chakraborty', '9187265454'),

-> (7, 7, 800, 'Sundar Rajan', '9172384782'),

-> (8, 8, 200, 'Arjun Menon', '9145432123');

Query OK, 8 rows affected (0.01 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> select \* from relief\_center;

+-----------+-------------+----------+------------------+----------------+

| center\_id | location\_id | capacity | contact\_person | contact\_number |

+-----------+-------------+----------+------------------+----------------+

| 1 | 1 | 500 | Rajesh Kumar | 9123456789 |

| 2 | 2 | 300 | Anita Singh | 9198765432 |

| 3 | 3 | 200 | Kiran Reddy | 9132156784 |

| 4 | 4 | 100 | Pradeep Das | 9145267890 |

| 5 | 5 | 450 | Ravi Singh | 9198234567 |

| 6 | 6 | 600 | Maya Chakraborty | 9187265454 |

| 7 | 7 | 800 | Sundar Rajan | 9172384782 |

| 8 | 8 | 200 | Arjun Menon | 9145432123 |

+-----------+-------------+----------+------------------+----------------+

8 rows in set (0.00 sec)

mysql> INSERT INTO Resources (resource\_id, resource\_type, quantity, center\_id)

-> VALUES

-> (1, 'Food', 1000, 1),

-> (2, 'Medicine', 200, 1),

-> (3, 'Water', 1500, 2),

-> (4, 'Blankets', 300, 3),

-> (5, 'Clothing', 500, 4);

-> (6, 'Food', 1200, 5),

-> (7, 'Medicine', 250, 6),

-> (8, 'Water', 2000, 7),

-> (9, 'Medical Kits', 400, 8),

-> (10, 'Clothing', 300, 8),

-> (11, 'Tents', 100, 7);

Query OK, 11 rows affected (0.01 sec)

Records: 11 Duplicates: 0 Warnings: 0

mysql> select \* from resources;

+-------------+---------------+----------+-----------+

| resource\_id | resource\_type | quantity | center\_id |

+-------------+---------------+----------+-----------+

| 1 | Food | 1000 | 1 |

| 2 | Medicine | 200 | 1 |

| 3 | Water | 1500 | 2 |

| 4 | Blankets | 300 | 3 |

| 5 | Clothing | 500 | 4 |

| 6 | Food | 1200 | 5 |

| 7 | Medicine | 250 | 6 |

| 8 | Water | 2000 | 7 |

| 9 | Medical Kits | 400 | 8 |

| 10 | Clothing | 300 | 8 |

| 11 | Tents | 100 | 7 |

+-------------+---------------+----------+-----------+

11 rows in set (0.00 sec)

mysql> INSERT INTO Volunteers (volunteer\_id, name, contact\_number, center\_id)

-> VALUES

-> (1, 'Vikram Sharma', '9123456780', 1),

-> (2, 'Priya Patel', '9198765433', 2),

-> (3, 'Sunil Narang', '9132156785', 3),

-> (4, 'Meera Joshi', '9145267891', 4);

-> (5, 'Ashish Mehta', '9123345678', 5),

-> (6, 'Rina Paul', '9143567878', 6),

-> (7, 'Suresh Kumar', '9178234567', 7),

-> (8, 'Arathi Nair', '9182345671', 8),

-> (9, 'Shivam Yadav', '9132456789', 6),

-> (10, 'Pooja Gupta', '9198765433', 5);

Query OK, 10 rows affected (0.01 sec)

Records: 10 Duplicates: 0 Warnings: 0

mysql> select \* from volunteers;

+--------------+---------------+----------------+-----------+

| volunteer\_id | name | contact\_number | center\_id |

+--------------+---------------+----------------+-----------+

| 1 | Vikram Sharma | 9123456780 | 1 |

| 2 | Priya Patel | 9198765433 | 2 |

| 3 | Sunil Narang | 9132156785 | 3 |

| 4 | Meera Joshi | 9145267891 | 4 |

| 5 | Ashish Mehta | 9123345678 | 5 |

| 6 | Rina Paul | 9143567878 | 6 |

| 7 | Suresh Kumar | 9178234567 | 7 |

| 8 | Arathi Nair | 9182345671 | 8 |

| 9 | Shivam Yadav | 9132456789 | 6 |

| 10 | Pooja Gupta | 9198765433 | 5 |

+--------------+---------------+----------------+-----------+

10 rows in set (0.00 sec)

mysql> INSERT INTO Relief\_Operation (operation\_id, disaster\_id, center\_id, operation\_details, date\_started, date\_ended)

-> VALUES

-> (1, 1, 1, 'Distributed relief material to flood victims in Mumbai', '2023-07-15', '2023-07-20'),

-> (2, 2, 2, 'Evacuated residents affected by earthquake in Patna', '2021-05-27', '2021-05-28'),

-> (3, 3, 3, 'Provided emergency shelter and food post-cyclone in Puri', '2022-10-13', '2022-10-18'),

-> (4, 4, 4, 'Cleared debris from landslide in Guwahati', '2020-08-23', '2020-08-25');

-> (5, 5, 5, 'Flood relief and evacuation in Varanasi', '2024-08-18', '2024-08-22'),

-> (6, 6, 6, 'Heatwave mitigation in Kolkata', '2024-05-16', '2024-05-20'),

-> (7, 7, 7, 'Cyclone relief operations in Chennai', '2023-11-21', '2023-11-26'),

-> (8, 8, 8, 'Providing supplies and relief during flood in Kochi', '2023-07-26', '2023-07-28');

Query OK, 8 rows affected (0.01 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> select \* from relief\_operation;

+--------------+-------------+-----------+----------------------------------------------------------+--------------+------------+

| operation\_id | disaster\_id | center\_id | operation\_details

| date\_started | date\_ended |

+--------------+-------------+-----------+----------------------------------------------------------+--------------+------------+

| 1 | 1 | 1 | Distributed relief material to flood victims in Mumbai | 2023-07-15 | 2023-07-20 |

| 2 | 2 | 2 | Evacuated residents affected by earthquake in Patna | 2021-05-27 | 2021-05-28 |

| 3 | 3 | 3 | Provided emergency shelter and food post-cyclone in Puri | 2022-10-13 | 2022-10-18 |

| 4 | 4 | 4 | Cleared debris from landslide in Guwahati | 2020-08-23 | 2020-08-25 |

| 5 | 5 | 5 | Flood relief and evacuation in Varanasi | 2024-08-18 | 2024-08-22 |

| 6 | 6 | 6 | Heat wave mitigation in Kolkata | 2024-05-16 | 2024-05-20 |

| 7 | 7 | 7 | Cyclone relief operations in Chennai | 2023-11-21 | 2023-11-26 |

| 8 | 8 | 8 | Providing supplies and relief during flood in Kochi | 2023-07-26 | 2023-07-28 |

+--------------+-------------+-----------+----------------------------------------------------------+--------------+------------+

8 rows in set (0.00 sec)

INSERT INTO Donations (donation\_id, amount, donor\_name, donor\_contact, disaster\_id)

-> VALUES

-> (1, 50000.00, 'Ajay Gupta', '9123456781', 1),

-> (2, 75000.00, 'Nisha Kapoor', '9198765434', 2),

-> (3, 30000.00, 'Anil Mehta', '9132156786', 3),

-> (4, 20000.00, 'Suman Verma', '9145267892', 4);

-> (5, 45000.00, 'Bhavna Desai', '9123456785', 5),

-> (6, 55000.00, 'Rahul Narang', '9178234568', 6),

-> (7, 35000.00, 'Kavita Mishra', '9145267892', 7),

-> (8, 25000.00, 'Ajit Patil', '9123456789', 8);

Query OK, 8 rows affected (0.01 sec)

Records: 8 Duplicates: 0 Warnings: 0

mysql> select \* from donations;

+-------------+----------+---------------+---------------+-------------+

| donation\_id | amount | donor\_name | donor\_contact | disaster\_id |

+-------------+----------+---------------+---------------+-------------+

| 1 | 50000.00 | Ajay Gupta | 9123456781 | 1 |

| 2 | 75000.00 | Nisha Kapoor | 9198765434 | 2 |

| 3 | 30000.00 | Anil Mehta | 9132156786 | 3 |

| 4 | 20000.00 | Suman Verma | 9145267892 | 4 |

| 5 | 45000.00 | Bhavna Desai | 9123456785 | 5 |

| 6 | 55000.00 | Rahul Narang | 9178234568 | 6 |

| 7 | 35000.00 | Kavita Mishra | 9145267892 | 7 |

| 8 | 25000.00 | Ajit Patil | 9123456789 | 8 |

+-------------+----------+---------------+---------------+-------------+

8 rows in set (0.00 sec)

mysql> UPDATE Relief\_Center

-> SET contact\_number = '9199999999'

-> WHERE center\_id = 2;

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from relief\_center;

+-----------+-------------+----------+------------------+----------------+

| center\_id | location\_id | capacity | contact\_person | contact\_number |

+-----------+-------------+----------+------------------+----------------+

| 1 | 1 | 500 | Rajesh Kumar | 9123456789 |

| 2 | 2 | 300 | Anita Singh | 9199999999 |

| 3 | 3 | 200 | Kiran Reddy | 9132156784 |

| 4 | 4 | 100 | Pradeep Das | 9145267890 |

| 5 | 5 | 450 | Ravi Singh | 9198234567 |

| 6 | 6 | 600 | Maya Chakraborty | 9187265454 |

| 7 | 7 | 800 | Sundar Rajan | 9172384782 |

| 8 | 8 | 200 | Arjun Menon | 9145432123 |

+-----------+-------------+----------+------------------+----------------+

8 rows in set (0.00 sec)

mysql> DELETE FROM Volunteers

-> WHERE volunteer\_id= 10;

Query OK, 1 row affected (0.00 sec)

mysql> select \* from volunteers;

+--------------+---------------+----------------+-----------+

| volunteer\_id | name | contact\_number | center\_id |

+--------------+---------------+----------------+-----------+

| 1 | Vikram Sharma | 9123456780 | 1 |

| 2 | Priya Patel | 9198765433 | 2 |

| 3 | Sunil Narang | 9132156785 | 3 |

| 4 | Meera Joshi | 9145267891 | 4 |

| 5 | Ashish Mehta | 9123345678 | 5 |

| 7 | Suresh Kumar | 9178234567 | 7 |

| 8 | Arathi Nair | 9182345671 | 8 |

| 9 | Shivam Yadav | 9132456789 | 6 |

+--------------+---------------+----------------+-----------+

8 rows in set (0.00 sec)

DML Commands used in the table- Insert, Select, Delete, Update:

The INSERT command is used to add new rows of data to a table. All the entries to each table are done using the INSERT command.

The UPDATE command modifies existing data in a table based on specific conditions. Here, it changes the contact\_number for the Relief\_Center with center\_id 2.

The DELETE command removes rows from a table based on given conditions. In this case, it deletes the row in the Volunteers table where volunteer\_id is 10.

The SELECT command is used to retrieve data from one or more tables. Here, SELECT \* indicates that all columns from the specified table are selected and retrieved.

TCL(Transaction Control Language) commands are used to manage transactions in SQL. These commands, which include BEGIN, COMMIT, and ROLLBACK, help ensure data integrity, especially when multiple operations need to be executed as a single logical unit.

TCL commands can be executed in the given tables as follows:

* Imagine a scenario, where a new donation is received for flood relief efforts, and an update to the available resources at a specific relief center is required. If both actions cannot be completed successfully, the transaction should be rolled back.

mysql> BEGIN;

Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Donations (donation\_id, amount, donor\_name, donor\_contact, disaster\_id)

-> VALUES (9, 60000.00, 'Rajat Verma', '9123456782', 1);

Query OK, 1 row affected (0.01 sec)

mysql> UPDATE Resources

-> SET quantity = quantity + 300

-> WHERE resource\_id = 1 AND center\_id = 1;

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

-- Check if both operations are successful and either commit or rollback the transaction. Assuming there's a manual check here or an application-side validation

mysql> COMMIT;

Query OK, 0 rows affected (0.00 sec)

* Suppose an update is attempted on the Relief\_Operation table, but an error in the start and end dates is noticed during the update. This can be rolled back to maintain data accuracy.

mysql> BEGIN;

Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Relief\_Operation

-> SET date\_ended = '2024-07-22'

-> WHERE operation\_id = 5;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

-- Realize there's an error in the date (too far in the future for this operation) and roll back the transaction

mysql> ROLLBACK;

Query OK, 0 rows affected (0.00 sec)

Nested queries, or subqueries, are queries inside another SQL query. They are useful for extracting data based on results from a different query. Here is how we can do nested queries in the given tables:

* **Find Volunteers in Areas of Recent High Severity Disasters**

This query finds volunteers assigned to relief centers in locations affected by recent high-severity disasters.

mysql> SELECT name, contact\_number FROM Volunteers

-> WHERE center\_id IN (

-> SELECT center\_id FROM Relief\_Center WHERE location\_id IN (

-> SELECT location\_id FROM Disaster

-> WHERE severity = 'High' AND date\_occurred > '2023-01-01'));

+---------------+----------------+

| name | contact\_number |

+---------------+----------------+

| Vikram Sharma | 9123456780 |

| Shivam Yadav | 9132456789 |

| Suresh Kumar | 9178234567 |

+---------------+----------------+

3 rows in set (0.03 sec)

* **Get Disaster Types That Received Donations Greater Than Average**

This query lists disaster types that have received donations larger than the average donation amount.

mysql> SELECT disaster\_type FROM Disaster

-> WHERE disaster\_id IN (

-> SELECT disaster\_id FROM Donations

-> WHERE amount > (SELECT AVG(amount) FROM Donations)

-> );

+---------------+

| disaster\_type |

+---------------+

| Flood |

| Earthquake |

| Flood |

| Heat Wave |

+---------------+

4 rows in set (0.02 sec)

Set operations include UNION, INTERSECT, and EXCEPT (or MINUS), allowing you to combine or differentiate rows from two or more queries.

* **Union: Combine Resource and Volunteer Contact Information**

This query combines the contact details of both resource contacts and volunteers, listing them as a unified contact list for rapid communication.

mysql> SELECT contact\_person AS name, contact\_number

-> FROM Relief\_Center

-> UNION

-> SELECT name, contact\_number

-> FROM Volunteers;

+------------------+----------------+

| name | contact\_number |

+------------------+----------------+

| Rajesh Kumar | 9123456789 |

| Anita Singh | 9199999999 |

| Kiran Reddy | 9132156784 |

| Pradeep Das | 9145267890 |

| Ravi Singh | 9198234567 |

| Maya Chakraborty | 9187265454 |

| Sundar Rajan | 9172384782 |

| Arjun Menon | 9145432123 |

| Vikram Sharma | 9123456780 |

| Priya Patel | 9198765433 |

| Sunil Narang | 9132156785 |

| Meera Joshi | 9145267891 |

| Ashish Mehta | 9123345678 |

| Suresh Kumar | 9178234567 |

| Arathi Nair | 9182345671 |

| Shivam Yadav | 9132456789 |

+------------------+----------------+

16 rows in set (0.01 sec)

* **Intersect: Find Centers Both Having Volunteers and Donations**

This query lists relief centers that have both volunteers assigned and received donations.

(MySQL has not implemented these set operations as standalone commands in its SQL dialect. As a result, attempting to use them will lead to syntax errors in the MySQL command line client.

Instead of INTERSECT, you can use a **JOIN** or a **WHERE EXISTS** clause to achieve similar results.)

mysql> SELECT DISTINCT c.center\_id

-> FROM Volunteers v

-> JOIN Relief\_Center c ON v.center\_id = c.center\_id;

+-----------+

| center\_id |

+-----------+

| 1 |

| 2 |

| 3 |

| 4 |

| 5 |

| 6 |

| 7 |

| 8 |

+-----------+

8 rows in set (0.00 sec)

**String Functions**

Let’s now see how we can use string and numeric functions in the given tables

* **Concatenation of Relief Center Contact Information**

Create a combined string of the contact person's name and their phone number for easy reference.

mysql> SELECT CONCAT(contact\_person, ' (', contact\_number, ')') AS contact\_info

-> FROM Relief\_Center;

+-------------------------------+

| contact\_info |

+-------------------------------+

| Rajesh Kumar (9123456789) |

| Anita Singh (9199999999) |

| Kiran Reddy (9132156784) |

| Pradeep Das (9145267890) |

| Ravi Singh (9198234567) |

| Maya Chakraborty (9187265454) |

| Sundar Rajan (9172384782) |

| Arjun Menon (9145432123) |

+-------------------------------+

8 rows in set (0.01 sec)

* **Finding Volunteers with Specific Patterns in Names**

Retrieve volunteers whose names start with the letter "A".

mysql> SELECT name

-> FROM Volunteers

-> WHERE name LIKE 'A%';

+--------------+

| name |

+--------------+

| Ashish Mehta |

| Arathi Nair |

+--------------+

2 rows in set (0.00 sec)

* **Calculating Total Donations per Disaster**

Get the total amount of donations received for each disaster.

mysql> SELECT d.disaster\_id, SUM(amount) AS total\_donations

-> FROM Donations d

-> GROUP BY d.disaster\_id;

+-------------+-----------------+

| disaster\_id | total\_donations |

+-------------+-----------------+

| 1 | 110000.00 |

| 2 | 75000.00 |

| 3 | 30000.00 |

| 4 | 20000.00 |

| 5 | 45000.00 |

| 6 | 55000.00 |

| 7 | 35000.00 |

| 8 | 25000.00 |

+-------------+-----------------+

8 rows in set (0.00 sec)

* **Average Capacity of Relief Centers**

Calculate the average capacity of all relief centers.

mysql> SELECT AVG(capacity) AS average\_capacity

-> FROM Relief\_Center;

+------------------+

| average\_capacity |

+------------------+

| 393.7500 |

+------------------+

1 row in set (0.00 sec)

**Group by and Order by Clause**

The GROUP BY clause is used to arrange identical data into groups. It is often used with aggregate functions like SUM, COUNT, AVG, etc.

Example: Count the number of volunteers at each relief center.

mysql> SELECT center\_id, name, COUNT(volunteer\_id) AS total\_volunteers FROM

Volunteers GROUP BY center\_id;

+-----------+---------------+------------------+

| center\_id | name | total\_volunteers |

+-----------+---------------+------------------+

| 1 | Vikram Sharma | 1 |

| 2 | Priya Patel | 1 |

| 3 | Sunil Narang | 1 |

| 4 | Meera Joshi | 1 |

| 5 | Ashish Mehta | 1 |

| 6 | Shivam Yadav | 1 |

| 7 | Suresh Kumar | 1 |

| 8 | Arathi Nair | 1 |

+-----------+---------------+------------------+

8 rows in set (0.00 sec)

**ORDER BY Clause**

The ORDER BY clause is used to sort the result set in either ascending or descending order.

Example: Ordering Relief Centers by Capacity

mysql> SELECT center\_id, capacity

-> FROM Relief\_Center

-> ORDER BY capacity DESC;

+-----------+----------+

| center\_id | capacity |

+-----------+----------+

| 7 | 800 |

| 6 | 600 |

| 1 | 500 |

| 5 | 450 |

| 2 | 300 |

| 3 | 200 |

| 8 | 200 |

| 4 | 100 |

+-----------+----------+

8 rows in set (0.00 sec)

**DISTINCT Keyword**

The DISTINCT keyword is used to return unique values in the result set, eliminating duplicates.

Example: Unique Disaster Types

mysql> SELECT DISTINCT disaster\_type

-> FROM Disaster;

+---------------+

| disaster\_type |

+---------------+

| Flood |

| Earthquake |

| Cyclone |

| Landslide |

| Heat Wave |

+---------------+

5 rows in set (0.00 sec)

**LIMIT Keyword**

The LIMIT keyword is used to specify the number of records to return, which is particularly useful for pagination.

Example: Top 3 Relief Centers by Capacity

mysql> SELECT center\_id, capacity

-> FROM Relief\_Center

-> ORDER BY capacity DESC

-> LIMIT 3;

+-----------+----------+

| center\_id | capacity |

+-----------+----------+

| 7 | 800 |

| 6 | 600 |

| 1 | 500 |

+-----------+----------+

3 rows in set (0.00 sec)

**JOINS**

Using joins in the context of the Disaster Management System (DMS) allows us to combine data from multiple tables, providing deeper insights and understanding of the relationships between various entities involved in disaster management.

**Inner Join: Donations and Disaster**

Retrieve a list of donations along with the associated disaster details.

mysql> SELECT d.donation\_id, d.amount, d.donor\_name, di.disaster\_type, di.date\_occurred

-> FROM Donations d

-> INNER JOIN Disaster di ON d.disaster\_id = di.disaster\_id;

+-------------+----------+---------------+---------------+---------------+

| donation\_id | amount | donor\_name | disaster\_type | date\_occurred |

+-------------+----------+---------------+---------------+---------------+

| 1 | 50000.00 | Ajay Gupta | Flood | 2023-07-14 |

| 2 | 75000.00 | Nisha Kapoor | Earthquake | 2021-05-26 |

| 3 | 30000.00 | Anil Mehta | Cyclone | 2022-10-12 |

| 4 | 20000.00 | Suman Verma | Landslide | 2020-08-22 |

| 5 | 45000.00 | Bhavna Desai | Flood | 2024-08-17 |

| 6 | 55000.00 | Rahul Narang | Heat Wave | 2024-05-15 |

| 7 | 35000.00 | Kavita Mishra | Cyclone | 2023-11-20 |

| 8 | 25000.00 | Ajit Patil | Flood | 2023-07-25 |

| 9 | 60000.00 | Rajat Verma | Flood | 2023-07-14 |

+-------------+----------+---------------+---------------+---------------+

9 rows in set (0.00 sec)

**Left Join: Relief Centers and Volunteers**

List all relief centers and the number of volunteers assigned to each, including centers that currently have no volunteers.

mysql> SELECT rc.center\_id, rc.contact\_person, COUNT(v.volunteer\_id) AS total\_volunteers

-> FROM Relief\_Center rc

-> LEFT JOIN Volunteers v ON rc.center\_id = v.center\_id

-> GROUP BY rc.center\_id;

+-----------+------------------+------------------+

| center\_id | contact\_person | total\_volunteers |

+-----------+------------------+------------------+

| 1 | Rajesh Kumar | 1 |

| 2 | Anita Singh | 1 |

| 3 | Kiran Reddy | 1 |

| 4 | Pradeep Das | 1 |

| 5 | Ravi Singh | 1 |

| 6 | Maya Chakraborty | 1 |

| 7 | Sundar Rajan | 1 |

| 8 | Arjun Menon | 1 |

+-----------+------------------+------------------+

8 rows in set (0.00 sec)

**Right Join: Disasters and Relief Operations**

Get a list of all relief operations and the corresponding disaster details, ensuring that all relief operations are shown.

mysql> SELECT ro.operation\_id, ro.operation\_details, di.disaster\_type, di.severity

-> FROM Relief\_Operation ro

-> RIGHT JOIN Disaster di ON ro.disaster\_id = di.disaster\_id;

+--------------+----------------------------------------------------------+---------------+----------+

| operation\_id | operation\_details | disaster\_type | severity |

+--------------+----------------------------------------------------------+---------------+----------+

| 1 | Distributed relief material to flood victims in Mumbai | Flood | High |

| 2 | Evacuated residents affected by earthquake in Patna | Earthquake | Medium |

| 3 | Provided emergency shelter and food post-cyclone in Puri | Cyclone | High |

| 4 | Cleared debris from landslide in Guwahati | Landslide | Low |

| 5 | Flood relief and evacuation in Varanasi | Flood | Medium |

| 6 | Heat wave mitigation in Kolkata | Heat Wave | High |

| 7 | Cyclone relief operations in Chennai | Cyclone | High |

| 8 | Providing supplies and relief during flood in Kochi | Flood | Low |

| NULL | NULL | Earthquake | Low |

+--------------+----------------------------------------------------------+---------------+----------+

9 rows in set (0.00 sec)

**View Definition Language**

VDL(View Definition Language) is part of SQL that deals with creating, modifying, and managing views in a database. A **view** is essentially a virtual table that provides a way to present data from one or more tables in a specific format or structure without changing the actual data in the database. This can be particularly useful in a Disaster Management System (DMS) for summarizing, filtering, or presenting data relevant to stakeholders.

**Creating Views: View for Total Donations by Disaster Type**

Create a view that summarizes the total donations received for each type of disaster.

mysql> CREATE VIEW Total\_Donations\_By\_Disaster AS

-> SELECT di.disaster\_type, SUM(d.amount) AS total\_donated

-> FROM Donations d

-> JOIN Disaster di ON d.disaster\_id = di.disaster\_id

-> GROUP BY di.disaster\_type;

Query OK, 0 rows affected (0.07 sec)

mysql> select \* from Total\_Donations\_By\_Disaster;

+---------------+---------------+

| disaster\_type | total\_donated |

+---------------+---------------+

| Flood | 180000.00 |

| Earthquake | 75000.00 |

| Cyclone | 65000.00 |

| Landslide | 20000.00 |

| Heat Wave | 55000.00 |

+---------------+---------------+

5 rows in set (0.01 sec)

**Dropping Views**

When a view is no longer needed, it can be removed.

mysql> DROP VIEW Total\_Donations\_By\_Disaster;

Query OK, 0 rows affected (0.01 sec)

**CONCLUSION**

The Disaster Management System (DMS) project implemented in this database management system showcases how SQL commands and database structures can significantly enhance the efficiency and effectiveness of disaster response efforts in India. Throughout the development of this project, various aspects of SQL were covered, including DDL, DML, TCL, VDL, relational algebra operations, nested queries, set operations, and the implementation of aggregate functions.

By leveraging these tools, stakeholders can enhance their decision-making processes and ultimately improve outcomes in disaster situations. By continuing to refine the database design and expanding its functionality, the Disaster Management System can serve as a crucial resource for effective disaster response and management in the future. If further enhancements or real-time data integration are developed, the system's capabilities will only grow, further supporting the safety and well-being of communities across India.

**REFERENCES**

[1] National Disaster Management Authority, “Disaster Management in India: An Overview,” 2020. [Online]. Available: <https://ndma.gov.in/>. [Accessed: Oct. 16, 2024].

[2] United Nations Office for Disaster Risk Reduction, “Disaster Risk Reduction in India,” 2019. [Online]. Available: <https://www.undrr.org/>. [Accessed: Oct. 16, 2024].

[3] V. Prasad and S. Rao, “Disaster Management and Humanitarian Response in India,” *International Journal of Disaster Risk Reduction*, vol. 50, pp. 101763, 2020. https://doi.org/10.1016/j.ijdrr.2020.101763.

[4] S. Dey and D. Mandal, “Role of Information Technology in Disaster Management,” *Journal of Information Technology Research*, vol. 13, no. 2, pp. 55-70, 2020. https://doi.org/10.4018/JITR.2020040104.

[5] R. Rajesh and K. Sharma, “Relief Operations in India: Challenges and Solutions,” *Disaster Science Journal*, vol. 11, no. 1, pp. 34-47, 2019. [Online]. Available: <http://www.disastersciencejournal.com>. [Accessed: Oct. 16, 2024].

[6] J. Silvano and A. Vargas, “Using Views in SQL for Data Management,” *Database Journal*. [Online]. Available: <https://www.databasejournal.com/>. [Accessed: Oct. 16, 2024].

[7] MySQL Documentation, “MySQL 8.0 Reference Manual,” [Online]. Available: <https://dev.mysql.com/doc/>. [Accessed: Oct. 16, 2024].