

SHIV NADAR
—UNIVERSITY—
CHENNAI

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF ENGINEERING**

LABORATORY RECORD

**B.TECH
(YEAR : 2023 - 2024)**

NAME : MONISHAA S
REG. NO. : 22D11101071
DEPT. : B.TECH SEM. : IV CLASS & SEC : AIDS - 'B'

SHIV NADAR
—UNIVERSITY—
CHENNAI

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

..... Laboratory by

Name Manisha S

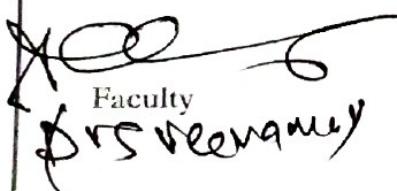
Register Number 22011101071

Semester 4 th Class & Sec. AIDS (B)

Branch B.Tech

SHIV NADAR UNIVERSITY Chennai

During the Academic year 2023 - 2024


Faculty
Dr. S. Venkateswaran

Head of the Department

Submitted for the Practical Examination held at
SNU CHENNAI on

[Signature]
Internal Examiner

[Signature]
External Examiner

INDEX

Name : **MANISHA S.** Reg. No. **22011101071**
 Sem : **IV** Class & Sec : **AI/DS - 'B'**

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
1	18/1/23	Problem Identification	1		
2	1/1/23	Requirement Gathering	2		
3	8/1/24	Design using ER diagram	4		
4	22/1/24	Design using UML diagram	6		
5	29/1/24	Working with DDL and DML commands	16		
6	5/2/24	Working with joins, set, aggregate function	22	A.7	
7	19/2/24	Working with queries related to project	25		
8	26/2/24	Working with basic PL/SQL programming	48		
9	04/3/24	Working with PL/SQL procedures	63		
10	11/3/24	Working with PL/SQL functions	66		
11	18/3/24	Working with PL/SQL cursor	69		
12	25/3/24	Working with PL/SQL triggers	73		
13	1/4/24	Project report	76-82		

<u>Ex-1</u>	<u>PROBLEM IDENTIFICATION</u>
--------------------	--------------------------------------

AIM:

To develop a Flood Risk Prediction and A Warning System using DBMS that integrates historical flood data, terrain information, and climate models to predict flood risks and henceforth Provide early warning alerts to residents and authorities, allowing for proactive measures to mitigate damage.

PROBLEM STATEMENT:

In light of the escalating frequency and heightened severity of flooding incidents, there arises an imperative for the development of an advanced Flood Risk Prediction and Warning System. The objective is to construct a sophisticated Database Management System (DBMS) that seamlessly amalgamates historical flood data, terrain particulars, and climate models with the utmost precision for the purpose of predicting flood risks. The resultant system is envisioned to furnish punctual and dependable early warning alerts to both residents and authorities, affording them the opportunity to proactively undertake measures and execute efficacious strategies in order to preemptively address and alleviate potential damages. The Key Objectives of this project include Data Integration, Database Design and Management, Flood Risk Prediction, Early Warning System, User-friendly Interface, Proactive Measures coupled with Mitigation providing Scalability and Reliability.

RESULT:

Our Flood Risk Prediction System works for all regions irrespective of many topological conditions with keen consciousness of the micro errors to provide an early warning which helps the government and the people.

The goal is to provide a Flood Risk Prediction coupled with a Warning system that contributes significantly to enhancing community resilience, reducing property damage, and saving lives in flood-prone regions.

<u>Ex-2</u>	
--------------------	--

REQUIREMENT GATHERING

AIM:

Designing a database management system(DBMS) for a Flood Risk Prediction System coupled with a warning system which involves identifying entities, Relationships, and attributes relevant to the application. Below is a relational database schema for the same.

REQUIREMENT GATHERING:

Requirement gathering for Flood Risk Prediction and warning system involves understanding the needs of the problem statement and a suitable implementation of the idea for the optimal solution.

Entities:

1. Location:

- LocationID (Primary Key)
- Latitude
- Longitude
- Region
- Elevation

2. HistoricalFloodData:

- FloodDataID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- Date
- Magnitude
- Duration

3. EarlyWarningAlert:

- AlertID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- DateIssued
- AlertLevel
- RecommendedActions

4. PreventiveMeasure:

- MeasureID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- MeasureType
- Description
- ImplementationDate

5. Resident:

- ResidentID (Primary Key)
- Name
- Address
- ContactNumber
- LocationID (Foreign Key referencing Location.LocationID)

6. Authority:

- AuthorityID (Primary Key)
- Name
- Organization
- ContactNumber
- LocationID (Foreign Key referencing Location.LocationID)

7. EmergencyResponsePlan:

- PlanID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- PlanDescription
- ActivationDate

8. EmergencyResponseLog:

- LogID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- Date
- Time
- LogDetails

9. EvacuationCenter:

- CenterID (Primary Key)
- LocationID (Foreign Key referencing Location.LocationID)
- CenterName
- Capacity
- Location Details

RELATIONSHIPS:

- One-to-Many with HistoricalFloodData (One Location can have Many

HistoricalFloodData entries).
One-to-Many with EarlyWarningAlert (One Location can have Many EarlyWarningAlert entries).

- Many-to-One with Location (Many HistoricalFloodData entries can be associated with One Location).
- Many-to-One with Location (Many EarlyWarningAlert entries can be associated with One Location).
- Many-to-One with Location (Many PreventiveMeasures can be associated with One Location).
- Many-to-One with Location (Many Residents can be associated with One Location).
- Many-to-One with Location (Many Authorities can be associated with One Location).
- Many-to-One with Location (Many EmergencyResponsePlans can be associated with One Location).
- Many-to-One with Location (Many EmergencyResponseLogs can be associated with One Location).
- Many-to-One with Location (Many EvacuationCenters can be associated with One Location).

RESULT:

Hence, basic necessary entities are created with their relationships and the necessary requirement analysis is done. Additionally, incorporating normalization techniques to eliminate data redundancy can be ideal for optimal solutions for the above problem statement.

Ex-3

DESIGN USING ER DIAGRAM

AIM:

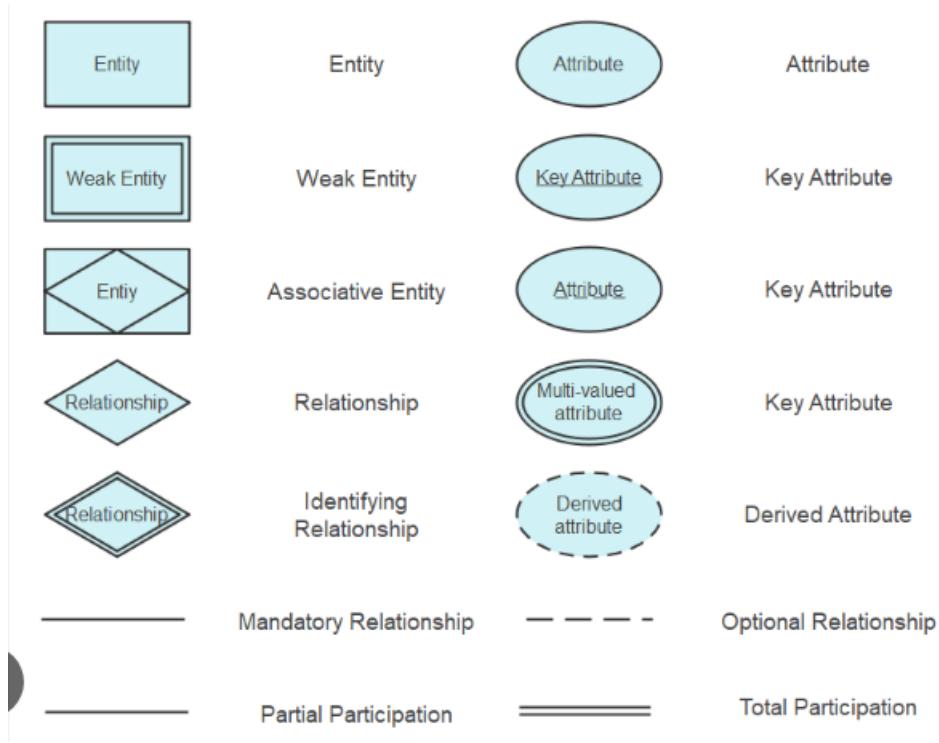
To brainstorm and create an ER Diagram for our project that is titled “Advanced Flood Risk Prediction and Warning System”.

ER DIAGRAM:

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are

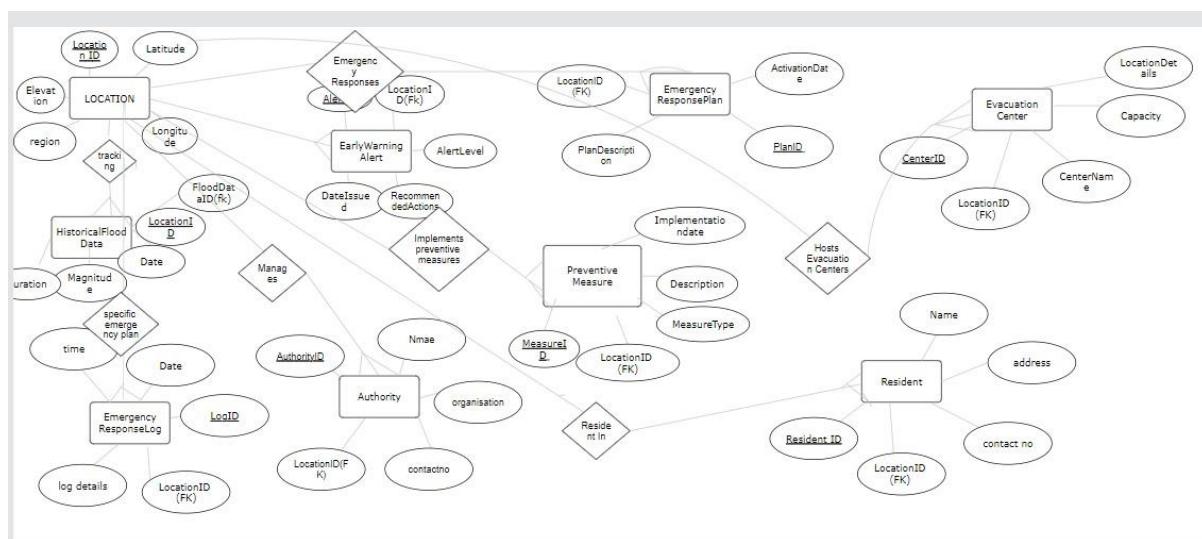
most often used to design or debug relational databases.

ER NOTATIONS:



Entities and their Attributes along with Primary Key:

ER DIAGRAM:



RESULT:

Therefore, an ER diagram for the topic Advanced Flood Risk Prevention and Early warning Database System has been designed successfully based on the given

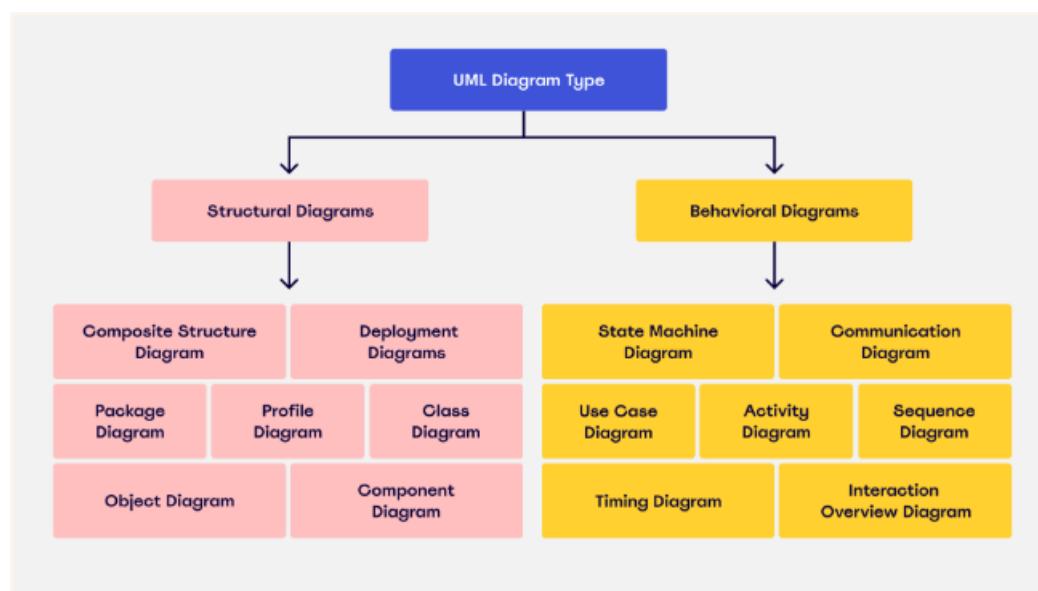
requirements.

<u>Ex-4</u>	<u>ARCHITECTURE OF DIAGRAMS</u>

AIM:

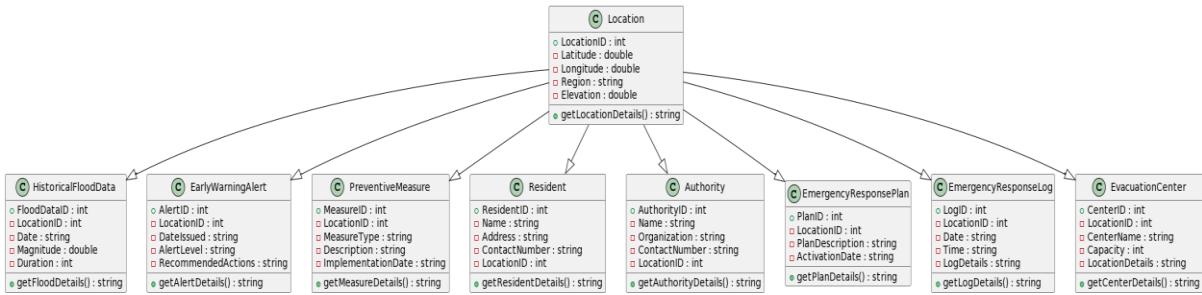
To understand the conceptually, the implantation part holds a huge role for it. A UML Diagram helps to visualize the system or progress of the project over time and its current situation. Errors can be reduced if detected.

Here's a breakdown of the different types of UML diagrams and whether they're classified as a structural diagram or behavioral diagram:



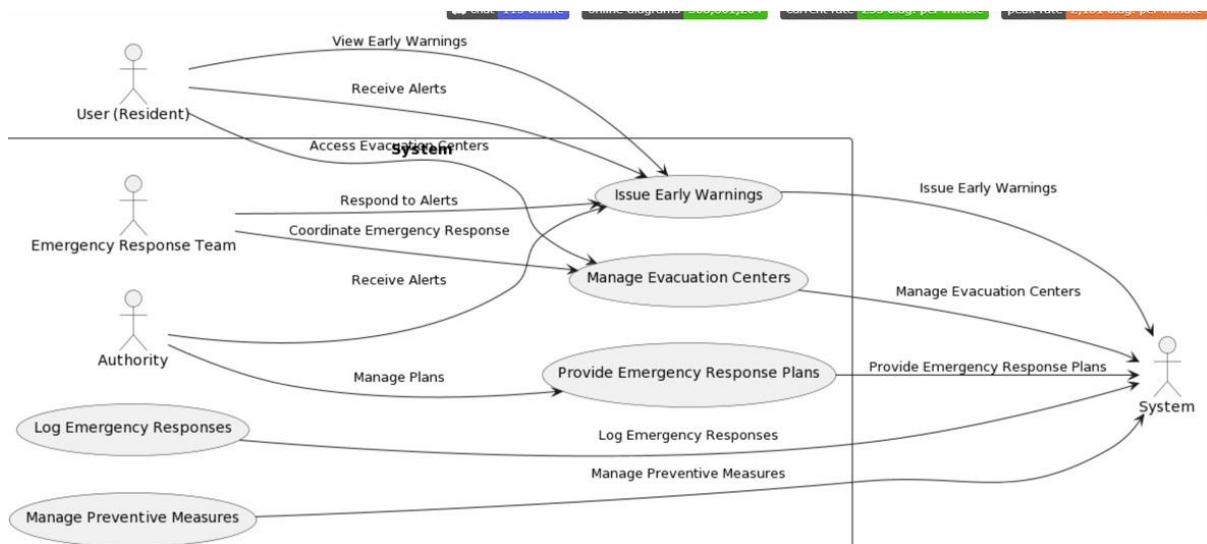
CLASS DIAGRAM:

A class diagram is a visual representation of the structure and relationships within a system, illustrating classes, their attributes, and methods. It provides a high-level view of the system's architecture, showcasing the static elements and their associations.



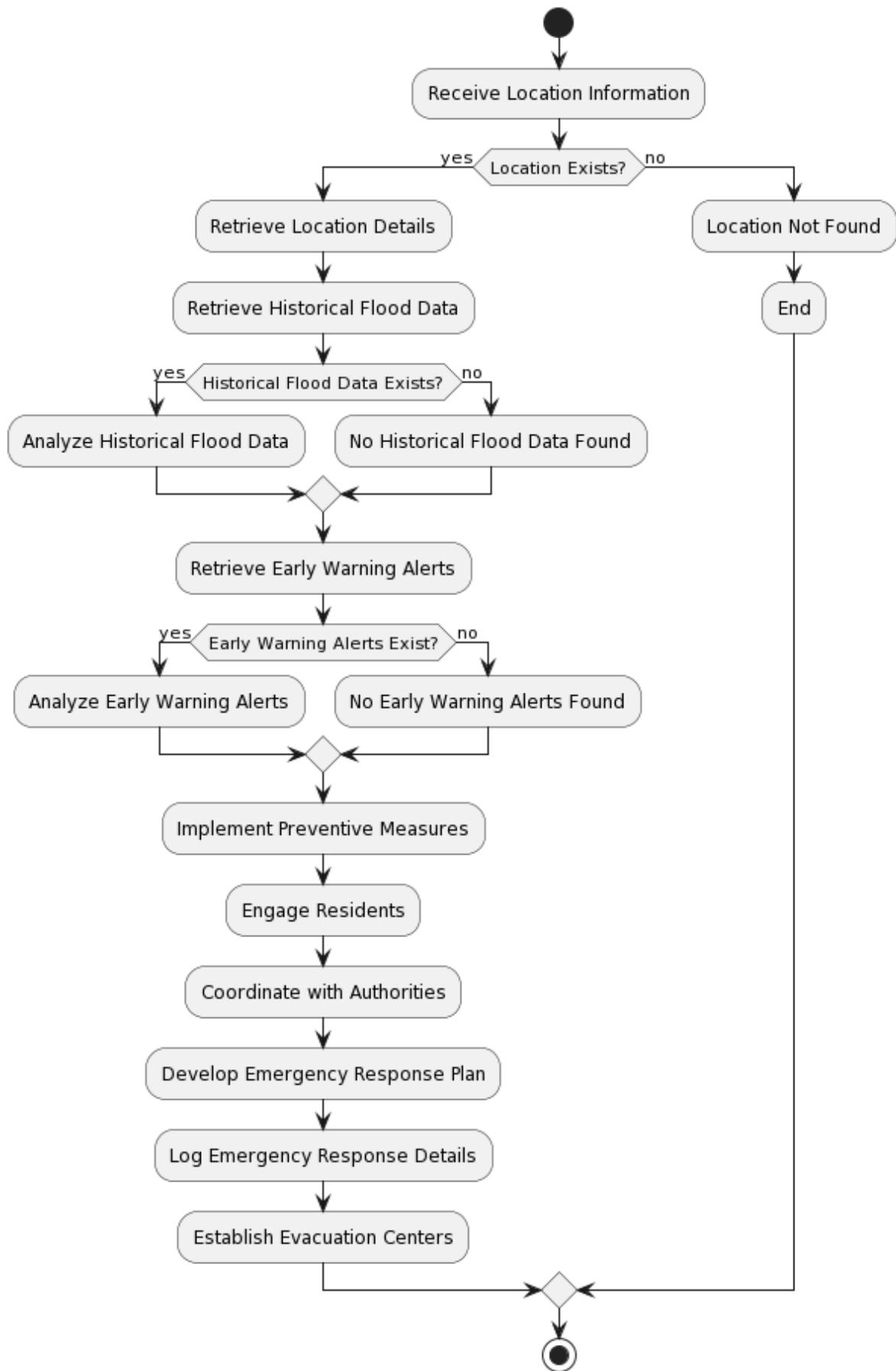
USE CASE DIAGRAM:

A use case diagram outlines the interactions between users and a system, illustrating various scenarios or use cases. It captures the functionalities a system offers from an external perspective, showcasing how actors (users) interact with the system to achieve specific goals.



ACTIVITY DIAGRAM:

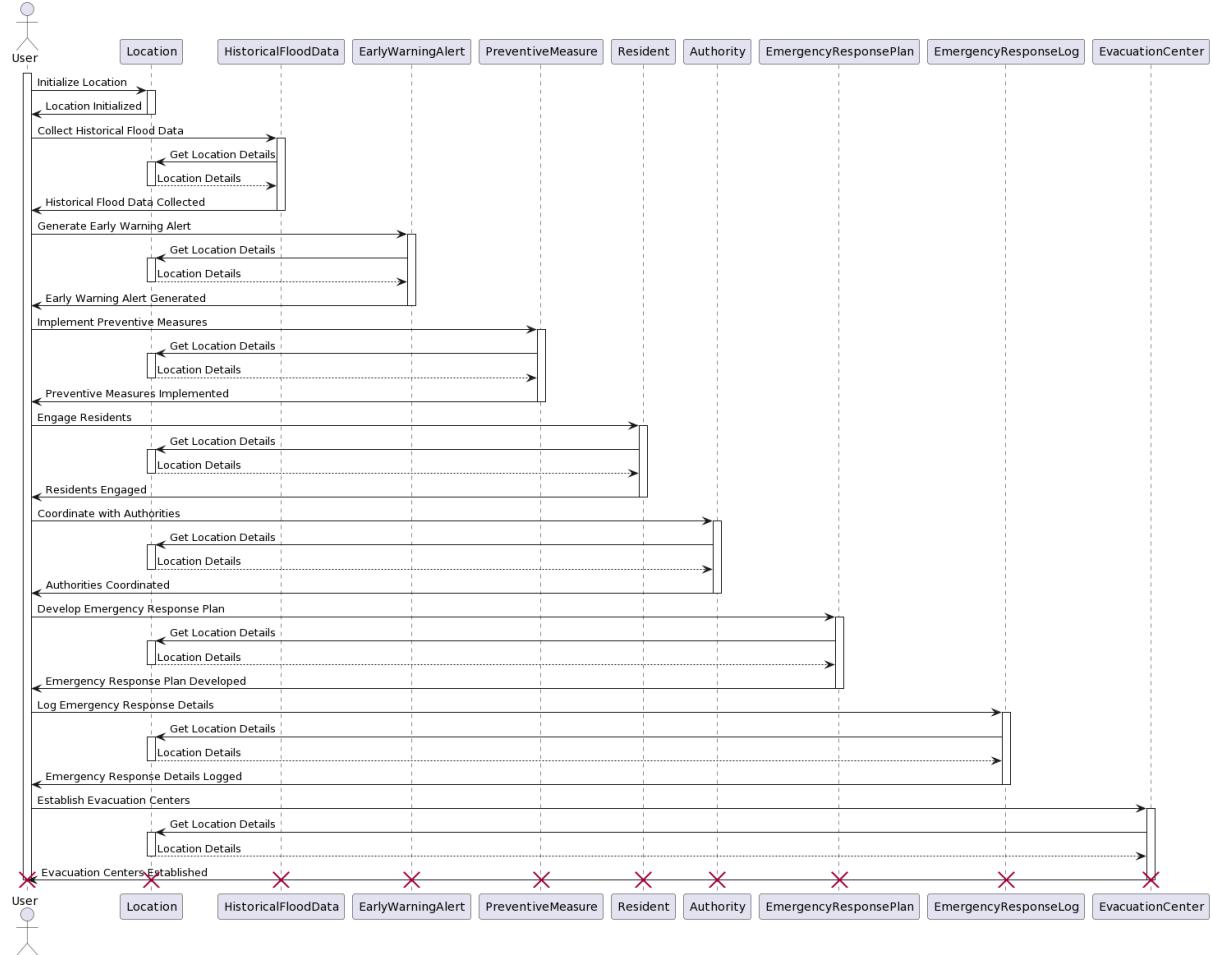
An activity diagram visually represents the flow of activities within a system, demonstrating the sequential steps and decision points in a process. It is particularly useful for modeling workflows and business processes, highlighting the dynamic aspects of system behavior.



SEQUENCE DIAGRAM:

A sequence diagram depicts the interactions between objects in a system over time, illustrating the order of messages exchanged among them. It provides a dynamic view of system behavior, showing how objects collaborate to accomplish specific

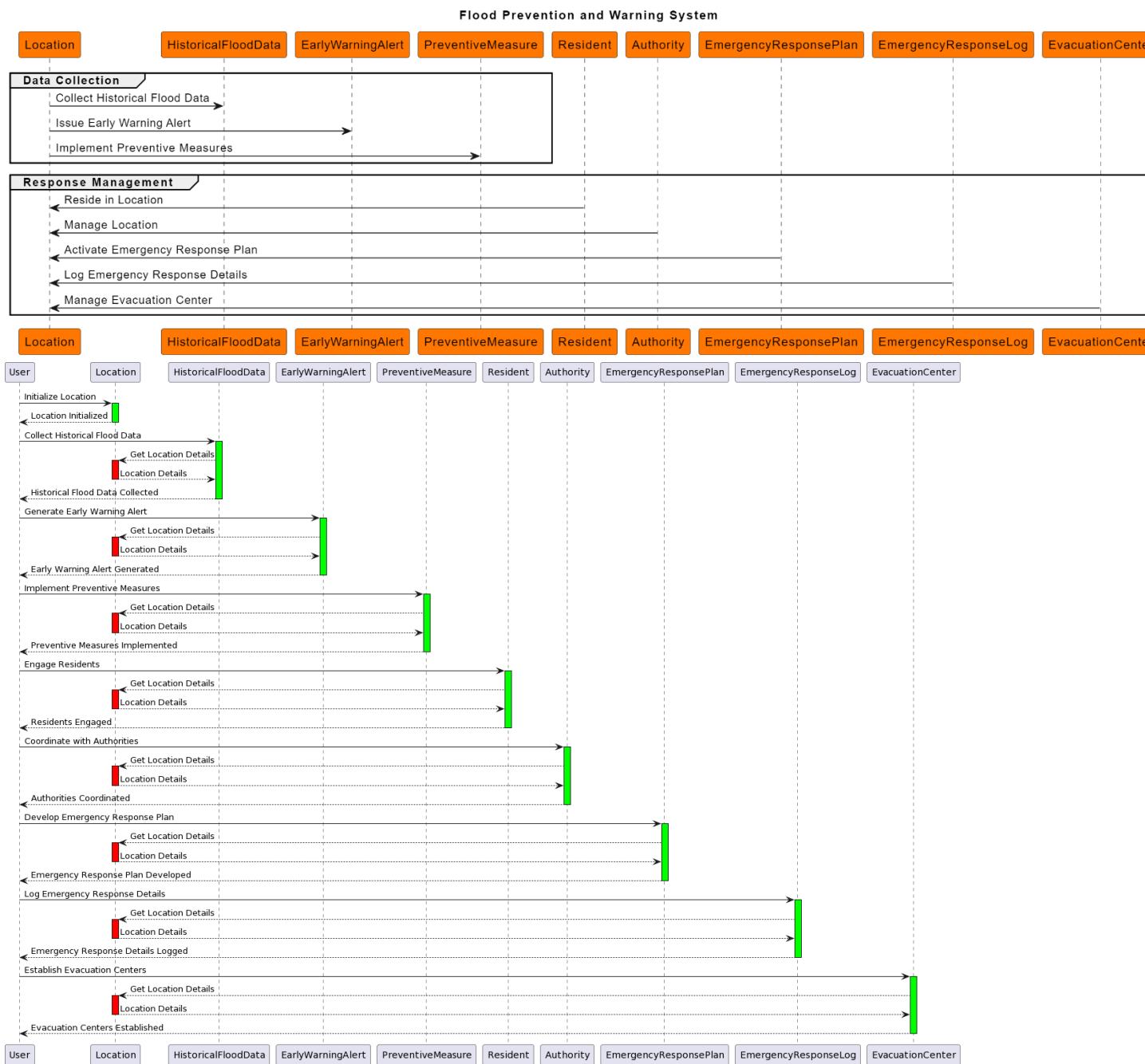
tasks or scenarios.



TIMING DIAGRAM:

A timing diagram portrays the timing aspects of interactions between objects in a system, emphasizing the durations of messages and the lifetimes of objects. It offers insights into the temporal aspects of a system's behavior, helping to understand the synchronization and timing constraints in a scenario.

=



RESULT:

To work on project designing using UML diagrams is successfully completed.

CONVERTING ER DIAGRAM TO TABLE

AIM:

To understand the conceptually, the implantation part holds a huge role for it. To

convert the Entity relationship diagram to tables and create data inside to feature and practice implementation through the venture of structured query language. SQL is a programming language which offers insights about the data inside the tables.

LIST OF TABLES:

- 1) Location Table
- 2) Historical Flood Data Table
- 3) Early Warning Alert Table
- 4) Preventive Measure Table
- 5) Resident Table
- 6) Authority Table
- 7) Emergency Response Plan Table
- 8) Emergency Response Log Table
- 9) Evacuation Center Table

1)LOCATION TABLE:

LocationID	Latitude	Longitude	Region	Elevation
-------------------	-----------------	------------------	---------------	------------------

LocationID	Latitude	Longitude	Region	Elevation
1	40.7128	-74.0060	New York	10
2	34.0522	-118.2437	Los Angeles	15
3	51.5074	-0.1278	London	20
4	48.8566	2.3522	Paris	30
5	35.6895	139.6917	Tokyo	25
6	37.7749	-122.4194	San Francisco	12
7	40.4168	-3.7038	Madrid	22
8	37.9838	23.7275	Athens	18
9	41.9028	12.4964	Rome	28
10	55.7558	37.6176	Moscow	35

2)HISTORICAL FLOOD DATA TABLE:

FloodDataID	LocationID	Date	Magnitude	Duration
--------------------	-------------------	-------------	------------------	-----------------

FloodDataID	LocationID	Date	Magnitude	Duration
1	1	2022-01-05	4	48
2	2	2022-02-12	3	36
3	3	2022-03-20	5	72

4	4	2022-04-18	3.5	24
5	5	2022-05-25	4.5	60
6	6	2022-06-10	3	48
7	7	2022-07-15	4	36
8	8	2022-08-22	3.5	24
9	9	2022-09-30	5	72
10	10	2022-10-18	4.5	60

3)EARLY WARNING ALERT TABLE:

AlertID | LocationID | DateIssued | AlertLevel | RecommendedActions

1	1	2022-01-03	High	Evacuate
2	2	2022-02-10	Moderate	Stay Alert
3	3	2022-03-18	High	Prepare
4	4	2022-04-15	Low	Monitor
5	5	2022-05-22	Moderate	Prepare
6	6	2022-06-08	High	Evacuate
7	7	2022-07-13	Low	Monitor
8	8	2022-08-20	Moderate	Prepare
9	9	2022-09-28	High	Evacuate
10	10	2022-10-15	Low	Monitor

4)PREVENTIVE MEASURE TABLE:

MeasureID | LocationID | MeasureType | Description | ImplementationDate

1	1	Structural	Reinforcement of riverbanks	2022-01-15
2	2	Early Warning	Installation of flood sensors	2022-02-20
3	3	Community	Conducting awareness campaigns	2022-03-25
4	4	Infrastructure	Upgrading drainage systems	2022-04-30
5	5	Evacuation	Establishment of evacuation routes	2022-05-10
6	6	Monitoring	Implementing real-time monitoring	2022-06-15

<u>7</u>	<u>7</u>	Preparedness	Training emergency response teams	2022-07-20
<u>8</u>	<u>8</u>	Communication	Enhancing public communication channels	2022-08-25
<u>9</u>	<u>9</u>	Equipment	Procurement of emergency supplies	2022-09-05
<u>10</u>	<u>10</u>	Technology	Implementing GIS-based flood modeling	2022-10-10

5)RESIDENT TABLE:

<u>ResidentID</u>	Name	Address	ContactNumber	<u>LocationID</u>
<u>1</u>	John Doe	123 Main St, City	123-456-7890	<u>1</u>
<u>2</u>	Jane Smith	456 Oak Ave, Town	456-789-0123	<u>2</u>
<u>3</u>	David Johnson	789 Elm Rd, Village	789-012-3456	<u>3</u>
<u>4</u>	Sarah Brown	321 Pine Blvd, Suburb	321-654-9870	<u>4</u>
<u>5</u>	Emily Wilson	567 Maple Dr, Hamlet	567-890-1234	<u>5</u>
<u>6</u>	Michael Lee	901 Cedar Ln, County	901-234-5678	<u>6</u>
<u>7</u>	Laura Miller	234 Birch St, Township	234-567-8901	<u>7</u>
<u>8</u>	Kevin Clark	678 Walnut Ave, Parish	678-901-2345	<u>8</u>
<u>9</u>	Amanda White	890 Spruce Rd, Borough	890-123-4567	<u>9</u>
<u>10</u>	Eric Taylor	345 Oakwood Dr, Manor	345-678-9012	<u>10</u>

6)AUTHORITY TABLE:

<u>AuthorityID</u>	Name	Organization	ContactNumber	<u>LocationID</u>
<u>1</u>	Mark Anderson	City Emergency Dept	111-222-3333	<u>1</u>
<u>2</u>	Lisa Williams	County Disaster Team	444-555-6666	<u>2</u>
<u>3</u>	James Wilson	Township Fire Dept	777-888-9999	<u>3</u>
<u>4</u>	Sarah Brown	Parish Police Dept	000-111-2222	<u>4</u>
<u>5</u>	Michael Clark	Village Emergency	333-444-5555	<u>5</u>
<u>6</u>	Emma Johnson	Borough Rescue Squad	666-777-8888	<u>6</u>
<u>7</u>	Ryan Martinez	Hamlet Emergency	999-000-1111	<u>7</u>
<u>8</u>	Rachel Lee	Town Disaster Relief	222-333-4444	<u>8</u>
<u>9</u>	David White	County Emergency	555-666-7777	<u>9</u>
<u>10</u>	Jessica Taylor	Township Rescue	888-999-0000	<u>10</u>

7)EMERGENCY RESPONSE TABLE:

<u>PlanID</u>	<u>LocationID</u>	<u>PlanDescription</u>	<u>ActivationDate</u>
1	1	Emergency Evacuation Plan for City	2022-01-01
2	2	Disaster Preparedness Plan for County	2022-02-05
3	3	Town Emergency Response Plan	2022-03-10
4	4	Parish Crisis Management Plan	2022-04-15
5	5	Hamlet Disaster Recovery Plan	2022-05-20
6	6	County Emergency Action Plan	2022-06-25
7	7	Township Disaster Management Plan	2022-07-30
8	8	Village Emergency Preparedness Strategy	2022-08-05
9	9	Borough Contingency Plan	2022-09-10
10	10	Manor Disaster Response Framework	2022-10-15

8)EMERGENCY RESPONSE LOG TABLE:

<u>LogID</u>	<u>LocationID</u>	<u>Date</u>	<u>Time</u>	<u>LogDetails</u>
1	1	2022-01-02	09:00	Flood monitoring initiated
2	2	2022-02-08	12:30	Warning alert issued for flooding
3	3	2022-03-12	15:45	Activation of emergency response teams
4	4	2022-04-20	10:15	Drainage system inspection completed
5	5	2022-05-18	11:20	Evacuation routes marked and verified
6	6	2022-06-24	08:45	Real-time monitoring system activated
7	7	2022-07-28	14:00	Training session conducted for responders
8	8	2022-08-30	16:30	Public communication channels tested
9	9	2022-09-05	13:45	Emergency supplies inventory updated
10	10	2022-10-12	07:55	GIS-based flood modeling in progress

9)EVACUATION CENTER TABLE:

CenterID	LocationID	CenterName	Capacity	LocationDetails
1	1	City Evacuation Center	500	City Hall
2	2	County Shelter	300	Community Center
3	3	Town Emergency Center	200	Town Hall
4	4	Parish Safe Haven	400	Parish School
5	5	Hamlet Evacuation Site	250	Hamlet Park
6	6	County Relief Center	350	County Fairgrounds
7	7	Township Refuge	150	Township Library
8	8	Village Shelter	280	Village Hall
9	9	Borough Emergency Hub	180	Borough Community Center
10	10	Manor Safe Zone	400	Manor Arena

Ex-5

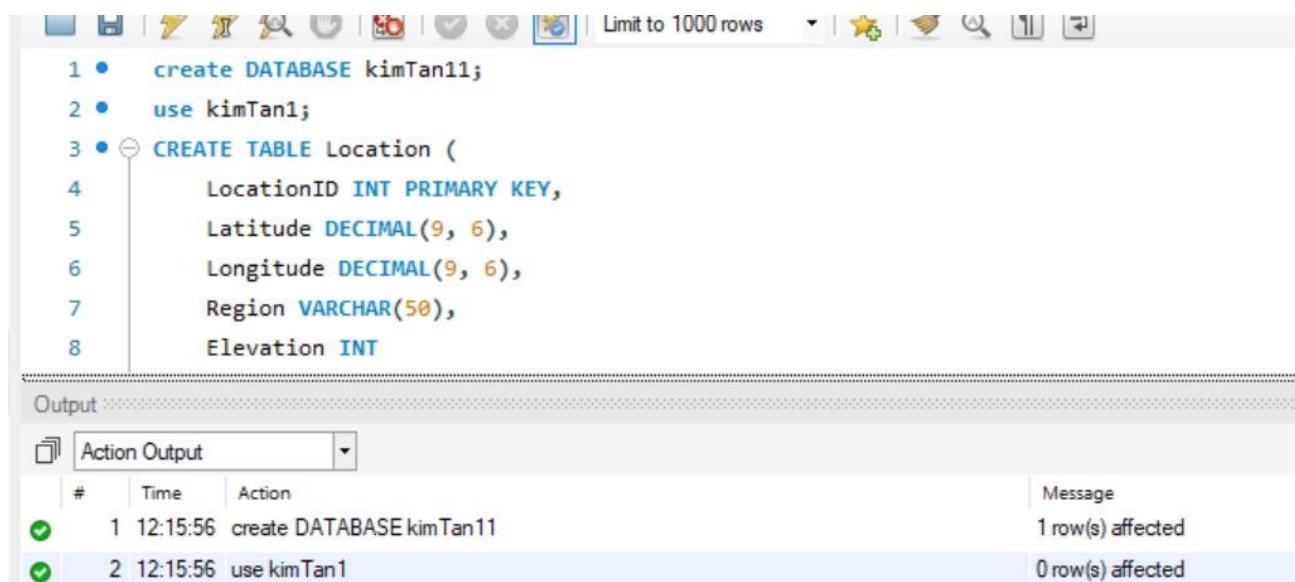
WORKING WITH DDL AND DML COMMANDS

AIM:

To work with DDL & DML commands.

1. DDL COMMANDS:-

- a. **CREATE:** This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).



The screenshot shows the MySQL Workbench interface. In the SQL editor pane, the following DDL commands are written:

```
1 •  create DATABASE kimTan11;
2 •  use kimTan1;
3 •  CREATE TABLE Location (
4      LocationID INT PRIMARY KEY,
5      Latitude DECIMAL(9, 6),
6      Longitude DECIMAL(9, 6),
7      Region VARCHAR(50),
8      Elevation INT
```

In the Output pane, the results of the execution are shown:

Action	Time	Action	Message
create DATABASE kimTan11	1 12:15:56		1 row(s) affected
use kimTan1	2 12:15:56		0 row(s) affected

```
27     );
28
29 • ⊖ CREATE TABLE PreventiveMeasure (
30     MeasureID INT PRIMARY KEY,
31     LocationID INT,
32     MeasureType VARCHAR(20),
33     Description VARCHAR(100),
34     ImplementationDate DATE,
35     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
36 );
37
38 • ⊖ CREATE TABLE Resident (
39     ResidentID INT PRIMARY KEY,
40     Name VARCHAR(50),
41     Address VARCHAR(100),
42     ContactNumber VARCHAR(15),
43     LocationID INT,
44     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
45 );
46
47 • ⊖ CREATE TABLE Authority (
48     AuthorityID INT PRIMARY KEY,
49     Name VARCHAR(50),
50     Organization VARCHAR(50),
51     ContactNumber VARCHAR(15),
52     LocationID INT,
```

```

22
56 • CREATE TABLE EmergencyResponse (
57     PlanID INT PRIMARY KEY,
58     LocationID INT,
59     PlanDescription VARCHAR(100),
60     ActivationDate DATE,
61     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
62 );
63
64 • CREATE TABLE EmergencyResponseLog (
65     LogID INT PRIMARY KEY,
66     LocationID INT,
67     Date DATE,
68     Time TIME,
69     LogDetails VARCHAR(100),
70     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
71 );
72
73 • CREATE TABLE EvacuationCenter (
74     CenterID INT PRIMARY KEY,
75     LocationID INT,
76     CenterName VARCHAR(50),
77     Capacity INT,
78     LocationDetails VARCHAR(100),
79     FOREIGN KEY (LocationID) REFERENCES Location(LocationID)
80 );

```

- b. **DROP:** This command is used to delete objects from the database.

30 21:06:37 DROP TABLE IF EXISTS FloodData 0 row(s) affected 0.031 sec

```

3      -- Drop FloodData table
4 •  DROP TABLE IF EXISTS FloodData;
5

```

- c. **ALTER:** This is used to alter the structure of the database.

```
Alter table account1 ADD email varchar(20);
```

- d. **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.

```

3
4      -- Truncate the tables with foreign key constraints
5 •   TRUNCATE TABLE EarlyWarningAlert;
6 •   TRUNCATE TABLE PreventiveMeasure;
7 •   TRUNCATE TABLE Resident;
8 •   TRUNCATE TABLE Authority;
9 •   TRUNCATE TABLE EmergencyResponse;
10 •  TRUNCATE TABLE EmergencyResponseLog;
11 •  TRUNCATE TABLE EvacuationCenter;
12
13      -- Now you can truncate the Location table
14 •  TRUNCATE TABLE Location;
15
16      -- Enable foreign key checks back
17 •  SET FOREIGN_KEY_CHECKS = 1;
18

```

✓	31	21:12:03	SET FOREIGN_KEY_CHECKS = 0	0 row(s) affected	0.000 sec
✓	32	21:12:03	TRUNCATE TABLE EarlyWarningAlert	0 row(s) affected	0.063 sec
✓	33	21:12:03	TRUNCATE TABLE PreventiveMeasure	0 row(s) affected	0.031 sec
✓	34	21:12:03	TRUNCATE TABLE Resident	0 row(s) affected	0.031 sec
✓	35	21:12:03	TRUNCATE TABLE Authority	0 row(s) affected	0.016 sec
✓	36	21:12:03	TRUNCATE TABLE EmergencyResponse	0 row(s) affected	0.031 sec
✓	37	21:12:03	TRUNCATE TABLE EmergencyResponseLog	0 row(s) affected	0.031 sec
✓	38	21:12:03	TRUNCATE TABLE EvacuationCenter	0 row(s) affected	0.032 sec
✓	39	21:12:03	TRUNCATE TABLE Location	0 row(s) affected	0.031 sec
✓	40	21:12:03	SET FOREIGN_KEY_CHECKS = 1	0 row(s) affected	0.000 sec

2)DML COMMANDS:-

- a. **INSERT**: It is used to insert data into a table.

```
1
2      -- Insert data into Location table
3 •   INSERT INTO Location (LocationID, Latitude, Longitude, Region, Elevation)
4     VALUES
5         (1, 40.7128, -74.0060, 'New York', 10),
6         (2, 34.0522, -118.2437, 'Los Angeles', 15),
7         (3, 51.5074, -0.1278, 'London', 20),
8         (4, 48.8566, 2.3522, 'Paris', 30),
9         (5, 35.6895, 139.6917, 'Tokyo', 25),
10        (6, 37.7749, -122.4194, 'San Francisco', 12),
11        (7, 40.4168, -3.7038, 'Madrid', 22),
12        (8, 37.9838, 23.7275, 'Athens', 18),
13        (9, 41.9028, 12.4964, 'Rome', 28),
14        (10, 55.7558, 37.6176, 'Moscow', 35);
15
16
17      -- Insert data into FloodData table
18 •   INSERT INTO FloodData (FloodDataID, LocationID, Date, Magnitude, Duration) VA
19         (1, 1, '2022-01-05', 4, 48),
20         (2, 2, '2022-02-12', 3, 36),
21         (3, 3, '2022-03-20', 5, 72),
22         (4, 4, '2022-04-18', 3.5, 24),
23         (5, 5, '2022-05-25', 4.5, 60),
24         (6, 6, '2022-06-10', 3, 48),
25         (7, 7, '2022-07-15', 4, 36),
26         (8, 8, '2022-08-22', 3.5, 24),
```

- b. **UPDATE:** It is used to update existing data within a table.

```
1      -- Update multiple columns of the Location table for a specific Locat
2 •   UPDATE Location
3     SET Region = 'New Region', Latitude = 50.0, Longitude = -90.0
4     WHERE LocationID = 1;
5
```

41 21:14:51 UPDATE Location SET Region = 'New Region', Latitude = 50.0, Longitude = -90.0 ... 0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0

- c. **DELETE:** It is used to delete records from a database table.

```
3 • delete from account1 where id=1;
4
```

put

Action Output

#	Time	Action	Message
3	09:07:37	Altertable account1 ADD email varchar(20)	Error Code: 1146. Table 'dbms_prj.account1' doesn't exist
4	09:08:10	CREATE TABLE account1 (id INT AUTO_INCREMENT,	0 row(s) affected
5	09:08:13	Altertable account1 ADD email varchar(20)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
6	09:09:32	Truncate table account1	0 row(s) affected
7	09:10:53	delete from account1 where id=1	0 row(s) affected

RESULT:

Thus, working with DDL & DML commands on MySQL is successfully verified.

Ex-6

WORKING WITH JOIN, SET OPERATIONS AND AGGREGATE FUNCTIONS

AIM:

To perform join, set and aggregate functions on MySQL.

Query to Identify Regions Most Affected by Floods:

```
SELECT l.LocationID, l.Region  
FROM Location l  
JOIN (SELECT LocationID FROM FloodData GROUP BY LocationID ORDER  
BY COUNT(*) DESC LIMIT 5) sub  
ON l.LocationID = sub.LocationID;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** The query is displayed in the Query 1 tab.
- Toolbar:** Includes standard database navigation icons (File, Database, Table, View, etc.) and a toolbar with various tools like Refresh, Save, and Help.
- Result Grid:** A table titled "Result Grid" showing the output of the query. It has two columns: "LocationID" and "Region". The data is as follows:

LocationID	Region
1	New York
2	Los Angeles
3	London
4	Paris
5	Tokyo

Count the number of preventive measures of each type:

```
SELECT MeasureType, COUNT(*) AS MeasureCount  
FROM PreventiveMeasure  
GROUP BY MeasureType;
```

```

1 •  SELECT MeasureType, COUNT(*) AS MeasureCount
2   FROM PreventiveMeasure
3   GROUP BY MeasureType;
4

```

MeasureType	MeasureCount
Structural	1
Early Warning	1
Community	1
Infrastructure	1
Evacuation	1
Monitoring	1
Preparedness	1
Communication	1
Equipment	1
Technology	1

Retrieve residents' contact information for regions with flood occurrences:

```

SELECT DISTINCT r.Name, r.Address, r.ContactNumber
FROM Resident r
JOIN Location l ON r.LocationID = l.LocationID
JOIN FloodData f ON l.LocationID = f.LocationID;

```

```

1 •  SELECT DISTINCT r.Name, r.Address, r.ContactNumber
2   FROM Resident r
3   JOIN Location l ON r.LocationID = l.LocationID
4   JOIN FloodData f ON l.LocationID = f.LocationID;
5

```

Name	Address	ContactNumber
John Doe	123 Main St, City	123-456-7890
Jane Smith	456 Oak Ave, Town	456-789-0123
David Johnson	789 Elm Rd, Village	789-012-3456
Sarah Brown	321 Pine Blvd, Suburb	321-654-9870
Emily Wilson	567 Maple Dr, Hamlet	567-890-1234
Michael Lee	901 Cedar Ln, County	901-234-5678
Laura Miller	234 Birch St, Township	234-567-8901
Kevin Clark	678 Walnut Ave, Parish	678-901-2345
Amanda White	890 Spruce Rd, Borough	890-123-4567
Eric Taylor	345 Oakwood Dr, Manor	345-678-9012

Retrieve authorities for regions with flood occurrences and high alert levels:

```

SELECT DISTINCT a.Name, a.Organization
FROM Authority a
JOIN Location l ON a.LocationID = l.LocationID
JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID
WHERE e.AlertLevel = 'High';

```

The screenshot shows a MySQL Workbench interface with multiple tabs at the top labeled 'Query 1' through 'SQL File 6'. The 'SQL File' tab is active. Below the tabs is a toolbar with various icons. A query window contains the following SQL code:

```

2   FROM Authority a
3   JOIN Location l ON a.LocationID = l.LocationID
4   JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID
5   WHERE e.AlertLevel = 'High';
6

```

Below the code is a 'Result Grid' showing the following data:

	Name	Organization
▶	Mark Anderson	City Emergency Dept
	James Wilson	Township Fire Dept
	Emma Johnson	Borough Rescue Squad
	David White	County Emergency

Retrieve flood data for regions with elevation less than 20 and magnitude greater than 2:

```

SELECT f.*
FROM FloodData f
JOIN Location l ON f.LocationID = l.LocationID
WHERE l.Elevation < 20
AND f.Magnitude > 2;

```

The screenshot shows a MySQL Workbench interface with multiple tabs at the top labeled 'Query 1' through 'SQL File 7'. The 'SQL File' tab is active. Below the tabs is a toolbar with various icons. A query window contains the following SQL code:

```

2   FROM FloodData f
3   JOIN Location l ON f.LocationID = l.LocationID
4   WHERE l.Elevation < 20
5   AND f.Magnitude > 2;
6

```

Below the code is a 'Result Grid' showing the following data:

	FloodDataID	LocationID	Date	Magnitude	Duration
▶	1	1	2022-01-05	4.00	48
	2	2	2022-02-12	3.00	36
	6	6	2022-06-10	3.00	48
	8	8	2022-08-22	3.50	24

RESULT: Thus, working with Join, Set operations and Aggregate functions on MySQL is successfully verified.

Ex-7

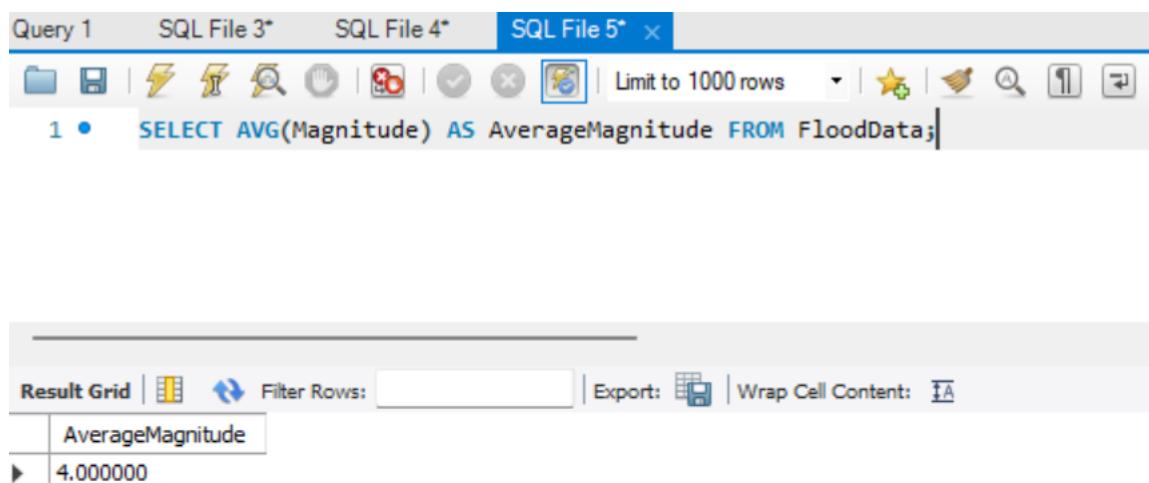
WORKING WITH QUERIES RELATED TO PROJECT

AIM:

To work with queries related to project on MySQL.

1) Query to Calculate Average Magnitude of Floods:

SELECT AVG(Magnitude) AS AverageMagnitude FROM FloodData;



The screenshot shows the MySQL Workbench interface. At the top, there are tabs for 'Query 1', 'SQL File 3*', 'SQL File 4*', and 'SQL File 5*' (which is active). Below the tabs is a toolbar with various icons for file operations, search, and database management. The main area contains a query editor window with the following content:

```
1 •    SELECT AVG(Magnitude) AS AverageMagnitude FROM FloodData;
```

Below the query editor is a results grid labeled 'Result Grid'. It has a single row with one column. The column header is 'AverageMagnitude' and the value is '4.000000'. There are also buttons for 'Filter Rows', 'Export', and 'Wrap Cell Content'.

2) Query to Identify Regions with Low Elevation:

```
SELECT LocationID, Region FROM Location WHERE Elevation < 15;
```

The screenshot shows the SQL Server Management Studio interface. The top ribbon has tabs for 'Query 1' and 'SQL File 3*' through 'SQL File 6*'. Below the tabs is a toolbar with various icons. The main area contains two numbered lines of SQL code:

```
1 • SELECT LocationID, Region FROM Location WHERE Elevation < 15;
2 |
```

Below the code is a 'Result Grid' window. It has a header row with columns 'LocationID' and 'Region'. The data grid contains three rows of data:

LocationID	Region
1	New York
6	San Francisco
HULL	HULL

3) Query to Retrieve Early Warning Alerts:

```
SELECT * FROM EarlyWarningAlert;
```

4) Query to Count Total Preventive Measures Implemented:

```
SELECT COUNT(*) AS TotalPreventiveMeasures FROM PreventiveMeasure;
```

The screenshot shows the SQL Server Management Studio interface. The top ribbon has tabs for 'Query 1' and 'SQL File 3*' through 'SQL File 6*'. Below the tabs is a toolbar with various icons. The main area contains two numbered lines of SQL code:

```
3 • SELECT COUNT(*) AS TotalPreventiveMeasures FROM PreventiveMeasure;
4 |
```

Below the code is a 'Result Grid' window. It has a header row with a single column 'TotalPreventiveMeasures'. The data grid contains one row of data:

TotalPreventiveMeasures
10

The screenshot shows the SQL Server Management Studio interface. The top ribbon has tabs for 'Query 1' and 'SQL File 3*' through 'SQL File 6*'. Below the tabs is a toolbar with various icons. The main area contains two numbered lines of SQL code:

```
3 • SELECT COUNT(*) AS TotalPreventiveMeasures FROM PreventiveMeasure;
4 |
```

Below the code is a 'Result Grid' window. It has a header row with a single column 'TotalPreventiveMeasures'. The data grid contains one row of data:

TotalPreventiveMeasures
10

```

1 •   SELECT * FROM EarlyWarningAlerts;
2
3

```

Sult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

AlertID	LocationID	DateIssued	AlertLevel	RecommendedActions
1	1	2022-01-03	High	Evacuate
2	2	2022-02-10	Moderate	Stay Alert
3	3	2022-03-18	High	Prepare
4	4	2022-04-15	Low	Monitor
5	5	2022-05-22	Moderate	Prepare
6	6	2022-06-08	High	Evacuate
7	7	2022-07-13	Low	Monitor
8	8	2022-08-20	Moderate	Prepare
9	9	2022-09-28	High	Evacuate
10	10	2022-10-15	Low	Monitor
NULL	NULL	NULL	NULL	NULL

5) Query to Identify Authorities Responsible for Flood-Prone Areas:

```
SELECT DISTINCT AuthorityID, Name, Organization
```

```
FROM Authority
```

```
WHERE LocationID IN (SELECT LocationID FROM Location WHERE
Elevation < 15);
```

Sult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

AuthorityID	Name	Organization
1	Mark Anderson	City Emergency Dept
6	Emma Johnson	Borough Rescue Squad
NULL	NULL	NULL

6) Query to Retrieve Activated Emergency Response Plans:

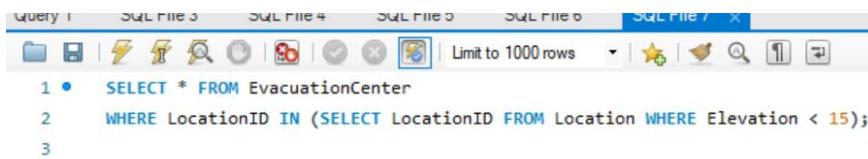
```
SELECT * FROM EmergencyResponse;
```

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* | Limit to 1000 rows | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

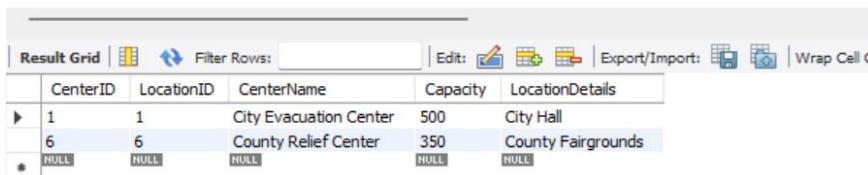
PlanID	LocationID	PlanDescription	ActivationDate
1	1	Emergency Evacuation Plan for City	2022-01-01
2	2	Disaster Preparedness Plan for County	2022-02-05
3	3	Town Emergency Response Plan	2022-03-10
4	4	Parish Crisis Management Plan	2022-04-15
5	5	Hamlet Disaster Recovery Plan	2022-05-20
6	6	County Emergency Action Plan	2022-06-25
7	7	Township Disaster Management Plan	2022-07-30
8	8	Village Emergency Preparedness Strategy	2022-08-05
9	9	Borough Contingency Plan	2022-09-10
10	10	Manor Disaster Response Framework	2022-10-15
NULL	NULL	NULL	NULL

7) Query to List Evacuation Centers in Flood-Prone Areas:

```
SELECT * FROM EvacuationCenter
WHERE LocationID IN (SELECT LocationID FROM Location WHERE
Elevation < 15);
```



```
Query 1 SQL File 3 SQL File 4 SQL File 5 SQL File 6 SQL File 7
1 • SELECT * FROM EvacuationCenter
2 WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);
3
```



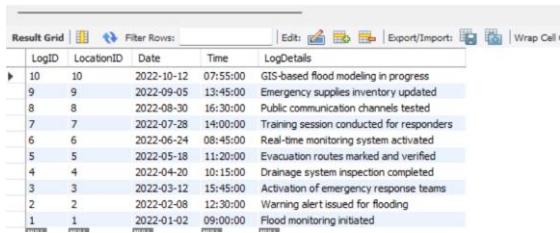
	CenterID	LocationID	CenterName	Capacity	LocationDetails
▶	1	1	City Evacuation Center	500	City Hall
▶	6	6	County Relief Center	350	County Fairgrounds
*	HULL	HULL	HULL	HULL	HULL

8) Query to Retrieve Recent Emergency Response Logs:

```
SELECT * FROM EmergencyResponseLog ORDER BY Date DESC, Time DESC
LIMIT 10;
```



```
Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7*
1 • SELECT * FROM EmergencyResponseLog ORDER BY Date DESC, Time DESC LIMIT 10;
2
```



	LogID	LocationID	Date	Time	LogDetails
▶	10	10	2022-10-12	07:55:00	GIS-based flood modeling in progress
▶	9	9	2022-09-05	13:45:00	Emergency supplies inventory updated
▶	8	8	2022-08-30	16:30:00	Public communication channels tested
▶	7	7	2022-07-28	14:00:00	Training session conducted for responders
▶	6	6	2022-06-24	08:45:00	Real-time monitoring system activated
▶	5	5	2022-05-18	11:20:00	Evacuation routes marked and verified
▶	4	4	2022-04-20	10:15:00	Drainage system inspection completed
▶	3	3	2022-03-12	15:45:00	Activation of emergency response teams
▶	2	2	2022-02-08	12:30:00	Warning alert issued for flooding
▶	1	1	2022-01-02	09:00:00	Flood monitoring initiated

9) Query to Calculate Total Duration of Floods:

```
SELECT SUM(Duration) AS TotalDuration FROM FloodData;
```

```
1 •  SELECT SUM(Duration) AS TotalDuration FROM FloodData;
2
```

TotalDuration
480

10) Query to Identify Regions Most Affected by Floods:

```
SELECT l.LocationID, l.Region
```

```
FROM Location l
```

```
JOIN (SELECT LocationID FROM FloodData GROUP BY LocationID ORDER  
BY COUNT(*) DESC LIMIT 5) sub
```

```
ON l.LocationID = sub.LocationID;
```

```
Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* ×
1 •  SELECT l.LocationID, l.Region
2   FROM Location l
3   JOIN (SELECT LocationID FROM FloodData GROUP BY LocationID ORDER BY COUNT(*) DESC LIMIT 5) sub
4     ON l.LocationID = sub.LocationID;
5
```

LocationID	Region
1	New York
2	Los Angeles
3	London
4	Paris
5	Tokyo

11) Retrieve all floods with magnitude greater than 4:

```
SELECT * FROM FloodData WHERE Magnitude > 4;
```

```
1 •  SELECT * FROM FloodData WHERE Magnitude > 4;
2
```

	FloodDataID	LocationID	Date	Magnitude	Duration
▶	3	3	2022-03-20	5.00	72
	5	5	2022-05-25	4.50	60
	9	9	2022-09-30	5.00	72
	10	10	2022-10-18	4.50	60
*	NULL	NULL	NULL	NULL	NULL

12) Retrieve preventive measures for a specific location:

SELECT * FROM PreventiveMeasure

WHERE LocationID = 1;

```
1 •  SELECT * FROM PreventiveMeasure
2   WHERE LocationID = 1;
3
```

	MeasureID	LocationID	MeasureType	Description	ImplementationDate
	1	1	Structural	Reinforcement of riverbanks	2022-01-15
*	NULL	NULL	NULL	NULL	NULL

13) Retrieve residents in low elevation areas:

SELECT * FROM Resident

WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);

```
Query 1      SQL File 3*    SQL File 4*    SQL File 5*    SQL File 6*    SQL File 7* ×
1 •  SELECT * FROM Resident
2   WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);
3
```

	ResidentID	Name	Address	ContactNumber	LocationID
▶	1	John Doe	123 Main St, City	123-456-7890	1
	6	Michael Lee	901 Cedar Ln, County	901-234-5678	6
*	NULL	NULL	NULL	NULL	NULL

14) Retrieve authorities' contact information:

SELECT Name, Organization, ContactNumber FROM Authority;

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* **SQL File 7***

1 • `SELECT Name, Organization, ContactNumber FROM Authority;`

2

Result Grid | Filter Rows: Export: Wrap Cell Content:

Name	Organization	ContactNumber
Mark Anderson	City Emergency Dept	111-222-3333
Lisa Williams	County Disaster Team	444-555-6666
James Wilson	Township Fire Dept	777-888-9999
Sarah Brown	Parish Police Dept	000-111-2222
Michael Clark	Village Emergency	333-444-5555
Emma Johnson	Borough Rescue Squad	666-777-8888
Ryan Martinez	Hamlet Emergency	999-000-1111
Rachel Lee	Town Disaster Relief	222-333-4444
David White	County Emergency	555-666-7777
Jessica Taylor	Township Rescue	888-999-0000

15) Retrieve emergency response plans for a specific location:

`SELECT * FROM EmergencyResponse`

`WHERE LocationID = 1;`

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* **SQL File 7***

1 • `SELECT * FROM EmergencyResponse`

2 `WHERE LocationID = 1;`

3

Result Grid | Filter Rows: Edit: Export/Import:

PlanID	LocationID	PlanDescription	ActivationDate
1	1	Emergency Evacuation Plan for City	2022-01-01
*	HULL	HULL	HULL

16) Retrieve evacuation centers with capacity greater than 200:

`SELECT * FROM EvacuationCenter WHERE Capacity > 200;`

File | New | Open | Save | Print | Help | Limit to 1000 rows | Star

1 • `SELECT * FROM EvacuationCenter WHERE Capacity > 200;`

2

Result Grid | Filter Rows: Edit: Export/Import:

CenterID	LocationID	CenterName	Capacity	LocationDetails
1	1	City Evacuation Center	500	City Hall
2	2	County Shelter	300	Community Center
4	4	Parish Safe Haven	400	Parish School
5	5	Hamlet Evacuation Site	250	Hamlet Park
6	6	County Relief Center	350	County Fairgrounds
8	8	Village Shelter	280	Village Hall
*	HULL	HULL	HULL	HULL

17) Retrieve recent emergency response logs for a specific location:

`SELECT * FROM EmergencyResponseLog`

```
WHERE LocationID = 1  
ORDER BY Date DESC, Time DESC  
LIMIT 10;
```

The screenshot shows a SQL query window titled "Query 1" with tabs for "SQL File 3*", "SQL File 4*", "SQL File 5*", "SQL File 6*", and "SQL File 7*" (the active tab). The query is:

```
1 •  SELECT * FROM EmergencyResponseLog  
2      WHERE LocationID = 1  
3      ORDER BY Date DESC, Time DESC  
4      LIMIT 10;  
5
```

The results grid displays one row of data:

	LogID	LocationID	Date	Time	LogDetails
▶	1	1	2022-01-02	09:00:00	Flood monitoring initiated
*	HULL	HULL	HULL	HULL	HULL

18) Retrieve flood data for a specific date:

```
SELECT * FROM FloodData
```

```
WHERE Date = '2022-01-01';
```

The screenshot shows a SQL query window titled "Query 1" with tabs for "SQL File 3*", "SQL File 4*", "SQL File 5*", and "SQL File 6*". The query is:

```
1 •  SELECT * FROM FloodData  
2      WHERE Date = '2021-01-01';  
3
```

The results grid displays one row of data:

	FloodDataID	LocationID	Date	Magnitude	Duration
*	HULL	HULL	HULL	HULL	HULL

19) Calculate the total duration of floods:

```
SELECT SUM(Duration) AS TotalDuration FROM FloodData;
```

The screenshot shows a SQL query window titled "Query 1" with tabs for "SQL File 3*", "SQL File 4*", "SQL File 5*", and "SQL File 6*". The query is:

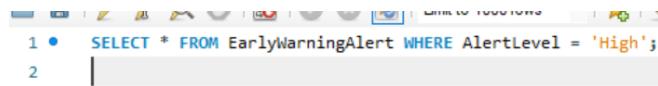
```
1 •  SELECT SUM(Duration) AS TotalDuration FROM FloodData;  
2
```

The results grid displays one row of data:

	TotalDuration
▶	480

20) Retrieve all early warning alerts with high alert levels:

```
SELECT * FROM EarlyWarningAlert WHERE AlertLevel = 'High';
```



A screenshot of a MySQL command-line interface. The query `SELECT * FROM EarlyWarningAlert WHERE AlertLevel = 'High';` is entered in the command line. The result grid shows four rows of data.

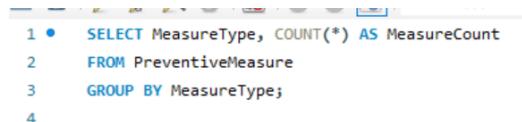
AlertID	LocationID	DateIssued	AlertLevel	RecommendedActions
1	1	2022-01-03	High	Evacuate
3	3	2022-03-18	High	Prepare
6	6	2022-06-08	High	Evacuate
9	9	2022-09-28	High	Evacuate
NULL	NULL	NULL	NULL	NULL

21) Count the number of preventive measures of each type:

```
SELECT MeasureType, COUNT(*) AS MeasureCount
```

```
FROM PreventiveMeasure
```

```
GROUP BY MeasureType;
```



A screenshot of a MySQL command-line interface. The query `SELECT MeasureType, COUNT(*) AS MeasureCount FROM PreventiveMeasure GROUP BY MeasureType;` is entered in the command line. The result grid shows ten rows of data, each representing a measure type and its count.

MeasureType	MeasureCount
Structural	1
Early Warning	1
Community	1
Infrastructure	1
Evacuation	1
Monitoring	1
Preparedness	1
Communication	1
Equipment	1
Technology	1

22) Retrieve residents' contact information for a specific region:

```
SELECT Name, Address, ContactNumber
```

```
FROM Resident
```

```
WHERE LocationID IN (SELECT LocationID FROM Location WHERE Region = 'New York');
```

```

1 •  SELECT Name, Address, ContactNumber
2   FROM Resident
3   WHERE LocationID IN (SELECT LocationID FROM Location WHERE Region = 'New York');
4

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Name	Address	ContactNumber
John Doe	123 Main St, City	123-456-7890

23) Retrieve emergency response plans activated after a specific date:

SELECT * FROM EmergencyResponse

WHERE ActivationDate > '2022-01-01';

```

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7*
File Edit View Insert Tools Options Help
1 •  SELECT * FROM EmergencyResponse
2   WHERE ActivationDate > '2022-01-01';
3

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

PlanID	LocationID	PlanDescription	ActivationDate
2	2	Disaster Preparedness Plan for County	2022-02-05
3	3	Town Emergency Response Plan	2022-03-10
4	4	Parish Crisis Management Plan	2022-04-15
5	5	Hamlet Disaster Recovery Plan	2022-05-20
6	6	County Emergency Action Plan	2022-06-25
7	7	Township Disaster Management Plan	2022-07-30
8	8	Village Emergency Preparedness Strategy	2022-08-05
9	9	Borough Contingency Plan	2022-09-10
10	10	Manor Disaster Response Framework	2022-10-15
*	NULL	NULL	NULL

24) Retrieve evacuation centers with capacity greater than 300 and located in low elevation areas:

SELECT * FROM EvacuationCenter

WHERE Capacity > 300

AND LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);

```

File Edit View Insert Tools Options Help
1 •  SELECT * FROM EvacuationCenter
2   WHERE Capacity > 300
3   AND LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);
4

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CenterID	LocationID	CenterName	Capacity	LocationDetails
1	1	City Evacuation Center	500	City Hall
6	6	County Relief Center	350	County Fairgrounds
*	NULL	NULL	NULL	NULL

25) Retrieve recent emergency response logs with flood-related details:

```
SELECT * FROM EmergencyResponseLog  
WHERE LogDetails LIKE '%flood%'  
ORDER BY Date DESC, Time DESC  
LIMIT 10;
```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  SELECT * FROM EmergencyResponseLog  
2 WHERE LogDetails LIKE '%flood%'  
3 ORDER BY Date DESC, Time DESC  
4 LIMIT 10;  
5
```

The results grid displays five rows of data:

	LogID	LocationID	Date	Time	LogDetails
▶	10	10	2022-10-12	07:55:00	GIS-based flood modeling in progress
2	2	2	2022-02-08	12:30:00	Warning alert issued for flooding
1	1	1	2022-01-02	09:00:00	Flood monitoring initiated
*	HULL	HULL	HULL	HULL	HULL

26) Retrieve flood data with magnitude greater than 3 and duration more than 24 hours:

```
SELECT * FROM FloodData  
WHERE Magnitude > 3  
AND Duration > 24;
```

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  SELECT * FROM FloodData  
2 WHERE Magnitude > 3  
3 AND Duration > 24;  
4
```

The results grid displays ten rows of data:

	FloodDataID	LocationID	Date	Magnitude	Duration
▶	1	1	2022-01-05	4.00	48
3	3	3	2022-03-20	5.00	72
5	5	5	2022-05-25	4.50	60
7	7	7	2022-07-15	4.00	36
9	9	9	2022-09-30	5.00	72
10	10	10	2022-10-18	4.50	60
*	HULL	HULL	HULL	HULL	HULL

27) Retrieve preventive measures implemented after a specific date:

```
SELECT * FROM PreventiveMeasure  
WHERE ImplementationDate > '2022-01-01';
```

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* **SQL File 7*** x

Limit to 1000 rows

```

1 •   SELECT * FROM PreventiveMeasure
2 WHERE ImplementationDate > '2022-01-01';
3

```

Result Grid

MeasureID	LocationID	MeasureType	Description	ImplementationDate
1	1	Structural	Reinforcement of riverbanks	2022-01-15
2	2	Early Warning	Installation of flood sensors	2022-02-20
3	3	Community	Conducting awareness campaigns	2022-03-25
4	4	Infrastructure	Upgrading drainage systems	2022-04-30
5	5	Evacuation	Establishment of evacuation routes	2022-05-10
6	6	Monitoring	Implementing real-time monitoring	2022-06-15
7	7	Preparedness	Training emergency response teams	2022-07-20
8	8	Communication	Enhancing public communication channels	2022-08-25
9	9	Equipment	Procurement of emergency supplies	2022-09-05
10	10	Technology	Implementing GIS-based flood modeling	2022-10-10

28) Retrieve residents' contact information for low elevation areas:

SELECT * FROM Resident

WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* **SQL File 7*** x

Limit to 1000 rows

```

1 •   SELECT * FROM Resident
2 WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation < 15);
3

```

Result Grid

ResidentID	Name	Address	ContactNumber	LocationID
1	John Doe	123 Main St, City	123-456-7890	1
6	Michael Lee	901 Cedar Ln, County	901-234-5678	6
HULL	HULL	HULL	HULL	HULL

29) Retrieve authorities responsible for issuing high alert early warning alerts:

SELECT * FROM Authority

WHERE LocationID IN (

SELECT LocationID

FROM EarlyWarningAlert

```
WHERE AlertLevel = 'High'  
);
```

The screenshot shows a SQL query window titled "Query 1" with the following code:

```
3   SELECT LocationID  
4   FROM EarlyWarningAlert  
5   WHERE AlertLevel = 'High'  
6   );  
7
```

Below the query window is a "Result Grid" showing the following data:

	AuthorityID	Name	Organization	ContactNumber	LocationID
▶	1	Mark Anderson	City Emergency Dept	111-222-3333	1
	3	James Wilson	Township Fire Dept	777-888-9999	3
	6	Emma Johnson	Borough Rescue Squad	666-777-8888	6
*	9	David White	County Emergency	555-666-7777	9
	NULL	NULL	NULL	NULL	NULL

30) Retrieve flood data for a specific location and date range:

```
SELECT * FROM FloodData
```

```
WHERE LocationID = 1
```

```
AND Date BETWEEN '2023-01-01' AND '2023-12-31';
```

The screenshot shows a SQL query window titled "Query 1" with the following code:

```
1 •  SELECT * FROM FloodData  
2 WHERE LocationID = 1  
3 AND Date BETWEEN '2023-01-01' AND '2023-12-31';  
4
```

Below the query window is a "Result Grid" showing the following data:

	FloodDataID	LocationID	Date	Magnitude	Duration
	NULL	NULL	NULL	NULL	NULL

31) Retrieve the highest magnitude flood recorded:

```
SELECT MAX(Magnitude) AS HighestMagnitude FROM FloodData;
```

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top: Query 1, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, and SQL File 7*. The SQL File 7* tab is active. Below the tabs is a toolbar with various icons. The main area contains two numbered lines of SQL code:

```
1  SELECT MAX(Magnitude) AS HighestMagnitude FROM FloodData;
2  |
```

Below the code is a result grid with one row labeled "HighestMagnitude". The value in the grid is "5.00".

32) Retrieve preventive measures with implementation dates within the last year:

```
SELECT * FROM PreventiveMeasure
```

```
WHERE ImplementationDate >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR);
```

The screenshot shows a SQL Server Management Studio window with multiple tabs at the top: Query 1, SQL File 3*, SQL File 4*, SQL File 5*, SQL File 6*, and SQL File 7*. The SQL File 7* tab is active. Below the tabs is a toolbar with various icons. The main area contains three numbered lines of SQL code:

```
1 •  SELECT * FROM PreventiveMeasure
2  WHERE ImplementationDate >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
3  |
```

Below the code is a result grid with columns: MeasureID, LocationID, MeasureType, Description, and ImplementationDate. All values in the grid are NULL.

33) Retrieve residents' contact information for regions with flood occurrences:

```
SELECT DISTINCT r.Name, r.Address, r.ContactNumber
```

```
FROM Resident r
```

```
JOIN Location l ON r.LocationID = l.LocationID
```

```
JOIN FloodData f ON l.LocationID = f.LocationID;
```

The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor displays the following SQL query:

```

1 •  SELECT DISTINCT r.Name, r.Address, r.ContactNumber
2   FROM Resident r
3   JOIN Location l ON r.LocationID = l.LocationID
4   JOIN FloodData f ON l.LocationID = f.LocationID;
5

```

Below the code editor is a "Result Grid" section. It contains a table with three columns: "Name", "Address", and "ContactNumber". The table displays the following data:

Name	Address	ContactNumber
John Doe	123 Main St, City	123-456-7890
Jane Smith	456 Oak Ave, Town	456-789-0123
David Johnson	789 Elm Rd, Village	789-012-3456
Sarah Brown	321 Pine Blvd, Suburb	321-654-9870
Emily Wilson	567 Maple Dr, Hamlet	567-890-1234
Michael Lee	901 Cedar Ln, County	901-234-5678
Laura Miller	234 Birch St, Township	234-567-8901
Kevin Clark	678 Walnut Ave, Parish	678-901-2345
Amanda White	890 Spruce Rd, Borough	890-123-4567
Eric Taylor	345 Oakwood Dr, Manor	345-678-9012

34) Count the number of floods per year:

```

SELECT YEAR(Date) AS Year, COUNT(*) AS FloodCount
FROM FloodData
GROUP BY YEAR(Date);

```

The screenshot shows a SQL query editor interface. At the top, there is a toolbar with various icons. Below the toolbar, a code editor displays the following SQL query:

```

1 •  SELECT YEAR(Date) AS Year, COUNT(*) AS FloodCount
2   FROM FloodData
3   GROUP BY YEAR(Date);
4

```

Below the code editor is a "Result Grid" section. It contains a table with two columns: "Year" and "FloodCount". The table displays the following data:

Year	FloodCount
2022	10

35) Retrieve authorities for regions with flood occurrences and high alert levels:

```

SELECT DISTINCT a.Name, a.Organization
FROM Authority a
JOIN Location l ON a.LocationID = l.LocationID
JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID
WHERE e.AlertLevel = 'High';

```

The screenshot shows a SQL query window with the following code:

```

2   FROM Authority a
3   JOIN Location l ON a.LocationID = l.LocationID
4   JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID
5   WHERE e.AlertLevel = 'High';
6

```

Below the code is a result grid with the following data:

	Name	Organization
▶	Mark Anderson	City Emergency Dept
	James Wilson	Township Fire Dept
	Emma Johnson	Borough Rescue Squad
	David White	County Emergency

36) Retrieve emergency response plans for regions with flood occurrences and high alert levels:

SELECT DISTINCT er.*

FROM EmergencyResponse er

JOIN Location l ON er.LocationID = l.LocationID

JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID

WHERE e.AlertLevel = 'High';

The screenshot shows a SQL query window with the following code:

```

2   FROM EmergencyResponse er
3   JOIN Location l ON er.LocationID = l.LocationID
4   JOIN EarlyWarningAlert e ON l.LocationID = e.LocationID
5   WHERE e.AlertLevel = 'High';
6

```

Below the code is a result grid with the following data:

	PlanID	LocationID	PlanDescription	ActivationDate
▶	1	1	Emergency Evacuation Plan for City	2022-01-01
	3	3	Town Emergency Response Plan	2022-03-10
	6	6	County Emergency Action Plan	2022-06-25
	9	9	Borough Contingency Plan	2022-09-10

37) Retrieve evacuation centers with available capacity for a specific region:

SELECT ec.*

FROM EvacuationCenter ec

JOIN Location l ON ec.LocationID = l.LocationID

WHERE l.Region = 'New York'

AND ec.Capacity > 0;

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled 'Query 1', 'SQL File 3*', 'SQL File 4*', 'SQL File 5*', 'SQL File 6*', and 'SQL File 7*'. Below the tabs is a toolbar with various icons. The main area contains a numbered SQL query:

```
1 •  SELECT ec.*  
2   FROM EvacuationCenter ec  
3   JOIN Location l ON ec.LocationID = l.LocationID  
4   WHERE l.Region = 'New York'  
5   AND ec.Capacity > 0;
```

Below the query is a result grid with the following columns: CenterID, LocationID, CenterName, Capacity, and LocationDetails. The data row shown is:

	CenterID	LocationID	CenterName	Capacity	LocationDetails
▶	1	1	City Evacuation Center	500	City Hall

38) Retrieve flood data for regions with elevation less than 20 and magnitude greater than 2:

SELECT f.*

FROM FloodData f

JOIN Location l ON f.LocationID = l.LocationID

WHERE l.Elevation < 20

AND f.Magnitude > 2;

The screenshot shows a SQL query editor interface with multiple tabs at the top labeled 'Query 1', 'SQL File 3*', 'SQL File 4*', 'SQL File 5*', 'SQL File 6*', and 'SQL File 7*'. Below the tabs is a toolbar with various icons. The main area contains a numbered SQL query:

```
2   FROM FloodData f  
3   JOIN Location l ON f.LocationID = l.LocationID  
4   WHERE l.Elevation < 20  
5   AND f.Magnitude > 2;  
6
```

Below the query is a result grid with the following columns: FloodDataID, LocationID, Date, Magnitude, and Duration. The data rows shown are:

	FloodDataID	LocationID	Date	Magnitude	Duration
▶	1	1	2022-01-05	4.00	48
	2	2	2022-02-12	3.00	36
	6	6	2022-06-10	3.00	48
	8	8	2022-08-22	3.50	24

39)FINAL QUERY:

-- Query 1: Analyze historical flood data

```
SELECT LocationID, AVG(Magnitude) AS AverageMagnitude, MAX(Magnitude)  
AS MaxMagnitude, MIN(Magnitude) AS MinMagnitude,  
AVG(Duration) AS AverageDuration, COUNT(*) AS TotalFloods  
FROM FloodData  
GROUP BY LocationID;
```

-- Query 2: Determine areas with the highest flood risk

```
SELECT LocationID, Region, Elevation, AVG(Magnitude) AS AverageMagnitude  
FROM Location  
JOIN FloodData ON Location.LocationID = FloodData.LocationID  
GROUP BY LocationID  
HAVING AVG(Magnitude) > 4; -- Example threshold for high flood risk
```

-- Query 3: Generate early warning alerts

```
SELECT LocationID, DateIssued, AlertLevel, RecommendedActions  
FROM EarlyWarningAlert  
WHERE DateIssued >= CURDATE()  
ORDER BY DateIssued;
```

-- Query 4: Retrieve preventive measures

```
SELECT LocationID, MeasureType, Description, ImplementationDate  
FROM PreventiveMeasure  
WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation  
< 15);
```

-- Query 5: Retrieve emergency response plans

```
SELECT LocationID, PlanDescription, ActivationDate  
FROM EmergencyResponse  
WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation  
< 15);
```

-- Query 6: Identify evacuation centers

```
SELECT LocationID, CenterName, Capacity, LocationDetails  
FROM EvacuationCenter  
WHERE LocationID IN (SELECT LocationID FROM Location WHERE Elevation  
< 15);
```

OUTPUT:

Result Grid | Filter Rows: Export: Wrap Cell Content:

	LocationID	AverageMagnitude	MaxMagnitude	MinMagnitude	AverageDuration	TotalFloods
▶	1	4.000000	4.00	4.00	48.0000	1
	2	3.000000	3.00	3.00	36.0000	1
	3	5.000000	5.00	5.00	72.0000	1
	4	3.500000	3.50	3.50	24.0000	1
	5	4.500000	4.50	4.50	60.0000	1
	6	3.000000	3.00	3.00	48.0000	1
	7	4.000000	4.00	4.00	36.0000	1
	8	3.500000	3.50	3.50	24.0000	1
	9	5.000000	5.00	5.00	72.0000	1
	10	4.500000	4.50	4.50	60.0000	1

The SQL queries aim to analyze historical flood data, determine areas with high flood risk, generate early warning alerts, retrieve preventive measures, retrieve emergency response plans, and identify evacuation centers based on location.

-- Analyzing Historical Flood Data

```
SELECT l.Region AS Location, AVG(fd.Magnitude) AS AverageMagnitude
FROM Location l
JOIN FloodData fd ON l.LocationID = fd.LocationID
GROUP BY l.Region;
```

-- Determining Areas with High Flood Risk

```
SELECT l.Region AS Location, AVG(fd.Magnitude) AS AverageMagnitude
FROM Location l
JOIN FloodData fd ON l.LocationID = fd.LocationID
GROUP BY l.Region
HAVING AVG(fd.Magnitude) > 4;
```

-- Generating Early Warning Alerts

```
SELECT l.Region AS Location, ewa.*
FROM Location l
JOIN EarlyWarningAlert ewa ON l.LocationID = ewa.LocationID
WHERE l.LocationID = 1; -- Example: LocationID 1
```

-- Retrieving Preventive Measures

```
SELECT l.Region AS Location, pm.*
FROM Location l
JOIN PreventiveMeasure pm ON l.LocationID = pm.LocationID
WHERE l.LocationID = 1; -- Example: LocationID 1
```

-- Retrieving Emergency Response Plans

```
SELECT l.Region AS Location, er.*
FROM Location l
JOIN EmergencyResponse er ON l.LocationID = er.LocationID
WHERE l.LocationID = 1; -- Example: LocationID 1
```

-- Identifying Evacuation Centers

```
SELECT l.Region AS Location, ec.*  
FROM Location l  
JOIN EvacuationCenter ec ON l.LocationID = ec.LocationID  
WHERE l.LocationID = 1; -- Example: LocationID 1
```

	Location	CenterID	LocationID	CenterName	Capacity	LocationDetails
▶	New York	1	1	City Evacuation Center	500	City Hall

40) Query to Retrieve Historical Flood Data:

```
SELECT * FROM FloodData;
```

	FloodDataID	LocationID	Date	Magnitude	Duration
▶	1	1	2022-01-05	4.00	48
2	2	2	2022-02-12	3.00	36
3	3	3	2022-03-20	5.00	72
4	4	4	2022-04-18	3.50	24
5	5	5	2022-05-25	4.50	60
6	6	6	2022-06-10	3.00	48
7	7	7	2022-07-15	4.00	36
8	8	8	2022-08-22	3.50	24
9	9	9	2022-09-30	5.00	72
10	10	10	2022-10-18	4.50	60
*	NULL	NULL	NULL	NULL	NULL

FINAL RESULT OF THIS PROJECT:

-- 1. Analyzing Historical Flood Data for All Cities

```
SELECT fd.*, l.Region AS Location  
FROM FloodData fd  
JOIN Location l ON fd.LocationID = l.LocationID;
```

-- 2. Average Magnitude of Floods in Each Region

```
SELECT l.Region AS Location, AVG(fd.Magnitude) AS AverageMagnitude
```

```
FROM Location l  
JOIN FloodData fd ON l.LocationID = fd.LocationID  
GROUP BY l.Region;
```

```
-- 3. Early Warning Alerts for All Cities  
SELECT ewa.*, l.Region AS Location  
FROM EarlyWarningAlert ewa  
JOIN Location 1 ON ewa.LocationID = l.LocationID;
```

```
-- 4. Preventive Measures for All Cities  
SELECT pm.*, l.Region AS Location  
FROM PreventiveMeasure pm  
JOIN Location 1 ON pm.LocationID = l.LocationID;
```

```
-- 5. Emergency Response Plans for All Cities  
SELECT er.*, l.Region AS Location  
FROM EmergencyResponse er  
JOIN Location 1 ON er.LocationID = l.LocationID;
```

```
-- 6. Evacuation Centers for All Cities  
SELECT ec.*, l.Region AS Location  
FROM EvacuationCenter ec  
JOIN Location 1 ON ec.LocationID = l.LocationID;
```

```
-- 7. Query 7 Description  
SELECT * FROM YourTableName WHERE YourCondition;
```

```
-- 8. Retrieve Emergency Response Logs for All Cities  
SELECT erl.*, l.Region AS Location  
FROM EmergencyResponseLog erl  
JOIN Location 1 ON erl.LocationID = l.LocationID;
```

```
-- 9. Retrieve Authorities for All Cities  
SELECT a.*, l.Region AS Location  
FROM Authority a  
JOIN Location 1 ON a.LocationID = l.LocationID;
```

```
-- 10. Flood Data for All Cities  
SELECT fd.*, l.Region AS Location  
FROM FloodData fd  
JOIN Location 1 ON fd.LocationID = l.LocationID;
```

OUTPUT:

	FloodDataID	LocationID	Date	Magnitude	Duration	Location
▶	1	1	2022-01-05	4.00	48	New York
	2	2	2022-02-12	3.00	36	Los Angeles
	3	3	2022-03-20	5.00	72	London
	4	4	2022-04-18	3.50	24	Paris
	5	5	2022-05-25	4.50	60	Tokyo
	6	6	2022-06-10	3.00	48	San Francisco
	7	7	2022-07-15	4.00	36	Madrid
	8	8	2022-08-22	3.50	24	Athens
	9	9	2022-09-30	5.00	72	Rome
	10	10	2022-10-18	4.50	60	Moscow

[Result 26](#) × [Result 27](#) [Result 28](#) [Result 29](#) [Result 30](#) [Result 31](#)

	Location	AverageMagnitude
▶	New York	4.000000
	Los Angeles	3.000000
	London	5.000000
	Paris	3.500000
	Tokyo	4.500000
	San Francisco	3.000000
	Madrid	4.000000
	Athens	3.500000
	Rome	5.000000
	Moscow	4.500000

	AlertID	LocationID	DateIssued	AlertLevel	RecommendedActions	Location
▶	1	1	2022-01-03	High	Evacuate	New York
	2	2	2022-02-10	Moderate	Stay Alert	Los Angeles
	3	3	2022-03-18	High	Prepare	London
	4	4	2022-04-15	Low	Monitor	Paris
	5	5	2022-05-22	Moderate	Prepare	Tokyo
	6	6	2022-06-08	High	Evacuate	San Francisco
	7	7	2022-07-13	Low	Monitor	Madrid
	8	8	2022-08-20	Moderate	Prepare	Athens
	9	9	2022-09-28	High	Evacuate	Rome
	10	10	2022-10-15	Low	Monitor	Moscow

	MeasureID	LocationID	MeasureType	Description	ImplementationDate	Location
▶	1	1	Structural	Reinforcement of riverbanks	2022-01-15	New York
	2	2	Early Warning	Installation of flood sensors	2022-02-20	Los Angeles
	3	3	Community	Conducting awareness campaigns	2022-03-25	London
	4	4	Infrastructure	Upgrading drainage systems	2022-04-30	Paris
	5	5	Evacuation	Establishment of evacuation routes	2022-05-10	Tokyo
	6	6	Monitoring	Implementing real-time monitoring	2022-06-15	San Francisco
	7	7	Preparedness	Training emergency response teams	2022-07-20	Madrid
	8	8	Communication	Enhancing public communication channels	2022-08-25	Athens
	9	9	Equipment	Procurement of emergency supplies	2022-09-05	Rome
	10	10	Technology	Implementing GIS-based flood modeling	2022-10-10	Moscow

Result Grid | Filter Rows: Export: Wrap Cell Content:

	PlanID	LocationID	PlanDescription	ActivationDate	Location
▶	1	1	Emergency Evacuation Plan for City	2022-01-01	New York
	2	2	Disaster Preparedness Plan for County	2022-02-05	Los Angeles
	3	3	Town Emergency Response Plan	2022-03-10	London
	4	4	Parish Crisis Management Plan	2022-04-15	Paris
	5	5	Hamlet Disaster Recovery Plan	2022-05-20	Tokyo
	6	6	County Emergency Action Plan	2022-06-25	San Francisco
	7	7	Township Disaster Management Plan	2022-07-30	Madrid
	8	8	Village Emergency Preparedness Strategy	2022-08-05	Athens
	9	9	Borough Contingency Plan	2022-09-10	Rome
	10	10	Manor Disaster Response Framework	2022-10-15	Moscow

Result Grid | Filter Rows: Export: Wrap Cell Content:

	CenterID	LocationID	CenterName	Capacity	LocationDetails	Location
▶	1	1	City Evacuation Center	500	City Hall	New York
	2	2	County Shelter	300	Community Center	Los Angeles
	3	3	Town Emergency Center	200	Town Hall	London
	4	4	Parish Safe Haven	400	Parish School	Paris
	5	5	Hamlet Evacuation Site	250	Hamlet Park	Tokyo
	6	6	County Relief Center	350	County Fairgrounds	San Francisco
	7	7	Township Refuge	150	Township Library	Madrid
	8	8	Village Shelter	280	Village Hall	Athens
	9	9	Borough Emergency Hub	180	Borough Community Center	Rome
	10	10	Manor Safe Zone	200	Manor Civic Center	Moscow

RESULT:

Thus, working with queries related to project on MySQL is successfully verified.

<u>Ex- 13</u>	<u>PROJECT REPORT ANALYSIS</u>
--------------------------	---------------------------------------

1. Introduction:

Flooding is a recurring natural disaster that poses significant threats to lives and properties worldwide. In response to the escalating risks associated with flooding incidents, the development of an advanced Flood Risk Prediction and Early Warning System is imperative. This project aims to address this need by leveraging data integration, predictive modeling, and advanced technologies to provide timely and accurate flood risk predictions and early warning alerts.

2. Objective:

The primary objective of the Flood Risk Prediction and Early Warning System project is to create a comprehensive platform that integrates historical flood data, terrain particulars, climate models, and predictive analytics to accurately forecast flood risks. The system will deliver proactive early warning alerts to residents and authorities, enabling them to take preemptive measures and mitigate potential damages effectively. Key objectives include data integration, database design and management, flood risk prediction, early warning system implementation, user-friendly interface development, and scalability and reliability enhancement.

3. Technologies Used:

- Programming Languages: Python, JavaScript
- Frameworks: Flask (Backend), React (Frontend)
- Database Management System: PostgreSQL
- Data Analysis and Modeling: Pandas, Scikit-learn
- Geographic Information System (GIS): ArcGIS, QGIS
- Cloud Services: AWS, Google Cloud Platform
- Security: SSL/TLS encryption, Authentication, Authorization

4. Features:

- **Data Integration:** Integration of historical flood data, terrain particulars, and climate models.
- **Flood Risk Prediction:** Utilization of predictive modeling techniques to forecast flood risks.
- **Early Warning System:** Provision of punctual and dependable early warning alerts to residents and authorities.

- **User-friendly Interface:** Development of an intuitive and easy-to-navigate interface for users.
- **Scalability and Reliability:** Implementation of scalable and reliable infrastructure to handle varying loads and ensure system availability.

5. Database Structure:

The database structure includes tables for storing historical flood data, terrain particulars, climate models, user information, early warning alerts, and system logs. Relationships between tables are established using primary and foreign keys to ensure data integrity and consistency.

6. Implementation:

- Frontend: Development of a responsive and intuitive frontend interface using React.js.
- Backend: Implementation of backend logic and APIs using Flask framework to handle data processing and communication with the frontend.
- Database Integration: Integration of PostgreSQL database to store and manage data efficiently.
- Security Measures: Implementation of SSL/TLS encryption, user authentication, and authorization mechanisms to ensure data security and user privacy.
- Maintenance: Regular monitoring, updates, and maintenance to ensure system reliability and performance optimizations.

7. Conclusion:

The Flood Risk Prediction and Early Warning System project aims to provide a comprehensive solution to address the challenges posed by flooding incidents. By integrating historical data, predictive analytics, and advanced technologies, the system offers proactive measures to mitigate risks and enhance community resilience. With its user-friendly interface and robust features, the system contributes significantly to reducing property damage and saving lives in flood-prone regions.

8. Future Scope and Enhancements:

- Integration of real-time data sources for more accurate predictions.
- Implementation of machine learning algorithms for adaptive forecasting.
- Enhancement of visualization tools for better data interpretation.
- Collaboration with government agencies and disaster management organizations for broader implementation and impact.

SAMPLE FRONTEND:

Flood Risk Prediction and Early Warning System

User and Location Details

Name:

Address:

Contact Number:

Region:

Elevation:

Submit

Flood Risk Prediction and Early Warning System

Name:

Address:

Contact Number:

Select Region:

Select Region



Select Region

- New York
- Los Angeles
- London
- Paris
- Tokyo
- San Francisco
- Madrid
- Athens
- Rome
- Moscow

Flood Risk Prediction and Early Warning System

Name:

MONISHAAS

Address:

HAL 2nd stage

Contact Number:

7411387492

Select Region:

Los Angeles

Submit

Warning: High alert level. Evacuate immediately.

Evacuation Centers:

Evacuation Center 1

Capacity: 500

Rangesh Suraj

Address:

Chennai

Contact Number:

8825714465

Select Region:

New York

Submit

Warning: High alert level. Evacuate immediately.

Evacuation Centers:

Evacuation Center 1

Capacity: 500

Location: City Hall

Evacuation Center 2

Capacity: 300

Location: Community Center

- The user fills out the form with their details and selects the region from the dropdown.
- Upon submission, the system fetches the alert details based on the selected region.
- If the alert level is "High," a warning message is displayed along with the recommended actions.

- Evacuation centers are displayed below the warning message, providing users with information on where to evacuate in case of a high alert level.
- The design is clean and user-friendly, with proper alignment and color scheme to enhance readability and usability.