PROJECT

DISTRIBUTED DENIAL OF SERVICE USING MYSQL RELATIONAL DATABASE STRUCTURE BASED ON NETWORK SECURITY

Prepared By:	**Guided By:**
Akshat Chaudhary	**Zakir Hussain**

TABLE OF CONTENTS

1) Distributed Denial of Service (DDoS):

Distributed Denial-of-Service (DDoS) attack is a type of cyberattack where an attacker attempts to make a computer or network resource unavailable by overwhelming it with traffic from multiple sources. This is typically done by using a network of compromised devices (bots) to flood the targeted system with traffic, causing it to become overwhelmed and unable to handle legitimate requests.

Here's how a DDoS attack works:

1) **Botnet Creation:** Attackers makes use of vulnerabilities in devices (like computers, IoT devices, etc.) to install malware, forming a botnet—a network of compromised machines controlled remotely.

2) **Traffic Generation:** Using this botnet, the attacker commands each bot to start sending requests, usually in a coordinated manner, to a target system (such as a server or network).

3) **Traffic Flood:** The compromised devices generate a massive amount of fake traffic or requests, flooding the target with far more data than it can handle. This traffic may consume bandwidth, processing power, or both.

4) **System Overload:** The target system, unable to distinguish legitimate traffic from the flood of malicious requests, becomes overwhelmed, leading to slowdowns or complete service interruptions.

Types of DDoS attacks:

1) **Volume-based attacks:**

Goal: Flood the target with a massive amount of traffic to saturate its bandwidth.

Effect: The target’s internet connection gets overwhelmed, making it impossible for legitimate traffic to get through.

2) **Protocol attacks:**

Goal: Exploit weaknesses in network protocols to deplete system resources (e.g., CPU, memory).

Effect: The target system’s resources get exhausted as it tries to handle malformed or excessive protocol-level requests.

3) **Application-layer attacks:**

Goal: Target specific applications or services running on the server, often mimicking legitimate user behavior.

Effect: Consumes the resources of the targeted application, making it unresponsive or slow for legitimate users.

DDoS attacks can be launched using various techniques, including:

1) **Botnets:**

Attackers build or rent botnets made up of compromised devices (like computers, IoT devices, routers) that are controlled remotely. These bots are commanded to send large volumes of traffic to a target, overwhelming it.

2) **Malware:**

Malware is used to infect devices, turning them into bots that can be controlled by the attacker. Common malware types include Trojans and worms, which are often used to gain unauthorized control over devices.

3) **Scripting:**

Attackers can use scripting languages (like Python, Perl, or Bash) to automate attack processes. These scripts can send a high number of requests to the target in an automated fashion, making the attack more efficient and scalable.

4) **Amplification Attacks:**

In an amplification attack, the attacker sends small requests to open services like DNS or NTP, which then reply with large responses to the target, amplifying the amount of traffic the victim receives. Examples include DNS amplification and NTP reflection attacks.

To protect against DDoS attacks, organizations can use:

1) **Firewalls:**

Firewalls act as a barrier between the internal network and the internet. They filter traffic by enforcing security rules, allowing only legitimate requests through while blocking suspicious or malicious traffic.

2) **Intrusion Detection/Prevention Systems (IDS/IPS):**

IDS monitors traffic for signs of an attack and alerts administrators when suspicious activity is detected.

IPS takes it a step further by actively blocking or mitigating malicious traffic in real-time, helping to stop attacks before they cause harm.

3) **Load Balancing:**

Load balancers distribute incoming traffic across multiple servers, helping to prevent any single server from becoming overwhelmed. This approach can also reroute traffic in the event of an attack, ensuring availability.

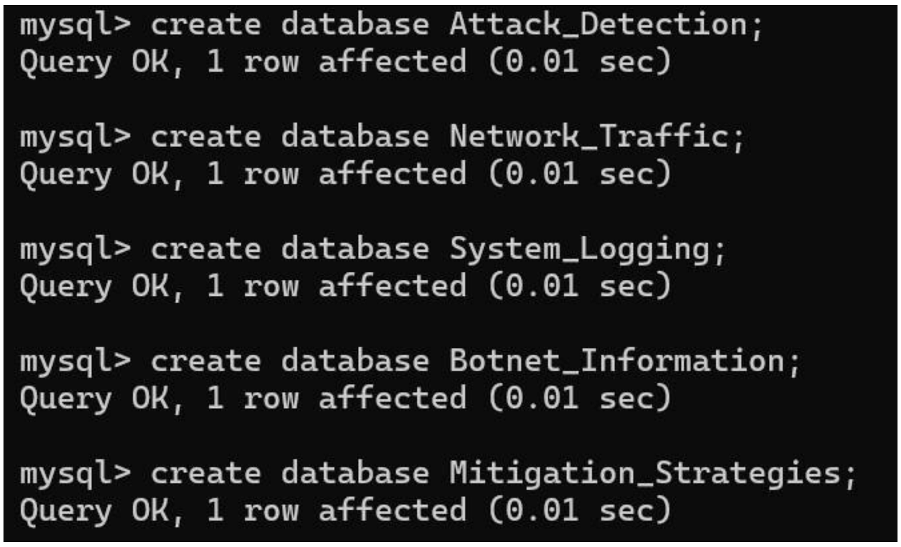
4) **Content Delivery Networks (CDNs):**

CDNs store cached copies of website content in multiple geographical locations. By distributing requests across their network, they reduce the load on the main server, absorb attack traffic, and ensure continuous service availability.

5) **DDoS Mitigation Services:**

Specialized services (such as Cloudflare, AWS Shield, or Akamai) are designed to detect and mitigate DDoS attacks. These services filter malicious traffic, absorb the excess load, and ensure that only legitimate requests reach the server.

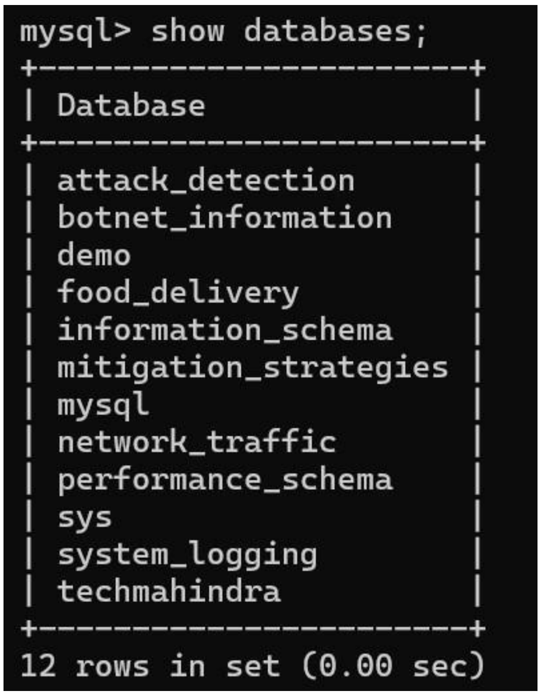
2) Databases used in this Project: - Create five database using the below syntax: create database [name of database];



```
mysql> create database Attack_Detection;
Query OK, 1 row affected (0.01 sec)

mysql> create database Network_Traffic;
Query OK, 1 row affected (0.01 sec)

mysql> create database System_Logging;
Query OK, 1 row affected (0.01 sec)

mysql> create database Botnet_Information;
Query OK, 1 row affected (0.01 sec)

mysql> create database Mitigation_Strategies;
Query OK, 1 row affected (0.01 sec)
```

- To display the names of created databases:

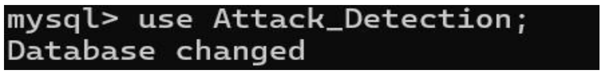
show databases;



```
mysql> show databases;
+-----------------------+
| Database              |
+-----------------------+
| attack_detection      |
| botnet_information    |
| demo                  |
| food_delivery         |
| information_schema    |
| mitigation_strategies |
| mysql                 |
| network_traffic       |
| performance_schema    |
| sys                   |
| system_logging        |
| techmahindra          |
+-----------------------+
12 rows in set (0.00 sec)
```

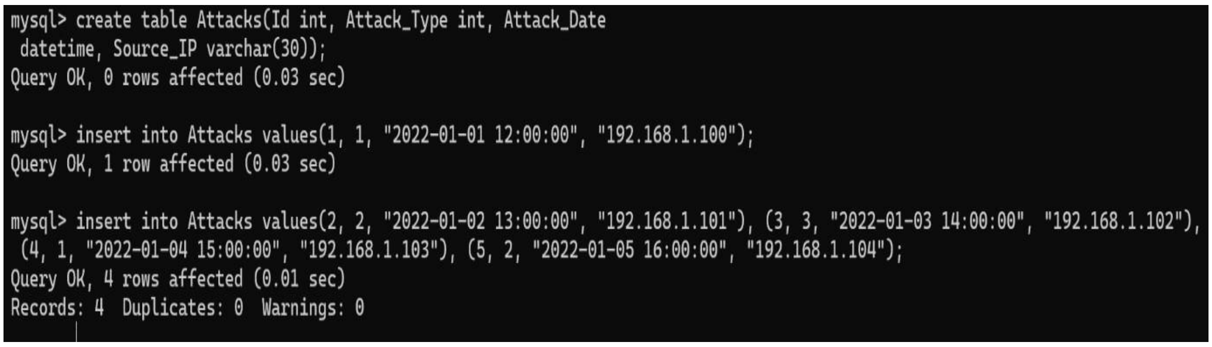
3) Tables used in each of the Databases:

Using **first database**, named as **‘Attack_Detection’**:



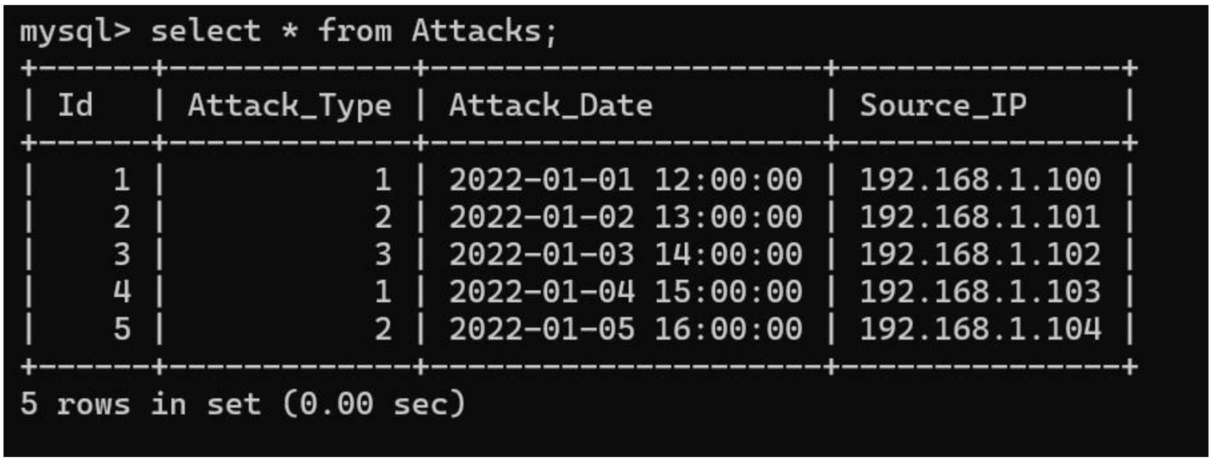
```
mysql> use Attack_Detection;
Database changed
```

- Creating first table, named as ‘Attacks’:



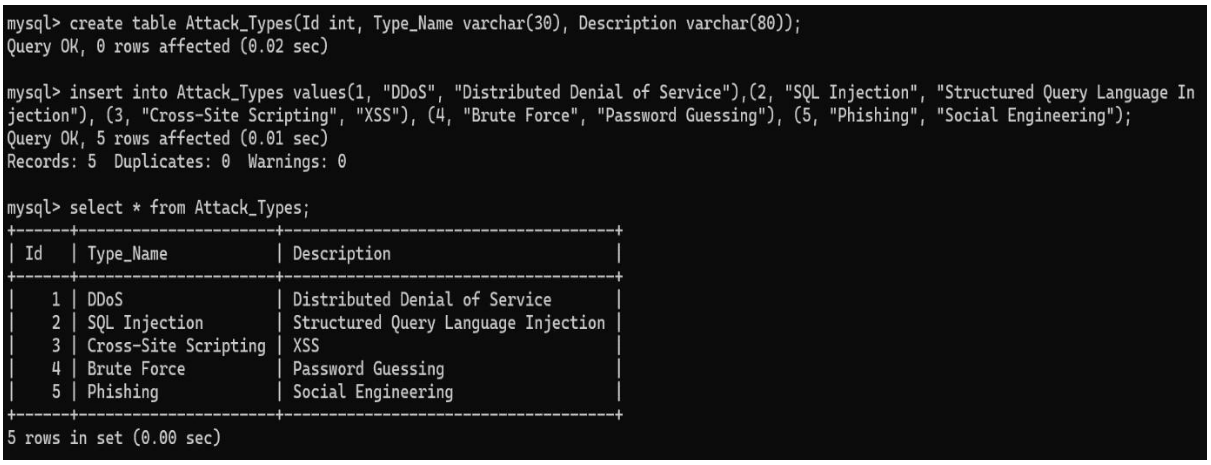
```
mysql> create table Attacks(Id int, Attack_Type int, Attack_Date
 datetime, Source_IP varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Attacks values(1, 1, "2022-01-01 12:00:00", "192.168.1.100");
Query OK, 1 row affected (0.03 sec)

mysql> insert into Attacks values(2, 2, "2022-01-02 13:00:00", "192.168.1.101"), (3, 3, "2022-01-03 14:00:00", "192.168.1.102"),
 (4, 1, "2022-01-04 15:00:00", "192.168.1.103"), (5, 2, "2022-01-05 16:00:00", "192.168.1.104");
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Displaying the entire table:



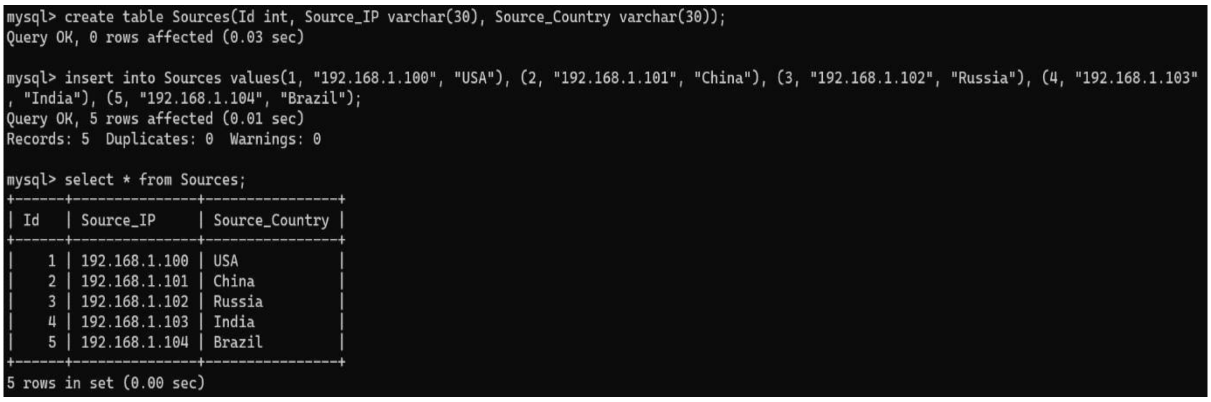
```
mysql> select * from Attacks;
+------+-------------+---------------------+---------------+
| Id   | Attack_Type | Attack_Date         | Source_IP     |
+------+-------------+---------------------+---------------+
|    1 |           1 | 2022-01-01 12:00:00 | 192.168.1.100 |
|    2 |           2 | 2022-01-02 13:00:00 | 192.168.1.101 |
|    3 |           3 | 2022-01-03 14:00:00 | 192.168.1.102 |
|    4 |           1 | 2022-01-04 15:00:00 | 192.168.1.103 |
|    5 |           2 | 2022-01-05 16:00:00 | 192.168.1.104 |
+------+-------------+---------------------+---------------+
5 rows in set (0.00 sec)
```

- Creating second table, named as ‘Attack_Types’ and displaying it:



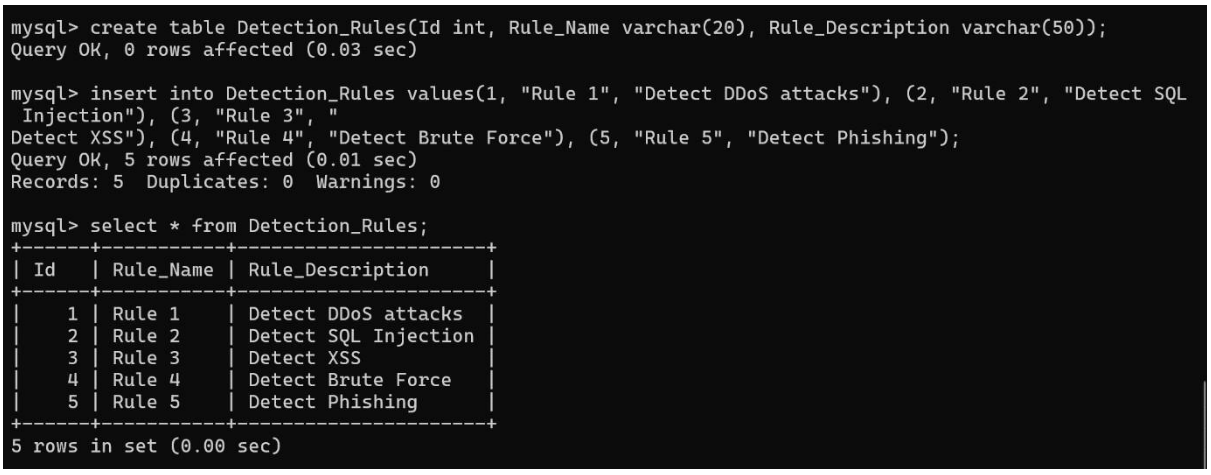
```
mysql> create table Attack_Types(Id int, Type_Name varchar(30), Description varchar(80));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into Attack_Types values(1, "DDoS", "Distributed Denial of Service"),(2, "SQL Injection", "Structured Query Language In
jection"), (3, "Cross-Site Scripting", "XSS"), (4, "Brute Force", "Password Guessing"), (5, "Phishing", "Social Engineering");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Attack_Types;
+------+----------------------+-----------------------------------+
| Id   | Type_Name            | Description                       |
+------+----------------------+-----------------------------------+
|    1 | DDoS                 | Distributed Denial of Service     |
|    2 | SQL Injection        | Structured Query Language Injection |
|    3 | Cross-Site Scripting | XSS                               |
|    4 | Brute Force          | Password Guessing                 |
|    5 | Phishing             | Social Engineering                |
+------+----------------------+-----------------------------------+
5 rows in set (0.00 sec)
```

- Creating third table, named as ‘Sources’, and displaying it:



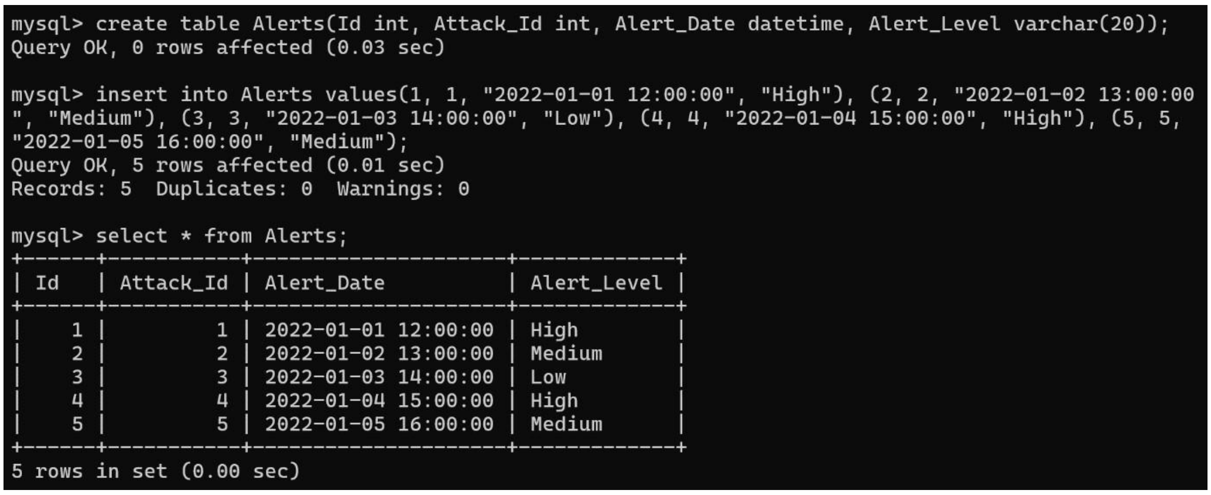
```
mysql> create table Sources(Id int, Source_IP varchar(30), Source_Country varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Sources values(1, "192.168.1.100", "USA"), (2, "192.168.1.101", "China"), (3, "192.168.1.102", "Russia"), (4, "192.168.1.103"
, "India"), (5, "192.168.1.104", "Brazil");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Sources;
+------+---------------+----------------+
| Id   | Source_IP     | Source_Country |
+------+---------------+----------------+
|    1 | 192.168.1.100 | USA            |
|    2 | 192.168.1.101 | China          |
|    3 | 192.168.1.102 | Russia         |
|    4 | 192.168.1.103 | India          |
|    5 | 192.168.1.104 | Brazil         |
+------+---------------+----------------+
5 rows in set (0.00 sec)
```

- Creating Fourth table, named as ‘Detection_Rules’ and displaying it:



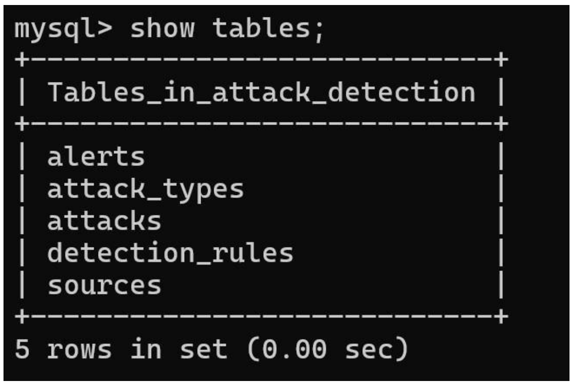
```
mysql> create table Detection_Rules(Id int, Rule_Name varchar(20), Rule_Description varchar(50));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Detection_Rules values(1, "Rule 1", "Detect DDoS attacks"), (2, "Rule 2", "Detect SQL
 Injection"), (3, "Rule 3", "
Detect XSS"), (4, "Rule 4", "Detect Brute Force"), (5, "Rule 5", "Detect Phishing");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Detection_Rules;
+------+-----------+----------------------+
| Id   | Rule_Name | Rule_Description     |
+------+-----------+----------------------+
|    1 | Rule 1    | Detect DDoS attacks  |
|    2 | Rule 2    | Detect SQL Injection |
|    3 | Rule 3    | Detect XSS           |
|    4 | Rule 4    | Detect Brute Force   |
|    5 | Rule 5    | Detect Phishing      |
+------+-----------+----------------------+
5 rows in set (0.00 sec)
```

- Creating Fifth table, named as ‘Alerts’ and displaying it:



```
mysql> create table Alerts(Id int, Attack_Id int, Alert_Date datetime, Alert_Level varchar(20));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Alerts values(1, 1, "2022-01-01 12:00:00", "High"), (2, 2, "2022-01-02 13:00:00
", "Medium"), (3, 3, "2022-01-03 14:00:00", "Low"), (4, 4, "2022-01-04 15:00:00", "High"), (5, 5,
"2022-01-05 16:00:00", "Medium");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Alerts;
+------+-----------+---------------------+-------------+
| Id   | Attack_Id | Alert_Date          | Alert_Level |
+------+-----------+---------------------+-------------+
|    1 |         1 | 2022-01-01 12:00:00 | High        |
|    2 |         2 | 2022-01-02 13:00:00 | Medium      |
|    3 |         3 | 2022-01-03 14:00:00 | Low         |
|    4 |         4 | 2022-01-04 15:00:00 | High        |
|    5 |         5 | 2022-01-05 16:00:00 | Medium      |
+------+-----------+---------------------+-------------+
5 rows in set (0.00 sec)
```

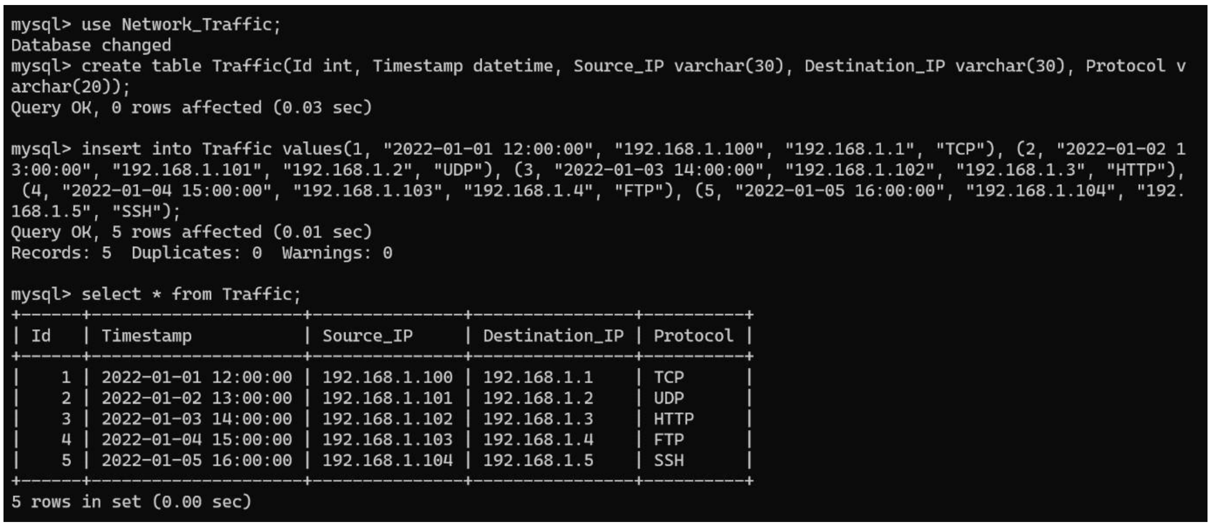
To list the tables available in database ‘Attack_Detection’:



```
mysql> show tables;
+----------------------------+
| Tables_in_attack_detection |
+----------------------------+
| alerts                     |
| attack_types               |
| attacks                    |
| detection_rules            |
| sources                    |
+----------------------------+
5 rows in set (0.00 sec)
```

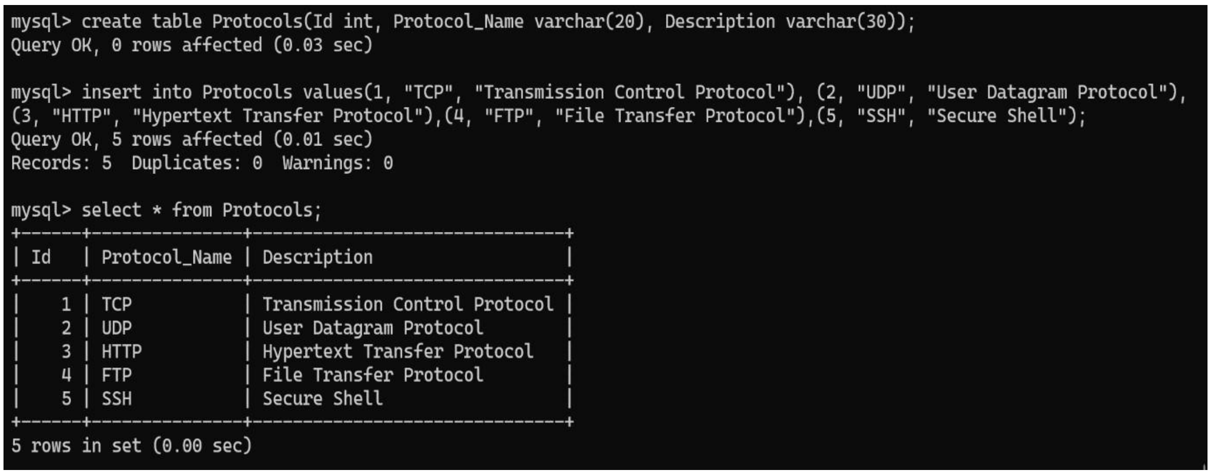
Using **second database**, named as **‘Network_Traffic’**:

- Creating first table, named as ‘Traffic’ and displaying it:



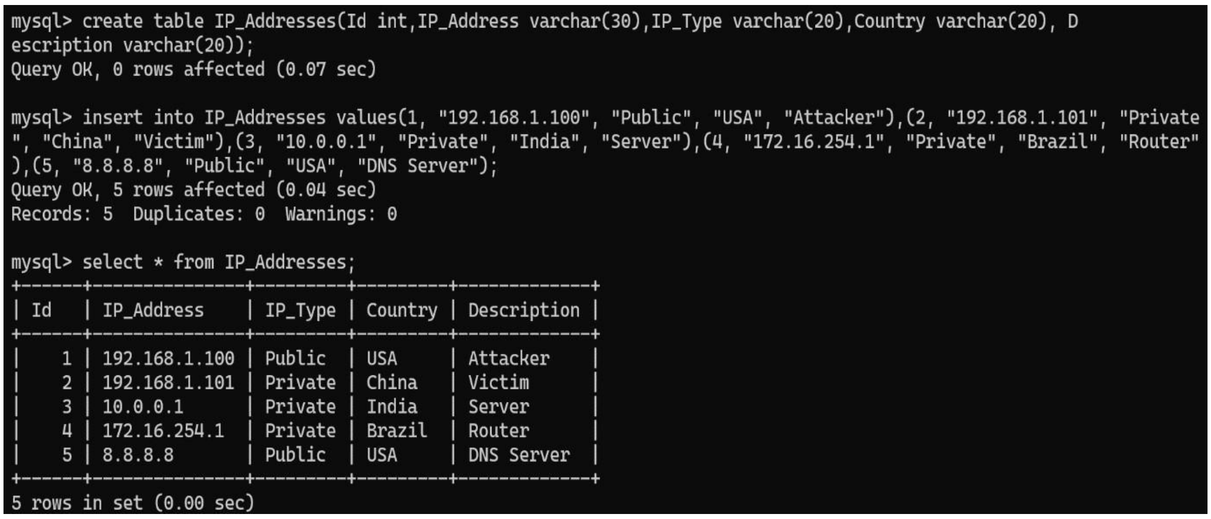
```
mysql> use Network_Traffic;
Database changed
mysql> create table Traffic(Id int, Timestamp datetime, Source_IP varchar(30), Destination_IP varchar(30), Protocol v
archar(20));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Traffic values(1, "2022-01-01 12:00:00", "192.168.1.100", "192.168.1.1", "TCP"), (2, "2022-01-02 1
3:00:00", "192.168.1.101", "192.168.1.2", "UDP"), (3, "2022-01-03 14:00:00", "192.168.1.102", "192.168.1.3", "HTTP"),
 (4, "2022-01-04 15:00:00", "192.168.1.103", "192.168.1.4", "FTP"), (5, "2022-01-05 16:00:00", "192.168.1.104", "192.
168.1.5", "SSH");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Traffic;
+------+---------------------+---------------+----------------+----------+
| Id   | Timestamp           | Source_IP     | Destination_IP | Protocol |
+------+---------------------+---------------+----------------+----------+
|    1 | 2022-01-01 12:00:00 | 192.168.1.100 | 192.168.1.1    | TCP      |
|    2 | 2022-01-02 13:00:00 | 192.168.1.101 | 192.168.1.2    | UDP      |
|    3 | 2022-01-03 14:00:00 | 192.168.1.102 | 192.168.1.3    | HTTP     |
|    4 | 2022-01-04 15:00:00 | 192.168.1.103 | 192.168.1.4    | FTP      |
|    5 | 2022-01-05 16:00:00 | 192.168.1.104 | 192.168.1.5    | SSH      |
+------+---------------------+---------------+----------------+----------+
5 rows in set (0.00 sec)
```

- Creating second table, named as ‘Protocols’ and displaying it:



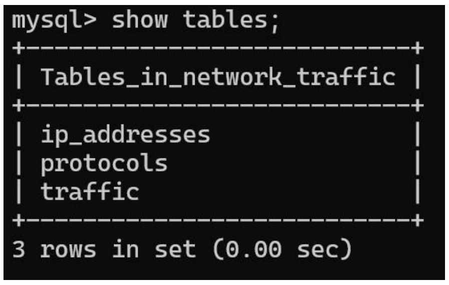
```
mysql> create table Protocols(Id int, Protocol_Name varchar(20), Description varchar(30));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into Protocols values(1, "TCP", "Transmission Control Protocol"), (2, "UDP", "User Datagram Protocol"),
(3, "HTTP", "Hypertext Transfer Protocol"),(4, "FTP", "File Transfer Protocol"),(5, "SSH", "Secure Shell");
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Protocols;
+------+---------------+-------------------------------+
| Id   | Protocol_Name | Description                   |
+------+---------------+-------------------------------+
|    1 | TCP           | Transmission Control Protocol |
|    2 | UDP           | User Datagram Protocol        |
|    3 | HTTP          | Hypertext Transfer Protocol   |
|    4 | FTP           | File Transfer Protocol        |
|    5 | SSH           | Secure Shell                  |
+------+---------------+-------------------------------+
5 rows in set (0.00 sec)
```

- Creating third table, named as ‘IP_Addresses’ and displaying it:



```
mysql> create table IP_Addresses(Id int,IP_Address varchar(30),IP_Type varchar(20),Country varchar(20), D
escription varchar(20));
Query OK, 0 rows affected (0.07 sec)

mysql> insert into IP_Addresses values(1, "192.168.1.100", "Public", "USA", "Attacker"),(2, "192.168.1.101", "Private
", "China", "Victim"),(3, "10.0.0.1", "Private", "India", "Server"),(4, "172.16.254.1", "Private", "Brazil", "Router"
),(5, "8.8.8.8", "Public", "USA", "DNS Server");
Query OK, 5 rows affected (0.04 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from IP_Addresses;
+------+---------------+---------+---------+-------------+
| Id   | IP_Address    | IP_Type | Country | Description |
+------+---------------+---------+---------+-------------+
|    1 | 192.168.1.100 | Public  | USA     | Attacker    |
|    2 | 192.168.1.101 | Private | China   | Victim      |
|    3 | 10.0.0.1      | Private | India   | Server      |
|    4 | 172.16.254.1  | Private | Brazil  | Router      |
|    5 | 8.8.8.8       | Public  | USA     | DNS Server  |
+------+---------------+---------+---------+-------------+
5 rows in set (0.00 sec)
```

To list the tables available in database ‘Network_Traffic’:



```
mysql> show tables;
+---------------------------+
| Tables_in_network_traffic |
+---------------------------+
| ip_addresses              |
| protocols                 |
| traffic                   |
+---------------------------+
3 rows in set (0.00 sec)
```

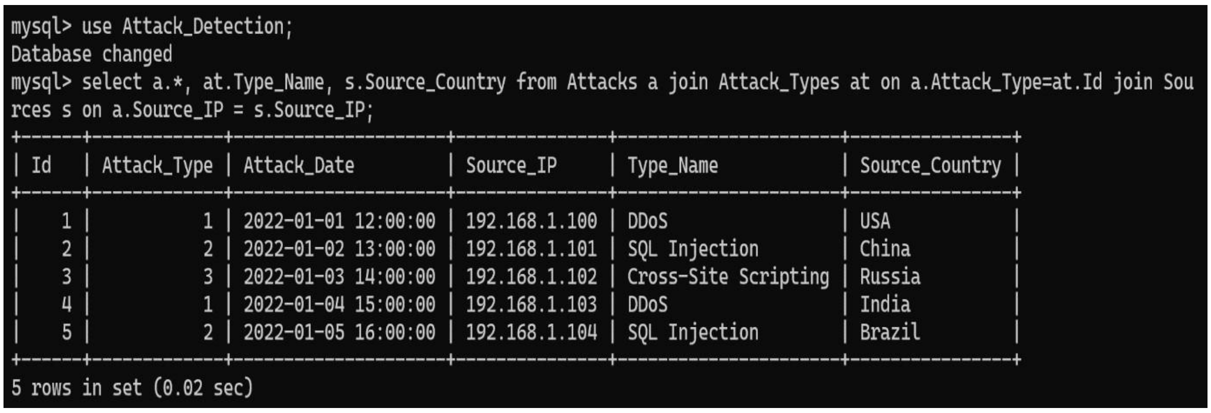
4) Database used:

MySQL is an open-source relational database management system (RDBMS) that allows you to store, manage, and retrieve data efficiently using Structured Query Language (SQL). It’s widely used in web applications due to its speed, reliability, and flexibility. MySQL is known for its scalability, handling everything from small to large databases. It supports ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring reliable transactions, and works across multiple platforms, including Linux, Windows, and macOS.

Version used is 8.4.1

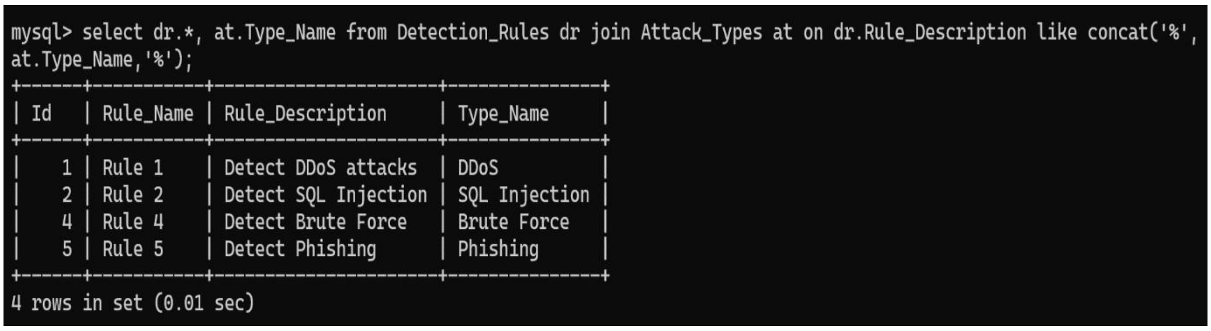
4) Queries identified by the Network Infra security team:

Retrieve all attacks with corresponding attack type and source



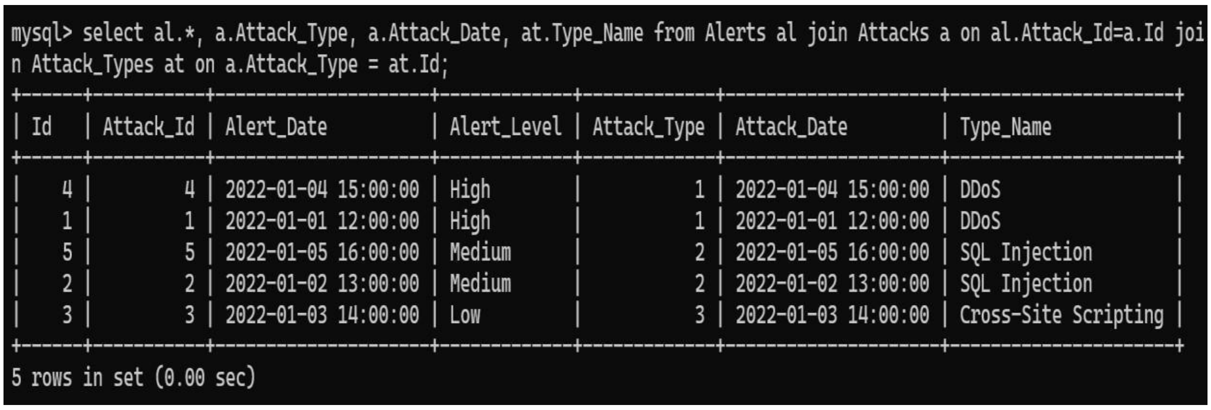
```
mysql> use Attack_Detection;
Database changed
mysql> select a.*, at.Type_Name, s.Source_Country from Attacks a join Attack_Types at on a.Attack_Type=at.Id join Sou
rces s on a.Source_IP = s.Source_IP;
+------+-------------+---------------------+---------------+----------------------+----------------+
| Id   | Attack_Type | Attack_Date         | Source_IP     | Type_Name            | Source_Country |
+------+-------------+---------------------+---------------+----------------------+----------------+
|    1 |           1 | 2022-01-01 12:00:00 | 192.168.1.100 | DDoS                 | USA            |
|    2 |           2 | 2022-01-02 13:00:00 | 192.168.1.101 | SQL Injection        | China          |
|    3 |           3 | 2022-01-03 14:00:00 | 192.168.1.102 | Cross-Site Scripting | Russia         |
|    4 |           1 | 2022-01-04 15:00:00 | 192.168.1.103 | DDoS                 | India          |
|    5 |           2 | 2022-01-05 16:00:00 | 192.168.1.104 | SQL Injection        | Brazil         |
+------+-------------+---------------------+---------------+----------------------+----------------+
5 rows in set (0.02 sec)
```

Retrieve all detection rules with corresponding attack type:



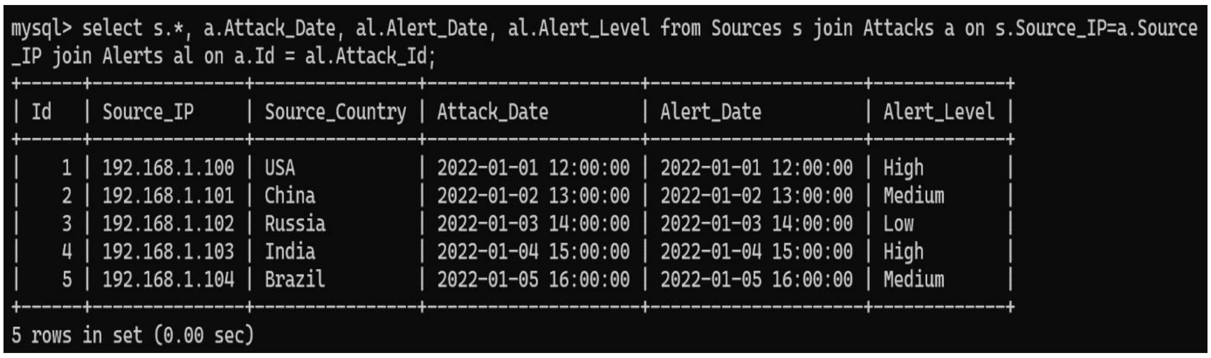
```
mysql> select dr.*, at.Type_Name from Detection_Rules dr join Attack_Types at on dr.Rule_Description like concat('%',
at.Type_Name,'%');
+------+-----------+----------------------+---------------+
| Id   | Rule_Name | Rule_Description     | Type_Name     |
+------+-----------+----------------------+---------------+
|    1 | Rule 1    | Detect DDoS attacks  | DDoS          |
|    2 | Rule 2    | Detect SQL Injection | SQL Injection |
|    4 | Rule 4    | Detect Brute Force   | Brute Force   |
|    5 | Rule 5    | Detect Phishing      | Phishing      |
+------+-----------+----------------------+---------------+
4 rows in set (0.01 sec)
```

Retrieve all alerts with corresponding attack information and alert level:



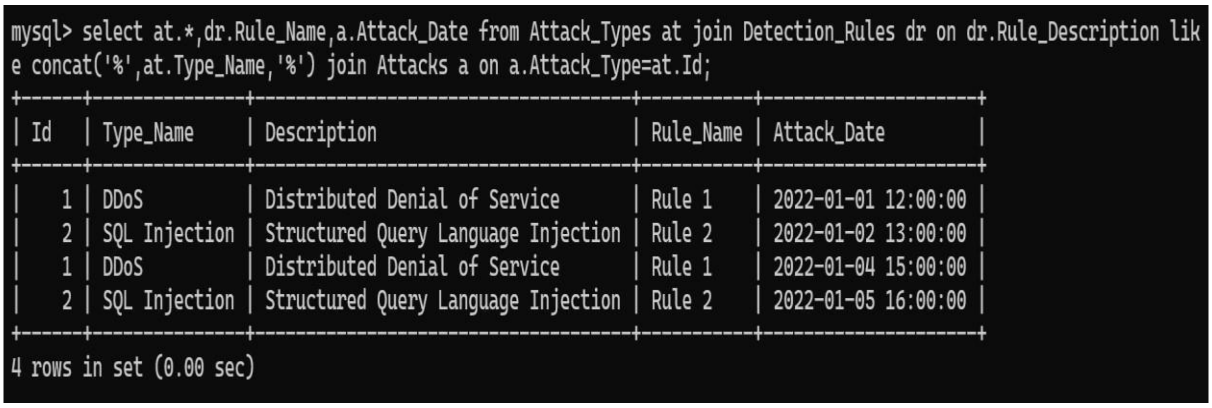
```
mysql> select al.*, a.Attack_Type, a.Attack_Date, at.Type_Name from Alerts al join Attacks a on al.Attack_Id=a.Id joi
n Attack_Types at on a.Attack_Type = at.Id;
+------+-----------+---------------------+-------------+-------------+---------------------+----------------------+
| Id   | Attack_Id | Alert_Date          | Alert_Level | Attack_Type | Attack_Date         | Type_Name            |
+------+-----------+---------------------+-------------+-------------+---------------------+----------------------+
|    4 |         4 | 2022-01-04 15:00:00 | High        |           1 | 2022-01-04 15:00:00 | DDoS                 |
|    1 |         1 | 2022-01-01 12:00:00 | High        |           1 | 2022-01-01 12:00:00 | DDoS                 |
|    5 |         5 | 2022-01-05 16:00:00 | Medium      |           2 | 2022-01-05 16:00:00 | SQL Injection        |
|    2 |         2 | 2022-01-02 13:00:00 | Medium      |           2 | 2022-01-02 13:00:00 | SQL Injection        |
|    3 |         3 | 2022-01-03 14:00:00 | Low         |           3 | 2022-01-03 14:00:00 | Cross-Site Scripting |
+------+-----------+---------------------+-------------+-------------+---------------------+----------------------+
5 rows in set (0.00 sec)
```

Retrieve all sources with corresponding attack and alert information:



```
mysql> select s.*, a.Attack_Date, al.Alert_Date, al.Alert_Level from Sources s join Attacks a on s.Source_IP=a.Source
_IP join Alerts al on a.Id = al.Attack_Id;
+------+---------------+----------------+---------------------+---------------------+-------------+
| Id   | Source_IP     | Source_Country | Attack_Date         | Alert_Date          | Alert_Level |
+------+---------------+----------------+---------------------+---------------------+-------------+
|    1 | 192.168.1.100 | USA            | 2022-01-01 12:00:00 | 2022-01-01 12:00:00 | High        |
|    2 | 192.168.1.101 | China          | 2022-01-02 13:00:00 | 2022-01-02 13:00:00 | Medium      |
|    3 | 192.168.1.102 | Russia         | 2022-01-03 14:00:00 | 2022-01-03 14:00:00 | Low         |
|    4 | 192.168.1.103 | India          | 2022-01-04 15:00:00 | 2022-01-04 15:00:00 | High        |
|    5 | 192.168.1.104 | Brazil         | 2022-01-05 16:00:00 | 2022-01-05 16:00:00 | Medium      |
+------+---------------+----------------+---------------------+---------------------+-------------+
5 rows in set (0.00 sec)
```

Retrieve all attack types with corresponding detection rules and attacks:



```
mysql> select at.*,dr.Rule_Name,a.Attack_Date from Attack_Types at join Detection_Rules dr on dr.Rule_Description lik
e concat('%',at.Type_Name,'%') join Attacks a on a.Attack_Type=at.Id;
+------+---------------+--------------------------------------+-----------+---------------------+
| Id   | Type_Name     | Description                          | Rule_Name | Attack_Date         |
+------+---------------+--------------------------------------+-----------+---------------------+
|    1 | DDoS          | Distributed Denial of Service        | Rule 1    | 2022-01-01 12:00:00 |
|    2 | SQL Injection | Structured Query Language Injection  | Rule 2    | 2022-01-02 13:00:00 |
|    1 | DDoS          | Distributed Denial of Service        | Rule 1    | 2022-01-04 15:00:00 |
|    2 | SQL Injection | Structured Query Language Injection  | Rule 2    | 2022-01-05 16:00:00 |
+------+---------------+--------------------------------------+-----------+---------------------+
4 rows in set (0.00 sec)
```

5) Final Goal of the Project:

The final goal of the project is to develop a robust, scalable, and secure system that detects, monitors, and mitigates various types of cyberattacks, like DDoS. By integrating real-time alerts, detection rules, and advanced analytics, the system aims to enhance overall cybersecurity and protect critical assets from threats.