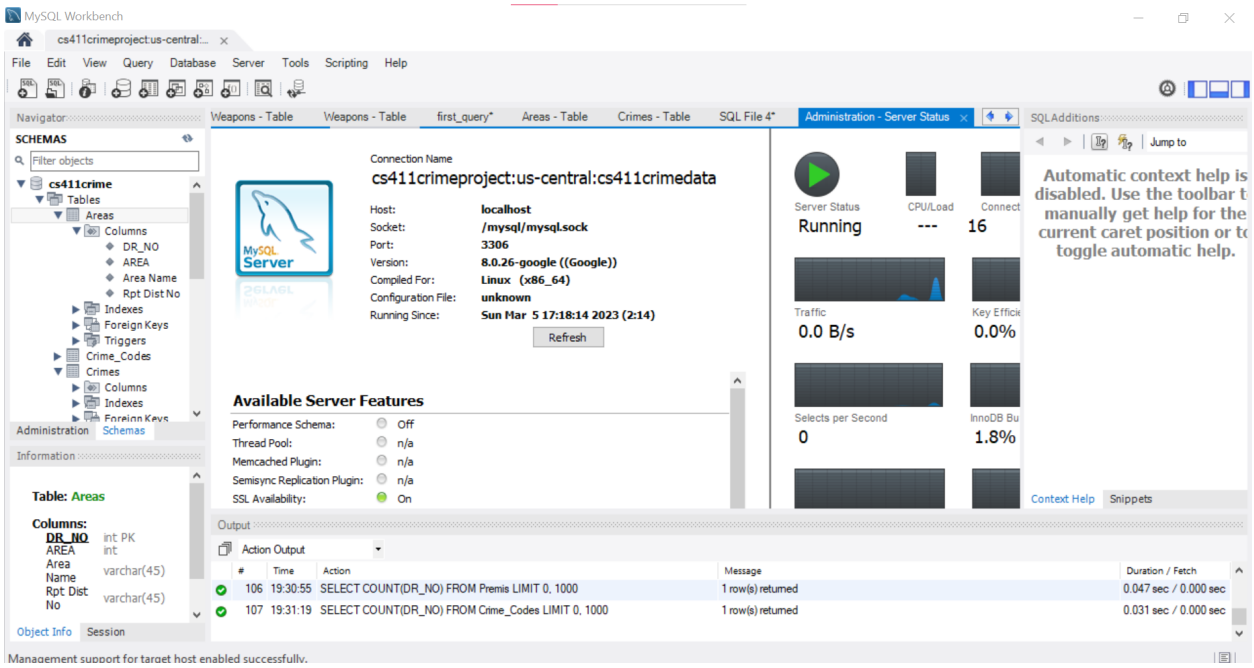
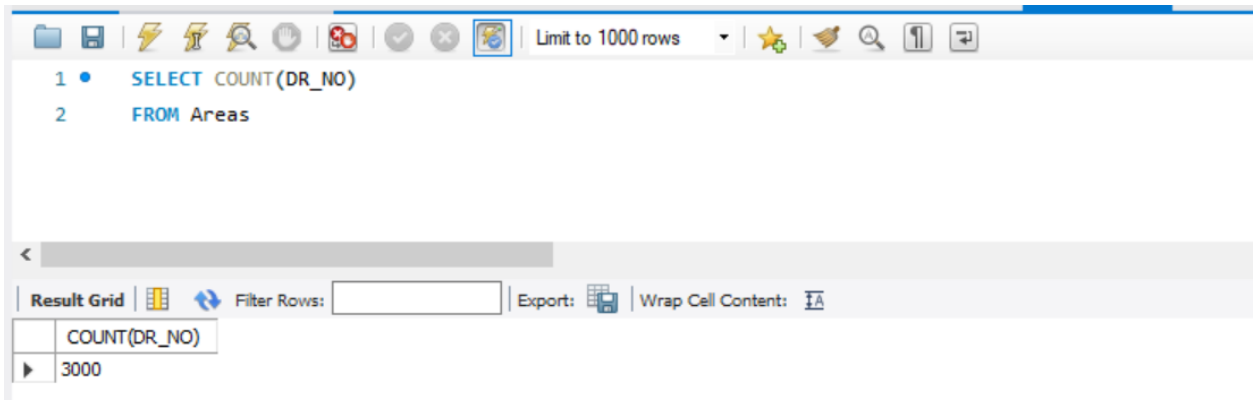
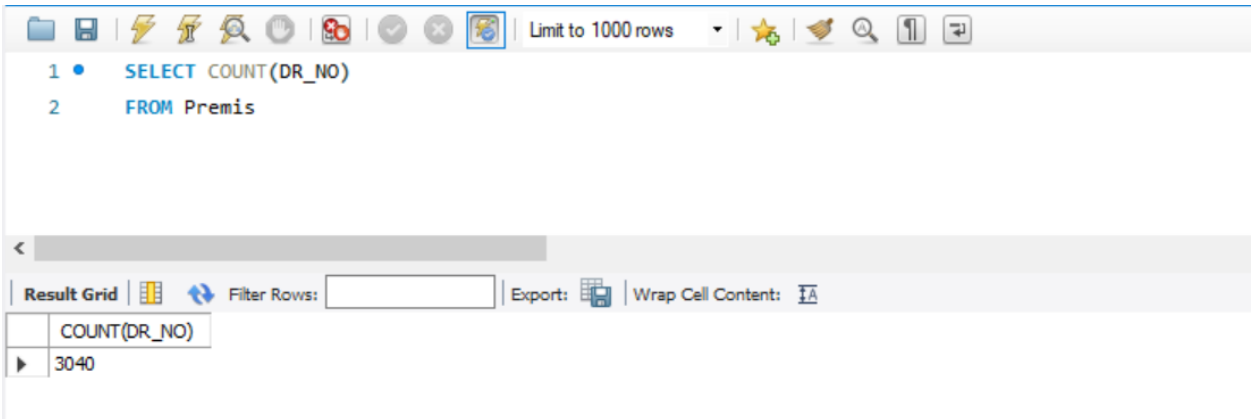
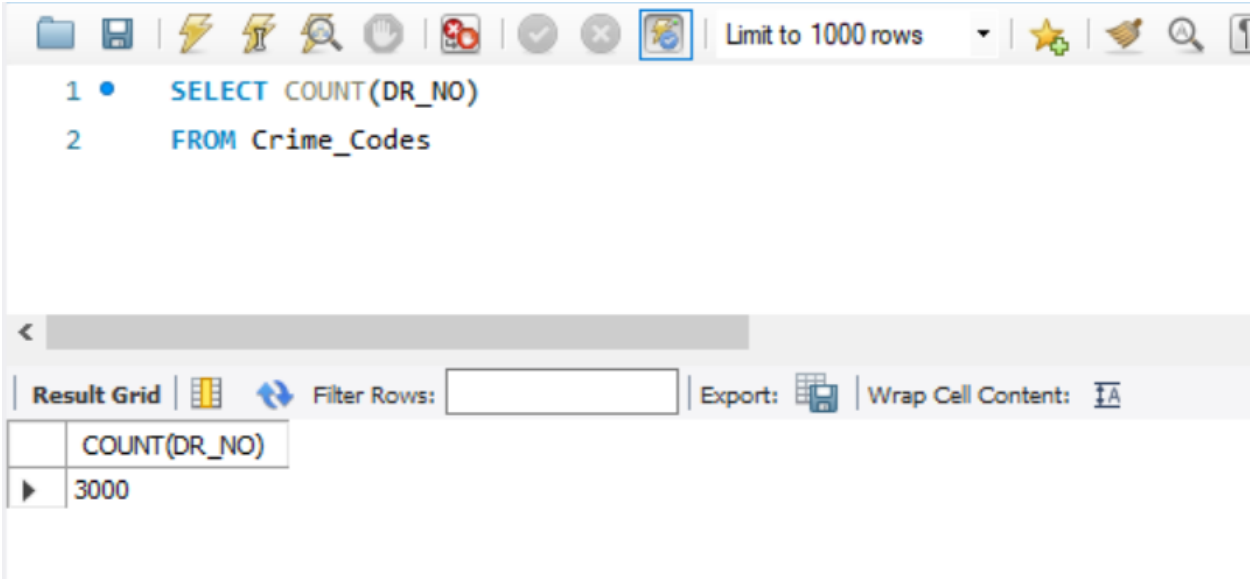
1. **Local Mysql server connection**



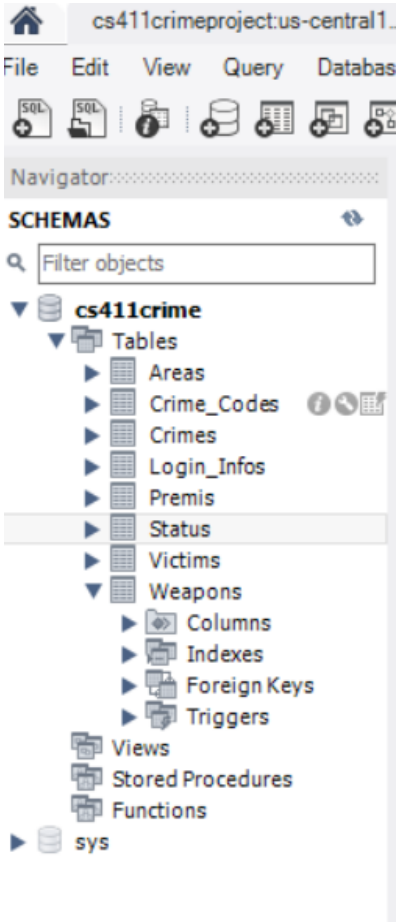
2. **Number of Rows in Tables**







3. **Implement Tables**



4. **DDL Commands**

Areas:
```
CREATE TABLE `Areas` (
  `DR_NO` int NOT NULL,
  `AREA` int DEFAULT NULL,
  `Area Name` varchar(45) DEFAULT NULL,
  `Rpt Dist No` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Crime Codes:
```
CREATE TABLE `Crime_Codes` (
  `DR_NO` int NOT NULL,
  `Crm Cd 1` int DEFAULT NULL,
  `Crm Cd Desc` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Login Infos:
```
CREATE TABLE `Login_Infos` (
  `User_Name` varchar(50) NOT NULL,
  `Hased_Password` varchar(50) NOT NULL,
  PRIMARY KEY (`User_Name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Premis:
```
CREATE TABLE `Premis` (
  `DR_NO` int NOT NULL,
  `Premis Cd` int DEFAULT NULL,
  `Premis Desc` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Status:
```
CREATE TABLE `Status` (
  `DR_NO` int NOT NULL,
  `Status` varchar(2) DEFAULT NULL,
  `Status Desc` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Victims:

```
CREATE TABLE `Victims` (
  `DR_NO` int NOT NULL,
  `Vict Age` int DEFAULT NULL,
  `Vict Sex` varchar(45) DEFAULT NULL,
  `Vict Descent` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

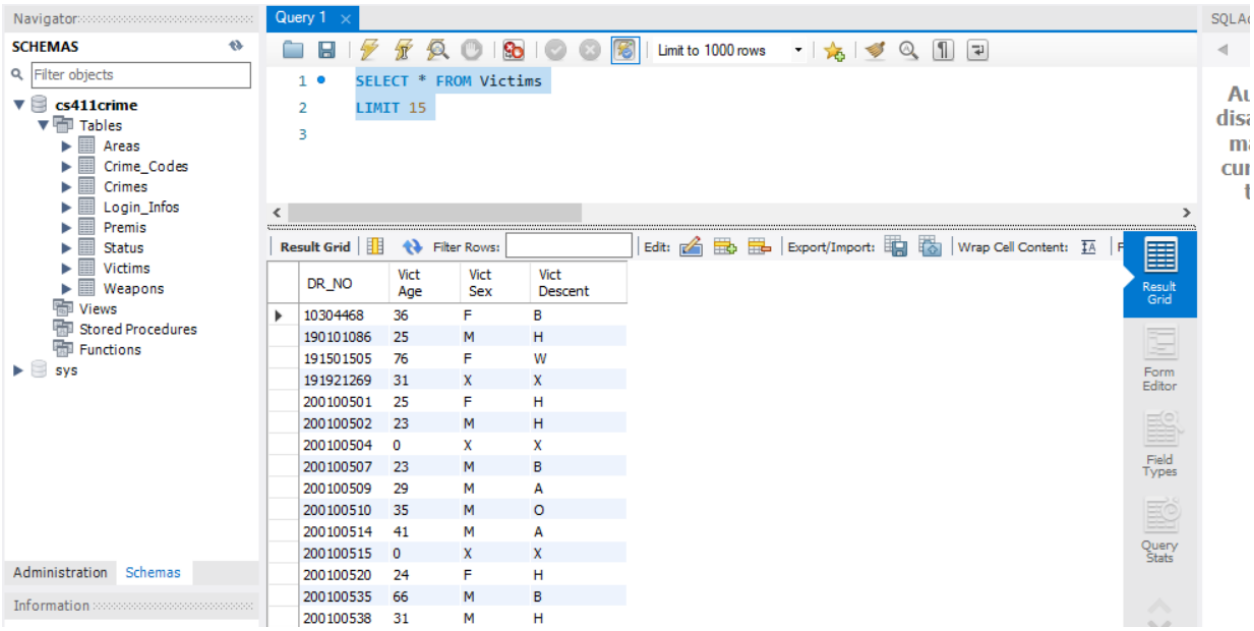
Weapons:

```
CREATE TABLE `Weapons` (
  `DR_NO` int NOT NULL,
  `Weapon Used Cd` int DEFAULT NULL,
  `Weapon Desc` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

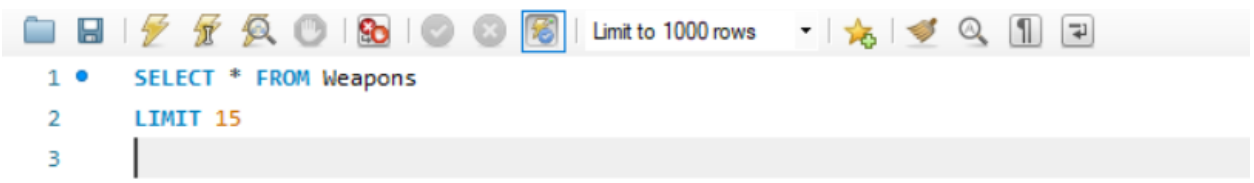
Crimes:

```
CREATE TABLE `Crimes` (
  `DR_NO` int NOT NULL,
  `Date Rptd` varchar(100) DEFAULT NULL,
  `DATE OCC` varchar(100) DEFAULT NULL,
  `TIME OCC` varchar(100) DEFAULT NULL,
  `Part 1-2` int DEFAULT NULL,
  `Crm Cd 2` varchar(100) DEFAULT NULL,
  `Crm Cd 3` varchar(100) DEFAULT NULL,
  `Crm Cd 4` varchar(100) DEFAULT NULL,
  `LOCATION` varchar(50) DEFAULT NULL,
  `Cross Street` text,
  `LAT` float DEFAULT NULL,
  `LON` float DEFAULT NULL,
  PRIMARY KEY (`DR_NO`),
  CONSTRAINT `DR NO` FOREIGN KEY (`DR_NO`) REFERENCES `Victims` (`DR_NO`),
  CONSTRAINT `DR_NO` FOREIGN KEY (`DR_NO`) REFERENCES `Areas` (`DR_NO`),
  CONSTRAINT `DR_NO1` FOREIGN KEY (`DR_NO`) REFERENCES `Crime_Codes` (`DR_NO`),
  CONSTRAINT `DR_NO2` FOREIGN KEY (`DR_NO`) REFERENCES `Premis` (`DR_NO`),
  CONSTRAINT `DR_NO3` FOREIGN KEY (`DR_NO`) REFERENCES `Status` (`DR_NO`),
  CONSTRAINT `DR_NO4` FOREIGN KEY (`DR_NO`) REFERENCES `Weapons` (`DR_NO`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

5. Insert data into tables For each table, at least 1000 rows of data was inserted



DR_NO	Vict Age	Vict Sex	Vict Descent
10304468	36	F	B
190101086	25	M	H
191501505	76	F	W
191921269	31	X	X
200100501	25	F	H
200100502	23	M	H
200100504	0	X	X
200100507	23	M	B
200100509	29	M	A
200100510	35	M	O
200100514	41	M	A
200100515	0	X	X
200100520	24	F	H
200100535	66	M	B
200100538	31	M	H



DR_NO	Weapon Used Cd	Weapon Desc
10304468	400	STRONG-ARM (HANDS, FIST, FEET OR BODILY ...
190101086	500	UNKNOWN WEAPON/OTHER WEAPON
200100501	500	UNKNOWN WEAPON/OTHER WEAPON
200100509	306	ROCK/THROWN OBJECT
200100510	511	VERBAL THREAT
200100515	500	UNKNOWN WEAPON/OTHER WEAPON
200100535	204	FOLDING KNIFE
200100546	500	UNKNOWN WEAPON/OTHER WEAPON
200100552	500	UNKNOWN WEAPON/OTHER WEAPON
200100556	400	STRONG-ARM (HANDS, FIST, FEET OR BODILY ...
200100568	500	UNKNOWN WEAPON/OTHER WEAPON
200100578	302	BLUNT INSTRUMENT
200100583	212	BOTTLE
200100708	400	STRONG-ARM (HANDS, FIST, FEET OR BODILY ...
200100712	500	UNKNOWN WEAPON/OTHER WEAPON

Query 1

Limit to 1000 rows

```
SELECT * FROM Crimes
LIMIT 15
```

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

DR_NO	Date Rptd	DATE OCC	TIME OCC	Part 1-2	Crm Cd 2	Crm Cd 3	Crm Cd 4
200100708	03/23/2020 12:00:00 AM	03/23/2020 12:00:00 AM	1530	2			
200100799	05/22/2020 12:00:00 AM	05/22/2020 12:00:00 AM	153	1	998.0		
200104946	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	1530	2			
200104953	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	1900	1			
200104954	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	2210	2			
200104955	01/12/2020 12:00:00 AM	01/12/2020 12:00:00 AM	230	2			
200104960	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	2330	2			
200104964	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	1830	1			
200104974	01/14/2020 12:00:00 AM	01/14/2020 12:00:00 AM	530	1			
200104977	01/14/2020 12:00:00 AM	01/14/2020 12:00:00 AM	820	2			
200104982	01/13/2020 12:00:00 AM	01/13/2020 12:00:00 AM	1400	2			
200104990	01/14/2020 12:00:00 AM	01/14/2020 12:00:00 AM	1630	2			
200104992	01/14/2020 12:00:00 AM	01/12/2020 12:00:00 AM	100	1			
200104993	01/14/2020 12:00:00 AM	01/14/2020 12:00:00 AM	1630	2			

Limit to 1000 rows

```
SELECT * FROM Status
LIMIT 15
```

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

DR_NO	Status	Status Desc
10304468	AO	Adult Other
190101086	IC	Invest Cont
191501505	IC	Invest Cont
191921269	IC	Invest Cont
200100501	IC	Invest Cont
200100502	IC	Invest Cont
200100504	IC	Invest Cont
200100507	IC	Invest Cont
200100509	IC	Invest Cont
200100510	IC	Invest Cont
200100514	AA	Adult Arrest
200100515	IC	Invest Cont
200100520	IC	Invest Cont
200100535	IC	Invest Cont
200100538	IC	Invest Cont

6. Advanced SQL queries

The first sql query found the number of victims attacked by the same type of weapon. For example, 8 victims had a rock/object thrown at them

```sql
SELECT `Weapon Desc`, COUNT(w.DR_NO) AS number_victims
FROM Weapons w JOIN Victims v ON (w.DR_NO = v.DR_NO)
GROUP BY `Weapon Desc`
LIMIT 15
```

Weapon Desc	number_victims
STRONG-ARM (HANDS, FIST, FEET OR BODILY ...	632
UNKNOWN WEAPON/OTHER WEAPON	139
ROCK/THROWN OBJECT	8
VERBAL THREAT	69
FOLDING KNIFE	5
BLUNT INSTRUMENT	7
BOTTLE	11
OTHER CUTTING INSTRUMENT	4
HAND GUN	48
PHYSICAL PRESENCE	4
VEHICLE	3
SCISSORS	4
STICK	12
MACHETE	7
OTHER KNIFE	20

Query below finds the number of victims in a certain area

```sql
SELECT `Area Name`, COUNT(v.DR_NO) as number_victims
FROM Areas a JOIN Victims v ON (a.DR_NO = v.DR_NO)
GROUP BY `Area Name`
LIMIT 15
```

Area Name	number_victims
Southwest	615
Central	1119
N Hollywood	25
Mission	22
Rampart	783
77th Street	32
Wilshire	27
Hollenbeck	29
Harbor	31
Hollywood	27
West LA	27
Van Nuys	24
West Valley	21
Northeast	24
Newton	27

7. Indexing

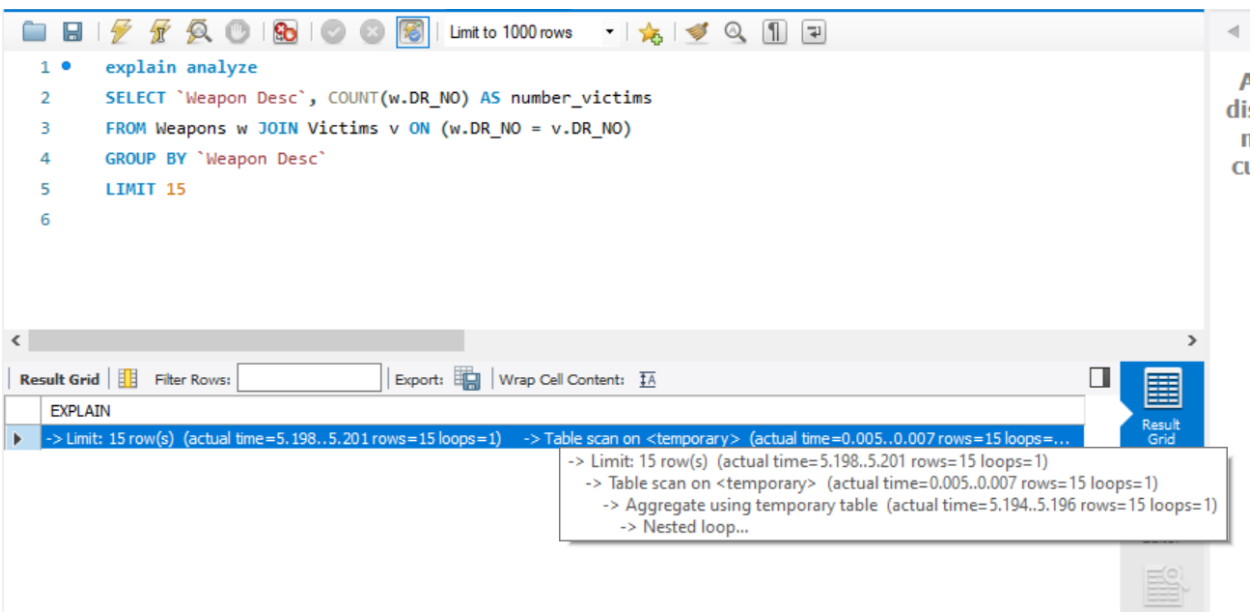
The indexing design allows faster lookup of records using a specific column. Clustered indexing can speed up performance of queries as it traverses a table based on key values.

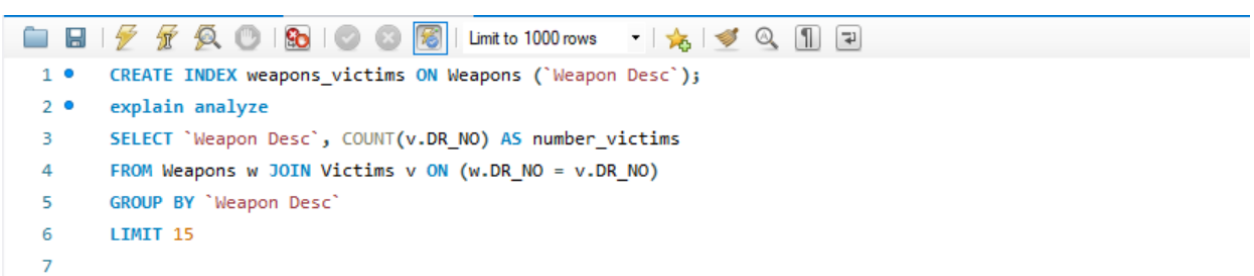
can help improve the performance of queries that traverse the table in key order

Query 1:

We indexed on each column for the table Weapons to understand which attribute, when indexed, provides us with the best time. After analyzing our results (displayed below), we found that indexing on the Weapon Desc column outputted the fastest time. Since indexing tends to speed up searching records, and the Weapons Desc column has the longest value, it makes sense as to why indexing on that speeds up the process the most.

Indexing Column	None	Weapons(‘Weapon Desc’)	Weapons('Weapons Used Cd')	Weapons('Weapons Desc', 'Weapons Used Cd)
Time (s)	5.198..5.201	5.087..5.091	5.139…5.143	5.200…5.203





```
# CREATE INDEX weapons_victims ON Weapons (`Weapon Desc`);
explain analyze
SELECT `Weapon Desc`, COUNT(v.DR_NO) AS number_victims
FROM Weapons w JOIN Victims v ON (w.DR_NO = v.DR_NO)
GROUP BY `Weapon Desc`
LIMIT 15
```

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

EXPLAIN

-> Limit: 15 row(s) (actual time=5.087..5.091 rows=15 loops=1) -> Table scan on <temporary> (actual time=0.006..0.007 rows=15 loops=...

```
-> Limit: 15 row(s) (actual time=5.087..5.091 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.006..0.007 rows=15 loops=1)
    -> Aggregate using temporary table (actual time=5.086..5.089 rows=15 loops=1)
      -> Nested loop...
```

```
CREATE INDEX weapons_victims_2 ON Weapons (`Weapon Used Cd`);
DROP INDEX weapons_victims ON Weapons;
explain analyze
SELECT `Weapon Desc`, COUNT(v.DR_NO) AS number_victims
FROM Weapons w JOIN Victims v ON (w.DR_NO = v.DR_NO)
GROUP BY `Weapon Desc`
LIMIT 15
```

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

EXPLAIN

-> Limit: 15 row(s) (actual time=5.139..5.143 rows=15 loops=1) -> Table scan on <temporary> (actual time=0.005..0.007 rows=15 loops=...

```
-> Limit: 15 row(s) (actual time=5.139..5.143 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.005..0.007 rows=15 loops=1)
    -> Aggregate using temporary table (actual time=5.138..5.140 rows=15 loops=1)
      -> Nested loop...
```

```
#CREATE INDEX weapons_victims_3 ON Weapons (`Weapon Desc`(50), `Weapon Used Cd`);
#DROP INDEX weapons_victims_3 ON Weapons;
explain analyze
SELECT `Weapon Desc`, COUNT(v.DR_NO) AS number_victims
FROM Weapons w JOIN Victims v ON (w.DR_NO = v.DR_NO)
GROUP BY `Weapon Desc`
LIMIT 15
```

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

EXPLAIN

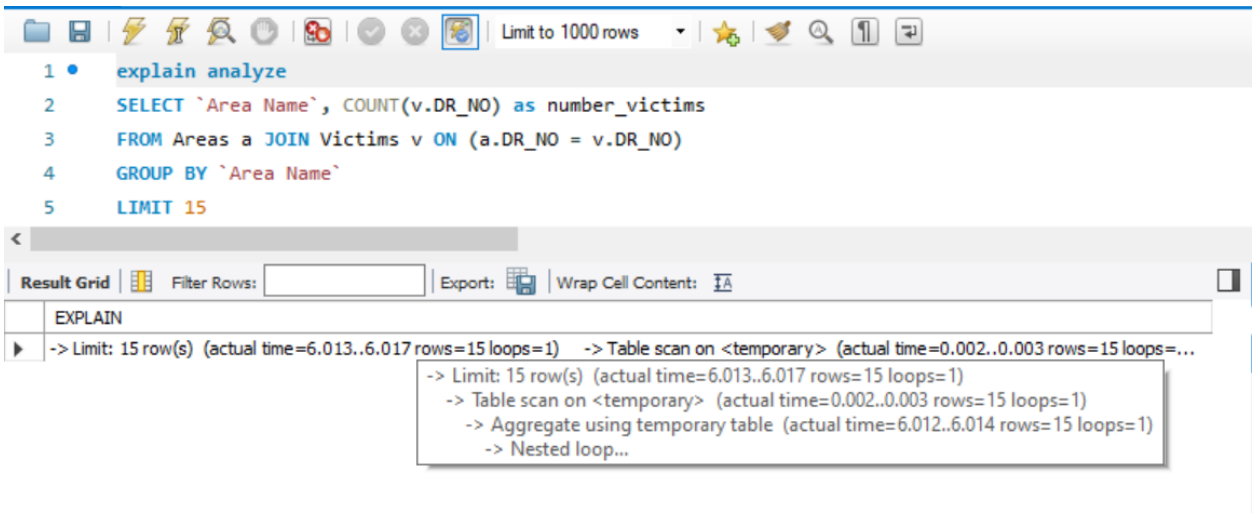
-> Limit: 15 row(s) (actual time=5.200..5.203 rows=15 loops=1) -> Table scan on <temporary> (actual time=0.006..0.008 rows=15 loops=...

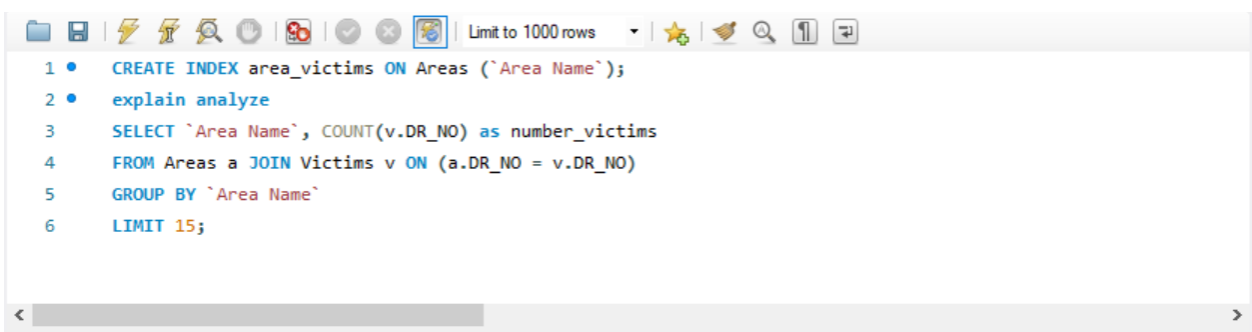
```
-> Limit: 15 row(s) (actual time=5.200..5.203 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.006..0.008 rows=15 loops=1)
    -> Aggregate using temporary table (actual time=5.199..5.201 rows=15 loops=1)
      -> Nested loop...
```

Query 2:

We indexed on each column for the table Areas to understand which attribute, when indexed, provides us with the best time. After analyzing our results (displayed below), we found that indexing on the Area Name column outputted the fastest time. We think that it is fastest because we grouped by Area Name.

Indexing Column	None	Areas('Area Name')	Areas('AREA')	Areas('Rpt Dist No')
Time (s)	6.013..6.017	0.123..3.356	7.645..7.648	7.262..7.265





```
# CREATE INDEX area_victims ON Areas (`Area Name`);
explain analyze
SELECT `Area Name`, COUNT(v.DR_NO) as number_victims
FROM Areas a JOIN Victims v ON (a.DR_NO = v.DR_NO)
GROUP BY `Area Name`
LIMIT 15;
```

Result Grid

EXPLAIN
-> Limit: 15 row(s) (cost=753.02 rows=15) (actual time=0.123..3.356 rows=15 loops=1) -> Group aggregate: count(v.DR_NO) (cost=753.0...

```
-> Limit: 15 row(s) (cost=753.02 rows=15) (actual time=0.123..3.356 rows=15 loops=1)
  -> Group aggregate: count(v.DR_NO) (cost=753.02 rows=15) (actual time=0.122..3.354 rows=15 loops=1)
    -> Nested loop inner join (cost=751.52 rows=15) (actua...
```

Result 17 — Read Only

```
CREATE INDEX area_victims_2 ON Areas (`AREA`);
DROP INDEX area_victims ON Areas;
explain analyze
SELECT `Area Name`, COUNT(v.DR_NO) as number_victims
FROM Areas a JOIN Victims v ON (a.DR_NO = v.DR_NO)
GROUP BY `Area Name`
LIMIT 15;
```

Result Grid

EXPLAIN
-> Limit: 15 row(s) (actual time=7.645..7.648 ...

```
-> Limit: 15 row(s) (actual time=7.645..7.648 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.001..0.002 rows=15 loops=1)
    -> Aggregate using temporary table (actual time=7.644..7.646 rows=15 loops=1)
      -> Nested loop...
```

```
#CREATE INDEX area_victims_3 ON Areas (`Rpt Dist No`);
#DROP INDEX area_victims ON Areas;
explain analyze
SELECT `Area Name`, COUNT(v.DR_NO) as number_victims
FROM Areas a JOIN Victims v ON (a.DR_NO = v.DR_NO)
GROUP BY `Area Name`
LIMIT 15;
```

Result Grid

EXPLAIN
-> Limit: 15 row(s) (actual time=7.262..7.265 rows=15 loops=1) -> Table scan on <temporary> (actual time=0.001..0.002 rows=15 loops=...

```
-> Limit: 15 row(s) (actual time=7.262..7.265 rows=15 loops=1)
  -> Table scan on <temporary> (actual time=0.001..0.002 rows=15 loops=1)
    -> Aggregate using temporary table (actual time=7.261..7.263 rows=15 loops=1)
      -> Nested loop...
```

Result 19 — Read Only