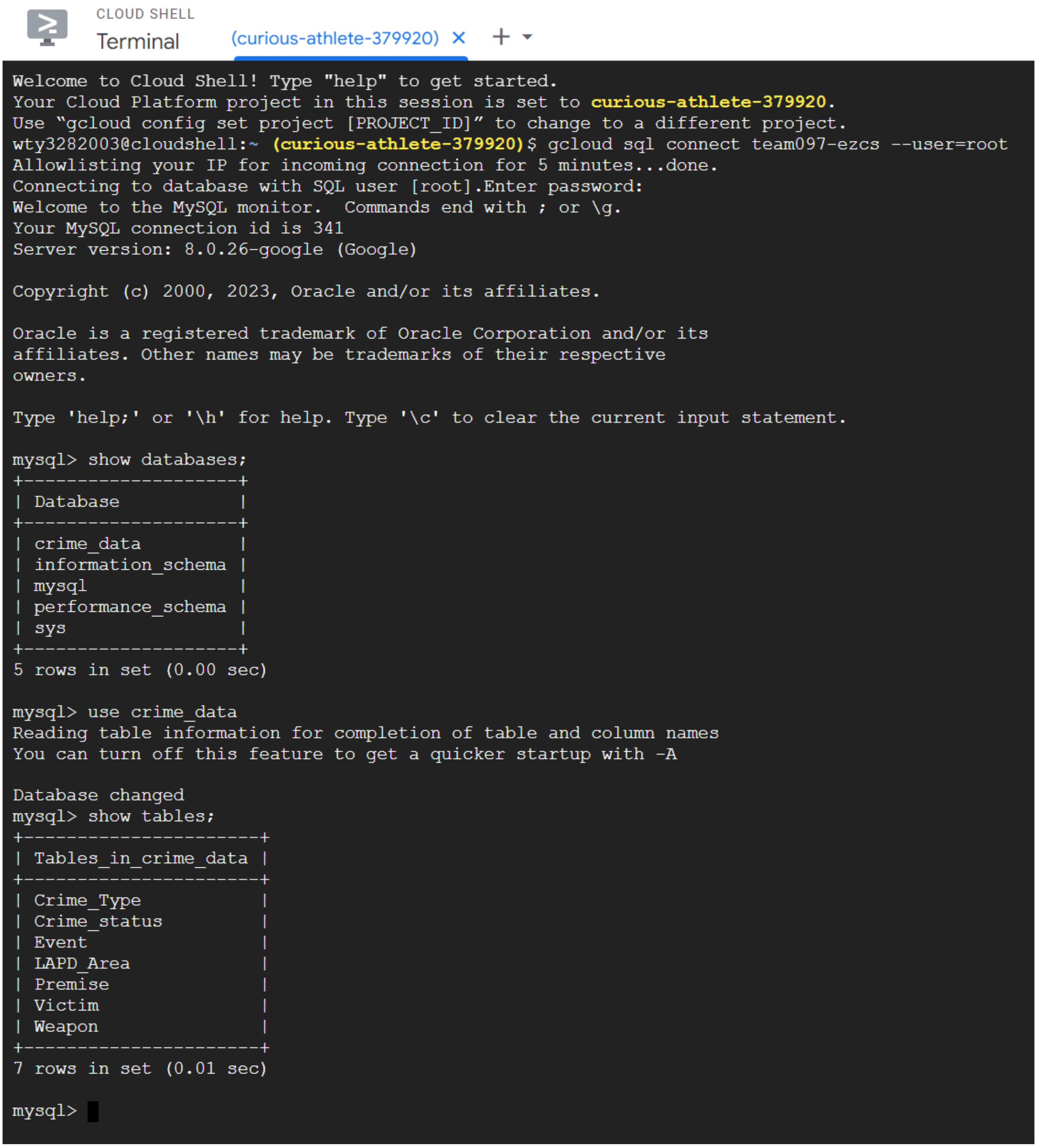
Database Information

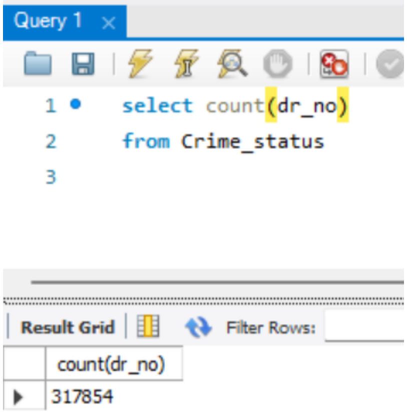


```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to curious-athlete-379920.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
wty3282003@cloudshell:~ (curious-athlete-379920)$ gcloud sql connect team097-ezcs --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 341
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| crime_data         |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use crime_data
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------------+
| Tables_in_crime_data |
+----------------------+
| Crime_Type           |
| Crime_status         |
| Event                |
| LAPD_Area            |
| Premise              |
| Victim               |
| Weapon               |
+----------------------+
7 rows in set (0.01 sec)

mysql> 
```

DDL implementation of tables

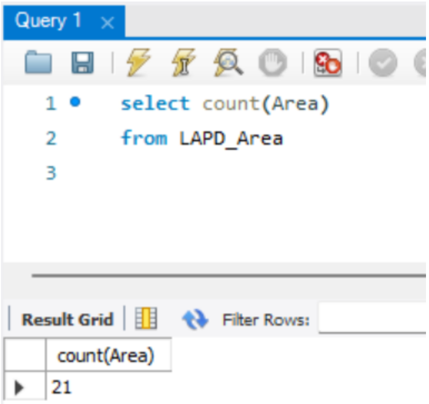
```
create table LAPD_Area  (Area INT primary key, Area_Name varchar(30));
create table Weapon     (Weapon_Type_Code INT primary key, Weapon_Description VARCHAR(30));
create table Crime_Type (Crime_Code INT primary key, Crime_Code_Description VARCHAR(30));
create table Premise    (Premise_Code INT primary key, Premise_Descent VARCHAR(30));
create table Event  (DR_NO INT primary key,
                Date_of_Report       VARCHAR(30),
                Date_of_Occurrence    VARCHAR(30),
                Time_of_Occurrence    INT,
                Location              VARCHAR(30),
                Cross_Street_Location VARCHAR(30),
                Latitude              REAL,
                Longitude             REAL,
                crime_code            INT not null,
                weapon_code           INT,
                area                  INT not null,
                premise_code          INT not null,
                foreign key (crime_code) references Crime_Type(Crime_Code)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE,
                foreign key (weapon_code) references Weapon(Weapon_Type_Code)
                  ON DELETE SET NULL
                  ON UPDATE CASCADE,
                foreign key (area) references LAPD_Area(Area)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE,
                foreign key (premise_code) references Premise(Premise_Code)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE);

create table Victim (dr_no INT primary key,
                Victim_Age INT,
                Victim_Sex VARCHAR(30),
                Victim_Descent VARCHAR(30),
                foreign key (dr_no) references Event(DR_NO)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
                );

create table Crime_status(crime_code INT not null,
                    dr_no INT not null,
                    Status VARCHAR(30),
                    Status_Desc VARCHAR(30),
                    foreign key(crime_code) references Crime_Type(Crime_Code)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE,
                    foreign key(dr_no) references Event(DR_NO)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE,
                    primary key(crime_code,dr_no)
                    );
```

Count on each table



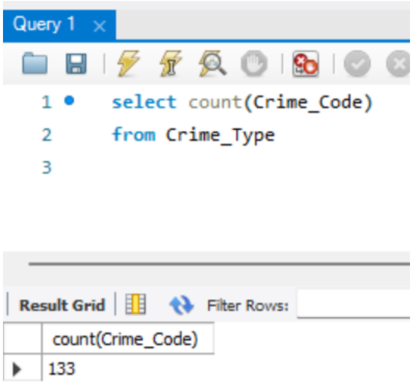
```
select count(dr_no)
from Crime_status
```

```
count(dr_no)
317854
```



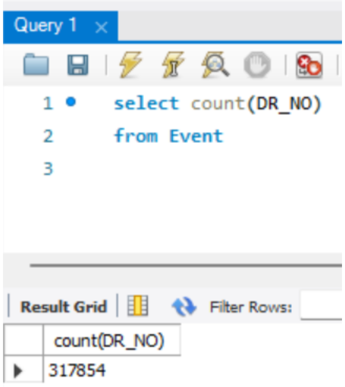
```
select count(Area)
from LAPD_Area
```

```
count(Area)
21
```



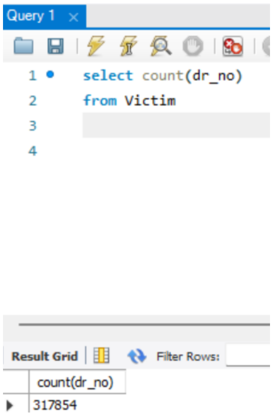
```
select count(Crime_Code)
from Crime_Type
```

```
count(Crime_Code)
133
```



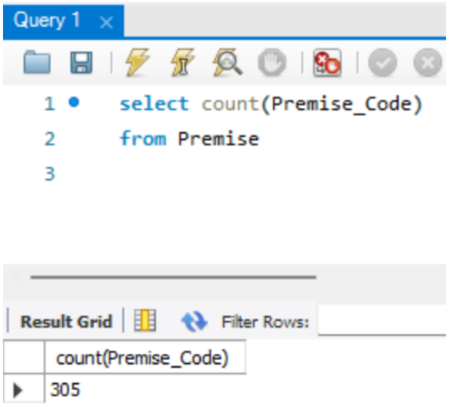
```
select count(DR_NO)
from Event
```

```
count(DR_NO)
317854
```



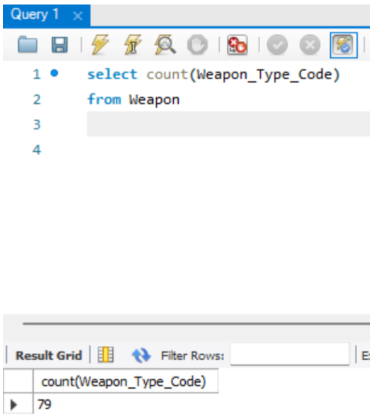
```
select count(dr_no)
from Victim
```

```
count(dr_no)
317854
```



```
select count(Premise_Code)
from Premise
```

```
count(Premise_Code)
305
```



```
select count(Weapon_Type_Code)
from Weapon
```

```
count(Weapon_Type_Code)
79
```

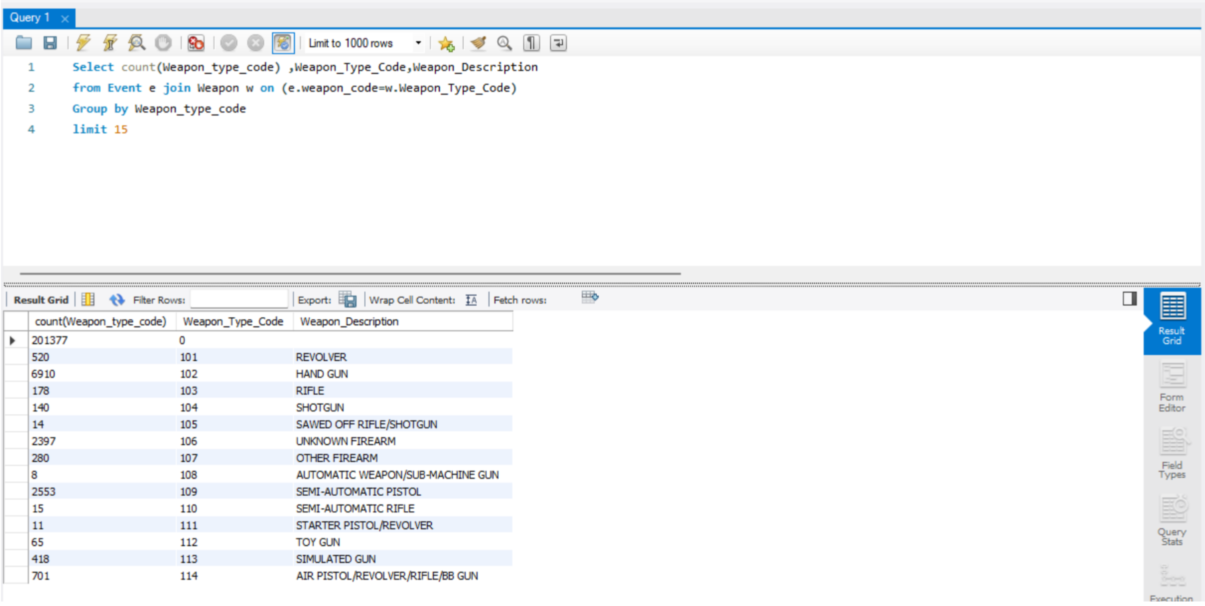
At least three of our tables: Crime_status, Event, and Victim have >= 1000 entries

Queries and indexing

Query 1

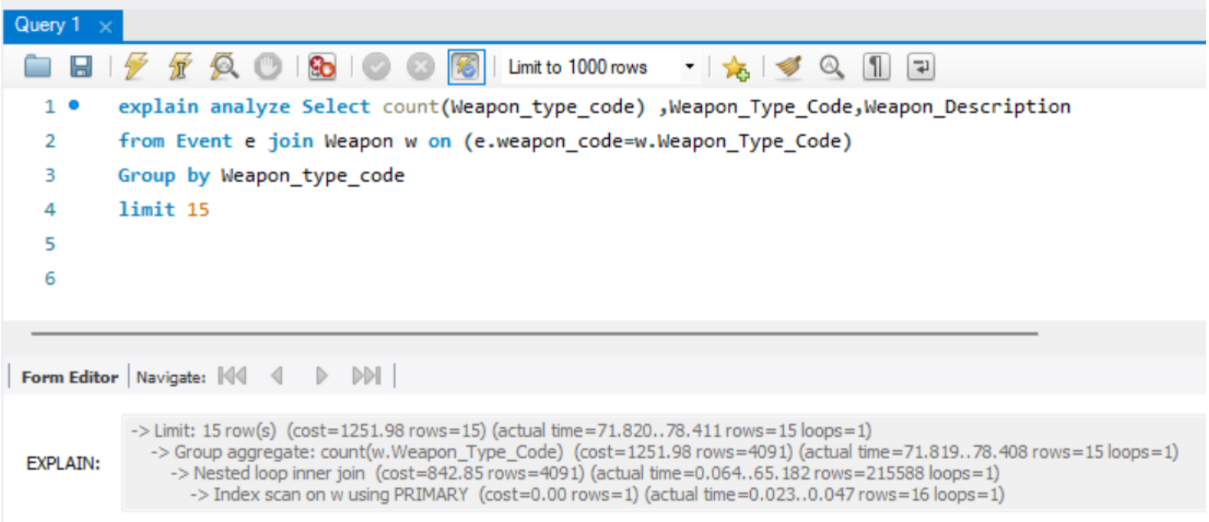
We want to know what is the count of events related to each type of weapons, the query returns count of event, weapon type code and description of weapon.

Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code) Group by Weapon_type_code
limit 15



```
Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description
from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code)
Group by Weapon_type_code
limit 15
```

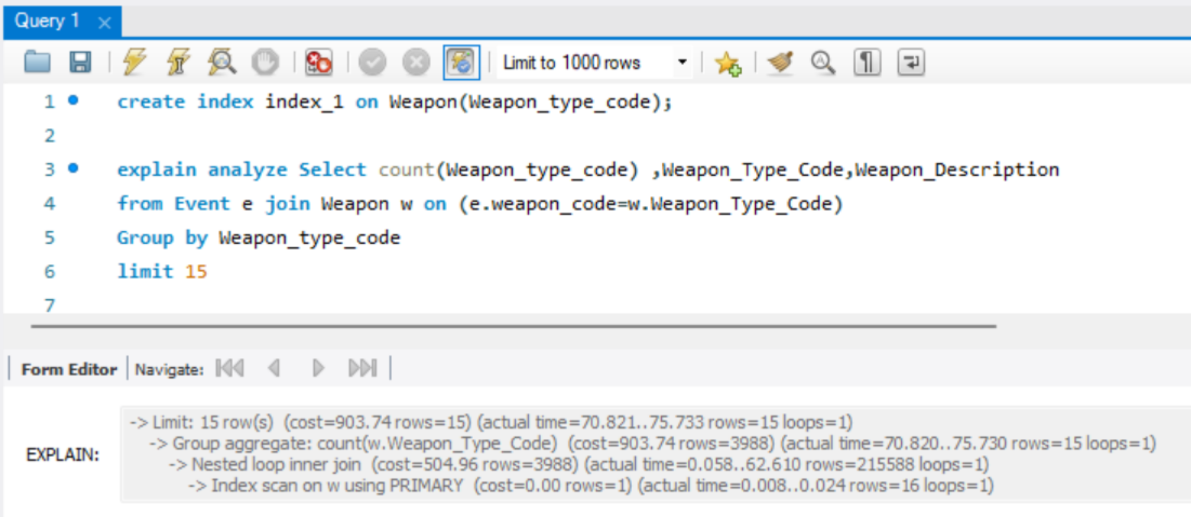
count(Weapon_type_code)	Weapon_Type_Code	Weapon_Description
201377	0	
520	101	REVOLVER
6910	102	HAND GUN
178	103	RIFLE
140	104	SHOTGUN
14	105	SAWED OFF RIFLE/SHOTGUN
2397	106	UNKNOWN FIREARM
280	107	OTHER FIREARM
8	108	AUTOMATIC WEAPON/SUB-MACHINE GUN
2553	109	SEMI-AUTOMATIC PISTOL
15	110	SEMI-AUTOMATIC RIFLE
11	111	STARTER PISTOL/REVOLVER
65	112	TOY GUN
418	113	SIMULATED GUN
701	114	AIR PISTOL/REVOLVER/RIFLE/BB GUN



```
explain analyze Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description
from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code)
Group by Weapon_type_code
limit 15
```

EXPLAIN:

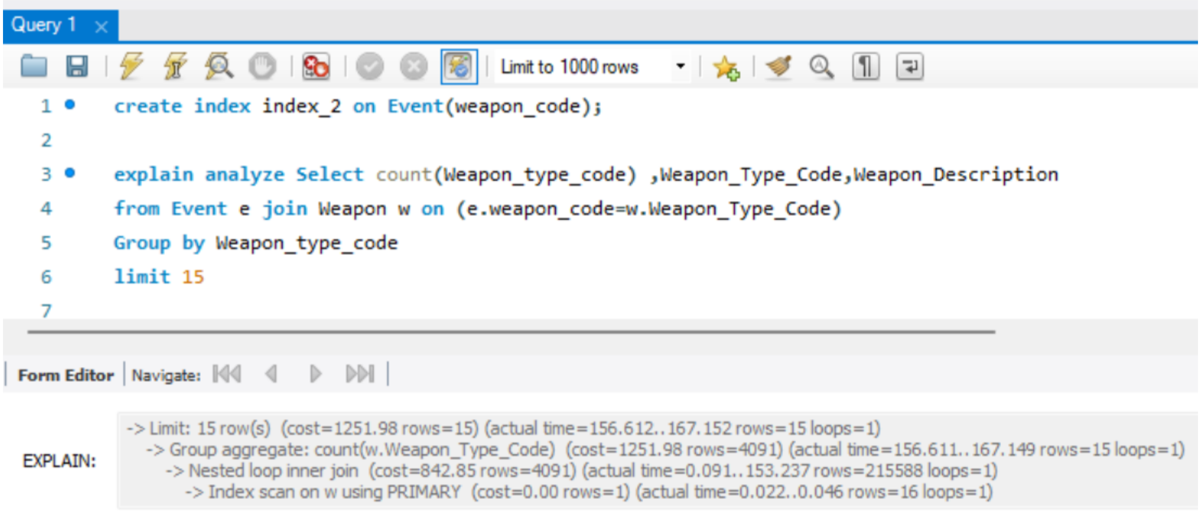
```
-> Limit: 15 row(s)  (cost=1251.98 rows=15) (actual time=71.820..78.411 rows=15 loops=1)
    -> Group aggregate: count(w.Weapon_Type_Code)  (cost=1251.98 rows=4091) (actual time=71.819..78.408 rows=15 loops=1)
        -> Nested loop inner join  (cost=842.85 rows=4091) (actual time=0.064..65.182 rows=215588 loops=1)
            -> Index scan on w using PRIMARY  (cost=0.00 rows=1) (actual time=0.023..0.047 rows=16 loops=1)
```



```sql
create index index_1 on Weapon(Weapon_type_code);

explain analyze Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description
from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code)
Group by Weapon_type_code
limit 15
```

EXPLAIN:

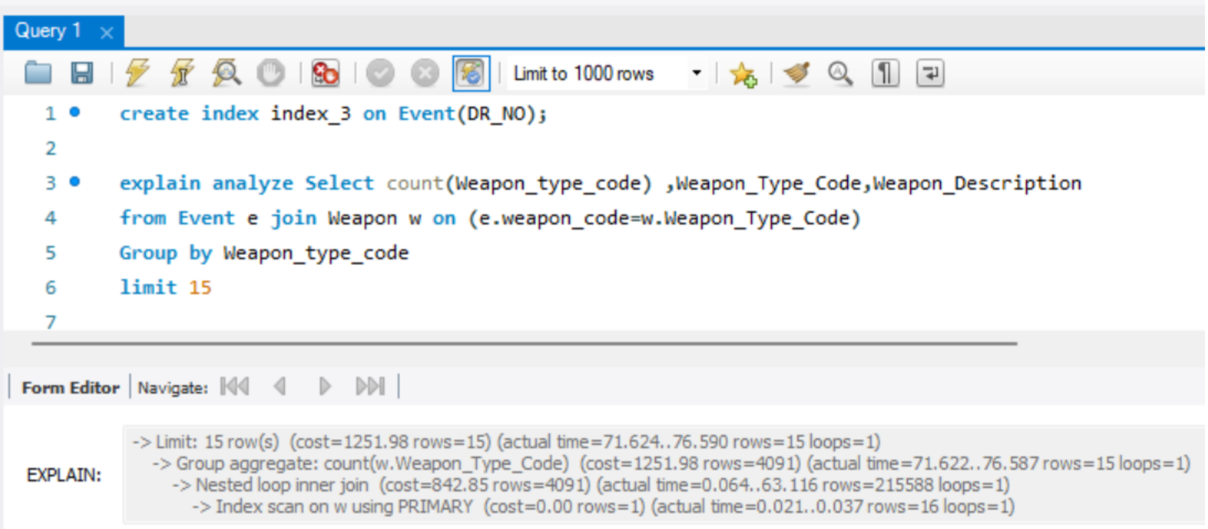
```
-> Limit: 15 row(s)  (cost=903.74 rows=15) (actual time=70.821..75.733 rows=15 loops=1)
    -> Group aggregate: count(w.Weapon_Type_Code)  (cost=903.74 rows=3988) (actual time=70.820..75.730 rows=15 loops=1)
        -> Nested loop inner join  (cost=504.96 rows=3988) (actual time=0.058..62.610 rows=215588 loops=1)
            -> Index scan on w using PRIMARY  (cost=0.00 rows=1) (actual time=0.008..0.024 rows=16 loops=1)
```



```sql
create index index_2 on Event(weapon_code);

explain analyze Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description
from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code)
Group by Weapon_type_code
limit 15
```

EXPLAIN:

```
-> Limit: 15 row(s)  (cost=1251.98 rows=15) (actual time=156.612..167.152 rows=15 loops=1)
    -> Group aggregate: count(w.Weapon_Type_Code)  (cost=1251.98 rows=4091) (actual time=156.611..167.149 rows=15 loops=1)
        -> Nested loop inner join  (cost=842.85 rows=4091) (actual time=0.091..153.237 rows=215588 loops=1)
            -> Index scan on w using PRIMARY  (cost=0.00 rows=1) (actual time=0.022..0.046 rows=16 loops=1)
```



```sql
create index index_3 on Event(DR_NO);

explain analyze Select count(Weapon_type_code) ,Weapon_Type_Code,Weapon_Description
from Event e join Weapon w on (e.weapon_code=w.Weapon_Type_Code)
Group by Weapon_type_code
limit 15
```

EXPLAIN:

```
-> Limit: 15 row(s)  (cost=1251.98 rows=15) (actual time=71.624..76.590 rows=15 loops=1)
    -> Group aggregate: count(w.Weapon_Type_Code)  (cost=1251.98 rows=4091) (actual time=71.622..76.587 rows=15 loops=1)
        -> Nested loop inner join  (cost=842.85 rows=4091) (actual time=0.064..63.116 rows=215588 loops=1)
            -> Index scan on w using PRIMARY  (cost=0.00 rows=1) (actual time=0.021..0.037 rows=16 loops=1)
```

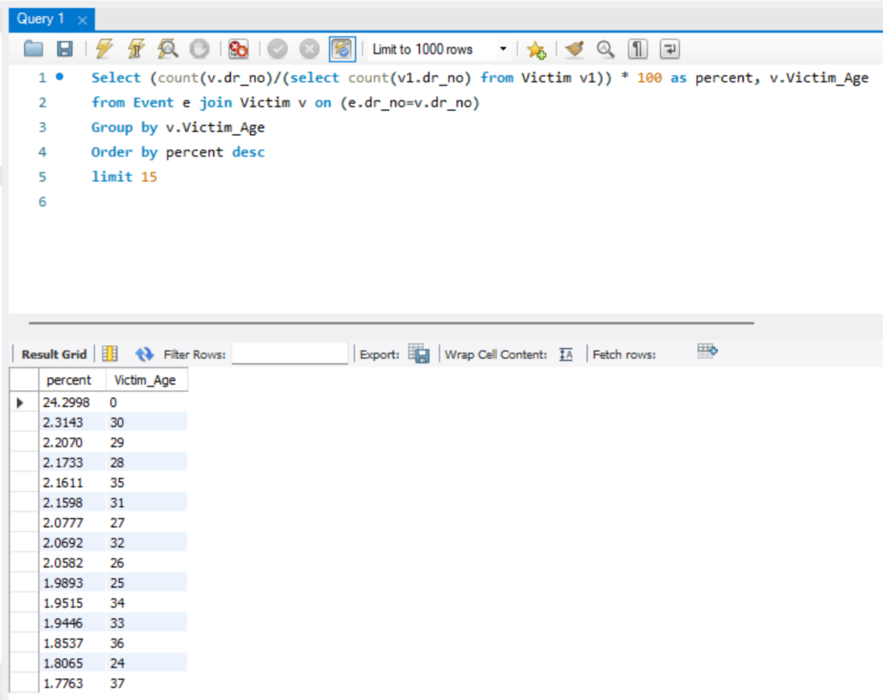
Analysis for Query 1:

Out of the default index, index_1, index_2, and index_3, we chose to use index_1 (indexing on Weapon(Weapon_type_code). This is since it had the best performance, with a cost of 903.74 for the limit. The other indexes all resulted in having 1251.98 as the cost for the limit. Indexing on Weapon(Weapon_type_code) is likely the best performing index since the group aggregate count function is done on Weapon_Type_Code, so indexing over Weapon_Type_Code saves a lot of time in accessing this attribute. Also, since the Event and Weapon tables are joined on Event.weapon_code and Weapon.Weapon_Type_Code, this indexing over Weapon_Type_Code likely speeds up finding the corresponding record in the Weapon table when iterating through the Event table to perform the join. The inverse is not true (indexing on Event.weapon_code does not speed up the join) probably because the query first iterates through each entry in event then searches for the corresponding Weapon_Type_Code in the Weapon table, never having to actually search the Event table beyond iterating through it. It probably does this since the Event.weapon_code is a foreign key that references the Weapon.Weapon_Type_Code rather than the inverse.

Query 2

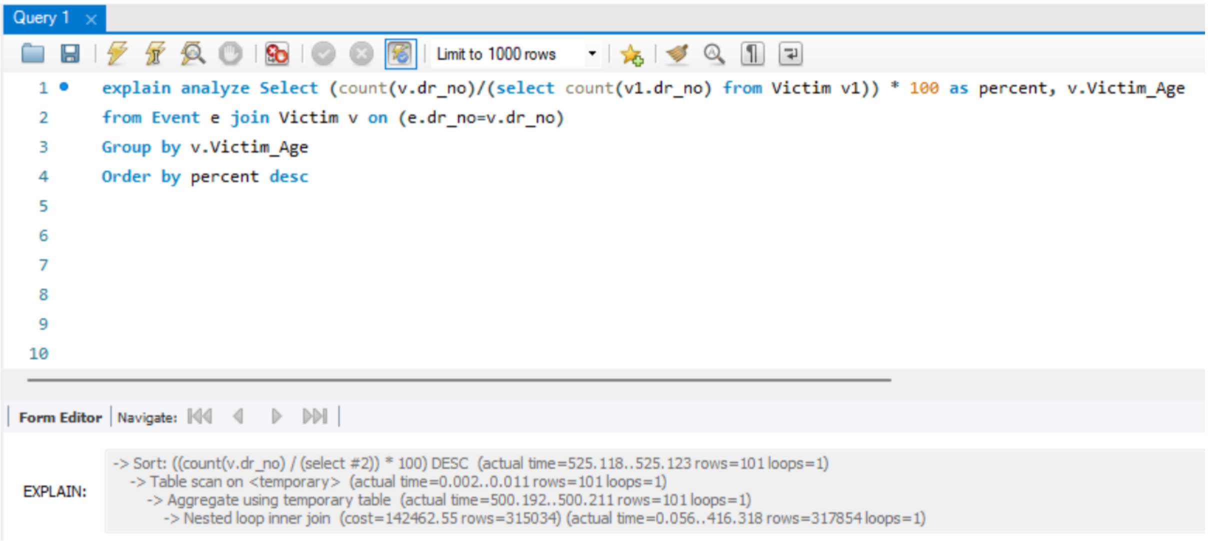
We want to know what is the distribution of age of victims, the query returns a percentage that corresponds to a victim age.

Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
limit 15



```
Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
limit 15
```

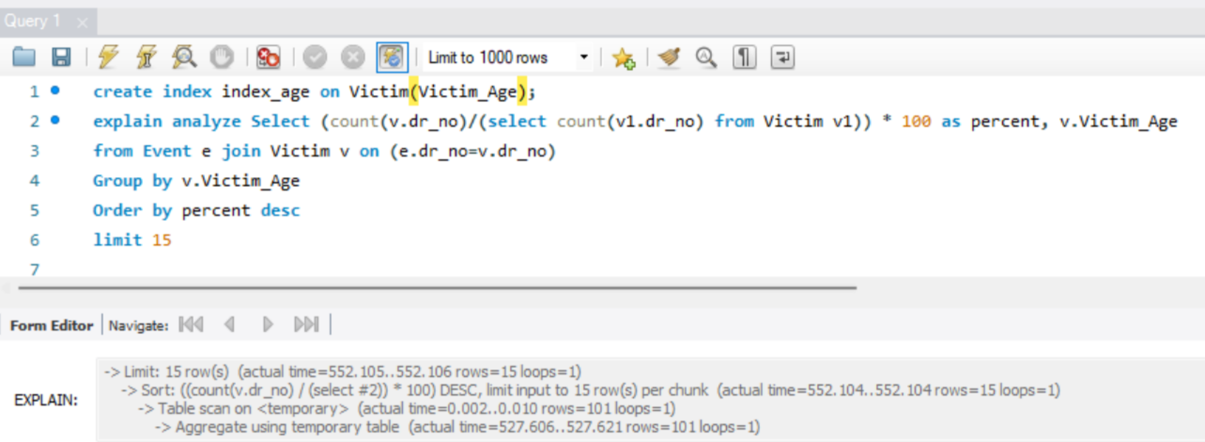
percent	Victim_Age
24.2998	0
2.3143	30
2.2070	29
2.1733	28
2.1611	35
2.1598	31
2.0777	27
2.0692	32
2.0582	26
1.9893	25
1.9515	34
1.9446	33
1.8537	36
1.8065	24
1.7763	37



```
explain analyze Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
```

EXPLAIN:

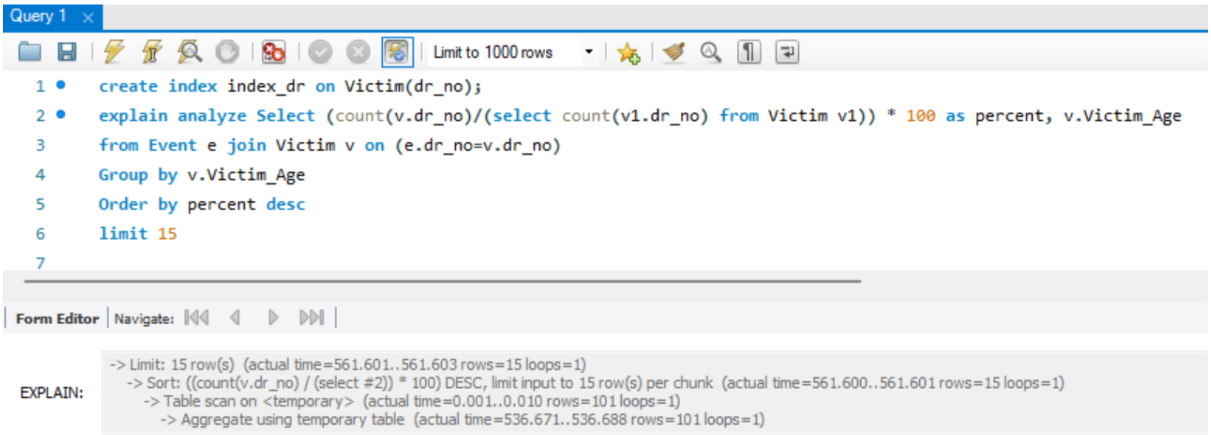
```
-> Sort: ((count(v.dr_no) / (select #2)) * 100) DESC  (actual time=525.118..525.123 rows=101 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.011 rows=101 loops=1)
        -> Aggregate using temporary table  (actual time=500.192..500.211 rows=101 loops=1)
            -> Nested loop inner join  (cost=142462.55 rows=315034) (actual time=0.056..416.318 rows=317854 loops=1)
```



```
create index index_age on Victim(Victim_Age);
explain analyze Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
limit 15
```

EXPLAIN:

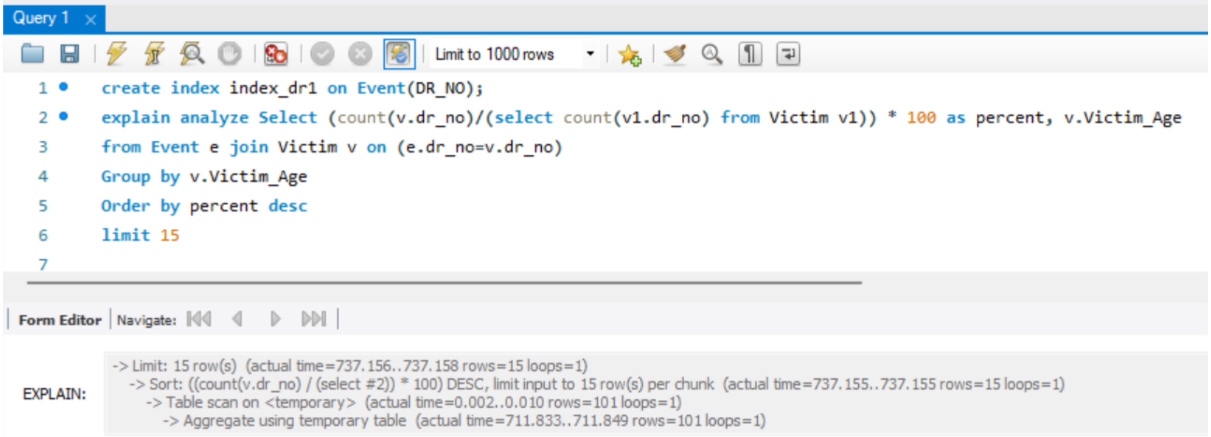
```
-> Limit: 15 row(s)  (actual time=552.105..552.106 rows=15 loops=1)
    -> Sort: ((count(v.dr_no) / (select #2)) * 100) DESC, limit input to 15 row(s) per chunk  (actual time=552.104..552.104 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=0.002..0.010 rows=101 loops=1)
            -> Aggregate using temporary table  (actual time=527.606..527.621 rows=101 loops=1)
```



```
create index index_dr on Victim(dr_no);
explain analyze Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
limit 15
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=561.601..561.603 rows=15 loops=1)
    -> Sort: ((count(v.dr_no) / (select #2)) * 100) DESC, limit input to 15 row(s) per chunk  (actual time=561.600..561.601 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=0.001..0.010 rows=101 loops=1)
            -> Aggregate using temporary table  (actual time=536.671..536.688 rows=101 loops=1)
```



```
create index index_dr1 on Event(DR_NO);
explain analyze Select (count(v.dr_no)/(select count(v1.dr_no) from Victim v1)) * 100 as percent, v.Victim_Age
from Event e join Victim v on (e.dr_no=v.dr_no)
Group by v.Victim_Age
Order by percent desc
limit 15
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=737.156..737.158 rows=15 loops=1)
    -> Sort: ((count(v.dr_no) / (select #2)) * 100) DESC, limit input to 15 row(s) per chunk  (actual time=737.155..737.155 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=0.002..0.010 rows=101 loops=1)
            -> Aggregate using temporary table  (actual time=711.833..711.849 rows=101 loops=1)
```

Analysis for Query 2:
Out of the default index, index_age, index_dr, and index_dr1, we chose to stick with the default index and not add the other indexes. This is since the default index had the best performance, with a time of 525.118 for the sorting, which was the limiting time in the query. The other indexes resulted in having 552.105 (index_age), 561.601 (index_dr), and 737.156 (index_dr1) for the time of the limit. Indexing on Victim(Victim_age) probably didn't improve the performance since the Victim.Victim_Age was mainly only used to group_by within the table, which does not require searching using indexes, rather only needing sorting within the table itself. Indexing on Victim(dr_no) didn't improve the performance since there was probably already a superior index on Victim.dr_no with the default index, since dr_no was the primary key of Victim. Similarly, indexing on Event(dr_no) didn't improve the performance since there was probably already a superior index on Event.dr_no with the default index, since dr_no was the primary key of Event.