DDL COMMANDS

```
DROP DATABASE IF EXISTS `CrimeData`;
CREATE DATABASE `CrimeData`;
USE `CrimeData`;

DROP TABLE IF EXISTS `Area`;
CREATE TABLE `Area`(
    `areaCode` INT NOT NULL,
    `areaName` VARCHAR(30) NOT NULL,
    PRIMARY KEY (`areaCode`)
);

DROP TABLE IF EXISTS `Location`;
CREATE TABLE `Location`(
    `locationID` INT AUTO_INCREMENT,
    `lat` REAL NOT NULL,
    `long` REAL NOT NULL,
    `addr` VARCHAR(255) NOT NULL,
    `premisCode` INT NOT NULL,
    `premisDesc` VARCHAR(255) NOT NULL,
    `areaCode` INT NOT NULL,
    PRIMARY KEY (`locationID`),
    FOREIGN KEY (`areaCode`) REFERENCES `Area`(`areaCode`)
);

DROP TABLE IF EXISTS `Weapon`;
CREATE TABLE `Weapon`(
    `weaponUsedCode` INT,
    `weaponDesc` VARCHAR(255),
    PRIMARY KEY (`weaponUsedCode`)
);

DROP TABLE IF EXISTS `Victim`;
CREATE TABLE `Victim`(
    `victimID` INT AUTO_INCREMENT,
    `age` INT,
    `sex` CHAR,
    `descent` CHAR,
    PRIMARY KEY (`victimID`)
);

DROP TABLE IF EXISTS `Crime`;
CREATE TABLE `Crime`(
    `crimeID` INT,
    `weaponUsedCode` INT,
    `locationID` INT NOT NULL,
    `victimID` INT,
    `dateRep` DATE NOT NULL,
```

```
    `dateOcc` DATE NOT NULL,
    `timeOcc` TIME NOT NULL,
    `code` INT NOT NULL,
    `codeDesc` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`crimeID`),
    FOREIGN KEY (`weaponUsedCode`) REFERENCES `Weapon` (`weaponUsedCode`),
    FOREIGN KEY (`locationID`) REFERENCES `Location` (`locationID`),
    FOREIGN KEY (`victimID`) REFERENCES `Victim` (`victimID`)
);
```

ADVANCED QUERIES

1. ***List of the all the crimes of the most common weapon code with victims ages 30+***

```
--------------------
SELECT
	a.areaCode,
	a.areaName,
	c.crimeID,
	v.age,
	w.weaponUsedCode
FROM
	Crime c
NATURAL JOIN
	Location l
NATURAL JOIN
	Area a
NATURAL JOIN Weapon w
NATURAL JOIN Victim v
WHERE
	w.weaponUsedCode
	=
	(
SELECT
			weaponUsedCode
FROM
	(
		SELECT weaponUsedCode, COUNT(weaponUsedCode) AS codeCount
		FROM Crime
		GROUP BY weaponUsedCode
	) e
WHERE
	codeCount >= ALL(
		SELECT COUNT(weaponUsedCode)
		FROM Crime
		GROUP BY weaponUsedCode)
	)
AND
	v.age > 30

ORDER BY
	a.areaCode ASC

--------------------
```

Crime (255r × 5c)

areaCode	areaName	crimeID	age	weaponUsedCode
1	Central	210,115,274	31	-1
1	Central	210,115,284	34	-1
1	Central	210,115,270	50	-1
1	Central	210,115,263	31	-1
1	Central	210,115,283	48	-1
1	Central	210,115,272	31	-1
1	Central	210,115,253	45	-1
1	Central	210,115,251	37	-1
1	Central	210,115,223	40	-1
1	Central	210,115,250	32	-1
1	Central	210,115,244	67	-1
1	Central	210,115,238	55	-1
1	Central	210,115,235	36	-1
1	Central	210,115,172	48	-1
1	Central	210,115,164	40	-1
2	Rampart	210,213,322	41	-1
2	Rampart	210,213,334	36	-1
2	Rampart	210,213,308	31	-1
3	Southwest	210,314,449	44	-1
3	Southwest	210,314,425	42	-1
3	Southwest	210,314,443	33	-1
3	Southwest	210,314,421	42	-1
3	Southwest	210,314,447	38	-1
3	Southwest	210,314,427	64	-1

INDEXING

NO INDEX

-> Sort: a.areaCode (actual time=4.965..5.003 rows=255 loops=1)
  -> Stream results (cost=356.94 rows=199) (actual time=0.056..4.781 rows=255 loops=1)
    -> Nested loop inner join (cost=356.94 rows=199) (actual time=0.052..4.632 rows=255 loops=1)
      -> Nested loop inner join (cost=287.30 rows=199) (actual time=0.047..4.143 rows=255 loops=1)

```
            -> Nested loop inner join  (cost=217.65 rows=199) (actual time=0.041..3.427 rows=255
loops=1)
                -> Filter: (v.age > 30)  (cost=101.00 rows=333) (actual time=0.025..0.735 rows=467 loops=1)
                    -> Table scan on v  (cost=101.00 rows=1000) (actual time=0.024..0.527 rows=1000
loops=1)
                -> Filter: (c.weaponUsedCode = (select #2))  (cost=0.25 rows=1) (actual time=0.005..0.005
rows=1 loops=467)
                    -> Index lookup on c using victimID (victimID=v.victimID)  (cost=0.25 rows=1) (actual
time=0.004..0.005 rows=1 loops=467)
                    -> Select #2 (subquery in condition; run only once)
                        -> Filter: <not>((e.codeCount < <max>(select #4)))  (cost=301.92..81.67 rows=667)
(actual time=0.711..0.720 rows=1 loops=1)
                            -> Table scan on e  (cost=302.26..317.25 rows=1000) (actual time=0.395..0.400
rows=32 loops=1)
                                -> Materialize  (cost=302.25..302.25 rows=1000) (actual time=0.393..0.393
rows=32 loops=1)
                                    -> Group aggregate: count(Crime.weaponUsedCode)  (cost=202.25 rows=1000)
(actual time=0.237..0.375 rows=32 loops=1)
                                        -> Covering index scan on Crime using weaponUsedCode  (cost=102.25
rows=1000) (actual time=0.054..0.292 rows=1000 loops=1)
                            -> Select #4 (subquery in condition; run only once)
                                -> Group aggregate: count(Crime.weaponUsedCode)  (cost=202.25 rows=1000)
(actual time=0.181..0.306 rows=32 loops=1)
                                    -> Covering index scan on Crime using weaponUsedCode  (cost=102.25
rows=1000) (actual time=0.016..0.232 rows=1000 loops=1)
            -> Single-row index lookup on l using PRIMARY (locationID=c.locationID)  (cost=0.25 rows=1)
(actual time=0.002..0.002 rows=1 loops=255)
        -> Single-row index lookup on a using PRIMARY (areaCode=l.areaCode)  (cost=0.25 rows=1)
(actual time=0.001..0.002 rows=1 loops=255)
```

CREATE INDEX areaName_idx ON `Area`(areaName);

```
-> Sort: a.areaCode  (actual time=3.098..3.130 rows=255 loops=1)
    -> Stream results  (cost=356.94 rows=199) (actual time=0.052..2.995 rows=255 loops=1)
        -> Nested loop inner join  (cost=356.94 rows=199) (actual time=0.049..2.899 rows=255 loops=1)
            -> Nested loop inner join  (cost=287.30 rows=199) (actual time=0.044..2.584 rows=255 loops=1)
                -> Nested loop inner join  (cost=217.65 rows=199) (actual time=0.038..2.158 rows=255
loops=1)
                    -> Filter: (v.age > 30)  (cost=101.00 rows=333) (actual time=0.024..0.431 rows=467 loops=1)
                        -> Table scan on v  (cost=101.00 rows=1000) (actual time=0.023..0.293 rows=1000
loops=1)
                    -> Filter: (c.weaponUsedCode = (select #2))  (cost=0.25 rows=1) (actual time=0.003..0.003
rows=1 loops=467)
                        -> Index lookup on c using victimID (victimID=v.victimID)  (cost=0.25 rows=1) (actual
time=0.002..0.003 rows=1 loops=467)
                        -> Select #2 (subquery in condition; run only once)
                            -> Filter: <not>((e.codeCount < <max>(select #4)))  (cost=301.92..81.67 rows=667)
(actual time=0.730..0.738 rows=1 loops=1)
```

-> Table scan on e (cost=302.26..317.25 rows=1000) (actual time=0.396..0.401 rows=32 loops=1)
-> Materialize (cost=302.25..302.25 rows=1000) (actual time=0.394..0.394 rows=32 loops=1)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.241..0.377 rows=32 loops=1)
-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.045..0.295 rows=1000 loops=1)
-> Select #4 (subquery in condition; run only once)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.195..0.324 rows=32 loops=1)
-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.015..0.236 rows=1000 loops=1)
-> Single-row index lookup on l using PRIMARY (locationID=c.locationID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)
-> Single-row index lookup on a using PRIMARY (areaCode=l.areaCode) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)

CREATE INDEX vicAge_idx ON `Victim`(age);

-> Sort: a.areaCode (actual time=2.934..2.985 rows=255 loops=1)
-> Stream results (cost=452.40 rows=279) (actual time=0.038..2.831 rows=255 loops=1)
-> Nested loop inner join (cost=452.40 rows=279) (actual time=0.034..2.735 rows=255 loops=1)
-> Nested loop inner join (cost=354.82 rows=279) (actual time=0.029..2.377 rows=255 loops=1)
-> Nested loop inner join (cost=257.24 rows=279) (actual time=0.022..1.948 rows=255 loops=1)
-> Filter: (v.age > 30) (cost=93.79 rows=467) (actual time=0.007..0.249 rows=467 loops=1)
-> Covering index range scan on v using vicAge_idx over (30 < age) (cost=93.79 rows=467) (actual time=0.007..0.154 rows=467 loops=1)
-> Filter: (c.weaponUsedCode = (select #2)) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=467)
-> Index lookup on c using victimID (victimID=v.victimID) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=467)
-> Select #2 (subquery in condition; run only once)
-> Filter: <not>((e.codeCount < <max>(select #4))) (cost=301.92..81.67 rows=667) (actual time=0.745..0.753 rows=1 loops=1)
-> Table scan on e (cost=302.26..317.25 rows=1000) (actual time=0.368..0.373 rows=32 loops=1)
-> Materialize (cost=302.25..302.25 rows=1000) (actual time=0.366..0.366 rows=32 loops=1)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.222..0.348 rows=32 loops=1)
-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.045..0.268 rows=1000 loops=1)
-> Select #4 (subquery in condition; run only once)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.231..0.359 rows=32 loops=1)

-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.019..0.284 rows=1000 loops=1)
-> Single-row index lookup on l using PRIMARY (locationID=c.locationID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)
-> Single-row index lookup on a using PRIMARY (areaCode=l.areaCode) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)

CREATE INDEX weapDesc_idx ON `Weapon`(weaponDesc);

-> Sort: a.areaCode (actual time=3.111..3.142 rows=255 loops=1)
-> Stream results (cost=356.94 rows=199) (actual time=0.081..3.008 rows=255 loops=1)
-> Nested loop inner join (cost=356.94 rows=199) (actual time=0.078..2.911 rows=255 loops=1)
-> Nested loop inner join (cost=287.30 rows=199) (actual time=0.071..2.585 rows=255 loops=1)
-> Nested loop inner join (cost=217.65 rows=199) (actual time=0.061..2.141 rows=255 loops=1)
-> Filter: (v.age > 30) (cost=101.00 rows=333) (actual time=0.024..0.414 rows=467 loops=1)
-> Table scan on v (cost=101.00 rows=1000) (actual time=0.023..0.280 rows=1000 loops=1)
-> Filter: (c.weaponUsedCode = (select #2)) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=467)
-> Index lookup on c using victimID (victimID=v.victimID) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=467)
-> Select #2 (subquery in condition; run only once)
-> Filter: <not>((e.codeCount < <max>(select #4))) (cost=301.92..81.67 rows=667) (actual time=0.702..0.710 rows=1 loops=1)
-> Table scan on e (cost=302.26..317.25 rows=1000) (actual time=0.359..0.364 rows=32 loops=1)
-> Materialize (cost=302.25..302.25 rows=1000) (actual time=0.358..0.358 rows=32 loops=1)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.204..0.340 rows=32 loops=1)
-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.036..0.261 rows=1000 loops=1)
-> Select #4 (subquery in condition; run only once)
-> Group aggregate: count(Crime.weaponUsedCode) (cost=202.25 rows=1000) (actual time=0.182..0.305 rows=32 loops=1)
-> Covering index scan on Crime using weaponUsedCode (cost=102.25 rows=1000) (actual time=0.016..0.229 rows=1000 loops=1)
-> Single-row index lookup on l using PRIMARY (locationID=c.locationID) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)
-> Single-row index lookup on a using PRIMARY (areaCode=l.areaCode) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=255)

Analysis:

We choose to use the vicAge_idx. All three indexing methods we chose performed about the same with very close runtimes and costs. They all also appeared to take the same steps which was somewhat

puzzling despite our multiple reruns to confirm that this was indeed the case. Therefore we decided to stick with the vicAge_idx as in the query we directly compare and check the age.

2. *Grabs the number of crimes in each area code on a certain date*

```
SELECT
	a.areaCode,
	COUNT(crimeTime.CrimeID) as crimeCount
FROM
	Area a
	NATURAL JOIN Location l
	NATURAL JOIN (
		SELECT
			LocationID,
			CrimeID
		FROM
			Crime
		WHERE
			dateOcc = '2021-08-16'
	) crimeTime
GROUP BY
	a.areaCode
ORDER BY
	crimeCount DESC
```

NO INDEX

```
-> Sort: crimeCount DESC  (actual time=0.973..0.975 rows=21 loops=1)
   -> Table scan on <temporary>  (actual time=0.955..0.959 rows=21 loops=1)
      -> Aggregate using temporary table  (actual time=0.954..0.954 rows=21 loops=1)
         -> Nested loop inner join  (cost=172.25 rows=100) (actual time=0.079..0.889 rows=155 loops=1)
            -> Nested loop inner join  (cost=137.25 rows=100) (actual time=0.075..0.801 rows=155
loops=1)
               -> Filter: (Crime.dateOcc = DATE'2021-08-16')  (cost=102.25 rows=100) (actual
time=0.060..0.498 rows=155 loops=1)
                  -> Table scan on Crime  (cost=102.25 rows=1000) (actual time=0.049..0.347 rows=1000
loops=1)
               -> Single-row index lookup on l using PRIMARY (locationID=Crime.locationID)  (cost=0.25
rows=1) (actual time=0.002..0.002 rows=1 loops=155)
            -> Single-row covering index lookup on a using PRIMARY (areaCode=l.areaCode)  (cost=0.25
rows=1) (actual time=0.000..0.000 rows=1 loops=155)
```

CREATE INDEX dateOcc_idx on Crime(dateOcc);

```
-> Sort: crimeCount DESC  (actual time=0.746..0.748 rows=21 loops=1)
   -> Table scan on <temporary>  (actual time=0.722..0.726 rows=21 loops=1)
      -> Aggregate using temporary table  (actual time=0.721..0.721 rows=21 loops=1)
         -> Nested loop inner join  (cost=130.75 rows=155) (actual time=0.202..0.655 rows=155 loops=1)
            -> Nested loop inner join  (cost=76.50 rows=155) (actual time=0.195..0.562 rows=155 loops=1)
               -> Index lookup on Crime using dateOcc_idx (dateOcc=DATE'2021-08-16')  (cost=22.25
rows=155) (actual time=0.184..0.282 rows=155 loops=1)
               -> Single-row index lookup on l using PRIMARY (locationID=Crime.locationID)  (cost=0.25
rows=1) (actual time=0.002..0.002 rows=1 loops=155)
            -> Single-row covering index lookup on a using PRIMARY (areaCode=l.areaCode)  (cost=0.25
rows=1) (actual time=0.000..0.000 rows=1 loops=155)
```

CREATE INDEX dateRep_idx on Crime(dateRep);

```
-> Sort: crimeCount DESC  (actual time=0.977..0.979 rows=21 loops=1)
   -> Table scan on <temporary>  (actual time=0.958..0.961 rows=21 loops=1)
      -> Aggregate using temporary table  (actual time=0.957..0.957 rows=21 loops=1)
         -> Nested loop inner join  (cost=172.25 rows=100) (actual time=0.082..0.892 rows=155 loops=1)
            -> Nested loop inner join  (cost=137.25 rows=100) (actual time=0.076..0.799 rows=155
loops=1)
               -> Filter: (Crime.dateOcc = DATE'2021-08-16')  (cost=102.25 rows=100) (actual
time=0.061..0.496 rows=155 loops=1)
                  -> Table scan on Crime  (cost=102.25 rows=1000) (actual time=0.049..0.376 rows=1000
loops=1)
               -> Single-row index lookup on l using PRIMARY (locationID=Crime.locationID)  (cost=0.25
rows=1) (actual time=0.002..0.002 rows=1 loops=155)
            -> Single-row covering index lookup on a using PRIMARY (areaCode=l.areaCode)  (cost=0.25
rows=1) (actual time=0.000..0.000 rows=1 loops=155)
```

CREATE INDEX code_idx on Crime(`code`);

```
-> Sort: crimeCount DESC  (actual time=0.994..0.995 rows=21 loops=1)
   -> Table scan on <temporary>  (actual time=0.974..0.977 rows=21 loops=1)
      -> Aggregate using temporary table  (actual time=0.973..0.973 rows=21 loops=1)
         -> Nested loop inner join  (cost=172.25 rows=100) (actual time=0.075..0.910 rows=155 loops=1)
            -> Nested loop inner join  (cost=137.25 rows=100) (actual time=0.070..0.821 rows=155
loops=1)
               -> Filter: (Crime.dateOcc = DATE'2021-08-16')  (cost=102.25 rows=100) (actual
time=0.057..0.550 rows=155 loops=1)
                  -> Table scan on Crime  (cost=102.25 rows=1000) (actual time=0.045..0.426 rows=1000
loops=1)
               -> Single-row index lookup on l using PRIMARY (locationID=Crime.locationID)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=155)
            -> Single-row covering index lookup on a using PRIMARY (areaCode=l.areaCode)  (cost=0.25
rows=1) (actual time=0.000..0.000 rows=1 loops=155)
```

Analysis:
We decided to choose the dateOcc_idx because this is the one that is being used in the query. This is being used because our main objective is to try and find the crimes that have occured on a certain day.

Compared to having no index, the instance with the dateOcc_idx was significantly faster in runtime due to the index lookup. Having no index required a table scan as well as an additional filter

PROOF

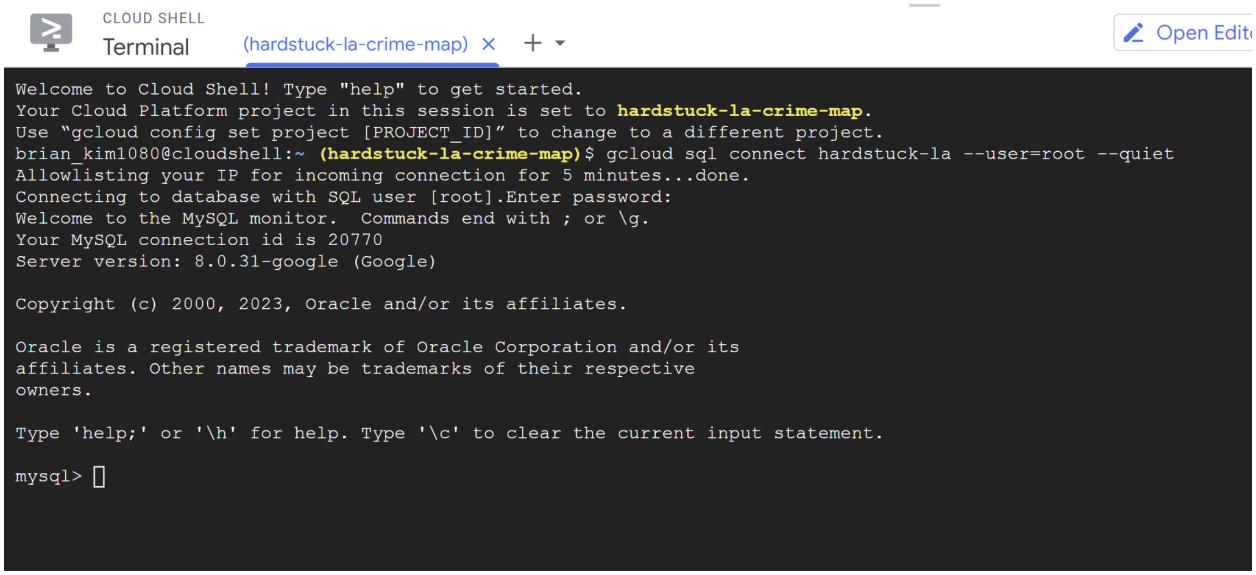
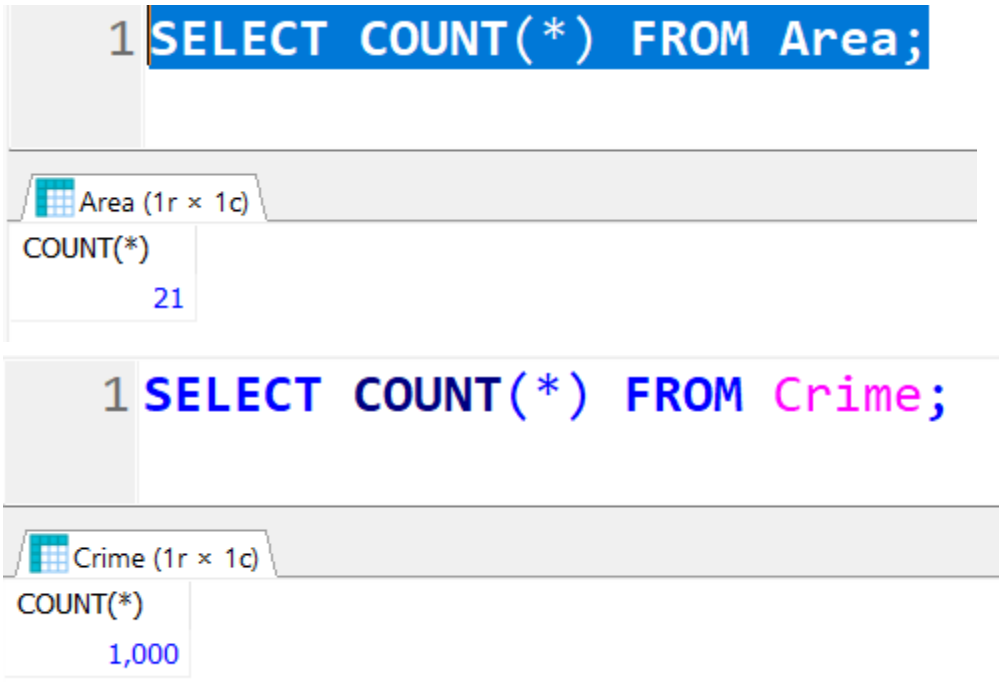


TABLE COUNTS



```
1 SELECT COUNT(*) FROM Location;
```

Location (1r × 1c)

COUNT(*)
1,000

```
1 SELECT COUNT(*) FROM Victim;
```

Victim (1r × 1c)

COUNT(*)
1,000

```
1 SELECT COUNT(*) FROM Weapon;
```

Weapon (1r × 1c)

COUNT(*)
32