US Shootings DBMS Final Report

CPS 510 - 032

November 30, 2020

Group Members

Manpreet Rajpal (500874585)

Sakshi Padhiar (500903964)

Alifiyah Hussain (500778471)

Nicolas de la Guardia (500806448)

Table of Contents

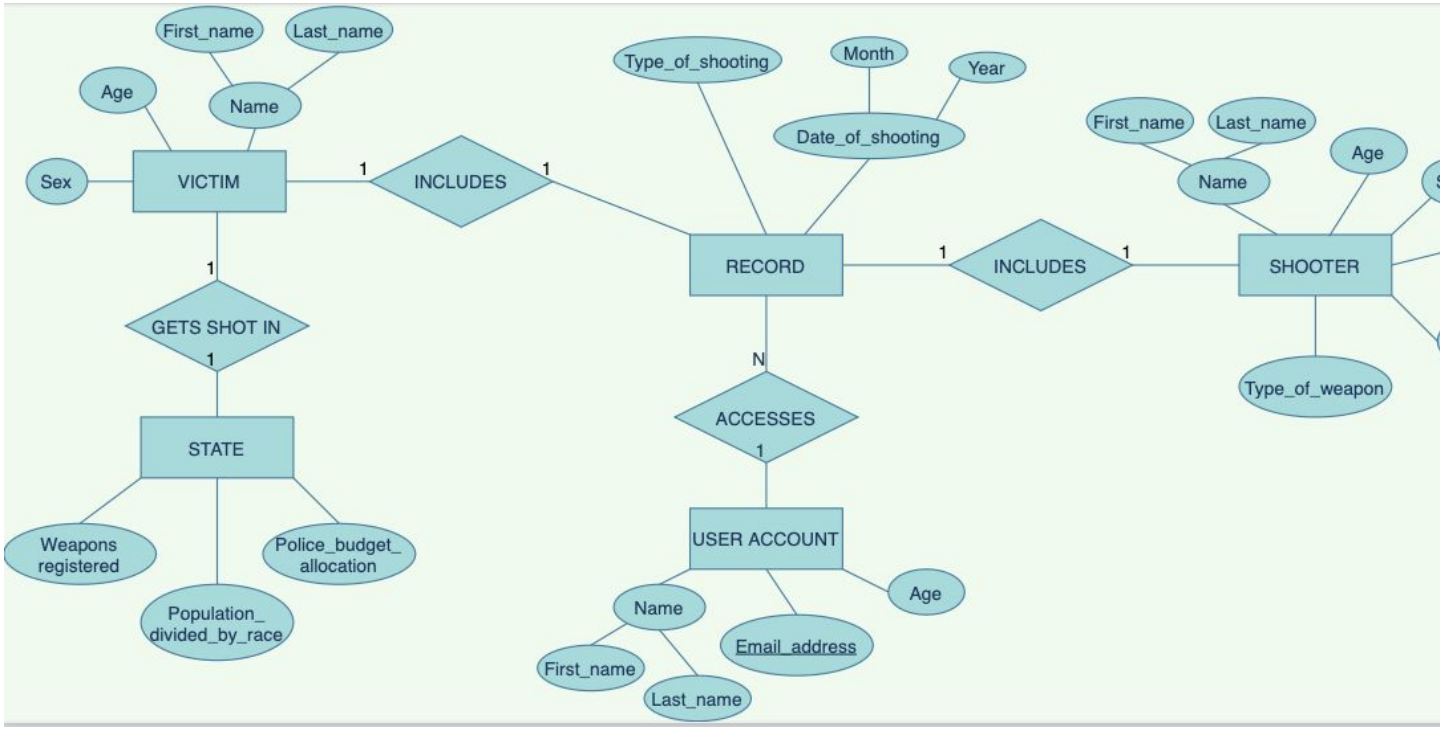
Assignment 1 Proposal: DBMS For Shootings in the United States

Shootings are a growing concern in the world's most dominant economic and military power, the United States. This problem has been around for decades but it has heightened to nearly an uncontrollable level in recent years. In light of recent events, such as the Black Lives Matter movement, it is important for citizens to have access to this information. The primary purpose of this database is to improve transparency and accountability. Although statistics do not capture the full extent of the story, they allow for a better understanding of the situation. For example currently, people are trying to understand whether police are disproportionately targeting people of colour, but simply looking at percentages of victims by race would prove otherwise. However, with a comprehensive database with multiple descriptors such as whether the victim was armed or not provides a whole new dimension to the story.

As a brief overview, our database application would include records of the number of shootings per state, information about the shooter (name, age, gender, race.), date of shooting, type of shooting (police, school etc.), number of deaths caused per year, how the person was killed (shot or tasered), if person was armed (gun, knife, other, none). Users can interact with the database in the following ways: they can contribute to the database (e.g. submit a shooting case, which must be approved by system administrators before being posted), discuss their thoughts with others either anonymously or by creating an account (this would require their full name, age, gender and email address) and subscribe to receive notifications when information is added or updated. Evidently, the database will be handling a large volume of data specifically from 2015 – present.

This application can be used by people who are curious about the correlations between certain individuals and their susceptibility to getting shot. It will bridge the gaps in their knowledge and hopefully lead to educated conclusions.

Assignment 2: ER model



Assignment 3: Schema design

DDL Commands

```
CREATE TABLE US_STATE (
        STATE_ID INT NOT NULL,
        WEAPONS NUMBER(10),
        POPULATION NUMBER(5),
        POLICE NUMBER(10),
        STATE_NAME VARCHAR(50) NOT NULL,
        PRIMARY KEY (STATE_ID)
);

CREATE TABLE VICTIM (
        VICTIM_ID INT NOT NULL,
        FIRST_NAME VARCHAR(50) NOT NULL,
        LAST_NAME VARCHAR(50) NOT NULL,
        GENDER VARCHAR(10),
        AGE NUMBER(3),
        STATE_ID INT NOT NULL,
        PRIMARY KEY (VICTIM_ID),
        FOREIGN KEY (STATE_ID) REFERENCES US_STATE(STATE_ID)
```

```
);

CREATE TABLE SHOOTER (
        SHOOTER_ID INT NOT NULL,
        FIRST_NAME VARCHAR(50) NOT NULL,
        LAST_NAME VARCHAR(50) NOT NULL,
        AGE NUMBER(3),
        GENDER VARCHAR(10),
        RACE VARCHAR(20),
        WEAPON_TYPE VARCHAR(50),
        OCCUPATION VARCHAR(50),
        STATE_ID INT NOT NULL,
        PRIMARY KEY (SHOOTER_ID),
        FOREIGN KEY (STATE_ID) REFERENCES US_STATE(STATE_ID)
);

CREATE TABLE RECORD (
        RECORD_ID INT NOT NULL,
        TYPE_OF_SHOOTING VARCHAR(50),
        MONTH_OF_SHOOTING VARCHAR(10),
        YEAR_OR_SHOOTING NUMBER(4),
        VICTIM_ID INT NOT NULL,
        SHOOTER_ID INT NOT NULL,
        PRIMARY KEY (RECORD_ID),
        FOREIGN KEY (VICTIM_ID) REFERENCES VICTIM(VICTIM_ID),
        FOREIGN KEY (SHOOTER_ID) REFERENCES SHOOTER(SHOOTER_ID)
);

CREATE TABLE USER_ACCOUNT (
        FIRST_NAME VARCHAR(50) NOT NULL,
        LAST_NAME VARCHAR(50) NOT NULL,
        AGE NUMBER(3),
        EMAIL VARCHAR(50),
        RECORD_ID INT NOT NULL,
        PRIMARY KEY (EMAIL),
        FOREIGN KEY (RECORD_ID) REFERENCES RECORD(RECORD_ID)
);
```

Table Creation Screenshots

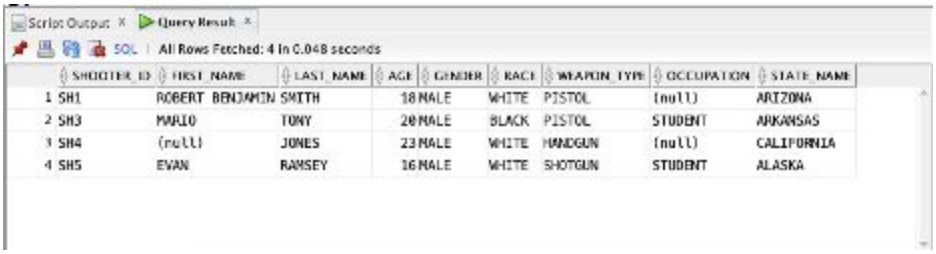
```
Table US_STATE created.

Table VICTIM created.

Table SHOOTER created.

Table RECORD created.

Table USER_ACCOUNT created.
```

Assignment 4 part 1: Demo of Designing Views/Simple Queries

Query 1:

```
SELECT * FROM
SHOOTER WHERE AGE
<= 30;
```

Output:



Script Output | Query Result

SQL | All Rows Fetched: 4 in 0.048 seconds

	SHOOTER_ID	FIRST_NAME	LAST_NAME	AGE	GENDER	RACE	WEAPON_TYPE	OCCUPATION	STATE_NAME
1	SH1	ROBERT BENJAMIN	SMITH	18	MALE	WHITE	PISTOL	(null)	ARIZONA
2	SH3	MARIO	TONY	20	MALE	BLACK	PISTOL	STUDENT	ARKANSAS
3	SH4	(null)	JONES	23	MALE	WHITE	HANDGUN	(null)	CALIFORNIA
4	SH5	EVAN	RAMSEY	16	MALE	WHITE	SHOTGUN	STUDENT	ALASKA

Query 2:

```
SELECT TYPE_OF_SHOOTING, MONTH_OF_SHOOTING,
YEAR_OF_SHOOTING FROM RECORD
WHERE YEAR_OF_SHOOTING = 2019;
```

Output:

Script Output | Query Result

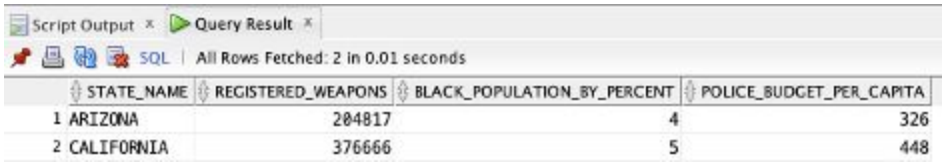
SQL | All Rows Fetched: 2 in 0.053 seconds

	TYPE_OF_SHOOTING	MONTH_OF_SHOOTING	YEAR_OF_SHOOTING
1	MASS	JANUARY	2019
2	SCHOOL	MARCH	2019

Query 3:

```
SELECT *
FROM US_STATE
WHERE REGISTERED_WEAPONS > 200000;
```

Output:



Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.01 seconds

	STATE_NAME	REGISTERED_WEAPONS	BLACK_POPULATION_BY_PERCENT	POLICE_BUDGET_PER_CAPITA
1	ARIZONA	204817	4	326
2	CALIFORNIA	376666	5	448

Query 4:

```
SELECT GENDER, AGE,
STATE_NAME FROM VICTIM
WHERE AGE >=19;
```

Output:

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.049 seconds

	GENDER	AGE	STATE_NAME
1	FEMALE	35	ALABAMA
2	MALE	19	ALASKA

Query 5:

```
SELECT *
FROM USER_ACCOUNT
WHERE NOT(GENDER = 'FEMALE');
```

Output:

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.01 seconds

	FIRST_NAME	LAST_NAME	AGE	GENDER	EMAIL	RECORD_ID
1	JEREMY	HIGGINS	32	MALE	JHIGGINS@GMAIL.COM	RCD1
2	MIHAIL	NOTAH	28	MALE	MIHAILNOTAH99@GMAIL.COM	RCD4
3	YONAH	RAYWOOD	45	MALE	YOHANRAYWOOD@GMAIL.COM	RCD5

Query 6:

```
SELECT DISTINCT STATE_NAME AS
WEALTHY_STATES FROM US_STATE
WHERE POLICE_BUDGET_PER_CAPITA >
'300' ORDER BY STATE_NAME;
```

Output:

Script Output | Query Result

SQL | All Rows Fetched: 3 in 0.009 seconds

	WEALTHY_STATES
1	ALASKA
2	ARIZONA
3	CALIFORNIA

Query 7:

```
SELECT RECORD.RECORD_ID,
VICTIM.STATE_NAME FROM RECORD
INNER JOIN VICTIM ON RECORD.VICTIM_ID = VICTIM.VICTIM_ID;
```

Output:

	RECORD_ID	STATE_NAME
1	RCD1	ARIZONA
2	RCD2	ALABAMA
3	RCD3	ARKANSAS
4	RCD4	CALIFORNIA
5	RCD5	ALASKA

Query 8:

```
SELECT SHOOTER_ID, LAST_NAME, AGE, GENDER, RACE,
STATE_NAME FROM SHOOTER
WHERE (RACE = 'BLACK' AND STATE_NAME =
'ARIZONA') OR
(RACE = 'WHITE' AND STATE_NAME = 'ARIZONA');
```

Output:

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.008 seconds

	SHOOTER_ID	LAST_NAME	AGE	GENDER	RACE	STATE_NAME
1	SH1	SMITH	18	MALE	WHITE	ARIZONA

Assignment 4 (Part 2) : Complex Queries

Query 1:
SELECT CAST(AVG(shooter.age) as int) as avg_age FROM SHOOTER;
Output:

	AVG_AGE
1	23

Query 2:
SELECT COUNT(shooter_id), race
FROM shooter
GROUP BY race;
Output:

	COUNT(SHOOTER_ID)	RACE
1	3	WHITE
2	2	BLACK

Query 3:
SELECT record.record_id, victim.state_name FROM record INNER JOIN victim ON record.victim_id = victim.victim_id;

Output:

	RECORD_ID	STATE_NAME
1	RCD1	ARIZONA
2	RCD2	ALABAMA
3	RCD3	ARKANSAS
4	RCD4	CALIFORNIA
5	RCD5	ALASKA

Query 4:
SELECT COUNT(type_of_shooting), type_of_shooting
FROM record
GROUP BY type_of_shooting
ORDER BY COUNT(type_of_shooting) DESC;

Output:

	COUNT(TYPE_OF_SHOOTING)	TYPE_OF_SHOOTING
1	3	SCHOOL
2	1	MASS
3	1	RESIDENTIAL

Query 5:

```
SELECT *
FROM US_STATE
WHERE POLICE_BUDGET_PER_CAPITA BETWEEN 300 AND 500;
```

Output:

	STATE_NAME	REGISTERED_WEAPONS	BLACK_POPULATION_BY_PERCENT	POLICE_BUDGET_PER_CAPITA
1	ALASKA	20520	3	494
2	ARIZONA	204817	4	326
3	CALIFORNIA	376666	5	448

Query 6:

```
SELECT us_state.state_name, us_state.registered_weapons,
us_state.police_budget_per_capita,
us_state.black_population_by_percent, shooter.race
FROM us_state RIGHT JOIN shooter
ON shooter.state_name = us_state.state_name
ORDER BY(registered_weapons) DESC;
```

Output:

	STATE_NAME	REGISTERED_WEAPONS	POLICE_BUDGET_PER_CAPITA	BLACK_POPULATION_BY_PERCENT	RACE
1	CALIFORNIA	376666	448	5	WHITE
2	ARIZONA	204817	326	4	WHITE
3	ALABAMA	168265	261	26	BLACK
4	ARKANSAS	108801	228	15	BLACK
5	ALASKA	20520	494	3	WHITE

Query 7:

```
CREATE VIEW male_accounts AS
SELECT FIRST_NAME, LAST_NAME
FROM USER_ACCOUNT
WHERE GENDER = 'MALE';
```

SELECT * FROM male_accounts;

Output:

	FIRST_NAME	LAST_NAME
1	JEREMY	HIGGINS
2	MIHAIL	NOTAH
3	YONAH	RAYWOOD

Query 8:

```
CREATE VIEW teenage_victims AS
SELECT FIRST_NAME, LAST_NAME
FROM VICTIM
WHERE AGE BETWEEN 13 AND 19;
```

SELECT * FROM teenage_victims;

Output:

	FIRST_NAME	LAST_NAME
1	JAMIE	WILLINGER
2	JOHNATHAN	MILLS

Query 9:

```
CREATE VIEW school_shooting_dates AS
SELECT MONTH_OF_SHOOTING, YEAR_OF_SHOOTING
FROM RECORD
WHERE TYPE_OF_SHOOTING = 'SCHOOL';
```

SELECT * FROM school_shooting_dates;

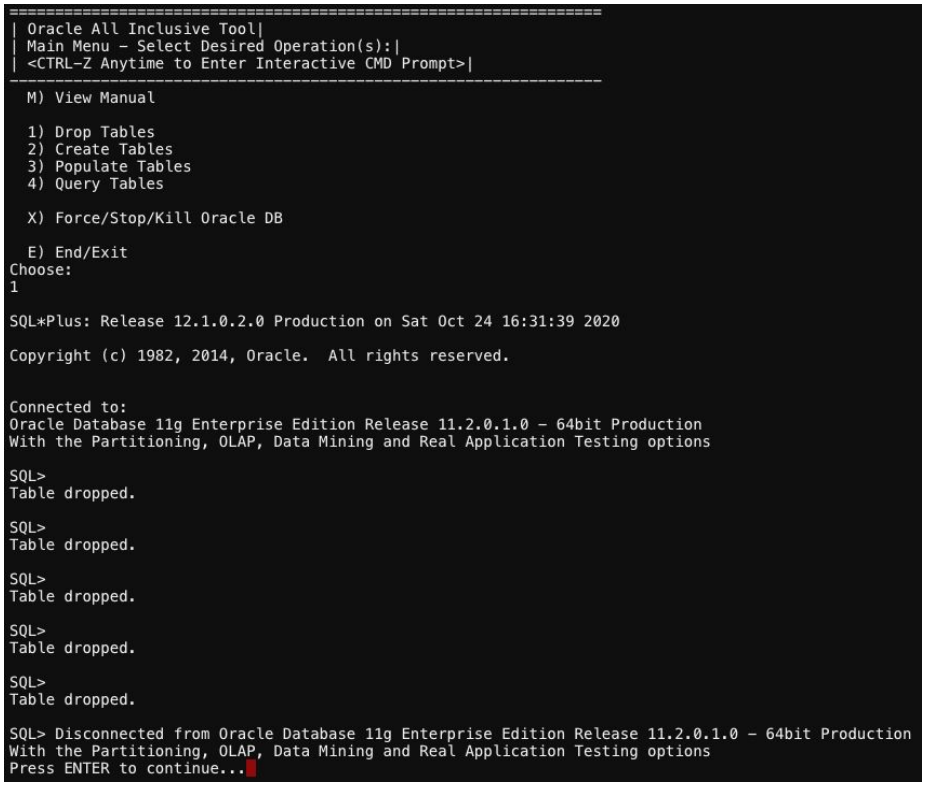
Output:

	MONTH_OF_SHOOTING	YEAR_OF_SHOOTING
1	NOVEMBER	2018
2	OCTOBER	2018
3	MARCH	2019

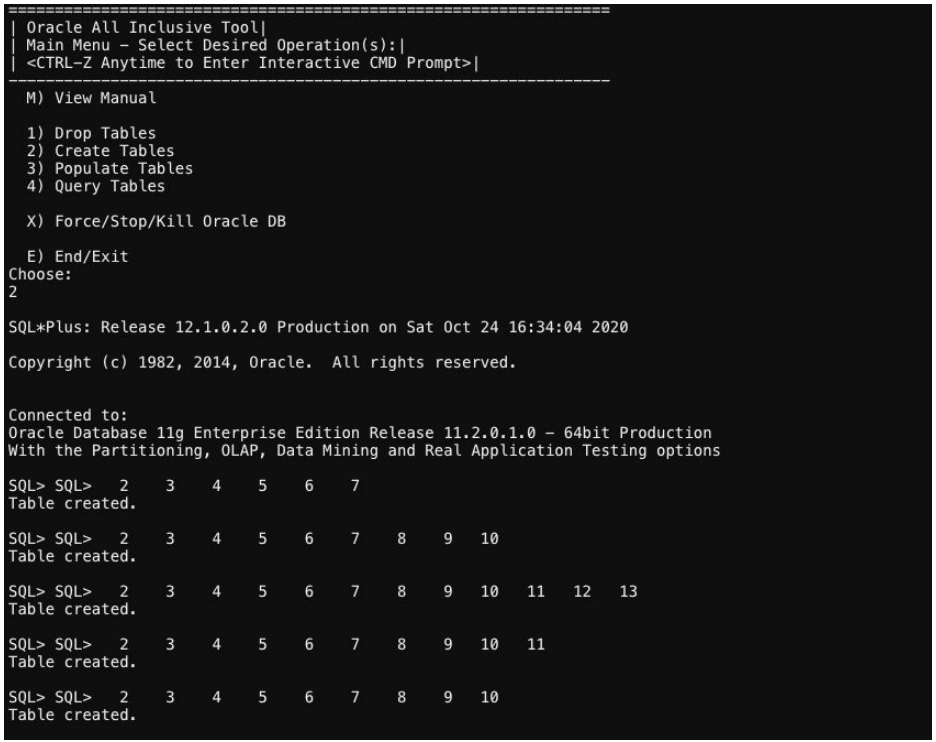
Assignment 5: Demonstration of advanced queries By Unix shell Implementation

Unix Shell Script demo:

1) Dropping tables



2) Creating tables



3) Populating tables

```
================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
----------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
3

SQL*Plus: Release 12.1.0.2.0 Production on Sat Oct 24 16:35:09 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.
```

4) Querying (photos only display a small portion of the queries for the sake of conciseness)

```
============================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
4

SQL*Plus: Release 12.1.0.2.0 Production on Mon Oct 26 21:24:00 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>   2
SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
SH1        ROBERT BENJAMIN
SMITH                                                18 MALE
WHITE                PISTOL

SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------

ARIZONA
```

```
SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
SH3        MARIO
TONY                                                 20 MALE
BLACK                PISTOL

SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
STUDENT
ARKANSAS


SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
SH4
JONES                                                23 MALE
WHITE                HANDGUN

SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                           AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
```

```
SQL> SQL>   2    3    4    5
STATE_NAME                                         REGISTERED_WEAPONS
-------------------------------------------------- ------------------
POLICE_BUDGET_PER_CAPITA BLACK_POPULATION_BY_PERCENT RACE
------------------------ --------------------------- --------------------
CALIFORNIA                                                     376666
                   448                           5 WHITE

ARIZONA                                                        204817
                   326                           4 WHITE

ALABAMA                                                        168265
                   261                          26 BLACK


STATE_NAME                                         REGISTERED_WEAPONS
-------------------------------------------------- ------------------
POLICE_BUDGET_PER_CAPITA BLACK_POPULATION_BY_PERCENT RACE
------------------------ --------------------------- --------------------
ARKANSAS                                                       108801
                   228                          15 BLACK

ALASKA                                                          20520
                   494                           3 WHITE


SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press ENTER to continue...
```

Assignment 6: Normalization of the Database /Functional Dependencies

Functional Dependencies
State table:
State_name → {registered_weapons, police_budget_per_capita, black_population_by_percent}

Victim table:
Victim_id → {first_name, last_name, gender, age}
Victim_id → record_id
Victim_id → shooter_id
Victim_id → state_name

Shooter table:
Shooter_id → {first_name, last_name, age, gender, race, weapon_type, occupation}
Shooter_id → state_name

Record table:
Record_id → {type_of_shooting, month_of_shooting, year_of_shooting, victim_id}

User_account table:
Email → {first_name, last_name, age}
Email → record_id

Note:
After careful analysis, we concluded that our previously defined relationship between the entities record and shooter is redundant in the sense that the following functional dependency is partial: victim_id, shoooter_id → record_id. To fix this, shooter_id is now a foreign key in the victim table and removed from the record table.

Assignment 7: Normalization / 3rd NF

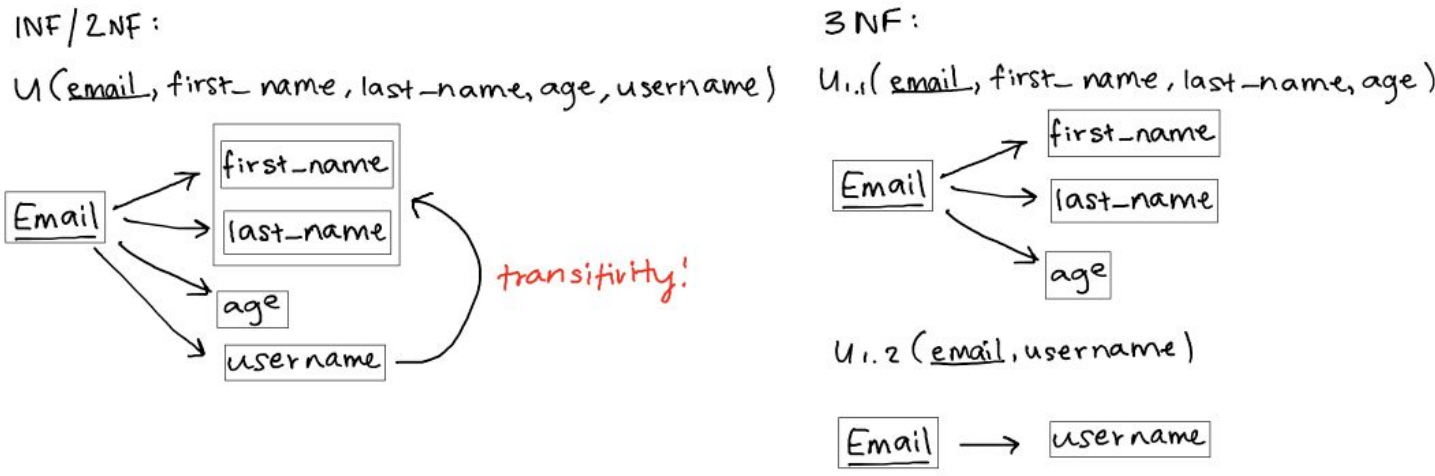
User Account Table
1NF/2NF:
U = (email, first_name, last_name, age, username)

Functional Dependencies:
Email → first_name, last_name, age, username
Username → first_name, last_name

Transitivity: Email → Username → first_name, last_name

3NF:
U1.1(email, first_name, last_name, age)
U1.2(email, username)

Diagram representation:

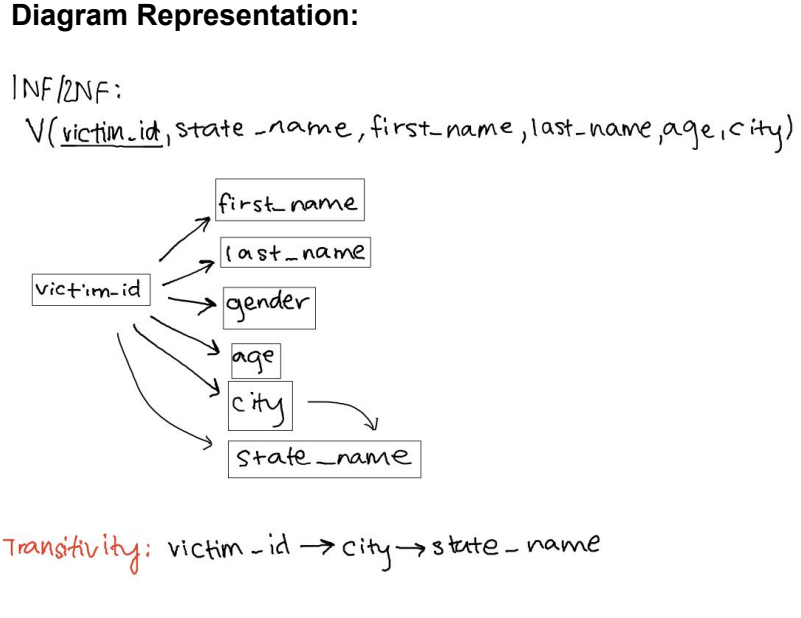


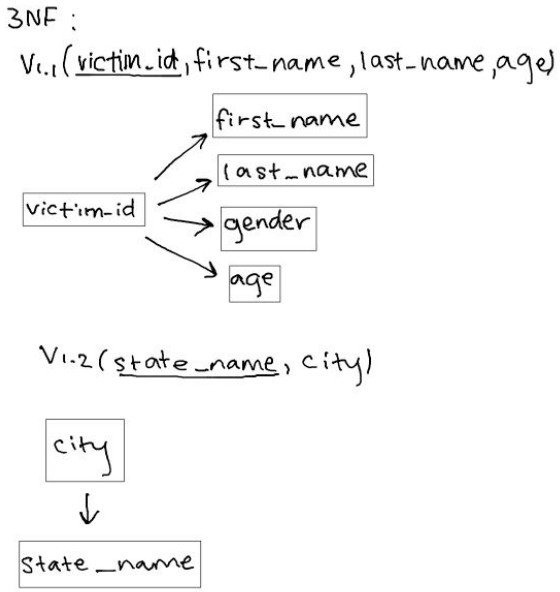
Victim Table
1NF/2NF:
V = (victim_id, state_name, first_name, last_name, gender, age, city)

Functional Dependencies:
Victim_id → first_name, last_name, gender, age, city, state_name
City → State_name
Transitivity: victim_id → city → state_name

3NF:
V1.1(victim_id, first_name, last_name, gender, age, city)
V1.2(city, state_name)

Diagram Representation:





Shooter Table:

1NF/2NF:

S = (Shooter_id, first_name, last_name_, age, gender, race, weapon, weapon_type, occupation, city, state_name)

Functional Dependencies:

Shooter_id → first_name, last_name, age, gender, race, weapon, weapon_type, occupation, city, state_name

Weapon → weapon_type

City → state_name

Transitivity:

shooter_id → city → state_name

Shooter_id → weapon → weapon_type

3NF:

S1.1(shooter_id, first_name, last_name, gender, age, race, weapon, occupation, city)

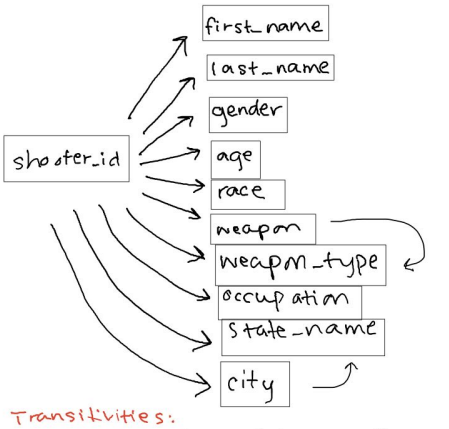
S1.2(city, state_name)

S1.3(weapon, weapon_type)

Diagram Representation:

1NF/2NF:

S = (shooter_id, first_name, last_name, age, gender, race, weapon, weapon_type, occupation, state_name)



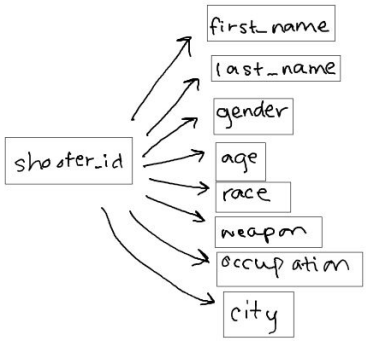
Transitivities:

shooter_id → city → state_name

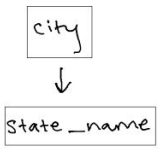
shooter_id → weapon → weapon_type

3NF:

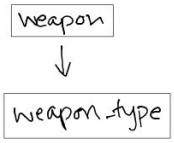
S1.1 (shooter_id, first_name, last_name, age, gender, race, weapon, occupation, city)



S1.2 (city, state_name)



S1.3 (weapon, weapon_type)



Note:

The Record and State tables are already in third normal form for the following reasons:

1. They are in second normal form (as per assignment 6)
    a. Each of their non-prime attributes are fully functionally dependent on their respective primary keys
    b. There are no partial dependencies
2. There are no transitive functional dependencies.

Assignment 8: Normalization 3NF/BCNF

Shooter table:
Shooter_id(I), first_name(F), last_name(L), age(A), gender(G), race(R), weapon(W), weapon_type(T), occupation(O), city(C), state_name(S).

Bernstein's Algorithm :- Relation: R(I, F, L, A, G, R, W, T, O, C, S)

1) Functional dependencies

FD = {I → F, I → L, I → A, I → G, I → R, I → W, I → T, I → O, I → C, W → T, C → S, I → S}

2) Find redundancies

I → F: I+ = {I, L, A, G, R, W, T, O, C, S} we do not get F, so not redundant
I → L: I+ = {I, F, A, G, R, W, T, O, C, S} we do not get L, so not redundant
I → A: I+ = {I, F, L, G, R, W, T, O, C, S} we do not get A, so not redundant
I → G: I+ = {I, F, L, A, R, W, T, O, C, S} we do not get G, so not redundant
I → R: I+ = {I, F, L, A, G, W, T, O, C, S} we do not get R, so not redundant
I → W: I+ = {I, F, L, A, G, R, W, T, O, C, S} we get W, so this FD is redundant
I → T: I+ = {I, F, L, A, G, R, W, O, C, S} we do not get T, so not redundant
I → O: I+ = {I, F, L, A, G, R, W, T, C, S} we do not get O, so not redundant
I → C: I+ = {I, F, L, A, G, R, W, T, O, C, S} we get C, so this FD is redundant
W → T: W+ = {W} we do not get T, so not redundant
C → S: C+ = {C} we do not get S, so not redundant
I → S: I+ = {I, F, L, A, G, R, W, T, O, C, S} we get S, so this FD is redundant

After removing redundancies, FD = {I → F, L, A, G, R, T, O, W → T, C → S}
Note: there are no partial dependencies.

3) Find keys

{I,W,C} are part of the key since they are in LHS and not in RHS
{F, L, A, G, R, T, O, S} cannot be part of the key since they are in RHS and not in LHS
Thus, IWC+ = {I, W, C, F, L, A, G, R, T, O, S} is the key since it determines all of the attributes

4) Make tables

We have FD = {I → F, L, A, G, R, T, O, W → T, C → S}
R1(I, F, L, A, G, R, T, O) with FD: I → F, L, A, G, R, T, O
R2(W,T) with FD: W → T
R3(C,S) with FD: C → S
R4(I,W,C) with no FD

User_account table:
Email(E) → first_name(F), last_name(L), age(A), username(U)
Username(U) → first_name(F), last_name(L)

Email(E) → record_id(R)

Bernstein's Algorithm :- Relation = {E, F, L, A, U, R}

1) Functional dependencies
   FD = { E → F, E → L, E → A, E → U, U → F, L, E → R, EU → F, EU → L}

2) Find redundancies
   E → F: E+= {E, L, A, U, F, R} we get F, so this FD is redundant
   E → L: E+= {E, F, A, U, L, R} we get L, so this FD is redundant
   E → A: E+= {E, F, L, U, R} we did not get A so this FD is not redundant
   E → U: E+= {E, F, L, A, R} we did not get U so this FD is not redundant
   U → F: U+= {U, L} we did not get F so this FD is not redundant
   U → L: U+= {U, F} we did not get L so this FD is not redundant
   E → R: E+= {E, F, L, A, U} we did not get R so this FD is not redundant
   EU → F: EU+={E, U, F, L, A, R} we got F so this FD is redundant
   EU → L: EU+={E, U, F, L, A, R} we got L so this FD is redundant

   Removing partial dependencies:
   EU → F: E+={E, F, L, A, U, R}, U+= {U, F, L} we get F, so this FD is not fully dependent
   EU → L: E+={E, F, L, A, U, R}, U+= {U, F, L} we get L, so this FD is not fully dependent

   After removing redundancies: FD={E → A, U, R, U → F, U → L}

3) Find keys
   {E, U} are part of the key since they are on LHS and not RHS
   {A, R, F, L} cannot be part of the key as they are only on RHS and not the LHS
   Therefore,
   EU+ = {E, U, A,, R, F, L} is the key

4) Make tables
   We have FD={E → A, U, R, U → F, U → L}
   R1(E, A, U, R) with FD: E → A, U, R
   R2(U, F) with FD: U → F
   R3(U, L) with FD: U → L
   R4(E, U) with no FD

Victim table:

Victim_id(V) → first_name(F), last_name(L), gender(G), age(A), city(C), state_name(S)
City → State_name

Victim_id → record_id(R)
Victim_id → shooter_id(I)

BCNF Decomposition algorithm
Relation = {V, F, L, G, A, C, S, R, I}
FD's = {V → F L G A C S, C → S, V → R, V → I}

{V → F, V → L, V → G, V → A, V → C, V → S, C → S: V → R, V → I}

All are minimal cover except V → S

- All singletons on the RHS
- No extraneous LHS
- No redundant FD

V → S is redundant since S can be found with C → S.

R1 = {**V**,F}
R2 = {**V**,L}
R3 = {**V**,G}
R4 = {**V**,A}
R5 = {**V**,C}
R6 = {**C**,S}
R7 = {**V**,R}
R8 = {**V**,I}

S1 = {**V**, F, L, G, A, C, S, R, I}
S2 = {**C**,S}

The following tables are already in 3NF/BCNF for the following reasons:

1. Each non-prime attribute is dependent on a prime attribute
2. There are no partial dependencies
3. There are no transitive functional dependencies
4. Each attribute on the RHS is a key

Functional dependencies:
Record table:
Record_id → type_of_shooting, month_of_shooting, year_of_shooting, victim_id

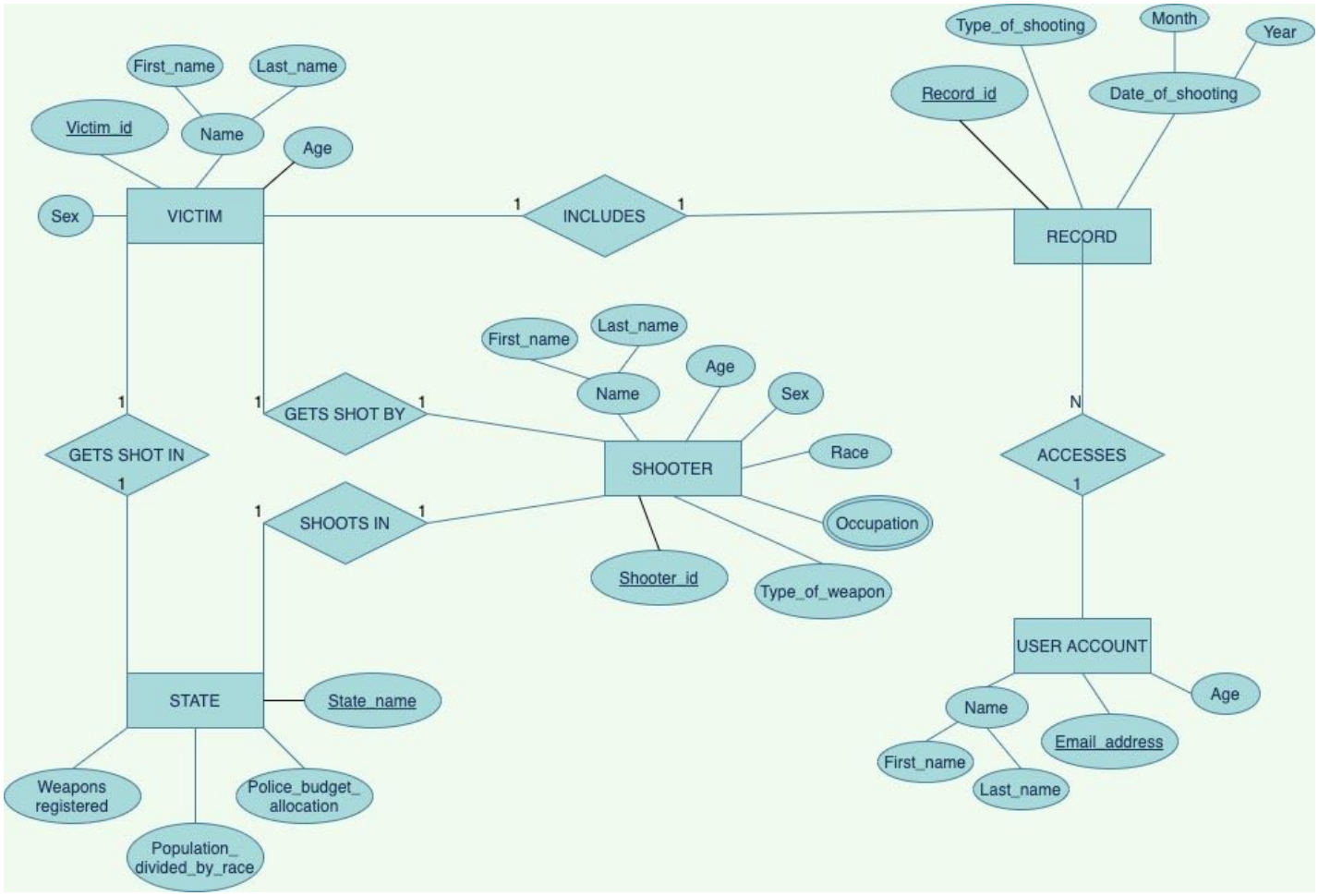
State table:
State_name → registered_weapons, police_budget_per_capita, black_population_by_percent

Assignment 9 /10: Demonstration of User Interface

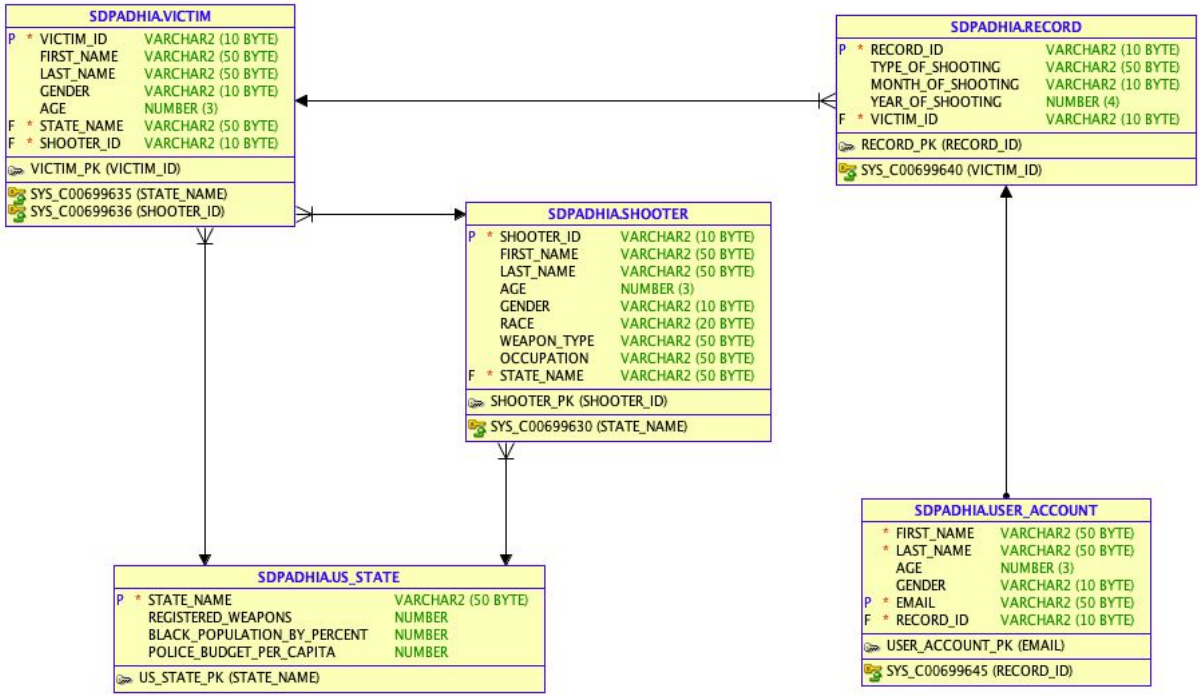
NOTE: U1 demoed during lab and screen captures included below

Assignment 10: Relational Algebra and Final Documentation

Final ER Diagram

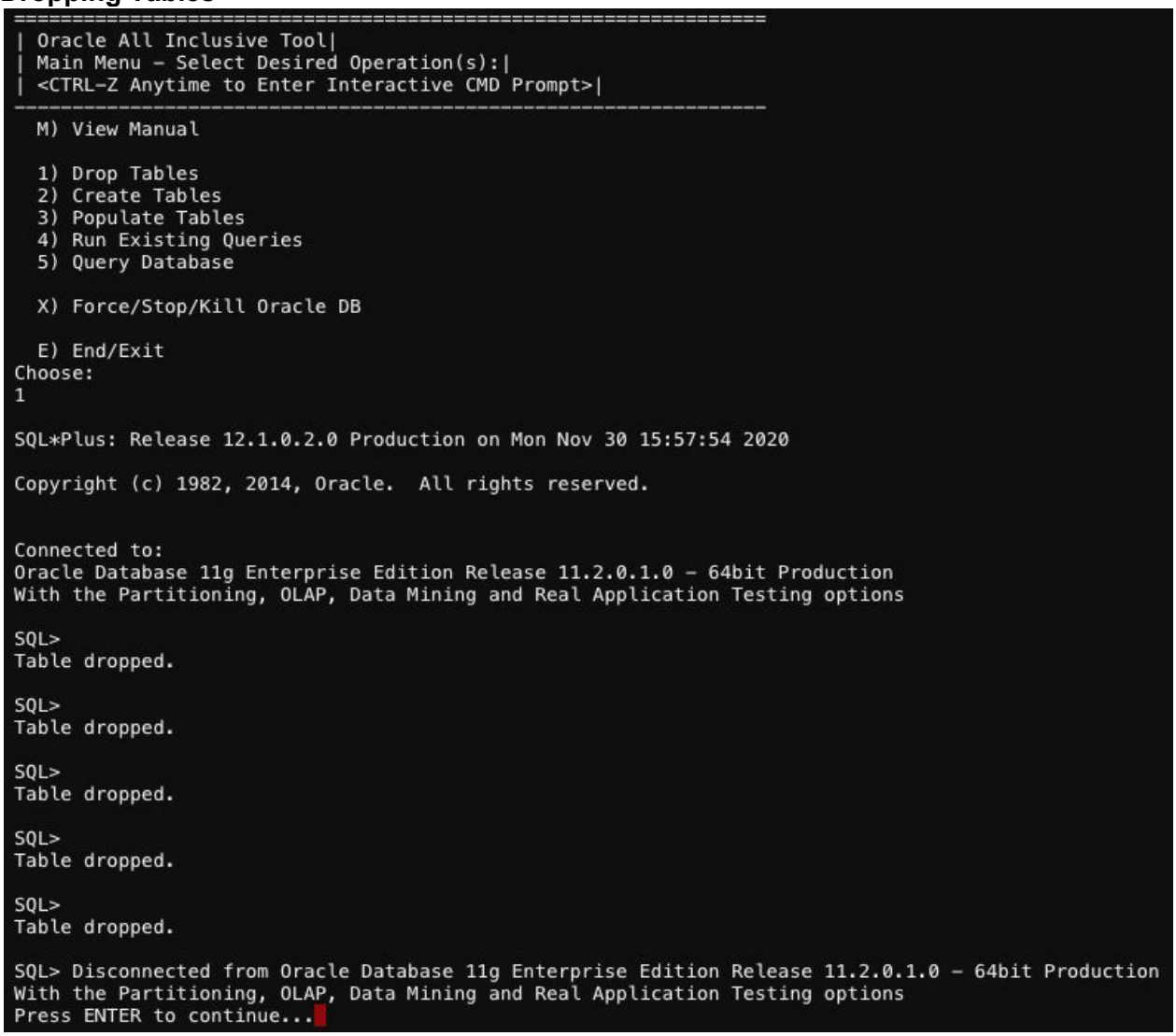


Final Relational Model

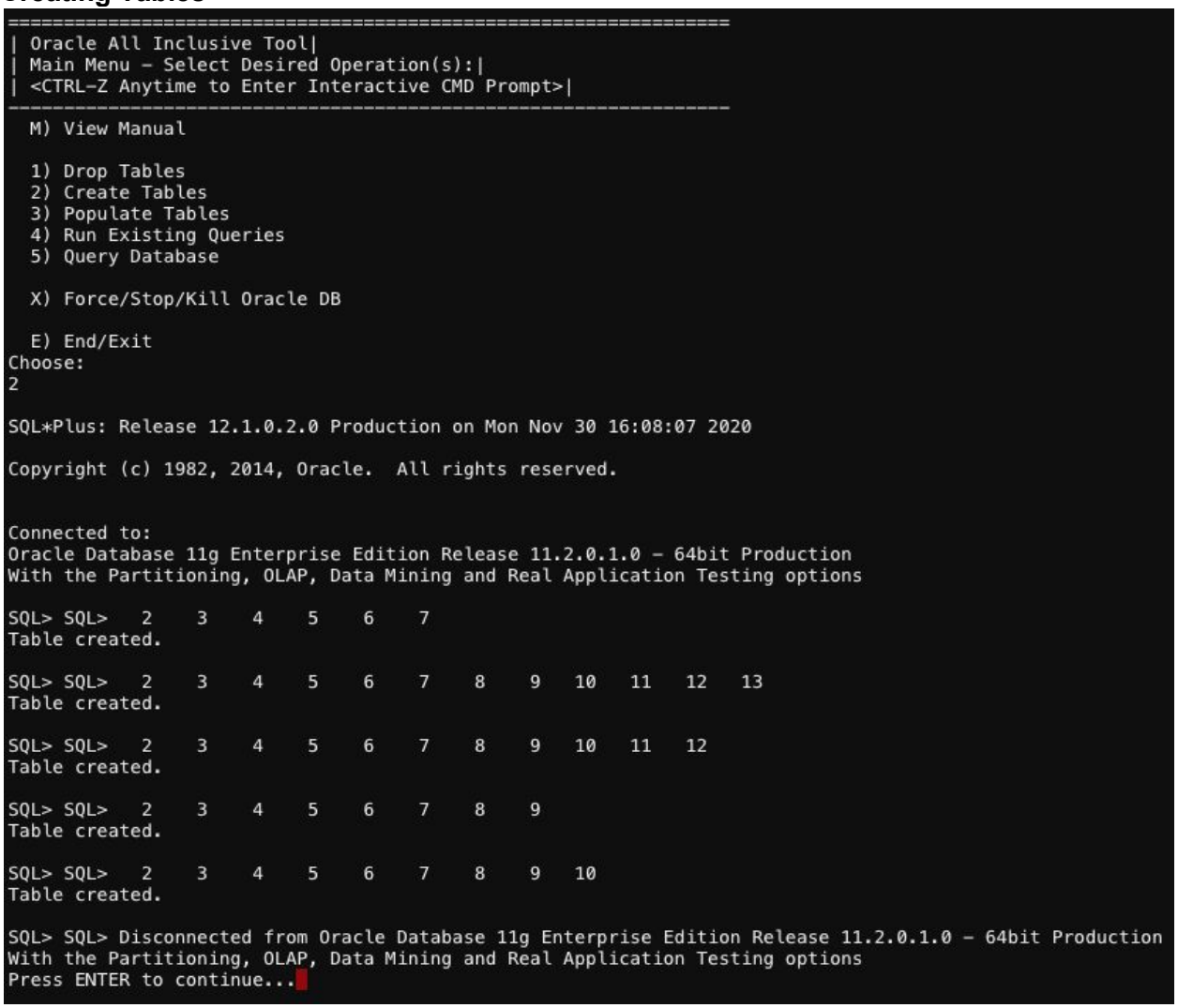


User Interface: with Normalized Database

Dropping Tables



Creating Tables



Populating Tables

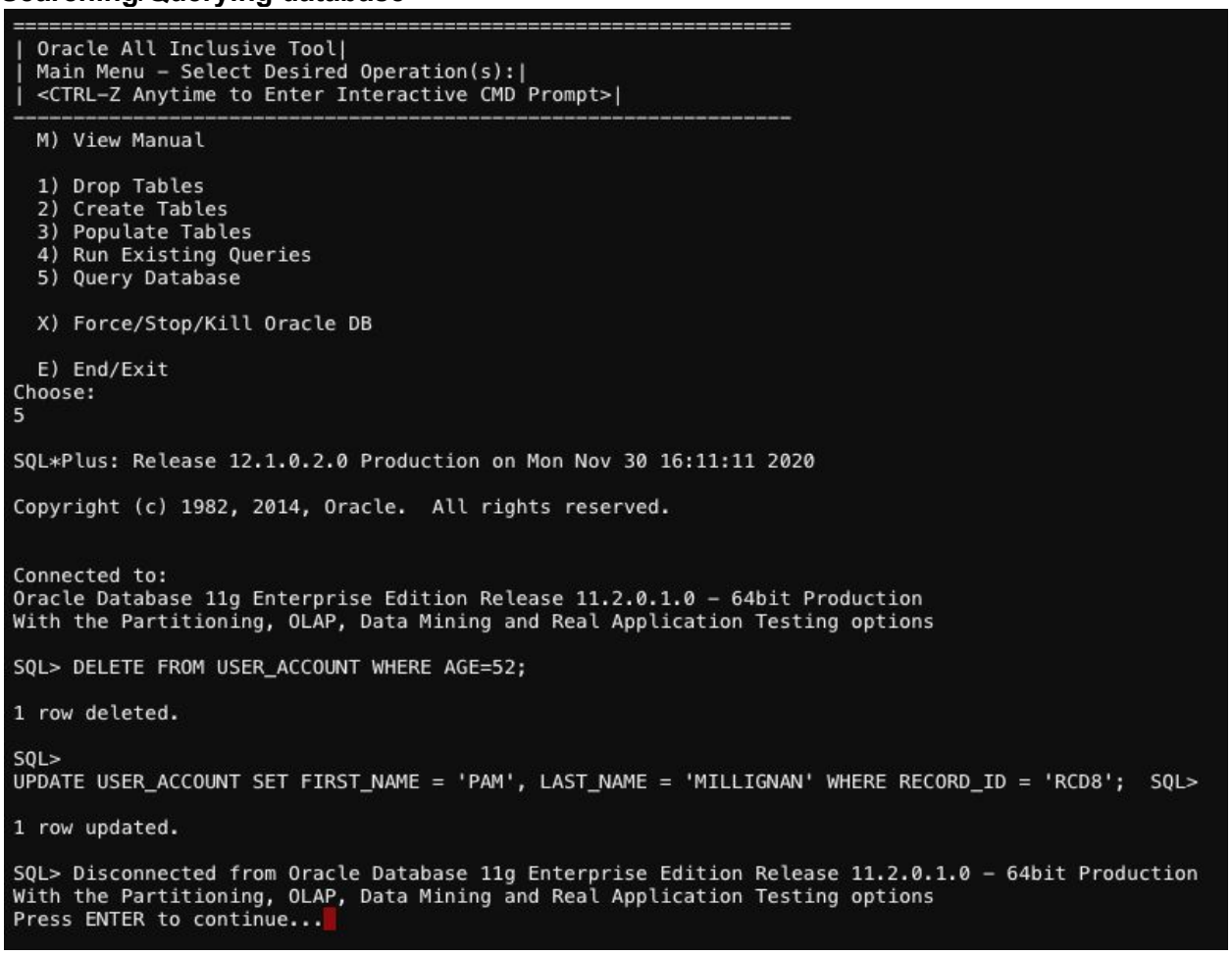
```
================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
----------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Run Existing Queries
  5) Query Database

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
3

SQL*Plus: Release 12.1.0.2.0 Production on Mon Nov 30 16:08:37 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.
```

Running pre-existing queries

```
================================================================
| Oracle All Inclusive Tool|
| Main Menu - Select Desired Operation(s):|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>|
----------------------------------------------------------------
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Run Existing Queries
  5) Query Database

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
4

SQL*Plus: Release 12.1.0.2.0 Production on Mon Nov 30 16:09:06 2020

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL>   2
SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                                AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------
SH1        ROBERT BENJAMIN
SMITH                                                     18 MALE
WHITE                PISTOL

SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
LAST_NAME                                                AGE GENDER
-------------------------------------------------- ---------- ----------
RACE                 WEAPON_TYPE
-------------------- --------------------------------------------------
OCCUPATION
--------------------------------------------------
STATE_NAME
--------------------------------------------------

ARIZONA


SHOOTER_ID FIRST_NAME
---------- --------------------------------------------------
```

```
SQL> SQL>   2    3    4
View created.

SQL> SQL>
FIRST_NAME
--------------------------------------------------
LAST_NAME
--------------------------------------------------
JAMIE
WILLINGER

JOHNATHAN
MILLS

HOLDEN
COOPER


SQL> SQL>
View dropped.

SQL> SQL>   2    3    4
View created.

SQL> SQL>
MONTH_OF_S YEAR_OF_SHOOTING
---------- ----------------
NOVEMBER               2018
OCTOBER                2018
MARCH                  2019
NOVEMBER               2018
DECEMBER               2017

SQL> SQL>
View dropped.

SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
Press ENTER to continue...
```

Searching/Querying database



Relational Algebra

Query 1:

SQL: SELECT CAST(AVG(shooter.age) as int) as avg_age
FROM SHOOTER;

Relational: $^{F}_{\text{average age}}$(Shooter)

Query 2:

SQL: SELECT COUNT(shooter_id), race
FROM shooter
GROUP BY race;

Relational: $\pi_{race}(race^{F}_{count\ shooter.id}(Shooter))$

Query 3:

SQL: SELECT record.record_id, victim.state_name
FROM record INNER JOIN victim
ON record.victim_id = victim.victim_id;

Relational: $\pi_{\text{record.record\_id, victim.state\_name}}(\text{record} \bowtie \text{victim})$

Query 4:

SQL:
```
SELECT COUNT(type_of_shooting), type_of_shooting
    FROM record
    GROUP BY type_of_shooting
    ORDER BY COUNT(type_of_shooting) DESC;
```

Relational: $_{\text{type\_of\_shooting}}\text{G}_{\text{count(type\_of\_shooting)}}$ (record)

Query 5:

SQL:
```
SELECT *
    FROM US_STATE
    WHERE POLICE_BUDGET_PER_CAPITA BETWEEN 300 AND 500;
```

Relational: $\sigma_{\text{police\_budget\_per\_capita < 300 ^ police\_budget\_per\_capita > 500}}$ (us_state)

Query 6:

SQL:
```
SELECT us_state.state_name, us_state.registered_weapons,
    us_state.police_budget_per_capita,
    us_state.black_population_by_percent, shooter.race
    FROM us_state RIGHT JOIN shooter
    ON shooter.state_name = us_state.state_name
    ORDER BY(registered_weapons) DESC;
```

Relational: $\pi_{\text{us\_state.state\_name, us\_state.registered\_weapons, us\_state.police\_budget\_per\_capita,us\_state.black\_population\_by\_percent, shooter.race}}(\text{us\_state} \bowtie \text{shooter})\tau(\text{registered\_weapons})$

Query 7:

SQL:
```
CREATE VIEW male_accounts AS
    SELECT FIRST_NAME, LAST_NAME
    FROM USER_ACCOUNT
    WHERE GENDER = 'MALE';

    SELECT * FROM male_accounts;
```

Relational: $\pi_{\text{FIRST\_NAME, LAST\_NAME}}(\sigma_{\text{gender='male'}}(\text{user\_account}))$

Query 8:

SQL:
```
CREATE VIEW teenage_victims AS
    SELECT FIRST_NAME, LAST_NAME
    FROM VICTIM
    WHERE AGE BETWEEN 13 AND 19;

    SELECT * FROM teenage_victims;
```

Relational: $\pi_{\text{FIRST\_NAME, LAST\_NAME}}(\sigma_{\text{age>13^age<19}}(\text{victim}))$

Query 9:

SQL: CREATE VIEW school_shooting_dates AS
SELECT MONTH_OF_SHOOTING, YEAR_OF_SHOOTING
FROM RECORD
WHERE TYPE_OF_SHOOTING = 'SCHOOL';

SELECT * FROM school_shooting_dates;

Relational: $\pi_{\text{MONTH\_OF\_SHOOTING, YEAR\_OF\_SHOOTING}}(\sigma_{\text{type\_of\_shooting='school'}}(\text{record}))$

Design experience

The normalization of 2NF relations to 3NF involves the removal of transitive dependencies, therefore our database is in 3NF because at least one of the following conditions holds for every non-trivial dependency $X \rightarrow Y$ in that either X is a super key in the relationship or Y is a prime attribute. The process of doing this was quite interesting because you get a deeper understanding of what each attribute in each relation truly depends on and gives you greater insight into how anomalies are created.