

Innocent Deaths caused by police ın amerıca

Güray Hatip, Emre Doğu, Cuma Bolat



In this project we wanted to shade a light on a very important topic that is ignored most of the times by the USA government. We have most of the data about the citizens who have been wrongfully killed without justification. In some cases, there wasn’t a need for deadly force but regardless of the situation some lives were taken away. We have chosen this data to illuminate this topic.

(2)CHANGES REGARDING TO THE DATABASE

* When we first got out data, it had many problems. There were duplicate entries about the victims. There were rows that were mismatched. There were characters that collided with our way of importing the data. We have solved all these issues. For this we have used Excel and some Java coding.
* The first dataset that we got from Kaggle was made regarding to the first normal form(1NF). After getting our specs we have implemented this data as third normal form (3NF) according to the specs.
* We had 5 tables in our first phase. Victim, Police, Links, Incident and Location. Each except Links table had their own primary keys. Currently we have 5 tables in RDF form. Victim, StringProperties, StringValues, NumericProperties and NumericValues.
* After our meeting we have removed all the null values from stringValues and numericValues tables.

(3)DESCRIPTION OF LOADING THE DATA AND THE CODES WRITTEN

1. After we downloaded the data from Kaggle, first we checked if the data had any duplications and mistakes like mismatches. For this we used Excel.
2. After we got our specs, we had to change values that were in the same row to the same column for our value tables. For this we have used a java code reading the information from a txt file, and write this data to another txt file.
3. We have put the results of these codes to xmls files and changed these xmls’s to csv files by using a site called [cloudconvert.com](https://cloudconvert.com/).
4. Since importing from a diagram take too long, after we forward engineered our data model to the database we have written SQL queries to import the data to the database.

Here is the Kaggle link: <https://www.kaggle.com/kannan1314/innocent-deaths-caused-by-police-all-time>

Example of our java code:

public static void main(String[] args) throws IOException {  
 Scanner reader = new Scanner(new FileReader("ervalues.txt"));  
 FileWriter writer = new FileWriter("valuesNew.txt");  
 boolean eof = true;  
  
 while (eof){  
 try {  
 writer.write(reader.next());  
 writer.write("\n");  
  
 } catch (NoSuchElementException | NullPointerException e) {  
 eof = false;  
 }  
 }  
}

Example of our SQL query:

LOAD DATA infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ER diagram Victim.csv'

INTO TABLE victim

FIELDS TERMINATED BY ','

IGNORE 1 LINES

(victim\_id, first\_name, @middle\_name, last\_Name, @age, @gender, @race)

SET middle\_name = NULLIF(@middle\_name, ''), age = NULLIF(@age, ''), gender = NULLIF(@gender, ''), race = REPLACE(@race,'\r','');

SHOW VARIABLES LIKE 'secure\_file\_priv';

(5)EXPLAINING VİEWS AND STORED PROCEDURES

**Views:** We have made 4 views named, department\_killcount, detailed\_victim\_numeric\_information, detailed\_victim\_string\_information and victims\_below\_twenty.

**Department\_killcount:** This view returns the victim counts of the departments involved in their death.

**Detailed\_victim\_numeric\_information:** Since we re using RDF format it is harder to find which id a value has. This view helps the user to see the numeric information details on the victim’s values and learn the id’s of these values.

**Detailed\_victim\_string\_information:** Same as the numeric information this view returns the string information of the victim and their id’s.

**Victims\_below\_twenty:** This view returns the information of victims who were below 20 years old when they got killed.

**Stored Procedures:** We have made 5 stored procedures named, description\_finder, finding\_specific\_victims\_before\_date, selecting\_all, victim\_count\_in\_county, `victims\_living\_in\_specific\_state`

**Description\_finder:** This procedure uses 2 variables victim\_to\_find as IN and it needs a victim\_id as INT, and victims\_description INOUT this variable will change the value passed onto this procedure and make it the death description of the victim passed onto the procedure.

**Finding\_specific\_victims\_before\_date:** This procedure needs 4 variables sent to it all of them are the type IN. These variables are gender, race, state and death date of victims that is needed to be found. The procedure will return all the victims fitting to the descriptions and that died before the entry date that is passed onto it.

**Selecting\_all:** This procedure is like the views regarding to finding information about the victims it doesn’t need and parameters passed onto it. The procedure will return all the values including numeric values, string values, their id’s and the victim’s details.

**Victim\_count\_in\_county:** This procedure will return the death numbers in a specific county. It needs a county name as IN variable and it will return victim numbers as OUT.

**Victims\_living\_in\_specific\_state:** This procedure will return all the victims living in a specific state. It only has one parameter state name as IN. After receiving the state name user will see names and last names of the people died in this state.

**(6)Here is an example on one of our stored procedures.**

**CALL finding\_specific\_victims\_before\_date('female', 'European-American/White', 'CA', 2007);**

'75', 'Jennifer', 'Lowe', 'Female', 'European-American/White', 'CA', '2000-02-19'

'96', 'Audra', 'Hynes', 'Female', 'European-American/White', 'CA', '2000-03-06'

'985', 'Glenda', 'Reymer', 'Female', 'European-American/White', 'CA', '2001-06-22'

'1163', 'Rebecca', 'Aramburo', 'Female', 'European-American/White', 'CA', '2001-09-21'

'1266', 'Katrina', 'Sarkissian', 'Female', 'European-American/White', 'CA', '2001-11-18'

'1278', 'Karen', 'Gordin', 'Female', 'European-American/White', 'CA', '2001-11-27'

'1380', 'Kristina', 'Priano', 'Female', 'European-American/White', 'CA', '2002-01-22'

'1601', 'Shanna', 'Dreiling', 'Female', 'European-American/White', 'CA', '2002-05-03'

'2390', 'Charla', 'Schwerstein', 'Female', 'European-American/White', 'CA', '2003-04-23'

'2611', 'Teale', 'Sisson', 'Female', 'European-American/White', 'CA', '2003-08-08'

'2701', 'Ellen', 'Derenge', 'Female', 'European-American/White', 'CA', '2003-09-18'

'2752', 'Jennifer', 'LeBlanc', 'Female', 'European-American/White', 'CA', '2003-10-13'

'3087', 'Mariline', 'Sacks', 'Female', 'European-American/White', 'CA', '2004-03-04'

'3428', 'Linda', 'Soper', 'Female', 'European-American/White', 'CA', '2004-08-10'

'3682', 'Linda', 'Wood', 'Female', 'European-American/White', 'CA', '2004-12-16'

'3759', 'Katherine', 'Dimercurio', 'Female', 'European-American/White', 'CA', '2005-01-12'

'4024', 'Rayne', 'McGowan', 'Female', 'European-American/White', 'CA', '2005-05-10'

'4025', 'Paige', 'McGowan', 'Female', 'European-American/White', 'CA', '2005-05-10'

'4027', 'Angela', 'McGowan', 'Female', 'European-American/White', 'CA', '2005-05-10'

'4028', 'Karen', 'McGowan', 'Female', 'European-American/White', 'CA', '2005-05-10'

'4319', 'Alexander', 'Maschhaupt', 'Female', 'European-American/White', 'CA', '2005-08-29'

'4800', 'Karen', 'Eklund', 'Female', 'European-American/White', 'CA', '2006-03-23'

'4847', 'Cheri', 'Moore', 'Female', 'European-American/White', 'CA', '2006-04-14'

'4882', 'Lynn', 'Thornton', 'Female', 'European-American/White', 'CA', '2006-04-29'

'5023', 'Cindy', 'Conolly', 'Female', 'European-American/White', 'CA', '2006-06-12'

'5157', 'Ashley', 'MacDonald', 'Female', 'European-American/White', 'CA', '2006-08-04'

'5375', 'Kathryn', 'Langley', 'Female', 'European-American/White', 'CA', '2006-10-26'

'5427', 'Karissa', 'Workman', 'Female', 'European-American/White', 'CA', '2006-11-13'

'5518', 'Kristin', 'Bruce', 'Female', 'European-American/White', 'CA', '2006-12-14'

(7)ANALYSIS OF THE SYSTEM

In current situation of the system it is harder for the user to find details on victims since they would need to do couple of joins in the dataset. At the same time this joining procedure is costly so it slows down the database. As suggestion these issues can be reduced by using indexes and lots of views.

**(8)Here is our Data Definition Language(DDL) of our dataset.**

-- -----------------------------------------------------

-- Table `innocentDeathsFinalTable`.`Victim`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `innocentDeathsFinalTable`.`Victim` (

`victim\_id` INT NOT NULL,

`first\_name` VARCHAR(45) NOT NULL,

`middle\_name` VARCHAR(45) NULL,

`last\_name` VARCHAR(45) NOT NULL,

`age` INT NULL,

`gender` VARCHAR(11) NULL,

`race` VARCHAR(45) NOT NULL,

PRIMARY KEY (`victim\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `innocentDeathsFinalTable`.`NumericProperties`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `innocentDeathsFinalTable`.`NumericProperties` (

`numeric\_id` INT NOT NULL,

`numeric\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`numeric\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `innocentDeathsFinalTable`.`StringProperties`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `innocentDeathsFinalTable`.`StringProperties` (

`string\_id` INT NOT NULL,

`string\_type` VARCHAR(45) NOT NULL,

PRIMARY KEY (`string\_id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `innocentDeathsFinalTable`.`StringValues`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `innocentDeathsFinalTable`.`StringValues` (

`victim\_id` INT NOT NULL AUTO\_INCREMENT,

`string\_id` INT NOT NULL,

`value\_string` TEXT NULL,

PRIMARY KEY (`victim\_id`, `string\_id`),

INDEX `fk\_VictimToPropertyToStringValue\_StringProperties1\_idx` (`string\_id` ASC) VISIBLE,

INDEX `fk\_VictimToPropertyToStringValue\_Victim1\_idx` (`victim\_id` ASC) VISIBLE,

CONSTRAINT `fk\_VictimToPropertyToStringValue\_StringProperties1`

FOREIGN KEY (`string\_id`)

REFERENCES `innocentDeathsFinalTable`.`StringProperties` (`string\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_VictimToPropertyToStringValue\_Victim1`

FOREIGN KEY (`victim\_id`)

REFERENCES `innocentDeathsFinalTable`.`Victim` (`victim\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `innocentDeathsFinalTable`.`NumericValues`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `innocentDeathsFinalTable`.`NumericValues` (

`victim\_id` INT NOT NULL AUTO\_INCREMENT,

`numeric\_id` INT NOT NULL,

`value\_numeric` VARCHAR(20) NULL,

INDEX `fk\_VictimToPropertyToNumericValue\_NumericProperties1\_idx` (`numeric\_id` ASC) VISIBLE,

INDEX `fk\_VictimToPropertyToNumericValue\_Victim1\_idx` (`victim\_id` ASC) VISIBLE,

PRIMARY KEY (`victim\_id`, `numeric\_id`),

CONSTRAINT `fk\_VictimToPropertyToNumericValue\_NumericProperties1`

FOREIGN KEY (`numeric\_id`)

REFERENCES `innocentDeathsFinalTable`.`NumericProperties` (`numeric\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_VictimToPropertyToNumericValue\_Victim1`

FOREIGN KEY (`victim\_id`)

REFERENCES `innocentDeathsFinalTable`.`Victim` (`victim\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

**(9) SQL codes that are used for the dataset.**

**View codes:**

create view department\_killcount as

select count(v.victim\_id) as killcount, value\_string as department\_name from victim v join stringvalues sv on (v.victim\_id = sv.victim\_id)

where string\_id = 7

group by value\_string

order by killcount desc;

create view detailed\_victim\_numeric\_information as

select v.victim\_id, v.first\_name, v.middle\_name, v.last\_name, v.age, v.gender, v.race, nv.value\_numeric, np.numeric\_type, nv.numeric\_id

from victim v join numericvalues nv on (nv.victim\_id = v.victim\_id)

join numericproperties np on (np.numeric\_id = nv.numeric\_id)

ORDER BY v.victim\_id;

create view detailed\_victim\_string\_information as

select v.victim\_id, v.first\_name, v.middle\_name, v.last\_name, v.age, v.gender, v.race, sv.value\_string, sp.string\_type, sv.string\_id

from victim v join stringvalues sv on (sv.victim\_id = v.victim\_id) join stringproperties sp on (sp.string\_id = sv.string\_id)

ORDER BY v.victim\_id;

create view victims\_below\_twenty as

select \* from victim

where age < 20;

**Stored procedure codes:**

CREATE DEFINER=`root`@`localhost` PROCEDURE `description\_finder`(IN victim\_id\_to\_find INT, INOUT victims\_description TEXT)

BEGIN

SELECT sv.value\_string INTO victims\_description

FROM victim v join stringvalues sv on (v.victim\_id = sv.victim\_id)

where sv.string\_id = 14 and v.victim\_id = victim\_id\_to\_find;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `victim\_count\_in\_county`(IN county\_name varchar(50), OUT victims INT)

BEGIN

select COUNT(v.victim\_id) as victim\_count INTO victims from victim v join stringvalues sv on (v.victim\_id = sv.victim\_id)

where sv.value\_string = county\_name;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `finding\_specific\_victims\_before\_date`(IN gender TEXT, IN race TEXT, IN state varchar(2), IN deathdate varchar(10))

BEGIN

select v.victim\_id, v.first\_name, v.last\_name, v.gender, v.race, sv.value\_string as state, nv.value\_numeric as death\_date

from victim v join stringvalues sv on (sv.victim\_id = v.victim\_id)

join stringproperties sp on (sp.string\_id = sv.string\_id)

join numericvalues nv on (nv.victim\_id = v.victim\_id)

join numericproperties np on (np.numeric\_id = nv.numeric\_id)

where v.gender = gender and v.race = race

and sv.value\_string = state and nv.value\_numeric < deathdate and nv.numeric\_id = 1;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `victims\_living\_in\_specific\_state`(IN state\_name varchar(2))

BEGIN

select v.victim\_id, v.first\_name, v.last\_name, sv.value\_string from victim v join stringvalues sv on (v.victim\_id = sv.victim\_id)

where sv.value\_string = state\_name;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `selecting\_all`()

BEGIN

select \* from victim v join stringvalues sv on (sv.victim\_id = v.victim\_id)

join stringproperties sp on (sp.string\_id = sv.string\_id)

join numericvalues nv on (nv.victim\_id = v.victim\_id)

join numericproperties np on (np.numeric\_id = nv.numeric\_id)

order by v.victim\_id;

END

**And here is an example table import. We have code like this for each table.**

LOAD DATA infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ER diagram StringValues.csv'

INTO TABLE stringvalues

FIELDS TERMINATED BY ','

IGNORE 1 LINES

(victim\_id, string\_id, @value\_string)

SET value\_string = NULLIF(@value\_string, '');

show variables like 'secure\_file\_priv';