Tuesday, 4/21/2020
CSC 436, Spring 2020
Submitted to: Dr. Hendawi
Submitted by: Matt Pitman,
Troy Durand, Jane Trapala
& Hassan Bhatti

Final Project Document

(Bloody Mary's Blood Bank)

1) Project Idea

With an increasing amount of blood donations, it is only more efficient to tally its associated data with the application of database systems. In this particular scenario of Bloody Mary's Blood Bank (BMBB), a database system was created for storing subjective information pertaining to a donor, receiver, blood bank, and the provisional staff. The 6 tables that were implemented into our database are:

1. Blood Table: Contains information on each of the blood bags that were obtained.
   i. Bloodbag_Number (Primary Key)
   ii. Blood_Type
   iii. Blood_Amount
   iv. Haemoglobin_Content
   v. Double_Red
   vi. Donor_ID (Foreign Key)

2. Blood Bank Table: Stores information about each of the blood banks
   i. BB_ID (Primary Key)
   ii. D_Name
   iii. Address
   iv. City
   v. State
   vi. Phone
   vii. Admin_Name

3. Blood Drive Table: Information on each of the blood drives held by the blood banks
   i. Bdrive_ID (Primary Key)
   ii. Name

iii. Address
iv. City
v. State
vi. Phone
vii. Bank_ID (Foreign Key)

4. Staff Table: The personal information of the different employees of the blood banks
   i. Staff_ID (Primary Key)
   ii. Name
   iii. Address
   iv. Phone
   v. Shift
   vi. Gender
   vii. DOB
   viii. Bank_ID (Foreign Key)

5. Donor Table: Personal information about all of the donors who gave blood
   i. Donor_ID (Primary Key)
   ii. Name
   iii. Address
   iv. Phone_Number
   v. Medical_Condition
   vi. Gender
   vii. DOB
   viii. Blood_Type
   ix. Bank_ID (Foreign Key)

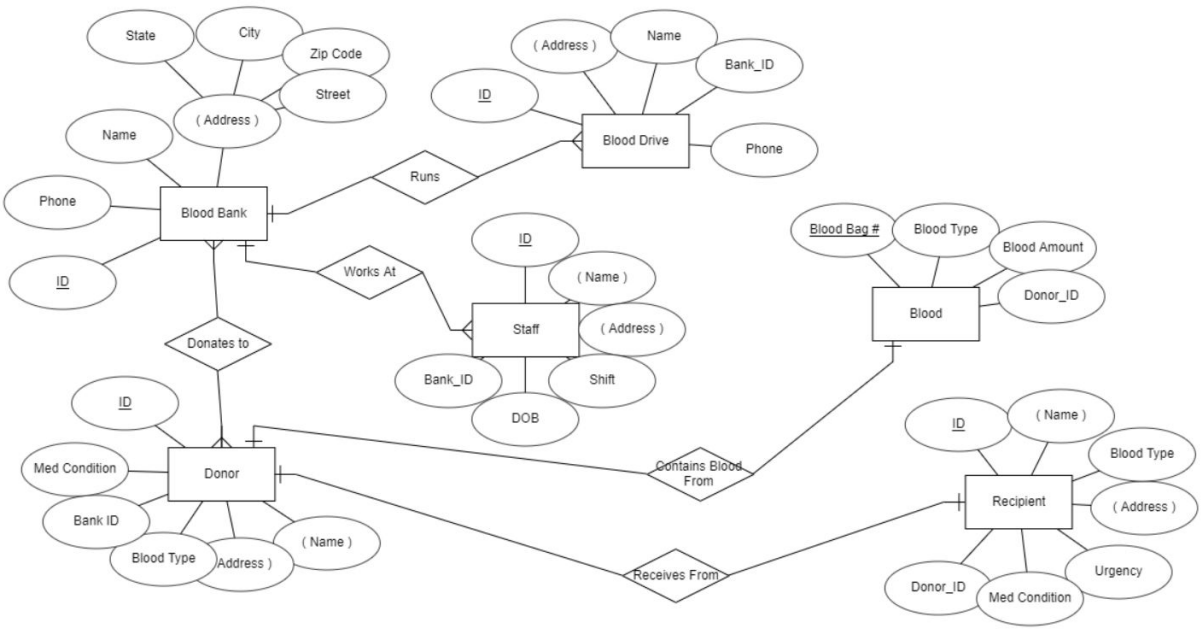
6. Recipient Table: Holds the personal information of all the recipients from the donors
   i. Recip_ID (Primary Key)
   ii. Name
   iii. Address
   iv. Phone_Number
   v. Medical_Condition
   vi. Gender
   vii. DOB
   viii. Urgency_Status
   ix. Blood_Type
   x. Donor_ID (Foreign Key)

In conjunction to the creation of data tables for the aforementioned subjects, certain functions were also implemented: insertion of new data, editing of pre-existing data, searching the data for a desired outcome, and the deletion of irrelevant data. By implementing such dynamic functions, one may use the BMBB database for a diverse amount of purposes. By using MySQL, the skeletal structure of the database program was laid out. Whereas, Python was used for objectively programming its core. In order to cater to an average user in a convenient manner, a graphical user interface (GUI) was developed. For further information pertaining to the implementational development of the project idea's conception, please see the following sections (2 to 8) below.

2) User requirements (Basic operations and questions)

User requirements for our database are MySQL, access to our schema or a copy of our sql source code, and information required for each table. In order to insert data into a specific table the user must know what the attributes are for that table and what the data type is for each attribute. For instance if inserting into the blood table you will need to know the blood bag Id number, the donor Id, the blood type (which should correspond to the donor), blood_amount, Haemeoglobin content, and finally if it is double red or not. Another important thing to keep in mind as the user is due to the foreign key restraints you have to manipulate the tables in a certain order. You can not drop the blood bank table if you have not previously dropped the child tables (staff, blood drive, and donor). You also will not be able to enter new values into child tables if the foreign key attribute does not match one from the parent table. Users will also not be able to repeat attributes that have been set as unique. The final requirement is if the user is interested in updating or deleting from a table, the user must know the information contained in the tables, in order to know they are validly searching

3) ERD with Database tables, relationships, constraints etc



4) Data sources, if exists

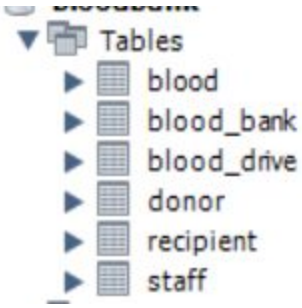
Due to confidentiality we were not able to access actual data from any blood banks, or blood drives. Therefore we were left with randomly producing the data. To accomplish this we used a website called http://filldb.info/. This website was very useful for us as it allowed us to produce as much data as we wanted and it held true to foriegn constraints. Since you upload all of your sql code before beginning it allowed you to implement data that held true throughout the tables. The only issue was naming the blood banks and blood drives, as the best they offered was a random word. A random word is not all that realistic for the title of a blood bank, however for these circumstances we felt that it was good enough.

5) SQL queries and sample results for each, (cover different types of queries DDL, DML, DCL)

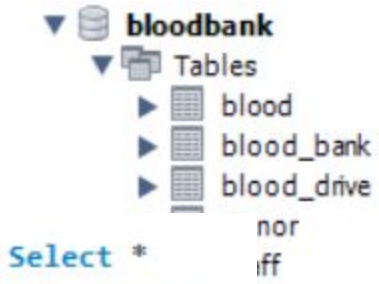
➔ DDL:

```
CREATE TABLE blood_bank (
  bb_ID int(9) NOT NULL primary key,
  d_name varchar(25) NOT NULL,
  address varchar(20),
  city varchar(20),
  state varchar(15),
  phone varchar(14),
  admin_Name varchar(25),
  unique(bb_ID)
);
```

```
CREATE TABLE blood_drive (
  bdrive_ID int(9) NOT NULL primary key,
  Name varchar(25) NOT NULL,
  Address varchar(20),
  City varchar(20),
  State varchar(15),
  Phone varchar(14),
  bank_ID int(9),
  unique(bdrive_ID),
  foreign Key (bank_ID)
    references blood_bank(bb_ID)
    on update cascade on delete cascade
);
```




```
drop table recipient;
```




```
Select *
From Staff;
```

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

Staff_ID	Name	Address	Phone	Shift	Gender	DOB	Bank_ID
0	Reina Waelchi	546 Reynolds Gateway	406-595-7534x3	night	m	2014-01-23	4
1	Hazle Huel I	35874 Harley Knoll A	1-390-749-3716	night	m	1974-04-17	394927219
2	Mr. Maxime Treutel PhD	222 Tillman Park Apt	171-087-6997x8	night	m	2013-12-12	4
3	Dr. Gracie Schulist DVM	94400 Carter Crossin	03260130223	morning	f	2003-03-31	762722
4	Rosalyn Medhurst	77583 Lisandro Ridge	1-558-985-1888	morning	m	2017-11-15	8
5	Preston Spinka	38925 Brakus Court A	768.899.1861x9	night	m	2008-11-19	7953
6	Evert Bogisich	3892 Lula Motorway C	+86(5)76064379	night	m	1970-02-28	0
7	Dr. Reanna Metz	19799 Nils Inlet Lak	011.306.3126	morning	f	1999-07-07	309675013
8	Mark Glover	4465 Okuneva Extensi	+31(9)17015813	morning	m	2011-05-21	9033940
17	Beulah Keebler	2937 Craig Camp Sout	(620)782-0263	morning	f	2004-05-05	9187
26	Sandra Schaefer DVM	183 Ronny Pine Apt.	286.595.3527	night	f	1989-06-12	76959982
28	Jay Hilpert	0235 Hudson Harbors	(081)277-6615x	night	m	2016-07-29	394927219
31	Miss Therese McGlynn MD	619 Caleigh Pine Sui	938-087-5952x3	night	f	1981-04-29	309675013
41	Joanne Quitzon II	160 Claud Haven Suit	792-514-9512x5	morning	m	1997-11-26	7953
53	Miss Karlee Lockman MD	8387 Leo Squares Sui	183.494.6045	night	m	1987-03-15	87651216

➔ DML:

```
Update staff
Set Phone = '955-712-4163'
Where Staff_Id = 183;
```

Staff_ID	Name	Address	Phone	Shift	Gender	DOB	Bank_ID
183	Toni Jakubowski	878 Durgan Village A	955-712-4163	morning	f	1977-09-12	762722

```
Delete From Staff
Where Staff_ID = 1;
```

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

Staff_ID	Name	Address	Phone	Shift	Gender	DOB	Bank_ID
0	Reina Waelchi	546 Reynolds Gateway	406-595-7534x3	night	m	2014-01-23	4
2	Mr. Maxime Treutel PhD	222 Tillman Park Apt	171-087-6997x8	night	m	2013-12-12	4
3	Dr. Gracie Schulist DVM	94400 Carter Crossin	03260130223	morning	f	2003-03-31	762722
4	Rosalyn Medhurst	77583 Lisandro Ridge	1-558-985-1888	morning	m	2017-11-15	8
5	Preston Spinka	38925 Brakus Court A	768.899.1861x9	night	m	2008-11-19	7953
6	Evert Bogisich	3892 Lula Motorway C	+86(5)76064379	night	m	1970-02-28	0

```
INSERT INTO `recipient` (`recip_ID`, `Name`, `Address`, `Phone_Number`, `Medical_Condition`, `Gender`, `DOB`, `Urgency_Status`, `Blood_
VALUES (0, 'Liza Bogisich', '6615 Wolff Shore Sui', '(946)049-1190x', ' Stomach Disease', 'm', '1984-11-05', 10, ' B-', 8);
INSERT INTO `recipient` (`recip_ID`, `Name`, `Address`, `Phone_Number`, `Medical_Condition`, `Gender`, `DOB`, `Urgency_Status`, `Blood_
VALUES (1, 'Darien Rippin Sr.', '02467 Mertz Light\nEv', '(980)843-5381', ' Blood Disease', 'm', '1982-09-25', 3, 'A+', 4381);
INSERT INTO `recipient` (`recip_ID`, `Name`, `Address`, `Phone_Number`, `Medical_Condition`, `Gender`, `DOB`, `Urgency_Status`, `Blood_
VALUES (2, 'Piper Mills', '30122 Tillman Route\n', '632-149-9581', ' Ebola Virus', 'm', '2006-02-17', 8, 'AB-', 40);
INSERT INTO `recipient` (`recip_ID`, `Name`, `Address`, `Phone_Number`, `Medical_Condition`, `Gender`, `DOB`, `Urgency_Status`, `Blood_
VALUES (3, 'Fletcher Kulas II', '82915 Edgar Groves S', '(714)973-7646x', ' Stomach Disease', 'f', '1977-07-31', 2, 'AB-', 5);
INSERT INTO `recipient` (`recip_ID`, `Name`, `Address`, `Phone_Number`, `Medical_Condition`, `Gender`, `DOB`, `Urgency_Status`, `Blood_
VALUES (4, 'Kurtis Terry', '0048 Klein Alley Apt', '1-215-780-0383', ' Corona Virus', 'm', '1973-11-07', 6, 'O+', 46);
```

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

recip_ID	Name	Address	Phone_Number	Medical_Condition	Gender	DOB	Urgency_Status	Blood_Type	donor_ID
0	Liza Bogisich	6615 Wolff Shore Sui	(946)049-1190x	Stomach Disease	m	1984-11-05	10	B-	8
1	Darien Rippin Sr.	02467 Mertz Light Ev	(980)843-5381	Blood Disease	m	1982-09-25	3	A+	4381
2	Piper Mills	30122 Tillman Route	632-149-9581	Ebola Virus	m	2006-02-17	8	AB-	40
3	Fletcher Kulas II	82915 Edgar Groves S	(714)973-7646x	Stomach Disease	f	1977-07-31	2	AB-	5
4	Kurtis Terry	0048 Klein Alley Apt	1-215-780-0383	Corona Virus	m	1973-11-07	6	O+	46
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Views and Joins:

```
Create View BBB as
Select bb_id, d_name
from blood_bank
where bb_id >= 100002;
```

bb_id	d_name
359532	et
762722	dolorem
9033940	qui
54932289	quasi
76959982	molestiae
87651216	enim
309675013	ut
394927219	corporis
866067274	maxime

```
Select Staff.Staff_ID, blood_bank.bb_id
From Staff
LEFT OUTER JOIN blood_bank
ON staff.Bank_ID = blood_bank.bb_id;
```

Staff_ID	bb_id
1017	4
1042	4
1084	4
1031	13
1055	13

```
Select donor.donor_id, recipient.recip_ID, recipient.blood_type
From Donor
Inner Join Recipient
On donor.donor_id = recipient.donor_ID;
```

donor_id	recip_ID	blood_type
132	17	O-
700	50	O+
432	57	B+
454	61	A+
648	80	B+
197	133	AB-
671	157	A-
341	190	B-
519	240	A+
512	251	B-
513	252	AB-
643	304	AB-
778	341	O+
692	383	B-
579	386	AB-
488	453	O+
547	467	AB-

```
Select *
From Donor Natural Join Blood;
```

Donor_ID	Blood_Type	Name	Address	Phone_Number	Medical_Condition	Gender
132	A+	Tillman Lang	3331 Lebsack Park Ke	1-017-658-1589	Lung Disease	m
163	O+	Miss Fae Kuhlman DDS	633 Braun Neck New J	676-600-3355x1	Blood Pressure	m
166	B+	Deangelo Corkery	69050 Schimmel Missi	1-326-041-0102	NULL	f
187	O+	Juston Connelly	559 Hoppe Ramp Apt.	961.563.5269	Asthma	f
197	O+	Prof. Gonzalo Ferry	5821 Evan Burg North	+57(2)26943639	Diabetes	f
218	O-	Daniela Rohan	140 Macie Station Po	(048)991-1092	Blood Pressure	m
302	AB-	Idella Mayer	797 Jacobson Lights	(759)017-6835	Lung Disease	f
513	A+	Mr. Salvador Hudson	869 Pfannerstill Est	+35(4)82037066	Asthma	m
525	AB-	Andre Rempel	603 Damaris Mountain	707.508.1361x9	NULL	f
531	B+	Maud Anderson	55018 Helene Coves A	(181)708-7043	Blood Pressure	f

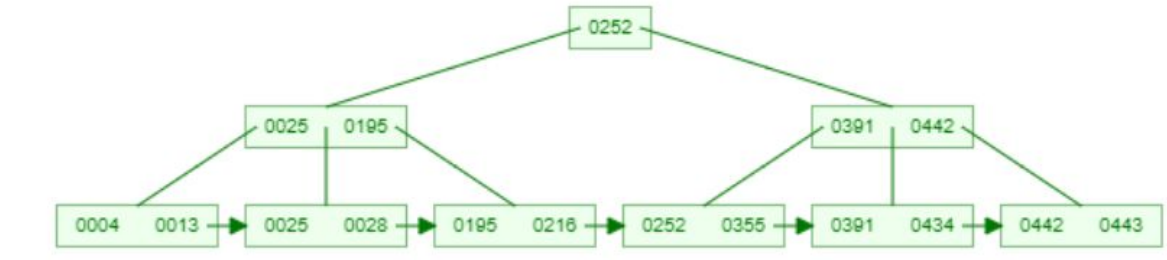
➔ DCL

```
Create user 'Matt_Pitman';
Grant All privileges on bloodbank to 'Matt_Pitman';
```

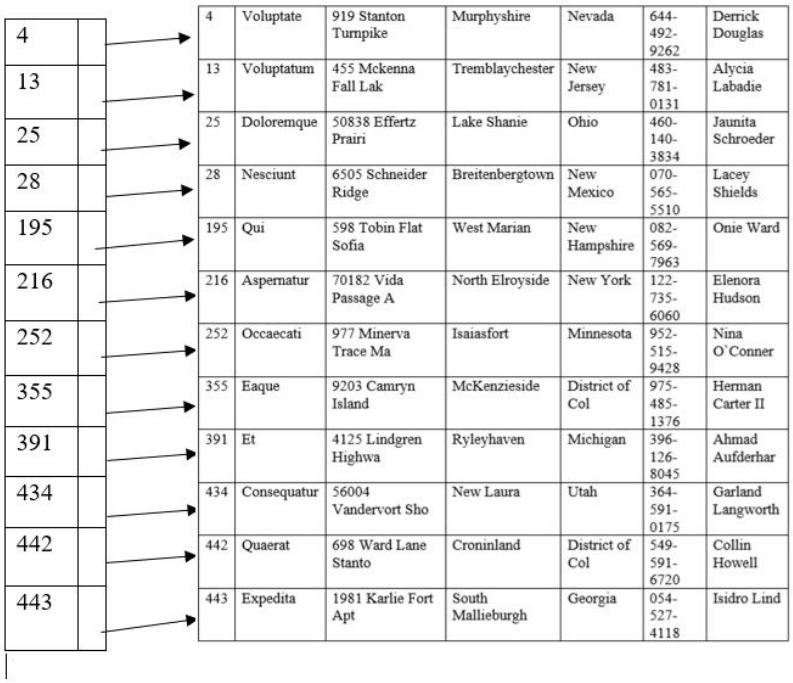
✓	422	21:20:11	Create user 'Matt_Pitman'	0 row(s) affected
✓	423	21:20:21	Grant All privileges on bloodbank to 'Matt_Pitman'	0 row(s) affected

6) Indexes

➔ B+ tree of Blood Bank Table ID with degree 4:



➔ Dense Index by Blood Bank ID



4	Voluptate	919 Stanton Turnpike	Murphyshire	Nevada	644-492-9262	Derrick Douglas
13	Voluptatum	455 Mckenna Fall Lak	Tremblaychester	New Jersey	483-781-0131	Alycia Labadie
25	Doloremque	50838 Effertz Prairi	Lake Shanie	Ohio	460-140-3834	Jaunita Schroeder
28	Nesciunt	6505 Schneider Ridge	Breitenbergtown	New Mexico	070-565-5510	Lacey Shields
195	Qui	598 Tobin Flat Sofia	West Marian	New Hampshire	082-569-7963	Onie Ward
216	Aspernatur	70182 Vida Passage A	North Elroyside	New York	122-735-6060	Elenora Hudson
252	Occaecati	977 Minerva Trace Ma	Isaiasfort	Minnesota	952-515-9428	Nina O`Conner
355	Eaque	9203 Camryn Island	McKenzieside	District of Col	975-485-1376	Herman Carter II
391	Et	4125 Lindgren Highwa	Ryleyhaven	Michigan	396-126-8045	Ahmad Aufderhar
434	Consequatur	56004 Vandervort Sho	New Laura	Utah	364-591-0175	Garland Langworth
442	Quaerat	698 Ward Lane Stanto	Croninland	District of Col	549-591-6720	Collin Howell
443	Expedita	1981 Karlie Fort Apt	South Mallieburgh	Georgia	054-527-4118	Isidro Lind

7) Views

Being the display of the necessary/required data with the respect to the demand/query provided, the implementation of ‘view’ was only necessary. Views were structured with the principles based on the materialized view approach. Such approach was specifically adapted due to the convenient nature of the materialized views. Being an important factor, the managerial processes associated with the materialized view maintenance were critically examined and the most suitable method was deliberated prior to their incorporation into the project. By manually defining triggers on view-related functions (insert, delete, and any form of update) in the view function’s definition, an automatic incremental view maintenance stature was established. Hence, the views would be updated rather quickly. Below is an example to demonstrate the implementation of BMBB’s materialized view.

Donor_ID	Blood_Type	Name	Address	Phone_Number	Medical_Condition	Gender
132	A+	Tillman Lang	3331 Lebsack Park Ke	1-017-658-1589	Lung Disease	m
163	O+	Miss Fae Kuhlman DDS	633 Braun Neck New J	676-600-3355x1	Blood Pressure	m
166	B+	Deangelo Corkery	69050 Schimmel Missi	1-326-041-0102	NULL	f
187	O+	Juston Connelly	559 Hoppe Ramp Apt.	961.563.5269	Asthma	f
197	O+	Prof. Gonzalo Ferry	5821 Evan Burg North	+57(2)26943639	Diabetes	f
218	O-	Daniela Rohan	140 Macie Station Po	(048)991-1092	Blood Pressure	m
302	AB-	Idella Mayer	797 Jacobson Lights	(759)017-6835	Lung Disease	f
513	A+	Mr. Salvador Hudson	869 Pfannerstill Est	+35(4)82037066	Asthma	m
525	AB-	Andre Rempel	603 Damaris Mountain	707.508.1361x9	NULL	f
531	B+	Maud Anderson	55018 Helene Coves A	(181)708-7043	Blood Pressure	f

8) Description of the Technology used for your interface implementation

For our frontend interface implementation we considered several options but ultimately landed on using tkinter. Tkinter is the standard Python interface. The reason we chose to use tkinter was that we felt it gave us the most flexibility when designing the frontend. There were few limitations with tkinter that a lot of other interfaces had. It also was easy to connect to our database in MySQL. This was most of our first time using tkinter so it took a while for us to get adjusted to using it but once we were able to start implementing it, we managed to get something presentable. The interface allows the user to update, delete, and insert into tables from the database along with viewing the different tables. The way we implemented this included a frontend and a backend. The frontend of tkinter was all of the visual stuff that the user would see when using the GUI. The backend of tkinter gave the buttons and visuals functionality. It also connected tkinter to our database. Overall we felt that we could have done more for the interface but were happy with what we produced, considering it was most of our first times.