INDEX

Abstract

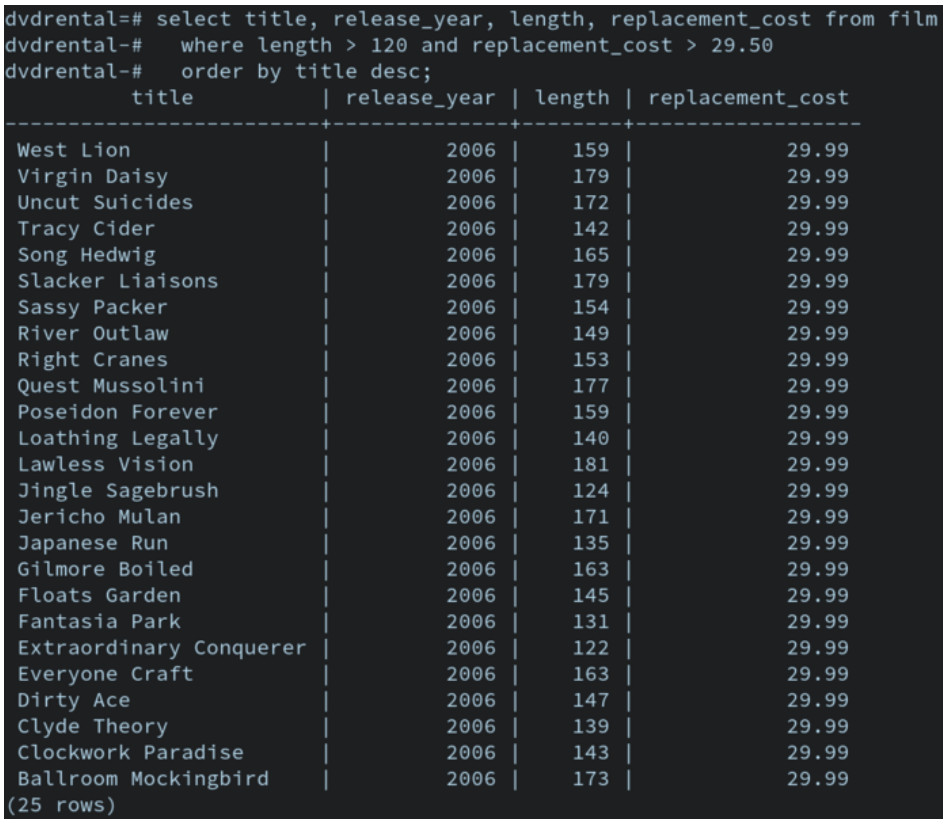
This paper is scrutinizes the use of different terms and syntaxes in Database Management System, enabling viewer to get the complete concept of different aspects of Database Management System. To satisfy this we created a database managed for Blood Bank System. Use of various syntaxes was used as a reference to the output, satisfying every need of a perfect Database Management.

Introduction

1. Database Management System:

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques. The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.



```
dvdrental=# select title, release_year, length, replacement_cost from film
dvdrental-#   where length > 120 and replacement_cost > 29.50
dvdrental-#   order by title desc;
          title          | release_year | length | replacement_cost
-------------------------+--------------+--------+------------------
 West Lion               |         2006 |    159 |            29.99
 Virgin Daisy            |         2006 |    179 |            29.99
 Uncut Suicides          |         2006 |    172 |            29.99
 Tracy Cider             |         2006 |    142 |            29.99
 Song Hedwig             |         2006 |    165 |            29.99
 Slacker Liaisons        |         2006 |    179 |            29.99
 Sassy Packer            |         2006 |    154 |            29.99
 River Outlaw            |         2006 |    149 |            29.99
 Right Cranes            |         2006 |    153 |            29.99
 Quest Mussolini         |         2006 |    177 |            29.99
 Poseidon Forever        |         2006 |    159 |            29.99
 Loathing Legally        |         2006 |    140 |            29.99
 Lawless Vision          |         2006 |    181 |            29.99
 Jingle Sagebrush        |         2006 |    124 |            29.99
 Jericho Mulan           |         2006 |    171 |            29.99
 Japanese Run            |         2006 |    135 |            29.99
 Gilmore Boiled          |         2006 |    163 |            29.99
 Floats Garden           |         2006 |    145 |            29.99
 Fantasia Park           |         2006 |    131 |            29.99
 Extraordinary Conquerer |         2006 |    122 |            29.99
 Everyone Craft          |         2006 |    163 |            29.99
 Dirty Ace               |         2006 |    147 |            29.99
 Clyde Theory            |         2006 |    139 |            29.99
 Clockwork Paradise      |         2006 |    143 |            29.99
 Ballroom Mockingbird    |         2006 |    173 |            29.99
(25 rows)
```

Fig.1. A Database

2. SQL:

SQL (Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables. SQL offers two main advantages over older read–write APIs such as ISAM or VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements,which may be informally classed as sublanguages, commonly: a data query language (DQL), a data definition language (DDL), a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is essentially a declarative language (4GL), it also includes procedural elements.

Fig.2. Structured Query Language (SQL)

3. RDBMS:

A relational database (RDBMS) is a digital database based on the relational model of data, as proposed by **E. F. Codd** in 1970. A software system used to maintain relational databases is a relational database management system (RDBMS). Many relational database systems have an option of using the SQL (Structured Query Language) for querying and maintaining the database. The most common definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules.

A second school of thought argues that if a database does not implement all of Codd's rules it is not relational. This view, shared by many theorists and other strict adherents to Codd's principles, would disqualify most DBMSs as not relational. For clarification, they often refer to some RDBMSs as truly-relational database management systems (TRDBMS), naming others pseudo-relational database management systems (PRDBMS).As of 2009, most commercial relational DBMSs employ SQL as their query language.Alternative query languages have been proposed and implemented, notably the pre-1996 implementation of Ingres QUEL.

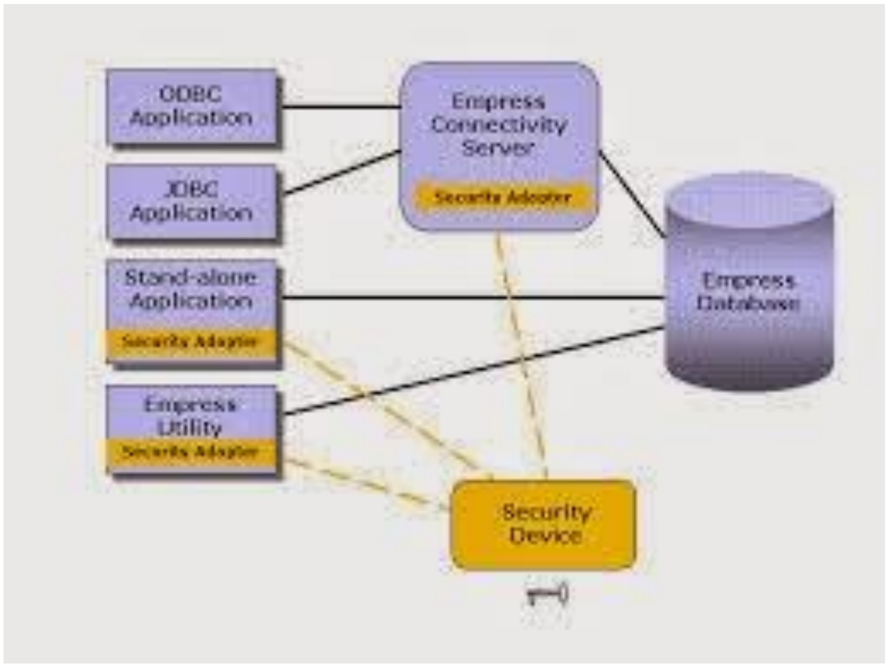


Fig.3. A RDBMS Structure

History

1. DBMS:

The sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. These performance increases were enabled by the technology progress in the areas of processors, computer memory, computer storage, and computer networks. The concept of a database was made possible by the emergence of direct access storage media such as magnetic disks, which became widely available in the mid 1960s; earlier systems relied on sequential storage of data on magnetic tape. The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2018 they remain dominant: IBM DB2, Oracle, MySQL, and Microsoft SQL Server are the most searched DBMS.The dominant database language, standardised SQL for the relational model, has influenced database languages for other data models.

The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.

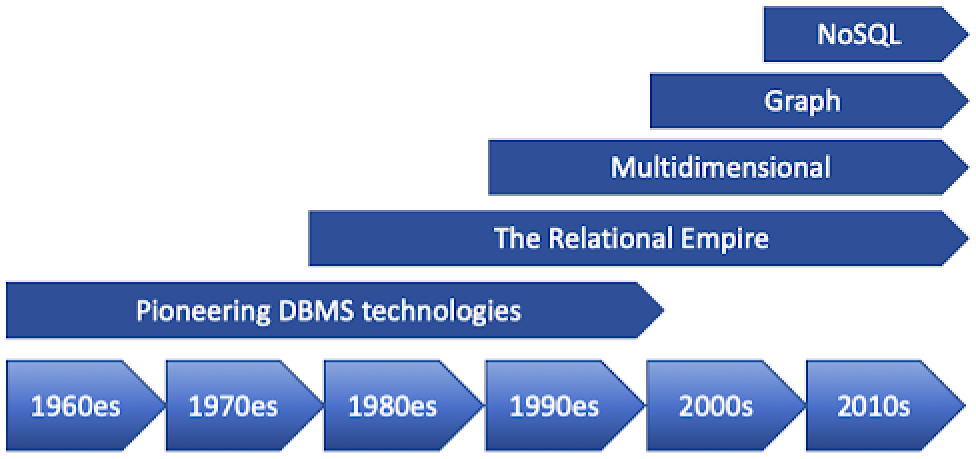


Fig.4. History

2. SQL:

SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce after learning about the relational model from Edgar F. Codd in the early 1970s. This version, initially called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, System R, which a group at IBM San Jose Research Laboratory had developed during the 1970s. Chamberlin and Boyce's first attempt at a relational database language was Square, but it was difficult to use due to subscript notation. After moving to the San Jose Research Laboratory in 1973, they began work on SEQUEL.

In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce, and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, Central Intelligence Agency, and other U.S. government agencies. In June 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers.

By 1986, ANSI and ISO standard groups officially adopted the standard "Database Language SQL" language definition. New versions of the standard were published in 1989, 1992, 1996, 1999, 2003, 2006, 2008, 2011 and, most recently, 2016.

Fig.5. Oracle Corporations

3. RDBMS:

The term "relational database" was invented by E. F. Codd at IBM in 1970. Codd introduced the term in his research paper "A Relational Model of Data for Large Shared Data Banks". In this paper and later papers, he defined what he meant by "relational". One well-known definition of what constitutes a relational database system is composed of Codd's 12 rules. However, no commercial implementations of the relational model conform to all of Codd's rules, so the term has gradually come to describe a broader class of database systems, which at a minimum.

The most common definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules.

As of 2009, most commercial relational DBMSs employ SQL as their query language.

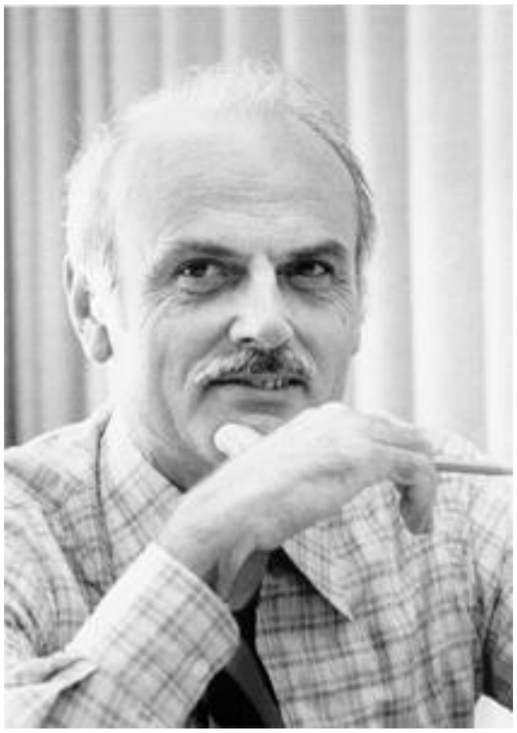


Fig.6. E.F. Codd

E-R Diagram

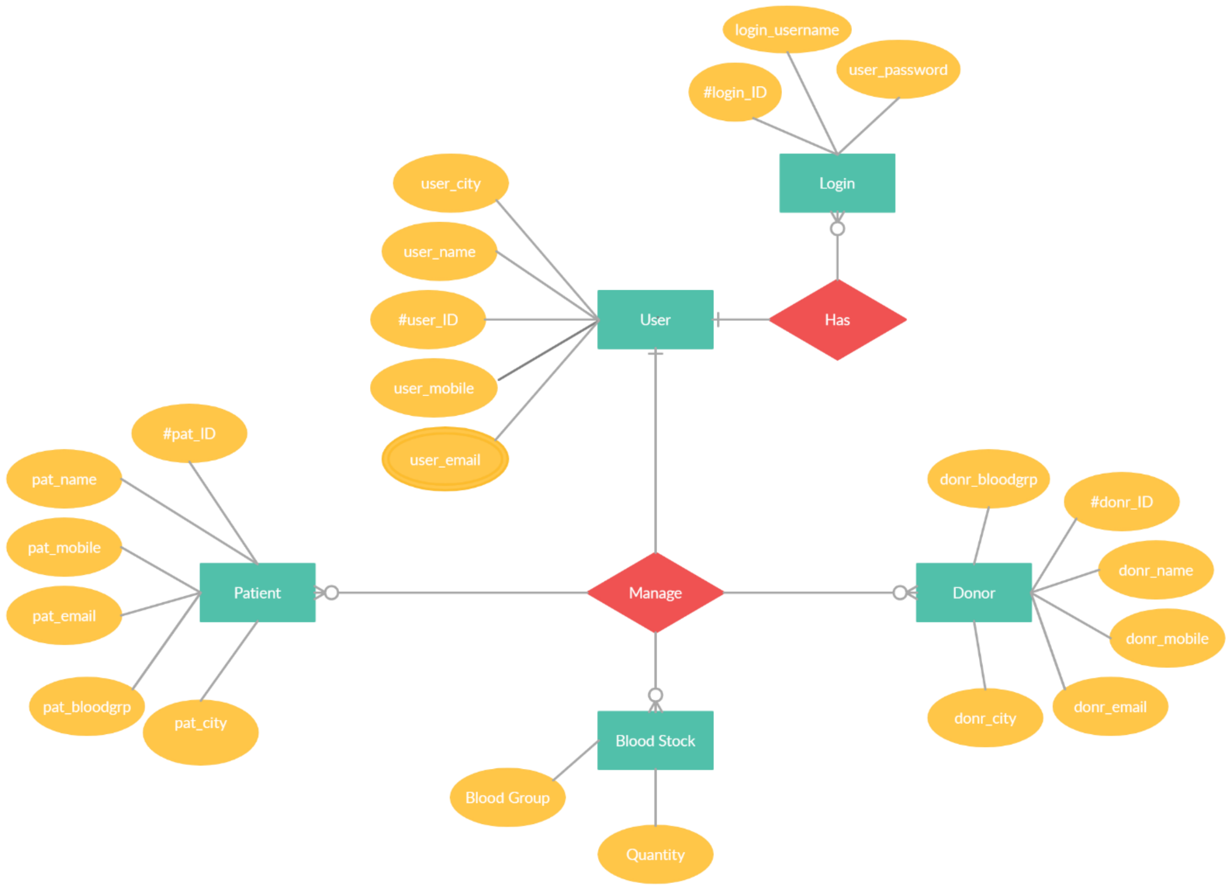
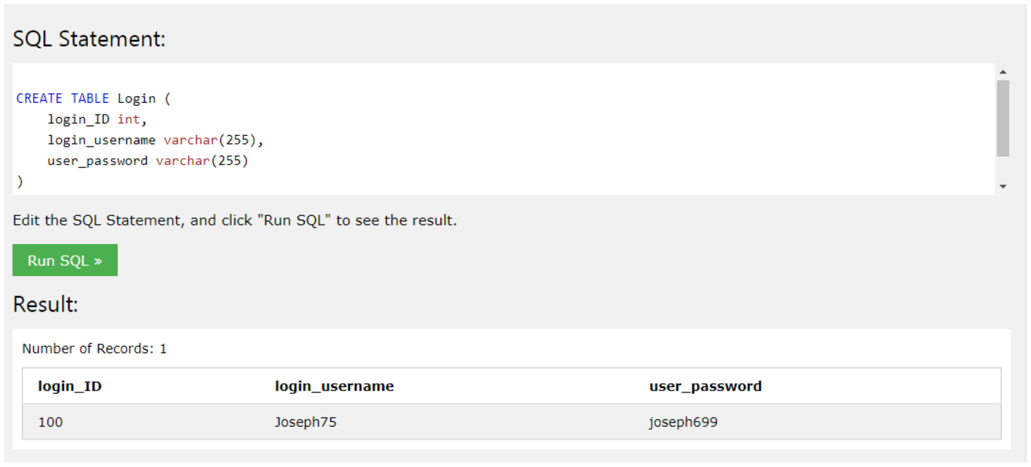


Fig. 7. E-R Diagram

Queries With Output

```
CREATE TABLE Login (
    login_ID int,
    login_username varchar(255),
    user_password varchar(255)
);
INSERT INTO Login (login_ID,login_username,user_password)
VALUES ('100','Joseph75','joseph699');
```



SQL Statement:

```
CREATE TABLE Login (
    login_ID int,
    login_username varchar(255),
    user_password varchar(255)
)
```

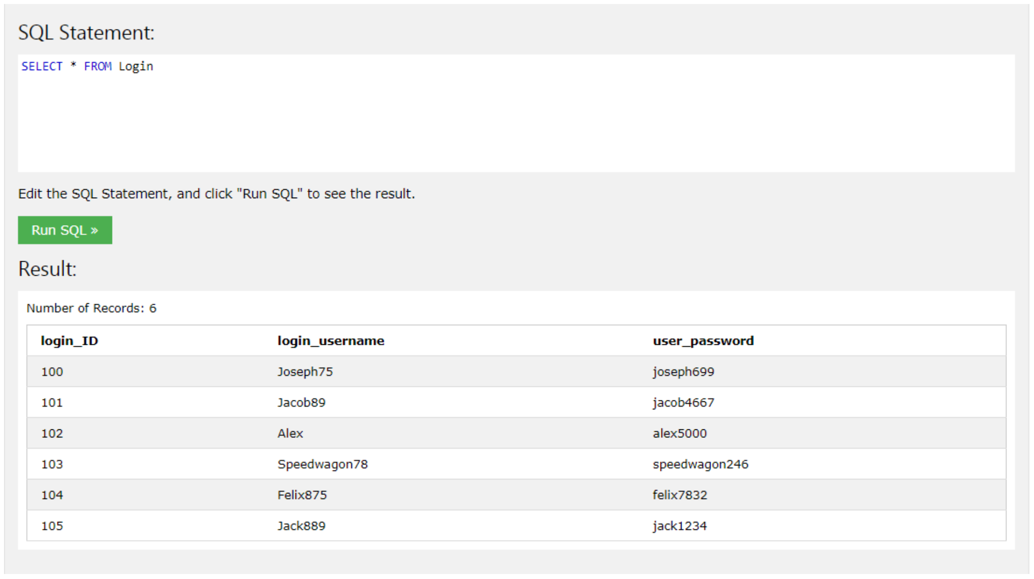
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 1

login_ID	login_username	user_password
100	Joseph75	joseph699

```
/* After 5 statements */
Select * from Login
```



SQL Statement:

```
SELECT * FROM Login
```

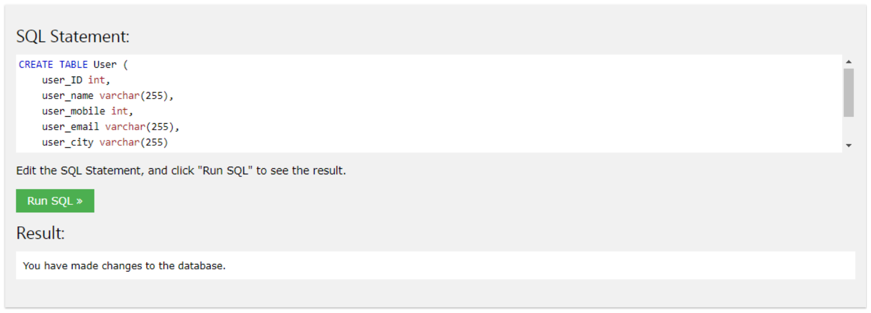
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 6

login_ID	login_username	user_password
100	Joseph75	joseph699
101	Jacob89	jacob4667
102	Alex	alex5000
103	Speedwagon78	speedwagon246
104	Felix875	felix7832
105	Jack889	jack1234

```
CREATE TABLE User (
    user_ID int,
    user_name varchar(255),
    user_mobile int,
    user_email varchar(255),
    user_city varchar(255)
);
```


SQL Statement:

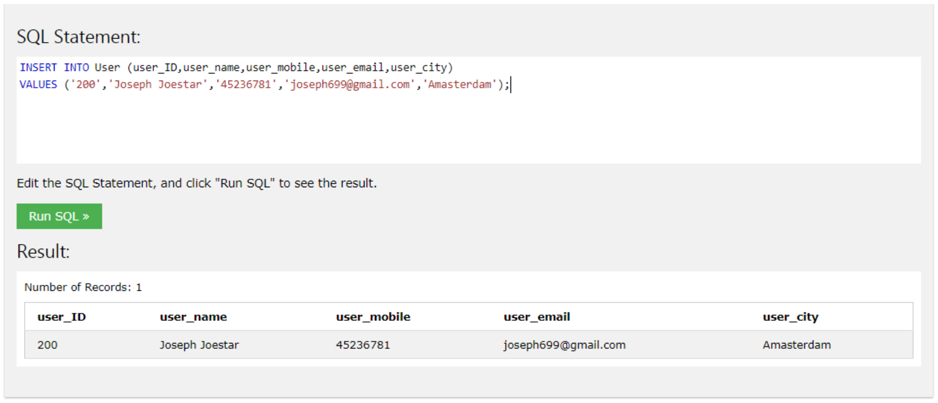
```
CREATE TABLE User (
    user_ID int,
    user_name varchar(255),
    user_mobile int,
    user_email varchar(255),
    user_city varchar(255)
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

You have made changes to the database.

```
INSERT INTO User (user_ID,user_name,user_mobile,user_email,user_city)
VALUES ('200','Joseph Joestar','45236781','joseph699@gmail.com','Amasterdam');
```


SQL Statement:

```
INSERT INTO User (user_ID,user_name,user_mobile,user_email,user_city)
VALUES ('200','Joseph Joestar','45236781','joseph699@gmail.com','Amasterdam');
```

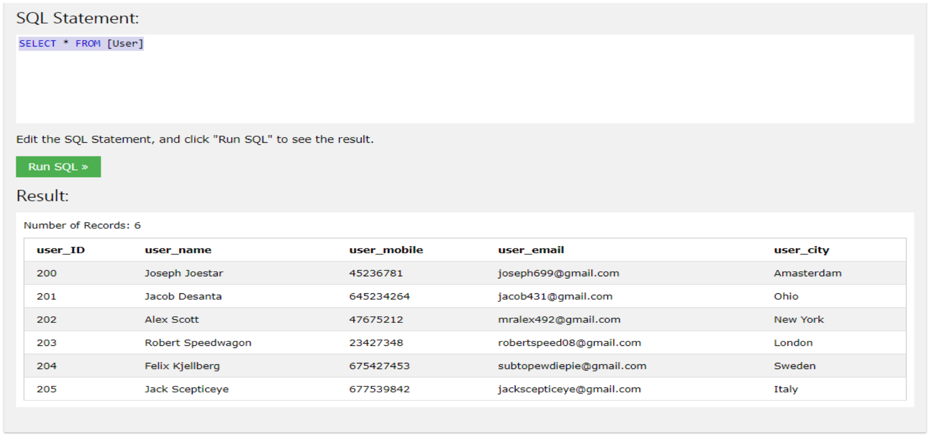
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 1

user_ID	user_name	user_mobile	user_email	user_city
200	Joseph Joestar	45236781	joseph699@gmail.com	Amasterdam

```
/* After 5 statements */
SELECT * FROM [User]
```


SQL Statement:

```
SELECT * FROM [User]
```

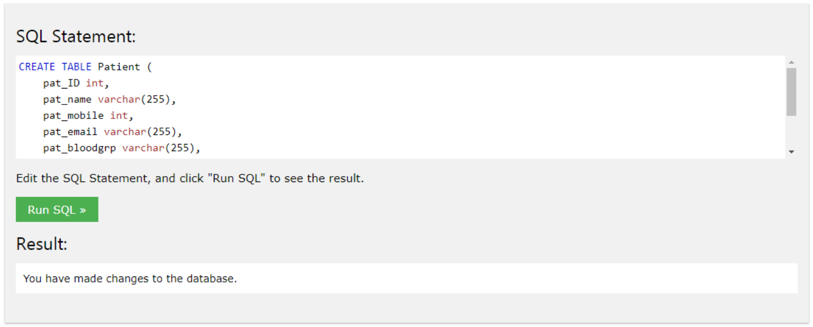
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 6

user_ID	user_name	user_mobile	user_email	user_city
200	Joseph Joestar	45236781	joseph699@gmail.com	Amasterdam
201	Jacob Desanta	645234264	jacob431@gmail.com	Ohio
202	Alex Scott	47675212	mralex492@gmail.com	New York
203	Robert Speedwagon	23427348	robertspeed08@gmail.com	London
204	Felix Kjellberg	675427453	subtopewdiepie@gmail.com	Sweden
205	Jack Scepticeye	677539842	jackscepticeye@gmail.com	Italy

```
CREATE TABLE Patient (

    pat_ID int,

    pat_name varchar(255),

    pat_mobile int,

    pat_email varchar(255),

    pat_bloodgrp varchar(255),

    pat_city varchar(255)

);
```



SQL Statement:

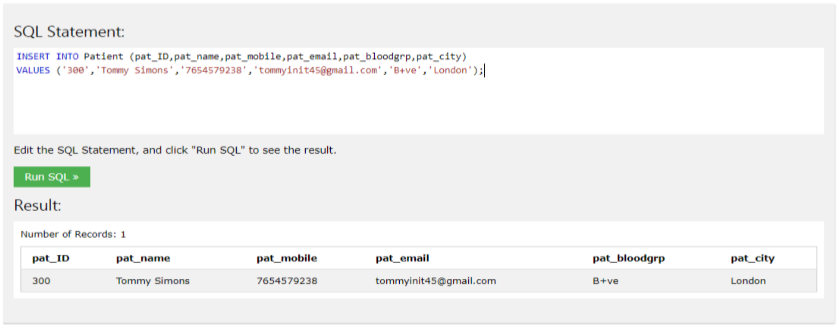
```
CREATE TABLE Patient (
    pat_ID int,
    pat_name varchar(255),
    pat_mobile int,
    pat_email varchar(255),
    pat_bloodgrp varchar(255),
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

```
You have made changes to the database.
```

```
INSERT INTO Patient (pat_ID,pat_name,pat_mobile,pat_email,pat_bloodgrp,pat_city)

VALUES ('300','Tommy Simons','7654579238','tommyinit45@gmail.com','B+ve','London');
```



SQL Statement:

```
INSERT INTO Patient (pat_ID,pat_name,pat_mobile,pat_email,pat_bloodgrp,pat_city)
VALUES ('300','Tommy Simons','7654579238','tommyinit45@gmail.com','B+ve','London');
```

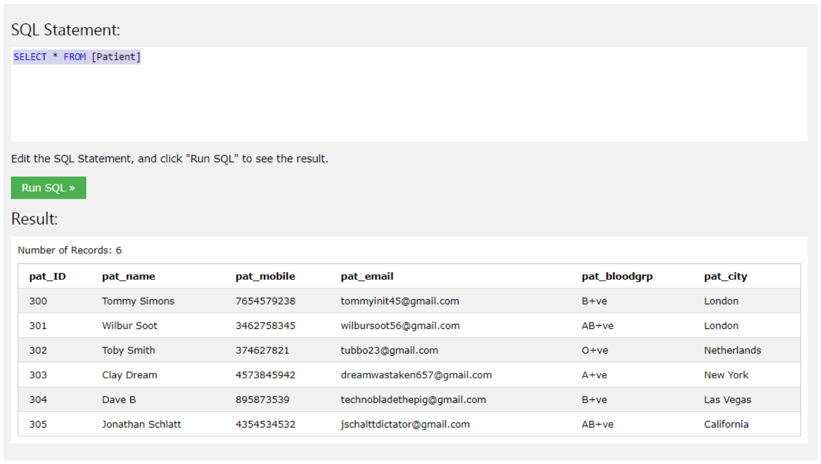
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 1

pat_ID	pat_name	pat_mobile	pat_email	pat_bloodgrp	pat_city
300	Tommy Simons	7654579238	tommyinit45@gmail.com	B+ve	London

```
/* After 5 statements */

SELECT * FROM [Patient]
```



SQL Statement:

```
SELECT * FROM [Patient]
```

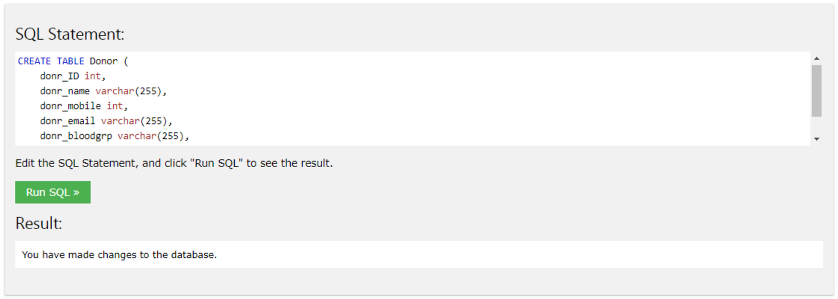
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 6

pat_ID	pat_name	pat_mobile	pat_email	pat_bloodgrp	pat_city
300	Tommy Simons	7654579238	tommyinit45@gmail.com	B+ve	London
301	Wilbur Soot	3462758345	wilbursoot56@gmail.com	AB+ve	London
302	Toby Smith	374627821	tubbo23@gmail.com	O+ve	Netherlands
303	Clay Dream	4573845942	dreamwastaken657@gmail.com	A+ve	New York
304	Dave B	895873539	technobladethepig@gmail.com	B+ve	Las Vegas
305	Jonathan Schlatt	4354534532	jschalttdictator@gmail.com	AB+ve	California

```
CREATE TABLE Donor (

    donr_ID int,

    donr_name varchar(255),

    donr_mobile int,

    donr_email varchar(255),

    donr_bloodgrp varchar(255),

    donr_city varchar(255)

);
```



SQL Statement:

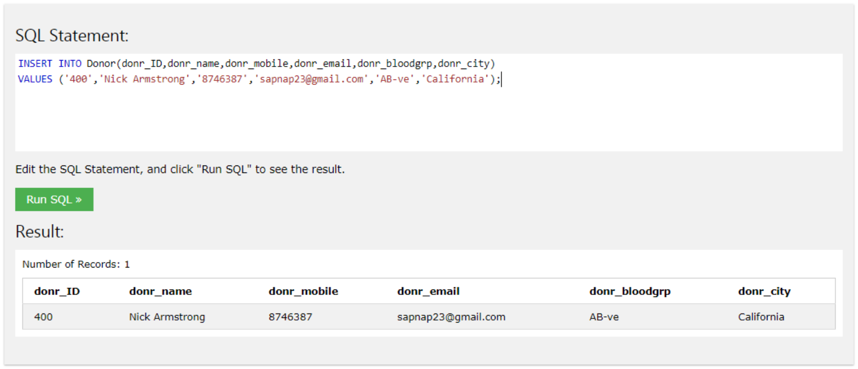
```
CREATE TABLE Donor (
    donr_ID int,
    donr_name varchar(255),
    donr_mobile int,
    donr_email varchar(255),
    donr_bloodgrp varchar(255),
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

```
You have made changes to the database.
```

```
INSERT INTO Donor(donr_ID,donr_name,donr_mobile,donr_email,donr_bloodgrp,donr_city)

VALUES ('400','Nick Armstrong','8746387','sapnap23@gmail.com','AB-ve','California');
```



SQL Statement:

```
INSERT INTO Donor(donr_ID,donr_name,donr_mobile,donr_email,donr_bloodgrp,donr_city)
VALUES ('400','Nick Armstrong','8746387','sapnap23@gmail.com','AB-ve','California');
```

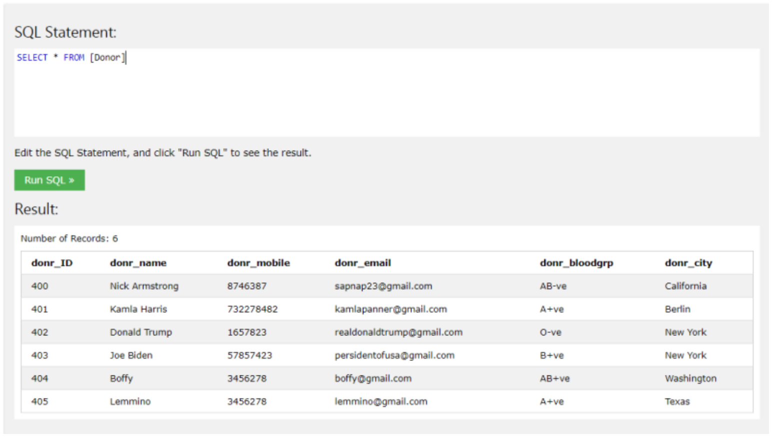
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 1

donr_ID	donr_name	donr_mobile	donr_email	donr_bloodgrp	donr_city
400	Nick Armstrong	8746387	sapnap23@gmail.com	AB-ve	California

```
/* After 5 statements */

SELECT * FROM [Donor]
```



SQL Statement:

```
SELECT * FROM [Donor]
```

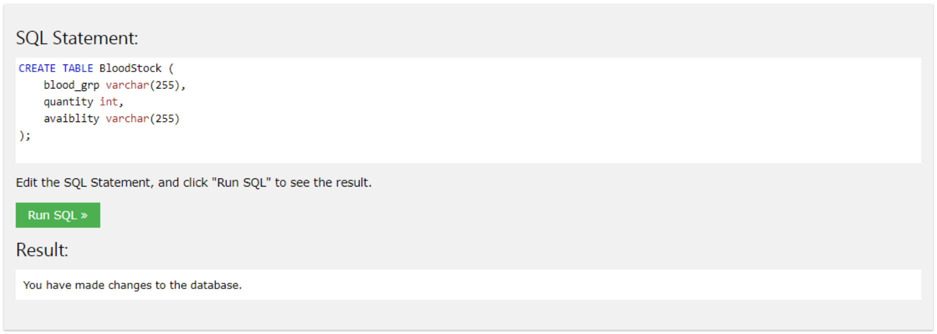
Edit the SQL Statement, and click "Run SQL" to see the result.

Result:

Number of Records: 6

donr_ID	donr_name	donr_mobile	donr_email	donr_bloodgrp	donr_city
400	Nick Armstrong	8746387	sapnap23@gmail.com	AB-ve	California
401	Kamla Harris	732278482	kamlapanner@gmail.com	A+ve	Berlin
402	Donald Trump	1657823	realdonaldtrump@gmail.com	O-ve	New York
403	Joe Biden	57857423	persidentofusa@gmail.com	B+ve	New York
404	Boffy	3456278	boffy@gmail.com	AB+ve	Washington
405	Lemmino	3456278	lemmino@gmail.com	A+ve	Texas

```
CREATE TABLE BloodStock (

    blood_grp varchar(255),

    quantity int,

    avaiblity varchar(255)

);
```



SQL Statement:

```
CREATE TABLE BloodStock (
    blood_grp varchar(255),
    quantity int,
    avaiblity varchar(255)
);
```

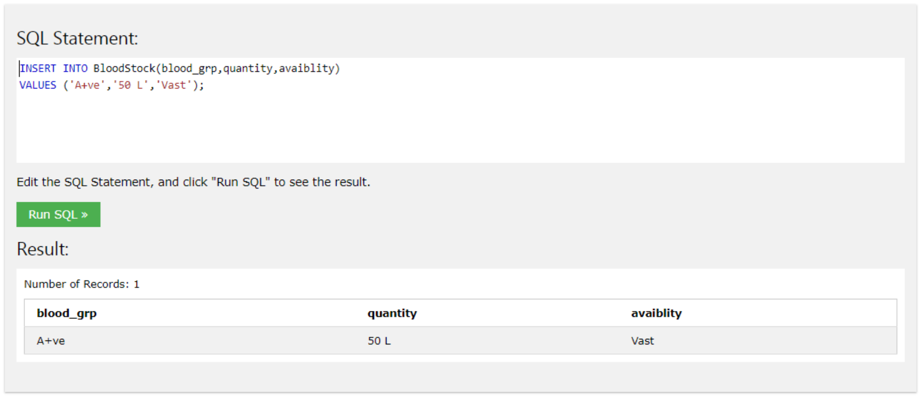
Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

You have made changes to the database.

```
INSERT INTO BloodStock(blood_grp,quantity,avaiblity)

VALUES ('A+ve','50 L','Vast');
```



SQL Statement:

```
INSERT INTO BloodStock(blood_grp,quantity,avaiblity)
VALUES ('A+ve','50 L','Vast');
```

Edit the SQL Statement, and click "Run SQL" to see the result.

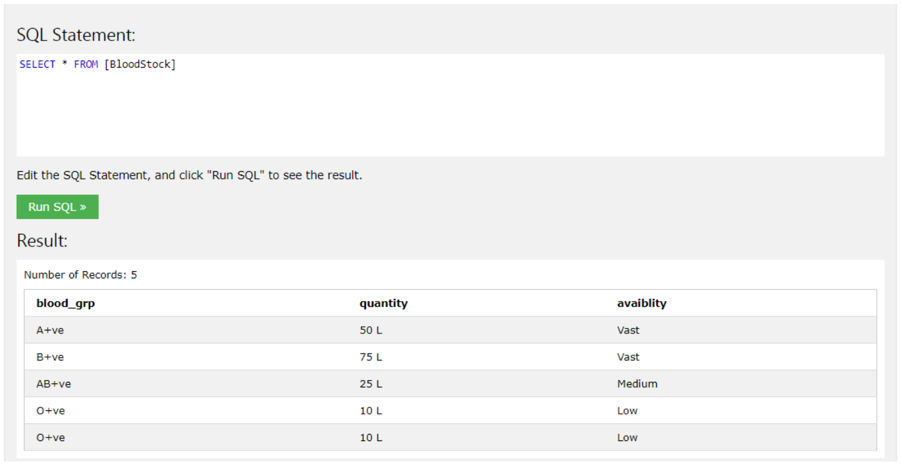
Run SQL »

Result:

Number of Records: 1

blood_grp	quantity	avaiblity
A+ve	50 L	Vast

```
/* After 5 statements */

SELECT * FROM [BloodStock]
```



SQL Statement:

```
SELECT * FROM [BloodStock]
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

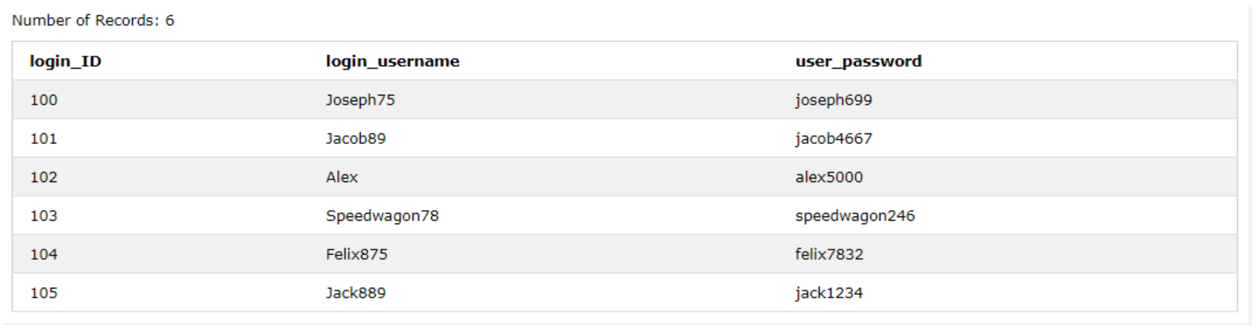
Result:

Number of Records: 5

blood_grp	quantity	avaiblity
A+ve	50 L	Vast
B+ve	75 L	Vast
AB+ve	25 L	Medium
O+ve	10 L	Low
O+ve	10 L	Low

Final Database

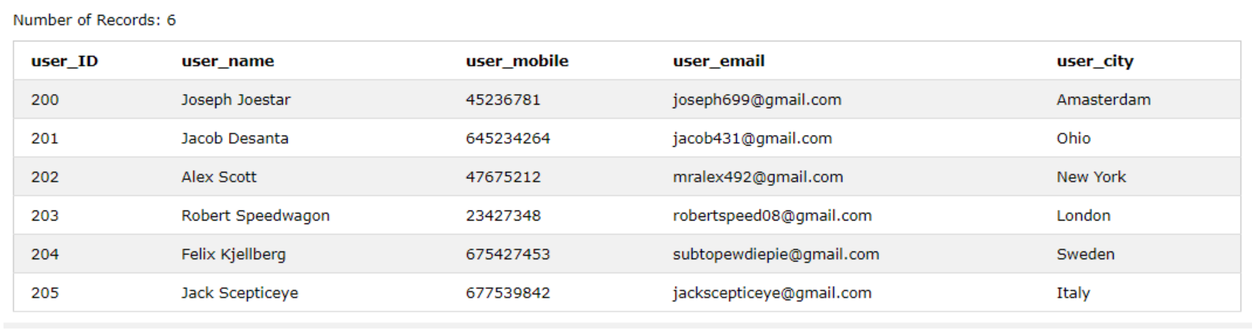
1. Login Table:



Number of Records: 6

login_ID	login_username	user_password
100	Joseph75	joseph699
101	Jacob89	jacob4667
102	Alex	alex5000
103	Speedwagon78	speedwagon246
104	Felix875	felix7832
105	Jack889	jack1234

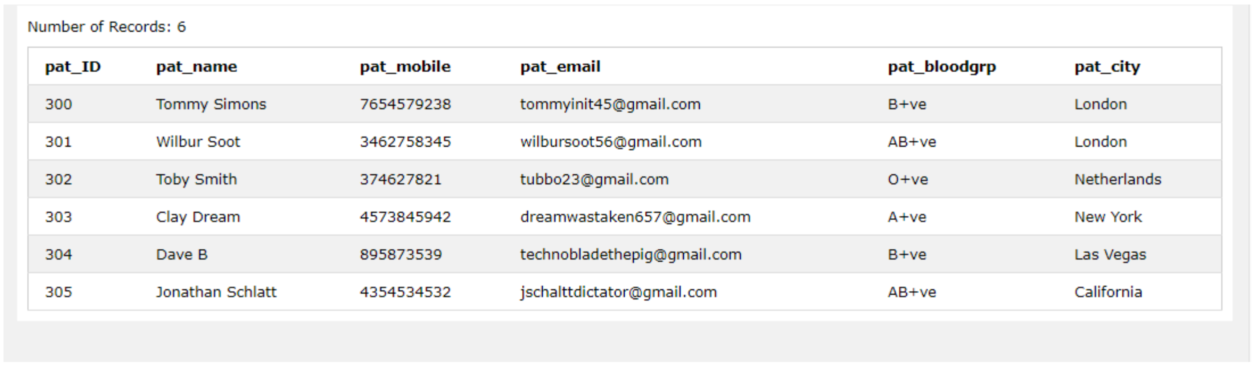
2. User Table:



Number of Records: 6

user_ID	user_name	user_mobile	user_email	user_city
200	Joseph Joestar	45236781	joseph699@gmail.com	Amasterdam
201	Jacob Desanta	645234264	jacob431@gmail.com	Ohio
202	Alex Scott	47675212	mralex492@gmail.com	New York
203	Robert Speedwagon	23427348	robertspeed08@gmail.com	London
204	Felix Kjellberg	675427453	subtopewdiepie@gmail.com	Sweden
205	Jack Scepticeye	677539842	jackscepticeye@gmail.com	Italy

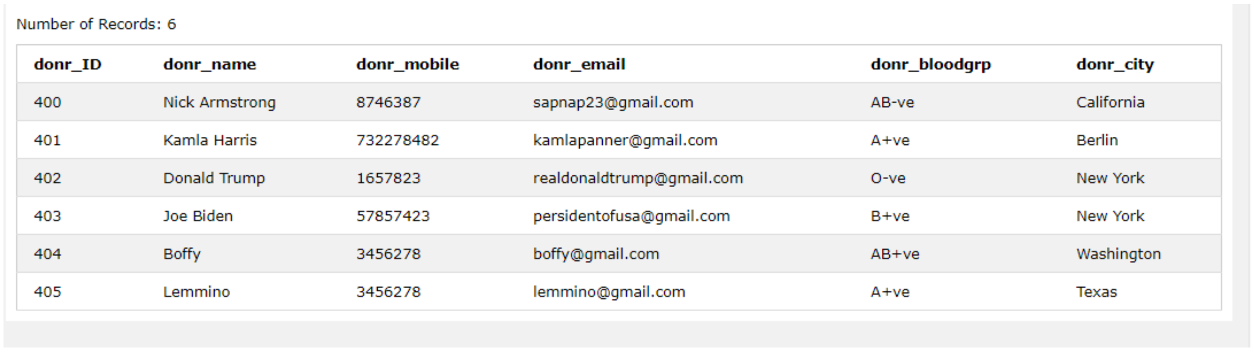
3. Patient Table:



Number of Records: 6

pat_ID	pat_name	pat_mobile	pat_email	pat_bloodgrp	pat_city
300	Tommy Simons	7654579238	tommyinit45@gmail.com	B+ve	London
301	Wilbur Soot	3462758345	wilbursoot56@gmail.com	AB+ve	London
302	Toby Smith	374627821	tubbo23@gmail.com	O+ve	Netherlands
303	Clay Dream	4573845942	dreamwastaken657@gmail.com	A+ve	New York
304	Dave B	895873539	technobladethepig@gmail.com	B+ve	Las Vegas
305	Jonathan Schlatt	4354534532	jschalttdictator@gmail.com	AB+ve	California

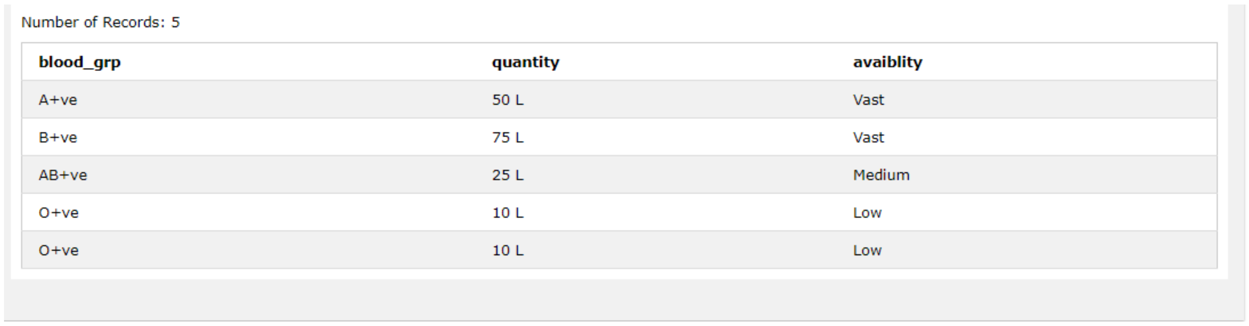
4. Donor Table:



Number of Records: 6

donr_ID	donr_name	donr_mobile	donr_email	donr_bloodgrp	donr_city
400	Nick Armstrong	8746387	sapnap23@gmail.com	AB-ve	California
401	Kamla Harris	732278482	kamlapanner@gmail.com	A+ve	Berlin
402	Donald Trump	1657823	realdonaldtrump@gmail.com	O-ve	New York
403	Joe Biden	57857423	persidentofusa@gmail.com	B+ve	New York
404	Boffy	3456278	boffy@gmail.com	AB+ve	Washington
405	Lemmino	3456278	lemmino@gmail.com	A+ve	Texas

5. Blood Stock



Number of Records: 5

blood_grp	quantity	avaiblity
A+ve	50 L	Vast
B+ve	75 L	Vast
AB+ve	25 L	Medium
O+ve	10 L	Low
O+ve	10 L	Low

Applications

Application of DBMS:

Sector	Use
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

Conclusion

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application. Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management.

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

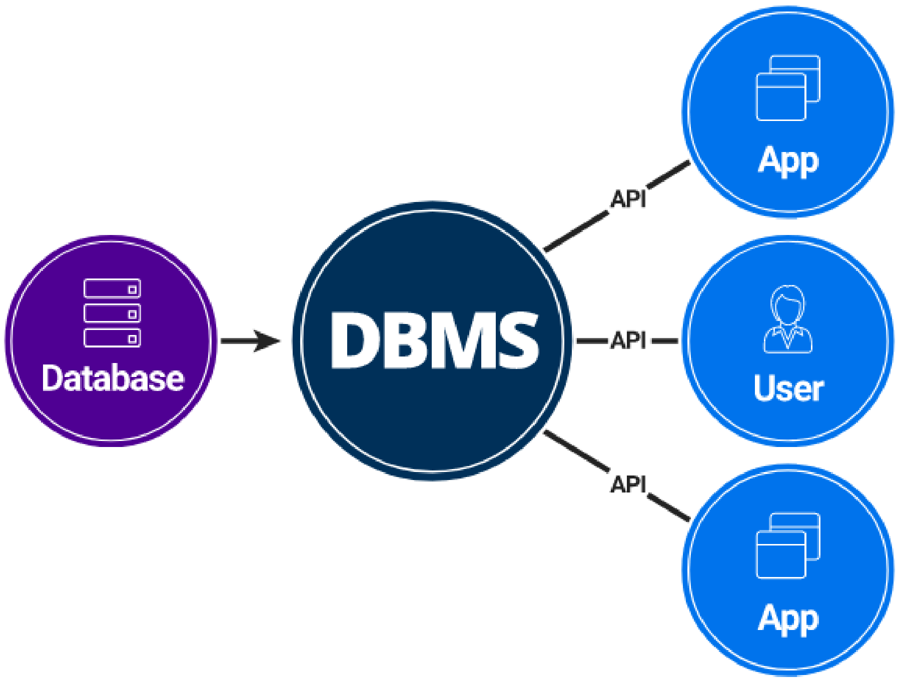


Fig.8. DBMS