A Report on

Developing a Voting Management System

Prepared For

Dr Amit Dua

(Instructor In-Charge)

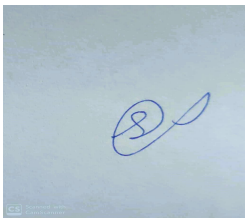
CS F212 - Database Management

Prepared By
Chaitanya Sethi - 2020B3A71961P
Stavya Puri - 2020B5A70912P

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
SECOND SEMESTER AY 2022-23
JANUARY - MAY 2023

DECLARATION STATEMENT

We hereby declare that this project was solely written by the members of this group. We are aware that the incorporation of code/material from other sources will be treated as plagiarism, subject to the custom and usage of the subject, according to the guidelines provided by the institution. We are fully responsible for its contents and declare all contents plagiarism-free. Wherever necessary, all the references are duly quoted, and a bibliography is provided at the end.

Chaitanya Sethi	**2020B3A71961P**	
Stavya Puri	**2020B5A70912P**	

Acknowledgement

The success of this report would not have been possible without the assistance and guidance of our teachers. We would like to thank the Vice-Chancellor (BITS Pilani University), Dr Souvik Bhattacharya and the Director of BITS Pilani (Pilani Campus), Prof. Sudhir Kumar Barai. We would thank our instructors and mentors for this assignment dedicated to International Economics, Prof. Dr Amit Dua (Instructor In-Charge) and his skilled team of instructors and scholars for allowing us to conduct this study under the course **CS F212 - Database Management** and for guiding us throughout the semester to make the project comprehensive through his insights and knowledge. Lastly, we would also like to thank our family members and close friends for their moral support.

Table of Contents

1. INTRODUCTION

A software programme called a "voting management system" is intended to administer the election process in a secure and transparent way. Voter registration, candidate registration, voting, and vote tallying are all made easier using it. Elections of all kinds, including those for national, state, local, and organisational offices, can be conducted using the system.

In order to ensure fair and transparent elections, a voting management system is essential for public administration. The technology offers a safe and effective way for voters to cast their ballots, as well as a way for candidates to sign up and keep track of the election. Additionally, it aids in avoiding fraud and other abnormalities that can jeopardise the election's credibility.

Public administration may make sure that all qualified voters can participate in the election process and that the results are reliable and accurate by implementing a voting management system. The technique can also aid in minimising the expense and time involved with conventional paper-based voting procedures.

In order to ensure fair and transparent elections and respect the democratic norms of free and fair elections, public administration must use a voting management system.

In the above spirit, as tasked, a Voting Management System has been developed using MySQL, starting with an Enhanced Entity Model, Relational Schema, which has been translated into actual SQL queries provided and implemented. The system is capable of registering(inserting) users(voters), candidates, and administrators for elections, creating new elections and generating reports of election winners, ranking and total votes cast.

2. Enhanced-Entity(or Entity Relationship) Relationship Diagram

This section explores the logical assumptions and the working of the EER.

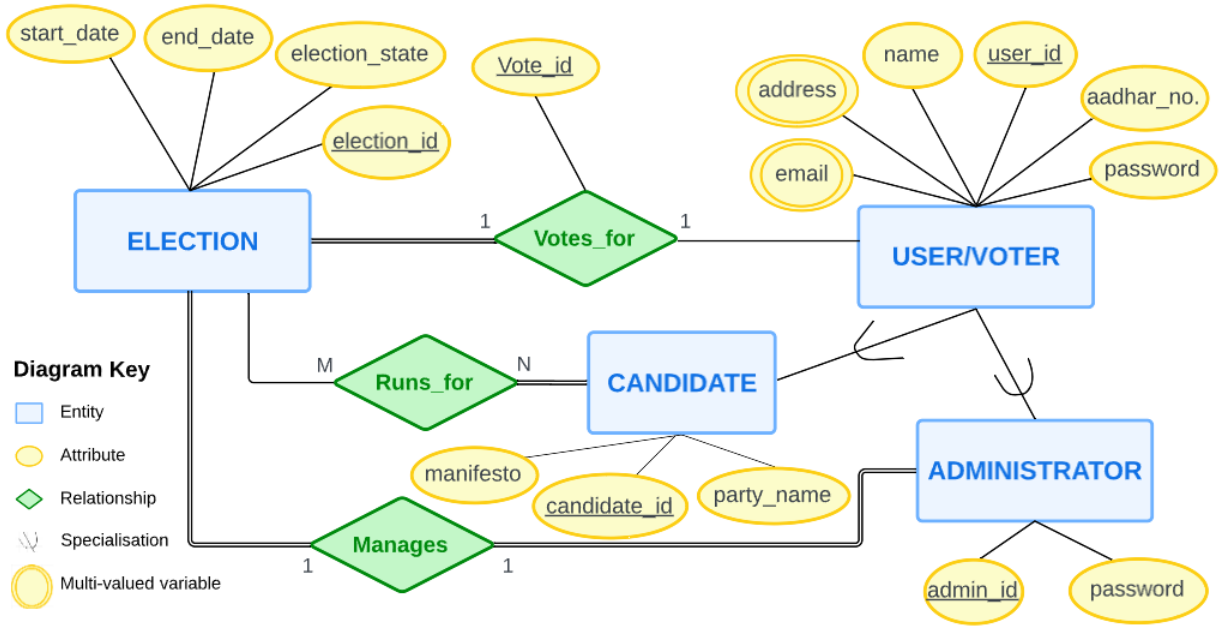


Figure 1. ENHANCED ENTITY RELATIONSHIP DIAGRAM

Figure 1. Enhanced Entity Relationship Diagram

Here, we have created Entities ELECTION, USER/VOTER, CANDIDATE AND ADMINISTRATOR. We have also defined Relationships VOTES_FOR, RUNS_FOR, and MANAGES. We have considered specialisation that is ADMINISTRATOR and CANDIDATE as the children of USER as they also have voting rights. Cardinalities between Entities on the basis of various Relationships are shown above.

3. KEY LOGICAL ASSUMPTIONS AND WORKING

As a complex and multi-variable system, certain logical assumptions have been considered alongside logical working for certain functions involved, further solidifying the understanding of the above-discussed EER diagram.

General Assumptions

1) The user is a generalisation of the candidate, administrator and other voters, as the candidates running in the election or the administrator handling the election also have the right to vote. This allows us to store all attributes of the candidate and administrator only once under user and reduce redundancy of repeated storage and update.
2) As per the standard voting regime, multiple candidates can stand for multiple elections across various states, but a voter can vote only once in a particular election. Additionally, each election is managed by a single administrator.
3) The administrator is a fair and just person who does not manipulate the voting system based on his personal interests.
4) Each election has an administrator, who manages the election and has the exclusive right to extract the election results, reports and ranks from the database system.
5) The Voting System does not provide tie-resolutions, as it is assumed that there might be coalitions and other political propaganda which is uncertain and outside the scope of the database.

Function/Working-based Assumptions

6) If a new candidate is being inserted into the system, he/she must enter all attributes corresponding to the user first, which only occurs if the user does not exist already. After this, it is checked whether he is already an administrator or a candidate for the same election already. If not, he is added as a candidate to the particular election through the election_id, which is determined as the candidate must also provide election_id.
   Else, insertion is rejected.
7) If a new administrator is being entered, he/she must enter all attributes corresponding to the user first, which only occurs if the user does not exist already. After this, it is checked whether he is already a candidate or an administrator of other elections. In case of any of the above, the insertion is rejected.
8) Deletion and updation of the user lead to corresponding changes in the candidate's/administrator's values' deletion or updation as well, as the user is a generalisation of these entities as earlier specified.

9) While inserting an election, a corresponding administrator from the existing users/voters must also be assigned to manage it. Hence, along with election attributes, administrator's attributes are also provided, which includes user_id and a unique administrator id. Then the user_id is subject to checks
    1. If the user does not exist, insertion is rejected
    2. If the user exists but
        a. Is already a candidate, insertion is rejected as candidates are not assigned as election administrators
        b. Inserted admin_id matches already an administrator, insertion is rejected as one election is managed by a single administrator.
        c. user_id matches any previously added administrator's user id, then insertion is rejected as user_id, and admin_id are to be unique for each election.

    If all the above constraints are satisfied, the insertion occurs in the order:
        a) Election
        b) Administrator
        c) Manages

4. RELATIONAL SCHEMA

Given the above assumptions, constraints, and the EER diagram, the development of a relational schema, observations of functional dependencies involved, and normalisation may be conducted to form a relational schema in the desired 3NF Normal form.

4.1. Functional Dependencies, Conversions and Normalisations

We have proved by first checking the 3NF Property that is Absence of Transitive Dependence and then going further to check for 2NF and 1NF as a Relation in 3NF is already in 2NF and 1NF.

4.1.1. User Entity

USER											
user_id	**INT**	name	**VARCHAR**	address	**VARCHAR**	emaill	**VARCHAR**	aadhar_no.	**INT**	password	**VARCHAR**

The user entity has **six attributes** with their corresponding input types as shown here.

Primary Key: user_id

The functional dependencies involved here are

user_id ⟶ name, address, email, aadhar_no., password

aadhar_no. ⟶ name, address, email, aadhar_no, password

For the given entity and functional dependencies, all the left attributes are superkeys, and hence 3NF(BCNF) form is achieved as **no transitive dependencies are formed.**

4.1.2. Candidate Entity

CANDIDATE							
candidate_id	**INT**	user_id	party_name	**VARCHAR**	manifesto	**VARCHAR**	

The candidate entity, being a specialisation of the user entry, has additional two entries as party_name and manifesto, with

Primary Key: candidate_id

***Foreign Key*:** user_id

(referencing User table, identifies the candidate's basic attributes as he is also a voter)

The functional dependencies involved here are

candidate_id ⟶ user_id, party_name, manifesto

user_id ⟶ candidate_id, party_name, manifesto

Having chosen candidate_id as the Primary Key, it can be observed that both the functional dependencies involved here haven their left attributes as a superkey, ensuring no transitive dependency occurs. Thus, we achieve a **3NF(BCNF)** again using similar argument.

4.1.3. Election Entity

ELECTION							
election_id	**INT**	Election_State	**VARCHAR**	start_date	**VARCHAR**	end_date	**VARCHAR**

The Election entity has four attributes, with their corresponding input types as shown here.

Primary Key: election_id

The functional dependencies involved here are

election_id ⟶ Election_State, start_date, end_date

Election_State, start_date ⟶ **election_id**, end_date

Election_State, end_date ⟶ **election_id**, start_date

The second and third functional dependencies are constraints that in a state Elections can't start or end on the same date. as seen from the above function dependencies left attributes are super keys and hence transitive dependencies are not maintained. Hence we can conclude a 3NF form.

4.1.4. Administrator Entity

ADMINISTRATOR				
admin_id	**INT**	password	**VARCHAR**	user_id

The administrator entity, being a specialisation of the user entry, has one additional entry as a password, with

Primary Key: admin_id

Foreign Key: user_id

(referencing User table, identifies the admin's basic attributes as he is also a voter)

admin_id ⟶ *user_id*, password

user_id ⟶ **admin_id**, password

Again, from the Similar logic of having the left attributes are super keys transitive depends cannot exist, therefore leading to a 3NF form once again.

4.1.5. Votes_for relation

VOTES_FOR			
Vote_id **INT**	election_id	user_id	candidate_id

Moving on to the relations involved, this relation holds a primary key itself to denote votes uniquely, as the primary key.

Primary Key: vote_id

Foreign Keys:

(i) election_id (referencing the Election table, identifies the election the vote was cast in)

(ii) user_id (referencing the User table, identifies the user who cast the vote)

(iii) candidate_id (referencing the Candidate table, identifies the candidate the user voted for)

The functional dependencies involved are:

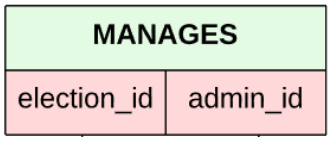
election_id, user_id ⟶ **vote_id**, *candidate_id*

vote_id ⟶ *election_id, user_id, candidate_id*

The unique primary key vote_id is used to register a vote. Additionally, the election ID along with the user ID of the voter can also determine every other attribute as given an election and voter we can get the vote because only one vote can be cast per voter in an election.

Here, since all the left attributes of both function dependencies are super key hence 3NF, that is transitivity is not achieved, rather BCNF is achieved.

4.1.6. Manages Relationship



Manages being a relationship between administrator and election has no additional attributes or primary keys.

Foreign Keys:

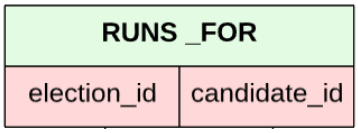
election_id (referencing the Election table, identifying which election is being maintained)
admin_id (referencing the User table, identifies the administrator responsible for the election management)
Involved Functional Dependencies are:

admin_id ⟶ *election_id*

election_id ⟶ *admin_id*

Only a single user can become the administrator for an election, and there is a one is to one correspondence. 3NF is maintained as both left attributes are superkeys and transitive dependency cannot exist.

4.1.7. Runs_for Relationship



Similar to manages runs_for relationship is between a candidate and the elections he or she is running for. It also has no additional attributes or primary keys.

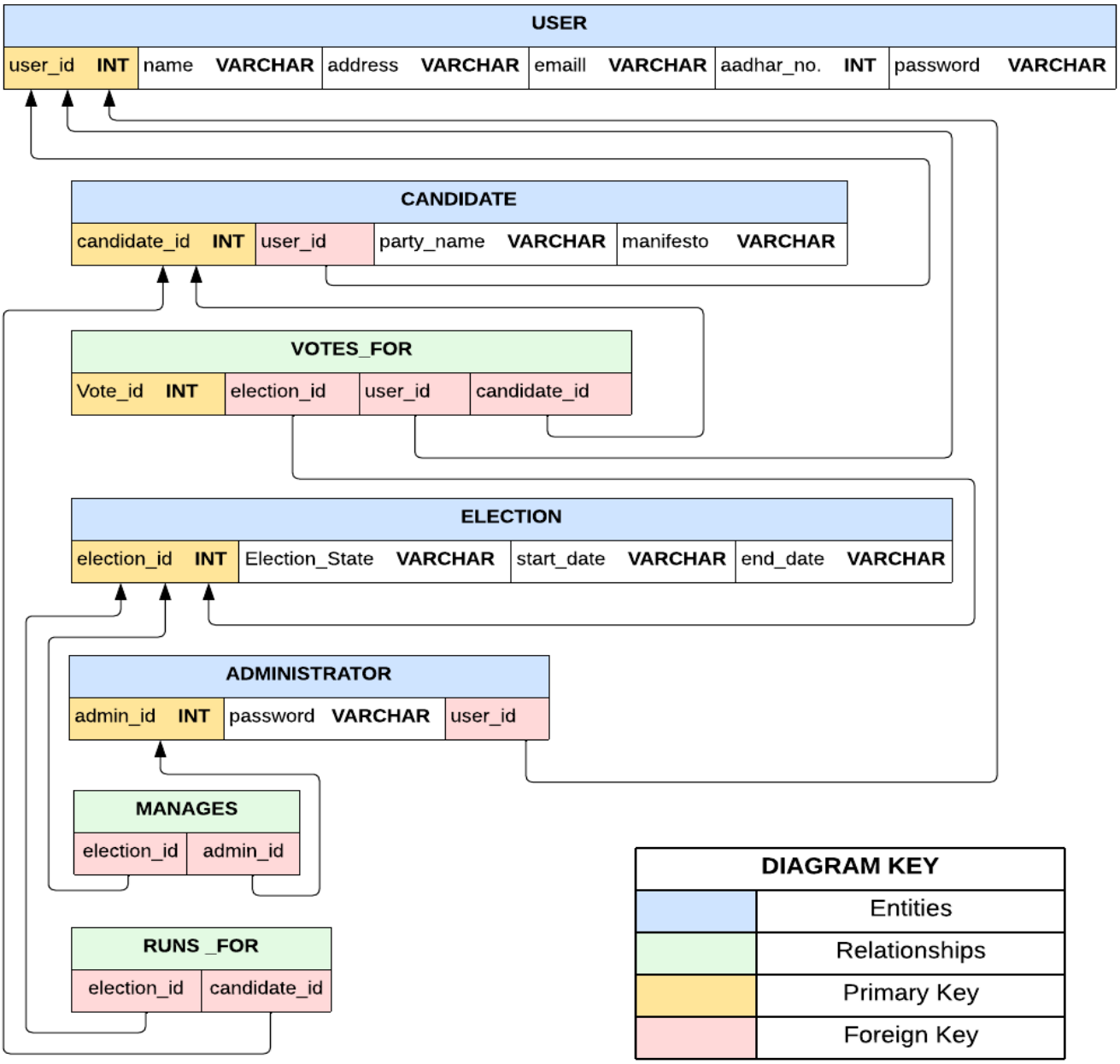
Foreign Keys:

election_id (referencing the Election table, identifying which election is the candidate running for)
candidate_id (referencing the candidate table, identifying which candidate is running for the election corresponding to the election_id)

No functional dependency as this is a relation with cardnality N:M. Thus, 3NF is maintained.

Through the above functional dependence observation and normalisation process, we now have the following relational schema, where the foreign key constraints across all relationships and entities are depicted through arrows:

Figure 2: Relational Schema - Voting Management System



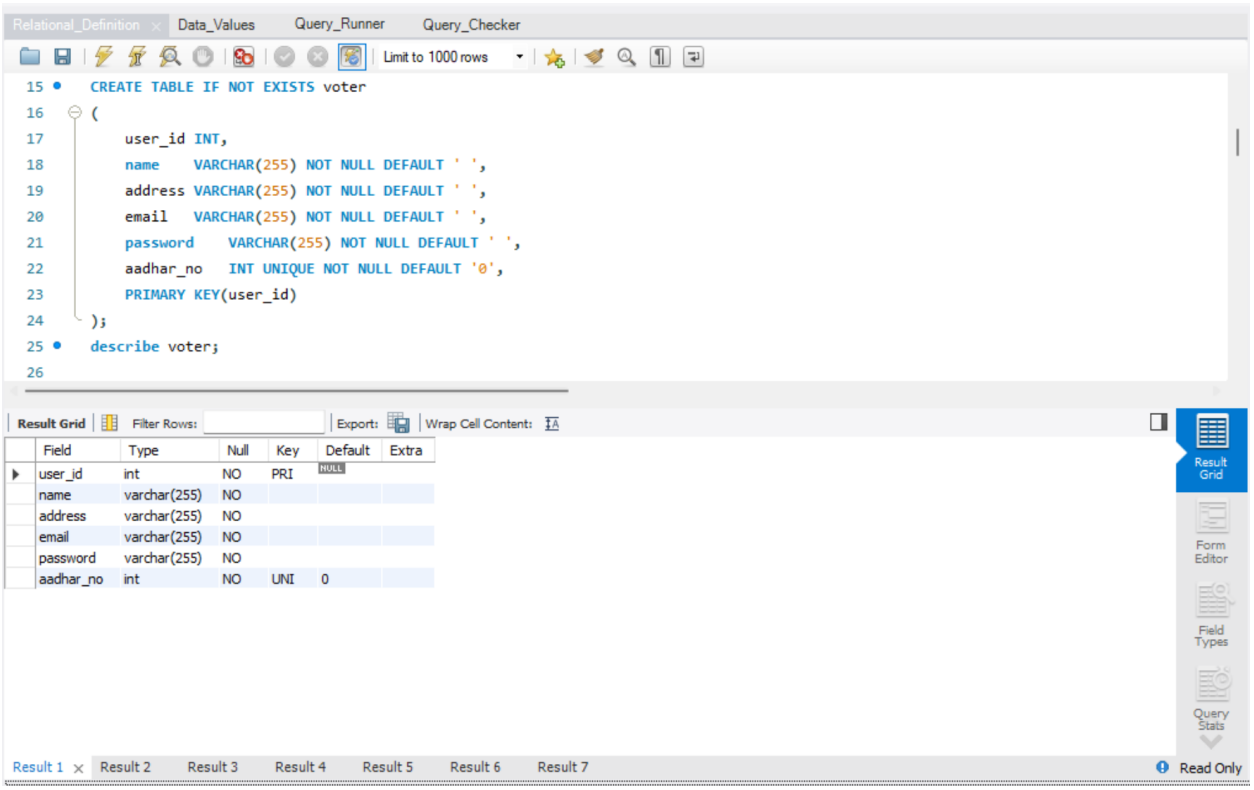
5. SQL Queries - Output Screenshot

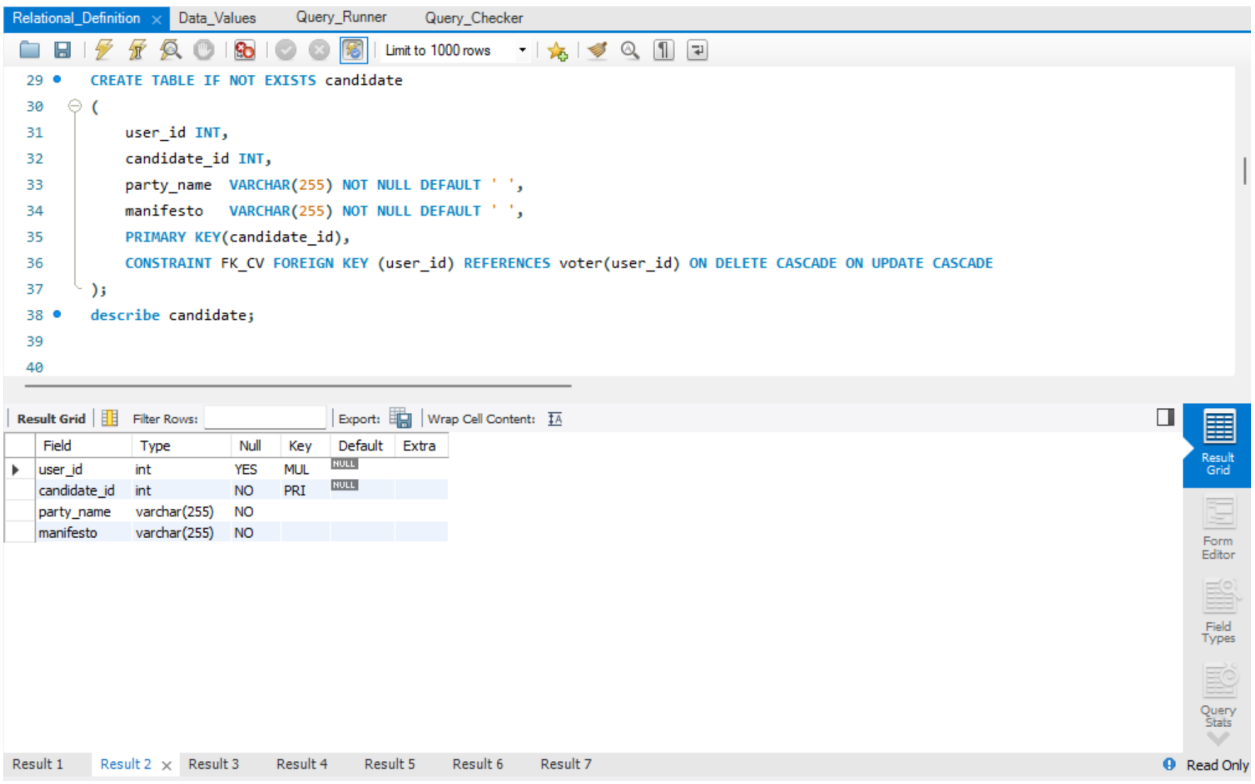
With all the theoretical and diagrammatical concepts in place, the following SQL queries which were provided have been implemented

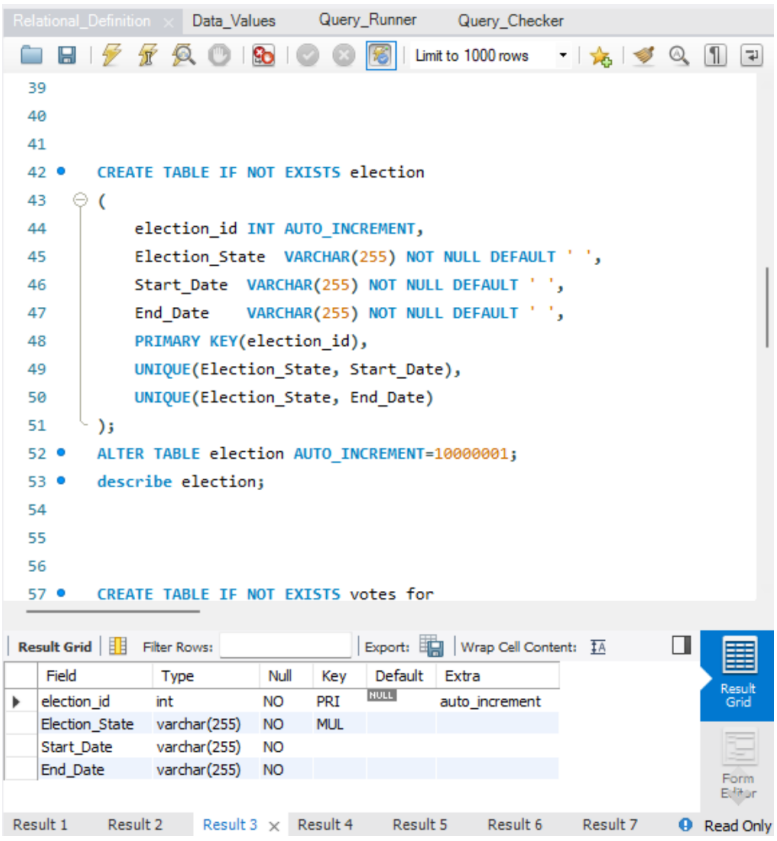
1. Create all the necessary tables, such as candidates table, users table,votes table elections table etc.
2. Insert a new user into the user table.
3. Insert a new candidate into the candidate table.
4. Write a query to register a vote.
5. Insert an election into the elections table.
6. Retrieve the number of votes for a candidate.
7. Retrieve the percentage of votes for a candidate.
8. Retrieve the winner of the election.
9. Retrieve the rank of each candidate based on the number of votes received.
10. Retrieve the total number of votes cast for an election.
11. Update the user's information.
12. Delete a specific user’s information.

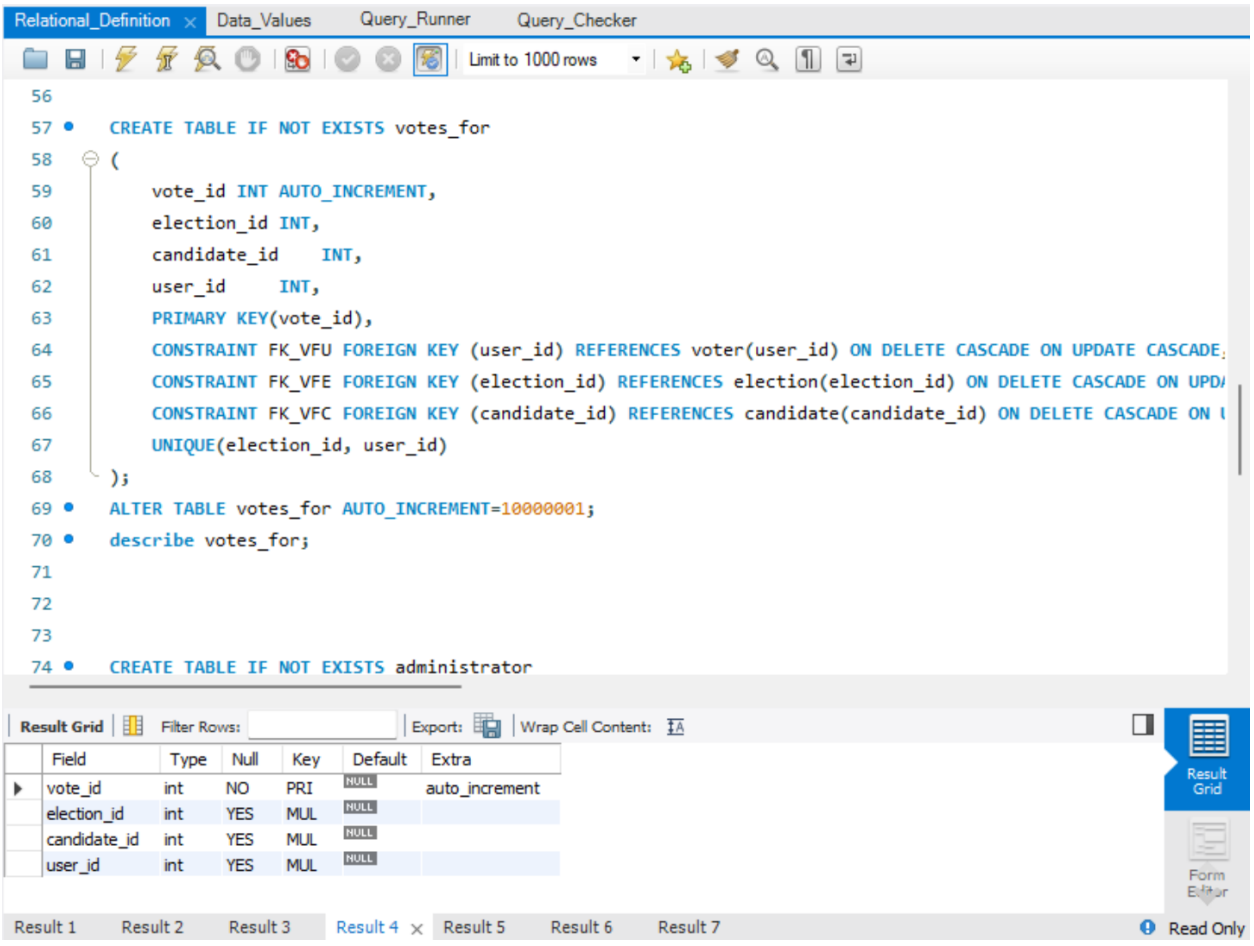
5.1. Relational Definition - Create all tables

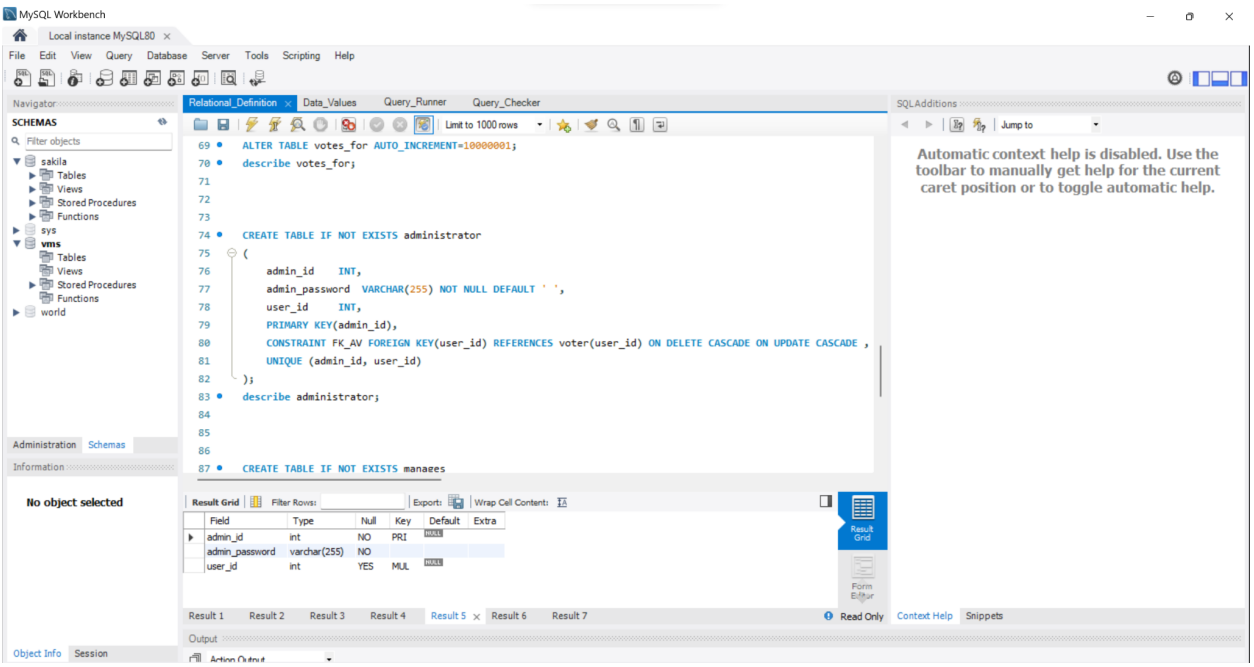
The database creation begins with the creation of all the tables involved (seven in number), which is executed by the Relational_Definition Query:

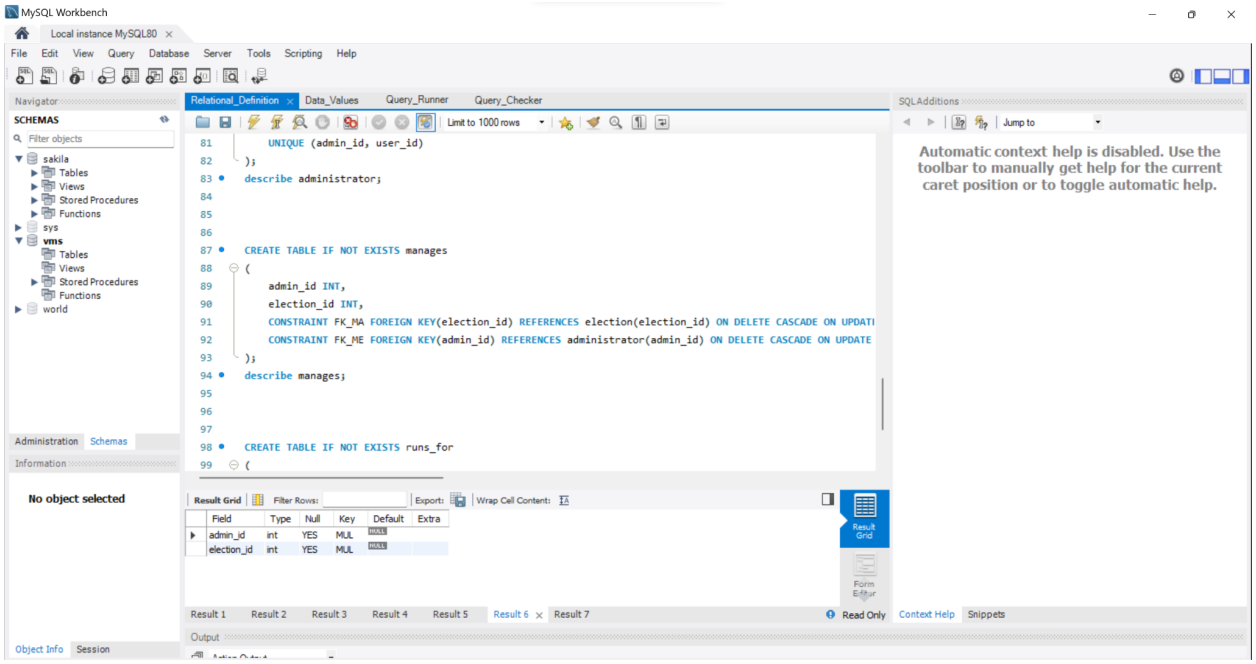

Relational_Definition
Data_Values
Query_Runner
Query_Checker
Limit to 1000 rows
15 CREATE TABLE IF NOT EXISTS voter
16 (
17 user_id INT,
18 name VARCHAR(255) NOT NULL DEFAULT ' ',
19 address VARCHAR(255) NOT NULL DEFAULT ' ',
20 email VARCHAR(255) NOT NULL DEFAULT ' ',
21 password VARCHAR(255) NOT NULL DEFAULT ' ',
22 aadhar_no INT UNIQUE NOT NULL DEFAULT '0',
23 PRIMARY KEY(user_id)
24);
25 describe voter;
26
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Field Type Null Key Default Extra
user_id int NO PRI NULL
name varchar(255) NO
address varchar(255) NO
email varchar(255) NO
password varchar(255) NO
aadhar_no int NO UNI 0
Result Grid
Form Editor
Field Types
Query Stats
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Read Only

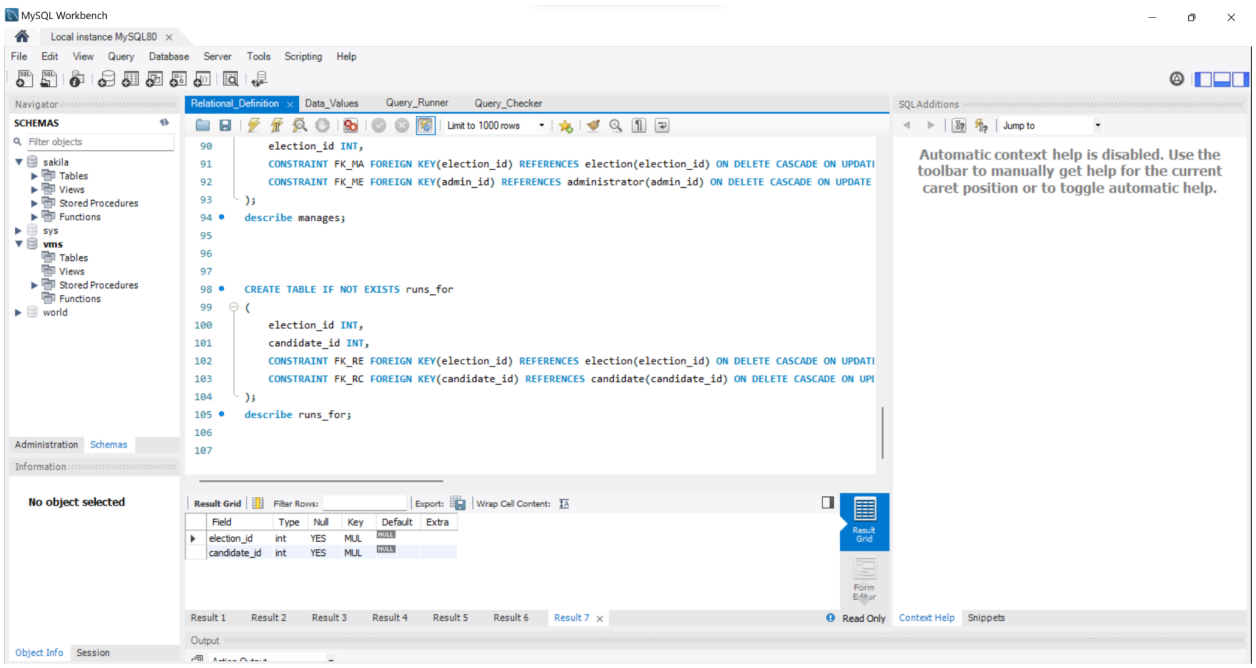

Relational_Definition
Data_Values
Query_Runner
Query_Checker
Limit to 1000 rows
29 CREATE TABLE IF NOT EXISTS candidate
30 (
31 user_id INT,
32 candidate_id INT,
33 party_name VARCHAR(255) NOT NULL DEFAULT ' ',
34 manifesto VARCHAR(255) NOT NULL DEFAULT ' ',
35 PRIMARY KEY(candidate_id),
36 CONSTRAINT FK_CV FOREIGN KEY (user_id) REFERENCES voter(user_id) ON DELETE CASCADE ON UPDATE CASCADE
37);
38 describe candidate;
39
40
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Field Type Null Key Default Extra
user_id int YES MUL NULL
candidate_id int NO PRI NULL
party_name varchar(255) NO
manifesto varchar(255) NO
Result Grid
Form Editor
Field Types
Query Stats
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Read Only


Relational_Definition
Data_Values
Query_Runner
Query_Checker
Limit to 1000 rows
39
40
41
42 CREATE TABLE IF NOT EXISTS election
43 (
44 election_id INT AUTO_INCREMENT,
45 Election_State VARCHAR(255) NOT NULL DEFAULT ' ',
46 Start_Date VARCHAR(255) NOT NULL DEFAULT ' ',
47 End_Date VARCHAR(255) NOT NULL DEFAULT ' ',
48 PRIMARY KEY(election_id),
49 UNIQUE(Election_State, Start_Date),
50 UNIQUE(Election_State, End_Date)
51);
52 ALTER TABLE election AUTO_INCREMENT=10000001;
53 describe election;
54
55
56
57 CREATE TABLE IF NOT EXISTS votes for
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Field Type Null Key Default Extra
election_id int NO PRI NULL auto_increment
Election_State varchar(255) NO MUL
Start_Date varchar(255) NO
End_Date varchar(255) NO
Result Grid
Form Editor
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Read Only

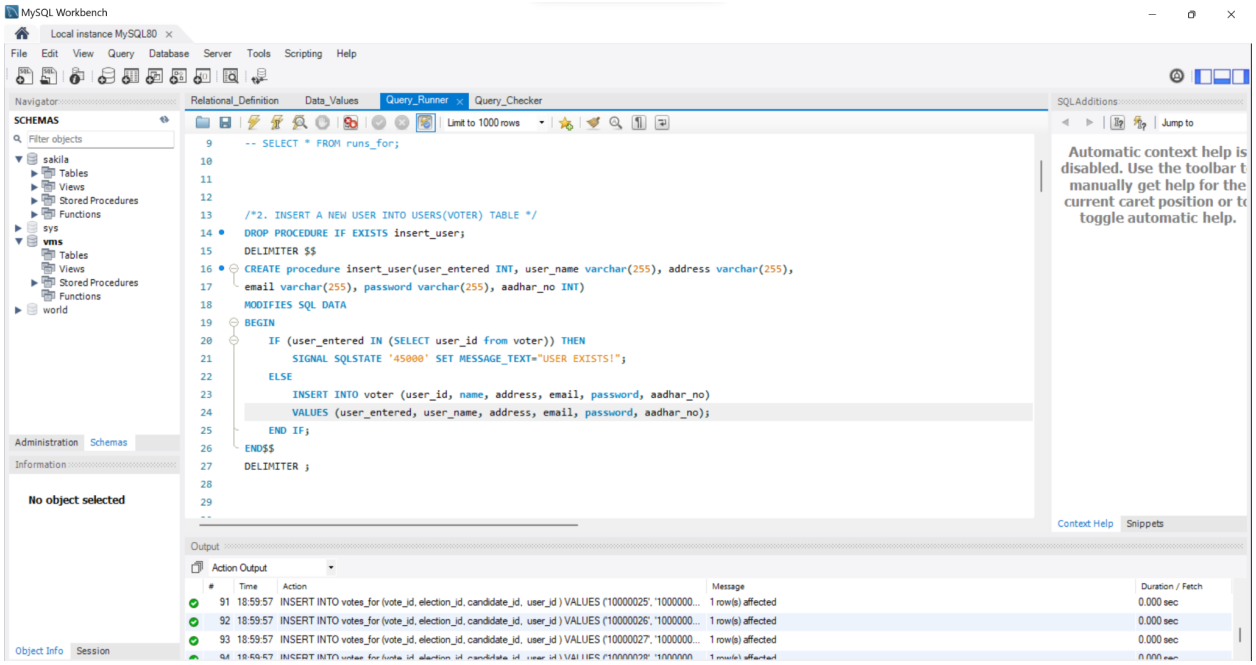

Relational_Definition
Data_Values
Query_Runner
Query_Checker
Limit to 1000 rows
CREATE TABLE IF NOT EXISTS votes_for
(
vote_id INT AUTO_INCREMENT,
election_id INT,
candidate_id INT,
user_id INT,
PRIMARY KEY(vote_id),
CONSTRAINT FK_VFU FOREIGN KEY (user_id) REFERENCES voter(user_id) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FK_VFE FOREIGN KEY (election_id) REFERENCES election(election_id) ON DELETE CASCADE ON UPD
CONSTRAINT FK_VFC FOREIGN KEY (candidate_id) REFERENCES candidate(candidate_id) ON DELETE CASCADE ON
UNIQUE(election_id, user_id)
);
ALTER TABLE votes_for AUTO_INCREMENT=10000001;
describe votes_for;
CREATE TABLE IF NOT EXISTS administrator
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
Field Type Null Key Default Extra
vote_id int NO PRI NULL auto_increment
election_id int YES MUL NULL
candidate_id int YES MUL NULL
user_id int YES MUL NULL
Result 1
Result 2
Result 3
Result 4
Result 5
Result 6
Result 7
Read Only

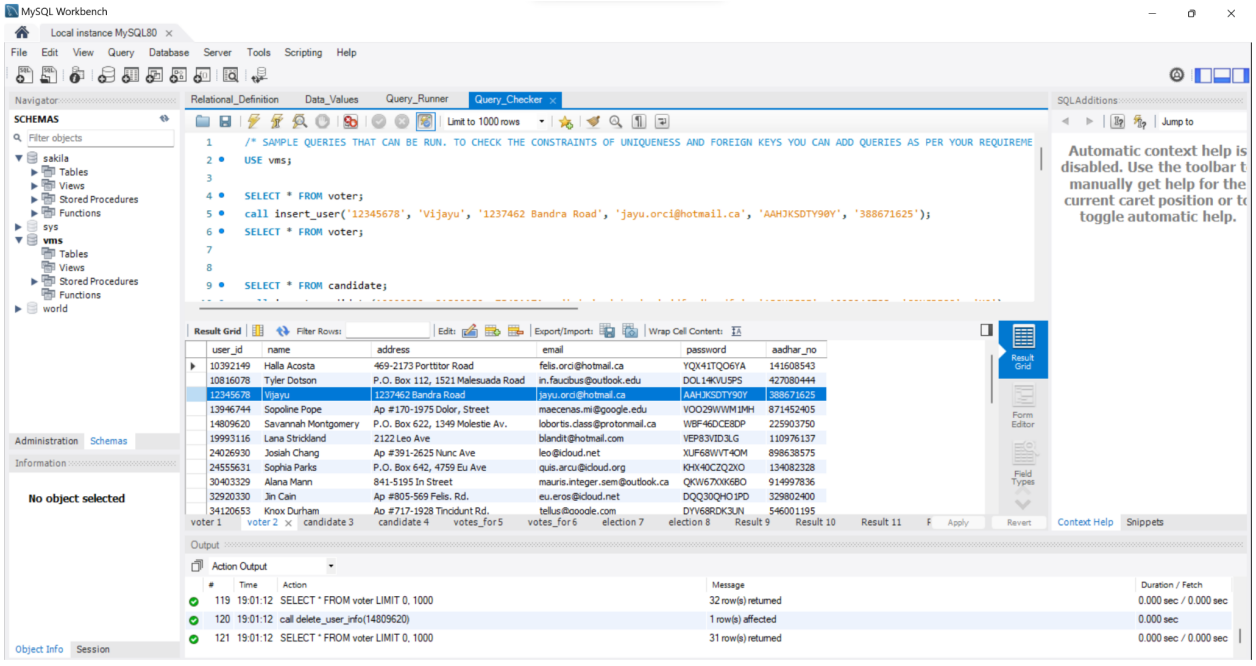

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
SCHEMAS
sakila
vms
world
Relational_Definition
Data_Values
Query_Runner
Query_Checker
ALTER TABLE votes_for AUTO_INCREMENT=10000001;
describe votes_for;
CREATE TABLE IF NOT EXISTS administrator
admin_id INT,
admin_password VARCHAR(255) NOT NULL DEFAULT ' ',
user_id INT,
PRIMARY KEY(admin_id),
CONSTRAINT FK_AV FOREIGN KEY(user_id) REFERENCES voter(user_id) ON DELETE CASCADE ON UPDATE CASCADE ,
UNIQUE (admin_id, user_id)
);
describe administrator;
CREATE TABLE IF NOT EXISTS manages
No object selected
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Field Type Null Key Default Extra
admin_id int NO PRI
admin_password varchar(255) NO
user_id int YES MUL
Result 5
Read Only
Context Help
Snippets
Output


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sakila
Tables
Views
Stored Procedures
Functions
sys
vms
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
No object selected
Object Info Session
Relational_Definition Data_Values Query_Runner Query_Checker
Limit to 1000 rows
UNIQUE (admin_id, user_id)
);
describe administrator;
CREATE TABLE IF NOT EXISTS manages
(
admin_id INT,
election_id INT,
CONSTRAINT FK_MA FOREIGN KEY(election_id) REFERENCES election(election_id) ON DELETE CASCADE ON UPDATI
CONSTRAINT FK_ME FOREIGN KEY(admin_id) REFERENCES administrator(admin_id) ON DELETE CASCADE ON UPDATE
);
describe manages;
CREATE TABLE IF NOT EXISTS runs_for
(
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
admin_id int YES MUL NULL
election_id int YES MUL NULL
Result 1 Result 2 Result 3 Result 4 Result 5 Result 6 Result 7
Read Only
Output
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets

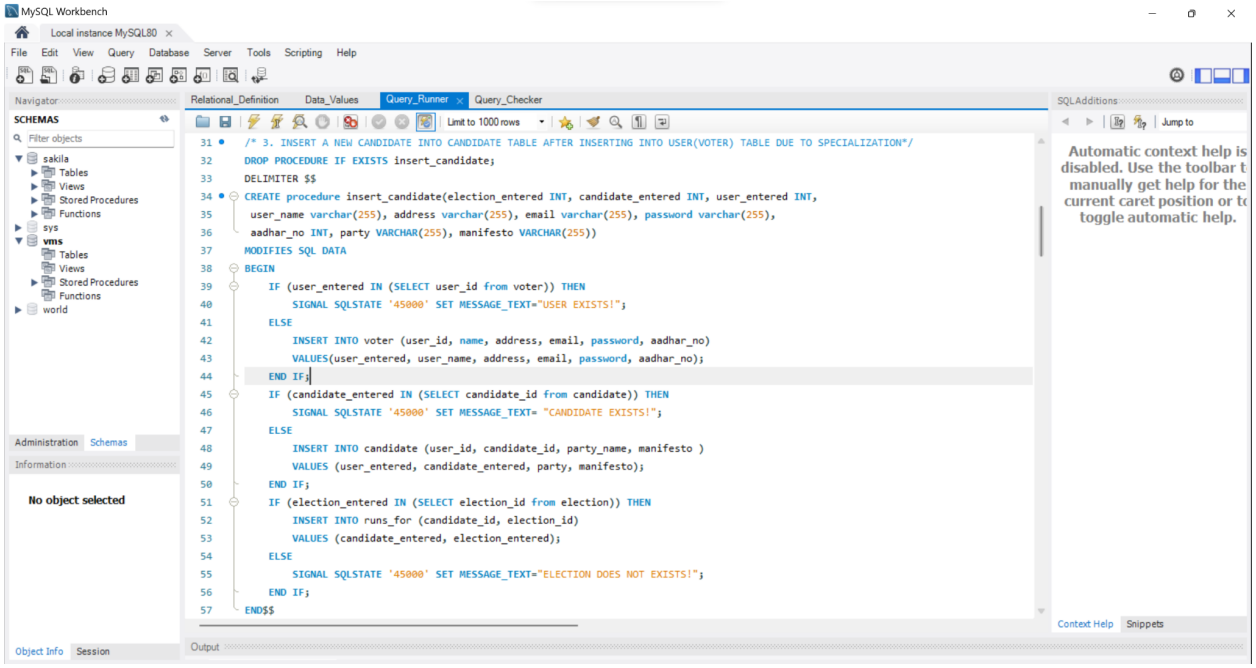

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sakila
Tables
Views
Stored Procedures
Functions
sys
vms
Tables
Views
Stored Procedures
Functions
world
Administration Schemas
Information
No object selected
Object Info Session
Relational_Definition Data_Values Query_Runner Query_Checker
Limit to 1000 rows
election_id INT,
CONSTRAINT FK_MA FOREIGN KEY(election_id) REFERENCES election(election_id) ON DELETE CASCADE ON UPDATI
CONSTRAINT FK_ME FOREIGN KEY(admin_id) REFERENCES administrator(admin_id) ON DELETE CASCADE ON UPDATE
);
describe manages;
CREATE TABLE IF NOT EXISTS runs_for
(
election_id INT,
candidate_id INT,
CONSTRAINT FK_RE FOREIGN KEY(election_id) REFERENCES election(election_id) ON DELETE CASCADE ON UPDATI
CONSTRAINT FK_RC FOREIGN KEY(candidate_id) REFERENCES candidate(candidate_id) ON DELETE CASCADE ON UPI
);
describe runs_for;
Result Grid Filter Rows: Export: Wrap Cell Content:
Field Type Null Key Default Extra
election_id int YES MUL NULL
candidate_id int YES MUL NULL
Result 1 Result 2 Result 3 Result 4 Result 5 Result 6 Result 7
Read Only
Output
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets

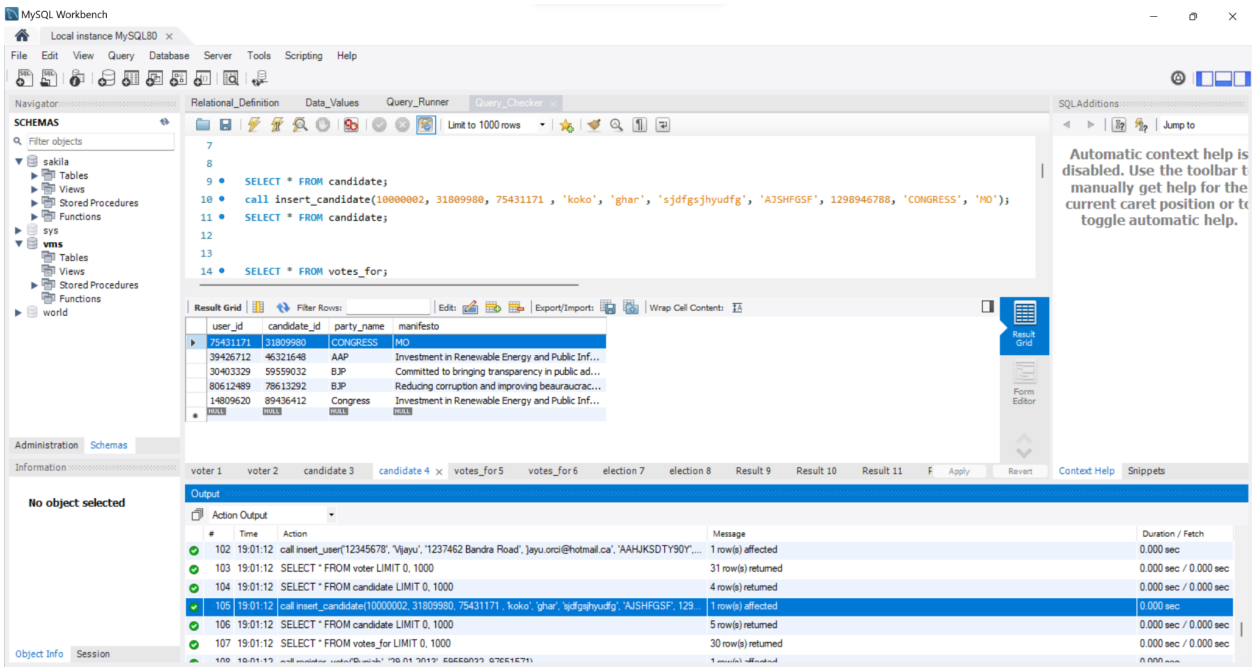
5.2. Insert User



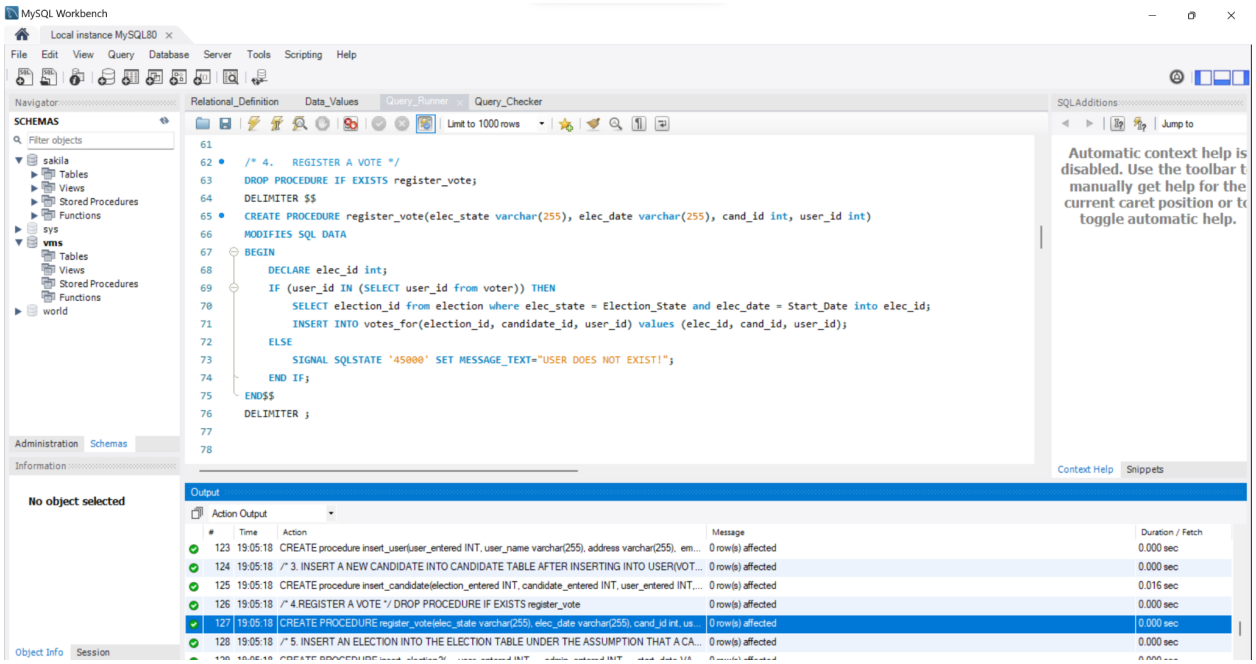


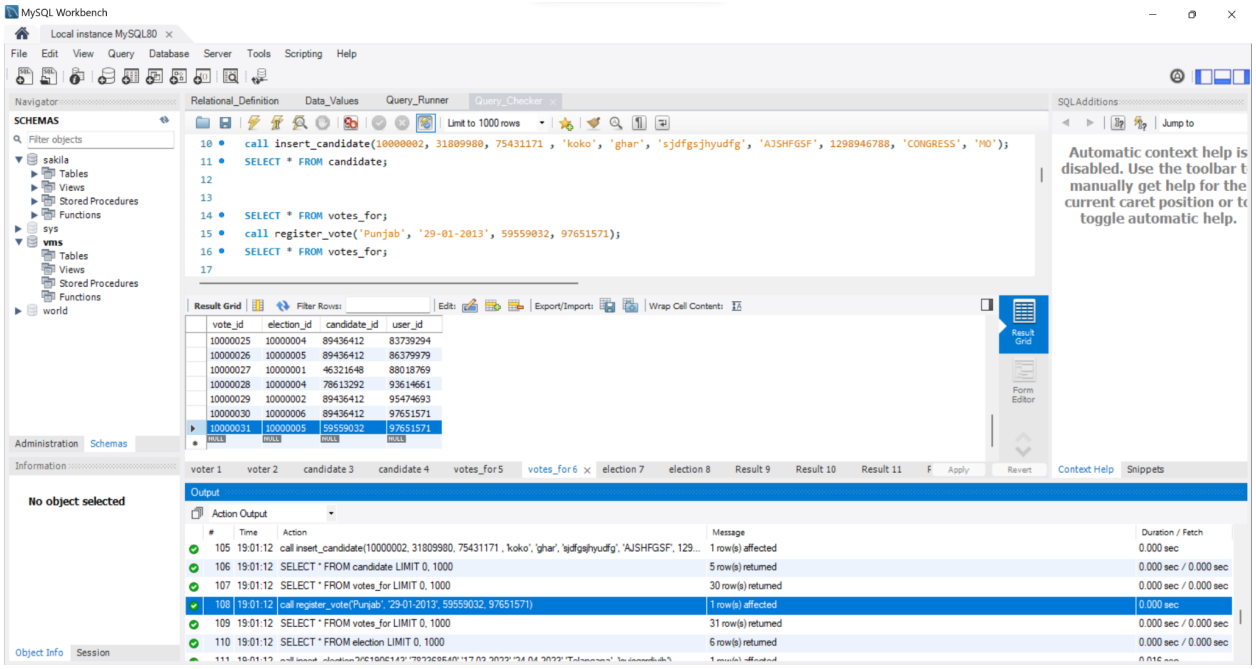
5.3. Insert Candidate



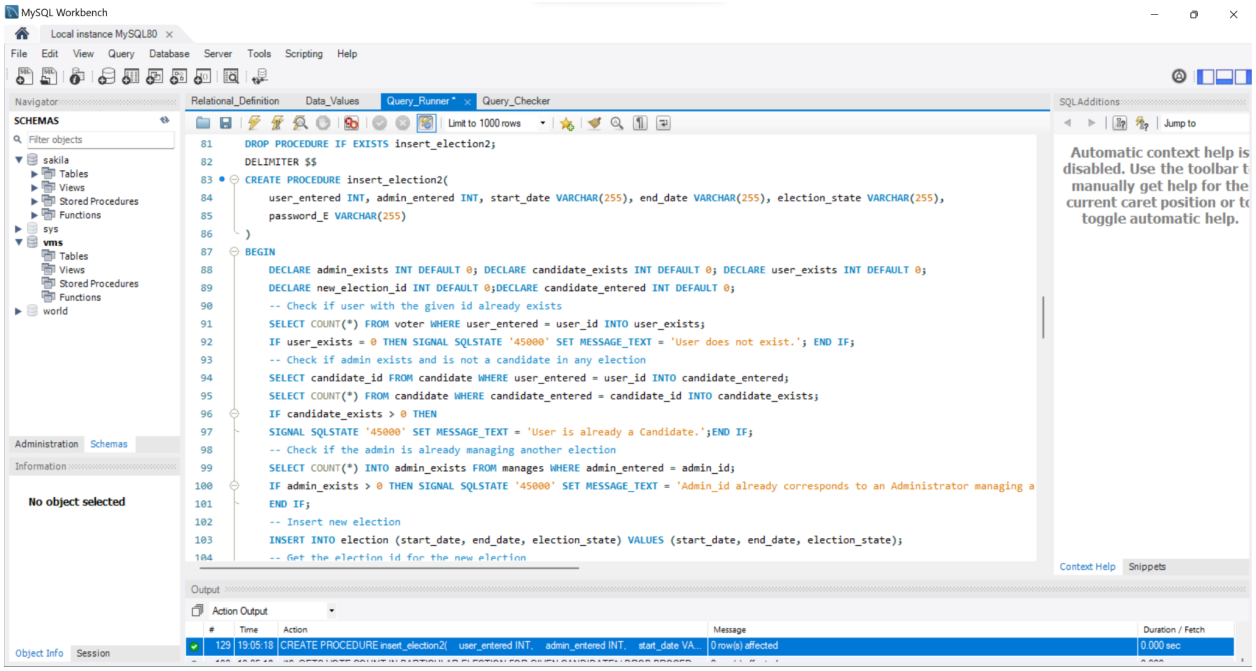


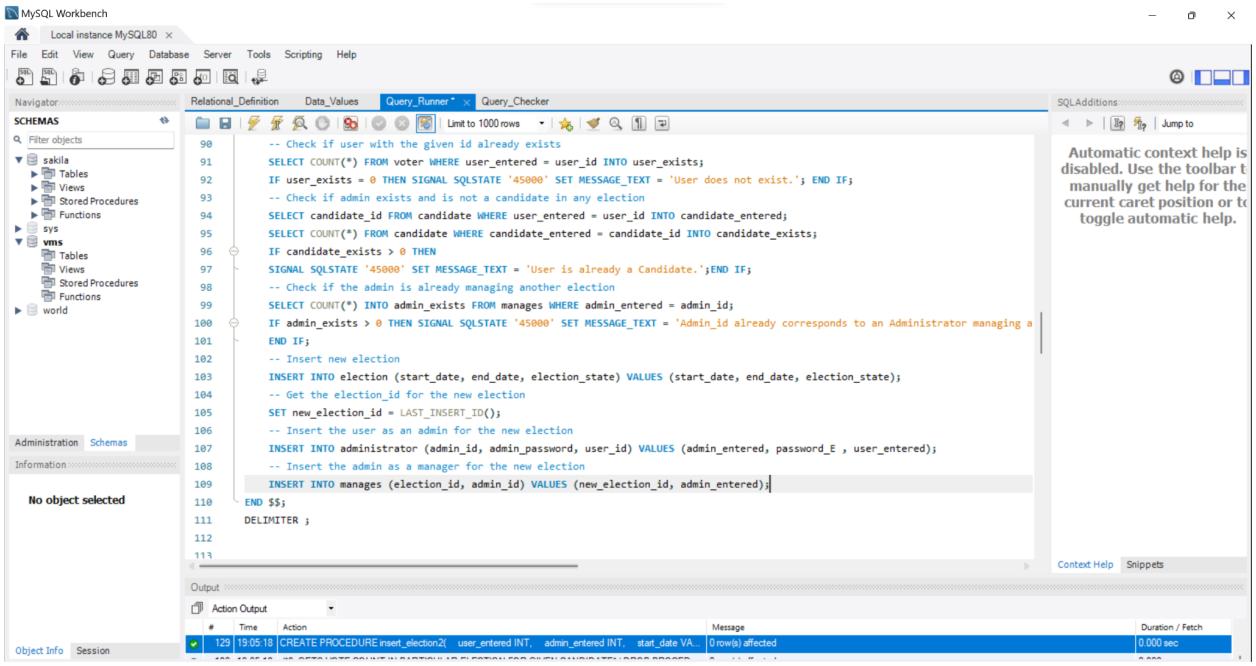
5.4. Register a vote - register_vote

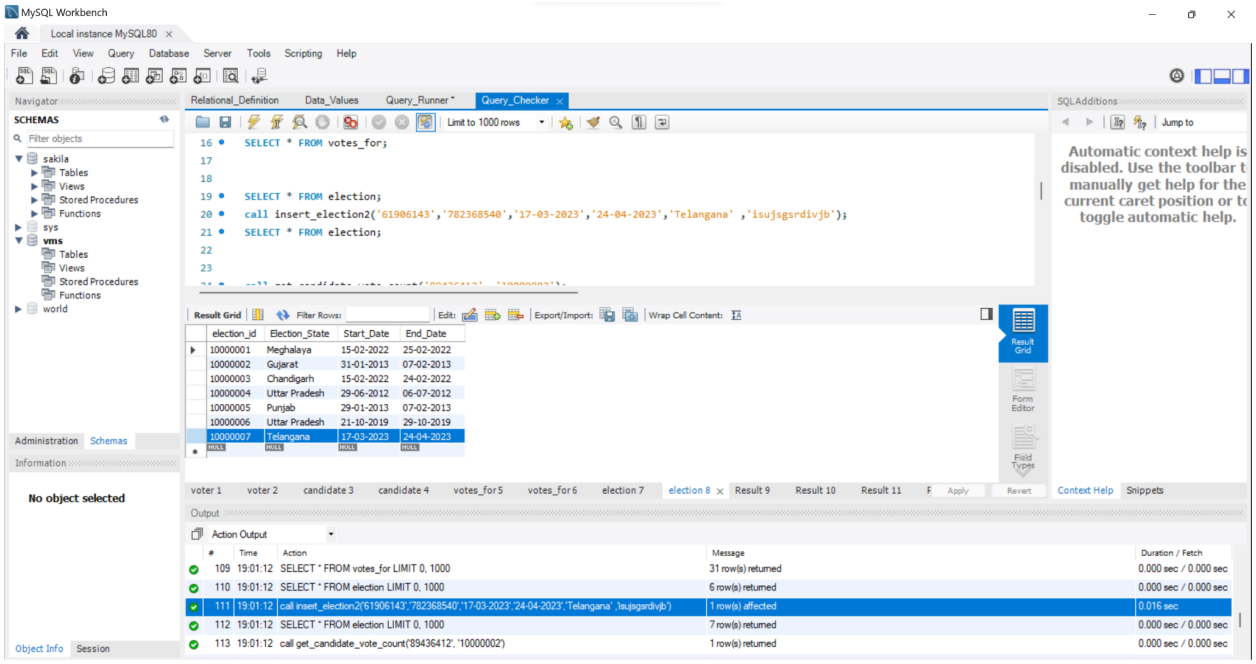




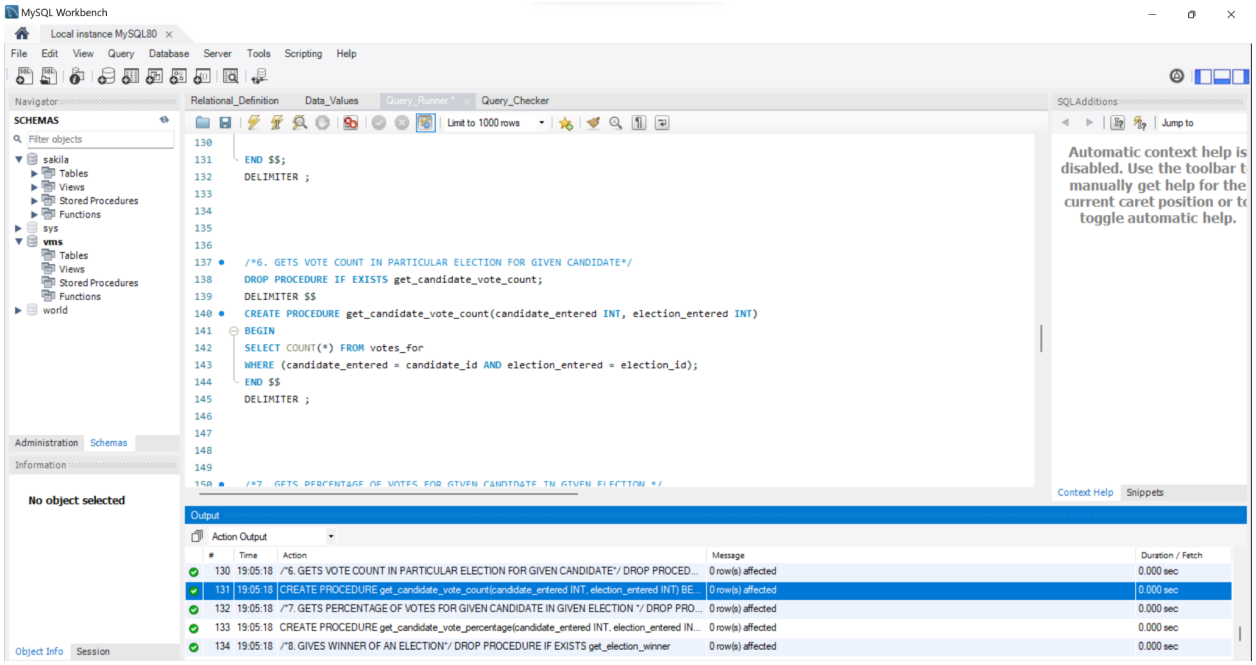
5.5. Insert election - insert_election

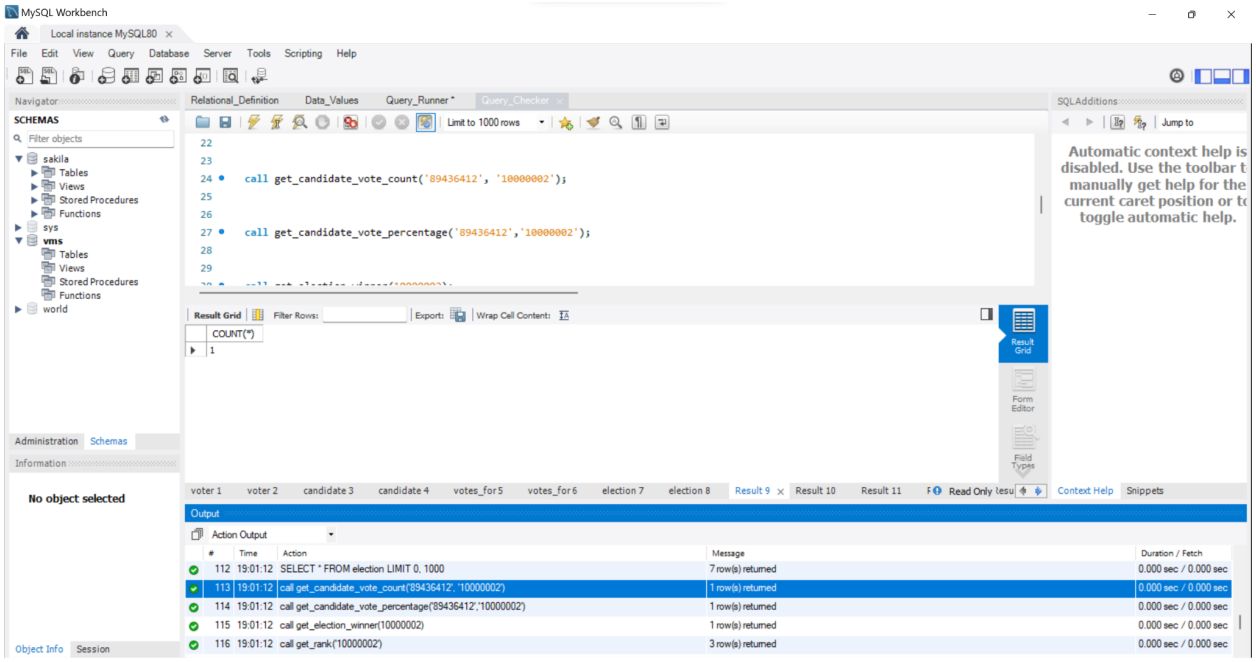




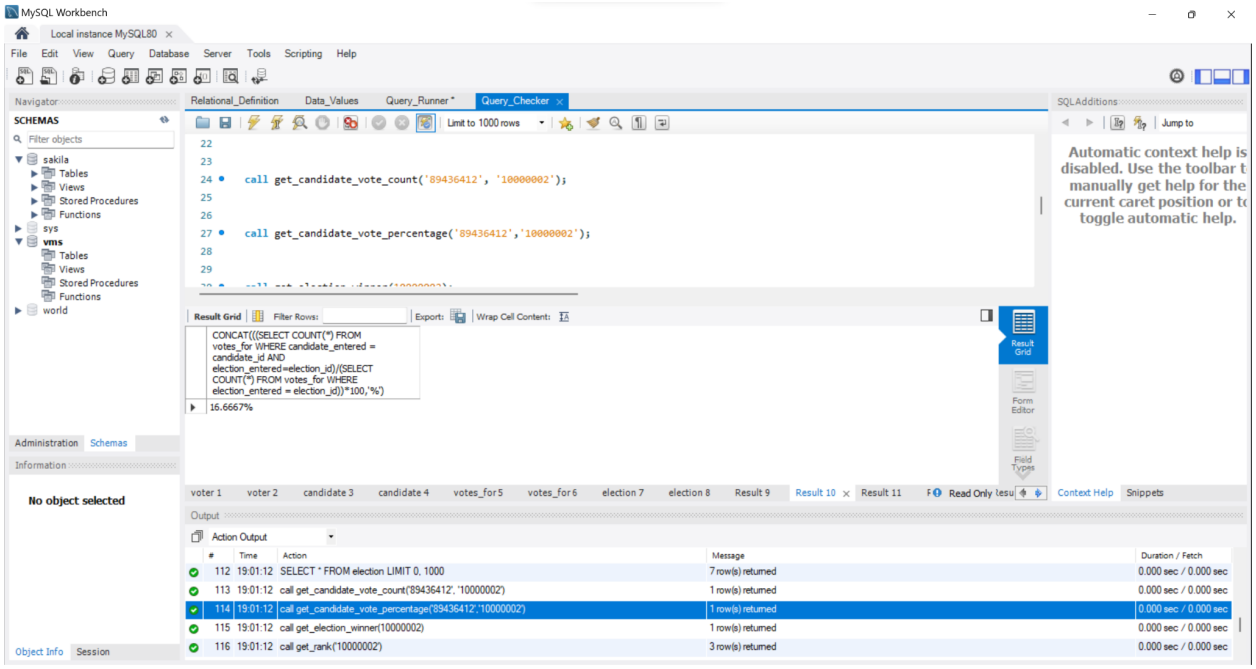


5.6. Retrieve no. of votes for Candidate (in given election)

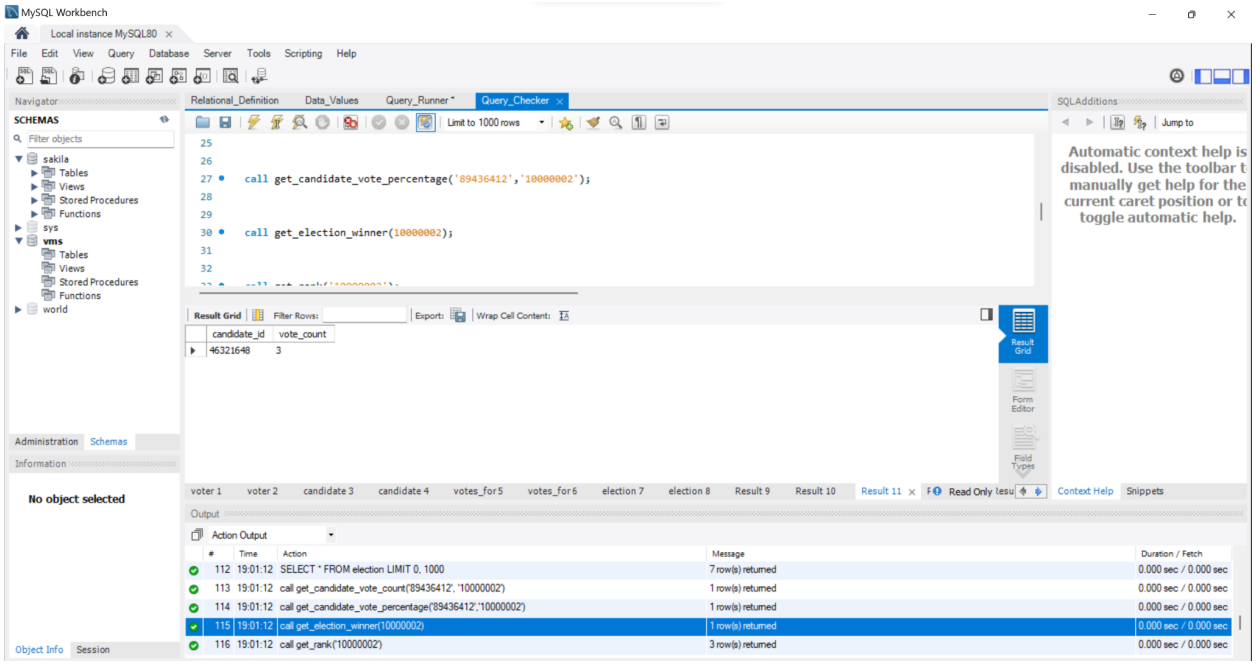




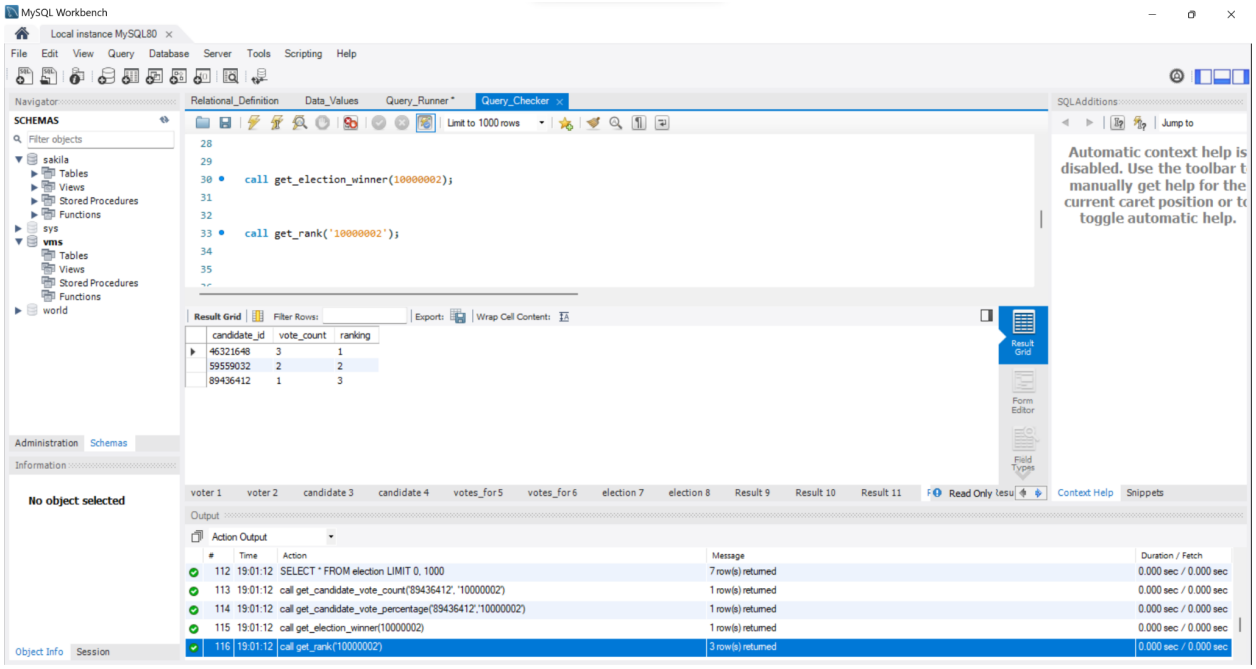
4.7. Retrieve the percentage of votes for a candidate.



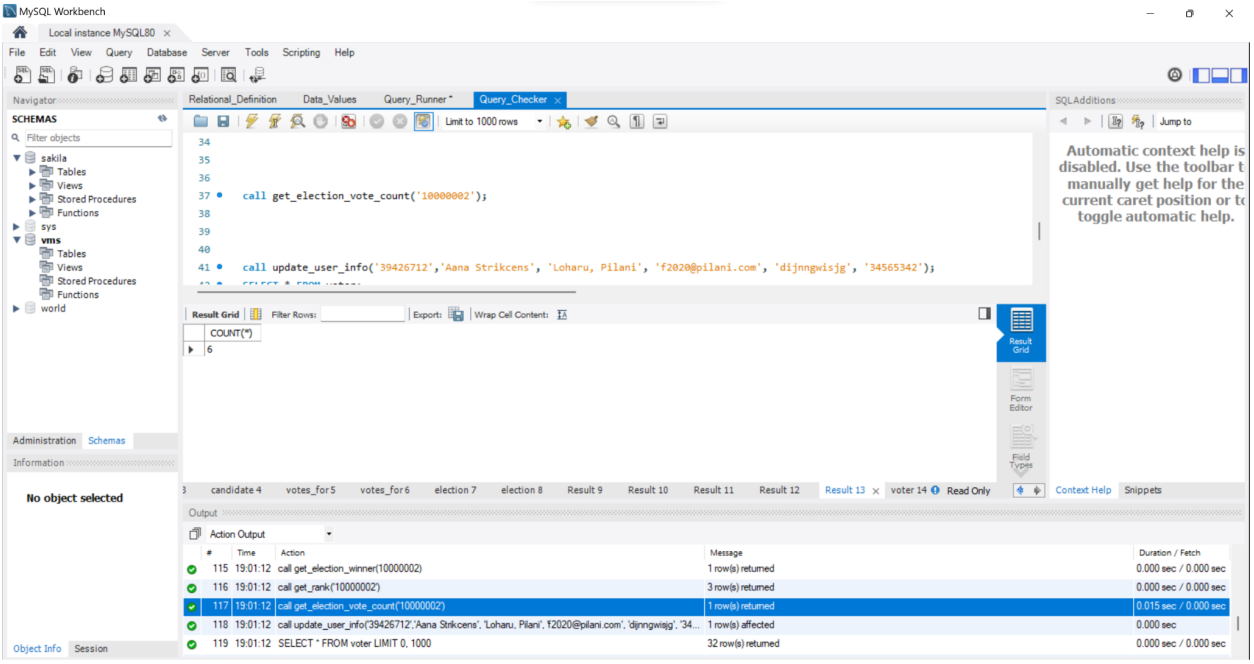
5.8. Retrieve the winner of the election.



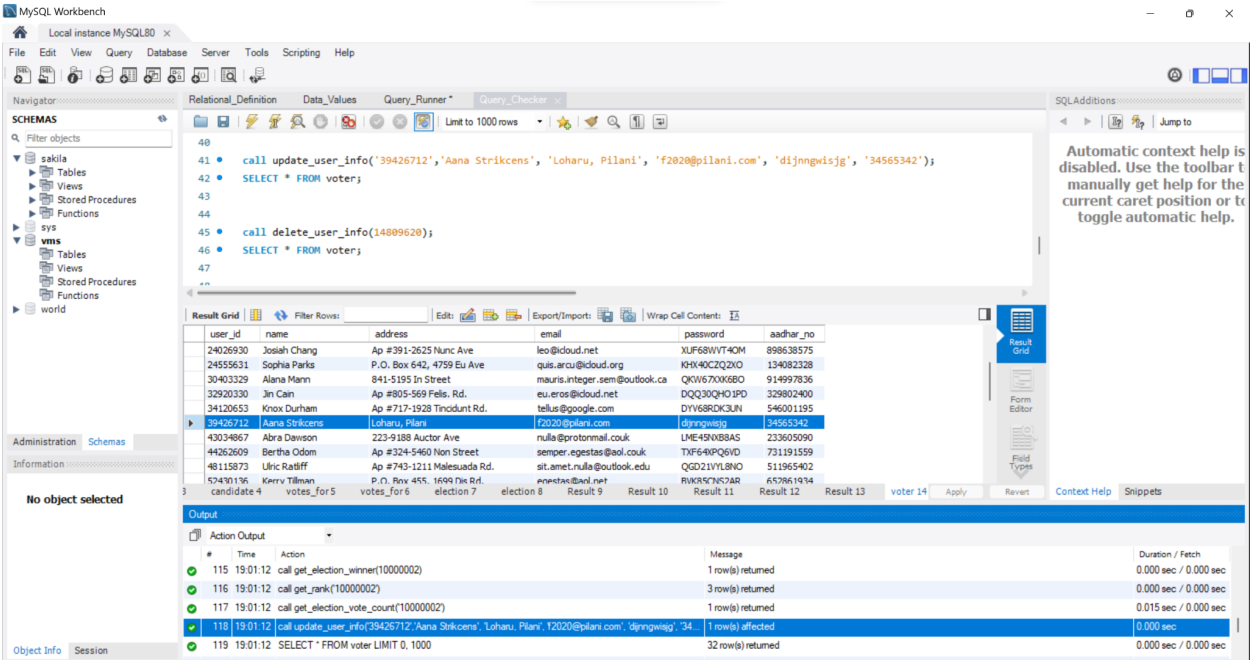
5.9. Retrieve the rank of each candidate based on the number of votes received.



5.10. Retrieve the total number of votes cast for an election.

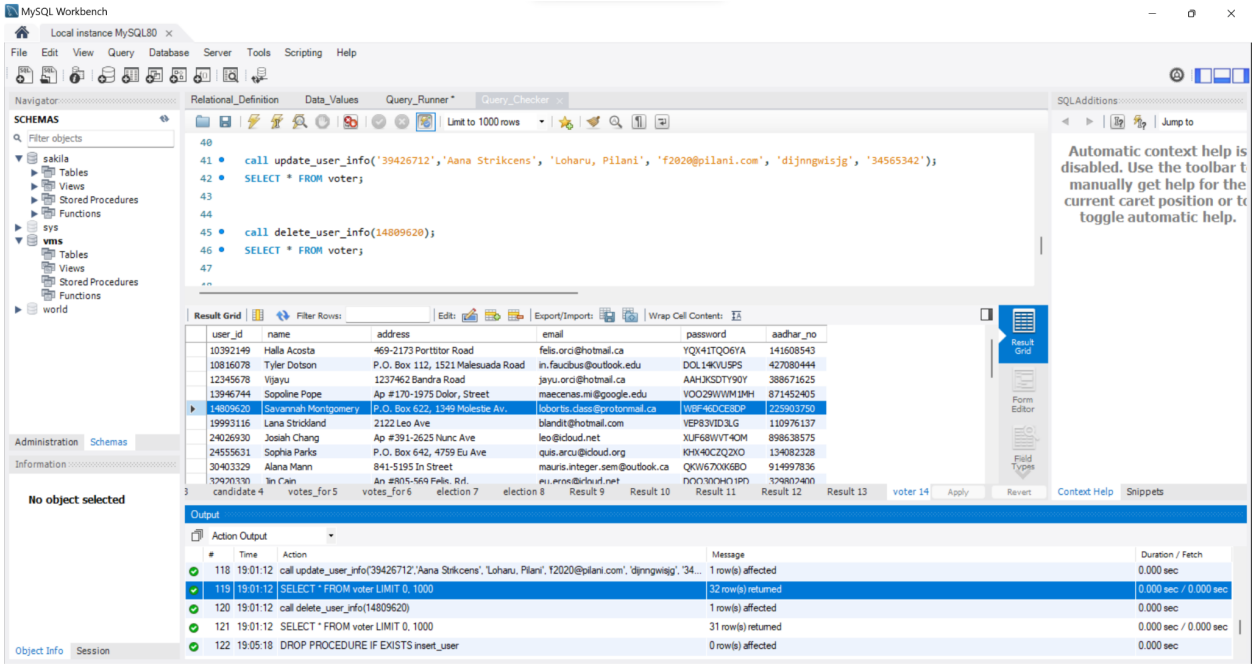


5.11. Update the user's information.

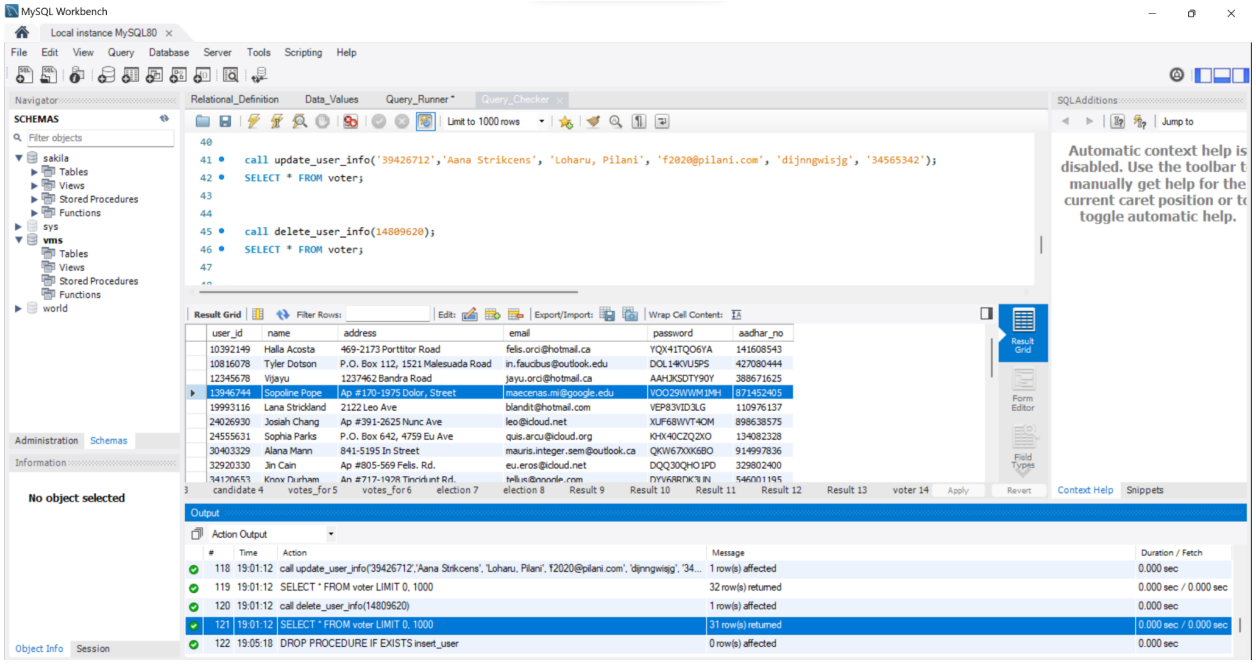


5.12. Delete a specific user's information.

Before Delete



After Delete



(Savanna Montogomery's entry is missing from the table, as it is deleted)

Links for Video

CHAITANYA SETHI - 2020B3A71961P
https://drive.google.com/file/d/1EKmC-feXqt7ngMpqBk3CrVRMz8PQY0KP/view?usp=sharing

**Note: In the end, the candidate_vote_percentage is different as one candidate was earlier deleted by the delete user query when I initially had run the code once. Hence, only 5 votes remained, out of which the winner had 3, leading to a 60% percentage. The repeating values issue has been corrected in the uploaded code, please check.

STAVYA PURI - 2020B5A70912P
https://drive.google.com/file/d/1dRX-kY-PgfdpJwKYeVPauehYHpN6Vy_W/view?usp=share_link