Subject Name: **DBMS**

Subject Code: **24CSE0209**

Cluster: **iBETA**

Department: **CSE**

Project Report File

Submitted By:

S. No.	Name	Roll No.	Contact
1	Ananya Mahajan	2410990919	8492958192
2	Amisha Jindal	2410990917	9878974293
3	Harshita Goyal	2410990338	6284154972

SmartVote: Secure Online Voting Platform

PROJECT REPORT

Table of Contents

1. Introduction

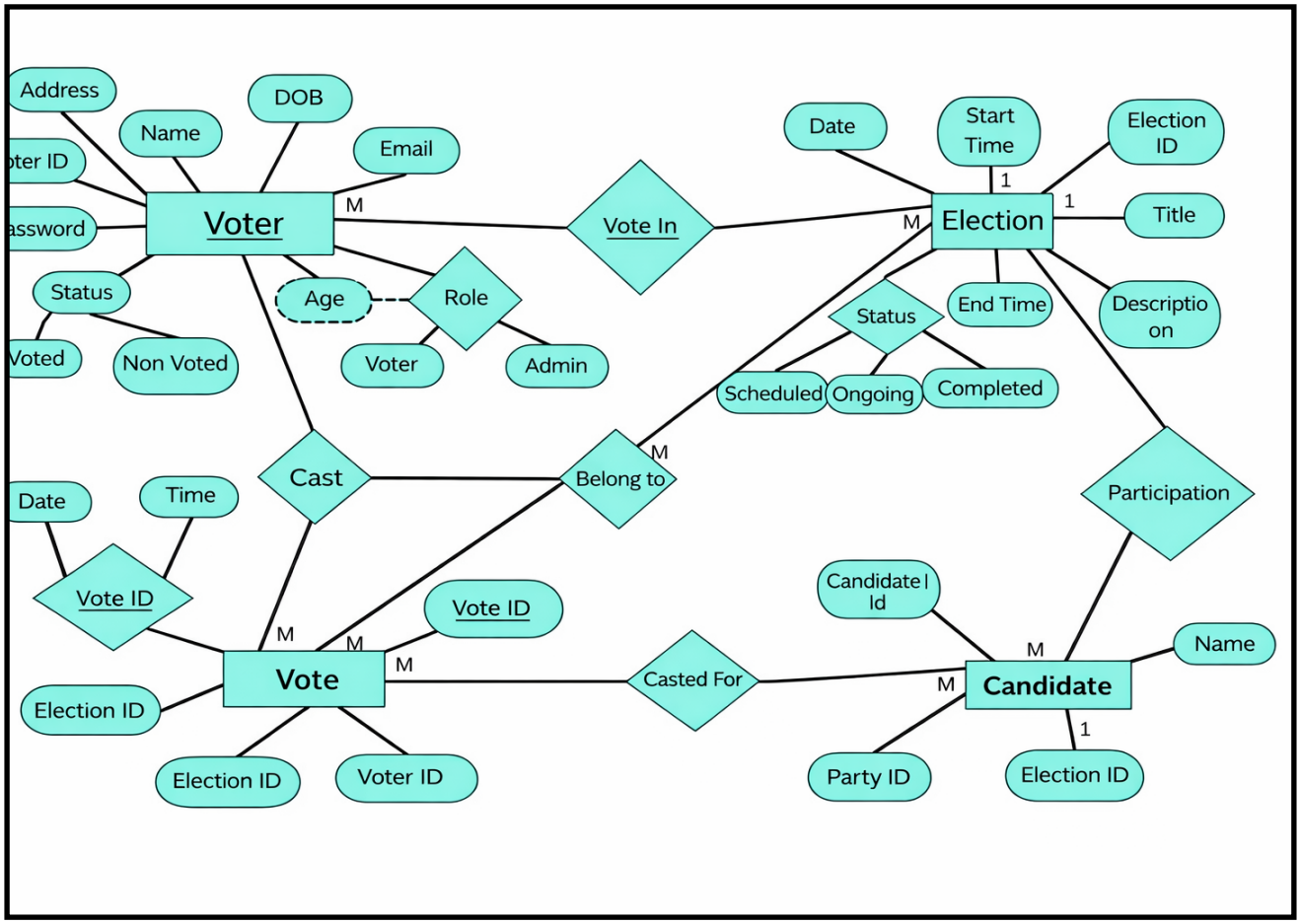
SmartVote is a modern, blockchain-inspired database solution designed to manage secure online elections. Traditional voting methods often face challenges such as vote tampering, lack of transparency, and inefficient result processing. This project addresses these issues by implementing a robust relational database that ensures data integrity, real-time result calculation, and comprehensive audit logging.

The system allows Administrators to manage elections and candidates, while Voters can cast secure votes. A unique feature of this implementation is the **"Operation Locks & Logs"** system, which utilizes database triggers to record every insertion, update, or deletion, ensuring a tamper-proof audit trail for Viva demonstration.

2. E-R Diagram Correction & Logic

We have refined the original ER Model to ensure strict data integrity:

1. **Voter to Vote (1:N):** A voter can cast multiple votes *over time* (in different elections), but logically only **one vote per election**. We enforce this using a composite unique constraint in the Schema, correcting the ambiguity in simple 1:N diagrams.
2. **Election to Candidate (1:N):** An election can have multiple candidates, but a candidate entity is specific to one election instance.
3. **Audit System (New Entity):** We added a Operation_Locks_Log entity that is not connected via direct relationships but monitors all other entities via Triggers.



3.Normalization

- Normalization is a systematic process in DBMS used to organize data in a database efficiently.
- It minimizes data redundancy, avoids update anomalies, and ensures data integrity.
- The process involves dividing large tables into smaller, well-structured tables and establishing proper relationships using keys.

Why Normalization is Important in SmartVote?

- Prevents duplicate voter, election, and candidate data
- Ensures one vote per voter per election
- Maintains accuracy and consistency in election records
- Makes the database secure, scalable, and easy to maintain
- Supports transparent auditing and logging

4. Database Schema

The database is normalized to 3NF and implemented in MySQL.

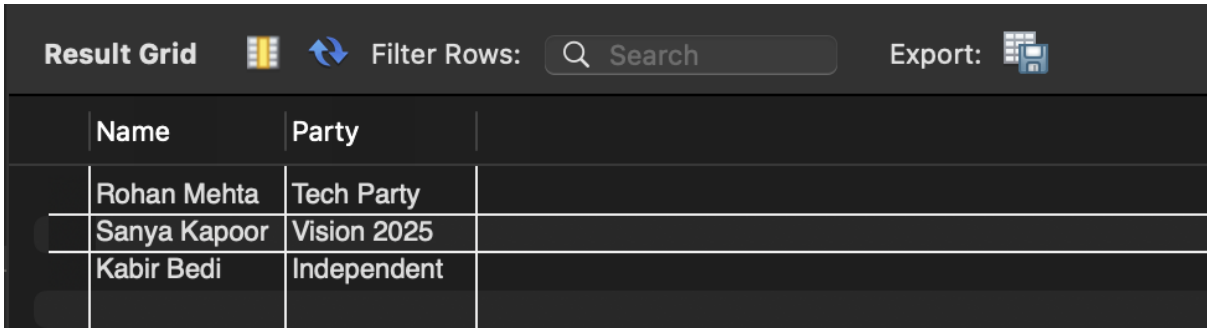
- Voter (VoterID [PK], Name, Email, Role, Status, PasswordHash)
- Election (ElectionID [PK], Title, Description, StartTime, EndTime, Status)
- Candidate (CandidateID [PK], Name, Party, ElectionID [FK])
- Vote (VoteID [PK], VoterID [FK], ElectionID [FK], CandidateID [FK], Timestamp)
    - Constraint: Unique Key (VoterID, ElectionID) — Ensures a voter cannot vote twice in the same election.
- Operation_Locks_Log (LogID [PK], TableName, OperationType, OperationDate, Details) — Stores the audit trail.

5. Use Cases (SQL Implementation)

Below are the 12 complex Use Cases derived from the original Relational Algebra requirements, converted to executable SQL.

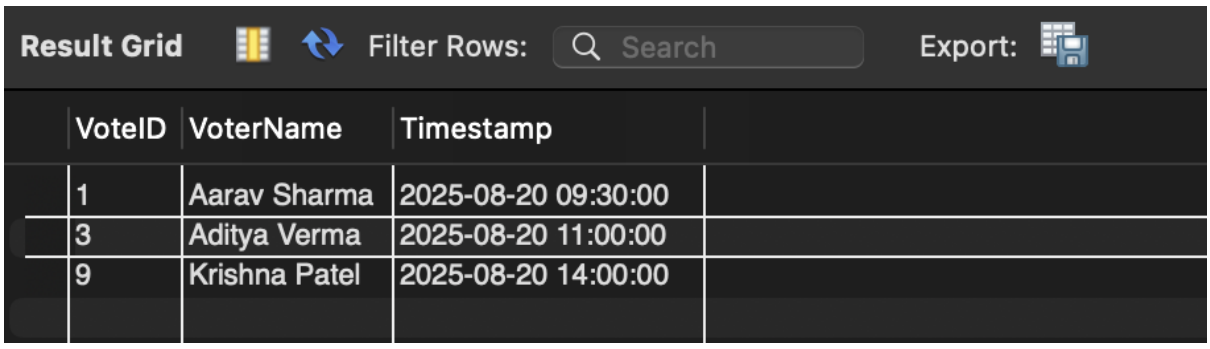
Use Case 1: List Names of Candidates in Election "E101"

SQL
```
SELECT C.Name, C.Party
FROM Candidate C
JOIN Election E ON C.ElectionID = E.ElectionID
WHERE E.ElectionID = 'E101';
```



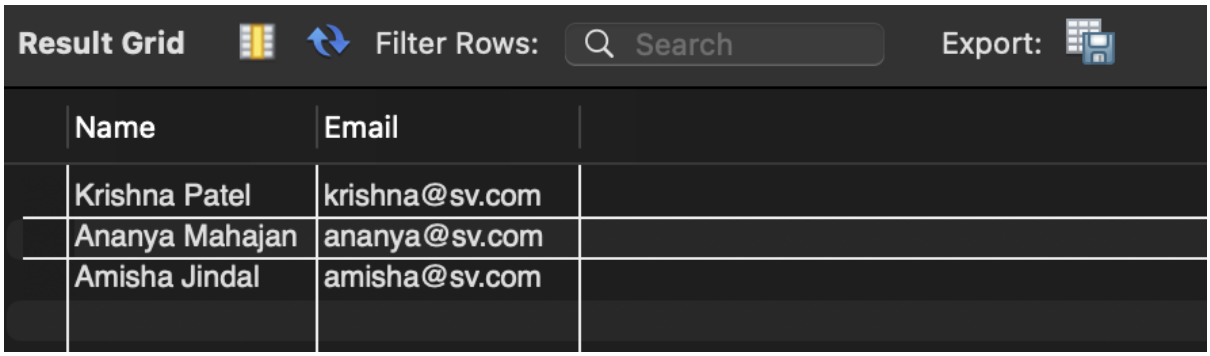
Use Case 2: Find All Votes Cast for Candidate "C201"

SQL
```
SELECT V.VoteID, Vr.Name AS VoterName, V.Timestamp
FROM Vote V
JOIN Voter Vr ON V.VoterID = Vr.VoterID
WHERE V.CandidateID = 'C201';
```



Use Case 3: Get Emails of All Admins

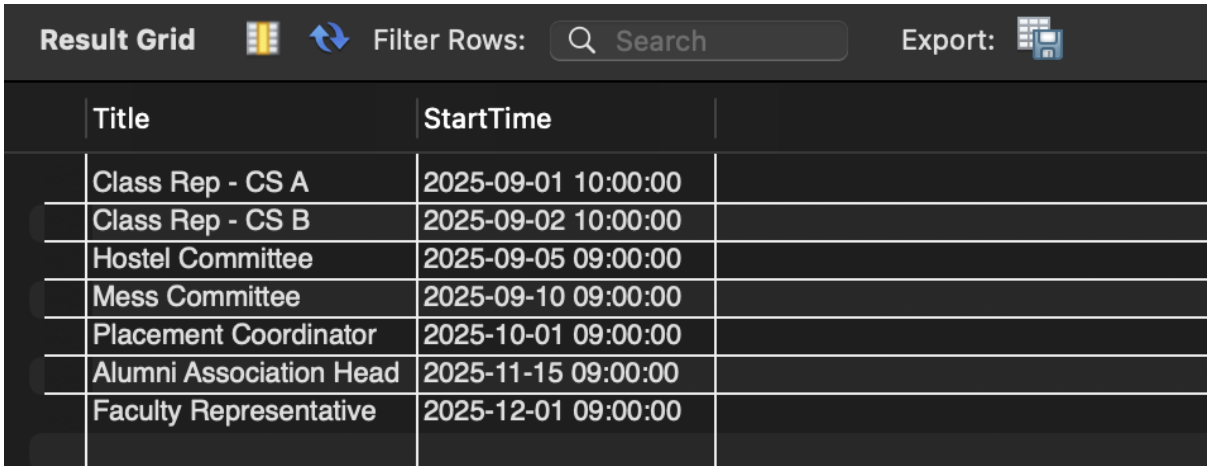
SQL
SELECT Name, Email
FROM Voter
WHERE Role = 'Admin';


Result Grid Filter Rows: Search Export:

Name	Email
Krishna Patel	krishna@sv.com
Ananya Mahajan	ananya@sv.com
Amisha Jindal	amisha@sv.com

Use Case 4: Find Elections Scheduled After Sept 1st, 2025

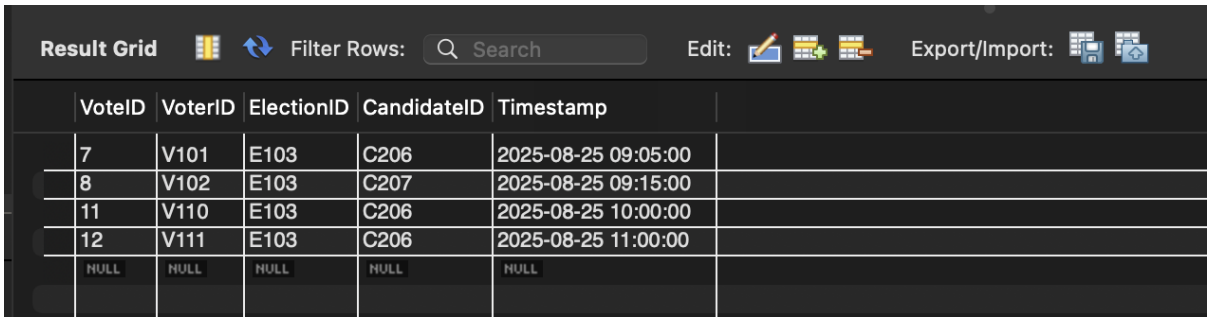
SQL
SELECT Title, StartTime
FROM Election
WHERE StartTime > '2025-09-01';


Result Grid Filter Rows: Search Export:

Title	StartTime
Class Rep - CS A	2025-09-01 10:00:00
Class Rep - CS B	2025-09-02 10:00:00
Hostel Committee	2025-09-05 09:00:00
Mess Committee	2025-09-10 09:00:00
Placement Coordinator	2025-10-01 09:00:00
Alumni Association Head	2025-11-15 09:00:00
Faculty Representative	2025-12-01 09:00:00

Use Case 5: List All Votes Cast on a Specific Date (25-Aug-2025)

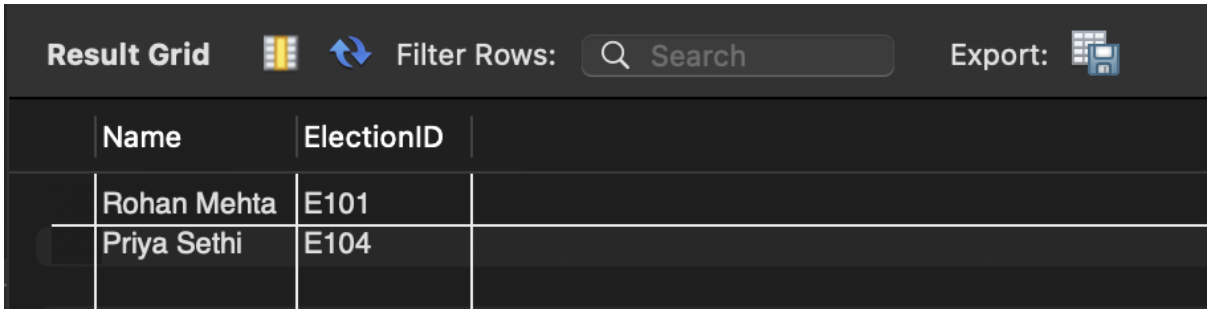
SQL
SELECT * FROM Vote
WHERE DATE(Timestamp) = '2025-08-25';



VoteID	VoterID	ElectionID	CandidateID	Timestamp
7	V101	E103	C206	2025-08-25 09:05:00
8	V102	E103	C207	2025-08-25 09:15:00
11	V110	E103	C206	2025-08-25 10:00:00
12	V111	E103	C206	2025-08-25 11:00:00
NULL	NULL	NULL	NULL	NULL

Use Case 6: Find Candidates Belonging to "Tech Party" (P101)

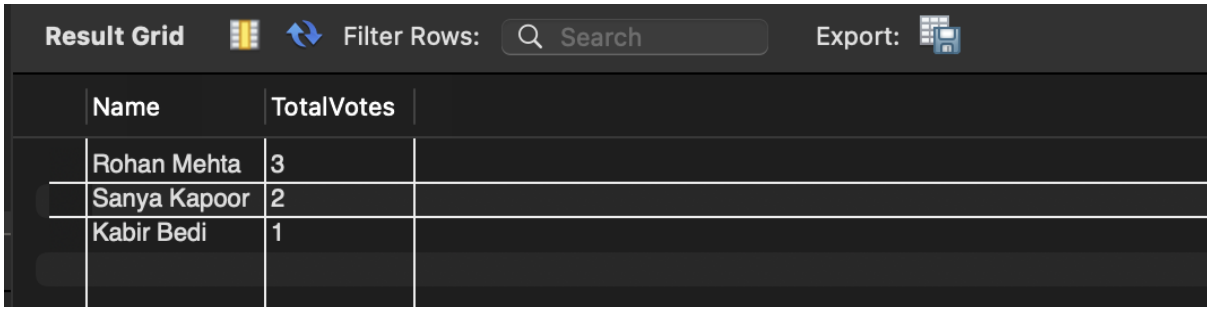
SQL
SELECT Name, ElectionID
FROM Candidate
WHERE Party = 'Tech Party';



Name	ElectionID
Rohan Mehta	E101
Priya Sethi	E104

Use Case 7: Count Votes per Candidate in Election "E101" (Result aggregation)

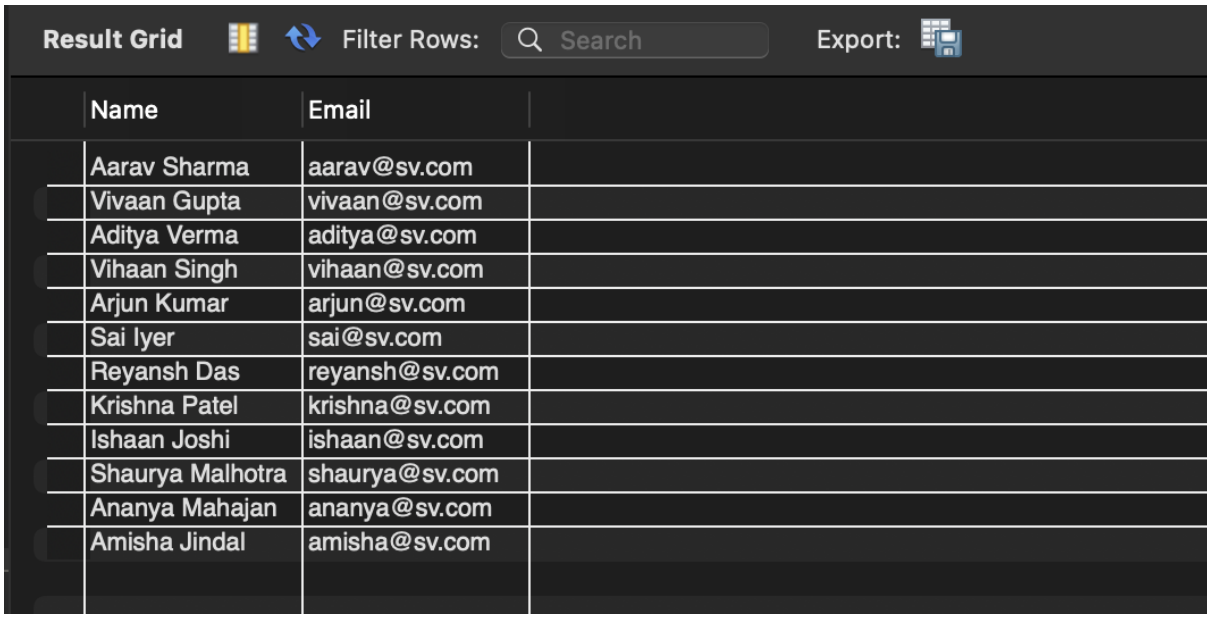
SQL
SELECT C.Name, COUNT(V.VoteID) as TotalVotes
FROM Candidate C
LEFT JOIN Vote V ON C.CandidateID = V.CandidateID
WHERE C.ElectionID = 'E101'
GROUP BY C.CandidateID, C.Name;


Result Grid Filter Rows: Search Export:

Name	TotalVotes
Rohan Mehta	3
Sanya Kapoor	2
Kabir Bedi	1

Use Case 8: Find Voters Who Have Not Voted Yet

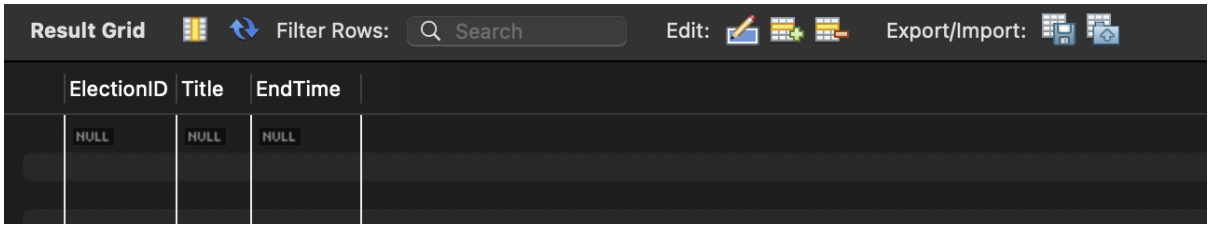
SQL
SELECT Name, Email
FROM Voter
WHERE Status = 'Not Voted';


Result Grid Filter Rows: Search Export:

Name	Email
Aarav Sharma	aarav@sv.com
Vivaan Gupta	vivaan@sv.com
Aditya Verma	aditya@sv.com
Vihaan Singh	vihaan@sv.com
Arjun Kumar	arjun@sv.com
Sai Iyer	sai@sv.com
Reyansh Das	reyansh@sv.com
Krishna Patel	krishna@sv.com
Ishaan Joshi	ishaan@sv.com
Shaurya Malhotra	shaurya@sv.com
Ananya Mahajan	ananya@sv.com
Amisha Jindal	amisha@sv.com

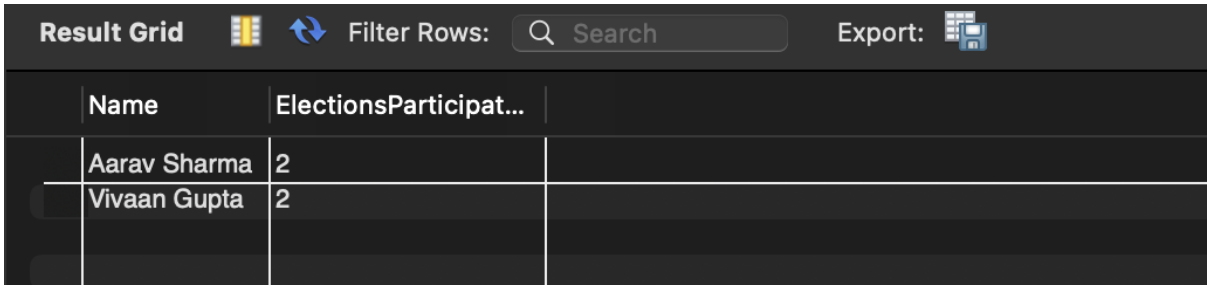
Use Case 9: List Elections with 'Ongoing' Status

SQL
SELECT ElectionID, Title, EndTime
FROM Election
WHERE Status = 'Ongoing';



Use Case 10: Identify Voters Who Voted in Multiple Elections

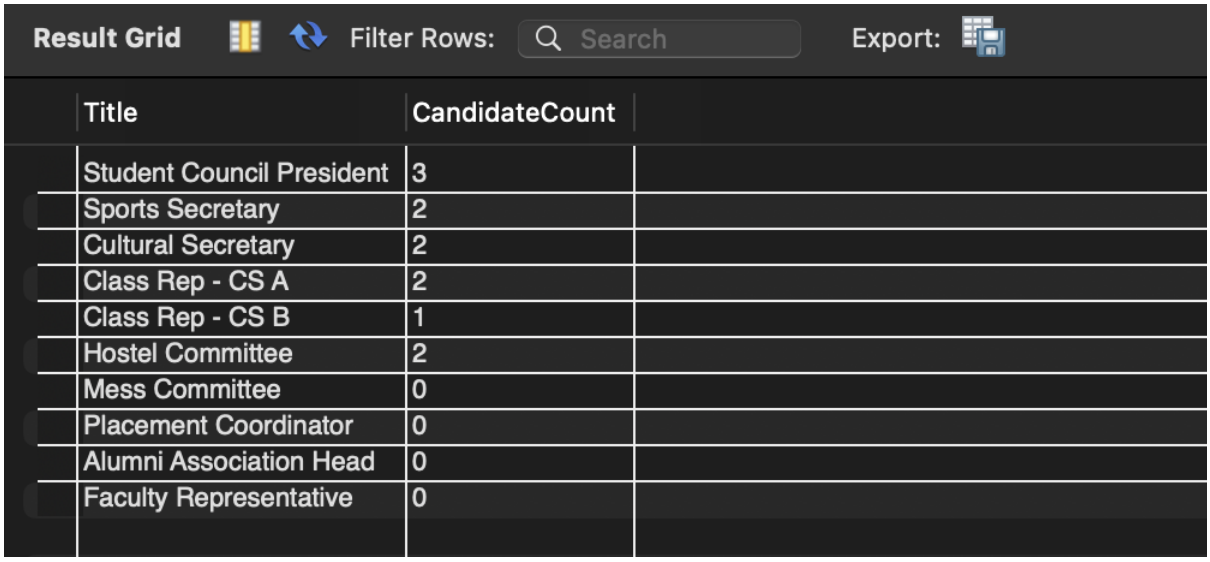
SQL
SELECT Vr.Name, COUNT(DISTINCT V.ElectionID) as ElectionsParticipated
FROM Vote V
JOIN Voter Vr ON V.VoterID = Vr.VoterID
GROUP BY Vr.VoterID, Vr.Name
HAVING ElectionsParticipated > 1;



Use Case 11: Complex Join - Election Details with Candidate Count

SQL

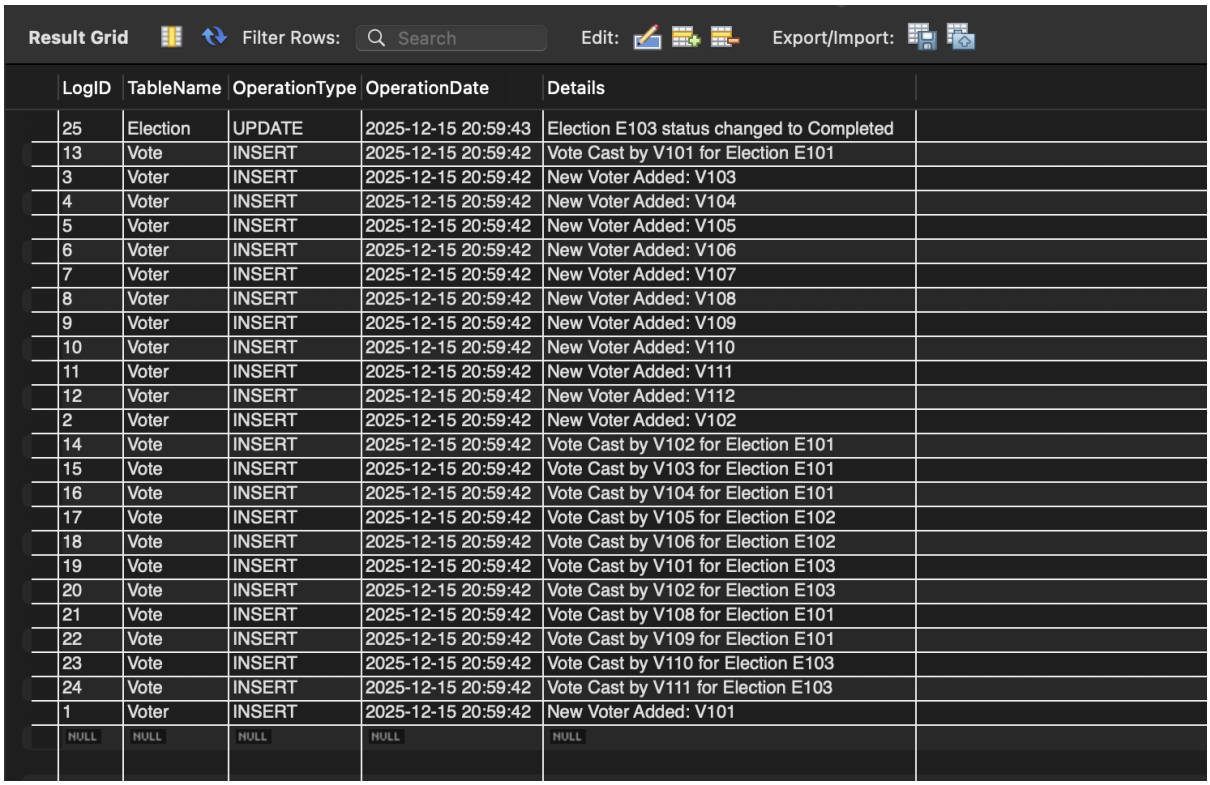
```
SELECT E.Title, COUNT(C.CandidateID) as CandidateCount
FROM Election E
LEFT JOIN Candidate C ON E.ElectionID = C.ElectionID
GROUP BY E.ElectionID, E.Title;
```



Title	CandidateCount
Student Council President	3
Sports Secretary	2
Cultural Secretary	2
Class Rep - CS A	2
Class Rep - CS B	1
Hostel Committee	2
Mess Committee	0
Placement Coordinator	0
Alumni Association Head	0
Faculty Representative	0

Use Case 12: Audit Log Retrieval

SQL
SELECT * FROM Operation_Locks_Log
ORDER BY OperationDate DESC;



Result Grid Filter Rows: Search Edit: Export/Import:

LogID	TableName	OperationType	OperationDate	Details
25	Election	UPDATE	2025-12-15 20:59:43	Election E103 status changed to Completed
13	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V101 for Election E101
3	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V103
4	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V104
5	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V105
6	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V106
7	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V107
8	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V108
9	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V109
10	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V110
11	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V111
12	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V112
2	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V102
14	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V102 for Election E101
15	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V103 for Election E101
16	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V104 for Election E101
17	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V105 for Election E102
18	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V106 for Election E102
19	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V101 for Election E103
20	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V102 for Election E103
21	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V108 for Election E101
22	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V109 for Election E101
23	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V110 for Election E103
24	Vote	INSERT	2025-12-15 20:59:42	Vote Cast by V111 for Election E103
1	Voter	INSERT	2025-12-15 20:59:42	New Voter Added: V101
NULL	NULL	NULL	NULL	NULL

6. SQL & PL/SQL Implementation

This section describes the SQL and PL/SQL implementation of the **SmartVote: Secure Online Voting Platform**.
The database is implemented using **MySQL**, and procedural logic is handled using **PL/SQL constructs such as Triggers, Stored Procedures, and Functions**.

6.1 SQL Implementation

Database Selection

```
CREATE DATABASE SmartVoteDB;
USE SmartVoteDB;
```

Sample Data Insertion

Insert Data into Voter Table

```
INSERT INTO Voter (VoterID, Name, Email, Role, Status, PasswordHash)
VALUES
('V101', 'Ananya Mahajan', 'ananya@gmail.com', 'Voter', 'Not Voted', 'hash123'),
('V102', 'Amisha Jindal', 'amisha@gmail.com', 'Voter', 'Voted', 'hash456'),
('A101', 'Admin User', 'admin@smartvote.com', 'Admin', 'Voted', ‘adminhash');
```

Insert Data into Election Table

```
INSERT INTO Election (ElectionID, Title, Description, StartTime, EndTime, Status)
VALUES
('E101', 'Student Council Election', 'Election for Student Council Members',
 '2025-08-20 09:00:00', '2025-08-20 18:00:00', 'Completed'),
('E102', 'Department Head Election', 'Election for Department Head',
 '2025-09-10 09:00:00', '2025-09-10 17:00:00', ‘Scheduled');
```

Insert Data into Candidate Table

INSERT INTO Candidate (CandidateID, Name, Party, ElectionID)
VALUES
('C201', 'Rohan Verma', 'Tech Party', 'E101'),
('C202', 'Neha Sharma', 'Innovation Party', 'E101'),
('C203', 'Amit Singh', 'Tech Party', ‘E102');

Insert Data into Vote Table

INSERT INTO Vote (VoteID, VoterID, ElectionID, CandidateID, Timestamp)
VALUES
(1, 'V101', 'E101', 'C201', '2025-08-20 10:30:00'),
(2, 'V102', 'E101', 'C202', '2025-08-20 11:15:00’);

Verification Queries

SELECT * FROM Voter;
SELECT * FROM Election;
SELECT * FROM Candidate;
SELECT * FROM Vote;

6.2 PL/SQL Implementation

Audit / Logs Table

To maintain a **tamper-proof audit trail**, an operation log table is created.

```
CREATE TABLE Operation_Locks_Log (
   LogID INT AUTO_INCREMENT PRIMARY KEY,
   TableName VARCHAR(50),
   OperationType VARCHAR(20),
   OperationDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   Details VARCHAR(255)
);
```

Trigger Implementation

Trigger: Vote Insert Logging

This trigger automatically logs every vote cast in the system.

```
DELIMITER

CREATE TRIGGER trg_vote_insert
AFTER INSERT ON Vote
FOR EACH ROW
BEGIN
   INSERT INTO Operation_Locks_Log (TableName, OperationType, Details)
   VALUES ('Vote', 'INSERT', CONCAT('Vote ID ', NEW.VoteID, ' inserted'));
END$$

DELIMITER ;
```

Trigger: Voter Update Logging

```
DELIMITER

CREATE TRIGGER trg_voter_update
AFTER UPDATE ON Voter
FOR EACH ROW
BEGIN
   INSERT INTO Operation_Locks_Log (TableName, OperationType, Details)
   VALUES ('Voter', 'UPDATE', CONCAT('Voter ID ', NEW.VoterID, ' updated'));
END$$

DELIMITER ;
```

Stored Procedure Implementation

Procedure: Fetch Voter Voting Details

This procedure displays voting details of a specific voter.

```
DELIMITER

CREATE PROCEDURE get_voter_votes(IN vid VARCHAR(10))
BEGIN
   SELECT V.VoteID, E.Title AS ElectionTitle, C.Name AS CandidateName,
V.Timestamp
   FROM Vote V
   JOIN Election E ON V.ElectionID = E.ElectionID
   JOIN Candidate C ON V.CandidateID = C.CandidateID
   WHERE V.VoterID = vid;
END$$

DELIMITER ;
```

Function Implementation

Function: Count Total Votes by a Voter

```
DELIMITER

CREATE FUNCTION total_votes_by_voter(vid VARCHAR(10))
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE total INT;
   SELECT COUNT(*) INTO total
   FROM Vote
   WHERE VoterID = vid;
   RETURN total;
END$$

DELIMITER ;
```

Function: Count Votes for a Candidate

```
DELIMITER

CREATE FUNCTION total_votes_for_candidate(cid VARCHAR(10))
RETURNS INT
DETERMINISTIC
BEGIN
   DECLARE total INT;
   SELECT COUNT(*) INTO total
   FROM Vote
   WHERE CandidateID = cid;
   RETURN total;
END$$

DELIMITER ;
```

Transaction Control

Demonstration of transaction handling using **ROLLBACK**.

```
START TRANSACTION;

INSERT INTO Voter
VALUES ('V999', 'Temp User', 'temp@smartvote.com', 'Voter', 'Not Voted',
'temp123');

ROLLBACK;
```

Execution Commands

```
SHOW TABLES;
SHOW TRIGGERS;

SELECT * FROM Operation_Locks_Log;

CALL get_voter_votes('V101');

SELECT total_votes_by_voter('V101');
SELECT total_votes_for_candidate('C201');
```