Project Final Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Understanding	3					
Analysis	4					
Implementation	8					
Report Writing	10					
Total obtained mark						

Semester: Fall 2024……

Course Code: SE 224 **Course Name: Database Management System Lab**

Course Teacher Name: Tapushe Rabaya Toma

Designation: Assistant Professor

Student Name	Student ID	Section
Afzal Hossain Raju	0242310005341031	40-D
Md. Asraful Alam Rahat	0242310005341392	40-D

Submission Date: 08 /12/2024

Online voting system

- Scenario overview:
- Table:
- Example Relationships:
- Create the database in SQL
- Now create tables in SQL
- Insert data into all tables

1. Insert into table voters
2. Insert into Election
3. Insert into Positions
4. Insert into Candidates
5. Insert into Votes
6. Insert into Election Results

- Delete Operation

1.a) Delete a voter who has not participated in any election
1.b) Remove votes for a specific candidate

- UPDATE Operation

2. a) Update a voter's status to indicate they have voted

2. b) Change the election date for an upcoming election

2. c) Update the number of votes received by a

Candidate

- ALTER Operation

3. a) Add a new column to store voter phone numbers

3. b) Modify the size of the email field in the Voters

table

3.c) Rename Voters table to Registered_Voters using

Alter

- Drop a Column:

4.a) Remove the party column from the Candidates table

- Basic SELECT Operations:

5.a) Retrieve all voters' information
5.b) Display candidates participating in a specific election

- SELECT with AND:

6.a) find full_name, email, status columns from the

Voters table where status = TRUE And email LIKE '%@example.com':

- SELECT with AND and OR:

7.a) Find full_name, email, and status from Voters where the status will TRUE AND email LIKE '%@example.com' OR email LIKE '%@gmail.com'

- SELECT with Aggregate Functions:

8.a) Count the total number of voters

8.b) Find the candidate with the highest votes in a specific election

- SELECT with ORDER BY:

9.a) Retrieve all candidates sorted by their votes in descending order

9.b) Fetch voters sorted by their names alphabetically

- SELECT with LIMIT:

10.a) Retrieve the top 3 candidates with the highest votes

- MIN and MAX:

11.a) Find the minimum and maximum number of votes received by any candidate in the election results

- AVG (Average)

12.a) Calculate the average votes received by candidates in a specific election

- GROUP BY

13.a)Count the number of candidates for each election

- HAVING

14.a) Find candidates who received more than 2 votes in total

- Combining GROUP BY, HAVING, and Aggregation

15.a) Calculate the average votes received by candidates in each election and display only those with an average of more than 1 votes

- JOIN

16.a) INNER JOIN

16.b) LEFT JOIN

16.c) RIGHT JOIN

16.d) JOIN with Aggregation

16.e) JOIN with WHERE Clause

- Subquery

17.a)the details of voters who have voted for the candidate that received the highest number of votes in an ongoing election.

- Create view

18.a) 16(e) create as view.

Online voting system

Scenario overview:

An online voting system allows registered voters to cast their votes during an election. Each election may involve multiple candidates competing for specific positions. The system tracks voters, elections, candidates, votes, and results while maintaining security and anonymity of votes.

The purpose of **voters** is to store registered voters' information including voter_id (primary), full_name (varchar), email (varchar, unique), password (encrypted, varchar), address (varchar), status (boolean). The purpose of **elections** is to define elections and their details including election_id (primary key) election_name (varchar) start_date (date) end_date (date) status (enum: 'upcoming', 'ongoing', 'completed'). The purpose of **positions** is to define positions for which candidates are competing in an election including position_id (primary key) position_name (varchar) election_id (foreign key). the purpose of **candidates** is to store information about candidates contesting in an election including candidate_id (primary key) full_name (varchar) party (varchar) position_id (foreign key) election_id (foreign key). The purpose of **votes** is to record votes cast by voters including vote_id (primary key) voter_id (foreign key) election_id (foreign key) candidate_id (foreign key) timestamp (datetime). The purpose of **election results** is to store the outcome of elections including result_id (primary key) election_id (foreign key) position_id (foreign key) candidate_id (foreign key) votes_received (integer).

Table:

Voters (voter_id (primary), full_name (varchar), email (varchar, unique), password (encrypted, varchar), address (varchar), status (boolean))

Elections (election_id (primary key) election_name (varchar) start_date (date) end_date (date) status (enum: 'upcoming', 'ongoing', 'completed'))

Positions (position_id (primary key) position_name (varchar) election_id (foreign key))

Candidates (candidate_id (primary key) full_name (varchar) party (varchar) position_id (foreign key) election_id (foreign key))

Votes (vote_id (primary key) voter_id (foreign key) election_id (foreign key) candidate_id (foreign key) timestamp (datetime))

Election results (result_id (primary key, auto increment) election_id (foreign key) position_id (foreign key) candidate_id (foreign key) votes_received (integer))

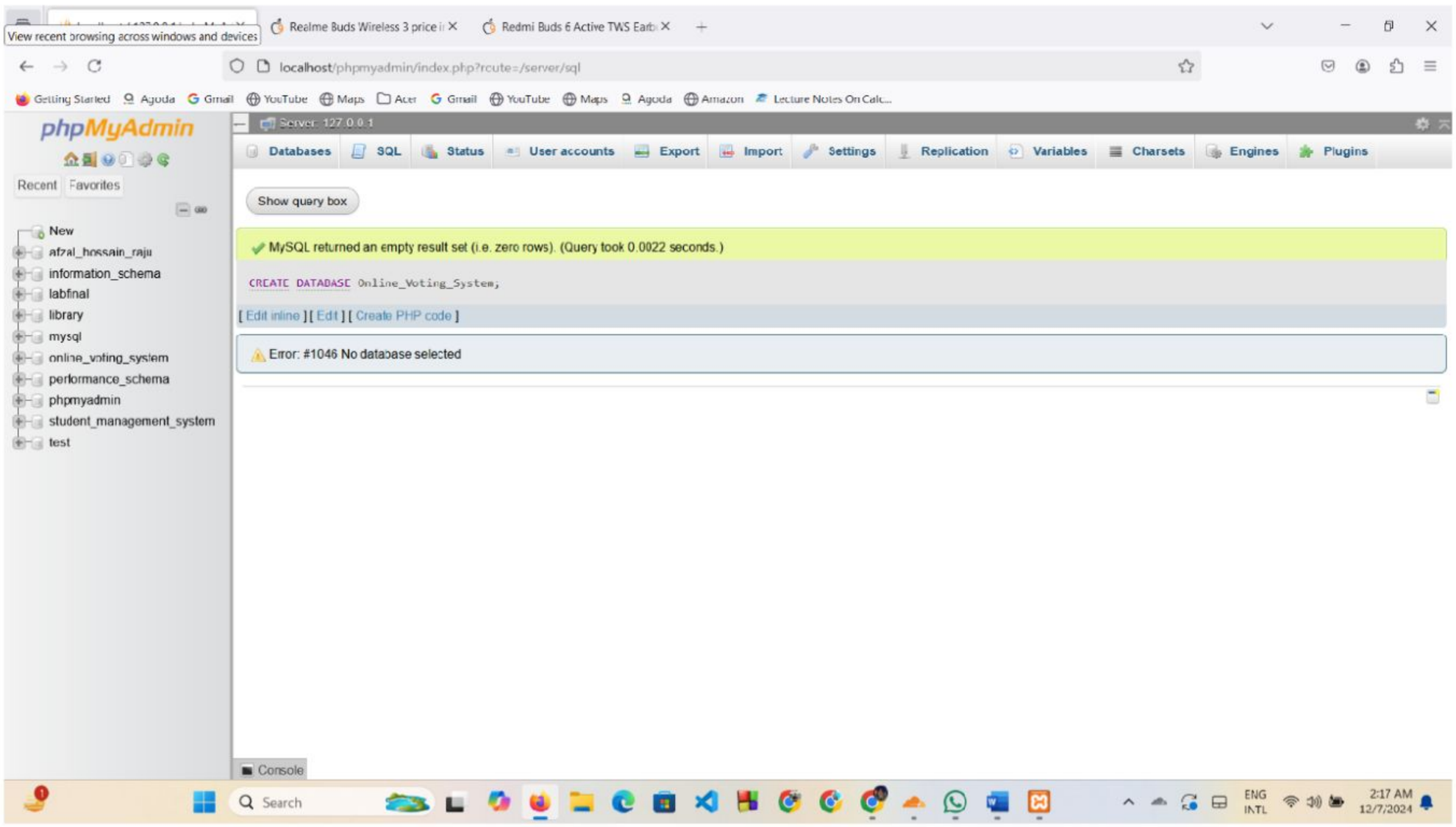
Example Relationships:

- **Voters** → Can vote in **Elections** (via the **Votes** table).
- **Elections** → Contain multiple **Positions**.
- **Positions** → Have multiple **Candidates**.
- **Candidates** → Receive votes in **Votes** table.
- **Election Results** → Summarize votes for candidates in an election

---

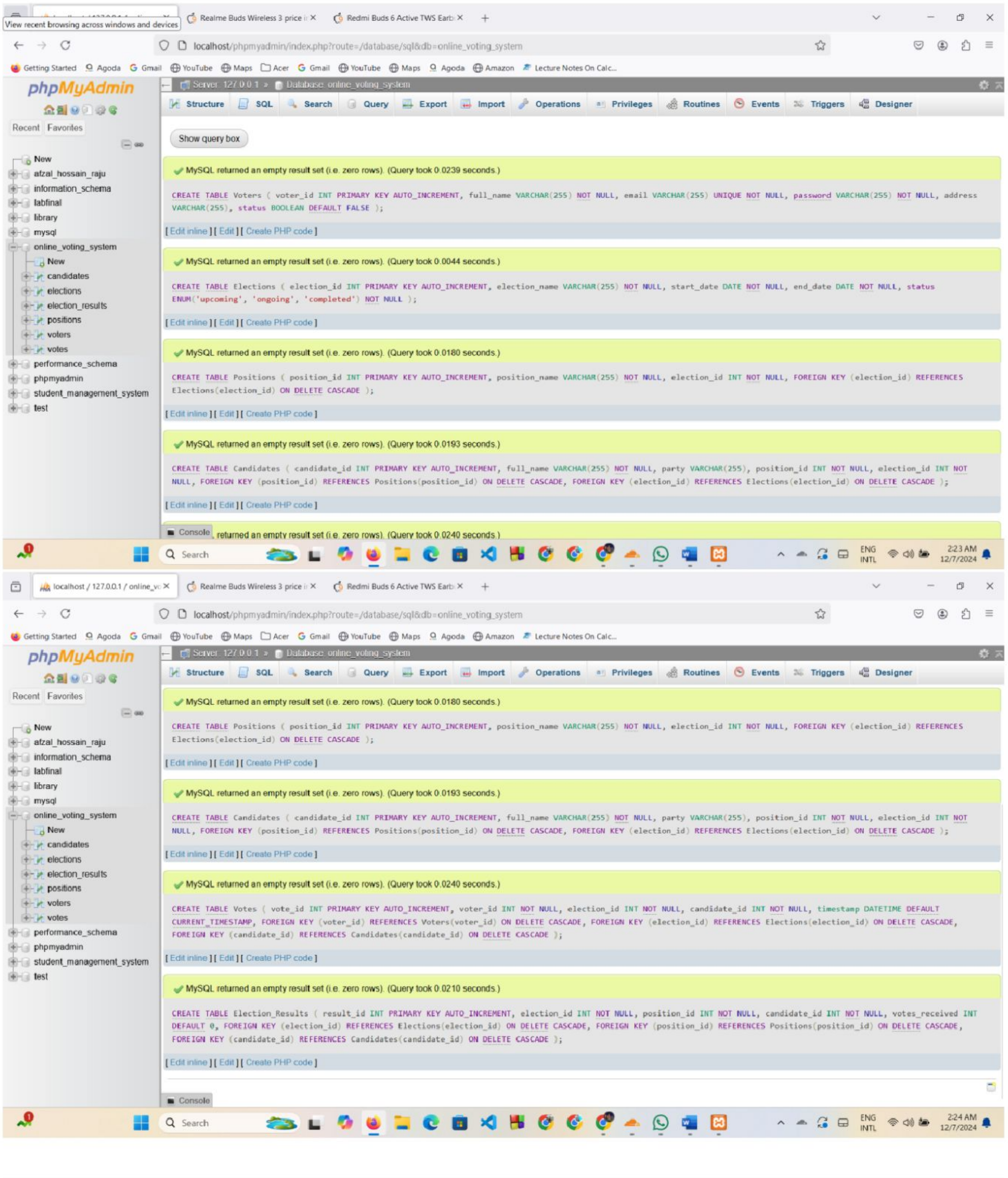
Create the database in SQL:

Create database: create the database Online Voting System



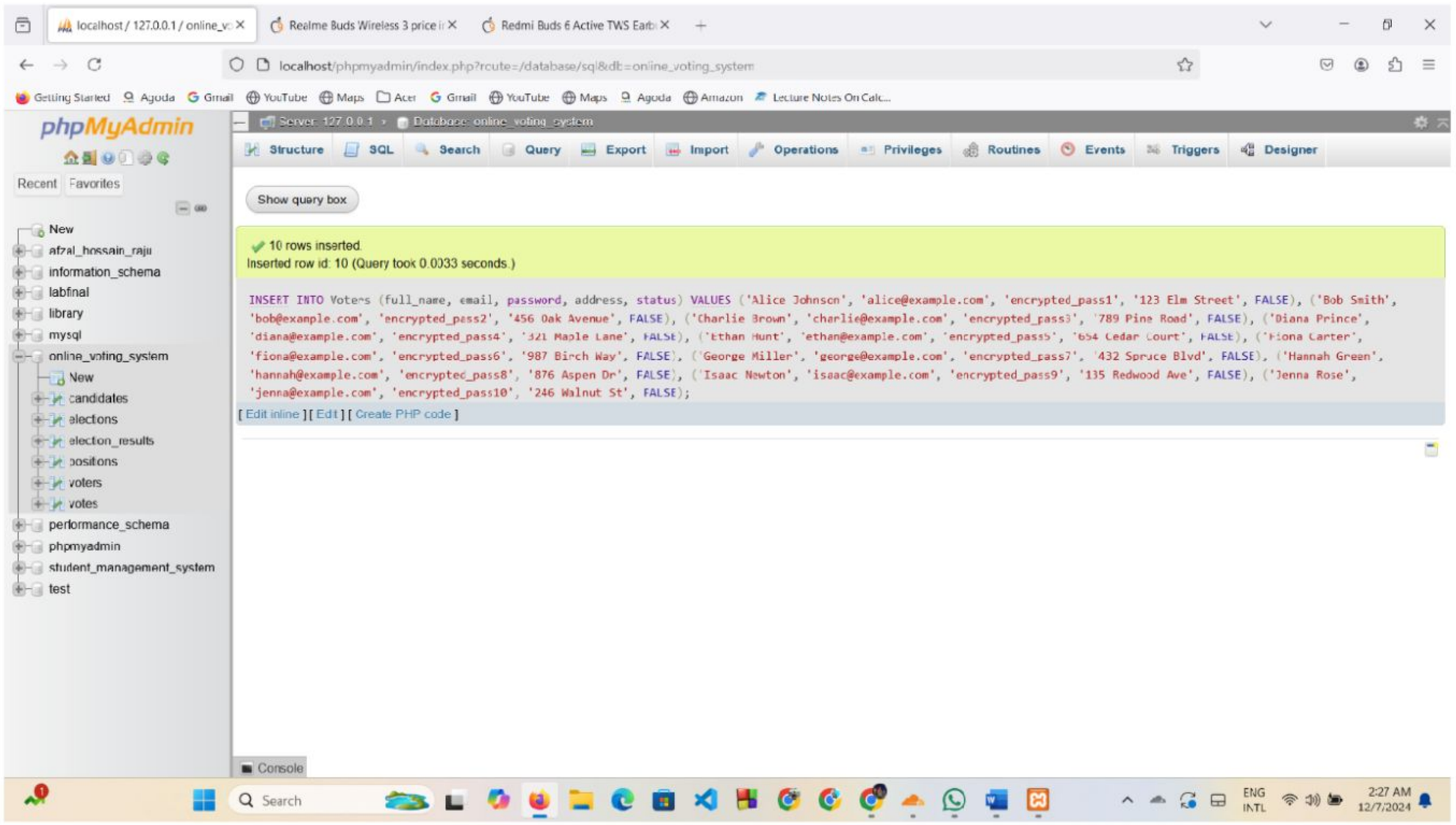
Now create tables in SQL:

Create Table: Create all tables(6 tables) with primary key and possible foreign key

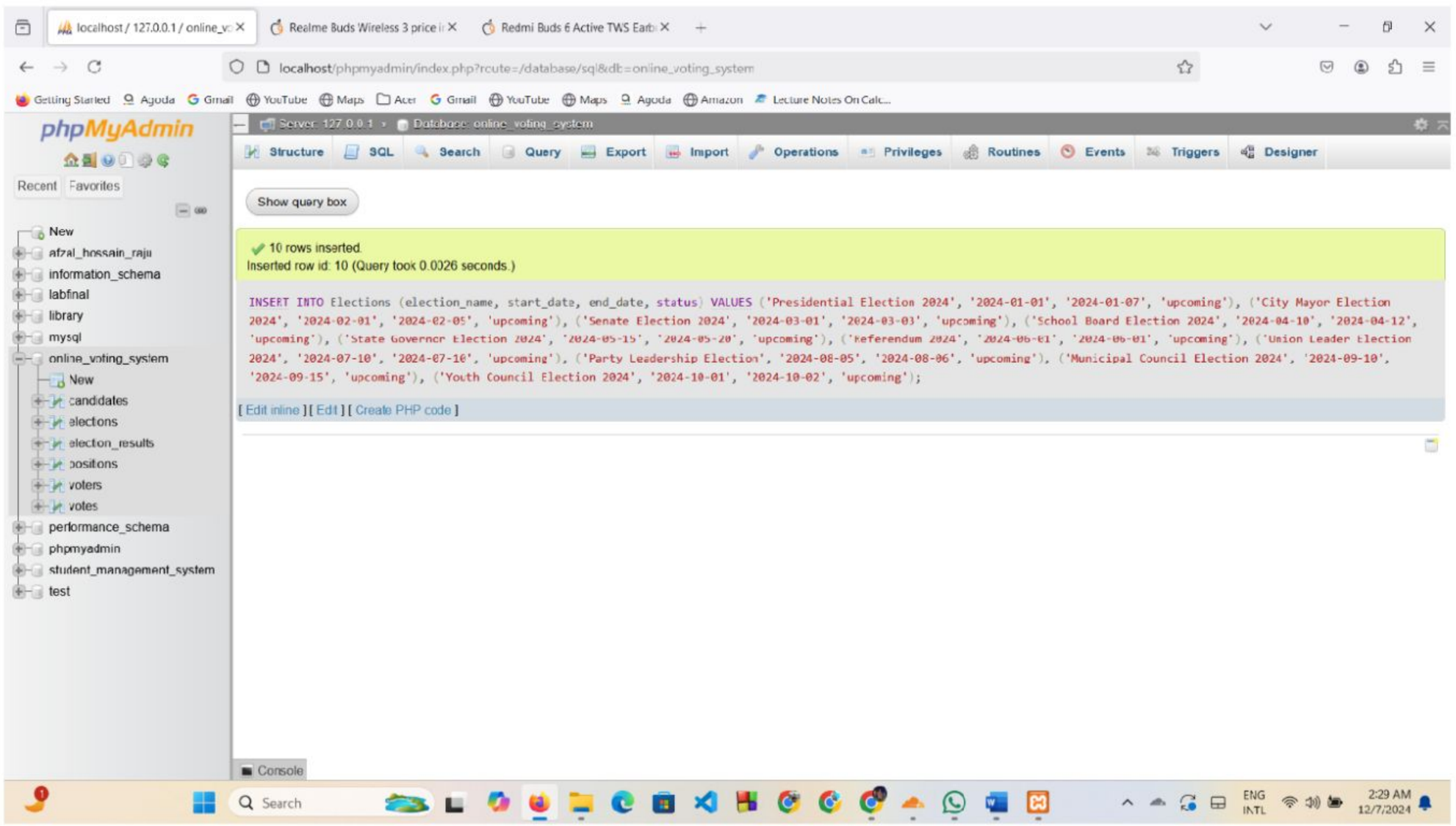


Insert data into all tables:

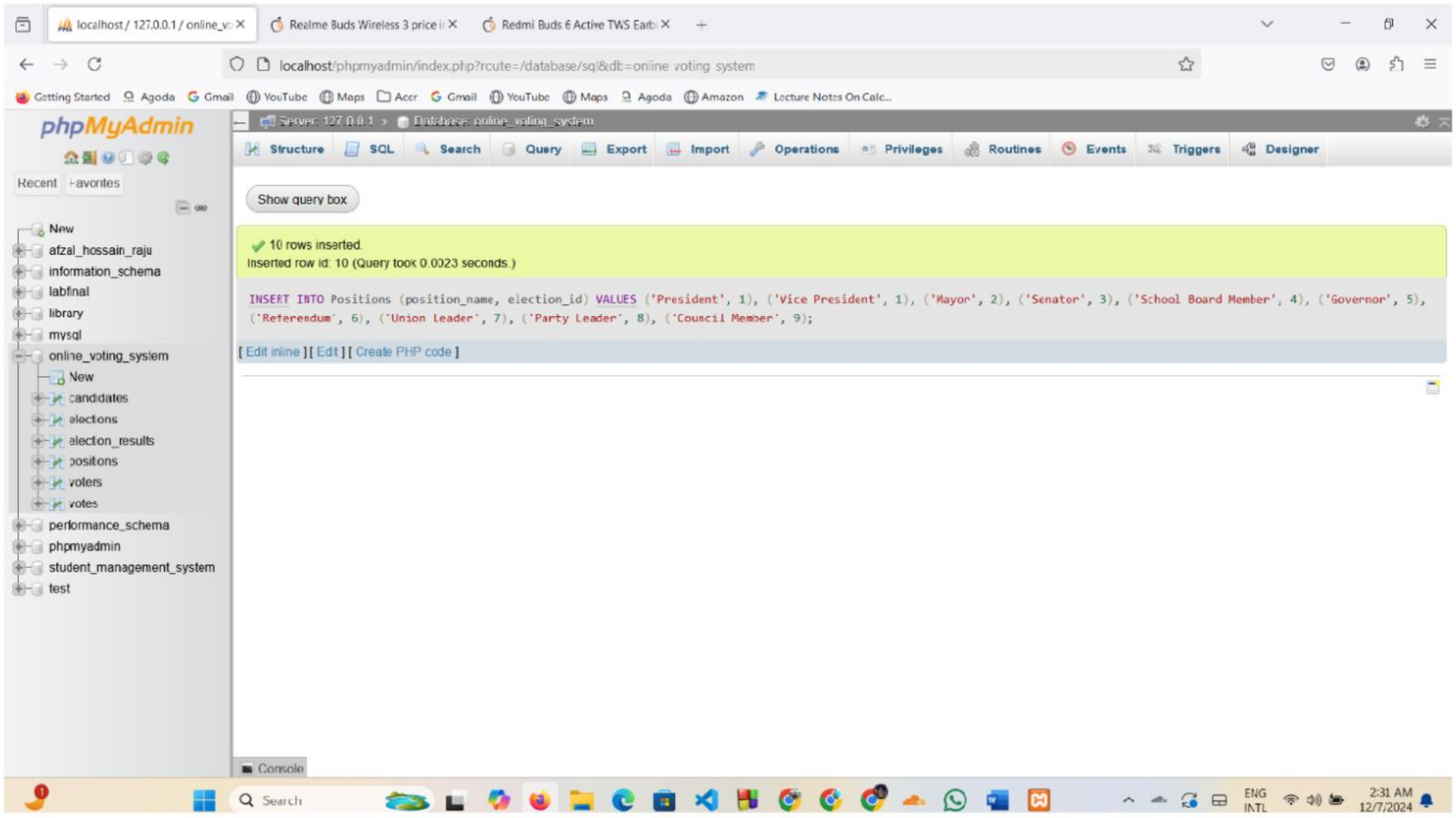
1. Insert into table voters:



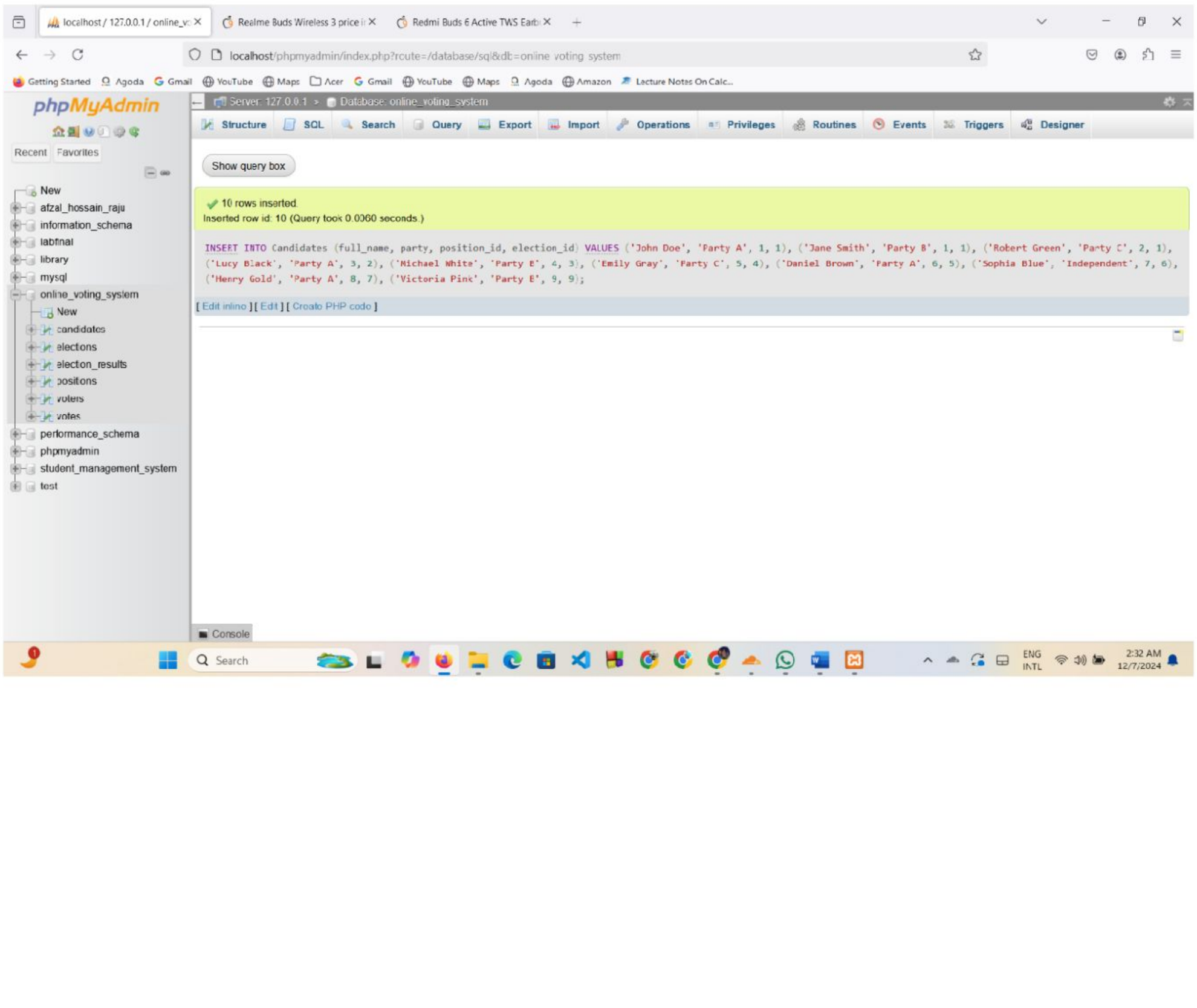
2. Insert into Election:



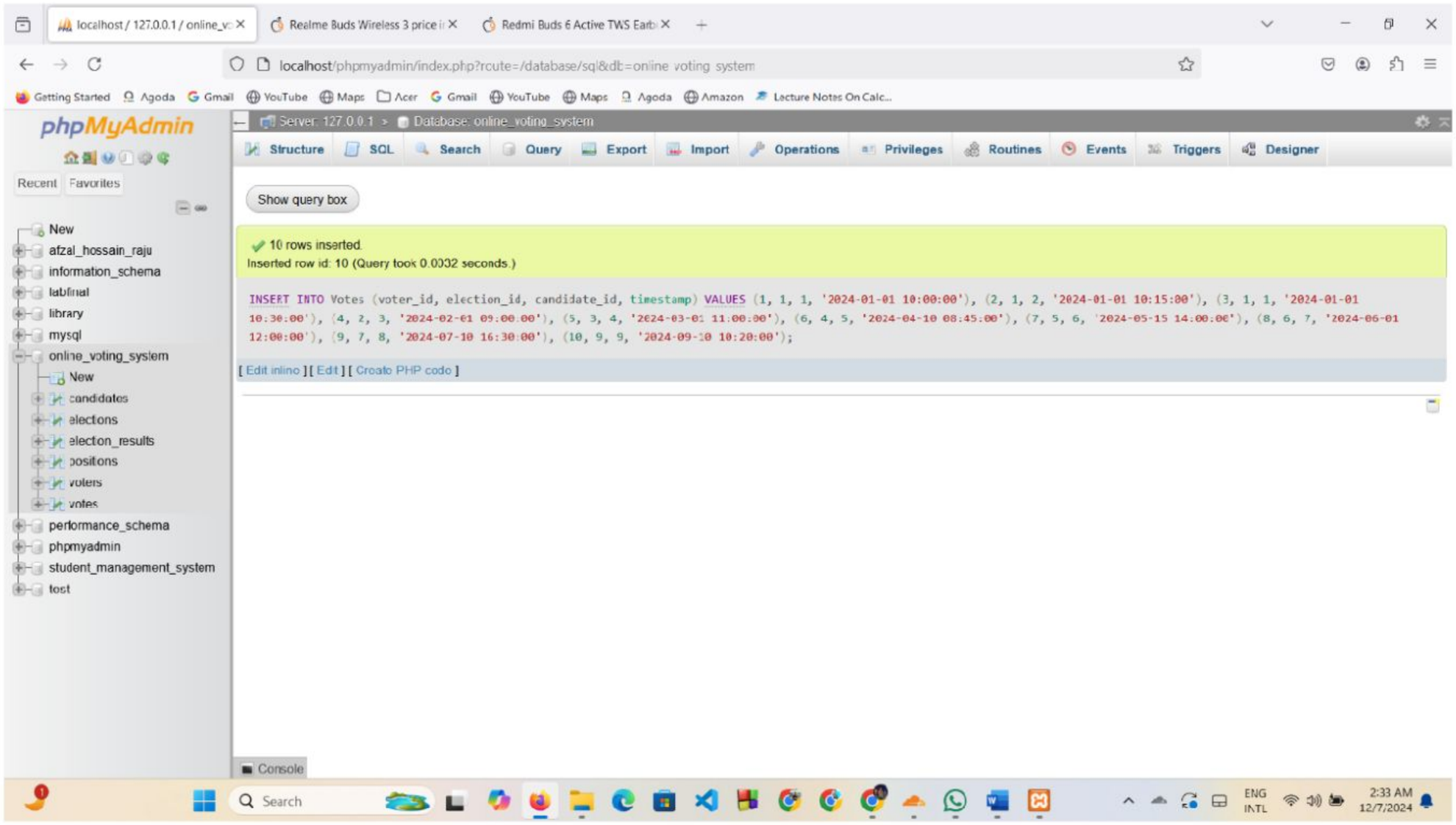
3. Insert into Positions:



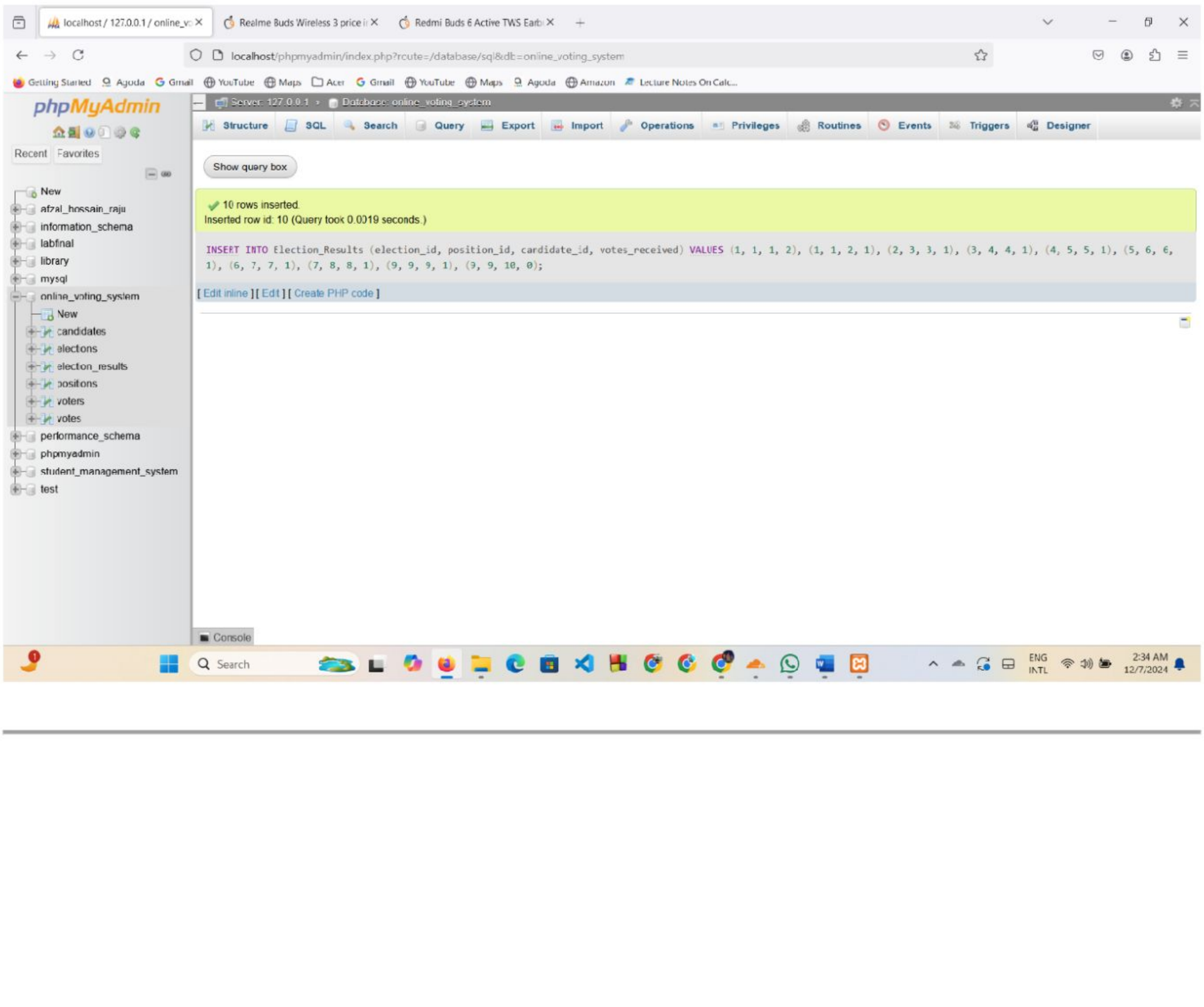
4. Insert into Candidates:



5. Insert into Votes:



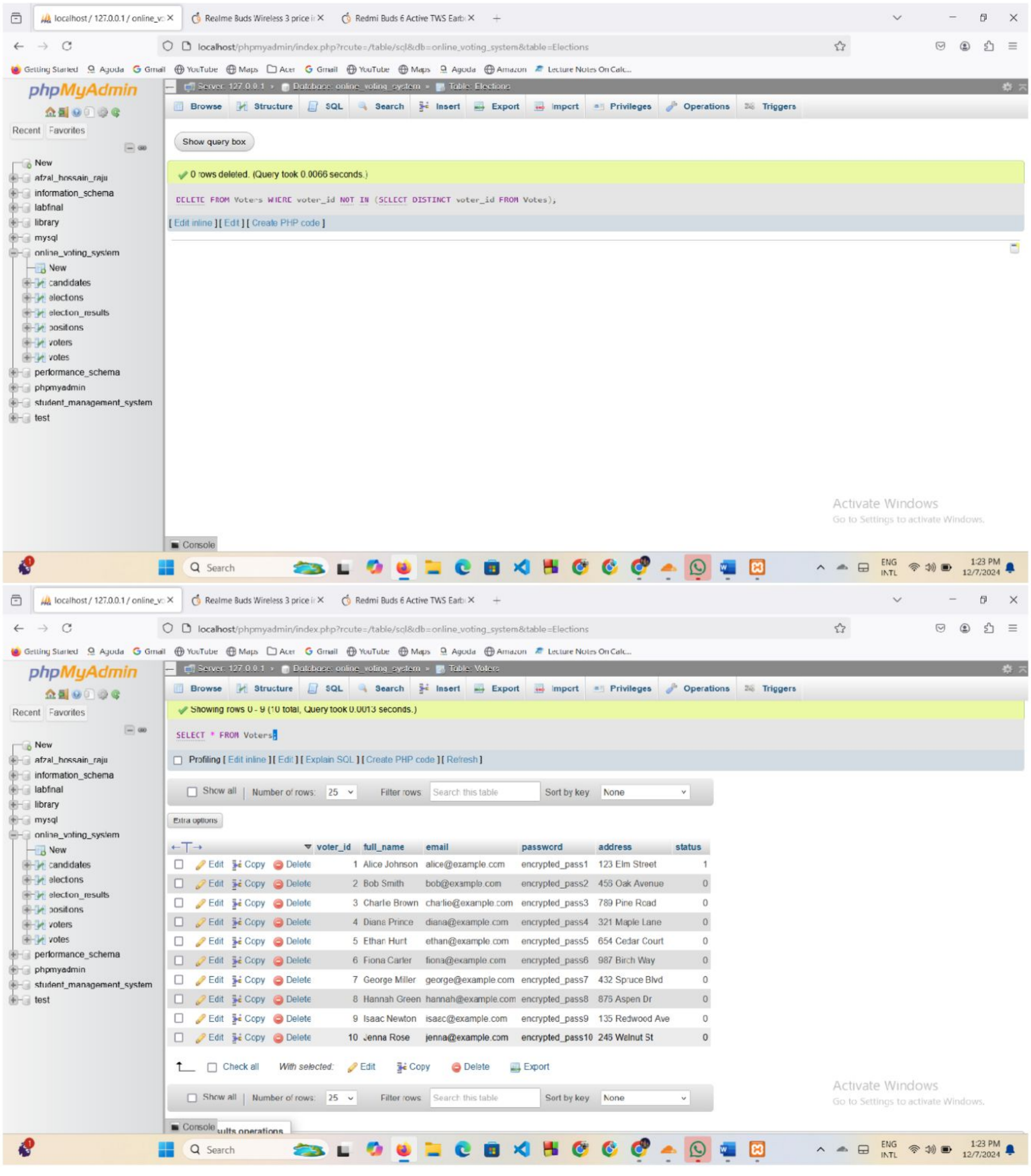
6. Insert into Election Results:



1. Delete Operation:

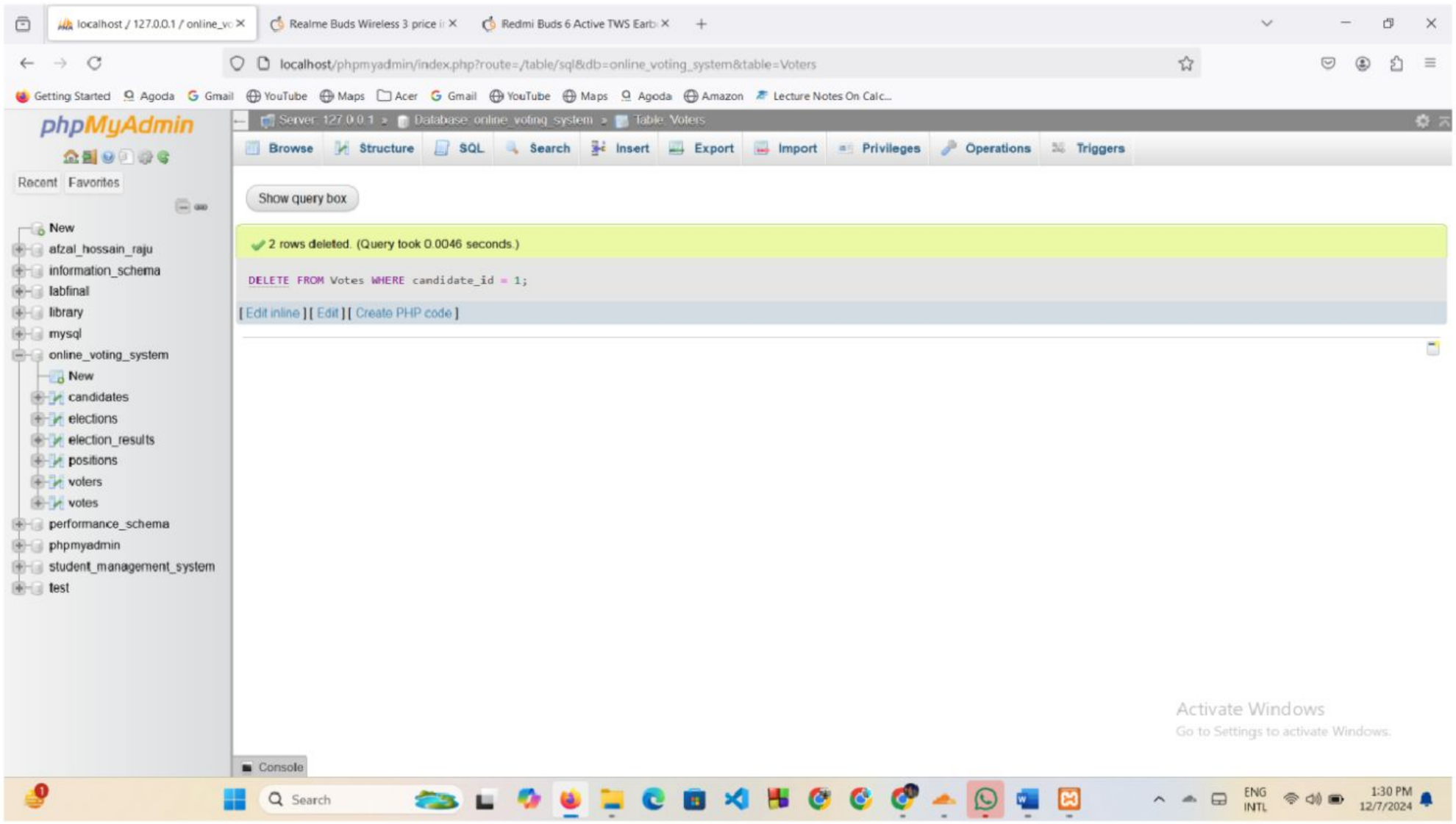
a) Delete a voter who has not participated in any election

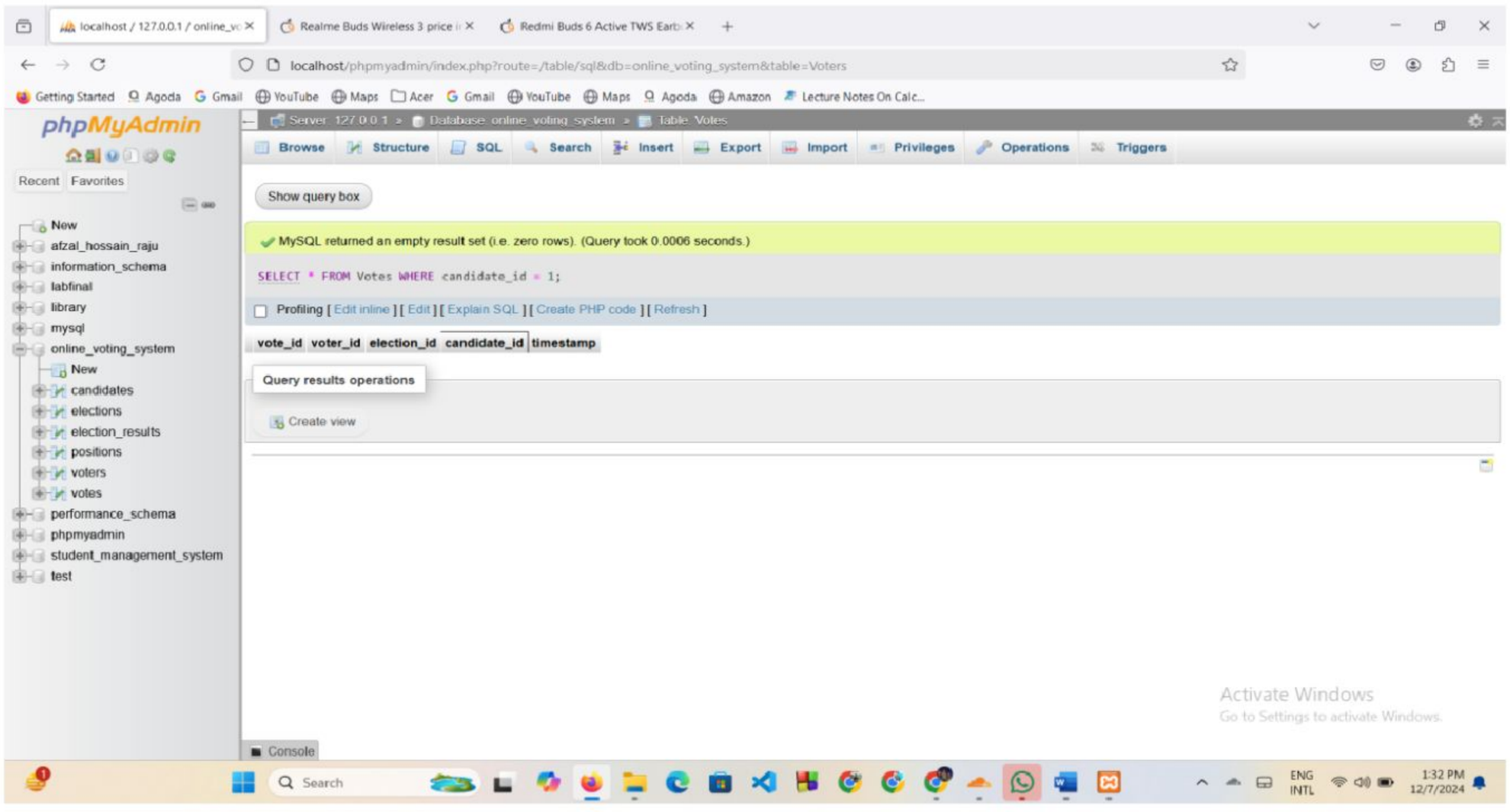
```
DELETE FROM Voters
WHERE voter_id NOT IN (SELECT DISTINCT voter_id FROM Votes);
```



b) Remove votes for a specific candidate

```
DELETE FROM Votes
WHERE candidate_id = 1;
```





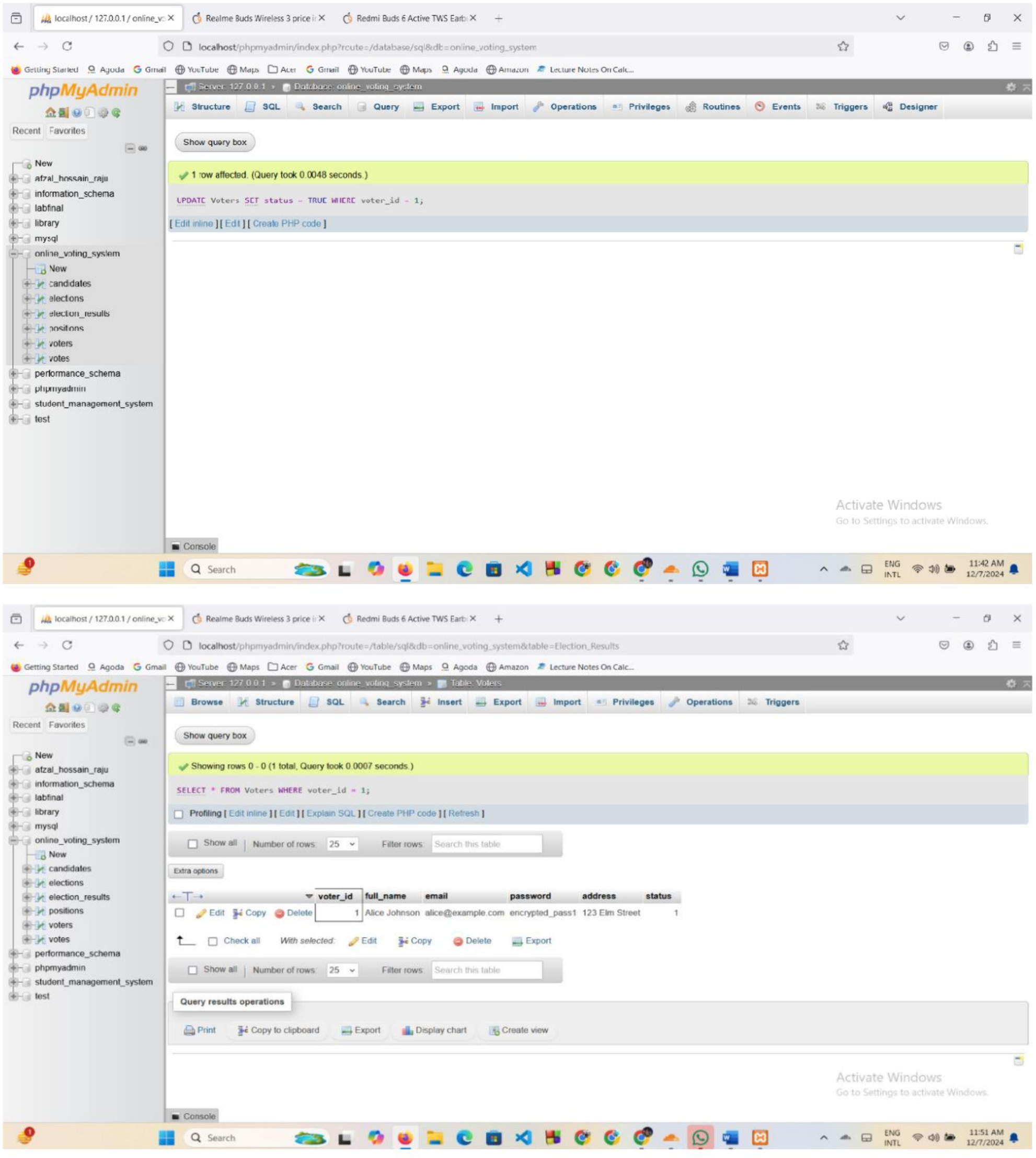
2. UPDATE Operation:

a) Update a voter's status to indicate they have voted

SQL: UPDATE Voters

SET status = TRUE

WHERE voter_id = 1;

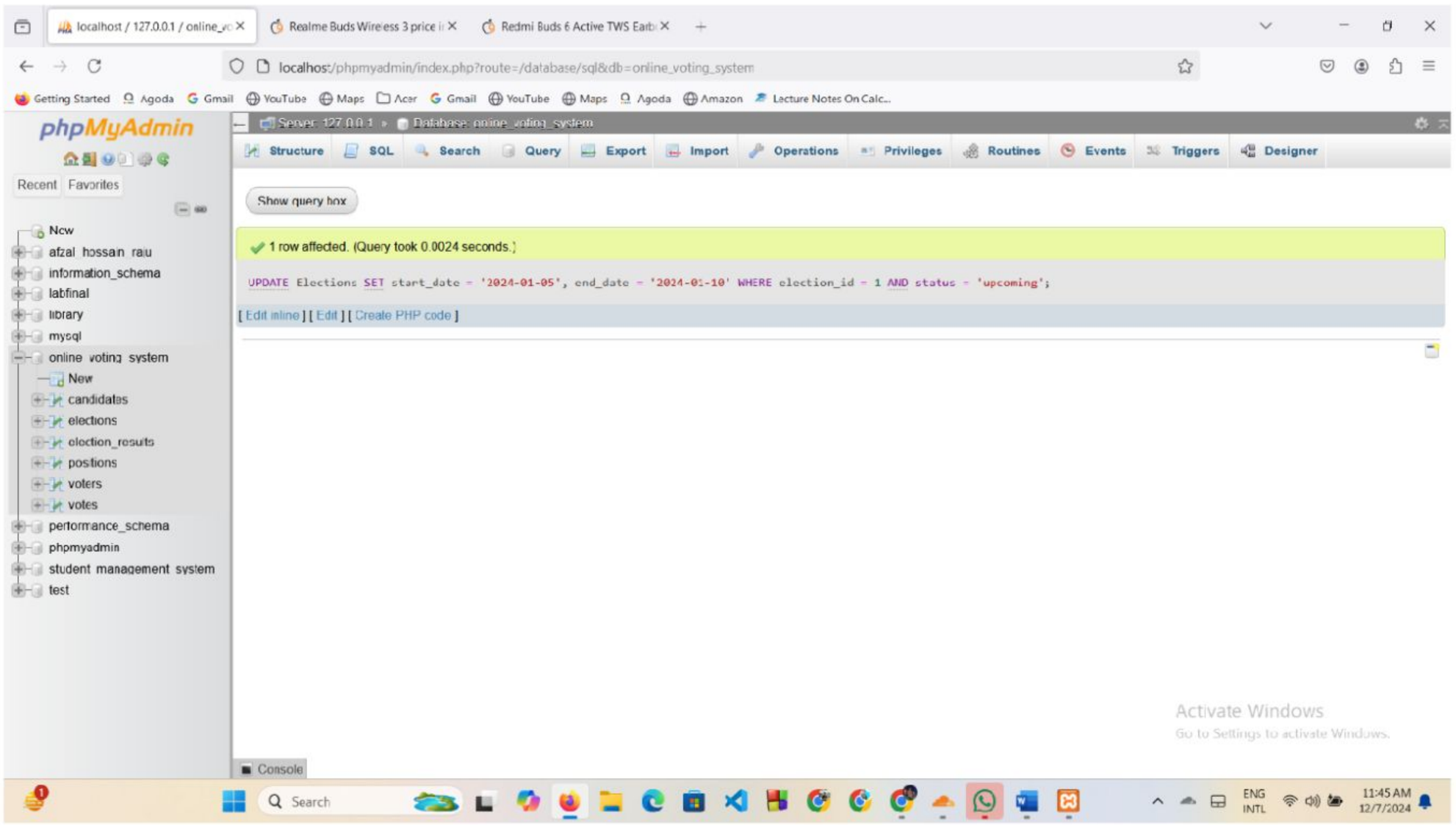


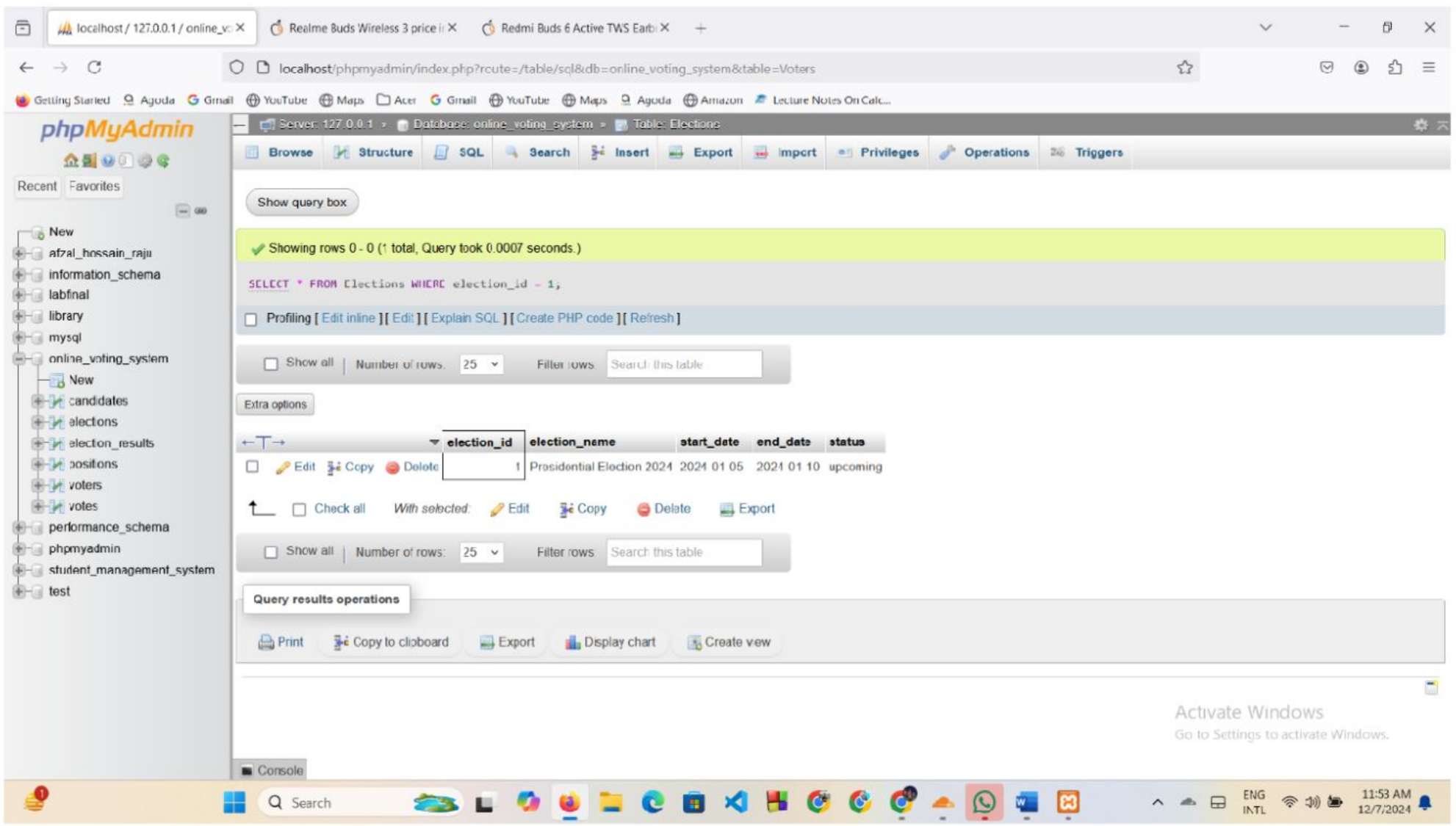
b) Change the election date for an upcoming election

SQL:UPDATE Elections

SET start_date = '2024-01-05', end_date = '2024-01-10'

WHERE election_id = 1 AND status = 'upcoming';



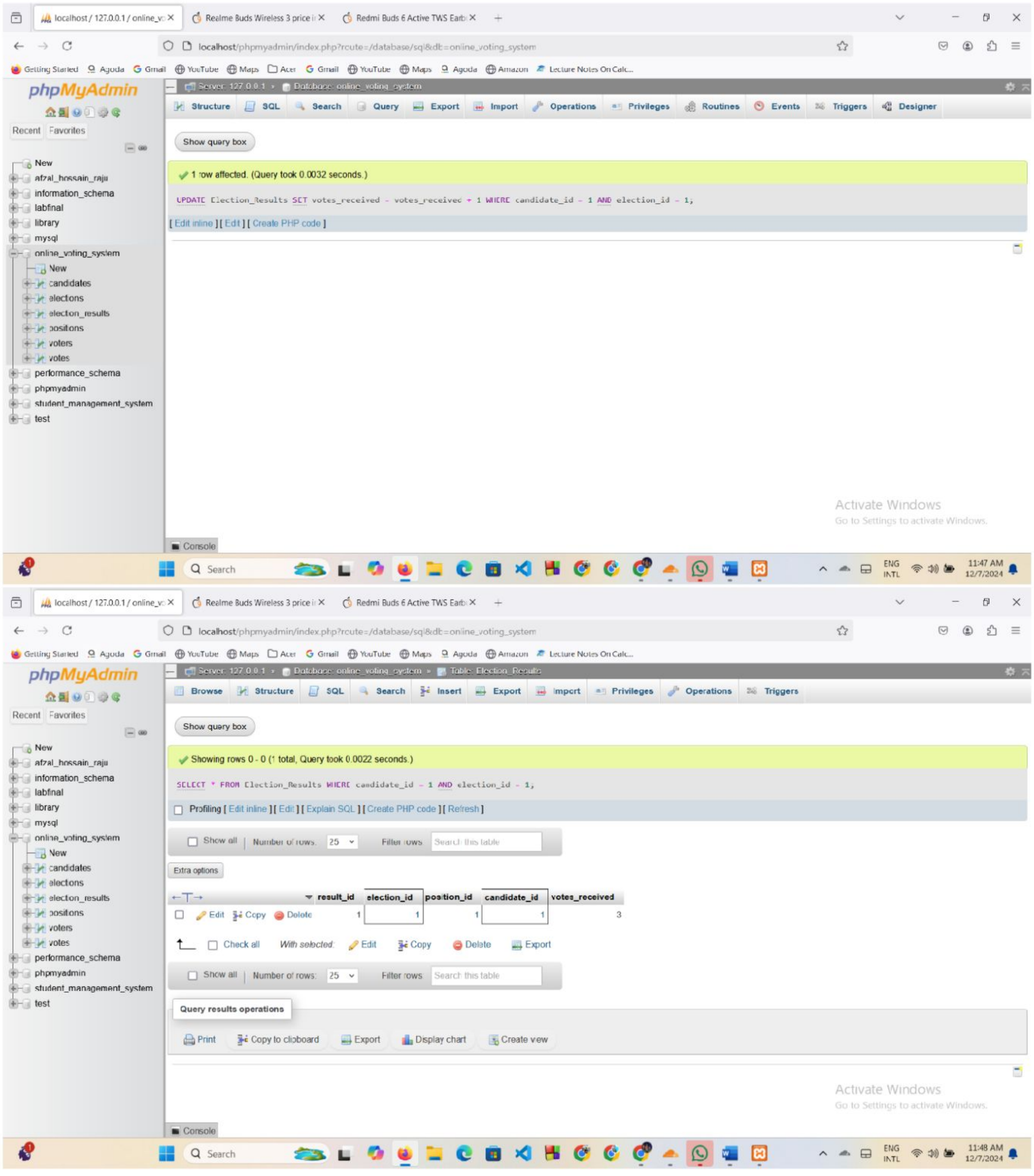


c) Update the number of votes received by a candidate

SQL: UPDATE Election_Results

SET votes_received = votes_received + 1

WHERE candidate_id = 1 AND election_id = 1;



3. ALTER Operation:

a) Add a new column to store voter phone numbers

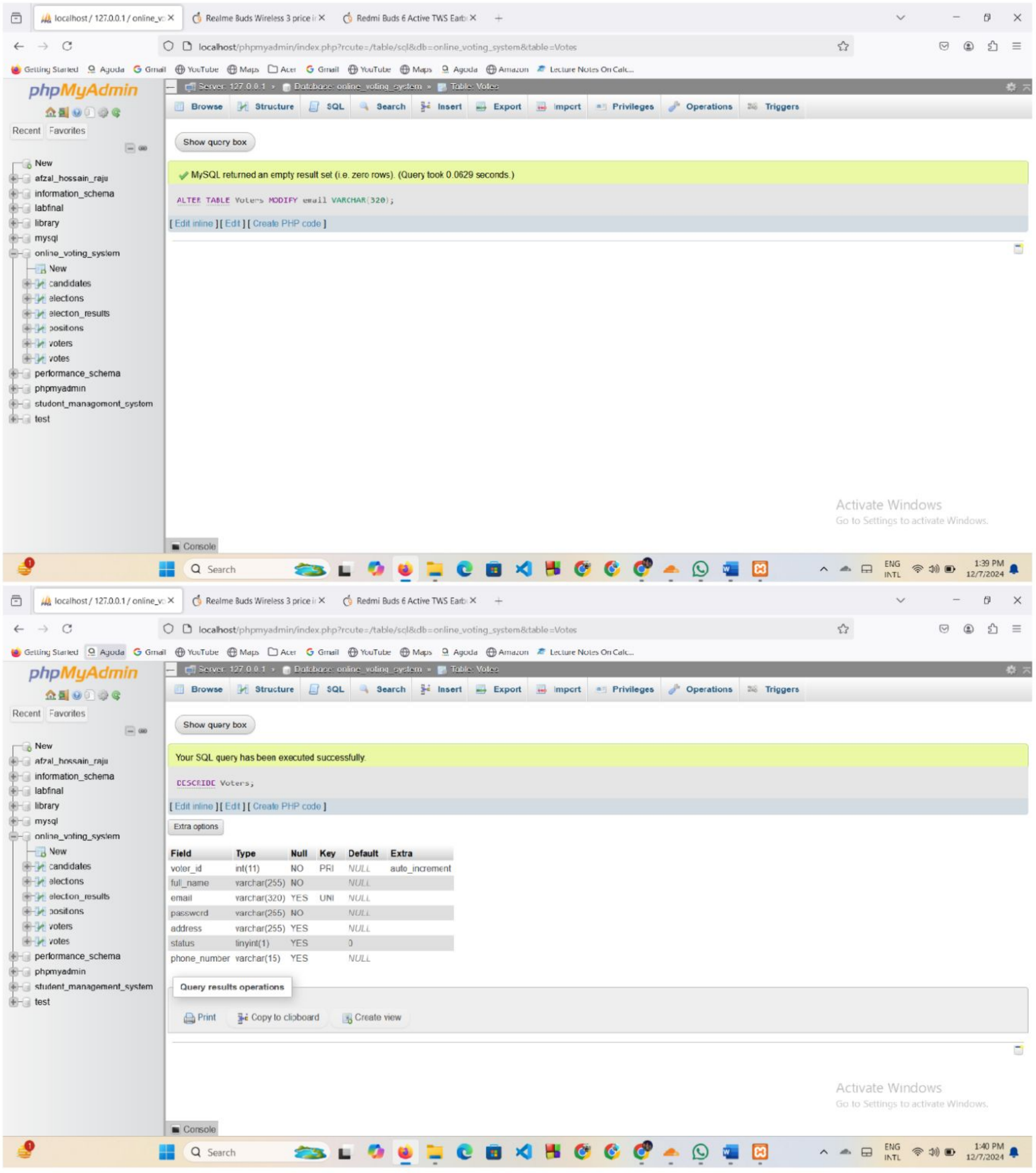
SQL: ALTER TABLE Voters

ADD phone_number VARCHAR(15);

b) Modify the size of the email field in the Voters table

SQL: ALTER TABLE Voters

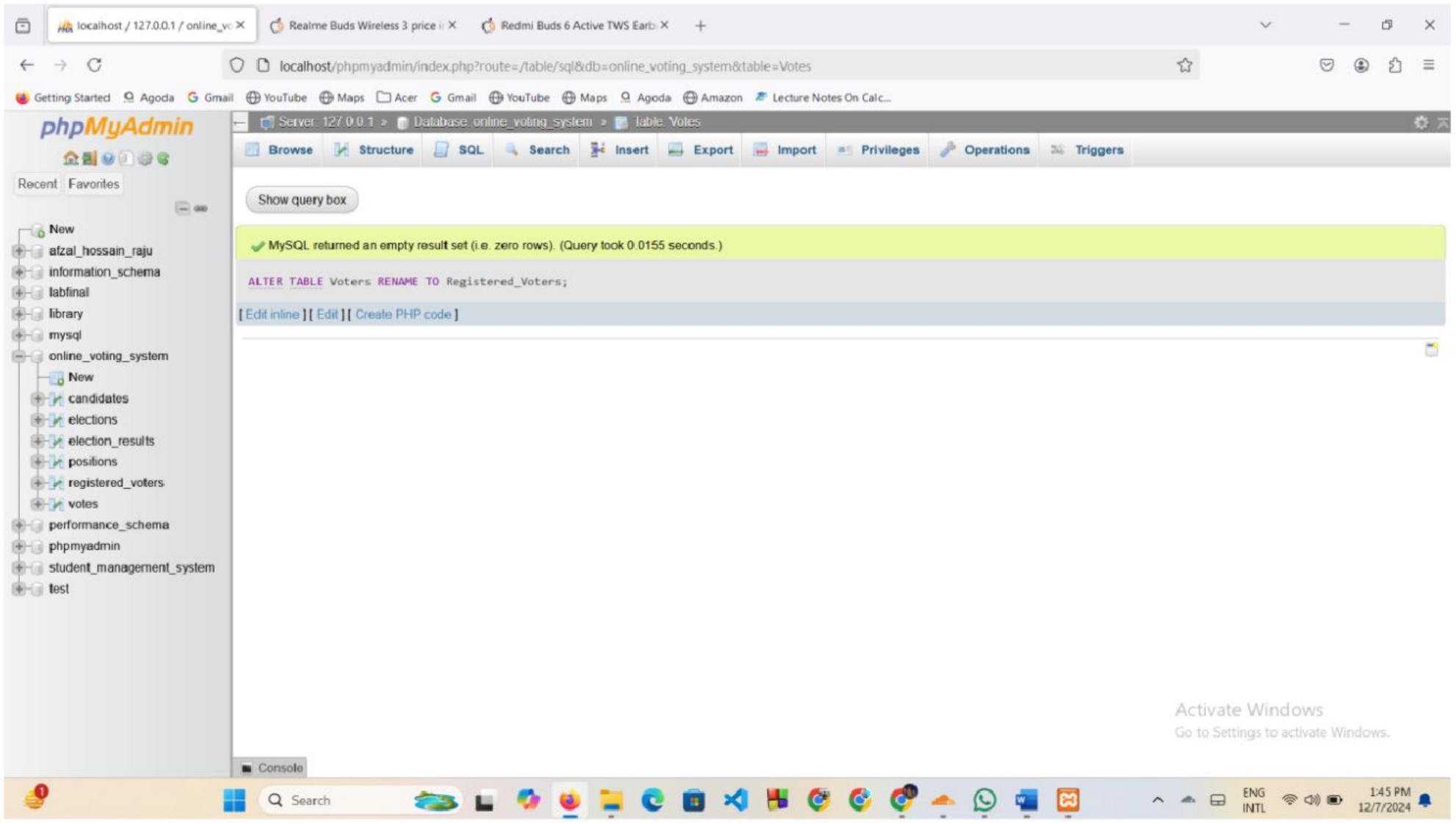
MODIFY email VARCHAR(320);

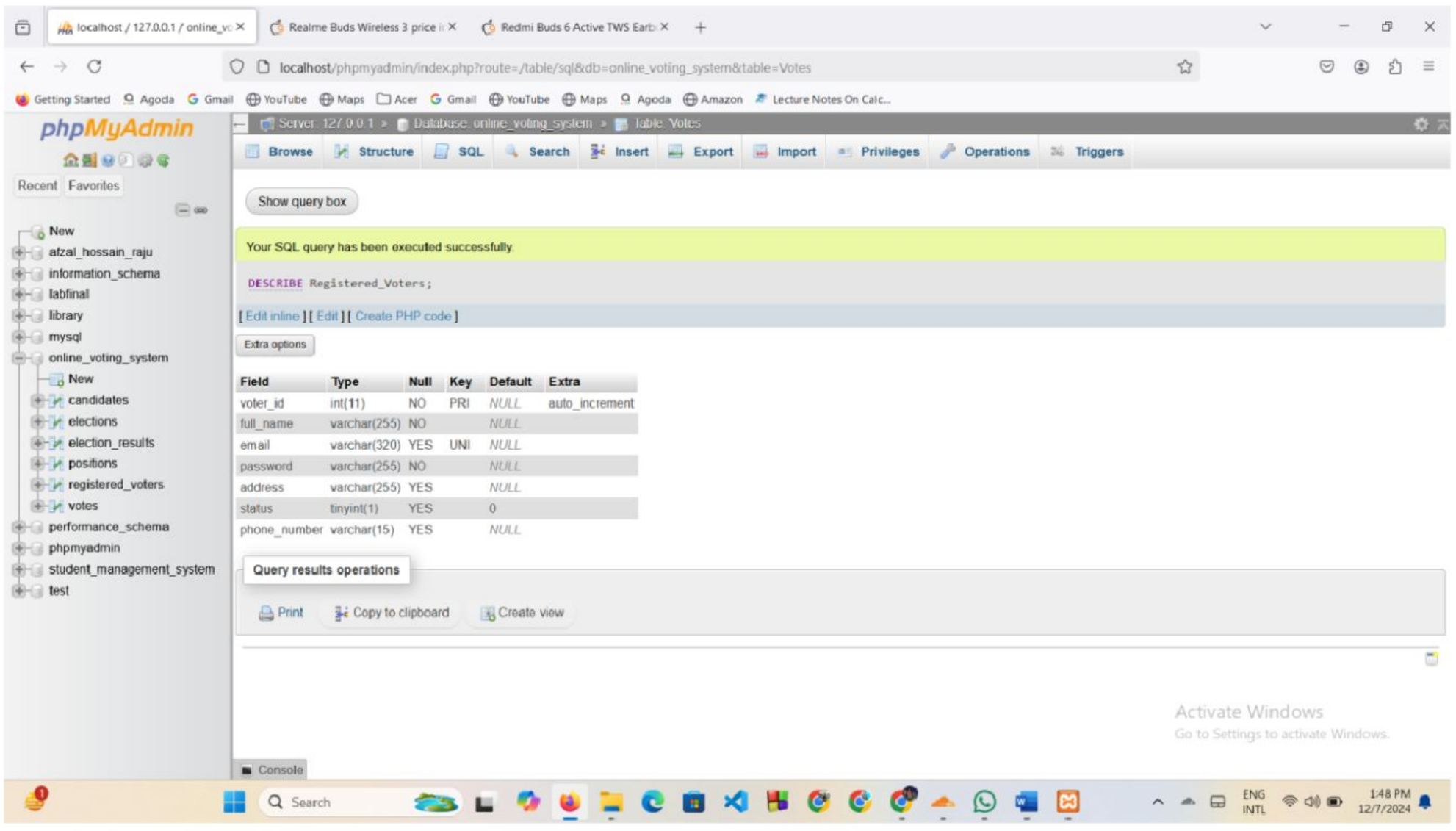


c) Rename Voters table to Registered_Voters using Alter

SQL: ALTER TABLE Voters

RENAME TO Registered_Voters;



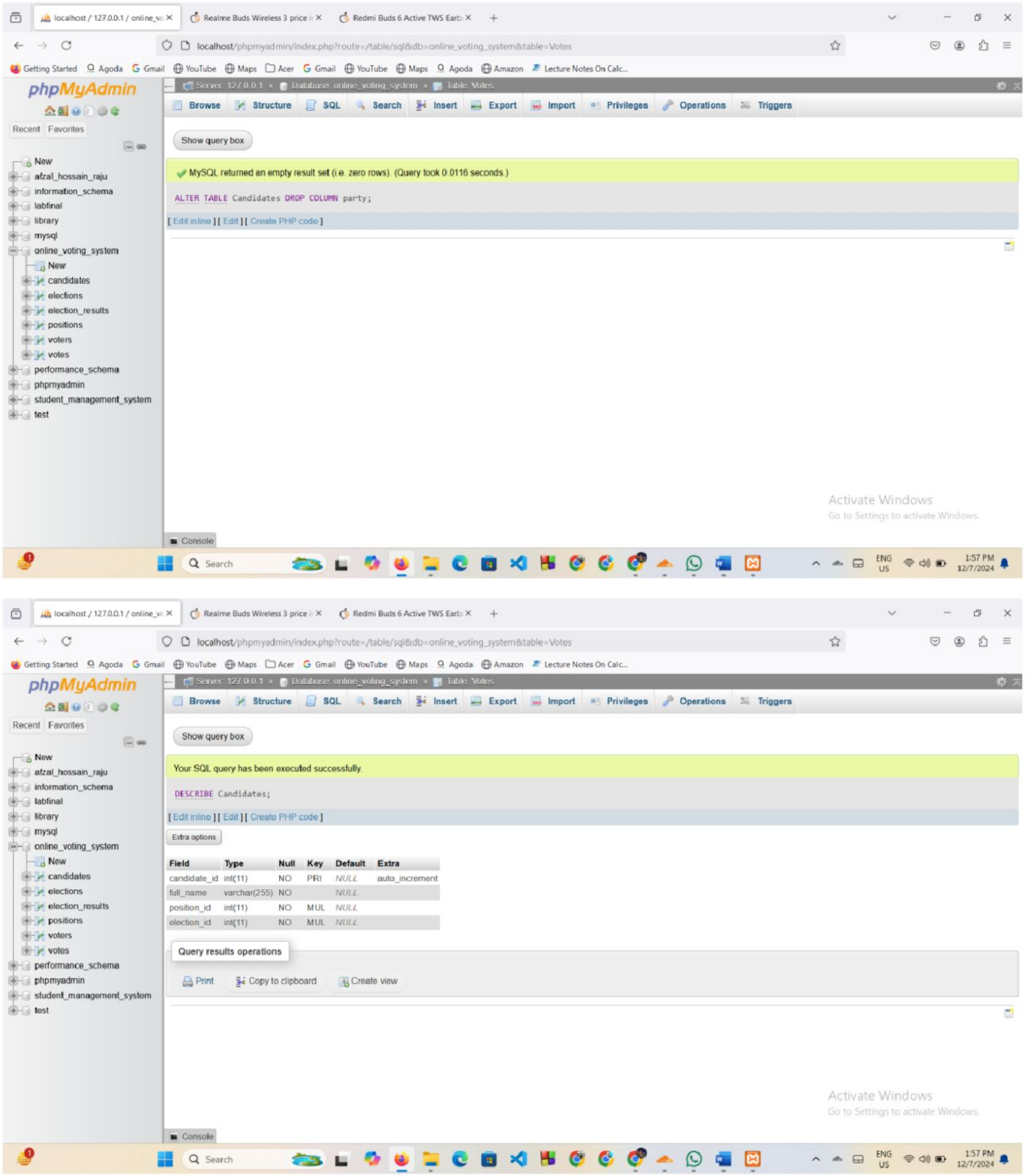


4. Drop a Column:

a) **Remove the party column from the Candidates table**

SQL: ALTER TABLE Candidates

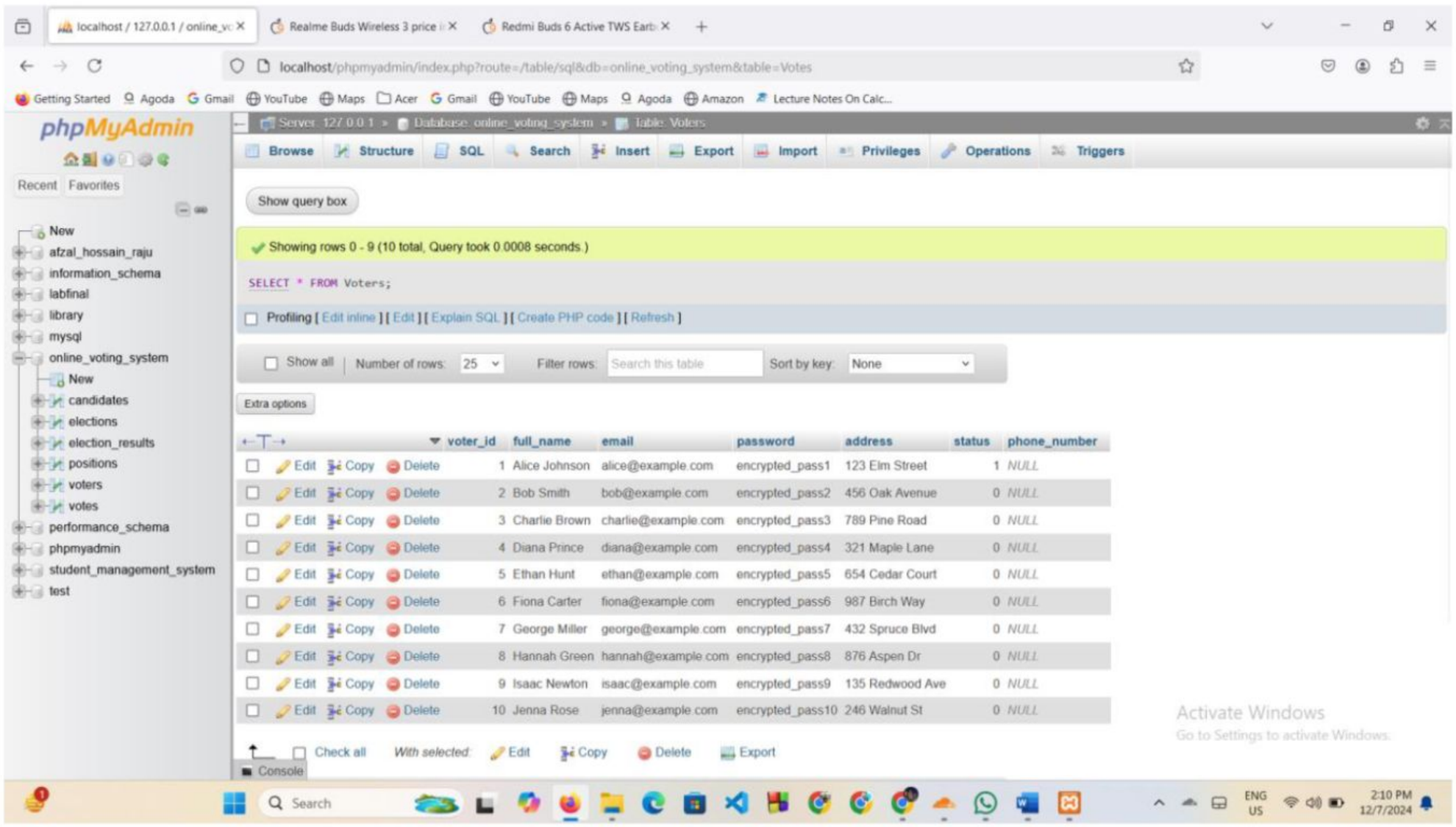
DROP COLUMN party;



5. **Basic SELECT Operations:**

a) Retrieve all voters' information

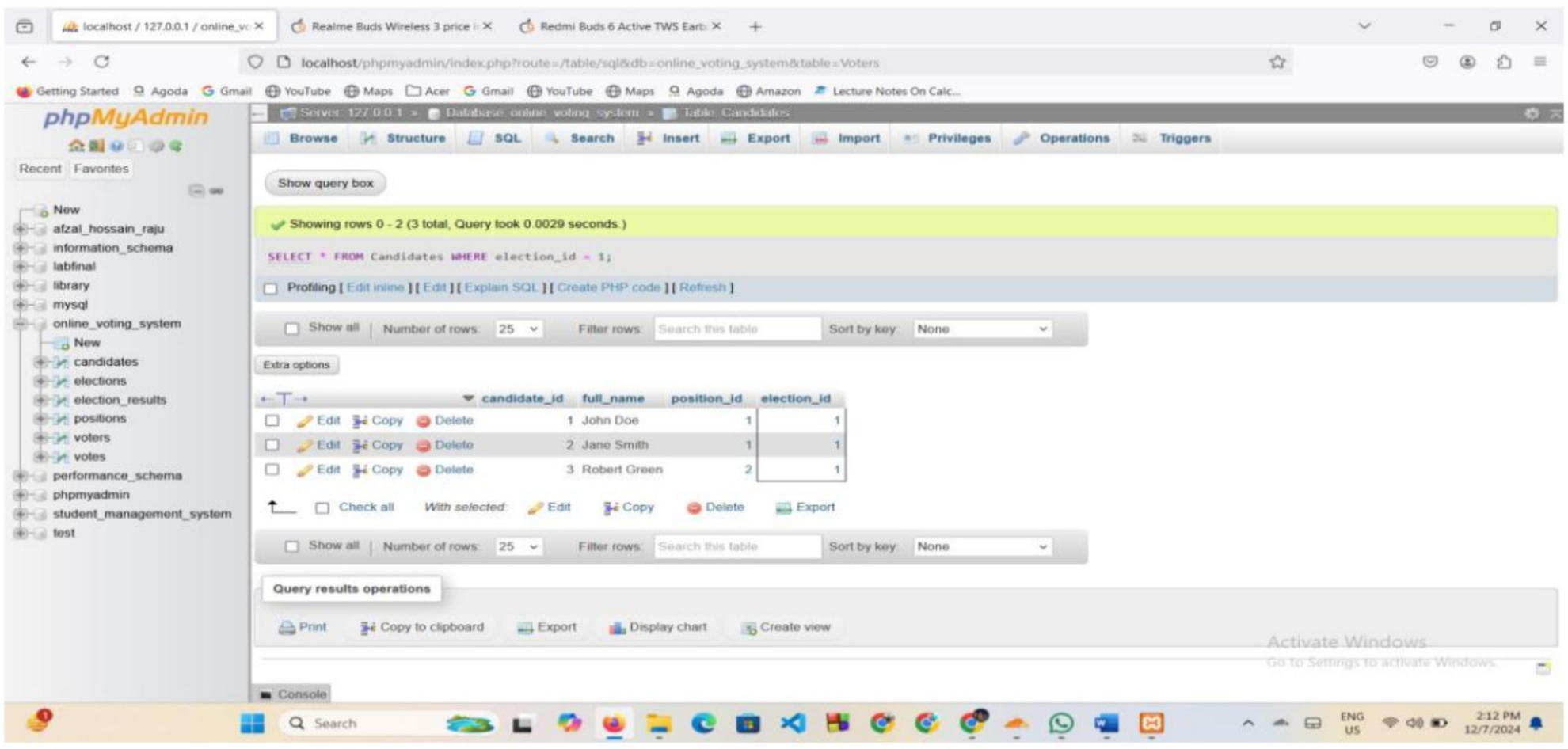
SQL: SELECT * FROM Voters;



b) Display candidates participating in a specific election

SQL: SELECT * FROM Candidates

WHERE election_id = 1;



6. SELECT with AND:

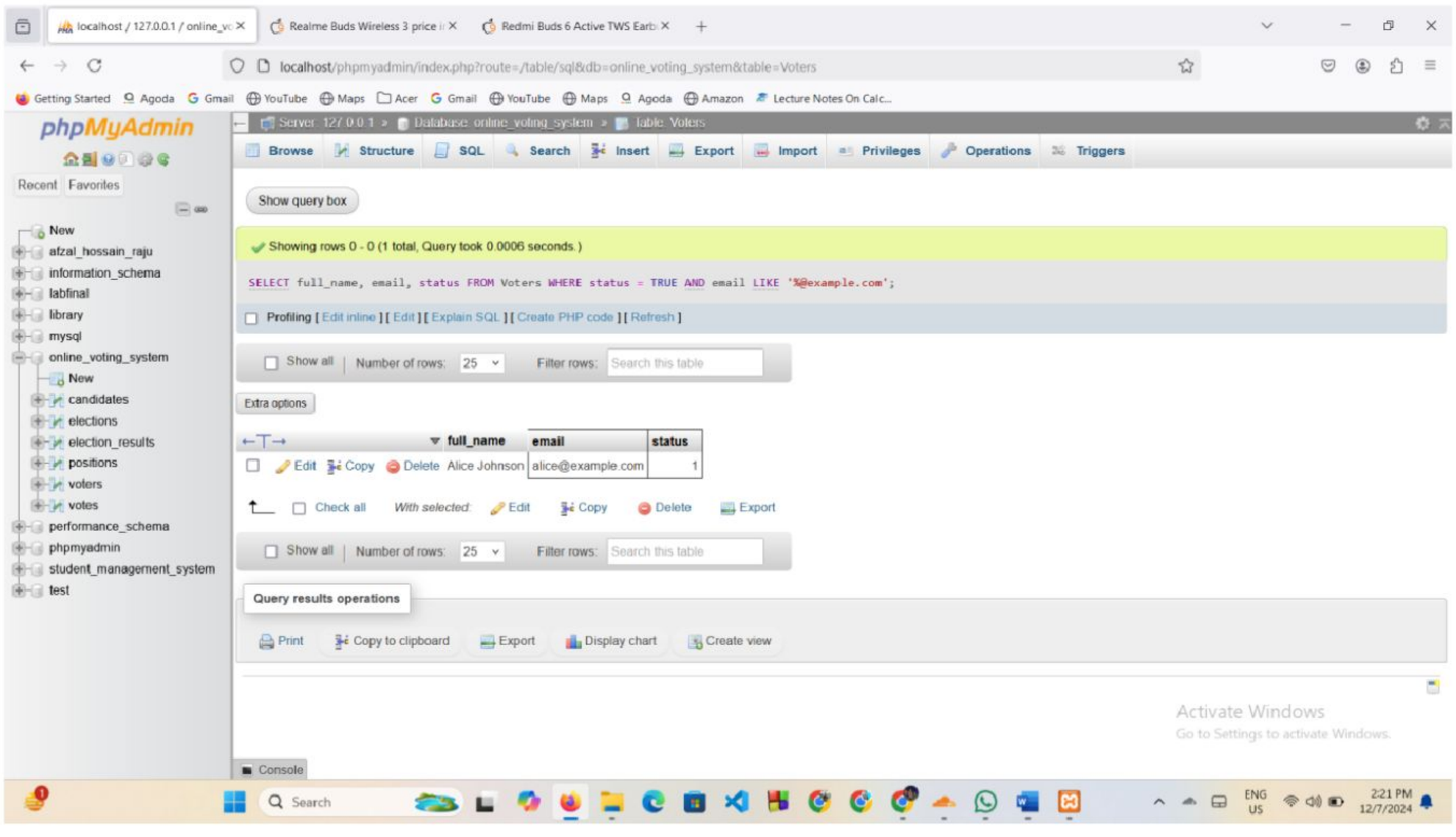
a) find full_name, email, status columns from the Voters table where status = TRUE And email LIKE '%@example.com':

SQL: SELECT full_name, email, status

FROM Voters

WHERE status = TRUE

AND email LIKE '%@example.com';



7. SELECT with AND and OR:

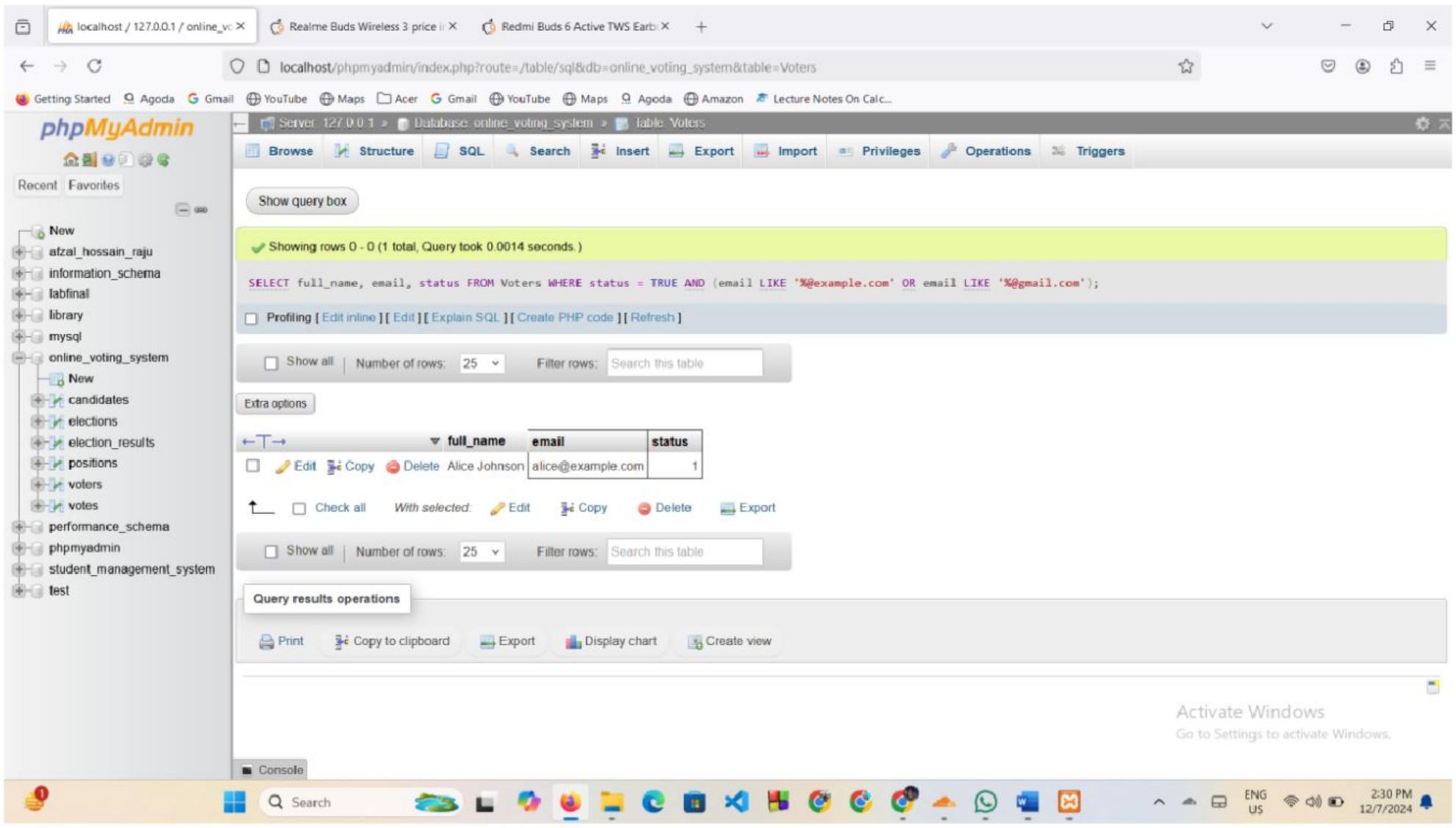
a) Find full_name, email, and status from Voters where the status will TRUE AND email LIKE '%@example.com' OR email LIKE '%@gmail.com'

SQL: SELECT full_name, email, status

FROM Voters

WHERE status = TRUE

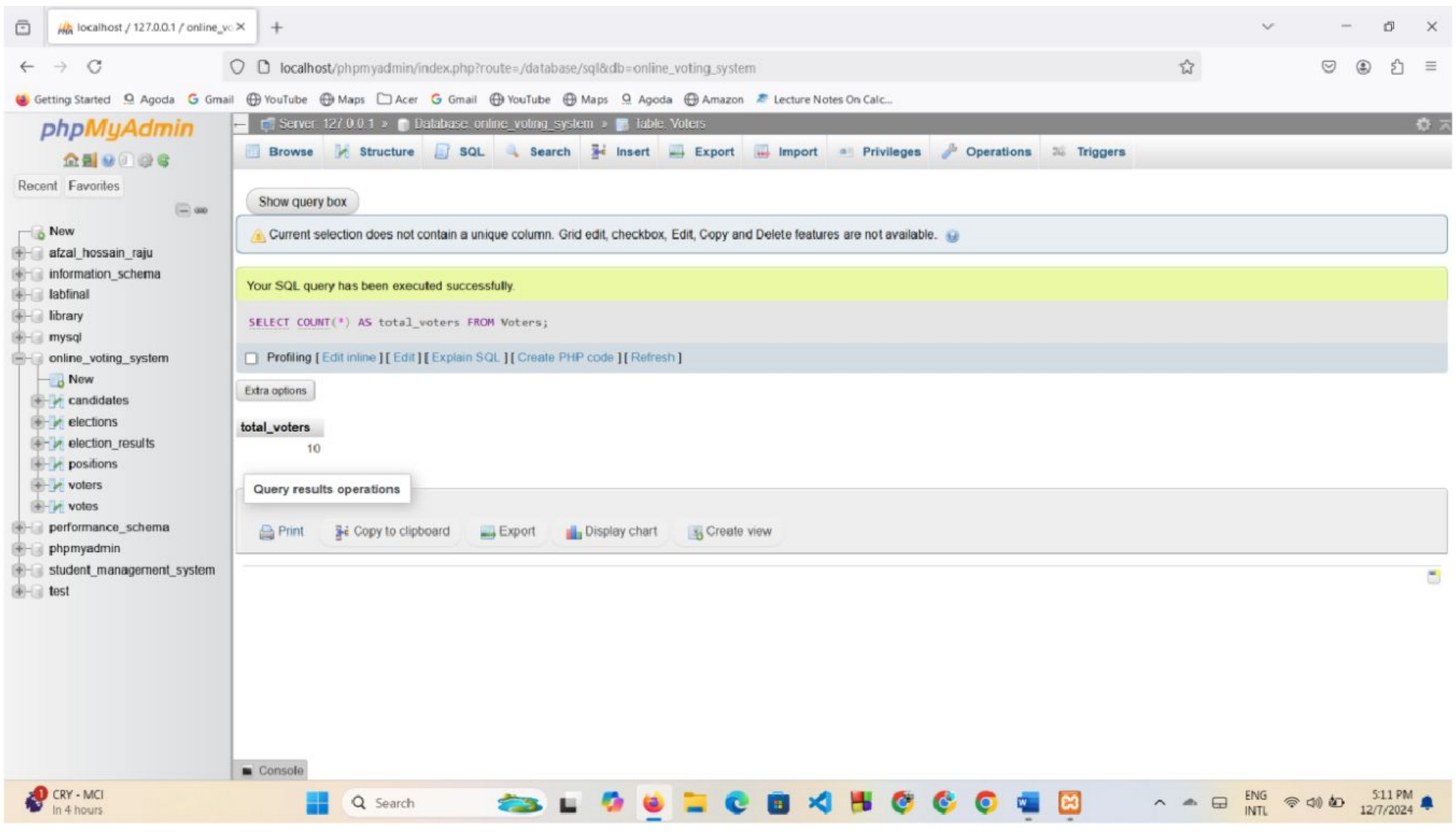
AND (email LIKE '%@example.com' OR email LIKE '%@gmail.com');



8. SELECT with Aggregate Functions:

a) Count the total number of voters

SQL: SELECT COUNT(*) AS total_voters FROM Voters;

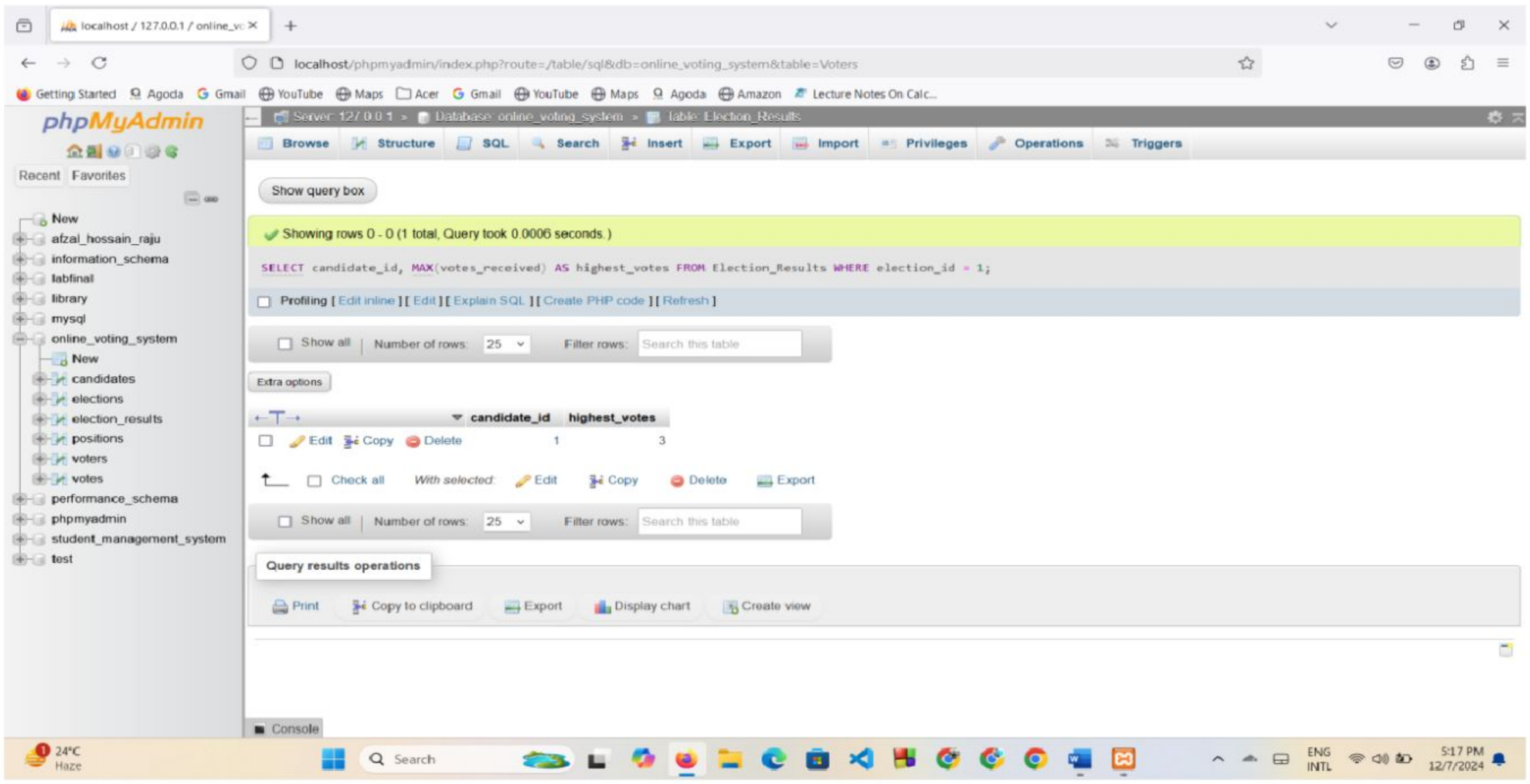


b) Find the candidate with the highest votes in a specific election

SQL: SELECT candidate_id, MAX(votes_received) AS highest_votes

FROM Election_Results

WHERE election_id = 1;



9. SELECT with ORDER BY:

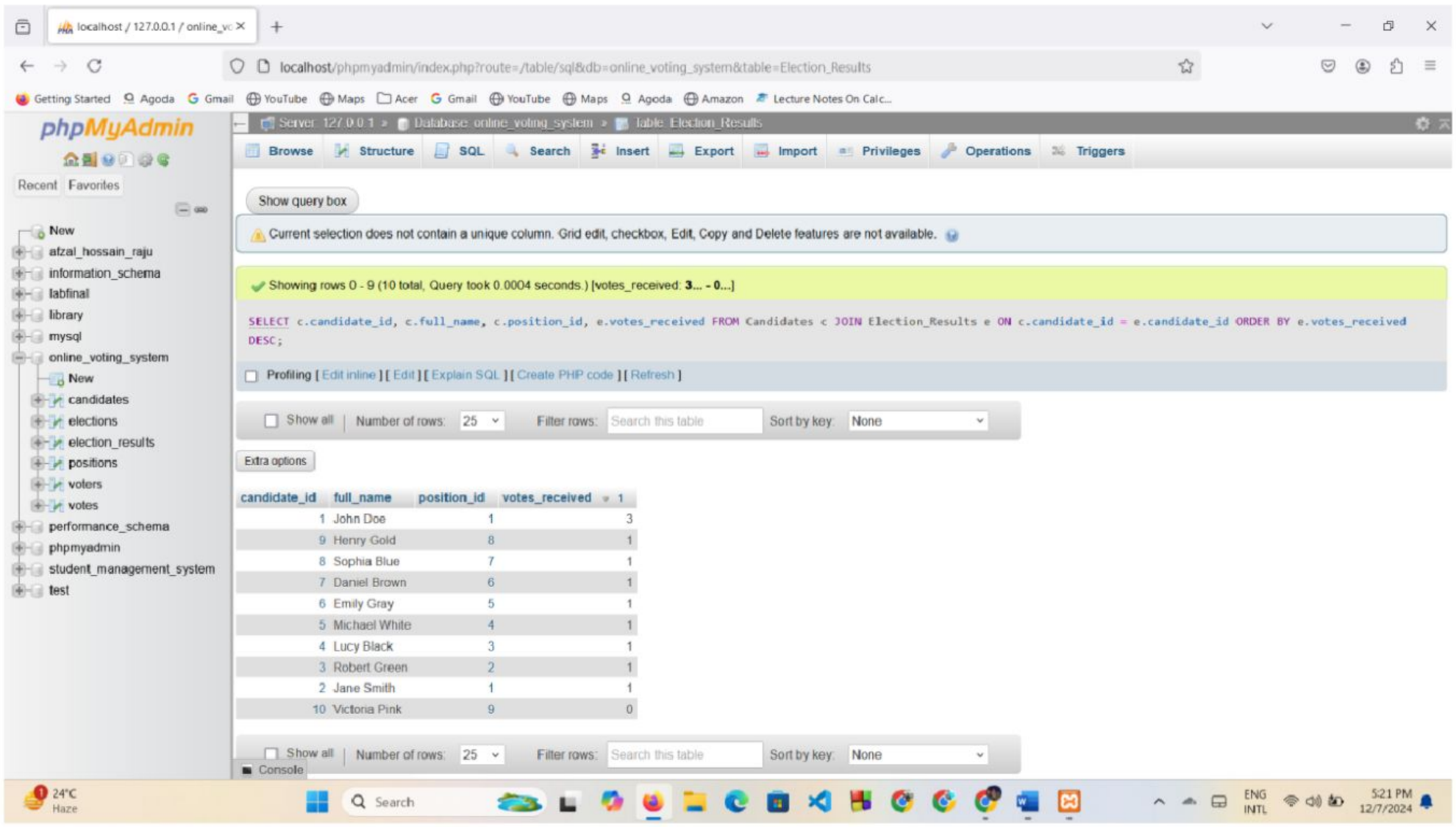
a) Retrieve all candidates sorted by their votes in descending order

SQL: SELECT c.candidate_id, c.full_name, c.position_id, e.votes_received

FROM Candidates c

JOIN Election_Results e ON c.candidate_id = e.candidate_id

ORDER BY e.votes_received DESC;

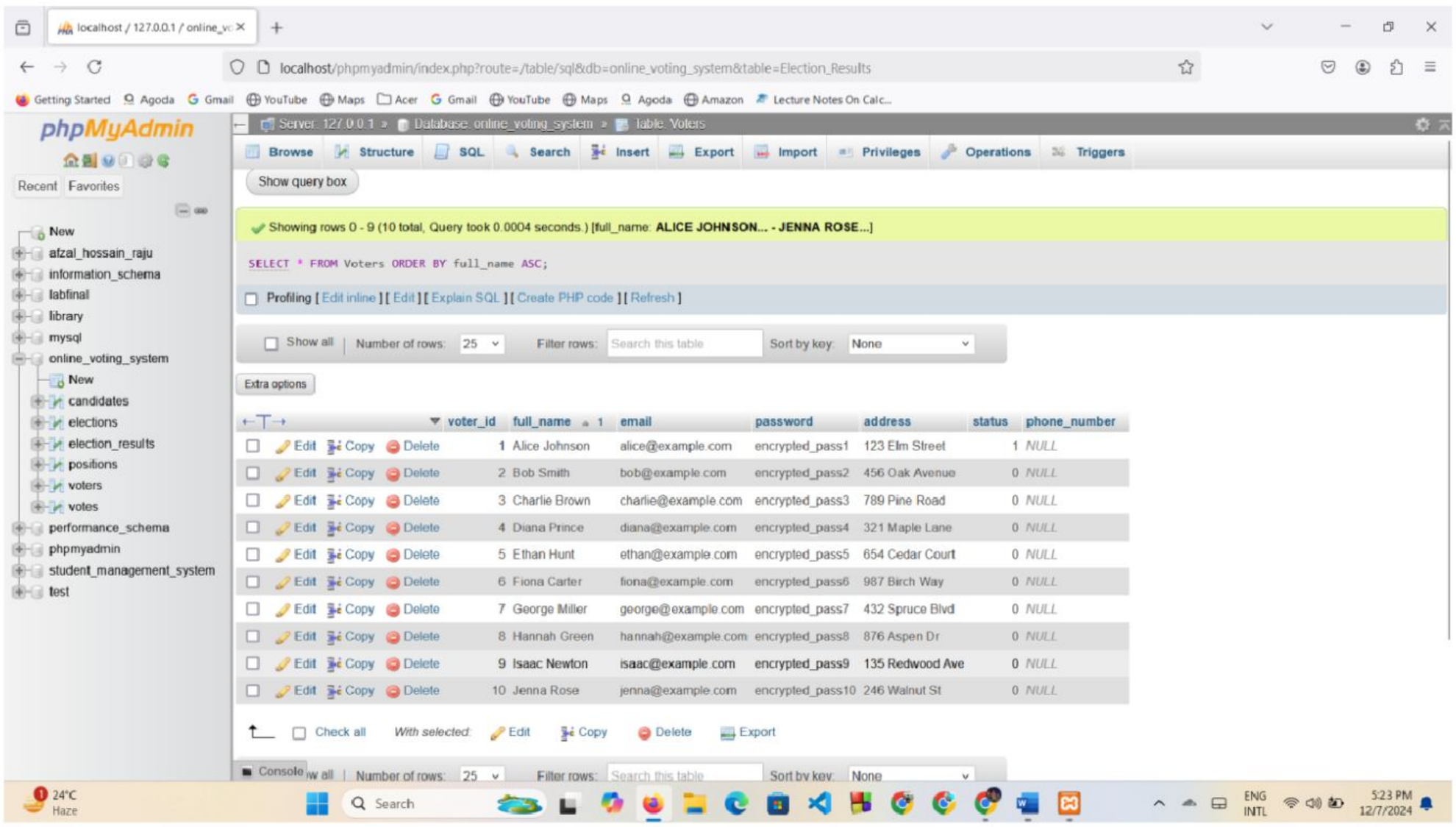


b) Fetch voters sorted by their names alphabetically

SQL: SELECT *

FROM Voters

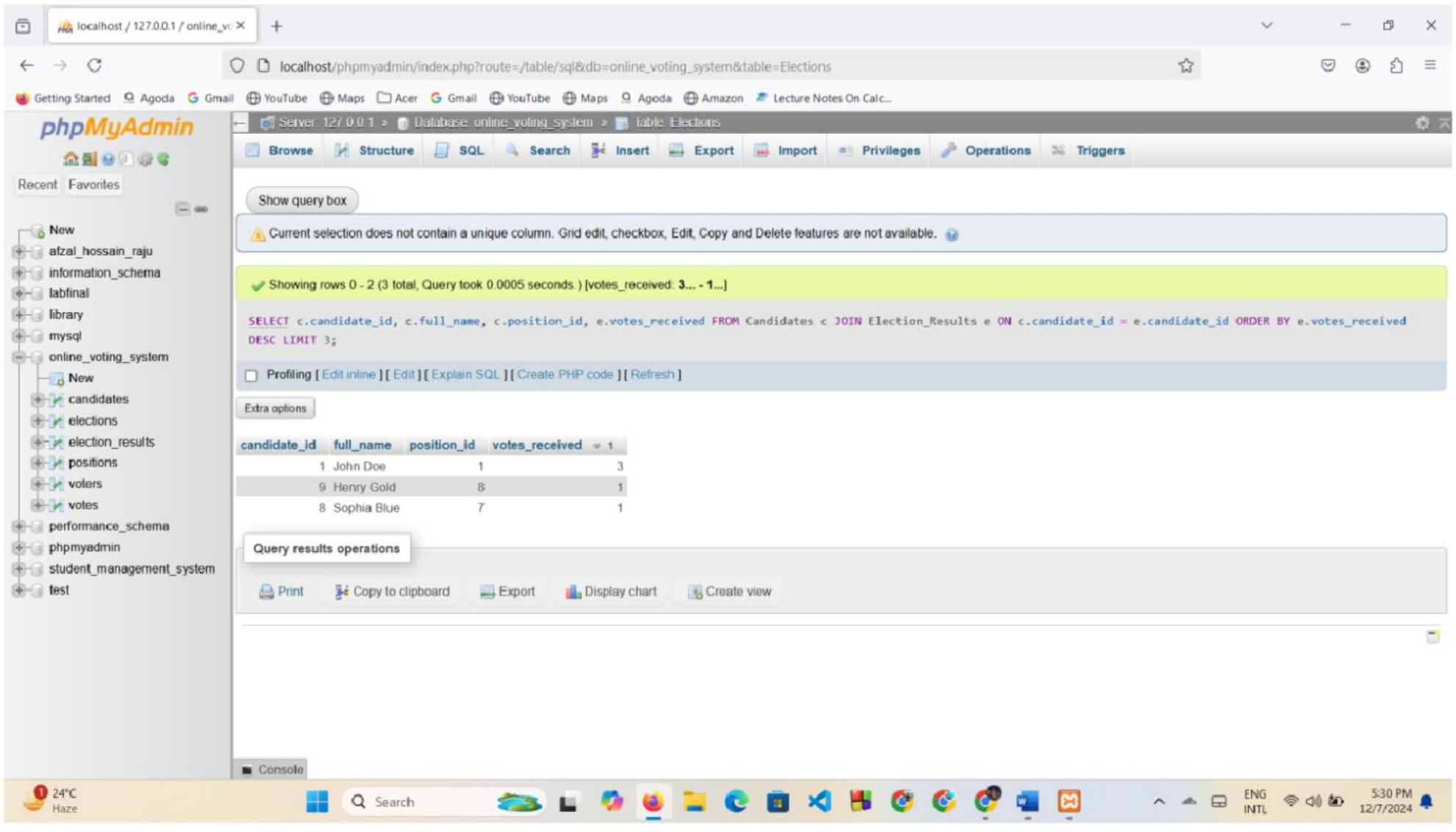
ORDER BY full_name ASC;



10. SELECT with LIMIT:

a) Retrieve the top 3 candidates with the highest votes

```
SELECT c.candidate_id, c.full_name, c.position_id, e.votes_received
FROM Candidates c
JOIN Election_Results e ON c.candidate_id = e.candidate_id
ORDER BY e.votes_received DESC
LIMIT 3;
```



11. MIN and MAX:

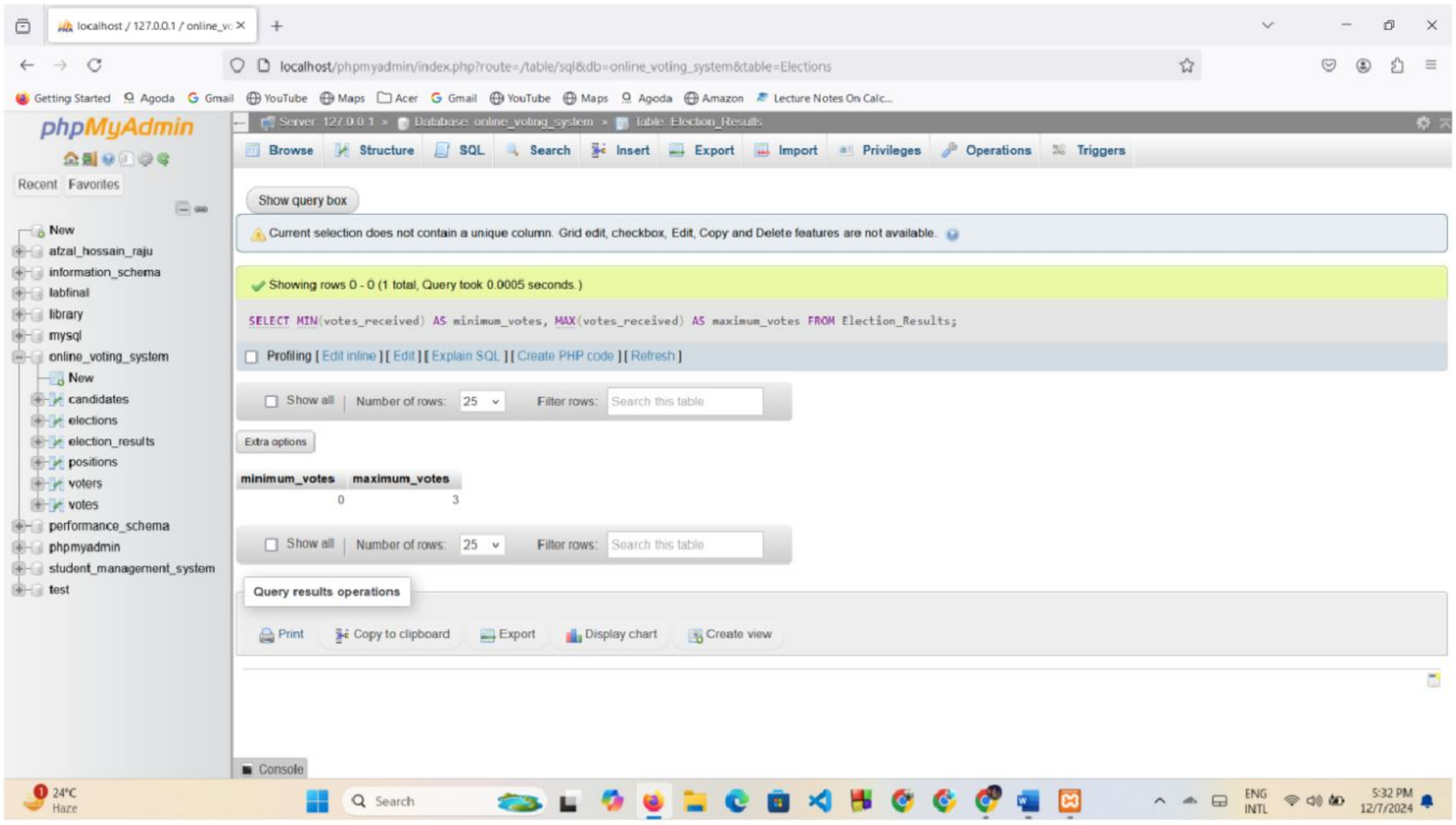
a) Find the minimum and maximum number of votes received by any candidate in the election results

SQL: SELECT

MIN(votes_received) AS minimum_votes,

MAX(votes_received) AS maximum_votes

FROM Election_Results;



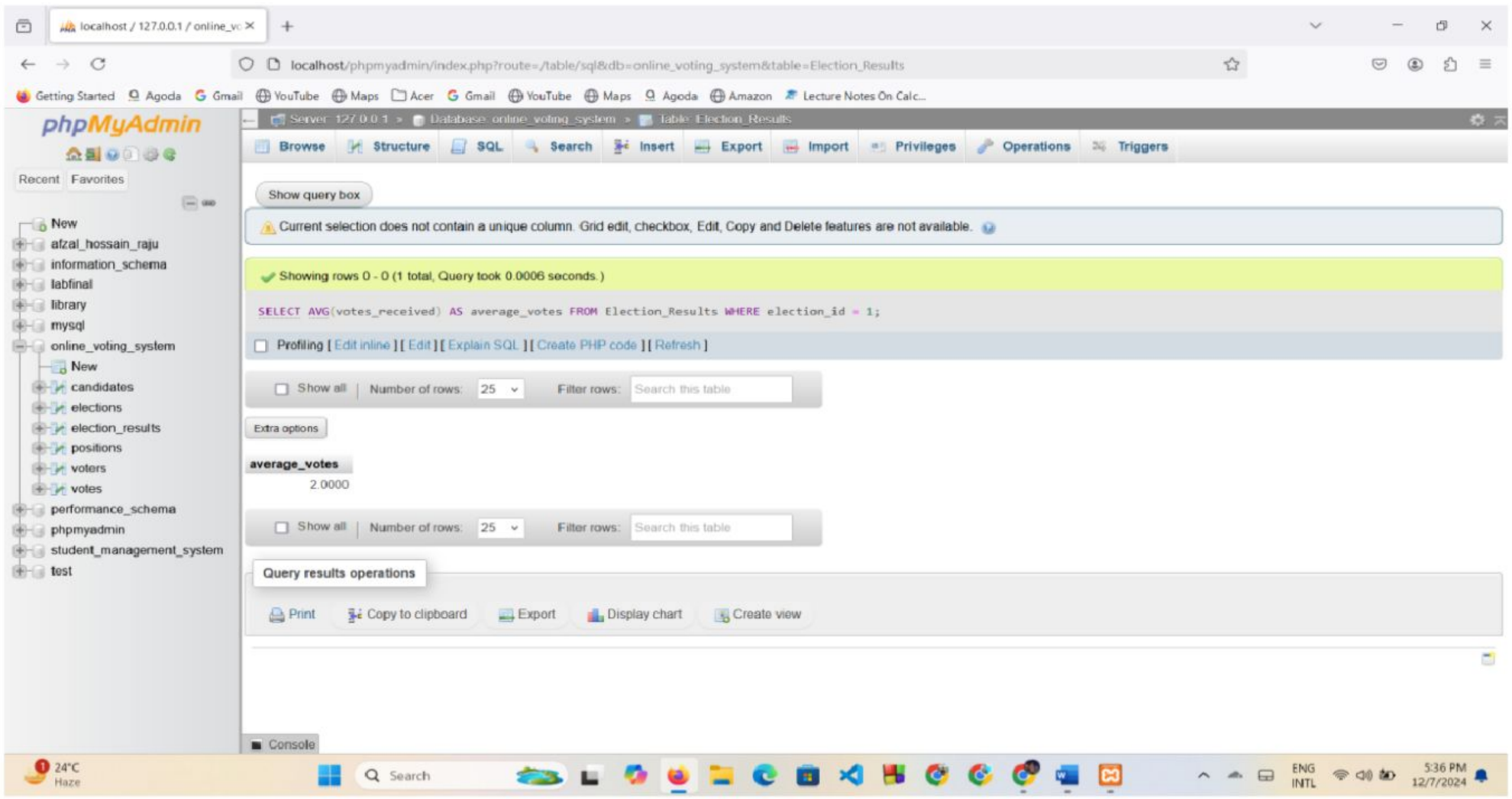
12. AVG (Average)

a) Calculate the average votes received by candidates in a specific election

SQL: SELECT AVG(votes_received) AS average_votes

FROM Election_Results

WHERE election_id = 1;



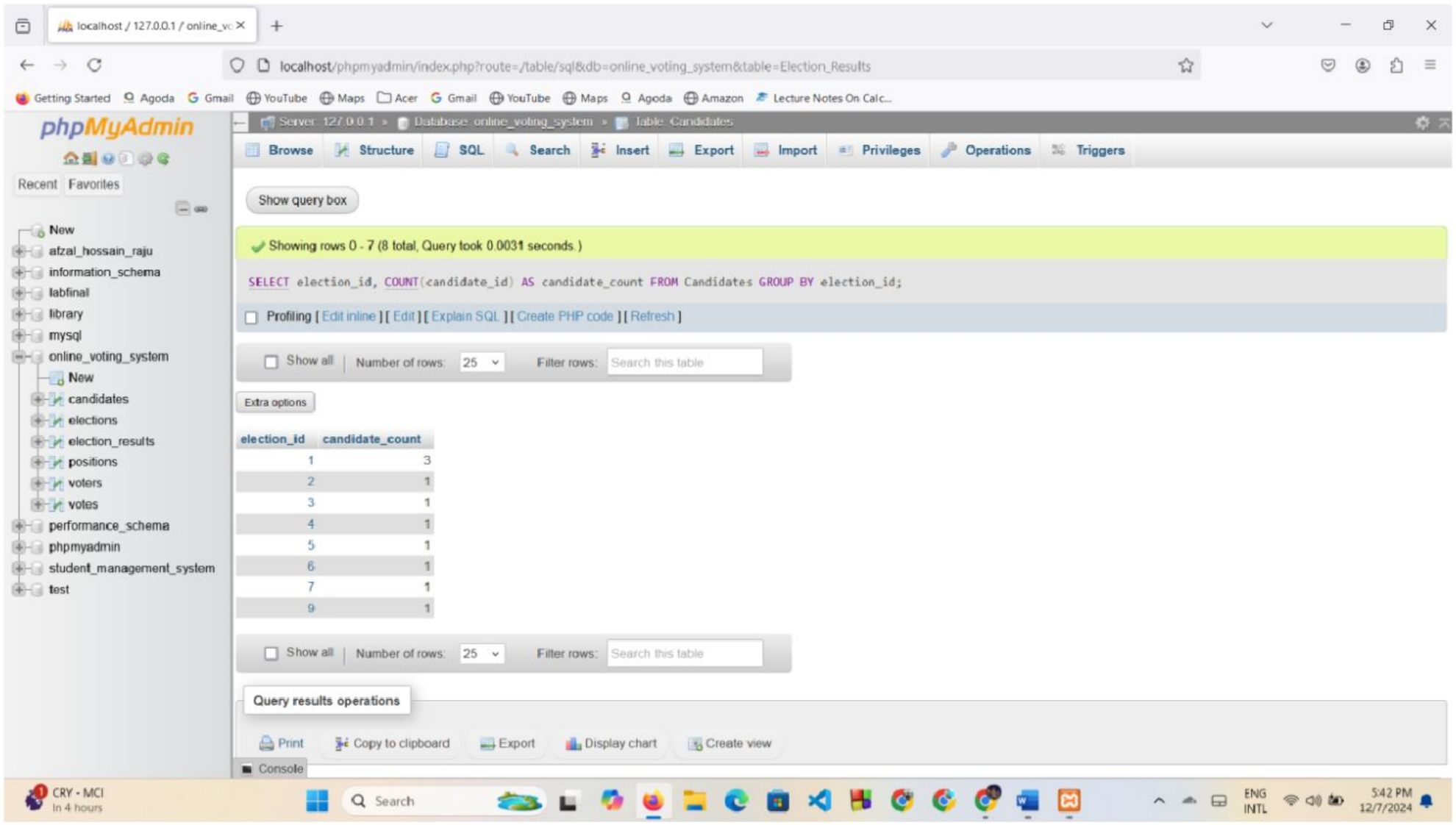
13. GROUP BY

a) Count the number of candidates for each election

SQL: SELECT election_id, COUNT(candidate_id) AS candidate_count

FROM Candidates

GROUP BY election_id;



14. HAVING:

a) Find candidates who received more than 2 votes in total

SQL: SELECT

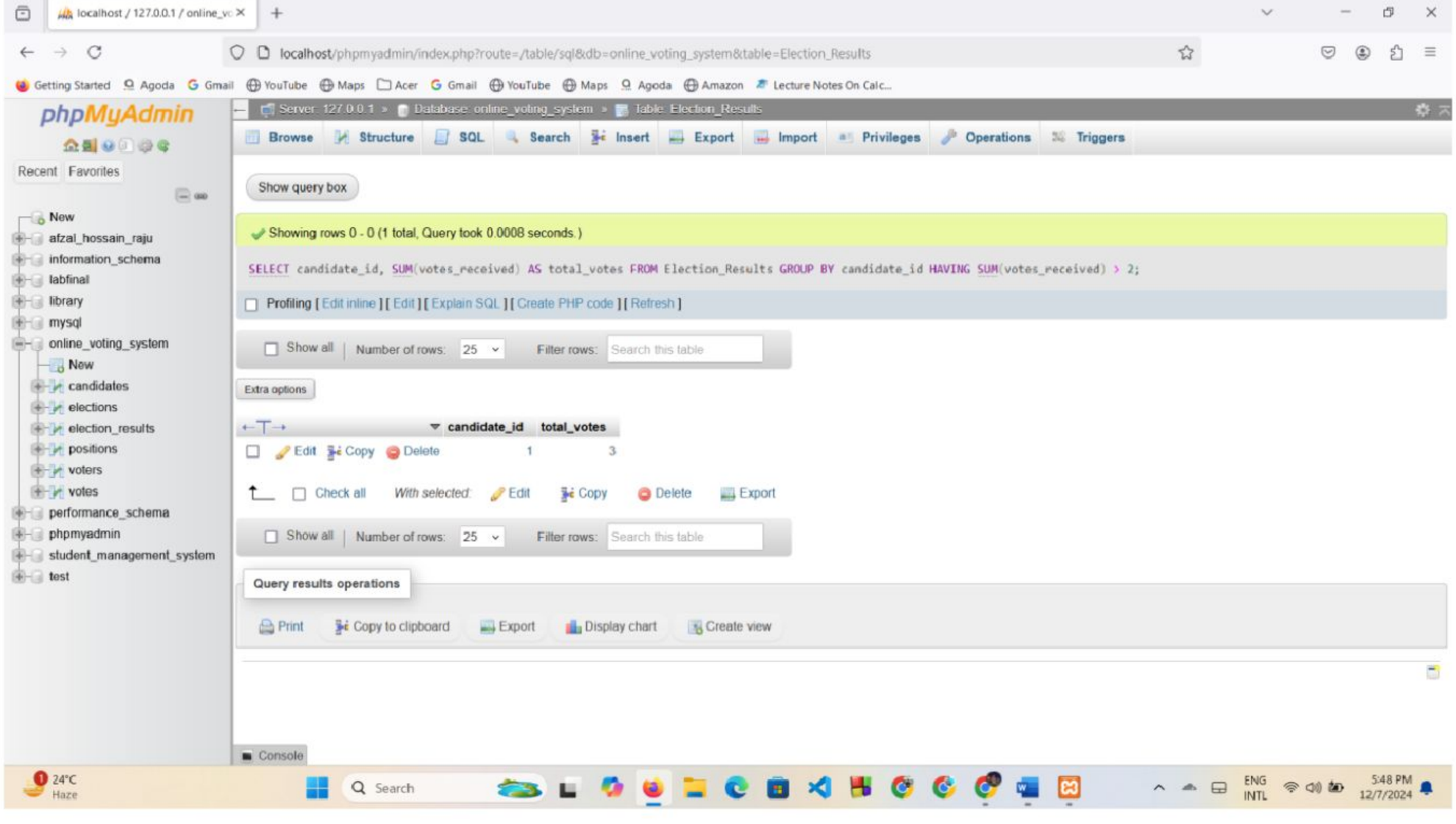
candidate_id,

SUM(votes_received) AS total_votes

FROM Election_Results

GROUP BY candidate_id

HAVING SUM(votes_received) > 2;



15. Combining GROUP BY, HAVING, and Aggregation:

a) Calculate the average votes received by candidates in each election and display only those with an average of more than 1 votes

SQL: SELECT

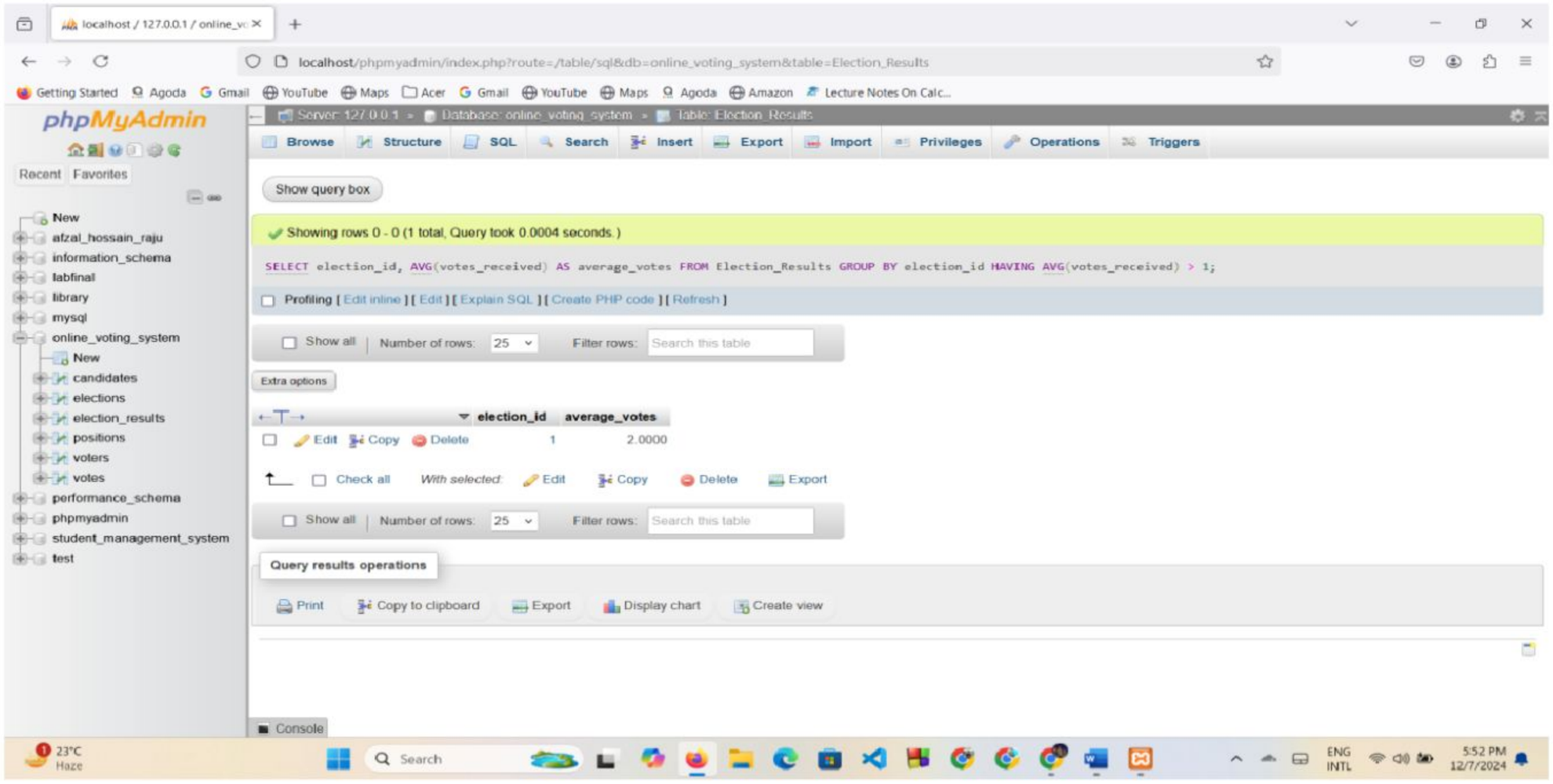
election_id,

AVG(votes_received) AS average_votes

FROM Election_Results

GROUP BY election_id

HAVING AVG(votes_received) > 1;



16. JOIN:

a) INNER JOIN

Fetch voter details along with the candidates they voted for

SQL: SELECT

v.voter_id,

v.full_name AS voter_name,

v.email,

c.candidate_id,

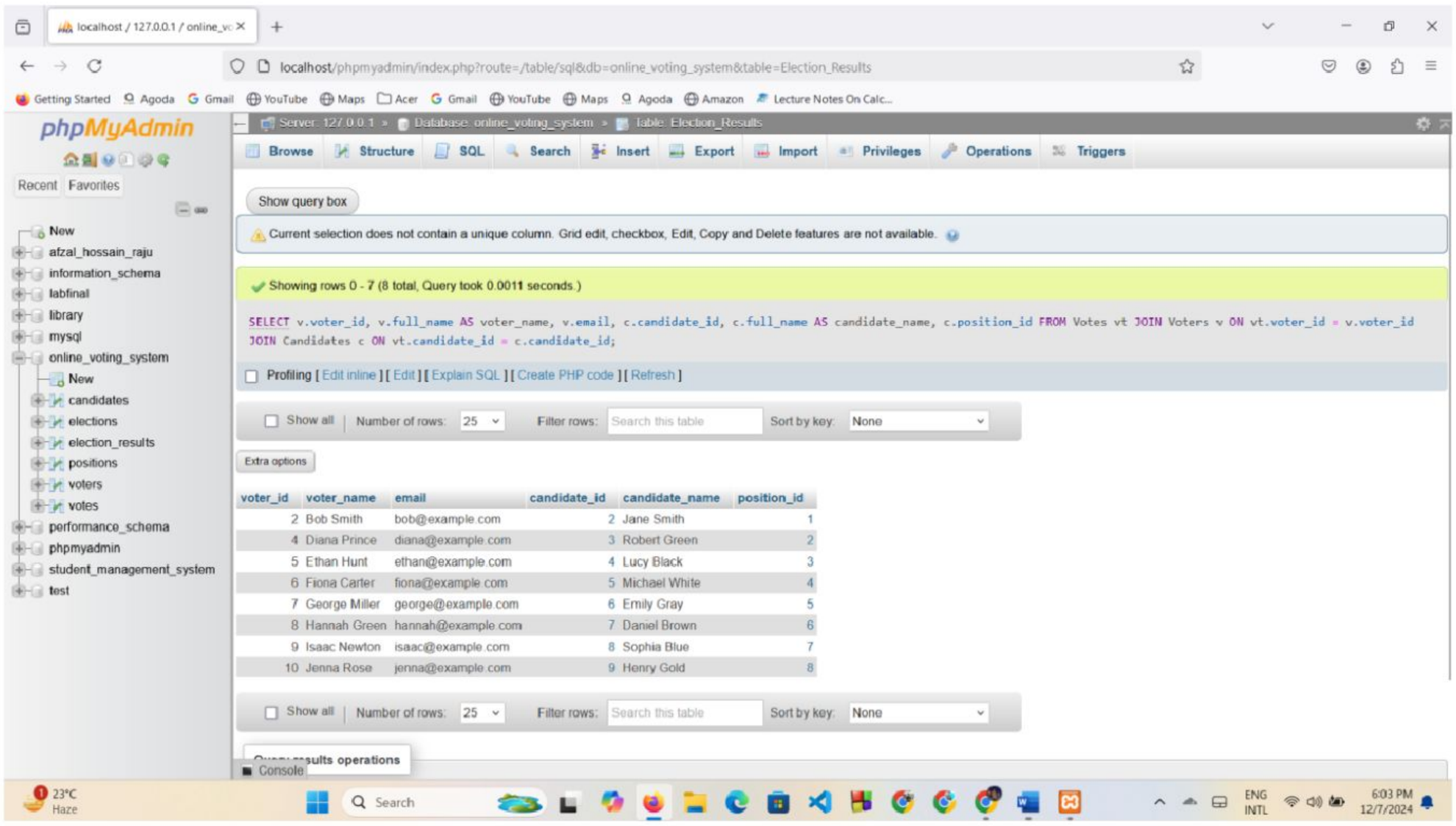
c.full_name AS candidate_name,

c.position_id

FROM Votes vt

JOIN Voters v ON vt.voter_id = v.voter_id

JOIN Candidates c ON vt.candidate_id = c.candidate_id;



b) LEFT JOIN

Retrieve all elections and the candidates participating in them (including elections without candidates)

SQL: SELECT

e.election_id,

e.election_name,

e.start_date,

e.end_date,

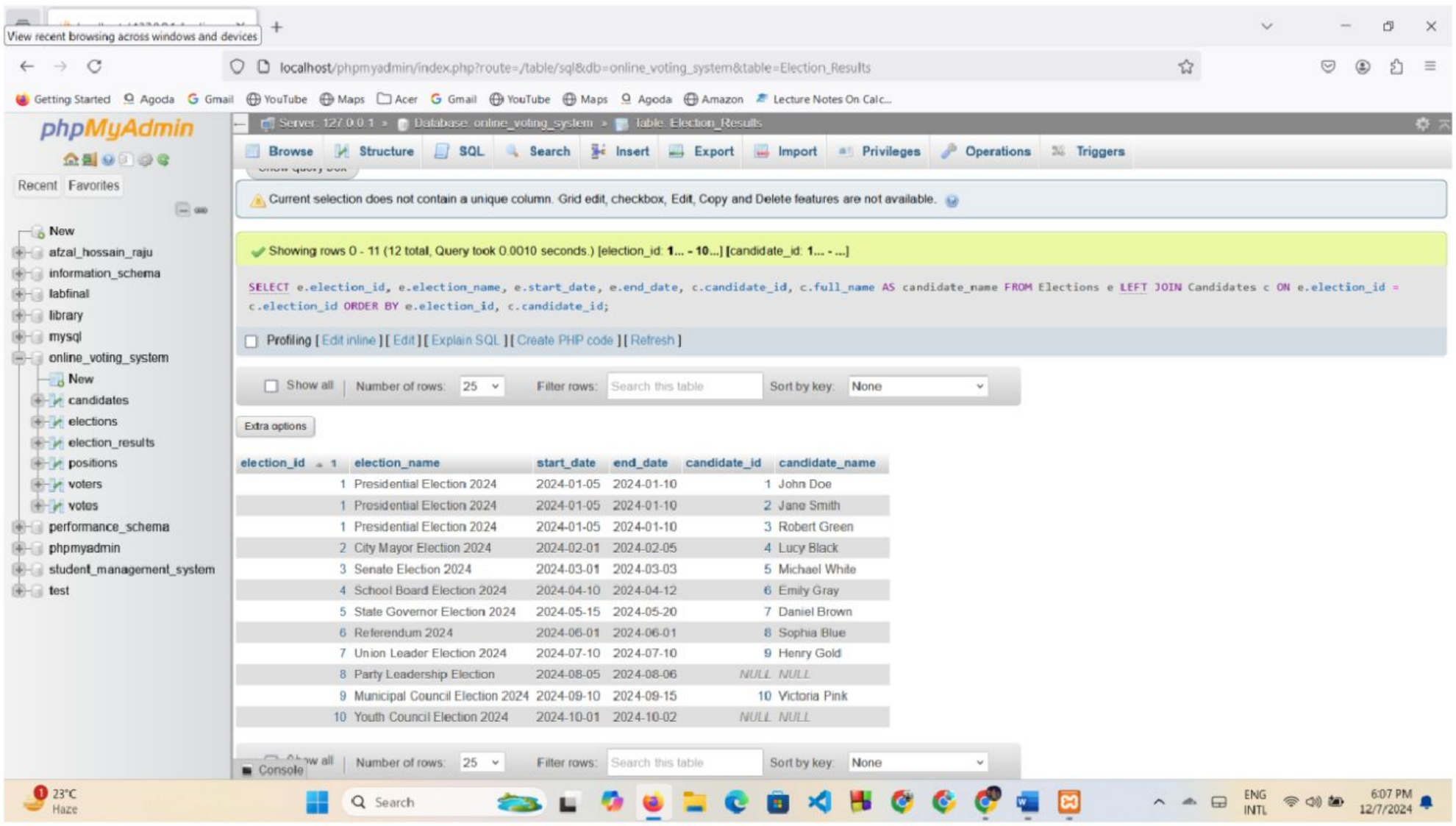
c.candidate_id,

c.full_name AS candidate_name

FROM Elections e

LEFT JOIN Candidates c ON e.election_id = c.election_id

ORDER BY e.election_id, c.candidate_id;



c) RIGHT JOIN

Retrieve all candidates and their respective elections

SQL: SELECT

c.candidate_id,

c.full_name AS candidate_name,

e.election_id,

e.election_name,

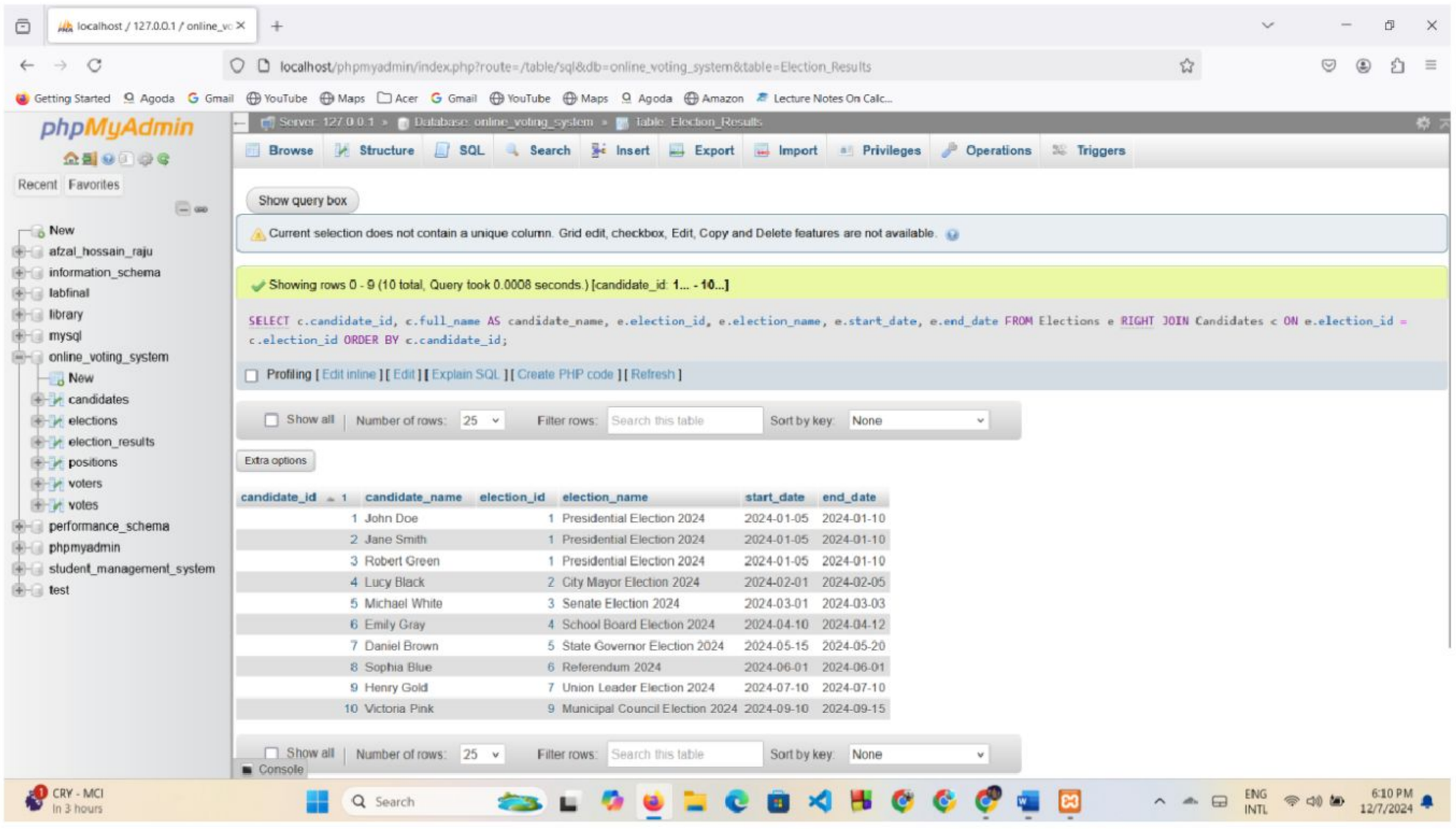
e.start_date,

e.end_date

FROM Elections e

RIGHT JOIN Candidates c ON e.election_id = c.election_id

ORDER BY c.candidate_id;



d) JOIN with Aggregation

Find the total votes received by each candidate with their election details

SQL: SELECT

Candidates.candidate_id,

Candidates.full_name AS candidate_name,

Elections.election_name,

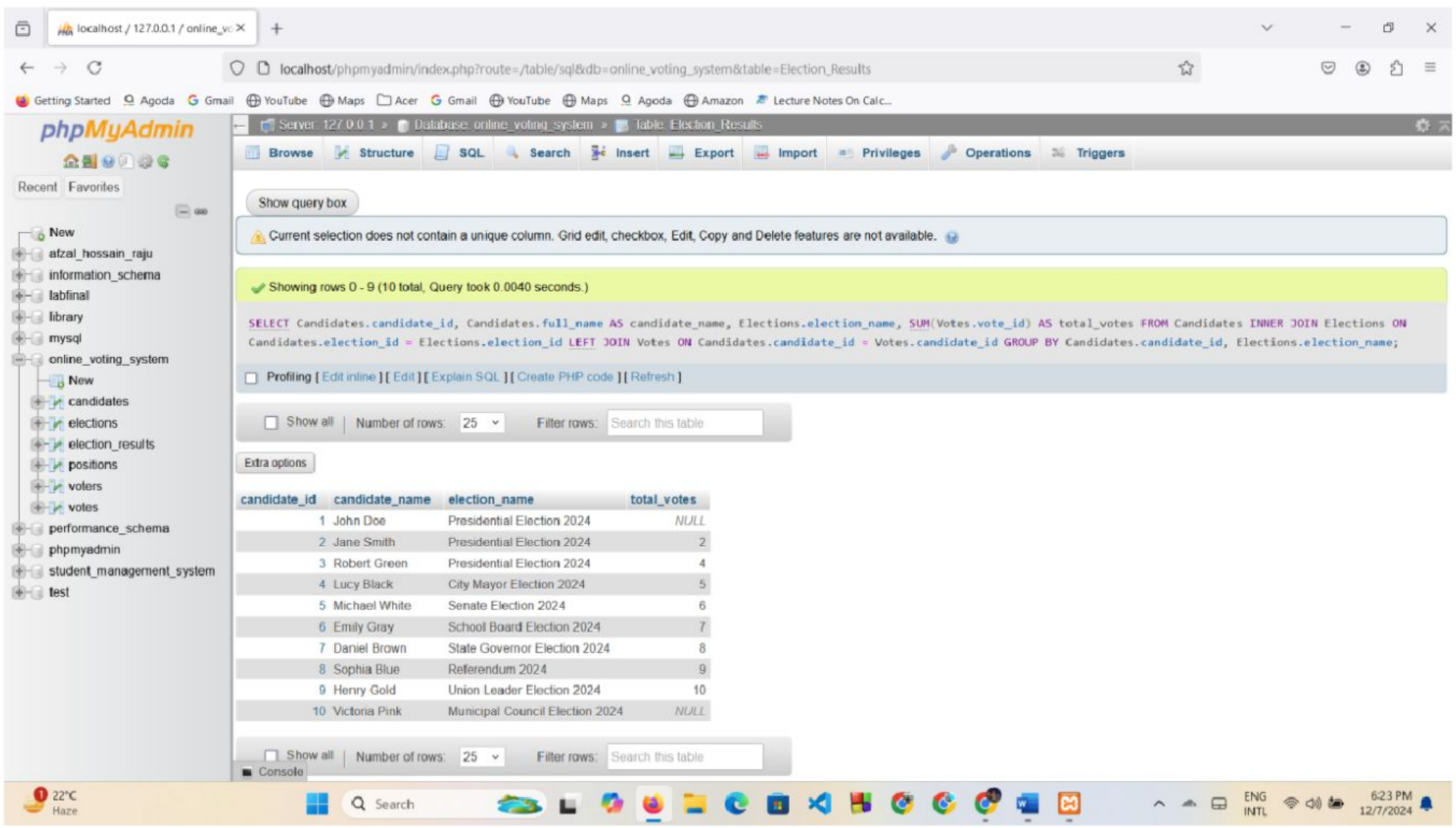
SUM(Votes.vote_id) AS total_votes

FROM Candidates

INNER JOIN Elections ON Candidates.election_id = Elections.election_id

LEFT JOIN Votes ON Candidates.candidate_id = Votes.candidate_id

GROUP BY Candidates.candidate_id, Elections.election_name;



e) JOIN with WHERE Clause

Retrieve voters who voted for a specific candidate

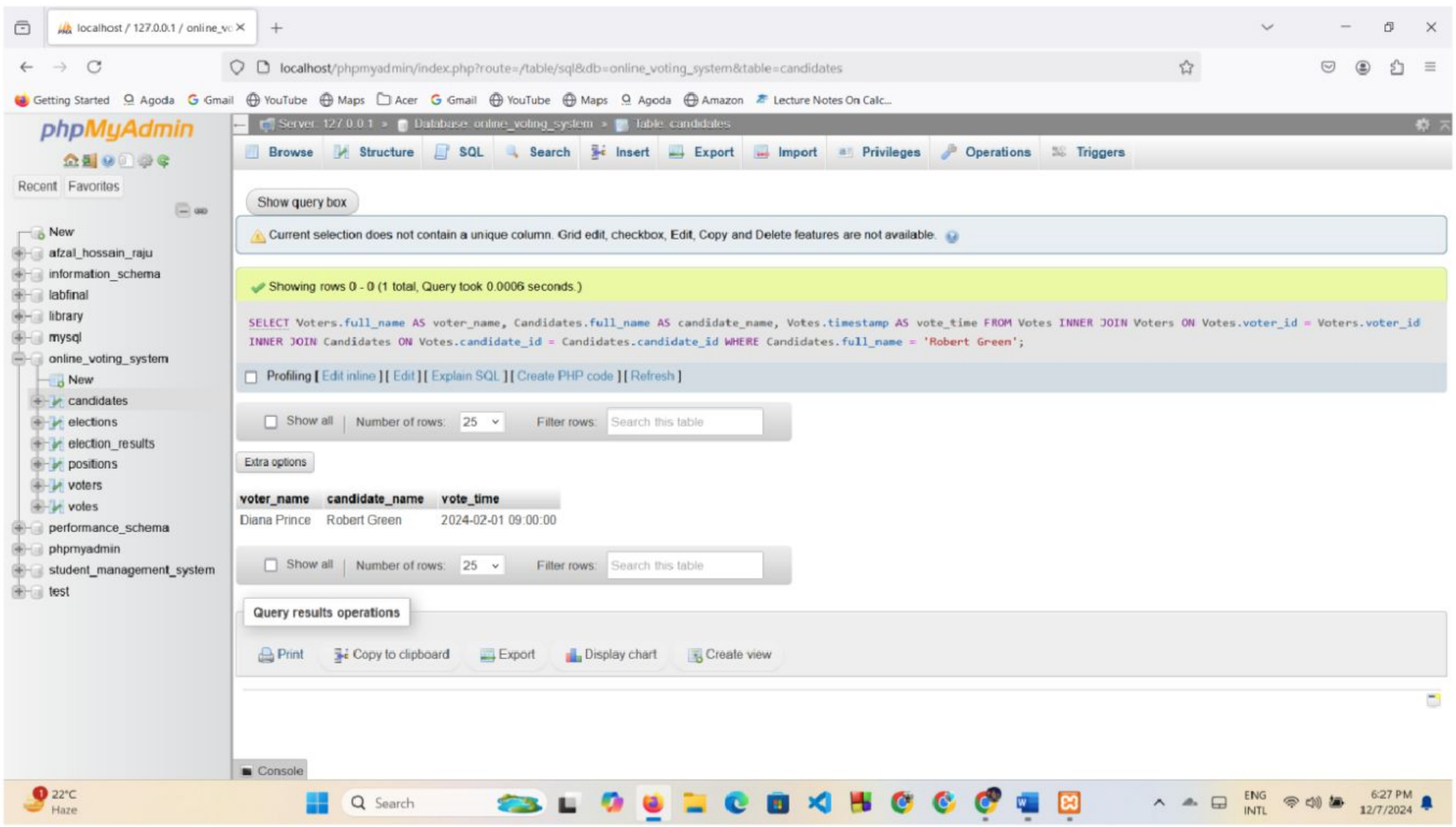
SQL: SELECT Voters.full_name AS voter_name, Candidates.full_name AS candidate_name, Votes.timestamp AS vote_time

FROM Votes

INNER JOIN Voters ON Votes.voter_id = Voters.voter_id INNER

JOIN Candidates ON Votes.candidate_id = Candidates.candidate_id

WHERE Candidates.full_name = 'Robert Green';



17. Subquery:

a)the details of voters who have voted for the candidate that received the highest number of votes in an ongoing election.

SQL: SELECT

v.voter_id,

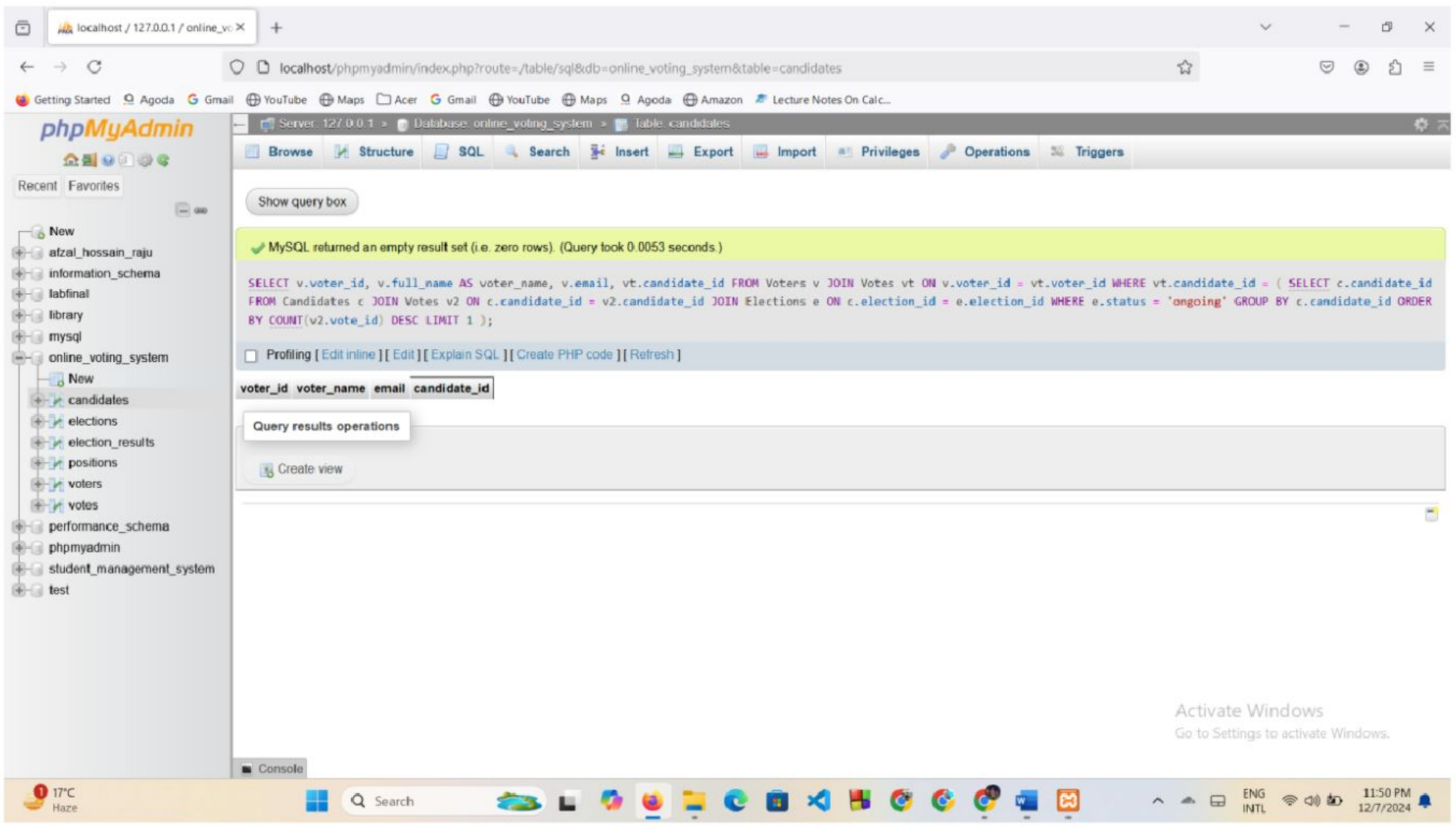
v.full_name AS voter_name,

v.email,

vt.candidate_id

```
FROM Voters v

JOIN Votes vt ON v.voter_id = vt.voter_id
WHERE vt.candidate_id = (
    SELECT c.candidate_id
    FROM Candidates c
    JOIN Votes v2 ON c.candidate_id = v2.
    JOIN Elections e ON c.election_id = e.election_id
    WHERE e.status = 'ongoing'
    GROUP BY c.candidate_id
    ORDER BY COUNT(v2.vote_id) DESC
    LIMIT 1
);
```



18. Create view:

a) 16(e) create as view.

SQL: CREATE VIEW Raju1031 AS

(SELECT Voters.full_name AS voter_name, Candidates.full_name AS candidate_name, Votes.timestamp AS vote_time

FROM Votes INNER JOIN Voters ON Votes.voter_id = Voters.voter_id INNER

JOIN Candidates ON Votes.candidate_id = Candidates.candidate_id

WHERE Candidates.full_name = 'Robert Green');