



Daffodil
International
University

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:
- Basic SELECT Operations:
 - 4.a) Remove the party column from the Candidates table
 - 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))

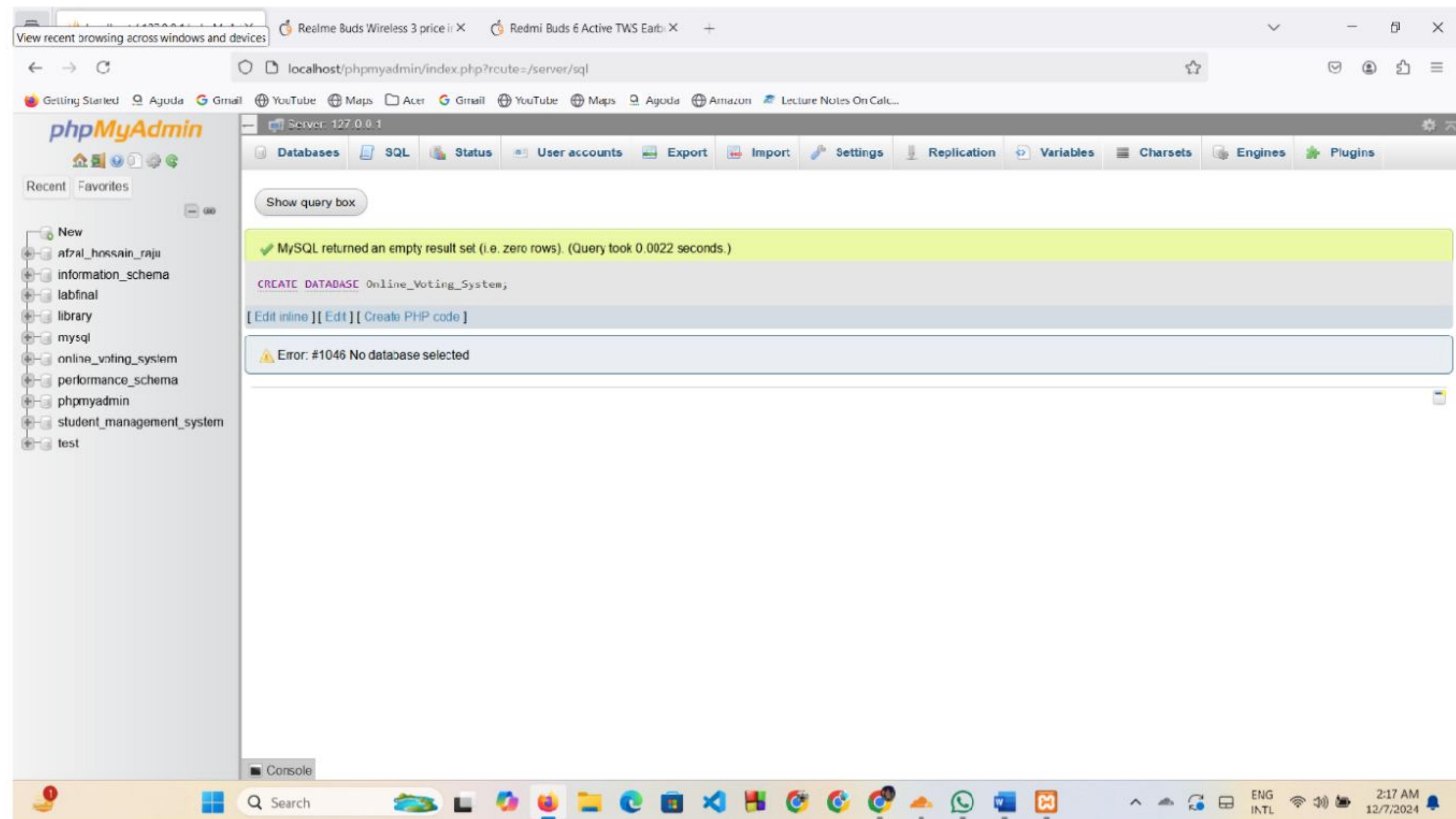
Election results (result_id (primary key, auto increment) election_id (foreign key) position_id (foreign key) candidate_id (foreign key) votes_received (integer))

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

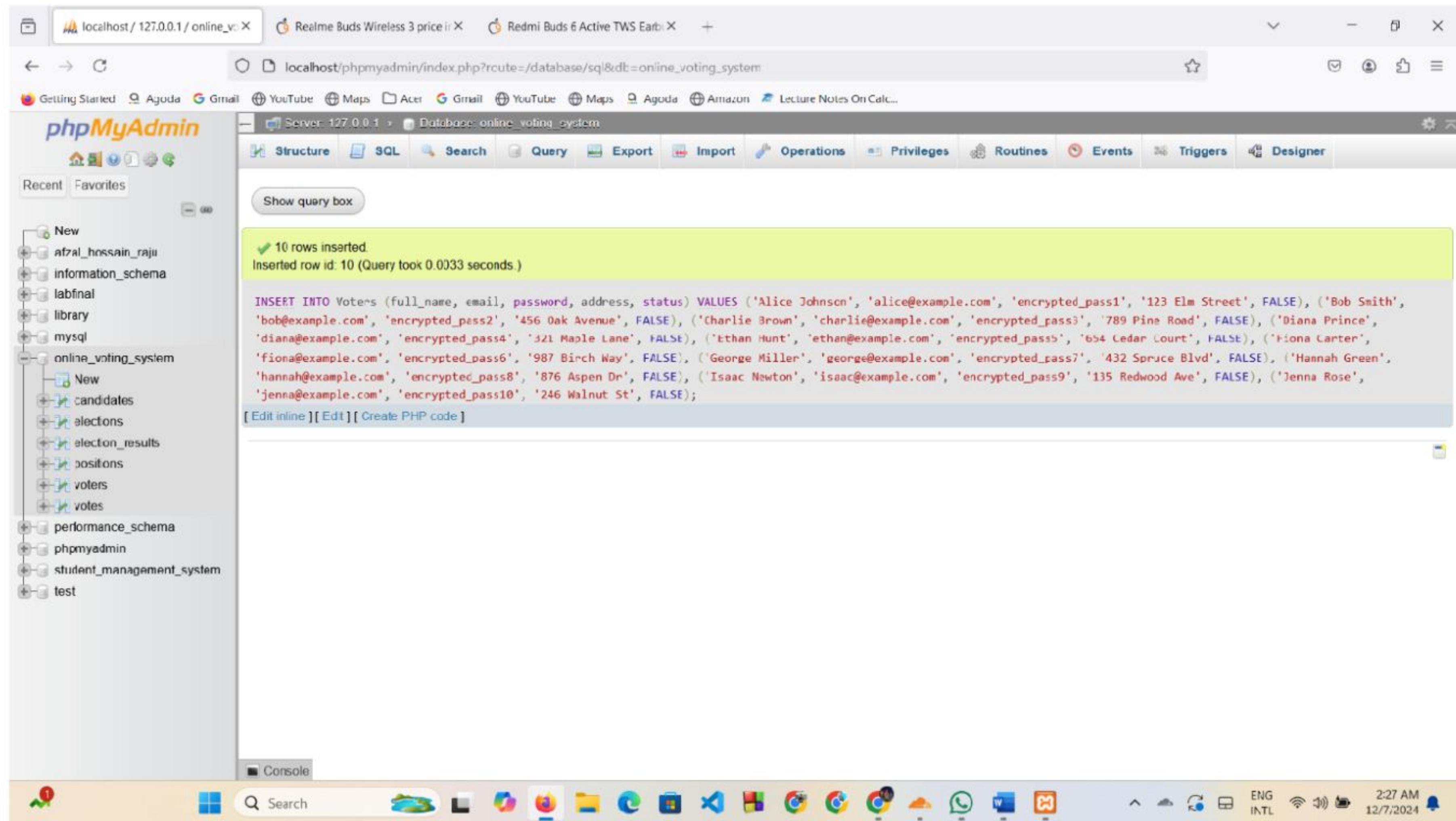
The screenshot shows the phpMyAdmin interface connected to a MySQL database named 'online_voting_system'. The left sidebar lists various databases and their structures. The main area displays the SQL queries for creating six tables:

- Voters:** A table with columns: voter_id (INT, primary key, auto-increment), full_name (VARCHAR(255), NOT NULL), email (VARCHAR(255), UNIQUE, NOT NULL), password (VARCHAR(255), NOT NULL), address (VARCHAR(255)), and status (BOOLEAN, default FALSE).
- Elections:** A table with columns: election_id (INT, primary key, auto-increment), election_name (VARCHAR(255), NOT NULL), start_date (DATE, NOT NULL), end_date (DATE, NOT NULL), and status (ENUM('upcoming', 'ongoing', 'completed'), NOT NULL).
- Positions:** A table with columns: position_id (INT, primary key, auto-increment), position_name (VARCHAR(255), NOT NULL), and election_id (INT, NOT NULL, foreign key referencing Elections.election_id).
- Candidates:** A table with columns: candidate_id (INT, primary key, auto-increment), full_name (VARCHAR(255), NOT NULL), party (VARCHAR(255)), position_id (INT, NOT NULL, foreign key referencing Positions.position_id), and election_id (INT, NOT NULL, foreign key referencing Elections.election_id).
- Votes:** A table with columns: vote_id (INT, primary key, auto-increment), voter_id (INT, NOT NULL, foreign key referencing Voters.voter_id), election_id (INT, NOT NULL, foreign key referencing Elections.election_id), candidate_id (INT, NOT NULL, foreign key referencing Candidates.candidate_id), timestamp (DATETIME, DEFAULT CURRENT_TIMESTAMP), and FOREIGN KEY constraints.
- Election_Results:** A table with columns: result_id (INT, primary key, auto-increment), election_id (INT, NOT NULL, foreign key referencing Elections.election_id), position_id (INT, NOT NULL, foreign key referencing Positions.position_id), candidate_id (INT, NOT NULL, foreign key referencing Candidates.candidate_id), and votes_received (INT, default 0).

Each query is followed by a success message indicating an empty result set and the execution time.

Insert data into all tables:

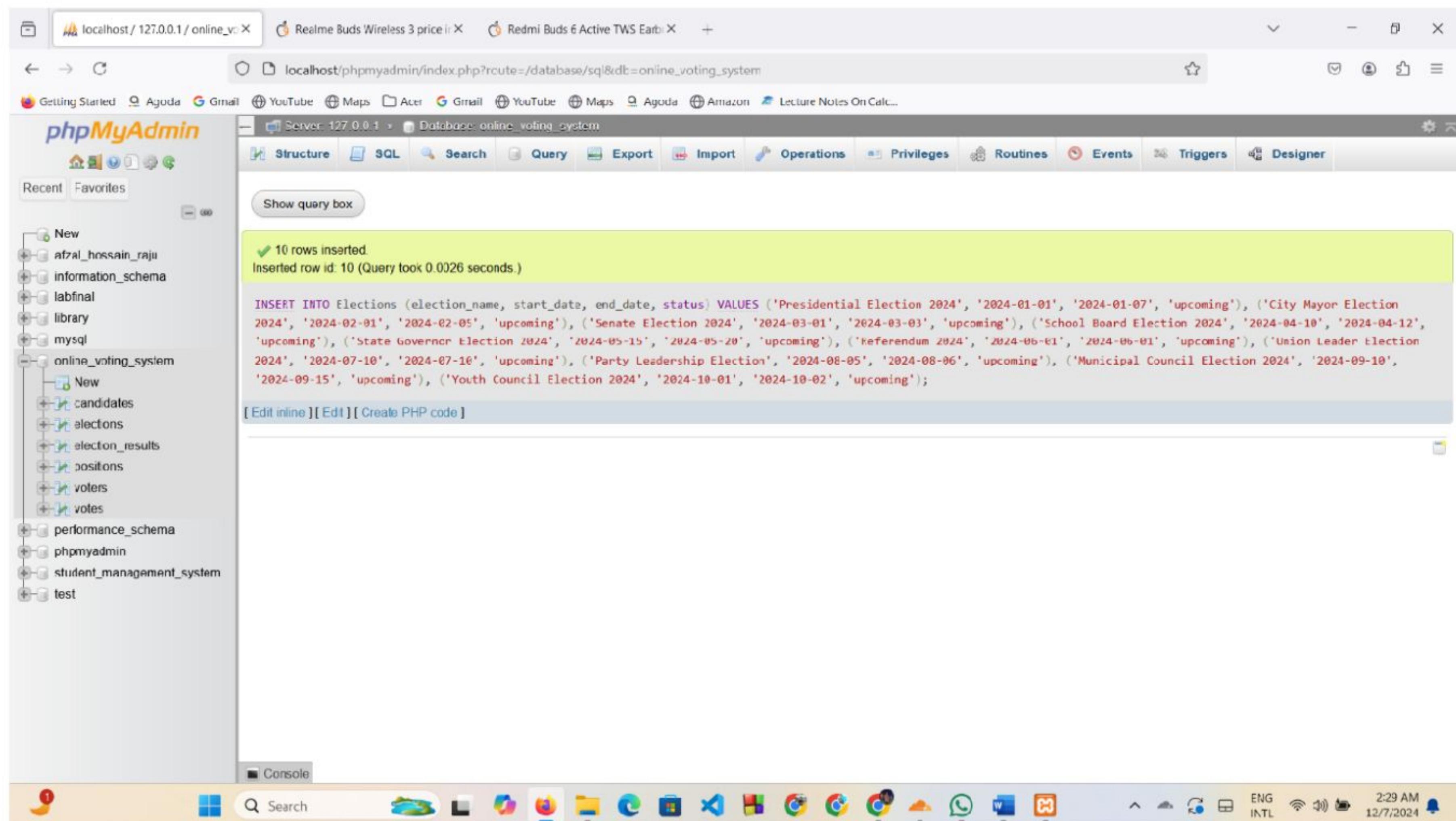
1. Insert into table voters:



The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database. The 'voters' table is selected. A SQL query has been run, resulting in 10 rows being inserted. The message '10 rows inserted.' is displayed in a green box, along with the query ID (10) and execution time (0.0033 seconds). The query itself is a multi-line INSERT INTO statement for the 'Voters' table, listing 10 different voter records with their names, emails, encrypted passwords, addresses, and status.

```
INSERT INTO Voters (full_name, email, password, address, status) VALUES ('Alice Johnson', 'alice@example.com', 'encrypted_pass1', '123 Elm Street', FALSE), ('Bob Smith', 'bob@example.com', 'encrypted_pass2', '456 Oak Avenue', FALSE), ('Charlie Brown', 'charlie@example.com', 'encrypted_pass3', '789 Pine Road', FALSE), ('Diana Prince', 'diana@example.com', 'encrypted_pass4', '321 Maple Lane', FALSE), ('Ethan Hunt', 'ethan@example.com', 'encrypted_pass5', '654 Cedar Court', FALSE), ('Fiona Carter', 'fiona@example.com', 'encrypted_pass6', '987 Birch Way', FALSE), ('George Miller', 'george@example.com', 'encrypted_pass7', '432 Spruce Blvd', FALSE), ('Hannah Green', 'hannah@example.com', 'encrypted_pass8', '876 Aspen Dr', FALSE), ('Isaac Newton', 'isaac@example.com', 'encrypted_pass9', '135 Redwood Ave', FALSE), ('Jenna Rose', 'jenna@example.com', 'encrypted_pass10', '246 Walnut St', FALSE);
```

2. Insert into Election:



The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database. The 'elections' table is selected. A SQL query has been run, resulting in 10 rows being inserted. The message '10 rows inserted.' is displayed in a green box, along with the query ID (10) and execution time (0.0026 seconds). The query is a multi-line INSERT INTO statement for the 'Elections' table, listing 10 different election entries with their names, start dates, end dates, and statuses.

```
INSERT INTO Elections (election_name, start_date, end_date, status) VALUES ('Presidential Election 2024', '2024-01-01', '2024-01-07', 'upcoming'), ('City Mayor Election 2024', '2024-02-01', '2024-02-05', 'upcoming'), ('Senate Election 2024', '2024-03-01', '2024-03-03', 'upcoming'), ('School Board Election 2024', '2024-04-10', '2024-04-12', 'upcoming'), ('State Governor Election 2024', '2024-05-15', '2024-05-20', 'upcoming'), ('Referendum 2024', '2024-06-01', '2024-06-01', 'upcoming'), ('Union Leader Election 2024', '2024-07-10', '2024-07-10', 'upcoming'), ('Party Leadership Election', '2024-08-05', '2024-08-06', 'upcoming'), ('Municipal Council Election 2024', '2024-09-10', '2024-09-15', 'upcoming'), ('Youth Council Election 2024', '2024-10-01', '2024-10-02', 'upcoming');
```

3. Insert into Positions:

The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database. The 'Structure' tab is selected. In the SQL query box, the following SQL code was run:

```
INSERT INTO Positions (position_name, election_id) VALUES ('President', 1), ('Vice President', 1), ('Mayor', 2), ('Senator', 3), ('School Board Member', 4), ('Governor', 5), ('Referendum', 6), ('Union Leader', 7), ('Party Leader', 8), ('Council Member', 9);
```

The results show a green success message: "10 rows inserted. Inserted row id: 10 (Query took 0.0023 seconds.)". The status bar at the bottom right indicates the date and time as 12/7/2024 2:31 AM.

4. Insert into Candidates:

The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database. The 'Structure' tab is selected. In the SQL query box, the following SQL code was run:

```
INSERT INTO Candidates (full_name, party, position_id, election_id) VALUES ('John Doe', 'Party A', 1, 1), ('Jane Smith', 'Party B', 1, 1), ('Robert Green', 'Party C', 2, 1), ('Lucy Black', 'Party A', 3, 2), ('Michael White', 'Party E', 4, 3), ('Emily Gray', 'Party C', 5, 4), ('Daniel Brown', 'Party A', 6, 5), ('Sophia Blue', 'Independent', 7, 6), ('Henry Gold', 'Party A', 8, 7), ('Victoria Pink', 'Party E', 9, 9);
```

The results show a green success message: "10 rows inserted. Inserted row id: 10 (Query took 0.0060 seconds.)". The status bar at the bottom right indicates the date and time as 12/7/2024 2:32 AM.

5. Insert into Votes:

The screenshot shows the phpMyAdmin interface connected to a MySQL database named 'online_voting_system'. The 'Votes' table is selected. A SQL query has been run, resulting in 10 rows being inserted. The message '10 rows inserted. Inserted row id: 10 (Query took 0.0032 seconds.)' is displayed in a green box. The SQL query itself is:

```
INSERT INTO Votes (voter_id, election_id, candidate_id, timestamp) VALUES (1, 1, 1, '2024-01-01 10:00:00'), (2, 1, 2, '2024-01-01 10:15:00'), (3, 1, 1, '2024-01-01 10:30:00'), (4, 2, 3, '2024-02-01 09:00:00'), (5, 3, 4, '2024-03-01 11:00:00'), (6, 4, 5, '2024-04-10 08:45:00'), (7, 5, 6, '2024-05-15 14:00:00'), (8, 6, 7, '2024-06-01 12:00:00'), (9, 7, 8, '2024-07-10 16:30:00'), (10, 9, 9, '2024-09-10 10:20:00');
```

6. Insert into Election Results:

The screenshot shows the phpMyAdmin interface connected to a MySQL database named 'online_voting_system'. The 'Election_Results' table is selected. A SQL query has been run, resulting in 10 rows being inserted. The message '10 rows inserted. Inserted row id: 10 (Query took 0.0019 seconds.)' is displayed in a green box. The SQL query itself is:

```
INSERT INTO Election_Results (election_id, position_id, candidate_id, votes_received) VALUES (1, 1, 1, 2), (1, 1, 2, 1), (2, 3, 3, 1), (3, 4, 4, 1), (4, 5, 5, 1), (5, 6, 6, 1), (6, 7, 7, 1), (7, 8, 8, 1), (9, 9, 9, 1), (9, 9, 10, 0);
```

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);

The screenshot shows two instances of the phpMyAdmin interface. The top instance is for the 'Elections' table, and the bottom instance is for the 'Voters' table.

Elections Table (Top):

- Query: `DELETE FROM Voters WHERE voter_id NOT IN (SELECT DISTINCT voter_id FROM Votes);`
- Result: 0 rows deleted. (Query took 0.0066 seconds.)

Voters Table (Bottom):

- Query: `SELECT * FROM Voters;`
- Result: Showing rows 0 - 9 (10 total, Query took 0.0013 seconds.)
- Data:

voter_id	full_name	email	password	address	status
1	Alice Johnson	alice@example.com	encrypted_pass1	123 Elm Street	1
2	Bob Smith	bob@example.com	encrypted_pass2	456 Oak Avenue	0
3	Charlie Brown	charlie@example.com	encrypted_pass3	789 Pine Road	0
4	Diane Prince	diana@example.com	encrypted_pass4	321 Maple Lane	0
5	Ethan Hurt	ethan@example.com	encrypted_pass5	654 Cedar Court	0
6	Fiona Carter	fiona@example.com	encrypted_pass6	987 Birch Way	0
7	George Miller	george@example.com	encrypted_pass7	432 Spruce Blvd	0
8	Hannah Green	hannah@example.com	encrypted_pass8	875 Aspen Dr	0
9	Isaac Newton	isaac@example.com	encrypted_pass9	135 Redwood Ave	0
10	Jenna Rose	jenna@example.com	encrypted_pass10	245 Walnut St	0

b) Remove votes for a specific candidate

DELETE FROM Votes

WHERE candidate_id = 1;

The screenshot shows the phpMyAdmin interface for an 'online_voting_system' database. The 'Votes' table is selected. In the SQL tab, the following query is run:

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

The result message indicates "2 rows deleted. (Query took 0.0046 seconds.)". The browser taskbar at the bottom shows the URL as localhost/127.0.0.1/online_voting_system.

The screenshot shows the phpMyAdmin interface for an 'online_voting_system' database. The 'Votes' table is selected. In the SQL tab, the following query is run:

```
SELECT * FROM Votes WHERE candidate_id = 1;
```

The result message indicates "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)". The browser taskbar at the bottom shows the URL as localhost/127.0.0.1/online_voting_system.

2. UPDATE Operation:

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

SQL: UPDATE Voters

SET status = TRUE

WHERE voter_id = 1;

The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database. In the SQL tab, the following query was run:

```
UPDATE Voters SET status = TRUE WHERE voter_id = 1;
```

The results show 1 row affected in 0.0048 seconds. The voter record with voter_id 1 has been updated.

The screenshot shows the phpMyAdmin interface for the 'online_voting_system' database, specifically viewing the 'Voters' table. The table structure includes columns: voter_id, full_name, email, password, address, and status. A single record is displayed:

voter_id	full_name	email	password	address	status
1	Alice Johnson	alice@example.com	encrypted_pass1	123 Elm Street	1

The status column for this voter is now set to 1, indicating they have voted.

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';
```

The screenshot shows the phpMyAdmin interface on a Windows desktop. The database selected is 'online_voting_system'. In the SQL tab, the following query is run:

```
UPDATE Elections SET start_date = '2024-01-05', end_date = '2024-01-10' WHERE election_id = 1 AND status = 'upcoming';
```

The result message indicates '1 row affected. (Query took 0.0024 seconds.)'. The status bar at the bottom right shows 'Activate Windows'.

The screenshot shows the phpMyAdmin interface on a Windows desktop. The database selected is 'online_voting_system'. In the Structure tab, the 'Elections' table is selected. A query is run:

```
SELECT * FROM Elections WHERE election_id = 1;
```

The results show one row:

election_id	election_name	start_date	end_date	status
1	Presidential Election 2024	2024-01-05	2024-01-10	upcoming

The status bar at the bottom right shows 'Activate Windows'.

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;

The screenshot shows the phpMyAdmin interface with the database 'online_voting_system' selected. In the center, the SQL query results are displayed in a green box: "1 row affected. (Query took 0.0032 seconds.)" Below it, the executed SQL command is shown: `UPDATE Election_Results SET votes_received = votes_received + 1 WHERE candidate_id = 1 AND election_id = 1;`. The left sidebar lists various databases and tables.

The screenshot shows the phpMyAdmin interface with the 'Election_Results' table selected under the 'online_voting_system' database. The table structure is visible with columns: result_id, election_id, position_id, candidate_id, and votes_received. A single row is selected and displayed: result_id=1, election_id=1, position_id=1, candidate_id=1, and votes_received=3. The bottom section shows the 'Query results operations' toolbar with options like Print, Copy to clipboard, Export, Display chart, and Create view.

3. ALTER Operation:

a) Add a new column to store voter phone numbers

SQL: ALTER TABLE Voters

ADD phone_number VARCHAR(15);

The screenshot shows two instances of the phpMyAdmin interface. In the top instance, a query is being run:

```
ALTER TABLE Voters ADD phone_number VARCHAR(15);
```

The message indicates that MySQL returned an empty result set (i.e., zero rows). In the bottom instance, the DESCRIBE command is run on the 'Voters' table:

```
DESCRIBE Voters;
```

The resulting table structure is as follows:

Field	Type	Null	Key	Default	Extra
voter_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(255)	NO		NULL	
email	varchar(255)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	
address	varchar(255)	YES		NULL	
status	tinyint(1)	YES		0	
phone_number	varchar(15)	YES		NULL	

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

SQL: ALTER TABLE Voters

MODIFY email VARCHAR(320);

The screenshot shows two instances of the phpMyAdmin interface. In the top instance, a query is being run:

```
ALTER TABLE Voters MODIFY email VARCHAR(320);
```

The message in the results area indicates that MySQL returned an empty result set (0 rows). In the bottom instance, the results of the query are displayed. The message says "Your SQL query has been executed successfully." Below this, the structure of the 'Voters' table is shown in a grid:

Field	Type	Null	Key	Default	Extra
voter_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(255)	NO		NULL	
email	varchar(320)	YES	UNI	NULL	
password	varchar(255)	NO		NULL	
address	varchar(255)	YES		NULL	
status	tinyint(1)	YES		0	
phone_number	varchar(15)	YES		NULL	

c) Rename Voters table to Registered_Voters using Alter

SQL: ALTER TABLE Voters

RENAME TO Registered_Voters;

The screenshot shows the phpMyAdmin interface for a MySQL database named 'online_voting_system'. The 'Structure' tab is selected for the 'Votes' table. In the SQL query box, the command `ALTER TABLE Voters RENAME TO Registered_Voters;` is entered. A green message bar at the top indicates that MySQL returned an empty result set. Below the message, the query is shown again with syntax highlighting.

The screenshot shows the phpMyAdmin interface for the same database and table. The 'Structure' tab is selected. The message bar now says 'Your SQL query has been executed successfully.' Below the message, the command `DESCRIBE Registered_Voters;` is shown. The table structure is displayed in a grid:

Field	Type	Null	Key	Default	Extra
voter_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(255)	NO		NULL	
email	varchar(320)	YES	UNI	NULL	
password	varchar(255)	NO		NULL	
address	varchar(255)	YES		NULL	
status	tinyint(1)	YES		0	
phone_number	varchar(15)	YES		NULL	

At the bottom of the interface, there are buttons for 'Print', 'Copy to clipboard', and 'Create view'.

4. Drop a Column:

a) Remove the party column from the Candidates table

SQL: ALTER TABLE Candidates

DROP COLUMN party;

The screenshot shows the phpMyAdmin interface. In the left sidebar, the database structure is visible, including the 'online_voting_system' schema which contains tables like 'candidates', 'elections', and 'votes'. The main area shows the SQL query: `ALTER TABLE Candidates DROP COLUMN party;`. A green message bar at the top indicates that MySQL returned an empty result set (0 rows). Below the query, there are edit and create PHP code options.

The screenshot shows the phpMyAdmin interface after the query has been executed. The main area displays the results of the `DESCRIBE Candidates;` query. The table structure is shown with the following columns:

Field	Type	Null	Key	Default	Extra
candidate_id	int(11)	NO	PRI	NULL	auto_increment
full_name	varchar(255)	NO		NULL	
position_id	int(11)	NO	MUL	NULL	
election_id	int(11)	NO	MUL	NULL	

Below the table, there are options for printing, copying to clipboard, or creating a view. A message bar at the top indicates that the SQL query has been executed successfully.

5. Basic SELECT Operations:

a) Retrieve all voters' information

SQL: SELECT * FROM Voters;

The screenshot shows the phpMyAdmin interface for the 'Votes' table in the 'online_voting_system' database. The table has columns: voter_id, full_name, email, password, address, status, and phone_number. The data shows 10 rows of voter information.

voter_id	full_name	email	password	address	status	phone_number
1	Alice Johnson	alice@example.com	encrypted_pass1	123 Elm Street	1	NULL
2	Bob Smith	bob@example.com	encrypted_pass2	456 Oak Avenue	0	NULL
3	Charlie Brown	charlie@example.com	encrypted_pass3	789 Pine Road	0	NULL
4	Diana Prince	diana@example.com	encrypted_pass4	321 Maple Lane	0	NULL
5	Ethan Hunt	ethan@example.com	encrypted_pass5	654 Cedar Court	0	NULL
6	Fiona Carter	fiona@example.com	encrypted_pass6	987 Birch Way	0	NULL
7	George Miller	george@example.com	encrypted_pass7	432 Spruce Blvd	0	NULL
8	Hannah Green	hannah@example.com	encrypted_pass8	876 Aspen Dr	0	NULL
9	Isaac Newton	isaac@example.com	encrypted_pass9	135 Redwood Ave	0	NULL
10	Jenna Rose	jenna@example.com	encrypted_pass10	246 Walnut St	0	NULL

b) Display candidates participating in a specific election

SQL: SELECT * FROM Candidates

WHERE election_id = 1;

The screenshot shows the phpMyAdmin interface for the 'Candidates' table in the 'online_voting_system' database, filtered by election_id = 1. The table has columns: candidate_id, full_name, position_id, and election_id. The data shows 3 rows of candidate information.

candidate_id	full_name	position_id	election_id
1	John Doe	1	1
2	Jane Smith	1	1
3	Robert Green	2	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';

The screenshot shows the phpMyAdmin interface on a Windows desktop. The browser tab is 'localhost / 127.0.0.1 / online_vc'. The phpMyAdmin sidebar lists databases like 'atfal_hossain_raju', 'information_schema', 'labfinal', 'library', 'mysql', 'online_voting_system' (selected), 'performance_schema', 'phpmyadmin', 'student_management_system', and 'test'. The main area shows the 'Voters' table with one row: Alice Johnson with email 'alice@example.com' and status '1'. The SQL query entered is: 'SELECT full_name, email, status FROM Voters WHERE status = TRUE AND email LIKE '%@example.com';'. The results show 1 row.

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');

The screenshot shows the phpMyAdmin interface for the 'Voters' table in the 'online_voting_system' database. A search query was run:

```
SELECT full_name, email, status FROM Voters WHERE status = TRUE AND (email LIKE '%@example.com' OR email LIKE '%@gmail.com');
```

The results show one row:

full_name	email	status
Alice Johnson	alice@example.com	1

Below the table, there are 'Query results operations' buttons: Print, Copy to clipboard, Export, Display chart, and Create view.

8. SELECT with Aggregate Functions:

a) Count the total number of voters

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

The screenshot shows the phpMyAdmin interface after executing the COUNT(*) query. A message indicates that the current selection does not contain a unique column, so grid edit, checkbox, Edit, Copy and Delete features are not available.

Your SQL query has been executed successfully.

```
SELECT COUNT(*) AS total_voters FROM Voters;
```

The result shows a single row with the value 10:

total_voters
10

Below the table, there are 'Query results operations' buttons: Print, Copy to clipboard, Export, Display chart, and Create view.

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;`

The screenshot shows the phpMyAdmin interface on a Windows desktop. The left sidebar lists databases and tables. The main area shows the results of a query:

```
Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)  
SELECT candidate_id, MAX(votes_received) AS highest_votes FROM Election_Results WHERE election_id = 1;
```

The results table has columns `candidate_id` and `highest_votes`, showing one row with `candidate_id` 1 and `highest_votes` 3.

At the bottom, there are various operations like Print, Copy to clipboard, Export, Display chart, and Create view.

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;`

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.) [votes_received: 3... - 0...]

```
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;
```

candidate_id	full_name	position_id	votes_received
1	John Doe	1	3
9	Henry Gold	8	1
8	Sophia Blue	7	1
7	Daniel Brown	6	1
6	Emily Gray	5	1
5	Michael White	4	1
4	Lucy Black	3	1
3	Robert Green	2	1
2	Jane Smith	1	1
10	Victoria Pink	9	0

b) Fetch voters sorted by their names alphabetically

SQL: SELECT *

FROM Voters

ORDER BY full_name ASC;

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.) [full_name: ALICE JOHNSON... - JENNA ROSE...]

```
SELECT * FROM Voters ORDER BY full_name ASC;
```

voter_id	full_name	email	password	address	status	phone_number
1	Alice Johnson	alice@example.com	encrypted_pass1	123 Elm Street	1	NULL
2	Bob Smith	bob@example.com	encrypted_pass2	456 Oak Avenue	0	NULL
3	Charlie Brown	charlie@example.com	encrypted_pass3	789 Pine Road	0	NULL
4	Diana Prince	diana@example.com	encrypted_pass4	321 Maple Lane	0	NULL
5	Ethan Hunt	ethan@example.com	encrypted_pass5	654 Cedar Court	0	NULL
6	Fiona Carter	fiona@example.com	encrypted_pass6	987 Birch Way	0	NULL
7	George Miller	george@example.com	encrypted_pass7	432 Spruce Blvd	0	NULL
8	Hannah Green	hannah@example.com	encrypted_pass8	876 Aspen Dr	0	NULL
9	Isaac Newton	isaac@example.com	encrypted_pass9	135 Redwood Ave	0	NULL
10	Jenna Rose	jenna@example.com	encrypted_pass10	246 Walnut St	0	NULL

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;
```

The screenshot shows the phpMyAdmin interface. The left sidebar lists databases and tables. The main area displays a query result for the 'Elections' table. The query is:`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;`

The result table shows three rows of data:

candidate_id	full_name	position_id	votes_received
1	John Doe	1	3
9	Henry Gold	8	1
8	Sophia Blue	7	1

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;
```

localhost / 127.0.0.1 / online_voting_system

localhost/phpmyadmin/index.php?route=/table/sql&db=online_voting_system&table=Election_Results

phpMyAdmin

Server: 127.0.0.1 > Database: online_voting_system > Table: Election_Results

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0005 seconds.)

SELECT MIN(votes_received) AS minimum_votes, MAX(votes_received) AS maximum_votes FROM Election_Results;

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table

Extra options

minimum_votes maximum_votes

0 3

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

24°C Haze

Search

ENG INTL 5:32 PM 12/7/2024

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;

localhost / 127.0.0.1 / online_voting_system

localhost/phpmyadmin/index.php?route=/table/sql&db=online_voting_system&table=Election_Results

phpMyAdmin

Server: 127.0.0.1 > Database: online_voting_system > Table: Election_Results

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0006 seconds.)

SELECT AVG(votes_received) AS average_votes FROM Election_Results WHERE election_id = 1;

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table

Extra options

average_votes

2.0000

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

24°C Haze

Search

ENG INTL 5:36 PM 12/7/2024

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;

The screenshot shows the phpMyAdmin interface for a MySQL database named 'online_voting_system'. The 'Candidates' table is selected. A query has been run:

```
SELECT election_id, COUNT(candidate_id) AS candidate_count FROM Candidates GROUP BY election_id;
```

The results are displayed in a table:

election_id	candidate_count
1	3
2	1
3	1
4	1
5	1
6	1
7	1
9	1

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;
```

The screenshot shows the phpMyAdmin interface on a Windows desktop. The browser address bar indicates the connection is to localhost. The phpMyAdmin sidebar lists databases like 'atfal_hossain_raju', 'information_schema', 'labinfall', 'library', 'mysql', and 'online_voting_system'. The 'online_voting_system' database is selected, and the 'Election_Results' table is chosen. The main area displays the results of the executed SQL query:

```
Showing rows 0 - 0 (1 total, Query took 0.0008 seconds.)  
SELECT candidate_id, SUM(votes_received) AS total_votes FROM Election_Results GROUP BY candidate_id HAVING SUM(votes_received) > 2;  
Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
```

The results table shows one row with candidate_id 1 and total_votes 3.

candidate_id	total_votes
1	3

Below the table are options for 'Query results operations' including Print, Copy to clipboard, Export, Display chart, and Create view. The taskbar at the bottom shows various application icons and the system clock.

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;

The screenshot shows the phpMyAdmin interface on a Windows desktop. The browser address bar indicates the connection is to localhost. The phpMyAdmin sidebar lists databases like afzal_hossain_raju, information_schema, tabfinal, library, mysql, online_voting_system, performance_schema, phpmyadmin, student_management_system, and test. The main area shows the 'Election_Results' table from the 'online_voting_system' database. A query has been run in the SQL tab:

```
SELECT election_id, AVG(votes_received) AS average_votes FROM Election_Results GROUP BY election_id HAVING AVG(votes_received) > 1;
```

The results show one row: election_id 1 with average_votes 2.0000. The desktop taskbar at the bottom shows various application icons and the system clock.

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;
```

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 / Database: online_voting_system / Table: Election_Results
- Query:** 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;
- Results:** 8 rows displayed.
- Table Headers:** voter_id, voter_name, email, candidate_id, candidate_name, position_id
- Table Data:**

voter_id	voter_name	email	candidate_id	candidate_name	position_id
2	Bob Smith	bob@example.com	2	Jane Smith	1
4	Diana Prince	diana@example.com	3	Robert Green	2
5	Ethan Hunt	ethan@example.com	4	Lucy Black	3
6	Fiona Carter	fiona@example.com	5	Michael White	4
7	George Miller	george@example.com	6	Emily Gray	5
8	Hannah Green	hannah@example.com	7	Daniel Brown	6
9	Isaac Newton	isaac@example.com	8	Sophia Blue	7
10	Jenna Rose	jenna@example.com	9	Henry Gold	8

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;

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** online_voting_system
- Table:** Election_Results
- Query:**

```
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;
```
- Result:** 12 rows displayed, showing elections from January 2024 to October 2024, each with up to 10 candidates.

election_id	election_name	start_date	end_date	candidate_id	candidate_name
1	Presidential Election 2024	2024-01-05	2024-01-10	1	John Doe
1	Presidential Election 2024	2024-01-05	2024-01-10	2	Jane Smith
1	Presidential Election 2024	2024-01-05	2024-01-10	3	Robert Green
2	City Mayor Election 2024	2024-02-01	2024-02-05	4	Lucy Black
3	Senate Election 2024	2024-03-01	2024-03-03	5	Michael White
4	School Board Election 2024	2024-04-10	2024-04-12	6	Emily Gray
5	State Governor Election 2024	2024-05-15	2024-05-20	7	Daniel Brown
6	Referendum 2024	2024-06-01	2024-06-01	8	Sophia Blue
7	Union Leader Election 2024	2024-07-10	2024-07-10	9	Henry Gold
8	Party Leadership Election	2024-08-05	2024-08-06	NULL	NULL
9	Municipal Council Election 2024	2024-09-10	2024-09-15	10	Victoria Pink
10	Youth Council Election 2024	2024-10-01	2024-10-02	NULL	NULL

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;

The screenshot shows the phpMyAdmin interface for a database named 'online_voting_system'. The 'Elections' table is selected. A query has been run:

```
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;
```

The results show 10 rows of data:

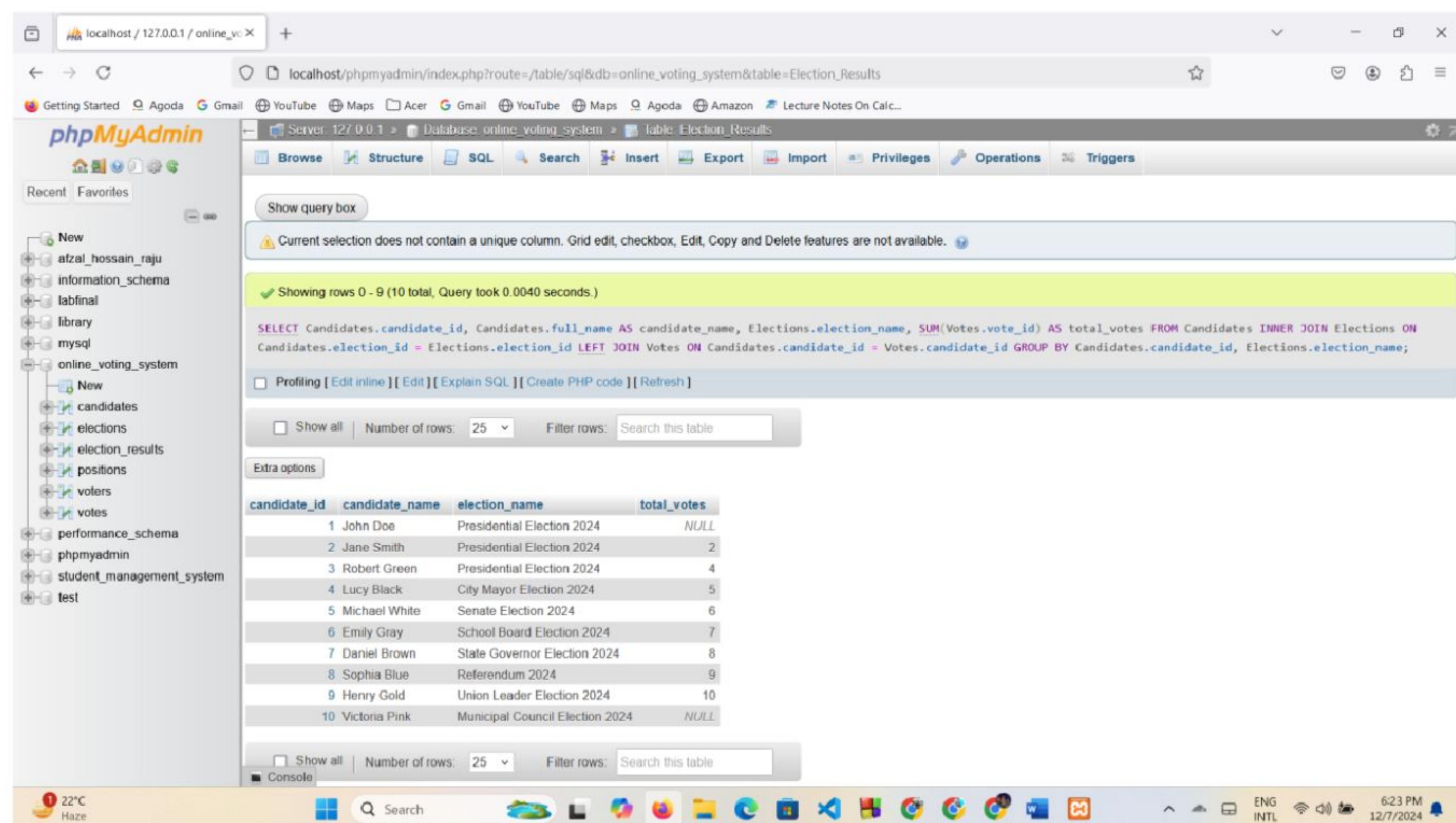
candidate_id	candidate_name	election_id	election_name	start_date	end_date
1	John Doe	1	Presidential Election 2024	2024-01-05	2024-01-10
2	Jane Smith	1	Presidential Election 2024	2024-01-05	2024-01-10
3	Robert Green	1	Presidential Election 2024	2024-01-05	2024-01-10
4	Lucy Black	2	City Mayor Election 2024	2024-02-01	2024-02-05
5	Michael White	3	Senate Election 2024	2024-03-01	2024-03-03
6	Emily Gray	4	School Board Election 2024	2024-04-10	2024-04-12
7	Daniel Brown	5	State Governor Election 2024	2024-05-15	2024-05-20
8	Sophia Blue	6	Referendum 2024	2024-06-01	2024-06-01
9	Henry Gold	7	Union Leader Election 2024	2024-07-10	2024-07-10
10	Victoria Pink	9	Municipal Council Election 2024	2024-09-10	2024-09-15

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;
```



candidate_id	candidate_name	election_name	total_votes
1	John Doe	Presidential Election 2024	NULL
2	Jane Smith	Presidential Election 2024	2
3	Robert Green	Presidential Election 2024	4
4	Lucy Black	City Mayor Election 2024	5
5	Michael White	Senate Election 2024	6
6	Emily Gray	School Board Election 2024	7
7	Daniel Brown	State Governor Election 2024	8
8	Sophia Blue	Referendum 2024	9
9	Henry Gold	Union Leader Election 2024	10
10	Victoria Pink	Municipal Council Election 2024	NULL

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';

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible, including the 'online_voting_system' database which contains the 'candidates', 'elections', 'election_results', 'positions', 'voters', and 'votes' tables. The main area displays the results of a SQL query:

```
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';
```

The results table shows the following data:

voter_name	candidate_name	vote_time
Diana Prince	Robert Green	2024-02-01 09:00:00

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

);

The screenshot shows the phpMyAdmin interface on a Windows desktop. The browser address bar shows 'localhost/phpmyadmin/index.php?route=/table/sql&db=online_voting_system&table=candidates'. The phpMyAdmin sidebar lists databases like 'atzz_hossain_raju', 'information_schema', 'labfinal', 'library', 'mysql', and 'online_voting_system'. Under 'online_voting_system', it shows tables: 'candidates' (selected), 'elections', 'election_results', 'positions', 'voters', and 'votes'. The main query results window displays the following SQL query and its execution message:

```
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.candidate_id 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 );
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0053 seconds.)

The status bar at the bottom shows 'Activate Windows Go to Settings to activate Windows.' and the system tray indicates '17°C Haze'.

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');
```

The screenshot shows the phpMyAdmin interface with two windows open. The left sidebar lists databases and tables. The top window is titled 'View: raju1031' and shows the results of a SELECT query. The bottom window is titled 'Table: candidates' and shows the SQL code for creating the view.

Top Window (View: raju1031):

```
SELECT * FROM raju1031;
```

voter_name	candidate_name	vote_time
Diana Prince	Robert Green	2024-02-01 09:00:00

Bottom Window (Table: candidates):

```
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');
```