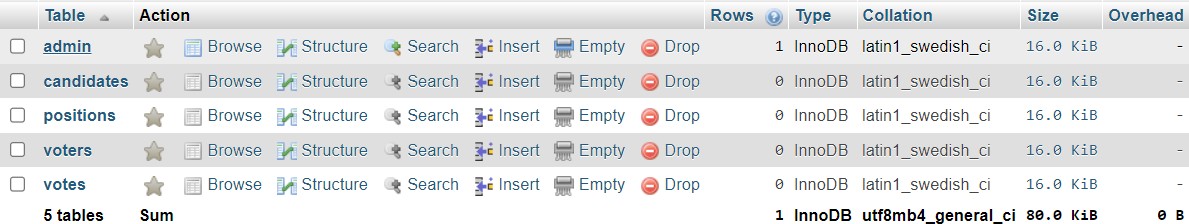
# Database

DBMS ASSIGNMENT

# Name – Vasu Dobariya

**PROJECT NAME- VOTING MANAGEMENT SYSTEM**

>VOTING Database consists of seven tables-



# Explaination

1. admin:

This table stores information about administrators who have access to the system. Columns:

id: Unique identiﬁer for each administrator.

username: The username used for login purposes.

password: The password for the administrator account (ideally hashed for security).

ﬁrstname: First name of the administrator. lastname: Last name of the administrator.

photo: Path or URL to the administrator's photo.

created\_on: The date when the administrator account was created.

1. candidates:

This table holds data about candidates participating in an election or voting process. Columns:

id: Unique identiﬁer for each candidate.

position\_id: The identiﬁer of the position the candidate is running for (foreign key referencing positions.id).

ﬁrstname: First name of the candidate. lastname: Last name of the candidate.

photo: Path or URL to the candidate's photo.

platform: Text ﬁeld describing the candidate's platform or agenda.

* No\_of\_votes: How many votes are received each candidate.

1. positions:

This table contains information about the positions available in an election or voting process.

Columns:

id: Unique identiﬁer for each position.

description: Description of the position.

max\_vote: Maximum number of votes allowed for this position.

priority: Priority or order of importance of the position in the election.

1. voters:

This table stores data about the voters who are eligible to cast votes. Columns:

id: Unique identiﬁer for each voter.

voters\_id: Identiﬁcation number or code for the voter.

password: Password for the voter account (ideally hashed for security). ﬁrstname: First name of the voter.

lastname: Last name of the voter.

photo: Path or URL to the voter's photo.

1. votes:

This table records the votes cast by voters for candidates in speciﬁc positions. Columns:

id: Unique identiﬁer for each vote.

voters\_id: The identiﬁer of the voter who cast the vote (foreign key referencing voters.id).

candidate\_id: The identiﬁer of the candidate voted for (foreign key referencing candidates.id).

position\_id: The identiﬁer of the position for which the vote was cast (foreign key

referencing positions.id).

# RELATIONSHIPS

candidates and positions have a one-to-many relationship. positions and votes have a one-to-many relationship. voters and votes have a one-to-many relationship.

candidates and votes have a one-to-many relationship.

1. Admin - Voters Relationship:

Foreign Key: No direct relationship evident.

1. Admin - Candidates Relationship:

Foreign Key: No direct relationship evident.

1. Admin - Positions Relationship:

Foreign Key: No direct relationship evident.

1. Admin - Votes Relationship:

Foreign Key: No direct relationship evident.

1. Candidates - Positions Relationship:

Foreign Key: position\_id in the candidates table references id in the positions table. Relationship: One-to-Many (One position can have many candidates, but one candidate belongs to only one position).

1. Positions - Votes Relationship:

Foreign Key: position\_id in the votes table references id in the positions table. Relationship: One-to-Many (One position can have many votes, but one vote is for only one position).

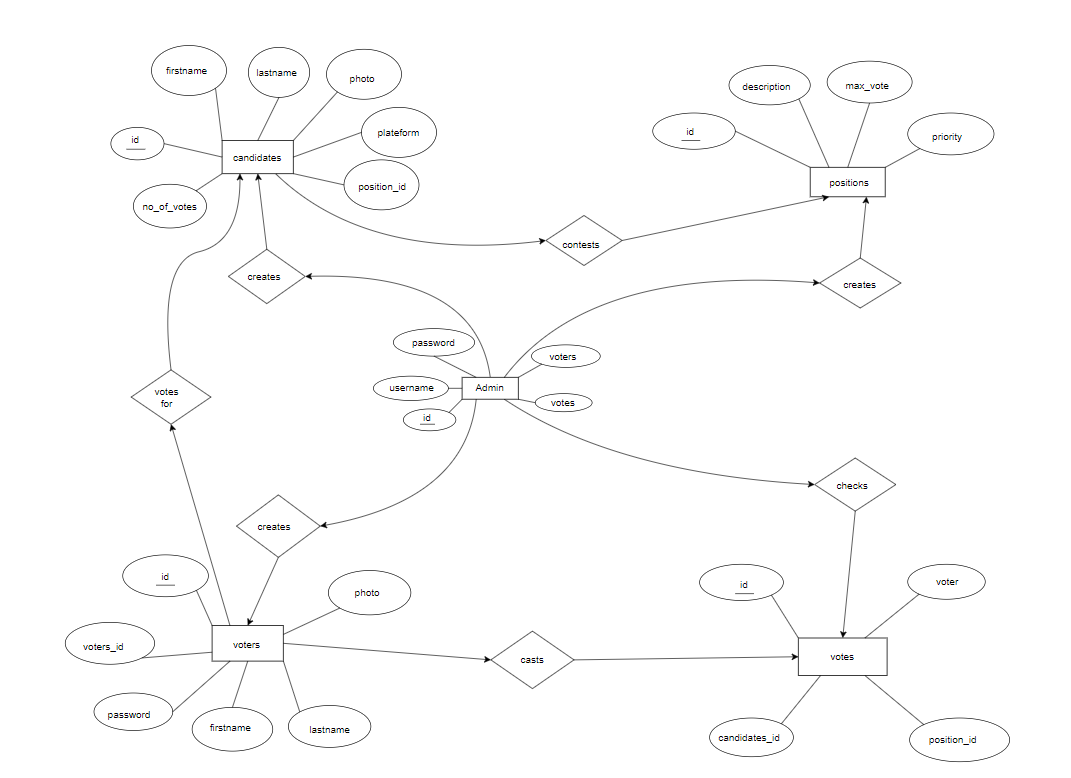
1. Voters - Votes Relationship:

Foreign Key: voters\_id in the votes table references id in the voters table. Relationship: One-to-Many (One voter can cast many votes, but one vote is cast by only one voter).

1. Candidates - Votes Relationship:

Foreign Key: candidate\_id in the votes table references id in the candidates table. Relationship: One-to-Many (One candidate can receive many votes, but one vote is for only one candidate).

# ER DIAGRAM



**NORMALIZATION AND FUNCTIONAL DEPENDENCIES**

Generally, the tables are present in 3rd Normal Form but Here the Normalized Tables are-

1. admin Table:

Primary Key: id

Functional Dependencies:

id -> username, password, ﬁrstname, lastname, photo, created\_on username -> id (assuming username is unique)

1. candidates Table:

Primary Key: id

Functional Dependencies:

id -> position\_id, ﬁrstname, lastname, photo, platform

1. positions Table:

Primary Key: id

Functional Dependencies:

id -> description, max\_vote, priority

1. voters Table:

Primary Key: id

Functional Dependencies:

id -> voters\_id, password, ﬁrstname, lastname, photo

1. votes Table:

Primary Key: id

Functional Dependencies:

id -> voters\_id, candidate\_id, position\_id

# Based on this analysis, we can see the following normalization:

First Normal Form (1NF):

All tables have atomic values in each cell.

Second Normal Form (2NF):

All tables have a primary key, and non-key attributes are fully functionally dependent on the primary key.

Third Normal Form (3NF):

There are no transitive dependencies in any table.

# These tables seem to be already normalized to at least 3NF.

# 

# Data Implementation:

# A) Schema

# Admin:

# 

# 

# Candidate:

# 

# 

# Position:

# 

# 

# Voters:

# 

# 

# Votes:

# 

# 

# Functionalities of Database:

# 

# Authenticate Admin login

# 

# 

# Add new Candidate

# 

# 

# Add Voter

# 

# 

# 

# Logout Admin

# 

# 

# Login Voter

# 

# 

# Show Candidate with their id, name and position

# 

# 

# Give your Vote

# 

# 

# Trigger for vote only once:

DELIMITER //

CREATE TRIGGER before\_insert\_vote

BEFORE INSERT ON votes

FOR EACH ROW

BEGIN

    DECLARE voter\_count INT;

    SELECT COUNT(\*) INTO voter\_count

    FROM votes

    WHERE voters\_id = NEW.voters\_id;

    IF voter\_count > 0 THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE\_TEXT = 'Voter has already voted for this position.';

    END IF;

END//

DELIMITER ;

# 

# 

# View overall voting statistics (e.g., total votes cast, voter turnout)

# 

# 

# View voting results for each position

# 

# 

# Display those candidate who have maximum votes

# 

# 