**Today’s Agenda**

1. **Understanding The Table Structure**
2. **Summary Of Relationships**
3. **Designing Entity Classes**
4. **Run The App**
5. **Verify The Tables In MySQL**
6. **Understanding The Table Structure**

There are **FOUR tables** used in **our project** and they are :  
 **voter**

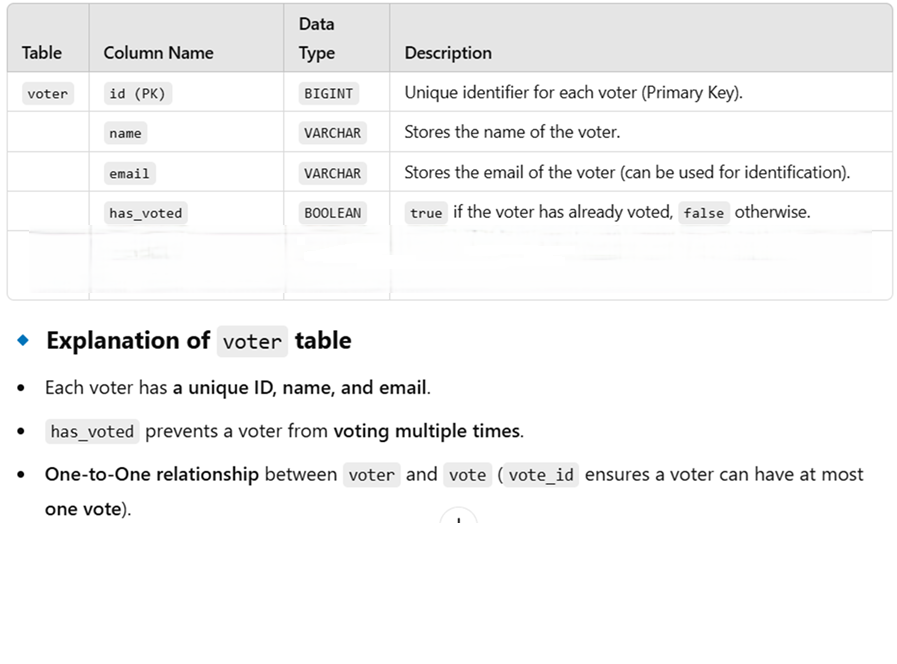
**candidate**

**vote**

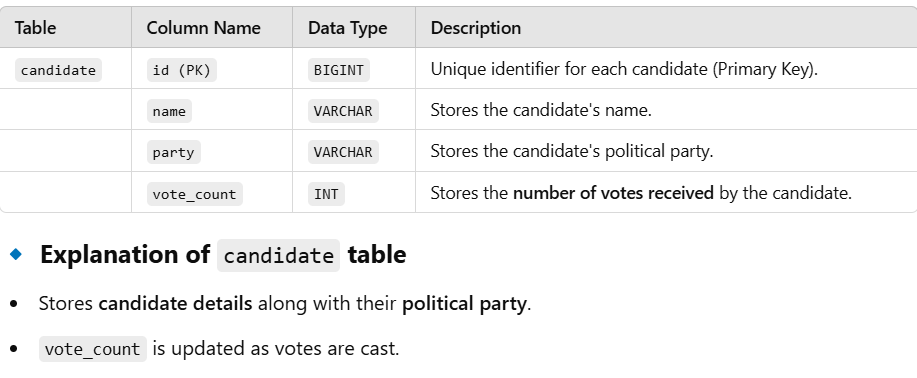
**election\_result**

**Following** is the **description** of **each table**:

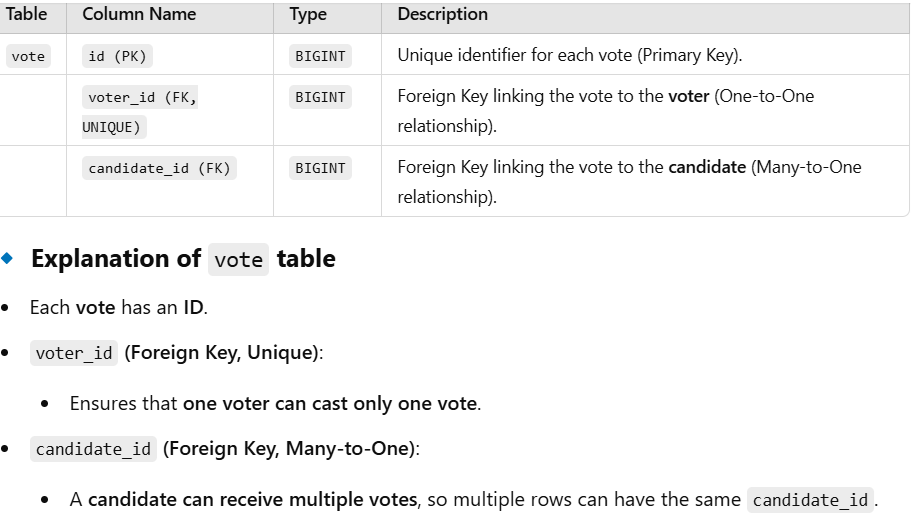
* + - 1. **The voter Table**



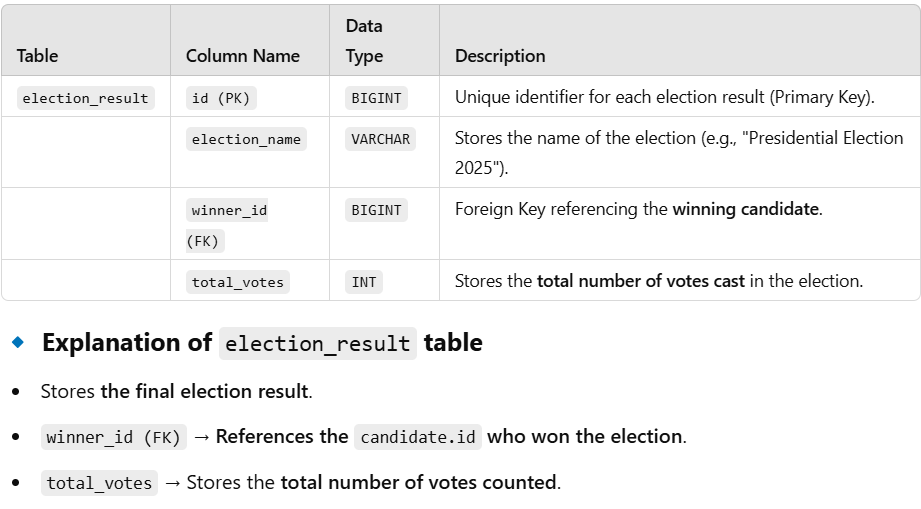
* + - 1. **The candidate Table**



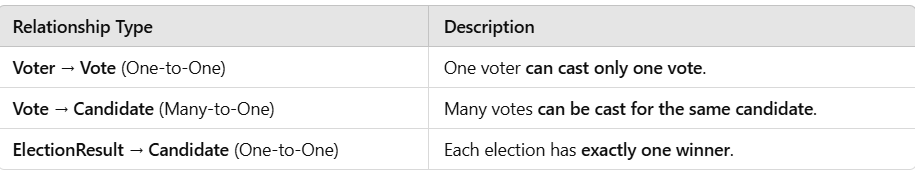
**3. The vote Table**



**4. The election\_result Table**

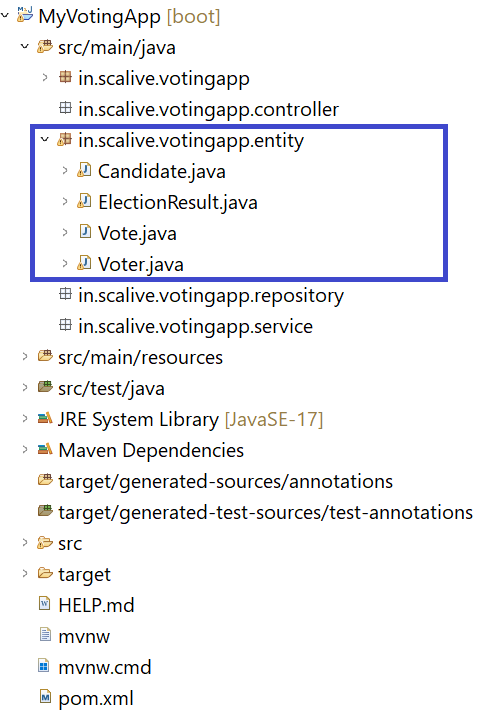


1. **Summary Of Relationships**

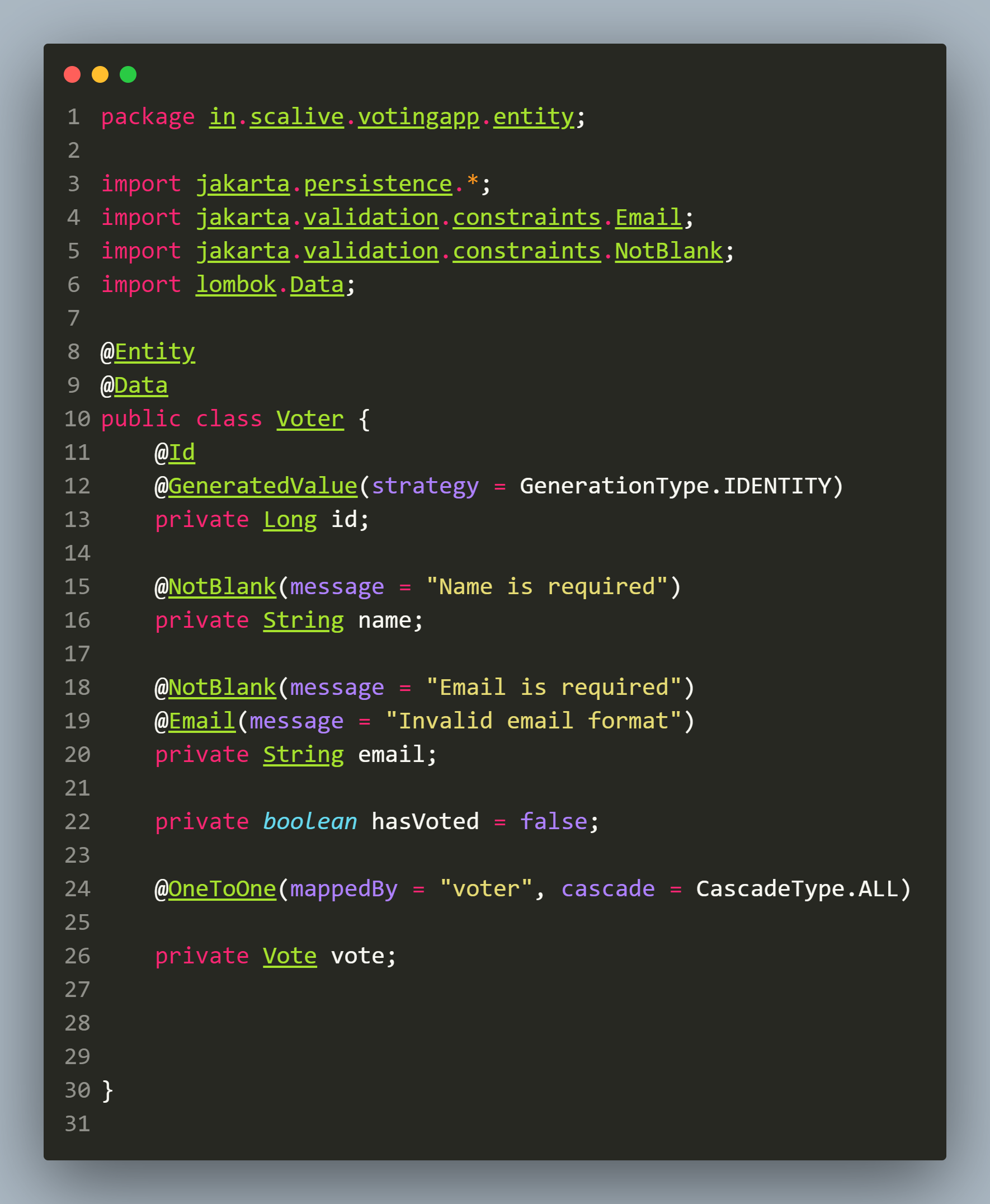


1. **Designing The Entity Classes**

Create **FOUR CLASSES** as **shown below** in the **entity** package :



**Code For Voter Class**

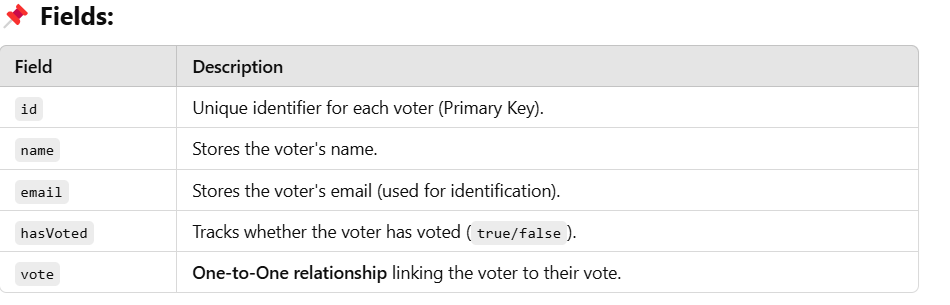


**📌 Explanation**

**Purpose**: **Represents** an **individual voter** who can **cast a vote** in the **election**.

📌 **Annotations Used:**

* **@Entity** → **Defines** this **class** as a **database table**.
* **@Id** → **Specifies** the **primary key**.
* **@GeneratedValue(strategy = GenerationType.IDENTITY)** → **Auto-generates** unique IDs for voters.
* **@OneToOne(mappedBy = "voter", cascade = CascadeType.ALL)** → **Defines** a **one-to-one relationship** with the **Vote** entity.
* **@Data** → **Lombok annotations** for **boilerplate reduction**.

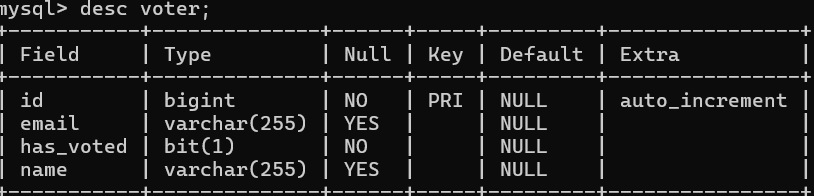


**🔹 Understanding the Relationships**

✅ **Why @OneToOne for Vote vote?**

* **Each voter can cast only one vote.**
* **A vote must belong to one voter.**

**Table Structure For VOTER Table Which Will Be Generated By MySQL**

****

**Code For Candidate Class**

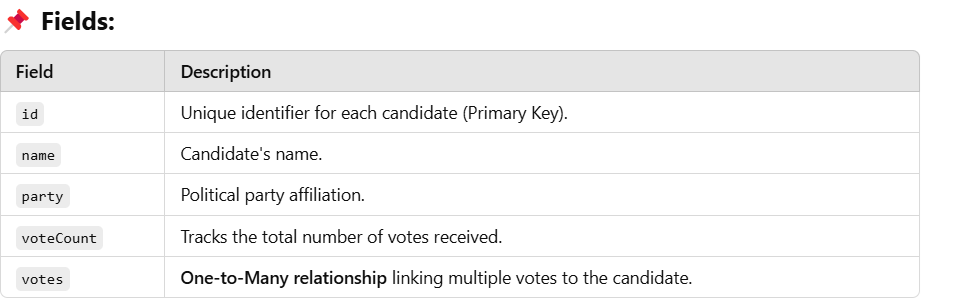


**📌 Explanation**

**Purpose:** **Represents** a **candidate** contesting in an **election** and tracks **votes received**.

**📌 Annotations Used:**

* **@Entity** → **Defines** this as a **database table**.
* **@Id** → **Specifies** the **primary key**.
* **@GeneratedValue(strategy = GenerationType.IDENTITY)** → **Auto-generates** unique IDs.
* **@OneToMany(mappedBy = "candidate", cascade = CascadeType.ALL)** → **Defines** a **one-to-many** relationship with the **Vote** entity.
* **@Data** → **Lombok annotations.**

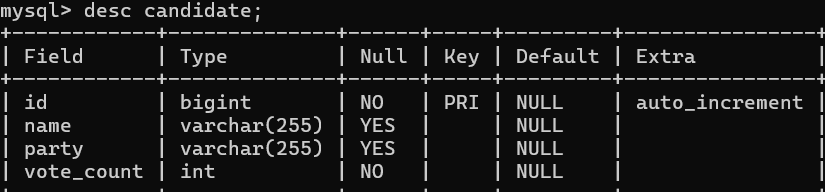


**🔹 Understanding the Relationships**

✅ **Why @OneToMany for List<Vote> votes?**

* A **candidate can receive multiple votes** from different voters.

**Table Structure For CANDIDATE Table Which Will Be Generated By MySQL**

****

**Code For Vote Class**

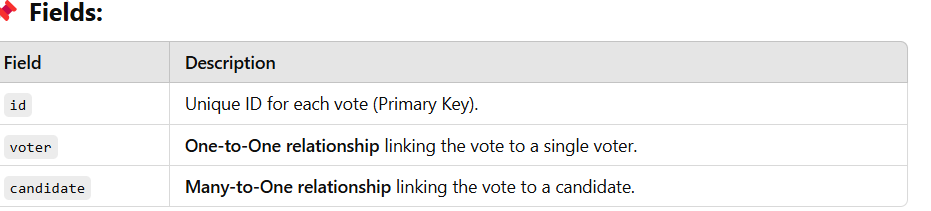


**📌 Explanation**

**Purpose:** **Represents** a **single vote cast** by a **voter** for a **candidate**.

**📌 Annotations Used:**

* **@Entity** → **Defines** this as a **database** table.
* **@Id** → **Primary key** for **Vote**.
* **@GeneratedValue(strategy = GenerationType.IDENTITY)** → **Auto-generates** unique IDs.
* **@OneToOne** → Links **one voter to one vote**.
* **@ManyToOne** → Links **multiple votes to one candidate**.
* **@Data** → **Lombok annotations.**



**🔹 Understanding the Relationships**

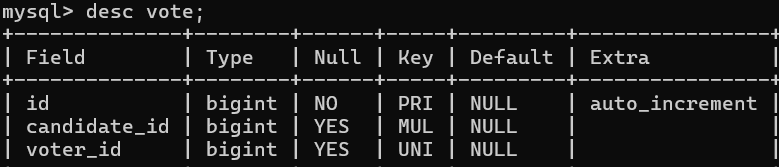
✅ **Why @OneToOne for Voter voter?**

* Each voter **can cast only one vote**.

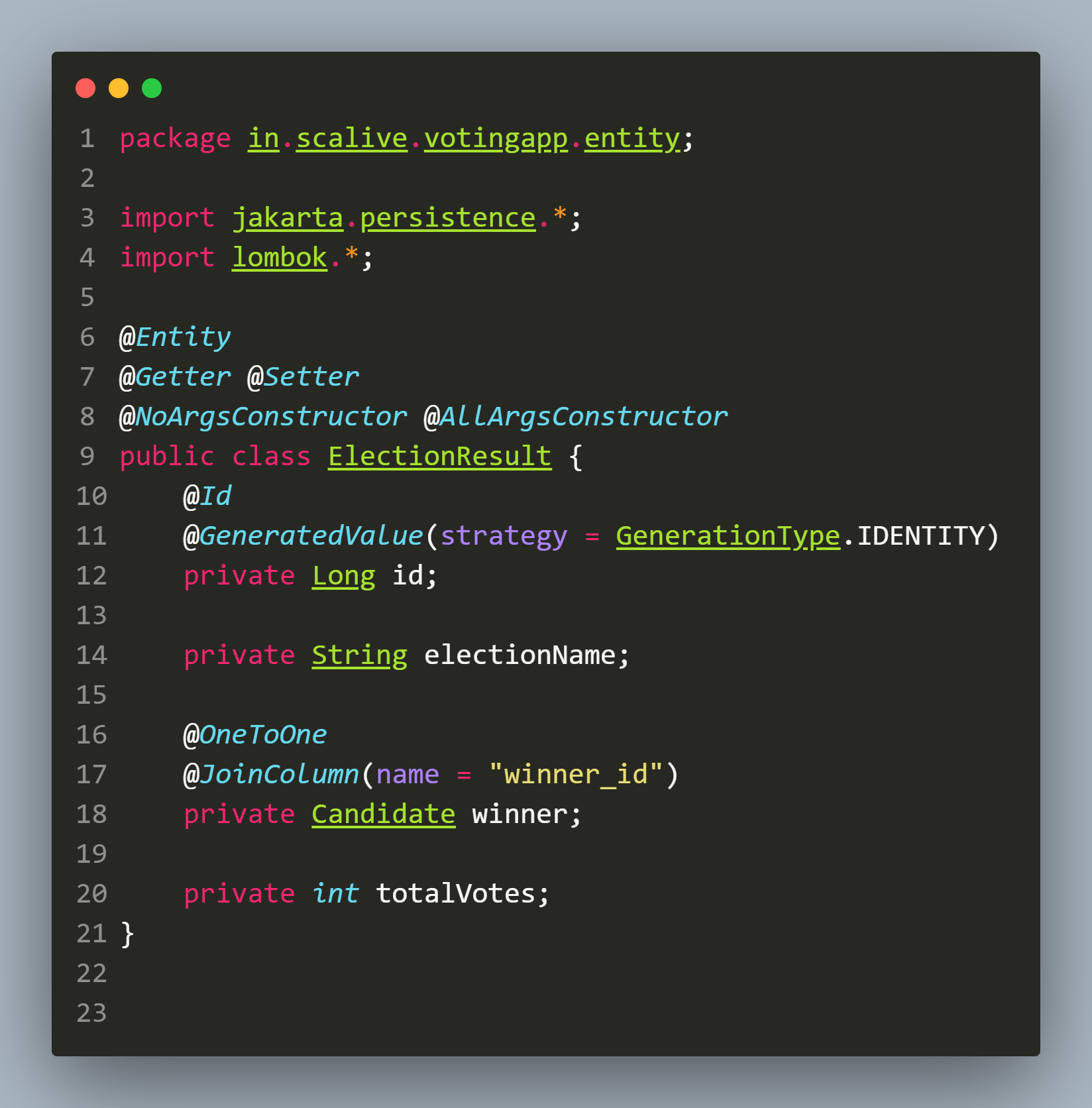
✅ **Why @ManyToOne for Candidate candidate?**

* A candidate **can receive multiple votes**, so multiple vote records can reference the same candidate.

**Table Structure For VOTE Table Which Will Be Generated By MySQL**

****

**Code For ElectionResult Class**

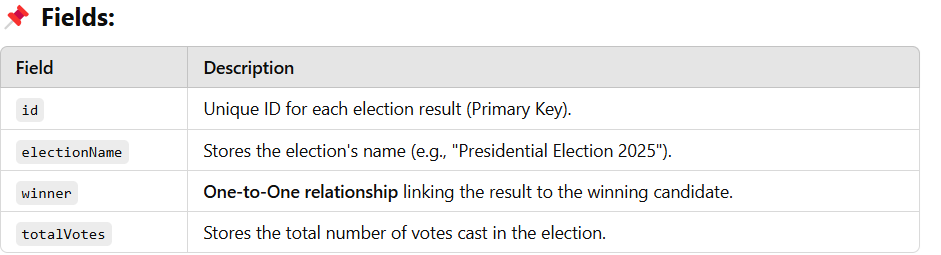


**📌 Explanation**

**Purpose**: **Stores** the **final election result**, including the **winner** and **total votes cast**.

📌 **Annotations Used:**

* **@Entity** → **Defines** this as a **database table.**
* **@Id** → **Primary key** for **ElectionResult.**
* **@GeneratedValue(strategy = GenerationType.IDENTITY)** → **Auto-generates** unique IDs.
* **@OneToOne** → **Links the election result to the winning candidate**.
* **@Getter, @Setter, @NoArgsConstructor, @AllArgsConstructor** → **Lombok annotations.**

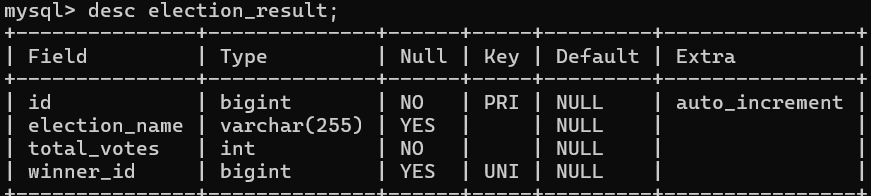


**🔹 Understanding the Relationships**

✅ **Why @OneToOne for Candidate winner?**

* **Each election has exactly one winner.**

**Table Structure For ELECTION\_RESULT Table Which Will Be Generated By MySQL**

****

1. **Run The App**

Now **run the app** so that the **tables** in **MySQL** are **autogenerated** by our code **as shown below**:

**Hibernate**: **create table candidate (id bigint not null auto\_increment, name varchar(255), party varchar(255), vote\_count integer not null, primary key (id)) engine=InnoDB**

**Hibernate**: **create table election\_result (id bigint not null auto\_increment, election\_name varchar(255), total\_votes integer not null, winner\_id bigint, primary key (id)) engine=InnoDB**

**Hibernate**: **create table vote (id bigint not null auto\_increment, candidate\_id bigint, voter\_id bigint, primary key (id)) engine=InnoDB**

**Hibernate**: **create table voter (id bigint not null auto\_increment, email varchar(255), has\_voted bit not null, name varchar(255), primary key (id)) engine=InnoDB**

**Hibernate:** **alter table election\_result drop index UK2o1khq97yq3m6476y7y9cflxs**

**Hibernate**: **alter table election\_result add constraint UK2o1khq97yq3m6476y7y9cflxs unique (winner\_id)**

**Hibernate**: **alter table vote drop index UKh4fikur6p8e2j06er82xii08b**

**Hibernate**: **alter table vote add constraint UKh4fikur6p8e2j06er82xii08b unique (voter\_id)**

**Hibernate**: **alter table election\_result add constraint FK96ikhe29cqu14qpvjtcahs1et foreign key (winner\_id) references candidate (id)**

**Hibernate**: **alter table vote add constraint FKsntn7i4l97xxwx420gb6ldpvn foreign key (candidate\_id) references candidate (id)**

**Hibernate**: **alter table vote add constraint FKbgqu449eayg16pkhliv4o0ejh foreign key (voter\_id) references voter (id)**

1. **Verify The Tables In MySQL**

