

**SCHOOL OF BUSINESS AND ECONOMICS**

**DEPARTMENT OF BUSINESS TECHNOLOGY**

**DATABASE MANAGEMENT SYSTEM**

**Project Name:**

**TALENT MANAGEMENT SYSTEM**

**Student name: NYANGOMA ANNET MUKABARISA**

**Reg No: 222001557**

**Section I**

**1. Describe all the entities and their corresponding attributes that are in your database.**

**Applicant: This entity represents an individual who has applied for a job opening. The attributes of this entity may include applicant ID, name, contact information, resume, cover letter, and application status.**

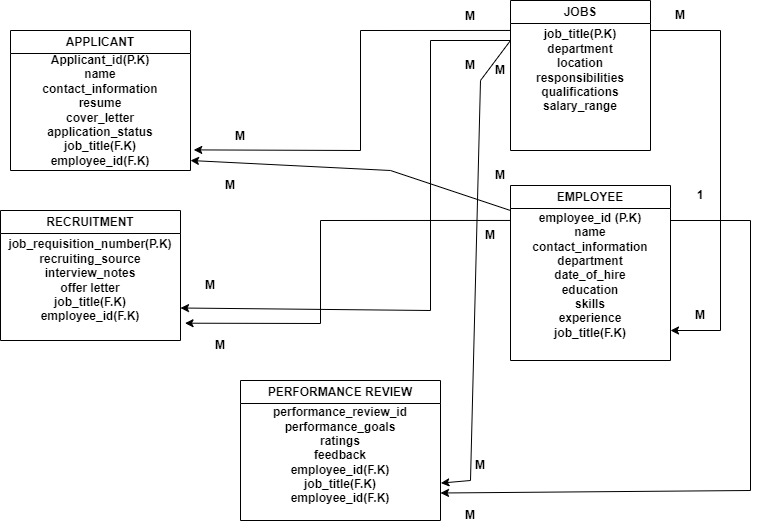
**Recruitment: This entity represents the process of hiring new employees. The attributes of this entity may include job requisition number, recruiting source, interview notes, and offer letter.**

**Performance review: This entity represents the process of evaluating employee performance. The attributes of this entity may include performance goals, ratings, and feedback.**

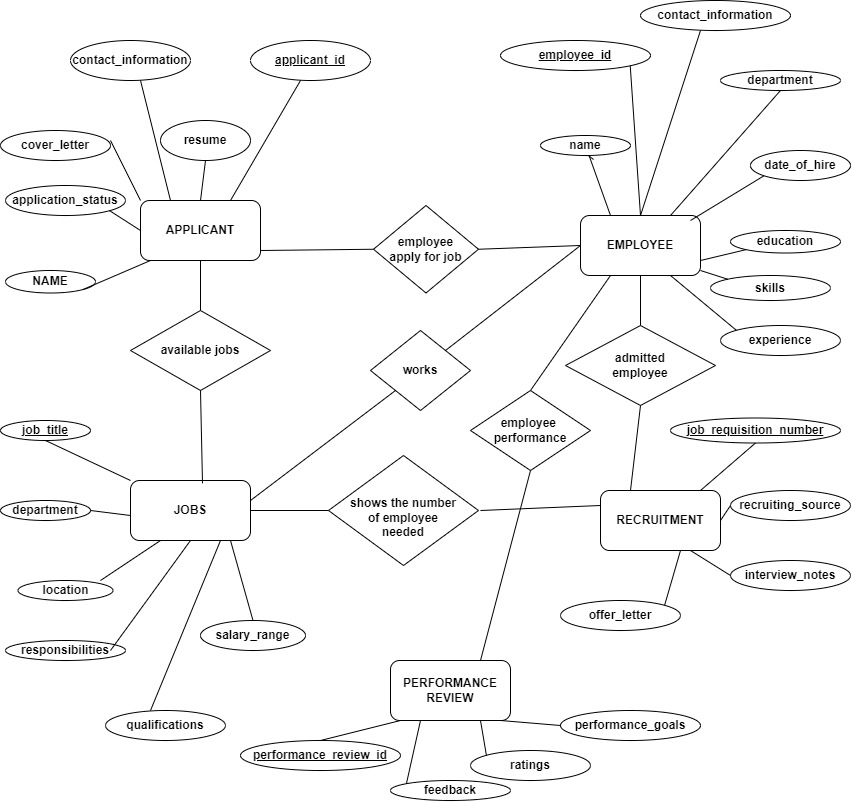
**Employee: This entity represents an individual employee. The attributes of this entity may include employee ID, name, contact information, job title, department, date of hire, education, skills, and experience.**

**Job: This entity represents a job opening in the organization. The attributes of this job may include job title, department, location, responsibilities, qualifications, and salary range.**

**2. Create an LDM of your entities**

****

**3. Create an ERD**

****

**Section II SQL**

**1. Create the database of your system**

**CREATE DATABASE TALENT\_MANAGEMENT\_SYSTEM;**

**2. Write queries to create all the tables and relationships of your system**

**CREATE TABLE employees (**

**employee\_id INT NOT NULL AUTO\_INCREMENT,**

**name VARCHAR(255) NOT NULL,**

**contact\_information VARCHAR(255) NOT NULL,**

**job\_title VARCHAR(255) NOT NULL,**

**department VARCHAR(255) NOT NULL,**

**date\_of\_hire DATE NOT NULL,**

**education VARCHAR(255) NOT NULL,**

**skills VARCHAR(255) NOT NULL,**

**experience VARCHAR(255) NOT NULL,**

**PRIMARY KEY (employee\_id)**

**);**

**CREATE TABLE jobs (**

**job\_title VARCHAR(255) NOT NULL,**

**department VARCHAR(255) NOT NULL,**

**location VARCHAR(255) NOT NULL,**

**responsibilities VARCHAR(255) NOT NULL,**

**qualifications VARCHAR(255) NOT NULL,**

**salary\_range INT NOT NULL,**

**PRIMARY KEY (job\_title)**

**);**

**CREATE TABLE applicants (**

**applicant\_id INT NOT NULL AUTO\_INCREMENT,**

**name VARCHAR(255) NOT NULL,**

**contact\_information VARCHAR(255) NOT NULL,**

**resume VARCHAR(255) NOT NULL,**

**cover\_letter VARCHAR(255) NOT NULL,**

**application\_status VARCHAR(255) NOT NULL,**

**PRIMARY KEY (applicant\_id)**

**);**

**CREATE TABLE recruitments (**

**job\_requisition\_number INT NOT NULL AUTO\_INCREMENT,**

**recruiting\_source VARCHAR(255) NOT NULL,**

**interview\_notes VARCHAR(255) NOT NULL,**

**offer\_letter VARCHAR(255) NOT NULL,**

**PRIMARY KEY (job\_requisition\_number)**

**);**

**CREATE TABLE performance\_reviews (**

**performance\_review\_id INT NOT NULL AUTO\_INCREMENT,**

**employee\_id INT NOT NULL,**

**job\_title VARCHAR(255) NOT NULL,**

**performance\_goals VARCHAR(255) NOT NULL,**

**ratings VARCHAR(255) NOT NULL,**

**feedback VARCHAR(255) NOT NULL,**

**PRIMARY KEY (performance\_review\_id),**

**FOREIGN KEY (employee\_id) REFERENCES employees (employee\_id)**

**);**

**ALTER TABLE performance\_reviews**

**ADD CONSTRAINT fk\_performance\_reviews\_job\_title**

**FOREIGN KEY (job\_title) REFERENCES jobs (job\_title);**

**3. write queries to insert data into your tables.**

**INSERT INTO employees (name, contact\_information, job\_title, department, date\_of\_hire, education, skills, experience)**

**VALUES ('John Doe', 'john.doe@email.com', 'Software Engineer', 'Engineering', '2023-03-08', 'BS in Computer Science', 'Java, Python, SQL', '3 years');**

**INSERT INTO jobs (job\_title, department, location, responsibilities, qualifications, salary\_range)**

**VALUES ('Software Engineer', 'Engineering', 'Kigali, Rwanda', 'Design, develop, and test software applications', 'BS in Computer Science', 'USD 50,000 - 100,000');**

**INSERT INTO applicants (name, contact\_information, resume, cover\_letter, application\_status)**

**VALUES ('Jane Doe', 'jane.doe@email.com', 'resume.pdf', 'cover\_letter.pdf', 'Pending');**

**INSERT INTO recruitments (job\_requisition\_number, recruiting\_source, interview\_notes, offer\_letter)**

**VALUES (123456, 'LinkedIn', 'Good candidate, but not the best fit for the role', NULL);**

**INSERT INTO performance\_reviews (employee\_id, job\_title, performance\_goals, ratings, feedback)**

**VALUES (1, 'Software Engineer', 'Meet all deadlines, maintain high code quality, and work well with others', 'Excellent', 'Great job!');**

**4. Write queries to display all the information in your tables.**

**SELECT \* FROM employees;**

**SELECT \* FROM jobs;**

**SELECT \* FROM applicants;**

**SELECT \* FROM recruitments;**

**SELECT \* FROM performance\_reviews;**

**5. Write a query to update information in any of the two tables of your system**

**UPDATE employees**

**SET job\_title = 'Senior Software Engineer'**

**WHERE employee\_id = 1;**

**UPDATE employees**

**SET job\_title = 'Senior Software Engineer',**

**department = 'Engineering'**

**WHERE employee\_id = 1;**

**Section III**

**1. Create a view to insert data into your tables.**

**CREATE VIEW all\_employees AS**

**SELECT \***

**FROM employees;**

**INSERT INTO all\_employees (name, job\_title, department)**

**VALUES ('Jane Doe', 'Software Engineer', 'Engineering');**

**CREATE VIEW insert\_job AS**

**SELECT \* FROM jobs;**

**INSERT INTO insert\_job (job\_title, department, location, responsibilities, qualifications, salary\_range)**

**VALUES ('Software Engineer', 'Engineering', 'Kigali, Rwanda', 'Design, develop, and test software applications', 'BS in Computer Science', 'USD 50,000 - 100,000');**

**CREATE VIEW insert\_performance\_review AS**

**SELECT employee\_id,**

**job\_title,**

**performance\_goals,**

**ratings,**

**feedback**

**FROM performance\_reviews;**

**INSERT INTO insert\_performance\_review (employee\_id, job\_title, performance\_goals, ratings, feedback)**

**VALUES (1, 'Software Engineer', 'Meet all deadlines, maintain high code quality, and work well with others', 'Excellent', 'Great job!');**

**CREATE VIEW applicant\_review\_view AS**

**SELECT applicant\_id, name, contact\_information, resume, cover\_letter, application\_status,**

**(SELECT job\_title FROM jobs WHERE job\_title = applicant\_review.job\_title) AS job\_title,**

**(SELECT performance\_goals FROM performance\_reviews WHERE performance\_reviews.employee\_id = applicant\_review.applicant\_id) AS performance\_goals,**

**(SELECT ratings FROM performance\_reviews WHERE performance\_reviews.employee\_id = applicant\_review.applicant\_id) AS ratings,**

**(SELECT feedback FROM performance\_reviews WHERE performance\_reviews.employee\_id = applicant\_review.applicant\_id) AS feedback**

**FROM applicants applicant\_review;**

**INSERT INTO performance\_reviews (employee\_id, job\_title, performance\_goals, ratings, feedback)**

**SELECT 1, (SELECT job\_title FROM applicants WHERE applicant\_id = 1), 'Meet all deadlines, maintain high code quality, and work well with others', 'Excellent', 'Great job!'**

**FROM applicant\_review\_view;**

**2. Create a view to display all the information in your tables.**

**CREATE VIEW all\_info\_view AS**

**SELECT \***

**FROM employees**

**UNION ALL**

**SELECT \***

**FROM jobs**

**UNION ALL**

**SELECT \***

**FROM applicants**

**UNION ALL**

**SELECT \***

**FROM recruitments**

**UNION ALL**

**SELECT \***

**FROM performance\_reviews;**

**3. Create a view to update information in any of the two tables of your system.**

**CREATE OR REPLACE VIEW updatable\_employee\_info AS**

**SELECT employee\_id, First\_name, Last\_name, phone\_number, email, location**

**FROM employee;**

**UPDATE updatable\_employee\_info**

**SET name=’john doe'**

**WHERE lcation=’kigali’;**

**CREATE OR REPLACE VIEW updatable\_applicant\_info AS**

**SELECT applicant\_id, name,contact\_information,resume**

**FROM applicant;**

**UPDATE updatable\_applicant\_info**

**SET name='Jane Doe'**

**WHERE applicant\_id=’01’;**

**4. Create a view to delete data in any two of your tables according to any simple condition of your choice.**

**CREATE VIEW delete\_data\_view AS**

**SELECT \***

**FROM employees**

**WHERE department = 'Engineering';**

**DELETE FROM performance\_reviews**

**WHERE employee\_id IN (SELECT id FROM delete\_data\_view);**

**CREATE VIEW delete\_applicant\_view AS**

**SELECT applicant\_id**

**FROM applicants**

**WHERE application\_status = 'Rejected';**

**5. In your database, create one view of your choice that considers sub-query.**

**CREATE VIEW applicants\_with\_high\_salary\_view AS**

**SELECT applicant\_id, name, contact\_information, resume, cover\_letter, application\_status, salary**

**FROM applicants**

**WHERE salary > (SELECT AVG(salary) FROM applicants);**

**Section IV**

**1. Create a stored procedure to insert data into your tables.**

**CREATE PROCEDURE insert\_employee (**

**IN employee\_name VARCHAR(255),**

**IN contact\_information VARCHAR(255),**

**IN job\_title VARCHAR(255),**

**IN department VARCHAR(255),**

**IN date\_of\_hire DATE,**

**IN education VARCHAR(255),**

**IN skills VARCHAR(255),**

**IN experience VARCHAR(255)**

**)**

**BEGIN**

**INSERT INTO employees (name, contact\_information, job\_title, department, date\_of\_hire, education, skills, experience)**

**VALUES (employee\_name, contact\_information, job\_title, department, date\_of\_hire, education, skills, experience);**

**END;**

**2. Create a stored procedure to display all the information in your tables.**

**CREATE PROCEDURE show\_all\_employees ()**

**BEGIN**

**SELECT \***

**FROM employees;**

**END;**

**3. Create a stored procedure to update information in any of the two tables of your system.**

**CREATE PROCEDURE update\_employee (**

**IN employee\_id INT,**

**IN employee\_name VARCHAR(255),**

**IN contact\_information VARCHAR(255),**

**IN job\_title VARCHAR(255),**

**IN department VARCHAR(255),**

**IN date\_of\_hire DATE,**

**IN education VARCHAR(255),**

**IN skills VARCHAR(255),**

**IN experience VARCHAR(255)**

**)**

**BEGIN**

**UPDATE employees**

**SET name = employee\_name,**

**contact\_information = contact\_information,**

**job\_title = job\_title,**

**department = department,**

**date\_of\_hire = date\_of\_hire,**

**education = education,**

**skills = skills,**

**experience = experience**

**WHERE employee\_id = employee\_id;**

**END;**

**4. Create a stored procedure to delete data in any two of your tables according to any simple condition of your choice.**

**CREATE PROCEDURE update\_employee (**

**IN employee\_id INT,**

**IN employee\_name VARCHAR(255),**

**IN contact\_information VARCHAR(255),**

**IN job\_title VARCHAR(255),**

**IN department VARCHAR(255),**

**IN date\_of\_hire DATE,**

**IN education VARCHAR(255),**

**IN skills VARCHAR(255),**

**IN experience VARCHAR(255)**

**)**

**BEGIN**

**UPDATE employees**

**SET name = employee\_name,**

**contact\_information = contact\_information,**

**job\_title = job\_title,**

**department = department,**

**date\_of\_hire = date\_of\_hire,**

**education = education,**

**skills = skills,**

**experience = experience**

**WHERE employee\_id = employee\_id;**

**END;**

**5. In your database, stored the procedure view of your choice that considers sub-query**

**CREATE PROCEDURE get\_applicants\_with\_high\_salary ()**

**BEGIN**

**SELECT \***

**FROM applicants**

**WHERE salary > (**

**SELECT AVG(salary)**

**FROM applicants**

**);**

**END;**

**Section V**

**1. Create after inserting triggers for any two tables of your choice.**

**-CREATE TRIGGER after\_insert\_trigger**

**AFTER INSERT ON applicants**

**FOR EACH ROW**

**BEGIN**

**INSERT INTO recruitments (applicant\_id, job\_title)**

**VALUES (NEW.applicant\_id, NEW.job\_title);**

**END;**

**-CREATE TRIGGER after\_insert\_job**

**AFTER INSERT ON jobs**

**FOR EACH ROW**

**BEGIN**

**INSERT INTO job\_log (job\_id, job\_title, department, description, created\_at, updated\_at)**

**VALUES (NEW.job\_id, NEW.job\_title, NEW.department, NEW.description, NEW.created\_at, NEW.updated\_at);**

**END;**

**2. Create after-update triggers for any two tables of your choice.**

**-CREATE TRIGGER after\_update\_job\_salary**

**AFTER UPDATE ON jobs**

**FOR EACH ROW**

**BEGIN**

**IF NEW.salary > OLD.salary THEN**

**INSERT INTO salary\_increase\_log (job\_id, old\_salary, new\_salary, updated\_at)**

**VALUES (NEW.job\_id, OLD.salary, NEW.salary, NOW());**

**END IF;**

**END;**

**-** **CREATE TRIGGER after\_update\_job\_and\_applicant**

**AFTER UPDATE ON jobs, applicants**

**FOR EACH ROW**

**BEGIN**

**IF NEW.job\_title != OLD.job\_title THEN**

**UPDATE applicants**

**SET job\_title = NEW.job\_title**

**WHERE job\_id = NEW.job\_id;**

**END IF;**

**END;**

**3. Create after deleting triggers for any two tables of your choice**

**-CREATE TRIGGER after\_delete\_job\_and\_applicant**

**AFTER DELETE ON jobs, applicants**

**FOR EACH ROW**

**BEGIN**

**INSERT INTO deleted\_jobs (job\_id, job\_title, department, description, created\_at, updated\_at)**

**VALUES (OLD.job\_id, OLD.job\_title, OLD.department, OLD.description, OLD.created\_at, OLD.updated\_at);**

**INSERT INTO deleted\_applicants (applicant\_id, name, contact\_information, resume, cover\_letter, application\_status, job\_id)**

**VALUES (OLD.applicant\_id, OLD.name, OLD.contact\_information, OLD.resume, OLD.cover\_letter, OLD.application\_status, OLD.job\_id);**

**END;**

**Section VI**

**1. Create a user with your name as username and your student number as password and grant all privileges to the created user.**

**mysql -u root -p**

**create user 'nyangomaannet'@'localhost'IDENTIFIED BY '222001557';**

**GRANT ALL PRIVILEGES ON \*.\* TO 'nyangomaannet'@'localhost';**

**FLUSH PRIVILEGES;**

**exit;**

**2. Create a user with your "names\_semi" as username and your student number as password and give him insert, update, and delete privileges to the created user.**

**mysql -u root -p**

**CREATE USER 'nyangoma'@'localhost' IDENTIFIED BY '222001557';**

**GRANT INSERT, UPDATE, DELETE ON \*.\* TO 'nyangoma'@'localhost';**

**FLUSH PRIVILEGES;**

**exit;**

**3. Revoke insert privileges to the last user you created**

**mysql -u root -p**

**REVOKE INSERT ON \*.\* FROM 'nyangoma'@'localhost';**

**FLUSH PRIVILEGES;**

**exit;**