SUBMITED BY:

Uzair Hassan (48525)

Daniyal Wajid (48528)

SUBMITTED TO:

Sir Talha Tariq

SECTION: BSSE-4B

Project: Job Portal

Contents:

Introduction:

In today's competitive job market, finding the right job or hiring the right talent can be a daunting task. Traditional methods of job hunting and recruitment often lack efficiency and transparency, leading to frustration for both job seekers and employers. Recognizing this need, we propose to develop a state-of-the-art job portal that leverages technology to simplify the job search and recruitment process.

Objective:

The objective of this project is to develop a comprehensive job portal platform that connects job seekers with employers efficiently. The platform aims to provide a user-friendly interface for both job seekers and employers, offering features such as job search, resume posting, job posting, application management, and more. By creating a robust job portal, we aim to streamline the job search and recruitment process, ultimately helping job seekers find suitable employment opportunities and assisting employers in finding qualified candidates for their vacancies.

Goals:

Organize Data Efficiently:

- Create a structured and normalized database to manage data related to users, resumes, company profiles, job postings, and job applications.
- Ensure data is stored without redundancy, maintaining data integrity.

Facilitate Data Retrieval:

- Enable efficient retrieval of data through well-defined relationships between tables.
- Support queries that allow for comprehensive reports on users, job applications, resumes, companies, and job postings.

Support Data Manipulation:

- Provide mechanisms for adding, updating, and deleting records across various tables.
- Ensure that data manipulations maintain database consistency and integrity.

Implement Data Constraints:

- Apply appropriate constraints such as primary keys, foreign keys, unique constraints, and not null constraints to ensure data validity and integrity.
- Enforce referential integrity between tables using foreign key constraints.

Normalize Data:

- Ensure the database schema is normalized to at least the third normal form (3NF) to eliminate redundancy and avoid anomalies during data operations.

Use Advanced SQL Features:

- Implement advanced SQL queries including joins, subqueries, aggregate functions, and date functions to facilitate complex data retrieval and reporting.
- Demonstrate the ability to handle complex queries and provide meaningful insights from the data.

Enhance Database Security and Integrity:

- Implement secure handling of user data, ensuring sensitive information like passwords is stored securely.
- Apply constraints to enforce data integrity and prevent invalid data entries.

Core Functions:

User Management Functions

- CreateUser
- RetrieveUserInformation
- UpdateUserInformation
- DeleteUser

Resume Management Functions

- CreateResume
- RetrieveResume
- UpdateResume
- DeleteResume

Company Profile Management Functions

- CreateCompanyProfile
- RetrieveCompanyProfile
- UpdateCompanyProfile
- DeleteCompanyProfile

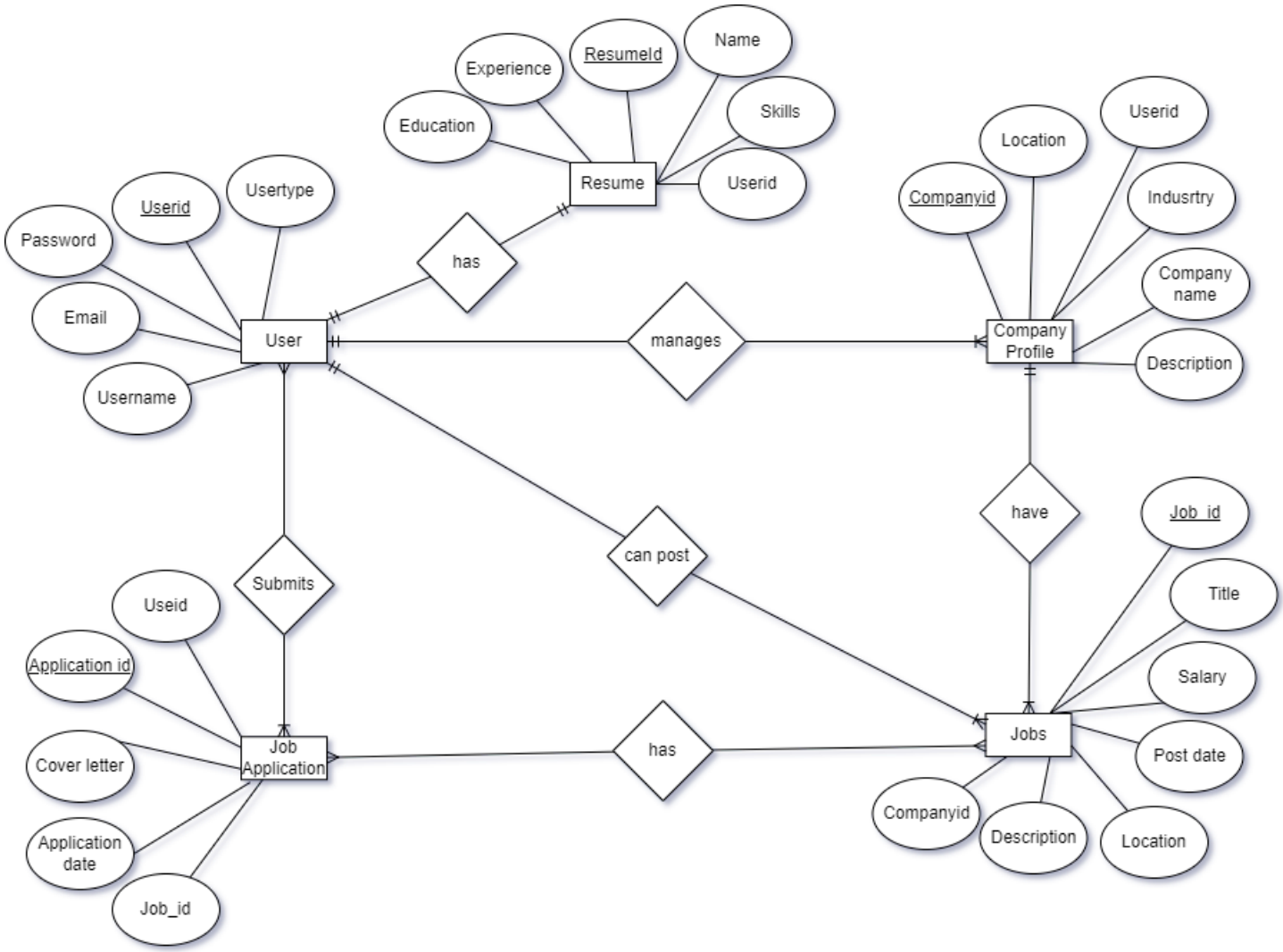
Job Management Functions

- CreateJob
- RetrieveJob
- UpdateJob
- DeleteJob

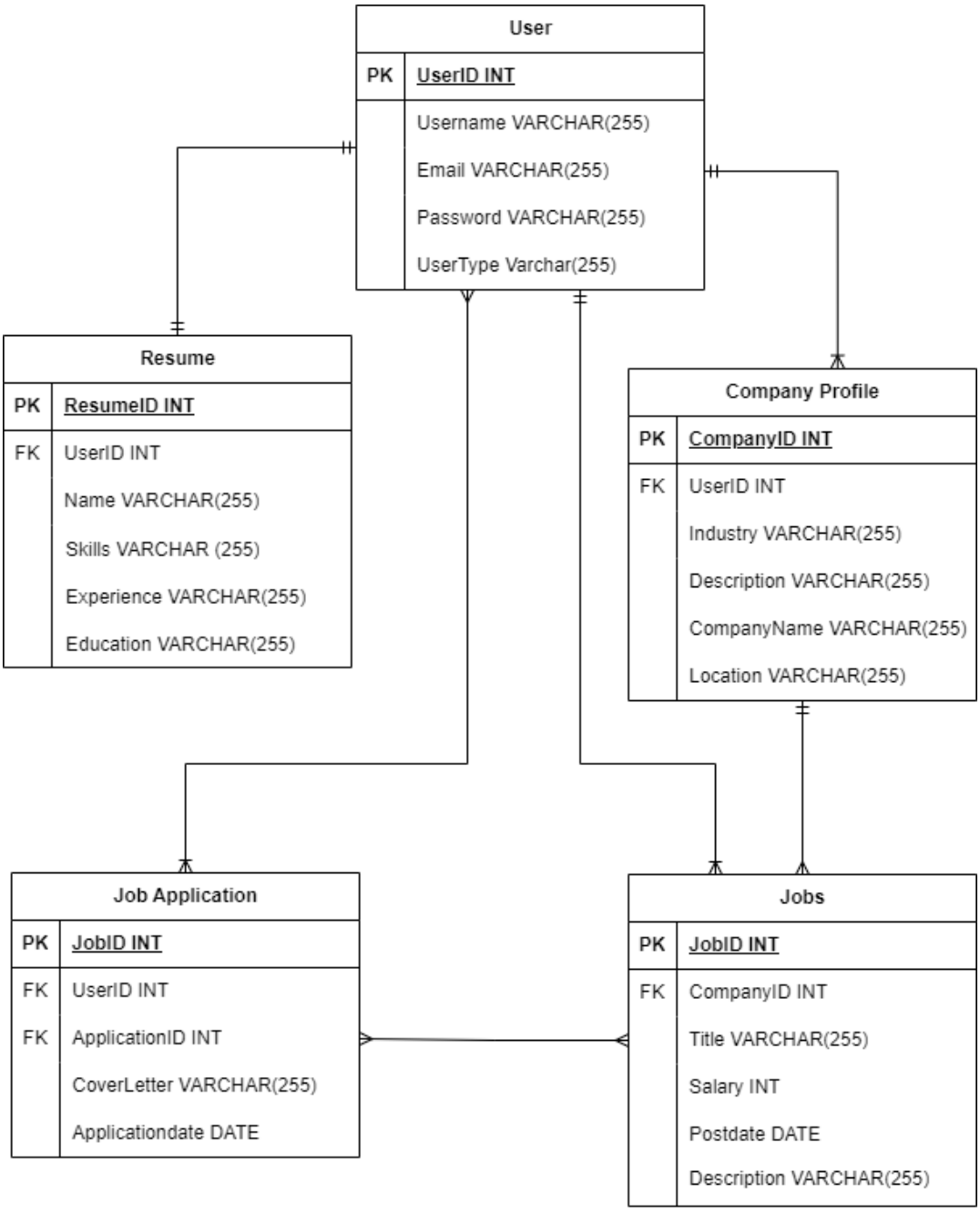
Job Application Management Functions

- SubmitJobApplication
- RetrieveJobApplication
- UpdateJobApplication
- DeleteJobApplication

ERD DIAGRAM


Name
ResumeId
Experience
Education
Skills
Resume
Userid
has
Userid
Usertype
Password
Email
User
Username
manages
Location
Userid
Companyid
Indusrtry
Company name
Company Profile
Description
have
Job_id
Title
Salary
Post date
Jobs
Companyid
Description
Location
can post
Submits
Useid
Application id
Cover letter
Application date
Job_id
Job Application
has

Relational Schema



Create and Insert Data in Users Table

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Username VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    UserType VARCHAR(255) NOT NULL
);
```

```
INSERT ALL
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (1, 'ahmed_ali', 'ahmed.ali@example.com', 'password1', 'jobseeker')
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (2, 'sara_khan', 'sara.khan@example.com', 'password2', 'jobseeker')
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (3, 'umar_iftikhar', 'umar.iftikhar@example.com', 'password3', 'jobseeker')
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (4, 'maria_butt', 'maria.butt@example.com', 'password4', 'employer')
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (5, 'zeeshan_haider', 'zeeshan.haider@example.com', 'password5', 'employer')
    INTO Users (UserID, Username, Email, Password, UserType) VALUES (6, 'aisha_naz', 'aisha.naz@example.com', 'password6', 'employer')
SELECT 1 FROM DUAL;
```

USERID	USERNAME	EMAIL	PASSWORD	USERTYPE
1	ahmed_ali	ahmed.ali@example.com	password1	jobseeker
2	sara_khan	sara.khan@example.com	password2	jobseeker
3	umar_iftikhar	umar.iftikhar@example.com	password3	jobseeker
4	maria_butt	maria.butt@example.com	password4	employer
5	zeeshan_haider	zeeshan.haider@example.com	password5	employer
6	aisha_naz	aisha.naz@example.com	password6	employer

6 rows returned in 0.00 seconds CSV Export

Create and Insert Data in Resume Table

```
CREATE TABLE Resume (
    ResumeID INT PRIMARY KEY,
    UserID INT,
    Name VARCHAR(255),
    Skills VARCHAR(255),
    Experience VARCHAR(255),
    Education VARCHAR(255),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

```
INSERT ALL
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (1, 1, 'Ahmed Ali', 'Java, C++', '3 years', 'BS Computer Science')
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (2, 2, 'Sara Khan', 'Python, SQL', '2 years', 'BS Information Technology')
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (3, 3, 'Umar Iftikhar', 'PHP, JavaScript', '4 years', 'BS Software Engineering')
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (4, 1, 'Ahmed Ali', 'HTML, CSS', '3 years', 'BS Computer Science')
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (5, 2, 'Sara Khan', 'R, MATLAB', '2 years', 'BS Information Technology')
    INTO Resume (ResumeID, UserID, Name, Skills, Experience, Education) VALUES (6, 3, 'Umar Iftikhar', 'Django, Flask', '4 years', 'BS Software Engineering')
SELECT 1 FROM DUAL;
```

RESUMEID	USERID	NAME	SKILLS	EXPERIENCE	EDUCATION
1	1	Ahmed Ali	Java, C++	3 years	BS Computer Science
2	2	Sara Khan	Python, SQL	2 years	BS Information Technology
3	3	Umar Iftikhar	PHP, JavaScript	4 years	BS Software Engineering
4	1	Ahmed Ali	HTML, CSS	3 years	BS Computer Science
5	2	Sara Khan	R, MATLAB	2 years	BS Information Technology
6	3	Umar Iftikhar	Django, Flask	4 years	BS Software Engineering

6 rows returned in 0.00 seconds CSV Export

Create and Insert Data in Company Profile

```
CREATE TABLE CompanyProfile (
    CompanyID INT PRIMARY KEY,
    UserID INT,
    Industry VARCHAR(255),
    Description VARCHAR(255),
    CompanyName VARCHAR(255),
    Location VARCHAR(255),
    FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
```

```
INSERT ALL
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (1, 4, 'IT', 'Software Development', 'TechVision', 'Karachi')
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (2, 5, 'Finance', 'Financial Services', 'FinCorp', 'Lahore')
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (3, 6, 'Healthcare', 'Healthcare Solutions', 'HealthTech', 'Islamabad')
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (4, 4, 'Education', 'Educational Services', 'EduFuture', 'Karachi')
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (5, 5, 'Retail', 'Retail Solutions', 'RetailHub', 'Lahore')
    INTO CompanyProfile (CompanyID, UserID, Industry, Description, CompanyName, Location) VALUES (6, 6, 'Construction', 'Construction Services', 'BuildRight', 'Islamabad')
SELECT 1 FROM DUAL;
```

COMPANYID	USERID	INDUSTRY	DESCRIPTION	COMPANYNAME	LOCATION
1	4	IT	Software Development	TechVision	Karachi
2	5	Finance	Financial Services	FinCorp	Lahore
3	6	Healthcare	Healthcare Solutions	HealthTech	Islamabad
4	4	Education	Educational Services	EduFuture	Karachi
5	5	Retail	Retail Solutions	RetailHub	Lahore
6	6	Construction	Construction Services	BuildRight	Islamabad

6 rows returned in 0.00 seconds CSV Export

Create and Insert Data in Jobs

```
CREATE TABLE Jobs (
    JobID INT PRIMARY KEY,
    CompanyID INT,
    Title VARCHAR(255),
    Salary INT,
    Postdate DATE,
    Description VARCHAR(255),
    FOREIGN KEY (CompanyID) REFERENCES CompanyProfile(CompanyID)
);
```

```
INSERT ALL
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (1, 1, 'Software Engineer', 70000, TO_DATE('2024-06-01', 'YYYY-MM-DD'), 'Develop software
applications')
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (2, 2, 'Financial Analyst', 80000, TO_DATE('2024-06-02', 'YYYY-MM-DD'), 'Analyze financial data')
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (3, 3, 'Healthcare Specialist', 75000, TO_DATE('2024-06-03', 'YYYY-MM-DD'), 'Provide healthcare
solutions')
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (4, 4, 'Education Consultant', 60000, TO_DATE('2024-06-04', 'YYYY-MM-DD'), 'Consult educational
services')
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (5, 5, 'Retail Manager', 65000, TO_DATE('2024-06-05', 'YYYY-MM-DD'), 'Manage retail operations')
    INTO Jobs (JobID, CompanyID, Title, Salary, Postdate, Description) VALUES (6, 6, 'Construction Manager', 90000, TO_DATE('2024-06-06', 'YYYY-MM-DD'), 'Manage construction
projects')
SELECT 1 FROM DUAL;
```

JOBID	COMPANYID	TITLE	SALARY	POSTDATE	DESCRIPTION
1	1	Software Engineer	70000	01-JUN-24	Develop software applications
2	2	Financial Analyst	80000	02-JUN-24	Analyze financial data
3	3	Healthcare Specialist	75000	03-JUN-24	Provide healthcare solutions
4	4	Education Consultant	60000	04-JUN-24	Consult educational services
5	5	Retail Manager	65000	05-JUN-24	Manage retail operations
6	6	Construction Manager	90000	06-JUN-24	Manage construction projects

6 rows returned in 0.00 seconds CSV Export

Create and Insert Data in Jobs Application

```
CREATE TABLE JobApplication (
    ApplicationID INT PRIMARY KEY,
    UserID INT,
    JobID INT,
    CoverLetter VARCHAR(255),
    Applicationdate DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (JobID) REFERENCES Jobs(JobID)
);
```

```
INSERT ALL
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (1, 1, 1, 'Interested in Software Engineer position.', TO_DATE('2024-06-10', 'YYYY-MM-DD'))
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (2, 2, 2, 'Interested in Financial Analyst position.', TO_DATE('2024-06-11', 'YYYY-MM-DD'))
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (3, 3, 3, 'Interested in Healthcare Specialist position.', TO_DATE('2024-06-12', 'YYYY-MM-DD'))
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (4, 1, 4, 'Interested in Education Consultant position.', TO_DATE('2024-06-13', 'YYYY-MM-DD'))
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (5, 2, 5, 'Interested in Retail Manager position.', TO_DATE('2024-06-14', 'YYYY-MM-DD'))
    INTO JobApplication (ApplicationID, UserID, JobID, CoverLetter, Applicationdate) VALUES (6, 3, 6, 'Interested in Construction Manager position.', TO_DATE('2024-06-15', 'YYYY-MM-DD'))
SELECT 1 FROM DUAL;
```

APPLICATIONID	USERID	JOBID	COVERLETTER	APPLICATIONDATE
1	1	1	Interested in Software Engineer position.	10-JUN-24
2	2	2	Interested in Financial Analyst position.	11-JUN-24
3	3	3	Interested in Healthcare Specialist position.	12-JUN-24
4	1	4	Interested in Education Consultant position.	13-JUN-24
5	2	5	Interested in Retail Manager position.	14-JUN-24
6	3	6	Interested in Construction Manager position.	15-JUN-24

6 rows returned in 0.00 seconds CSV Export

JOINS Queries

Get all Jobs with Company Details

```
SELECT -- Get all jobs with company details
    Jobs.JobID,
    Jobs.Title,
    Jobs.Salary,
    Jobs.Postdate,
    CompanyProfile.CompanyName,
    CompanyProfile.Location
FROM
    Jobs
JOIN
    CompanyProfile ON Jobs.CompanyID = CompanyProfile.CompanyID;
```

JOBID	TITLE	SALARY	POSTDATE	COMPANYNAME	LOCATION
1	Software Engineer	70000	01-JUN-24	TechVision	Karachi
2	Financial Analyst	80000	02-JUN-24	FinCorp	Lahore
3	Healthcare Specialist	75000	03-JUN-24	HealthTech	Islamabad
4	Education Consultant	60000	04-JUN-24	EduFuture	Karachi
5	Retail Manager	65000	05-JUN-24	RetailHub	Lahore
6	Construction Manager	90000	06-JUN-24	BuildRight	Islamabad

6 rows returned in 0.00 seconds CSV Export

JOINS Queries

Get all Job Applicants with User and Job details

```
SELECT  --Get all job applications with user and job details
    JobApplication.ApplicationID,
    Users.Username,
    Jobs.Title,
    JobApplication.CoverLetter,
    JobApplication.Applicationdate
FROM
    JobApplication
JOIN
    Users ON JobApplication.UserID = Users.UserID
JOIN
    Jobs ON JobApplication.JobID = Jobs.JobID;
```

APPLICATIONID	USERNAME	TITLE	COVERLETTER	APPLICATIONDATE
1	ahmed_ali	Software Engineer	Interested in Software Engineer position.	10-JUN-24
2	sara_khan	Financial Analyst	Interested in Financial Analyst position.	11-JUN-24
3	umar_iftikhar	Healthcare Specialist	Interested in Healthcare Specialist position.	12-JUN-24
4	ahmed_ali	Education Consultant	Interested in Education Consultant position.	13-JUN-24
5	sara_khan	Retail Manager	Interested in Retail Manager position.	14-JUN-24
6	umar_iftikhar	Construction Manager	Interested in Construction Manager position.	15-JUN-24

6 rows returned in 0.00 seconds CSV Export

Nested Queries

Get the user detail who applied for a specific job having a specific user id

```
SELECT -- Get the user details for those who applied for a specific job (e.g., JobID = 1)
    Username,
    Email
FROM
    Users
WHERE
    UserID IN (
        SELECT
            UserID
        FROM
            JobApplication
        WHERE
            JobID = 1
    );
```

USERNAME	EMAIL
ahmed_ali	ahmed.ali@example.com

1 rows returned in 0.00 seconds CSV Export

Nested Queries

Get the job title and company names for jobs applied by a specific user

```
SELECT    -- Get the job titles and company names for jobs applied by a specific user (e.g., UserID = 1)
    Jobs.Title,
    CompanyProfile.CompanyName
FROM
    Jobs
JOIN
    CompanyProfile ON Jobs.CompanyID = CompanyProfile.CompanyID
WHERE
    Jobs.JobID IN (
        SELECT
            JobID
        FROM
            JobApplication
        WHERE
            UserID = 1
    );
```

TITLE	COMPANYNAME
Software Engineer	TechVision
Education Consultant	EduFuture

2 rows returned in 0.00 seconds CSV Export

Aggregate Functions

Count the number of job applications per user

```
SELECT -- Count the number of job applications per user
    Users.Username,
    COUNT(JobApplication.ApplicationID) AS ApplicationCount
FROM
    Users
JOIN
    JobApplication ON Users.UserID = JobApplication.UserID
GROUP BY
    Users.Username;
```

USERNAME	APPLICATIONCOUNT
sara_khan	2
ahmed_ali	2
umar_iftikhar	2

3 rows returned in 0.00 seconds CSV Export

Aggregate Functions

Get the average salary of all jobs

```
SELECT  -- Get the average salary of all jobs
    AVG(Salary) AS AverageSalary
FROM
    Jobs;
```

AVERAGESALARY
73333.333333333333333333333333333333333

1 rows returned in 0.00 seconds CSV Export

Nested Queries

Get the latest job posted by each company

```
SELECT   -- Get the latest job posted by each company
    CompanyID,
    Title,
    Postdate
FROM
    Jobs
WHERE
    (CompanyID, Postdate) IN (
        SELECT
            CompanyID,
            MAX(Postdate)
        FROM
            Jobs
        GROUP BY
            CompanyID
    );
```

COMPANYID	TITLE	POSTDATE
1	Software Engineer	01-JUN-24
2	Financial Analyst	02-JUN-24
3	Healthcare Specialist	03-JUN-24
4	Education Consultant	04-JUN-24
5	Retail Manager	05-JUN-24
6	Construction Manager	06-JUN-24

6 rows returned in 0.00 seconds CSV Export

Co related Query

Find users who have applied for jobs in companies located in 'Karachi'.

```
SELECT
    U.Username
FROM
    Users U
WHERE
    EXISTS (
        SELECT
            1
        FROM
            JobApplication JA
        JOIN
            Jobs J ON JA.JobID = J.JobID
        JOIN
            CompanyProfile C ON J.CompanyID = C.CompanyID
        WHERE
            JA.UserID = U.UserID
            AND C.Location = 'Karachi'
    );
-----------------------------------
```

USERNAME
ahmed_ali

1 rows returned in 0.00 seconds CSV Export

USING ANY ALL NOT Queries

Any Query Find users who have applied to a job with any salary greater than 75000

```
SELECT  --ANY query(Find users who have applied to a job with any salary greater than 75000.)
    Username
FROM
    Users
WHERE
    UserID IN (
        SELECT
            UserID
        FROM
            JobApplication
        WHERE
            JobID = ANY (
                SELECT
                    JobID
                FROM
                    Jobs
                WHERE
                    Salary > 75000
            )
    );
```

USERNAME
sara_khan
umar_iftikhar

2 rows returned in 0.01 seconds CSV Export

All Query Find jobs with a salary greater than all jobs in the IT industry.)

```
SELECT --All query(Find jobs with a salary greater than all jobs in the IT industry.)
    Title
FROM
    Jobs
WHERE
    Salary > ALL (
        SELECT
            Salary
        FROM
            Jobs J
        JOIN
            CompanyProfile C ON J.CompanyID = C.CompanyID
        WHERE
            C.Industry = 'IT'
    );
```

TITLE
Financial Analyst
Healthcare Specialist
Construction Manager

3 rows returned in 0.00 seconds CSV Export

NOT IN Query (Find users who have not applied to any job.)

```
SELECT --NOT IN Query (Find users who have not applied to any job.)
    Username
FROM
    Users
WHERE
    UserID NOT IN (
        SELECT
            UserID
        FROM
            JobApplication
    );
```

USERNAME
maria_butt
zeeshan_haider
aisha_naz

3 rows returned in 0.02 seconds CSV Export

Proof of Normalization

1NF (First Normal Form)

Definition: A table is in 1NF if all columns contain atomic, indivisible values, and each column contains values of a single type with unique names.

Proof for Each Table:

- **Users Table**:
  - **Attributes**: UserID, UserName, UserEmail, UserPassword
  - **Atomic Values**: Each attribute contains only atomic values.
  - **Single Type Values**: Each attribute contains values of a single type.
  - **Unique Names**: Each attribute has a unique name.
  - **Conclusion**: The Users table is in 1NF.
- **Resumes Table**:
  - **Attributes**: ResumeID, UserID, ResumeContent
  - **Atomic Values**: Each attribute contains only atomic values.
  - **Single Type Values**: Each attribute contains values of a single type.
  - **Unique Names**: Each attribute has a unique name.
  - **Conclusion**: The Resumes table is in 1NF.
- **Company Profiles Table**:
  - **Attributes**: CompanyID, CompanyName, CompanyLocation
  - **Atomic Values**: Each attribute contains only atomic values.
  - **Single Type Values**: Each attribute contains values of a single type.
  - **Unique Names**: Each attribute has a unique name.
  - **Conclusion**: The Company Profiles table is in 1NF.
- **Jobs Table**:
  - **Attributes**: JobID, CompanyID, JobTitle, Salary, Location
  - **Atomic Values**: Each attribute contains only atomic values.
  - **Single Type Values**: Each attribute contains values of a single type.
  - **Unique Names**: Each attribute has a unique name.
  - **Conclusion**: The Jobs table is in 1NF.
- **Job Applications Table**:
  - **Attributes**: ApplicationID, UserID, JobID, ApplicationDate
  - **Atomic Values**: Each attribute contains only atomic values.
  - **Single Type Values**: Each attribute contains values of a single type.
  - **Unique Names**: Each attribute has a unique name.
  - **Conclusion**: The Job Applications table is in 1NF.

2NF (Second Normal Form)

Definition: A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key.

Proof for Each Table:

- **Users Table**:
  - **Primary Key**: UserID
  - **Functional Dependencies**: UserID → UserName, UserEmail, UserPassword
  - **Full Dependency**: All non-key attributes fully depend on UserID.
  - **Conclusion**: The Users table is in 2NF.
- **Resumes Table**:
  - **Primary Key**: ResumeID
  - **Functional Dependencies**: ResumeID → UserID, ResumeContent
  - **Full Dependency**: All non-key attributes fully depend on ResumeID.
  - **Conclusion**: The Resumes table is in 2NF.
- **Company Profiles Table**:
  - **Primary Key**: CompanyID
  - **Functional Dependencies**: CompanyID → CompanyName, CompanyLocation
  - **Full Dependency**: All non-key attributes fully depend on CompanyID.
  - **Conclusion**: The Company Profiles table is in 2NF.
- **Jobs Table**:
  - **Primary Key**: JobID
  - **Functional Dependencies**: JobID → CompanyID, JobTitle, Salary, Location
  - **Full Dependency**: All non-key attributes fully depend on JobID.
  - **Conclusion**: The Jobs table is in 2NF.
- **Job Applications Table**:
  - **Primary Key**: ApplicationID
  - **Functional Dependencies**: ApplicationID → UserID, JobID, ApplicationDate
  - **Full Dependency**: All non-key attributes fully depend on ApplicationID.
  - **Conclusion**: The Job Applications table is in 2NF.

3NF (Third Normal Form)

Definition: A table is in 3NF if it is in 2NF and all attributes are functionally dependent only on the primary key.

Proof for Each Table:

- **Users Table**:
  - **Primary Key**: UserID
  - **Functional Dependencies**: UserID → UserName, UserEmail, UserPassword
  - **No Transitive Dependency**: No non-key attribute depends on another non-key attribute.
  - **Conclusion**: The Users table is in 3NF.
- **Resumes Table**:
  - **Primary Key**: ResumeID
  - **Functional Dependencies**: ResumeID → UserID, ResumeContent
  - **No Transitive Dependency**: No non-key attribute depends on another non-key attribute.
  - **Conclusion**: The Resumes table is in 3NF.
- **Company Profiles Table**:
  - **Primary Key**: CompanyID
  - **Functional Dependencies**: CompanyID → CompanyName, CompanyLocation
  - **No Transitive Dependency**: No non-key attribute depends on another non-key attribute.
  - **Conclusion**: The Company Profiles table is in 3NF.
- **Jobs Table**:
  - **Primary Key**: JobID
  - **Functional Dependencies**: JobID → CompanyID, JobTitle, Salary, Location
  - **No Transitive Dependency**: No non-key attribute depends on another non-key attribute.
  - **Conclusion**: The Jobs table is in 3NF.
- **Job Applications Table**:
  - **Primary Key**: ApplicationID
  - **Functional Dependencies**: ApplicationID → UserID, JobID, ApplicationDate
  - **No Transitive Dependency**: No non-key attribute depends on another non-key attribute.
  - **Conclusion**: The Job Applications table is in 3NF.

BCNF (Boyce-Codd Normal Form)

Definition: A table is in BCNF if it is in 3NF and for every functional dependency (X → Y), X is a superkey.

Proof for Each Table:

- **Users Table**:
  - **Primary Key**: UserID
  - **Functional Dependencies**: UserID → UserName, UserEmail, UserPassword
  - **Superkey**: UserID is a superkey.
  - **Conclusion**: The Users table is in BCNF.
- **Resumes Table**:
  - **Primary Key**: ResumeID
  - **Functional Dependencies**: ResumeID → UserID, ResumeContent
  - **Superkey**: ResumeID is a superkey.
  - **Conclusion**: The Resumes table is in BCNF.
- **Company Profiles Table**:
  - **Primary Key**: CompanyID
  - **Functional Dependencies**: CompanyID → CompanyName, CompanyLocation
  - **Superkey**: CompanyID is a superkey.
  - **Conclusion**: The Company Profiles table is in BCNF.
- **Jobs Table**:
  - **Primary Key**: JobID
  - **Functional Dependencies**: JobID → CompanyID, JobTitle, Salary, Location
  - **Superkey**: JobID is a superkey.
  - **Conclusion**: The Jobs table is in BCNF.
- **Job Applications Table**:
  - **Primary Key**: ApplicationID
  - **Functional Dependencies**: ApplicationID → UserID, JobID, ApplicationDate
  - **Superkey**: ApplicationID is a superkey.
  - **Conclusion**: The Job Applications table is in BCNF.

Conclusion

By thoroughly analyzing and proving the normalization process from 1NF through BCNF for each table in our job portal database schema, we ensure that our design is efficient, free from redundancy, and maintains data integrity.