JobConnect

Job Application Management System

Our Team

Megha Nair

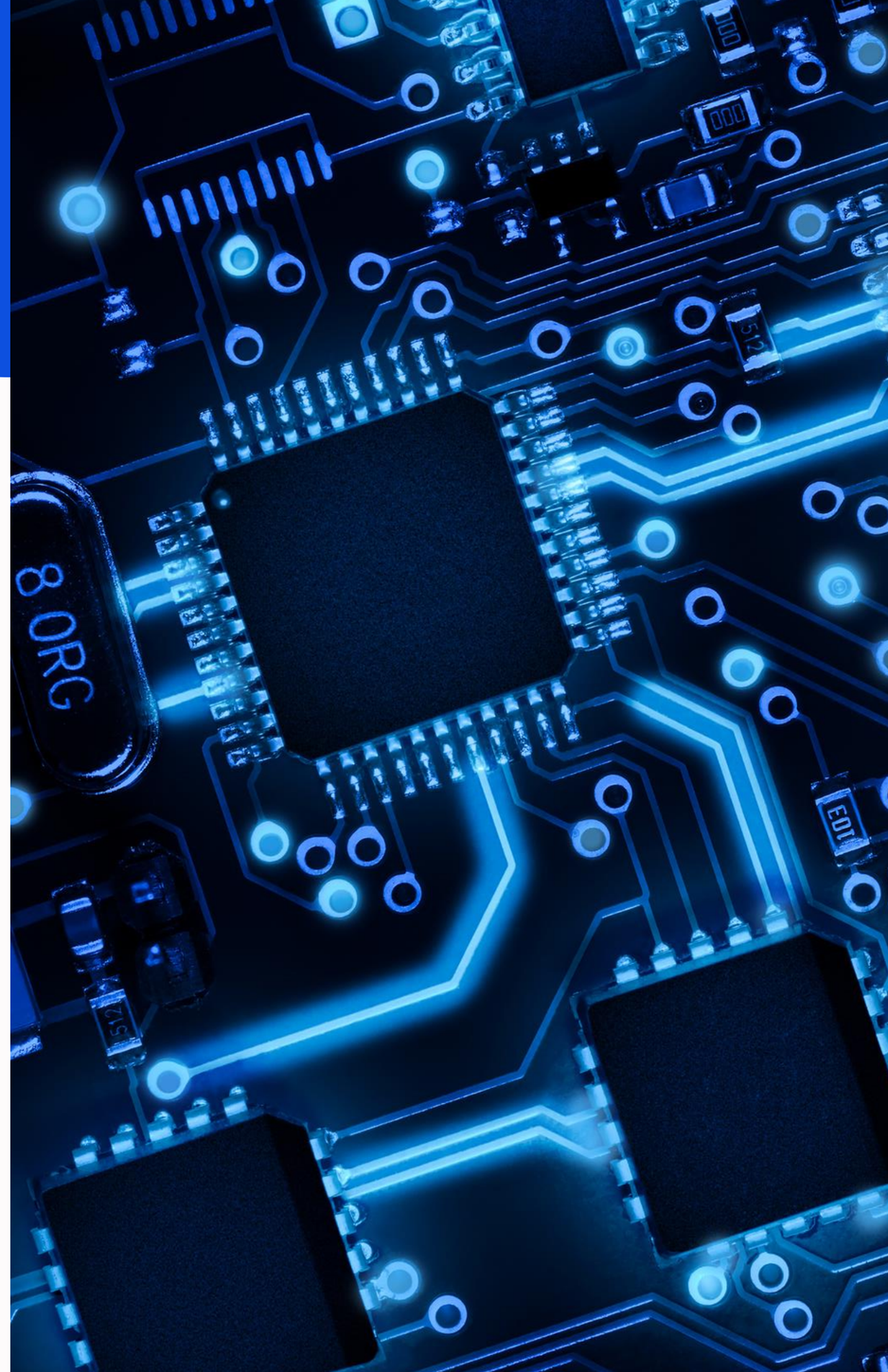
Urvi Chalodiya

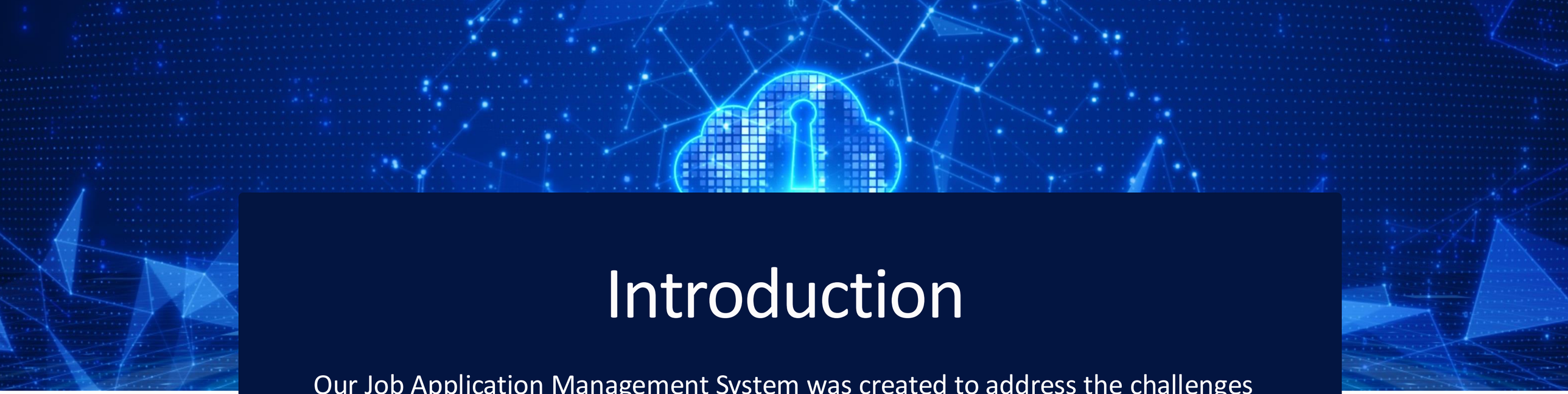
Chinmyee Gurav

Ameya Naik

Jay Tarde

Overview





Introduction

Our Job Application Management System was created to address the challenges faced by job seekers and hiring managers in today’s competitive market. The system simplifies the job search process by providing personalized tools for tracking applications and discovering opportunities while empowering administrators with efficient recruitment management capabilities. Its purpose is to bridge the gap between talent and opportunity, ensuring a seamless and effective hiring process for all.

Objective

Streamline	• Streamline the job search and application process for job seekers.
Enable	• Enable job seekers to save their information for easy and centralized access.
Enhance	• Enhance matching accuracy with personalized recommendations.
Facilitate	• Facilitate recruiting managers by streamlining access to applicant data.
Ensure	• Ensure a seamless and efficient connection between job seekers and employers.

The platform bridges the gap between job seekers and hiring managers, fostering collaboration and ensuring an effective hiring process.

System Workflow

Gathered Requirements:

Thoroughly analyzed the needs of job seekers and hiring managers to ensure the platform addressed critical challenges like tracking applications and managing job listings effectively.

Conceptual & Logical Design:

Created ER diagrams and data flow diagrams to define system functionality and relationships. Designed logical workflows for efficient interaction between jobs, users, and recruiters.

User Interface Design:

Developed a user-friendly interface for both job seekers and administrators. Designed dashboards for easy application tracking and streamlined job management.

Physical Design:

Built a robust database structure with tables for job details, user profiles, work authorizations, and application statuses. Ensured efficient storage and retrieval of data for seamless performance.

Website Workflow

This step-by-step workflow ensures a seamless interaction between job seekers, hiring managers, and administrators, creating an efficient and user-friendly platform.

TABLES

The core tables ensure smooth workflows and seamless functionality:

1. Job Seekers & Job Postings:

- Central entities enabling job seekers to find relevant opportunities.
- Recruiters list detailed job openings for targeted hiring.

2. Applications:

- Tracks the status and progress of every job application.
- Ensures transparency for both seekers and recruiters.

3. Keywords & Job Titles:

- Enhances searchability and recommendation precision.
- Links skills to job postings for targeted results.

4. Employers & Work Authorization:

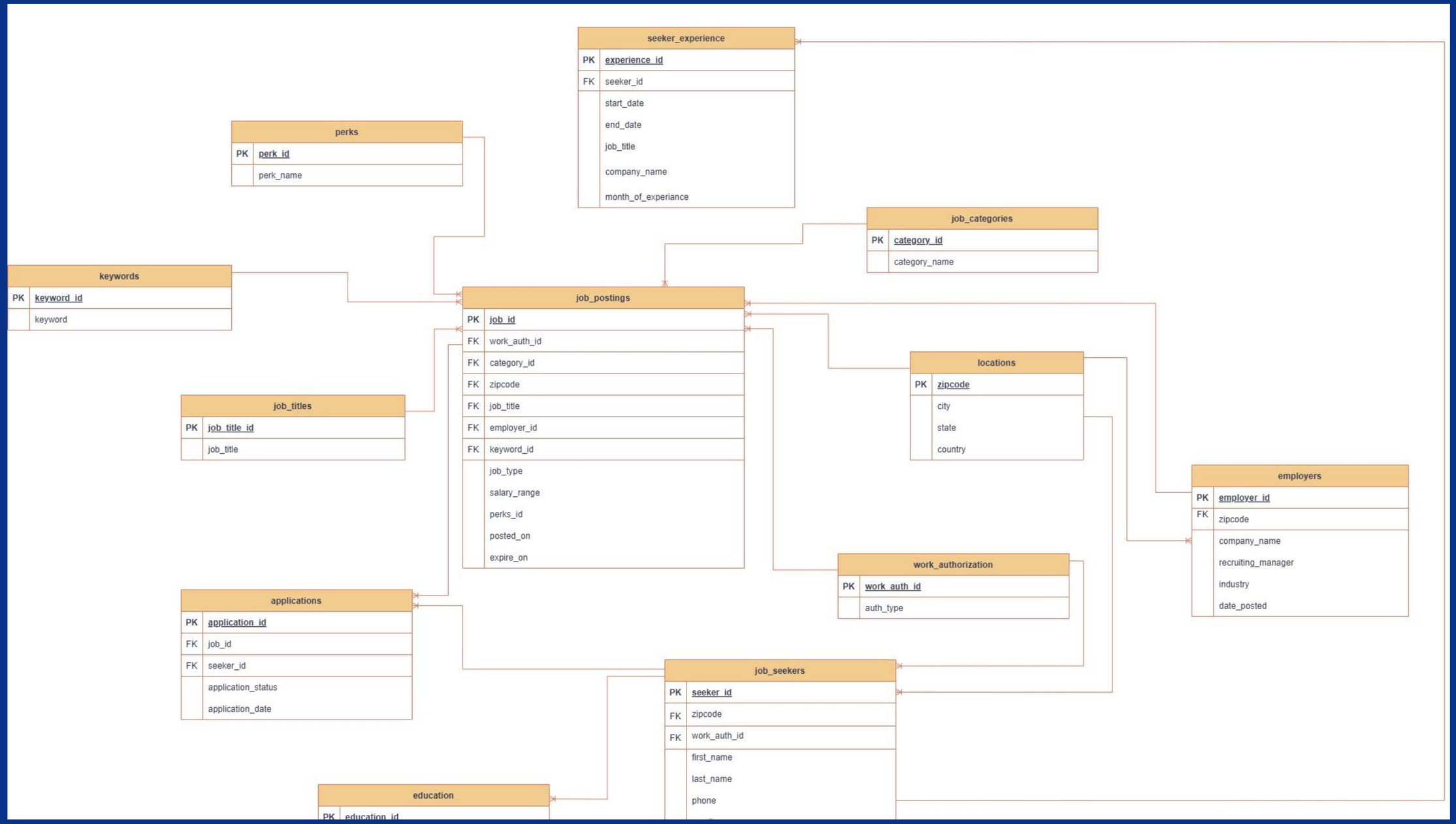
- Ensures compliance by managing company details and candidate eligibility.
- Matches seekers with jobs based on legal work authorization.

5. Admin Job Postings:

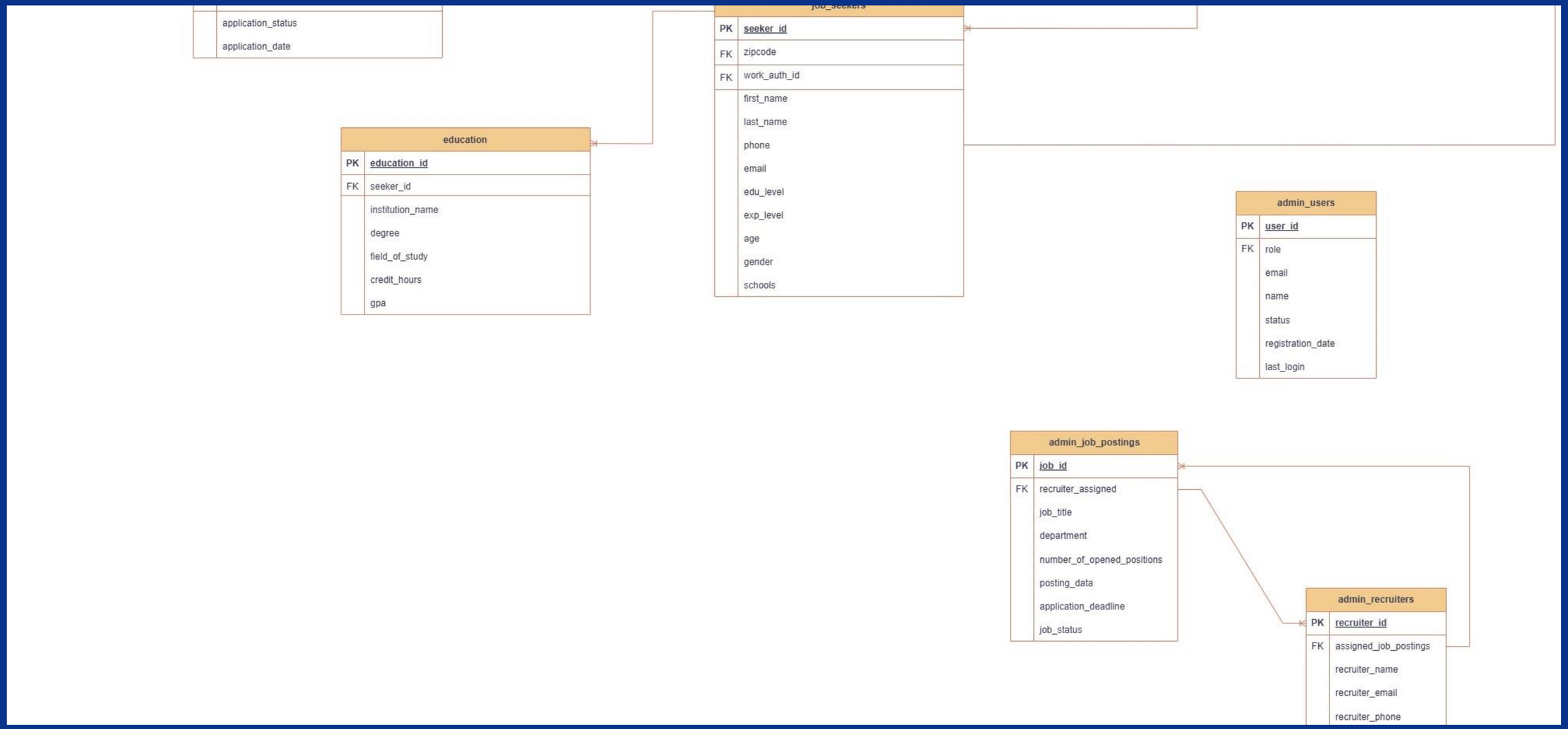
- Enables platform oversight for quality and relevance.
- Supports governance by managing deadlines and flagged postings.

These tables work together to create a robust, efficient, and user-focused platform, bridging the gap between job seekers and employers.

Entity Relationship Diagram

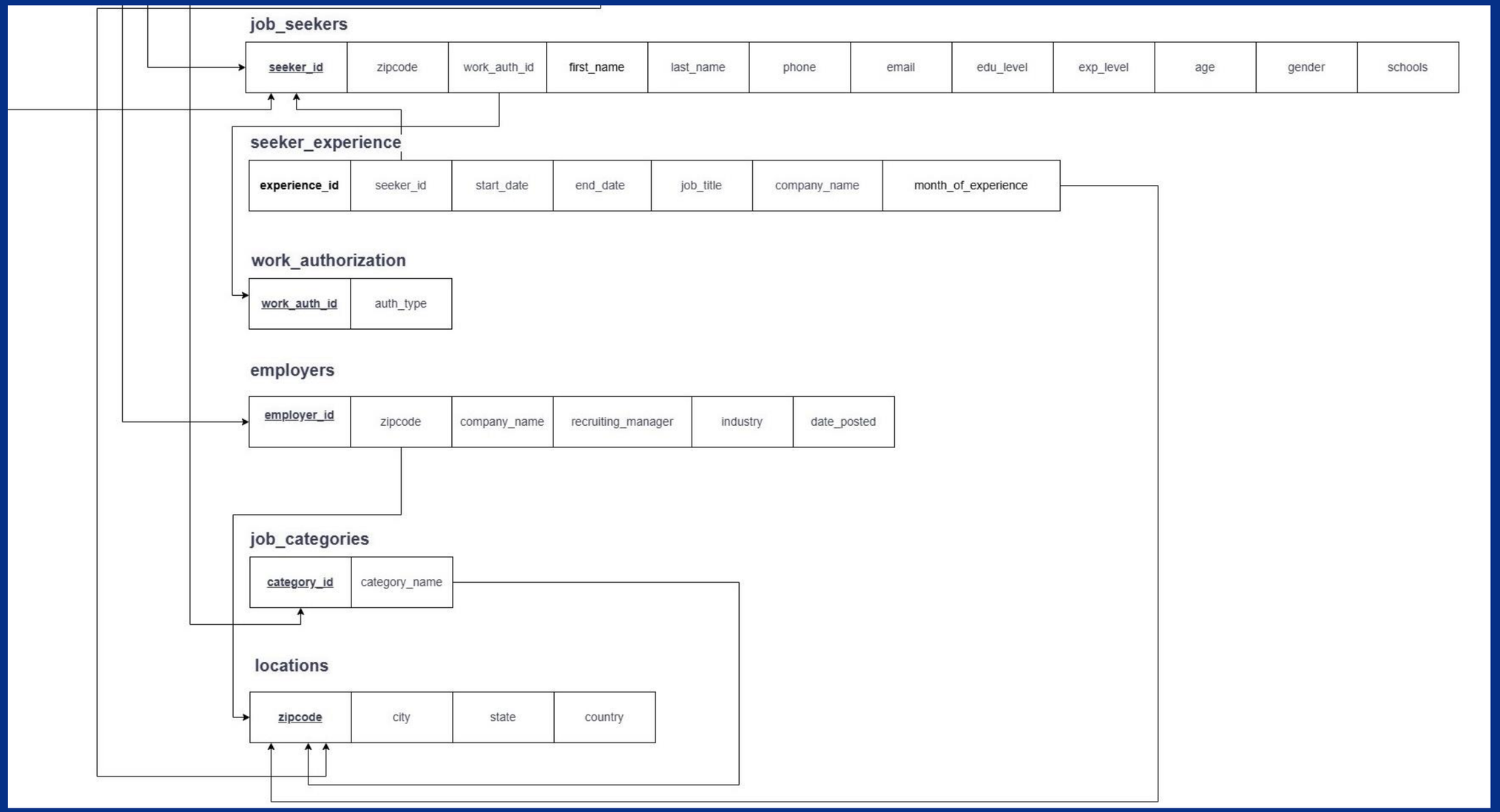


Entity Relationship Diagram

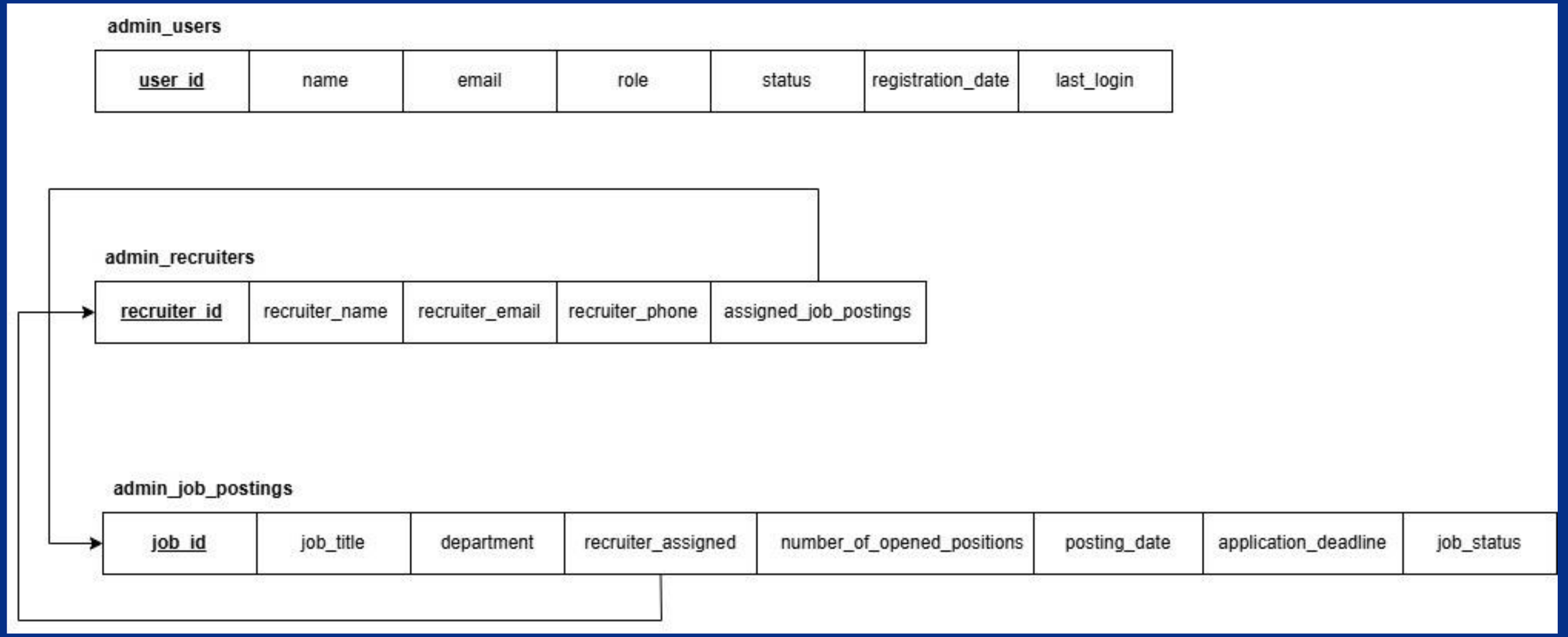


Relational Schema: Users

Relational Schema: Users



Relational Schema: Admin



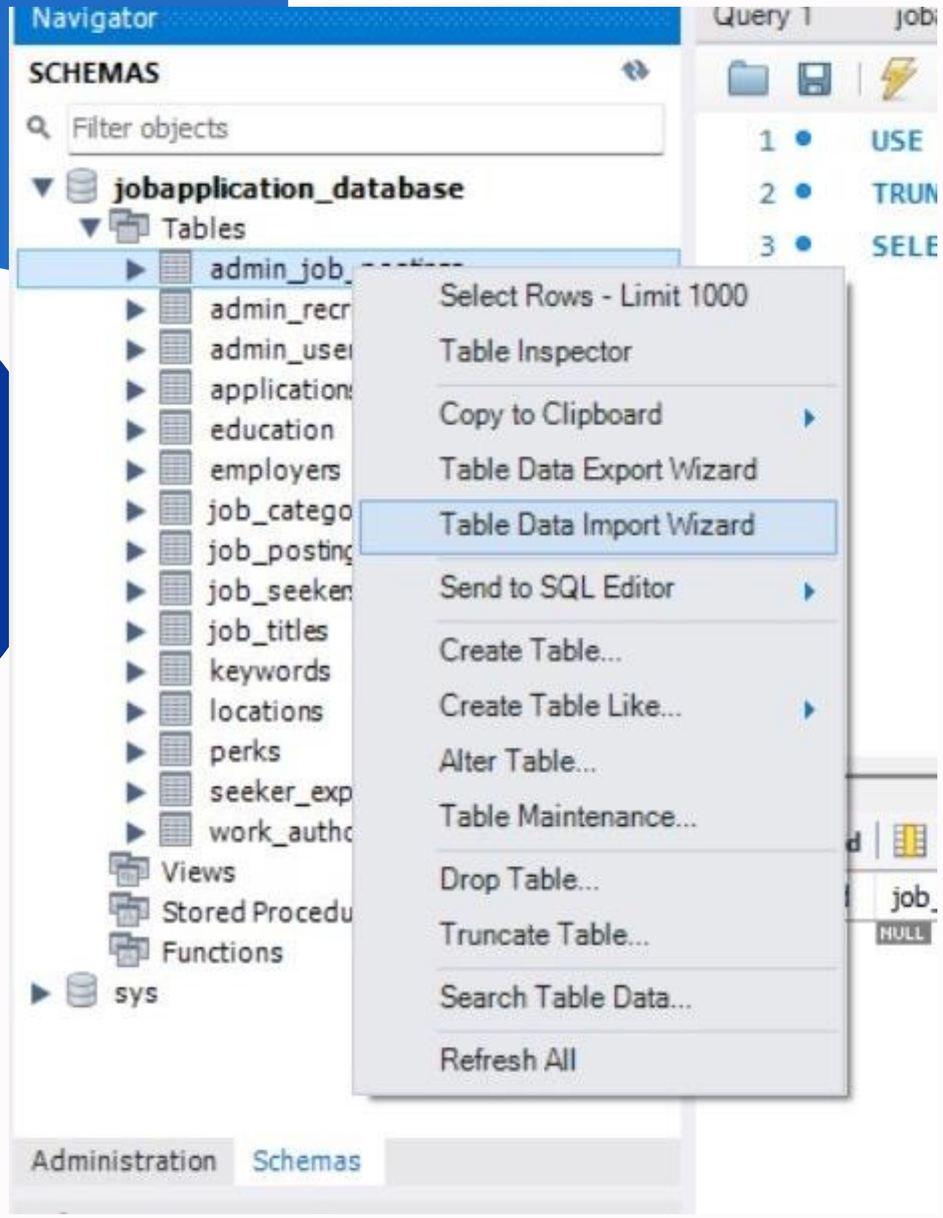
Data Collection & Import

Data Collection:

- Dummy data generated using Data Generator tool for all required entities.
- Additional data structured using Excel for enhanced flexibility and organization.
- Ensured diverse and realistic data points to simulate real-world scenarios effectively.

Data Import:

- Utilized INSERT INTO statements for adding data into MySQL tables.
- Created and structured CSV files for different entities to streamline the import process.
- Imported data into MySQL using the Data Import Wizard for efficiency and accuracy.


Navigator
SCHEMAS
Filter objects
jobapplication_database
Tables
admin_job_postings
admin_recr
admin_user
applications
education
employers
job_catego
job_posting
job_seeker
job_titles
keywords
locations
perks
seeker_exp
work_autho
Views
Stored Procedu
Functions
sys
Select Rows - Limit 1000
Table Inspector
Copy to Clipboard
Table Data Export Wizard
Table Data Import Wizard
Send to SQL Editor
Create Table...
Create Table Like...
Alter Table...
Table Maintenance...
Drop Table...
Truncate Table...
Search Table Data...
Refresh All
Administration
Schemas
Query 1
1 USE
2 TRUN
3 SELE
job_
NULL

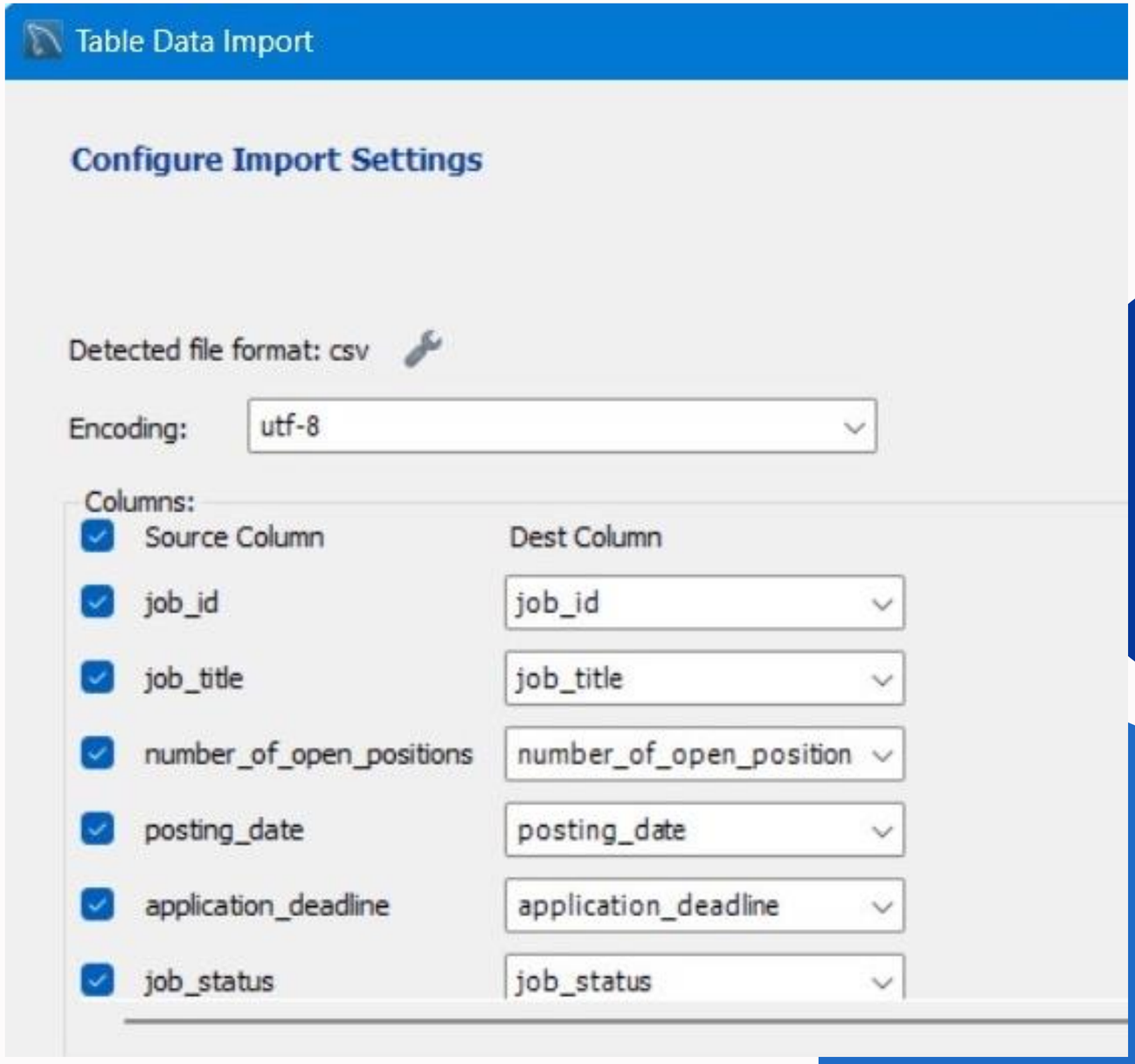


Table Data Import
Configure Import Settings
Detected file format: csv
Encoding:
utf-8
Columns:
Source Column
Dest Column
job_id
job_id
job_title
job_title
number_of_open_positions
number_of_open_position
posting_date
posting_date
application_deadline
application_deadline
job_status
job_status

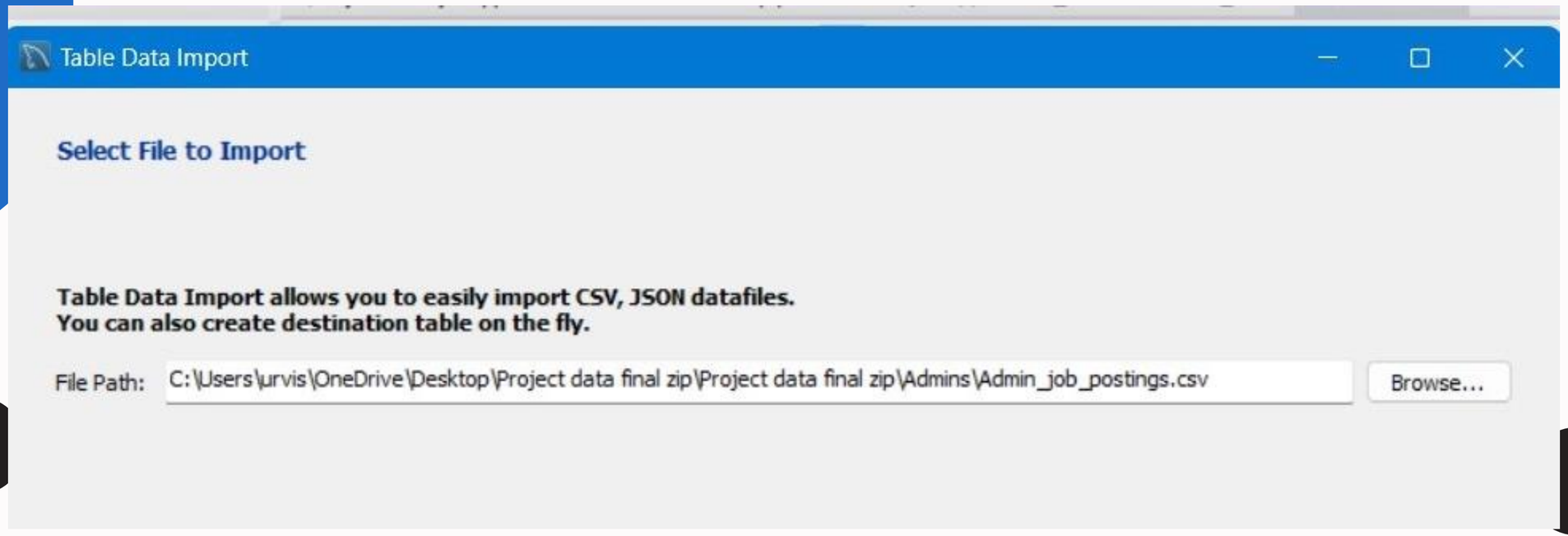


Table Data Import
Select File to Import
Table Data Import allows you to easily import CSV, JSON datafiles.
You can also create destination table on the fly.
File Path:
C:\Users\urvis\OneDrive\Desktop\Project data final zip\Project data final zip\Admins\Admin_job_postings.csv
Browse...

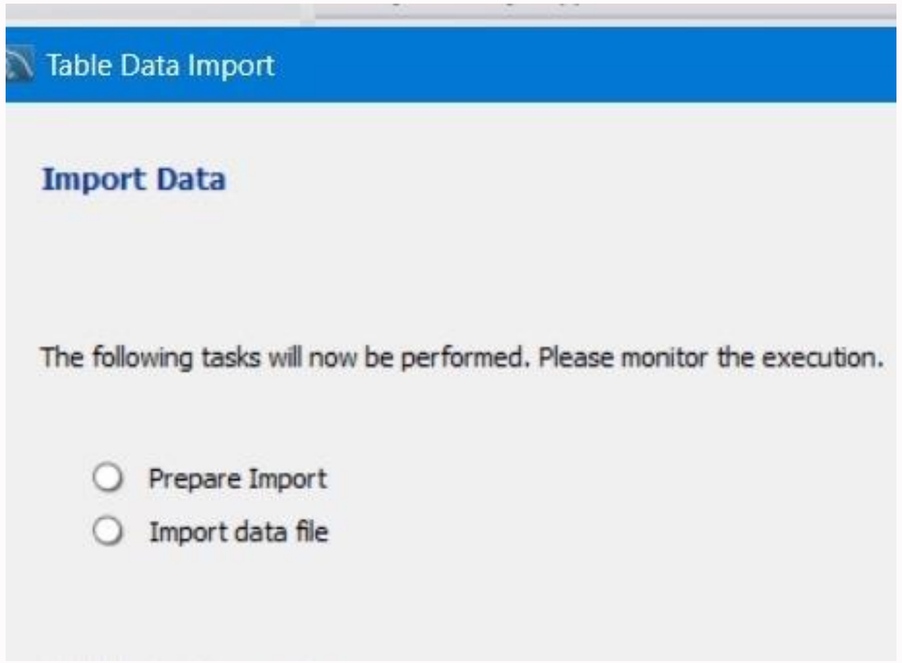


Table Data Import
Import Data
The following tasks will now be performed. Please monitor the execution.
Prepare Import
Import data file

Normalization

Step 1: Removed Repeating Groups (1NF):

Ensured all data was stored in tabular form with unique rows and columns, removing any multi-valued or repeating attributes from raw dummy data.

Step 2: Eliminated Partial Dependencies (2NF):

Separated composite key dependencies by creating separate tables for entities like job seekers, job postings, and employers, ensuring that every non-key attribute was fully dependent on the primary key.

Step 3: Removed Transitive Dependencies (3NF):

Identified and removed indirect dependencies by breaking down tables further, such as moving attributes like city and state into a separate locations table. This ensured that non-key attributes depended only on the primary key.

Outcome:

- Achieved a clean and optimized database structure, reducing redundancy and improving data consistency.
- Ensured easier data maintenance and scalability for future system enhancements.

SQL Queries and Output

1. Most Popular Job Categories

```sql
SELECT
    jc.category_name,
    COUNT(a.application_id) AS total_applications
FROM
    applications_clean a
JOIN
    job_postings_clean jp
    ON a.job_id = jp.job_id
JOIN
    job_categories_clean jc
    ON jp.category_id = jc.category_id
GROUP BY
    jc.category_name
ORDER BY
    total_applications DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

category_name	total_applications
UI/UX Design	250
System Administration	248
Project Management	248
Cloud Computing	204
IT Support	195
Cybersecurity	194
Software Development	182
Data Science	181
Product Management	154

2. Query to recommend job seekers whose GPA is greater than or equal to 3

```sql
SELECT DISTINCT js.seeker_id, js.first_name, js.last_name, js.email, e.gpa, wa.auth_type AS work_authorization
FROM job_seekers js
JOIN education e ON js.seeker_id = e.seeker_id
JOIN work_authorization wa ON js.work_auth_id = wa.work_auth_id
WHERE
    e.gpa >= 3
ORDER BY
    e.gpa DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

seeker_id	first_name	last_name	email	gpa	work_authorization
S0040	Erika	Martin	pamelahampton@hotmail.com	4	F1 CPT
S0031	Brittney	Stevens	brianna16@yahoo.com	3.99	F1 OPT
S0628	Krista	Decker	lisamartin@gmail.com	3.99	US Citizen
S0689	Joseph	Anthony	fjohnson@johnson.org	3.99	Permanent Resident
S0696	Diana	Taylor	jasonbriggs@wright.net	3.99	F1 CPT
S0936	Ashley	Young	stokesmarcus@yahoo.com	3.99	F1 CPT
S0179	Larry	Peters	kristinaday@meyer.com	3.98	US Citizen
S0401	Monique	Sutton	jacob01@anthony.org	3.98	US Citizen
S0450	Sarah	Anderson	christopher19@holloway.com	3.98	US Citizen
S0608	Todd	Whitehead	philipshah@smith-anderson.com	3.98	Permanent Resident
S0657	Richard	Hensley	hodgeskristin@miller.com	3.98	F1 CPT
S0680	Amanda	Turner	joshuapatterson@hotmail.com	3.98	Permanent Resident
S0135	Amber	Cunningham	andersonkimberly@hayden.com	3.97	US Citizen
S0255	Michelle	Greer	jfrancis@yahoo.com	3.97	F1 OPT

Result 78

SQL Queries and Output

3. Top Keywords Used in Job Postings

```sql
SELECT
    k.keyword,
    COUNT(jp.job_id) AS usage_count
FROM
    keywords k
JOIN
    job_postings_clean jp
    ON k.keyword_id = jp.keyword_id
WHERE
    STR_TO_DATE(jp.expire_on, '%Y-%m-%d') > CURDATE()
GROUP BY
    k.keyword
ORDER BY
    usage_count DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

keyword	usage_count
SQL	134
Machine Learning	62
Firewalls	55
Mongo Db	54
Redshift	48
API Development	47
Cloud Computing	43
Web Design	40
ITIL	35
DNS Configuration	34

4. Determine the average salary for job categories across different states

```sql
SELECT
    jc.category_name, l.state,
    ROUND(AVG(CAST(REPLACE(REPLACE(SUBSTRING_INDEX(jp.salary_range, '-', -1), '$', ''), 'k', '') AS UNSIGNED) * 1000), 2) AS avg
FROM
    job_postings_clean jp
JOIN
    job_categories_clean jc
    ON jp.category_id = jc.category_id
JOIN
    locations l
    ON jp.zipcode = l.zipcode
GROUP BY
    jc.category_name, l.state
ORDER BY
    avg_salary DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

seeker_id	first_name	last_name	email	gpa	work_authorization
S0040	Erika	Martin	pamelahampton@hotmail.com	4	F1 CPT
S0031	Brittney	Stevens	brianna16@yahoo.com	3.99	F1 OPT
S0628	Krista	Decker	lisamartin@gmail.com	3.99	US Citizen
S0689	Joseph	Anthony	fjohnson@johnson.org	3.99	Permanent Resident
S0696	Diana	Taylor	jasonbriggs@wright.net	3.99	F1 CPT
S0936	Ashley	Young	stokesmarcus@yahoo.com	3.99	F1 CPT
S0179	Larry	Peters	kristinaday@meyer.com	3.98	US Citizen
S0401	Monique	Sutton	jacob01@anthony.org	3.98	US Citizen
S0450	Sarah	Anderson	christopher19@holloway.com	3.98	US Citizen
S0608	Todd	Whitehead	philipshah@smith-anderson.com	3.98	Permanent Resident
S0657	Richard	Hensley	hodgeskristin@miller.com	3.98	F1 CPT
S0680	Amanda	Turner	joshuapatterson@hotmail.com	3.98	Permanent Resident
S0135	Amber	Cunningham	andersonkimberly@hayden.com	3.97	US Citizen
S0255	Michelle	Greer	jfrancis@yahoo.com	3.97	F1 OPT
S0258	Michelle	Chambers	meghan69@davis.com	3.97	Permanent Resident

Result Grid | Form Editor | Field Types | Query Stats

SQL Queries and Output

5. Retrieve the most popular job location

```sql
SELECT
    l.city,
    l.state,
    COUNT(jp.job_id) AS total_jobs
FROM
    job_postings_clean jp
JOIN
    locations l
    ON jp.zipcode = l.zipcode
GROUP BY
    l.city, l.state
ORDER BY
    total_jobs DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | W

city	state	total_jobs
Jacksonmouth	Montana	1

6. Jobs with most benefits offered

```sql
SELECT
    jc.category_name,
    COUNT(p.perk_id) AS total_benefits
FROM
    job_postings_clean jp
JOIN
    perks p
    ON jp.perks_id = p.perk_id
JOIN
    job_categories_clean jc
    ON jp.category_id = jc.category_id
GROUP BY
    jc.category_name
ORDER BY
    total_benefits DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

category_name	total_benefits
Project Management	113
UI/UX Design	110
Cloud Computing	99
System Administration	96
IT Support	93

7. Match Job Seekers to Jobs Based on Multiple Criteria with Weighted Scoring

```sql
WITH seeker_skills AS (
    SELECT
        js.seeker_id, js.first_name, js.last_name,
        js.zip_code AS seeker_location,
        js.work_auth_id,
        js.exp_level AS seeker_experience
    FROM job_seekers js
),
job_requirements AS (
    SELECT
        jp.job_id, jp.job_title,
        jp.zipcode AS job_location, jp.work_auth_id,
        jp.category_id AS job_experience,
        k.keyword AS required_skill
    FROM job_postings_clean jp
    JOIN keywords k ON jp.keyword_id = k.keyword_id
),
compatibility AS (
    SELECT
        ss.seeker_id, ss.first_name, ss.last_name, jr.job_id, jr.job_title,
        -- Skill Matching (50% Weight)
        0.5 AS skill_match,
        -- Location Match (20% Weight)
        CASE WHEN ss.seeker location = jr.job location THEN 1 ELSE 0 END * 0.2 AS location match,
```

```sql
        0.5 AS skill_match,
        -- Location Match (20% Weight)
        CASE WHEN ss.seeker_location = jr.job_location THEN 1 ELSE 0 END * 0.2 AS location_match,
        -- Experience Match (20% Weight)
        CASE
            WHEN ss.seeker_experience >= jr.job_experience THEN 1
            WHEN ABS(ss.seeker_experience - jr.job_experience) <= 1 THEN 0.5
            ELSE 0
        END * 0.2 AS experience_match,
        -- Work Authorization Match (10% Weight)
        CASE WHEN ss.work_auth_id = jr.work_auth_id THEN 1 ELSE 0 END * 0.1 AS auth_match
    FROM seeker_skills ss
    CROSS JOIN job_requirements jr
),
ranked_matches AS (
    SELECT
        seeker_id, first_name, last_name, job_id, job_title,
        (skill_match + location_match + experience_match + auth_match) AS compatibility_score
    FROM compatibility
)
SELECT *
FROM ranked_matches
WHERE compatibility_score >= 0.7 -- Only show matches with 70%+ compatibility
ORDER BY compatibility score DESC, seeker id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

seeker_id	first_name	last_name	job_id	job_title	compatibility_score
S0175	Stacy	Smith	995	Automotive Engineer	1.00
S0279	Anthony	Williams	26	Marine Engineer	0.90
S0727	Kyle	Parker	639	Marine Engineer	0.90
S0780	Andrew	Clark	460	Mobile App Developer	0.90
S0799	Gary	Brown	338	Front-End Developer	0.90

8. Top 3 candidates for a job based on their education & experience

```sql
WITH aggregated_experience AS (
    SELECT
        seeker_id,
        SUM(months_of_experience) AS total_experience
    FROM
        seeker_experience
    GROUP BY
        seeker_id
)
SELECT
    a.job_id, js.seeker_id, js.first_name, js.last_name, e.gpa, ae.total_experience
FROM
    applications_clean a
JOIN
    job_seekers js
    ON a.seeker_id = js.seeker_id
JOIN
    education e
    ON js.seeker_id = e.seeker_id
JOIN
    aggregated_experience ae
    ON js.seeker_id = ae.seeker_id
```

```sql
180         ON js.seeker_id = e.seeker_id
181     JOIN
182         aggregated_experience ae
183         ON js.seeker_id = ae.seeker_id
184     WHERE
185         a.job_id = '171' -- Replace with the desired job ID
186     ORDER BY
187         CAST(ae.total_experience AS UNSIGNED) DESC,
188         CAST(e.gpa AS DECIMAL(10, 2)) DESC
189     LIMIT
190         3;
191
192
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	total_benefits
Project Management	113
UI/UX Design	110
Cloud Computing	99
System Administration	96
IT Support	93

SQL Queries and Output

9. Pivot Table: Count Applications Per Status for Each Job category

```sql
SELECT
    jc.category_name,
    SUM(CASE WHEN a.application_status = 'Applied' THEN 1 ELSE 0 END) AS Applied,
    SUM(CASE WHEN a.application_status = 'Under Review' THEN 1 ELSE 0 END) AS Under_Review,
    SUM(CASE WHEN a.application_status = 'Rejected' THEN 1 ELSE 0 END) AS Rejected,
    SUM(CASE WHEN a.application_status = 'Accepted' THEN 1 ELSE 0 END) AS Accepted
FROM
    applications_clean a
JOIN
    job_postings_clean jp
    ON a.job_id = jp.job_id
JOIN
    job_categories_clean jc
    ON jp.category_id = jc.category_id
GROUP BY
    jc.category_name
ORDER BY
    jc.category_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	Applied	Under_Review	Rejected	Accepted
Cloud Computing	51	60	51	42
Cybersecurity	51	51	56	36
Data Science	45	42	40	54
Engineering	37	31	42	34
IT Support	44	50	51	50
Product Management	40	40	34	40
Project Management	61	61	64	62
Software Development	50	51	38	43
System Administration	64	61	62	61
UI/UX Design	62	71	67	50

10. Highest no of accepted applicants which was their field of study (Top3)

```sql
SELECT
    e.field_of_study,
    COUNT(a.application_id) AS total_accepted_applicants
FROM
    applications_clean a
JOIN
    education e
    ON a.seeker_id = e.seeker_id
WHERE
    a.application_status = 'Accepted'
GROUP BY
    e.field_of_study
ORDER BY
    total_accepted_applicants DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

field_of_study	total_accepted_applicants
IT	135
Business	131
Management	130

SQL Queries and Output

11. Identify the Conversion Rate from Application to Offer Per Job Category

```
37
38 •  SELECT
39        jc.category_name,
40        COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) * 100.0 / COUNT(*) AS conversion_rate
41    FROM
42        applications_clean a
43    JOIN
44        job_postings_clean jp ON a.job_id = jp.job_id
45    JOIN
46        job_categories_clean jc ON jp.category_id = jc.category_id
47    GROUP BY
48        jc.category_name
49    ORDER BY
50        conversion_rate DESC;
51
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	conversion_rate
Data Science	29.83425
Product Management	25.97403
IT Support	25.64103
Project Management	25.00000
System Administration	24.59677
Software Development	23.62637
Engineering	23.61111
Unknown	23.61111
Cloud Computing	20.58824
UI/UX Design	20.00000
Cybersecurity	18.55670

12. Determine the Top 10 Most Frequently Rejected Job Categories

```
253 •  SELECT
254        jc.category_name,
255        COUNT(CASE WHEN a.application_status = 'Rejected' THEN 1 END) AS rejection_count
256    FROM
257        applications_clean a
258    JOIN
259        job_postings_clean jp ON a.job_id = jp.job_id
260    JOIN
261        job_categories_clean jc ON jp.category_id = jc.category_id
262    GROUP BY
263        jc.category_name
264    ORDER BY
265        rejection_count DESC
266    LIMIT 10;
267
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

category_name	rejection_count
UI/UX Design	67
Project Management	64
System Administration	62
Cybersecurity	56
IT Support	51
Cloud Computing	51
Engineering	42
Unknown	42
Data Science	40
Software Development	38

SQL Queries and Output

13. Count per job category for each work authorization type : Pivot Table

```sql
SELECT
    jc.category_name,
    wa.auth_type AS work_authorization,
    COUNT(a.application_id) AS application_count
FROM
    applications_clean a
JOIN
    job_postings_clean jp
    ON a.job_id = jp.job_id
JOIN
    job_categories_clean jc
    ON jp.category_id = jc.category_id
JOIN
    work_authorization wa
    ON jp.work_auth_id = wa.work_auth_id
GROUP BY
    jc.category_name, wa.auth_type
ORDER BY
    jc.category_name, application_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

category_name	work_authorization	application_count
Cloud Computing	F1 CPT	57
Cloud Computing	Permanent Resident	44
Cloud Computing	H1B	38
Cloud Computing	F1 OPT	36
Cloud Computing	US Citizen	29
Cybersecurity	F1 CPT	47
Cybersecurity	US Citizen	42
Cybersecurity	Permanent Resident	36
Cybersecurity	H1B	35
Cybersecurity	F1 OPT	34

14. Find Keywords Leading to the Highest Conversion Rates (Application to Offer)

```sql
SELECT
    k.keyword,
    COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) * 100.0 / COUNT(*) AS conversion_rate
FROM
    applications_clean a
JOIN
    job_postings_clean jp ON a.job_id = jp.job_id
JOIN
    keywords k ON jp.keyword_id = k.keyword_id
GROUP BY
    k.keyword
ORDER BY
    conversion_rate DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

keyword	conversion_rate
ITIL	36.84211
UI/UX Design	30.76923
Cloud Computing	29.67033
Statistics	29.11392
Mongo Db	28.84615
Cybersecurity	28.57143
DevOps	28.57143
DNS Configuration	28.15534
SQL	23.50877
Predictive Modeling	23.25581

SQL Queries and Output

15. Calculate the Ratio of Job Applications to Job Openings Per Category

```sql
SELECT
    jc.category_name,
    COUNT(a.application_id) AS total_applications,
    COUNT(DISTINCT jp.job_id) AS job_openings,
    ROUND(COUNT(a.application_id) / COUNT(DISTINCT jp.job_id), 2) AS application_to_opening_ratio
FROM
    applications_clean a
JOIN
    job_postings_clean jp ON a.job_id = jp.job_id
JOIN
    job_categories_clean jc ON jp.category_id = jc.category_id
GROUP BY
    jc.category_name
ORDER BY
    application_to_opening_ratio DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	total_applications	job_openings	application_to_opening_ratio
System Administration	248	57	4.35
Cybersecurity	194	46	4.22
Data Science	181	43	4.21
Cloud Computing	204	49	4.16
Software Development	182	44	4.14
Project Management	248	61	4.07
Product Management	154	38	4.05
UI/UX Design	250	62	4.03
IT Support	195	51	3.82
Engineering	144	38	3.79

16. Query to Find the Company with the Most Job Postings

```sql
SELECT
    se.company_name,
    COUNT(se.company_name) AS total_job_postings
FROM
    seeker_experience se
GROUP BY
    se.company_name
ORDER BY
    total_job_postings DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Con

company_name	total_job_postings
Smith PLC	5

SQL Queries and Output

17. Predict Job Success Based on Work Authorization

```sql
SELECT
    jc.category_name,
    wa.auth_type,
    COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) * 100.0 / COUNT(*) AS acceptance_rate
FROM
    applications_clean a
JOIN
    job_postings_clean jp ON a.job_id = jp.job_id
JOIN
    work_authorization wa ON jp.work_auth_id = wa.work_auth_id
JOIN
    job_categories_clean jc ON jp.category_id = jc.category_id
GROUP BY
    jc.category_name, wa.auth_type
ORDER BY
    acceptance_rate DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	auth_type	acceptance_rate
System Administration	H1B	36.92308
Project Management	H1B	35.71429
Data Science	F1 OPT	34.48276
Product Management	Permanent Resident	34.21053
IT Support	H1B	33.33333
Software Development	US Citizen	32.00000
Data Science	F1 CPT	31.81818
Data Science	US Citizen	31.25000
Cloud Computing	US Citizen	31.03448
Data Science	Permanent Resident	29.54545
IT Support	US Citizen	29.09091
UI/UX Design	Permanent Resident	28.84615
Cybersecurity	H1B	28.57143
Engineering	F1 OPT	28.57143

18. Monthly Trends and Growth in Job Applications

```sql
WITH monthly_applications AS (
    SELECT
        DATE_FORMAT(a.application_date, '%Y-%m') AS application_month,
        COUNT(a.application_id) AS total_applications
    FROM
        applications_clean a
    GROUP BY
        DATE_FORMAT(a.application_date, '%Y-%m')
),
monthly_growth AS (
    SELECT
        ma.application_month, ma.total_applications, ma.total_applications - LAG(ma.total_applications) OVER (ORDER BY ma.applic
    FROM
        monthly_applications ma
)
SELECT
    application_month, total_applications, monthly_growth
FROM
    monthly_growth
ORDER BY
    monthly_growth DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

application_month	total_applications	monthly_growth
2024-01	199	52
2024-11	183	25
2024-08	181	23
2024-03	157	9
2024-06	167	5

Result Grid | Form Editor

19. Compare Application Success Rates by Gender

```
435 •   SELECT
436         js.gender,
437         COUNT(a.application_id) AS total_applications,
438         COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) AS accepted_applications,
439         ROUND(COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) * 100.0 / COUNT(a.application_id), 2) AS acceptance_rate
440     FROM
441         job_seekers js
442     JOIN
443         applications_clean a
444         ON js.seeker_id = a.seeker_id
445     GROUP BY
446         js.gender
447     ORDER BY
448         acceptance_rate DESC;
449
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender	total_applications	accepted_applications	acceptance_rate
Non-binary	668	175	26.20
Female	662	153	23.11
Male	670	144	21.49

Result Grid

Form Editor

20. Job Categories with the Lowest Application-to-Acceptance Ratio

```
412 •  SELECT
413        jc.category_name,
414        COUNT(a.application_id) AS total_applications,
415        COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) AS accepted_applications,
416        ROUND(COUNT(CASE WHEN a.application_status = 'Accepted' THEN 1 END) * 100.0 / COUNT(a.application_id), 2) AS acceptance_rati
417    FROM
418        applications_clean a
419    JOIN
420        job_postings_clean jp
421        ON a.job_id = jp.job_id
422    JOIN
423        job_categories_clean jc
424        ON jp.category_id = jc.category_id
425    GROUP BY
426        jc.category_name
427    ORDER BY
428        acceptance_ratio ASC, total_applications DESC;
429
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category_name	total_applications	accepted_applications	acceptance_ratio
Cybersecurity	194	36	18.56
UI/UX Design	250	50	20.00
Cloud Computing	204	42	20.59
Engineering	144	34	23.61
Unknown	144	34	23.61
Software Development	182	43	23.63
System Administration	248	61	24.60
Project Management	248	62	25.00
IT Support	195	50	25.64
Product Management	154	40	25.97
Data Science	181	54	29.83

Result Grid | Form Editor | Field Types

NoSQL Queries and Output

1. List all recruiters

```
[41]: print("\n--- List All Admin Users ---")
users = admin_users_collection.find({}, {"name": 1, "email": 1, "_id": 0})
for user in users:
    print(user)
```

```
--- List All Admin Users ---
{'name': 'Richard Gibson', 'email': 'jamiewaters@york-payne.com'}
{'name': 'Mr. Austin Payne', 'email': 'bcastro@yahoo.com'}
{'name': 'Cynthia Bridges', 'email': 'ralph75@kirk-solomon.com'}
{'name': 'Thomas Nichols', 'email': 'taylor83@gmail.com'}
{'name': 'Alejandra Jones', 'email': 'xprice@pearson.biz'}
{'name': 'Heidi Mitchell', 'email': 'westtammy@gmail.com'}
{'name': 'Tiffany Davis', 'email': 'joshua93@smith-brown.net'}
{'name': 'Bradley Turner', 'email': 'christopher22@rodriguez-herman.com'}
{'name': 'Crystal Jimenez', 'email': 'ddalton@gmail.com'}
```

2. Find recruiter by name

```
[35]: print("\n--- Find Recruiter by Name ---")
recruiter_name = "Shane Gross"  # Replace with actual name
recruiter = admin_recruiters_collection.find_one({"name": recruiter_name})
if recruiter:
    print(recruiter)
else:
    print(f"No recruiter found with name {recruiter_name}")
```

```
--- Find Recruiter by Name ---
{'_id': ObjectId('674fcf2806a469a4c6066eb0'), 'recruiter_id': 1, 'name': 'Shane Gross', 'email': 'edwardmay@hotmail.com', 'phone': '+1 (809) 458 7696',
'assigned_job_postings': 214}
```

3. Find inactive admin users

```
[42]: # 4. Find inactive admin users
print("\n--- Find Inactive Admin Users ---")
inactive_users = admin_users_collection.find({"status": "Inactive"})
for user in inactive_users:
    print(user)
```

```
--- Find Inactive Admin Users ---
{'_id': ObjectId('674fcf3c06a469a4c60670a5'), 'user_id': 1, 'name': 'Richard Gibson', 'email': 'jamiewaters@york-payne.com', 'status': 'Inactive', 're
gistration_date': '7/27/23', 'last_login': '10/9/24'}
{'_id': ObjectId('674fcf3c06a469a4c60670a9'), 'user_id': 5, 'name': 'Alejandra Jones', 'email': 'xprice@pearson.biz', 'status': 'Inactive', 'registrat
ion_date': '9/26/23', 'last_login': '9/4/24'}
{'_id': ObjectId('674fcf3c06a469a4c60670ac'), 'user_id': 8, 'name': 'Bradley Turner', 'email': 'christopher22@rodriguez-herman.com', 'status': 'Inacti
ve', 'registration_date': '10/24/23', 'last_login': '5/8/24'}
{'_id': ObjectId('674fcf3c06a469a4c60670ae'), 'user_id': 10, 'name': 'John Fowler', 'email': 'fwood@gibson.com', 'status': 'Inactive', 'registration_d
ate': '5/8/23', 'last_login': '2/12/24'}
```

4. Update recruiter phone number

```
[40]: # 7. Update recruiter phone number
print("\n--- Update Recruiter Phone Number ---")
updated_recruiter = admin_recruiters_collection.update_one(
    {"name": "Veronica Vincent"},  # Replace with an actual recruiter name
    {"$set": {"phone": "+1 (123) 456 7890"}}  # Replace with new phone number
)
if updated_recruiter.modified_count > 0:
    print("Phone number updated successfully.")
else:
    print("No update made.")
```

```
--- Update Recruiter Phone Number ---
Phone number updated successfully.
```

NoSQL Queries and Output

5. List all admin users

```
[41]: print("\n--- List All Admin Users ---")
users = admin_users_collection.find({}, {"name": 1, "email": 1, "_id": 0})
for user in users:
    print(user)
```

```
--- List All Admin Users ---
{'name': 'Richard Gibson', 'email': 'jamiewaters@york-payne.com'}
{'name': 'Mr. Austin Payne', 'email': 'bcastro@yahoo.com'}
{'name': 'Cynthia Bridges', 'email': 'ralph75@kirk-solomon.com'}
{'name': 'Thomas Nichols', 'email': 'taylor83@gmail.com'}
{'name': 'Alejandra Jones', 'email': 'xprice@pearson.biz'}
{'name': 'Heidi Mitchell', 'email': 'westtammy@gmail.com'}
{'name': 'Tiffany Davis', 'email': 'joshua93@smith-brown.net'}
{'name': 'Bradley Turner', 'email': 'christopher22@rodriguez-herman.com'}
{'name': 'Crystal Jimenez', 'email': 'ddalton@gmail.com'}
{'name': 'John Fowler', 'email': 'fwood@gibson.com'}
{'name': 'Amy Nguyen', 'email': 'torreslarry@yahoo.com'}
{'name': 'Michael Johnson', 'email': 'joshuaherring@gray.com'}
{'name': 'Carol Villarreal', 'email': 'qortiz@yahoo.com'}
{'name': 'Corey Garza', 'email': 'richard66@hotmail.com'}
{'name': 'Mrs. Laurie House MD', 'email': 'lisa21@wade.com'}
{'name': 'Jose Mills', 'email': 'ashley30@yahoo.com'}
```

Advantages

1. Streamlined Job Search:

Provides job seekers with a centralized platform to browse, filter, and apply for jobs effortlessly.

2. Centralized Data Management:

Allows job seekers and hiring managers to access and manage all necessary information in one place.

3. Real-Time Updates:

Offers real-time notifications on application status changes, ensuring transparency throughout the process.

4. Improved Efficiency for Hiring Managers:

Simplifies recruitment workflows by enabling streamlined posting, tracking, and managing of job applications.

5. Personalized User Experience:

Features tailored job recommendations based on user preferences, skills, and location.

6. Enhanced Collaboration:

Bridges the gap between job seekers and employers, fostering better communication and engagement.

7. Scalability:

Designed to handle growing user bases and expanding functionalities as the platform evolves.

8. Secure and Reliable:

Implements robust security measures to protect user data and ensure system reliability.

CONCLUSION

JobConnect is a cutting-edge platform designed to transform the job application and recruitment experience for both job seekers and hiring managers. By centralizing all job-related processes into a single, user-friendly interface, it eliminates the common challenges of managing applications across multiple platforms. Job seekers benefit from features like tailored job recommendations, real-time tracking, and easy profile management, enabling them to focus on their career growth with confidence.

At the same time, hiring managers are equipped with robust tools to streamline recruitment workflows, post jobs efficiently, and manage candidate information in a structured manner. The inclusion of admin controls ensures data integrity and smooth operations throughout the system.

In an increasingly competitive job market, JobConnect stands out as a reliable and efficient solution that fosters collaboration, saves time, and ensures a seamless connection between talent and opportunity. Its focus on user experience, operational efficiency, and transparency makes it an indispensable tool for modern-day recruitment and career management

Resources

MongoDB:
A NoSQL database used for scalable and flexible data storage.

Microsoft Excel:
A versatile tool for organizing, analyzing, and visualizing data.

SQL Workbench:
A graphical tool for managing and querying relational databases.

Generatedata.com:
A tool for creating realistic dummy data for testing.

XAMPP:
A local development environment for testing web applications.

Draw.io:
A diagramming tool for creating flowcharts, ER diagrams, and system designs.