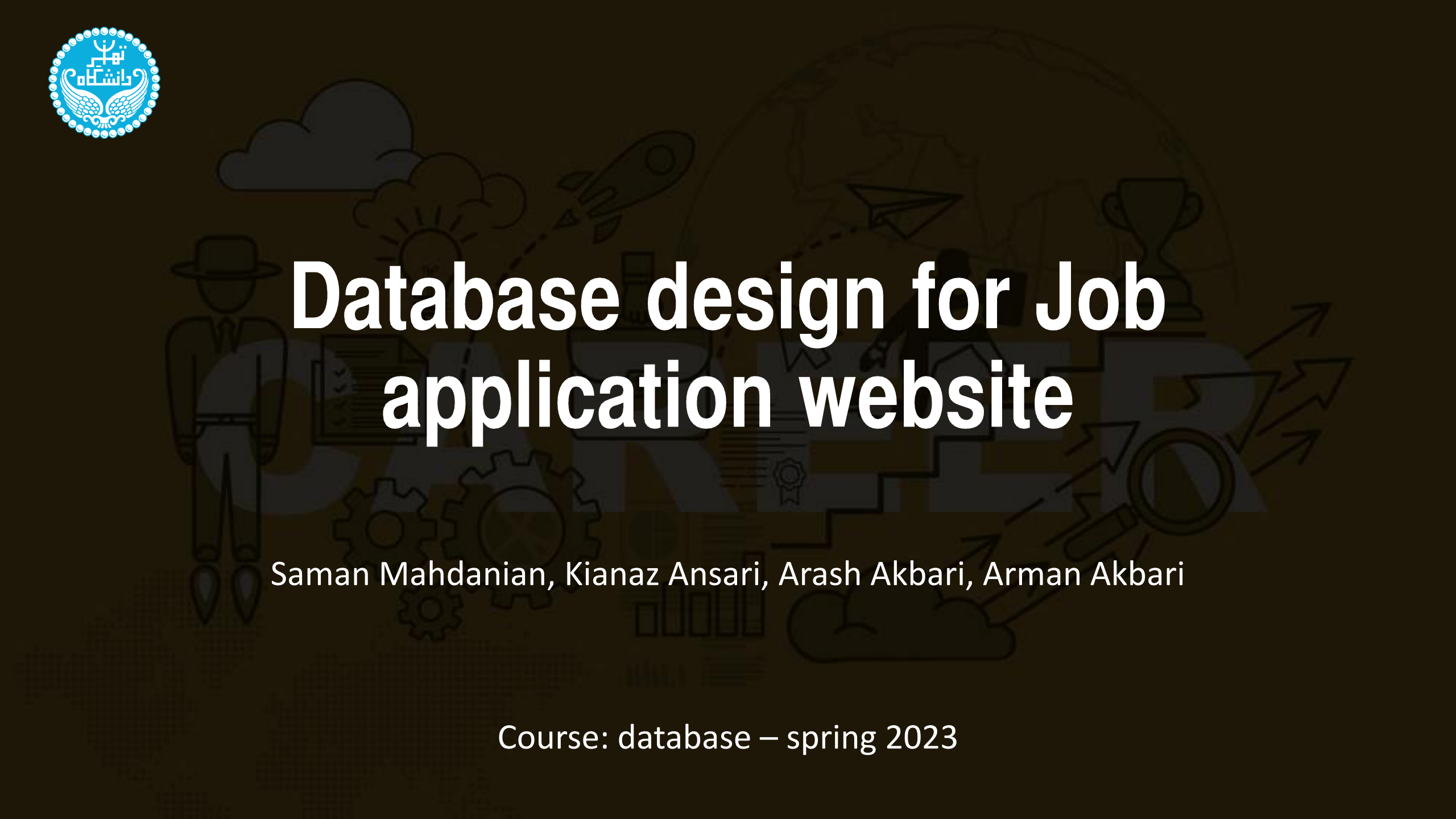
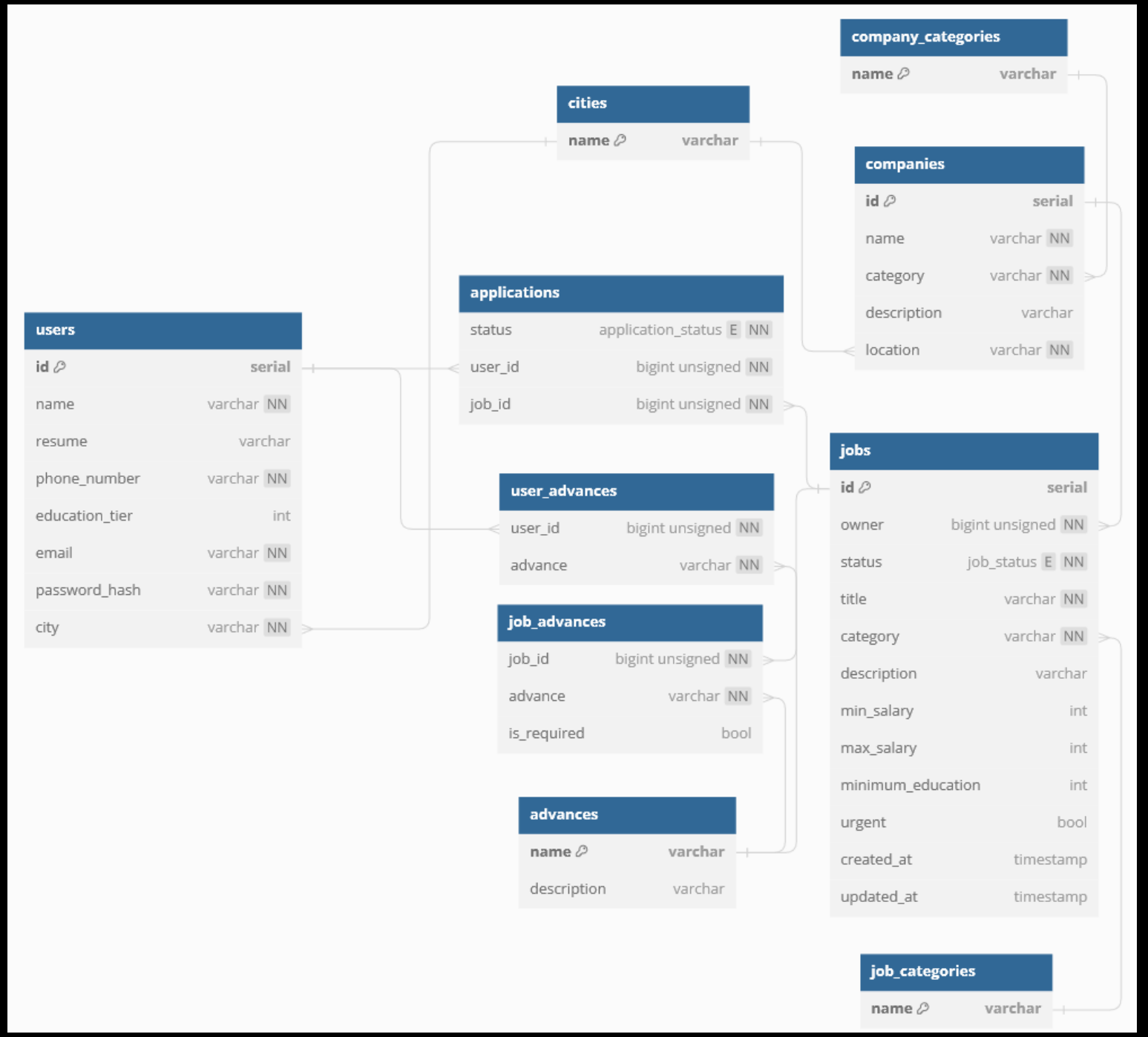


Database design for Job application website

Saman Mahdanian, Kianaz Ansari, Arash Akbari, Arman Akbari

Course: database – spring 2023

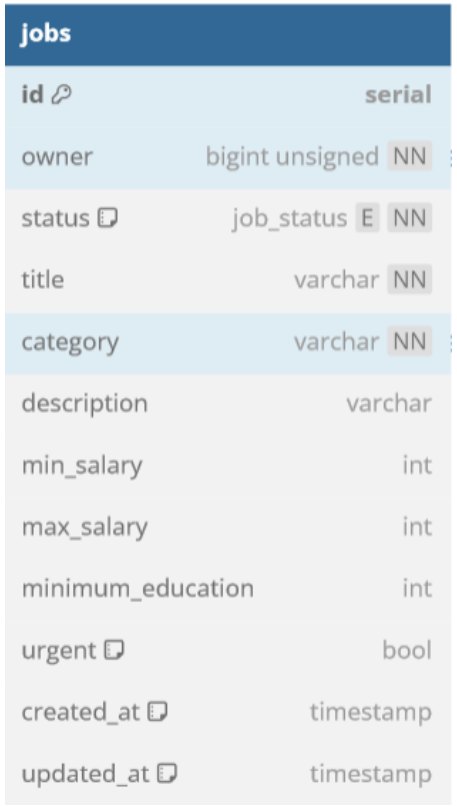
Overview of the tables



Tables

- Users
- Applications
- Cities
- Use_advances
- Job_advances
- Advances
- Company_categories
- Companies
- Jobs
- Jobs_categories

```sql
CREATE TABLE `jobs` (
  `id` serial PRIMARY KEY,
  `owner` bigint unsigned NOT NULL,
  `status` ENUM ('draft', 'active', 'expired') NOT NULL,
  `title` varchar(255) NOT NULL,
  `category` varchar(255) NOT NULL,
  `description` varchar(255),
  `min_salary` int,
  `max_salary` int,
  `minimum_education` int,
  `urgent` bool DEFAULT false,
  `created_at` timestamp DEFAULT (now()),
  `updated_at` timestamp DEFAULT (now())
);
```

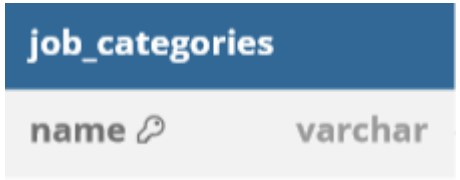


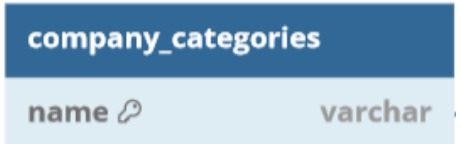

```sql
CREATE TABLE `job_advances` (
  `job_id` bigint unsigned NOT NULL,
  `advance` varchar(255) NOT NULL,
  `is_required` bool DEFAULT false
);
```

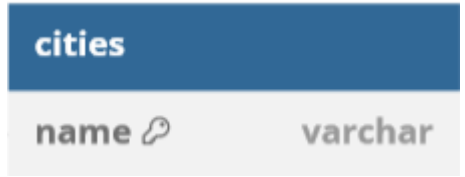
```sql
CREATE TABLE `companies` (
  `id` serial PRIMARY KEY,
  `name` varchar(255) NOT NULL,
  `category` varchar(255) NOT NULL,
  `description` varchar(255),
  `location` varchar(255) NOT NULL
);
```

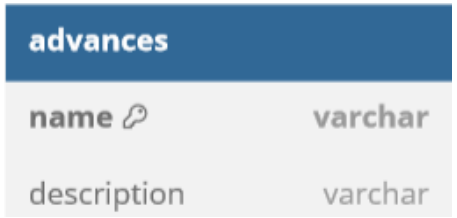
```sql
CREATE TABLE `job_categories` (
  `name` varchar(255) PRIMARY KEY
);

CREATE TABLE `company_categories` (
  `name` varchar(255) PRIMARY KEY
);

CREATE TABLE `cities` (
  `name` varchar(255) PRIMARY KEY
);
```

```sql
CREATE TABLE `advances` (
  `name` varchar(255) PRIMARY KEY,
  `description` varchar(255)
);
```









```sql
CREATE TABLE `users` (
  `id` serial PRIMARY KEY,
  `name` varchar(255) NOT NULL,
  `resume` varchar(255),
  `phone_number` varchar(255) NOT NULL,
  `education_tier` int,
  `email` varchar(255) NOT NULL,
  `password_hash` varchar(255) NOT NULL,
  `city` varchar(255) NOT NULL
);

CREATE TABLE `user_advances` (
  `user_id` bigint unsigned NOT NULL,
  `advance` varchar(255) NOT NULL
);
```


```sql
CREATE TABLE `applications` (
  `status` ENUM ('created', 'rejected', 'waiting_for_interview',
                 'interview_rejected', 'interview_accepted',
                 'offer_rejected', 'done') NOT NULL,
  `user_id` bigint unsigned NOT NULL,
  `job_id` bigint unsigned NOT NULL
);
```

Queries

- Q1: getting skills(advances) of an individual user
- Q2: getting the available jobs for which a specific user can apply
- Q3: getting the jobs that meet the skills of a specific user
- Q4: all matching jobs with additional advances
- Q5: all matching jobs in same city
- Q6: all user that have the appropriate skills for a specific job
- Q7: all users that has applied for a specific job

* We were suppose to write eight queries, but since our database wasn’t so huge we only figured out 7 meaningful queries. The queries are quit long and high quality. We could have add some basic and unmeaningfull queries but we rather stay with less.

```sql
-- q1

set @user_id = 6
SELECT
    advance
FROM
    user_advances
WHERE
    user_advances.user_id = @user_id
```

```sql
-- q2
SELECT
    j.id
FROM
    users
LEFT JOIN cities c ON
    users.city = c.name
INNER JOIN companies c2 ON
    c.name = c2.location
INNER JOIN jobs j ON
    c2.id = j.owner
```

```sql
-- q3
SELECT id FROM (
    SELECT id, COUNT(ja.advance) - COUNT(ua.advance) as missing_advances FROM
    (
        SELECT advance FROM
            (SELECT id FROM users WHERE id = 6) selected_user
            LEFT JOIN user_advances aua
            on selected_user.id = aua.user_id
    ) ua
    RIGHT JOIN (
        SELECT id, owner, advance
        FROM (
            jobs j
            LEFT JOIN (
                SELECT * FROM job_advances WHERE is_required=true
            ) ja
            ON j.id = ja.job_id
        )
    ) ja

    ON ja.advance = ua.advance
    GROUP BY id HAVING missing_advances = 0
) result;
```

```sql
-- q4: all matching jobs with additional advances
SELECT id, additional_advances FROM (
    SELECT
            id,
            COUNT(ja.advance AND is_required) - COUNT(ua.advance AND is_required) as missing_advances,
            COUNT(ua.advance AND NOT is_required) as additional_advances
    FROM
    (
        SELECT advance FROM
            (SELECT id FROM users WHERE id = 6) selected_user
            LEFT JOIN user_advances aua
            on selected_user.id = aua.user_id
    ) ua
    RIGHT JOIN (
        SELECT id, owner, advance, is_required
        FROM (
            jobs j
            LEFT JOIN job_advances ja
            ON j.id = ja.job_id
        )
    ) ja

    ON ja.advance = ua.advance
    GROUP BY id HAVING missing_advances = 0
) result
```

```sql
-- q5: all matching jobs in same city
SET @user_id=6;
SET @user_city=(SELECT city FROM users WHERE id=@user_id);
SELECT id, additional_advances FROM (
    SELECT
           id,
           COUNT(ja.advance AND is_required) - COUNT(ua.advance AND is_required) as missing_advances,
           COUNT(ua.advance AND NOT is_required) as additional_advances
    FROM
    (
        SELECT advance, city FROM
            (SELECT id, city FROM users WHERE id = @user_id) selected_user
            LEFT JOIN user_advances aua
            on selected_user.id = aua.user_id
    ) ua
    RIGHT JOIN (
        SELECT _ja.id as id, location, advance, is_required FROM
        (
           SELECT id, owner, advance, is_required
           FROM (
                 jobs j
                    LEFT JOIN job_advances ja
                              ON j.id = ja.job_id
                )
        ) _ja
        LEFT JOIN companies
        ON companies.id = _ja.owner
        WHERE location = @user_city
    ) ja
    ON ja.advance = ua.advance
    GROUP BY id HAVING missing_advances = 0
) result;
```

```sql
-- q6: all user that have the appropriate skills for a specific job

SET @company_id=1
SELECT
    title, name, resume, email, city
FROM
  applications
INNER JOIN (
    SELECT
        jobs.id as id, jobs.title as title
    FROM
        jobs
    LEFT JOIN companies
        ON jobs.owner = companies.id
        WHERE companies.id = @company_id
    )selecter_jobs
  ON applications.job_id = selecter_jobs.id
  LEFT JOIN users
  ON users.id = applications.user_id
  WHERE
    applications.status = "created"
```

```sql
-- q7 all users that has applied for a specific job
SET @job_id=6
SELECT
    name, resume, phone_number, email, city
FROM
    applications
LEFT JOIN
    users
ON users.id = applications.user_id
WHERE
    applications.job_id = @job_id and applications.status = "created"
```

Data

- The data has been generated by Gatgpt. The queries for adding data is stored in inti_data.sql. Here are some examples:

```sql
INSERT INTO job_categories (name) VALUES
    ('Technology'),
    ('Healthcare'),
    ('Finance'),
    ('Education'),
    ('Sales'),
    ('Marketing'),
    ('Engineering'),
    ('Hospitality'),
    ('Art'),
    ('Manufacturing');
```

```sql
INSERT INTO job_advances (job_id, advance, is_required) VALUES
  (1, 'Certification', true),
  (1, 'Experience', true),
  (2, 'Certification', true),
  (2, 'Experience', true),
  (3, 'Experience', true),
  (4, 'Certification', false),
  (5, 'Experience', false),
  (6, 'Experience', true),
  (6, 'Technical Skills', true),
  (7, 'Certification', true),
  (7, 'Experience', true),
  (8, 'Certification', false),
  (8, 'Analytical Skills', false),
  (9, 'Certification', false),
  (10, 'Experience', true),
  (10, 'Communication Skills', true);
```