1. Provide a SQL script that initializes the database for the Job Board scenario “CareerHub”.

(SQL SCRIPTS)...

```
-- Schema careerhub
CREATE SCHEMA IF NOT EXISTS `careerhub` DEFAULT CHARACTER SET utf8 ;
USE `careerhub` ;

 -- Table `careerhub`.`company`
CREATE TABLE IF NOT EXISTS `careerhub`.`company` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `company_name` VARCHAR(255) NULL,
  `location` VARCHAR(255) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

-- Table `careerhub`.`applicants`
CREATE TABLE IF NOT EXISTS `careerhub`.`applicants` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(255) NULL,
  `last_name` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `phone` VARCHAR(255) NULL,
  `resume` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

 -- Table `careerhub`.`jobs`
CREATE TABLE IF NOT EXISTS `careerhub`.`jobs` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `job_title` VARCHAR(255) NULL,
  `job_description` TEXT NULL,
  `job_location` VARCHAR(255) NULL,
```

```
  `salary` DOUBLE NULL,
  `job_type` VARCHAR(255) NULL,
  `posted_date` DATE NULL,
  `company_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_jobs_company_idx` (`company_id` ASC),
  CONSTRAINT `fk_jobs_company`
    FOREIGN KEY (`company_id`)
    REFERENCES `careerhub`.`company` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


 -- Table `careerhub`.`applications`
CREATE TABLE IF NOT EXISTS `careerhub`.`applications` (
  `jobs_id` INT NOT NULL,
  `applicants_id` INT NOT NULL,
  `id` INT NOT NULL AUTO_INCREMENT,
  `application_date` VARCHAR(255) NULL,
  `cover_letter` TEXT NULL,
  INDEX `fk_jobs_has_applicants_applicants1_idx` (`applicants_id` ASC),
  INDEX `fk_jobs_has_applicants_jobs1_idx` (`jobs_id` ASC),
  PRIMARY KEY (`id`),
  CONSTRAINT `fk_jobs_has_applicants_jobs1`
    FOREIGN KEY (`jobs_id`)
    REFERENCES `careerhub`.`jobs` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_jobs_has_applicants_applicants1`
    FOREIGN KEY (`applicants_id`)
    REFERENCES `careerhub`.`applicants` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

INSERTIONS

```
INSERT INTO company(company_name, location) VALUES
('Hexaware', 'Chennai'), ('Google', 'Pune'), ('Microsoft', 'Mumbai'),
('Amazon', 'Banglore'), ('Adobe', 'Mumbai'), ('Uber', 'Mumbai'),
('TCS', 'Banglore'), ('Wipro', 'Banglore'), ('JP Morgan', 'Hyderabad'),
('IBM', 'Banglore');
```

==========

```
INSERT INTO applicants(first_name, last_name, email, phone, resume) VALUES
('Harry', 'Potter', 'harry@hogwards.com', '90909090', 'harry_resume.pdf'),
('Ronald', 'Weasley', 'ronald@hogwards.com', '90909080', 'ronald_resume.pdf'),
('Hermione', 'Granger', 'hermione@hogwards.com', '90909070', 'hermione_resume.pdf'),
('Draco', 'Malfoy', 'draco@hogwards.com', '90909060', 'draco_resume.pdf'),
('Neville', 'Longbottom', 'neville@hogwards.com', '90909050', 'neville_resume.pdf'),
('Ginny', 'Weasley', 'ginny@hogwards.com', '90909040', 'ginny_resume.pdf');
```

==========

```
INSERT INTO jobs(job_title, job_description, job_location, salary, job_type, posted_date, company_id)
VALUES
('Software Engineer', 'Develop software applications', 'Banglore', 900000, 'Full-time', '2024-02-01', 3),
('Data Analyst', 'Analyze huge data', 'Mumbai', 700000, 'Part-time', '2024-02-09', 4),
('Frontend Developer', 'Design and develop UI/UX', 'Pune', 800000, 'Full-time', '2024-01-09', 1),
('DevOps Engineer', 'Develop CI/CD solutions', 'Hyderabad', 1500000, 'Full-time', '2024-02-19', 2),
```

('ML Engineer', 'Develop ML applications', 'Chennai', 1000000, 'Full-time', '2023-12-12', 9),

('Business Analyst', 'Plans business solutions', 'Banglore', 800000, 'Full-time', '2023-11-29', 10),

('Backend Developer', 'Develop backend queries', 'Pune', 950000, 'Full-time', '2024-03-01', 5),

('Blockchain Engineer', 'Develop CI/CD solutions', 'Banglore', 1300000, 'Full-time', '2024-02-28', 6);

INSERT INTO jobs(job_title, job_description, job_location, salary, job_type, posted_date, company_id) VALUES

('Software Developer', 'Develop Software Products', 'Chennai', 0, 'Internship', '2024-01-02', 7);

==========

INSERT INTO applications (jobs_id, applicants_id, application_date, cover_letter) VALUES

(4, 3, '2024-02-01', 'cover_letter.pdf'),

(1, 1, '2024-03-21', 'my_cover_letter.pdf'),

(5, 4, '2024-01-17', 'cover_letter1.pdf'),

(8, 5, '2024-12-18', 'company_cover_letter.pdf'),

(6, 2, '2024-01-27', 'cover_letter.pdf'),

(4, 6, '2024-02-10', '1cover_letter.pdf');

==========

5. Write an SQL query to count the number of applications received for each job listing in the "Jobs" table. Display the job title and the corresponding application count. Ensure that it lists all jobs, even if they have no applications.

=> select j.job_title, COUNT(ap.applicants_id) as num_of_applications from applications ap

JOIN applicants a ON ap.applicants_id=a.id

RIGHT JOIN jobs j ON j.id=ap.jobs_id group by j.job_title;

6. Develop an SQL query that retrieves job listings from the "Jobs" table within a specified salary range. Allow parameters for the minimum and maximum salary values. Display the job title, company name, location, and salary for each matching job.

=> select c.company_name, j.job_title, j.job_location, j.salary from jobs j

JOIN company c ON j.company_id=c.id WHERE salary between 800000 AND 1200000;

7. Write an SQL query that retrieves the job application history for a specific applicant. Allow a parameter for the ApplicantID, and return a result set with the job titles, company names, and application dates for all the jobs the applicant has applied to.

=> select c.company_name, j.job_title, ap.application_date from applications ap

JOIN applicants a ON a.id=ap.applicants_id

JOIN jobs j ON ap.jobs_id=j.id

JOIN company c ON j.company_id=c.id

where a.first_name='Harry' AND a.last_name='Potter';

8. Create an SQL query that calculates and displays the average salary offered by all companies for job listings in the "Jobs" table. Ensure that the query filters out jobs with a salary of zero.

=> select c.company_name, AVG(j.salary) from jobs j

JOIN company c ON c.id=j.company_id group by c.company_name having AVG(j.salary>0);

9. Write an SQL query to identify the company that has posted the most job listings. Display the company name along with the count of job listings they have posted. Handle ties if multiple companies have the same maximum count.

=> select c.company_name, COUNT(c.company_name) from jobs j

JOIN company c ON j.company_id=c.id group by c.company_name order by COUNT(c.company_name) DESC limit 1;

10. Find the applicants who have applied for positions in companies located in 'CityX' and have at least 3 years of experience.

=> select a.first_name, a.last_name, c.company_name from applications ap

JOIN applicants a ON a.id=ap.applicants_id

JOIN jobs j ON j.id=ap.jobs_id

JOIN company c ON c.id=j.company_id

where c.location='Mumbai';

11. Retrieve a list of distinct job titles with salaries between $60,000 and $80,000.

(values according to my database)...

=> select distinct job_title from jobs where salary between 900000 and 1300000;

12. Find the jobs that have not received any applications.

=> select * from jobs where id not in (select jobs_id from applications);

13. Retrieve a list of job applicants along with the companies they have applied to and the positions they have applied for.

=> select a.first_name, a.last_name, c.company_name, j.job_title from applications ap

JOIN applicants a ON ap.applicants_id=a.id

JOIN jobs j ON j.id=ap.jobs_id

JOIN company c ON c.id=j.company_id;

14. Retrieve a list of companies along with the count of jobs they have posted, even if they have not received any applications.

=> select c.company_name, COUNT(j.company_id) from jobs j

RIGHT JOIN company c ON j.company_id=c.id group by c.company_name;

15. List all applicants along with the companies and positions they have applied for, including those

who have not applied.

=> select a.first_name, a.last_name, c.company_name, j.job_title from applications ap

RIGHT JOIN applicants a ON ap.applicants_id=a.id

JOIN jobs j ON j.id=ap.jobs_id

JOIN company c ON c.id=j.company_id;

16. Find companies that have posted jobs with a salary higher than the average salary of all jobs.

=> select c.company_name from jobs j

JOIN company c ON j.company_id=c.id

where j.salary>(select avg(salary) from jobs);

17. Display a list of applicants with their names and a concatenated string of their city and state. (values according to my database)...

=> select CONCAT(first_name,' ',last_name) as applicant_name, email, phone from applicants;

18. Retrieve a list of jobs with titles containing either 'Developer' or 'Engineer'.

=> select * from jobs where job_title like '%developer%' OR job_title like '%engineer%';

19. Retrieve a list of applicants and the jobs they have applied for, including those who have not applied and jobs without applicants.

=> select a.first_name, a.last_name, j.job_title from applications ap

RIGHT JOIN applicants a ON ap.applicants_id=a.id

RIGHT JOIN jobs j ON ap.jobs_id=j.id;

20. List all combinations of applicants and companies where the company is in a specific city and the

applicant has more than 2 years of experience. For example: city=Chennai

=> select a.first_name, a.last_name, c.company_name from applications ap

RIGHT JOIN applicants a ON ap.applicants_id=a.id

RIGHT JOIN jobs j ON ap.jobs_id=j.id

RIGHT JOIN company c ON c.id=j.company_id

WHERE c.location='mumbai';