Coding Challenges: CareerHub, The Job Board

1. **Provide a SQL script that initializes the database for the Job Board scenario "CareerHub".**

```
CREATE DATABASE CareerHub;
use CareerHub;
show databases;
```

(The CareerHub database is created and displayed)

Database
assignment
careerhub
case_studies
information_schema
mysql
performance_schema
sys
testdb

2. **Create tables for Companies, Jobs, Applicants and Applications.**

```
CREATE TABLE Companies (
    CompanyID INT AUTO_INCREMENT PRIMARY KEY,
    CompanyName VARCHAR(255) NOT NULL,
    Location VARCHAR(255) NOT NULL
);
```

For efficiently answering the queries I have added 2 more columns:

```
ALTER TABLE Companies
ADD COLUMN City VARCHAR(100) NOT NULL,
ADD COLUMN State VARCHAR(100) NOT NULL;
desc companies;
```

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

Field	Type	Null	Key	Default	Extra
CompanyID	int	NO	PRI	NULL	auto_increment
CompanyName	varchar(255)	NO		NULL	
Location	varchar(255)	NO		NULL	
City	varchar(100)	NO		NULL	
State	varchar(100)	NO		NULL	

```
CREATE TABLE Jobs (
    JobID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    CompanyID INT NOT NULL,
    JobTitle VARCHAR(255) NOT NULL,
    JobDescription TEXT NOT NULL,
    JobLocation VARCHAR(255) NOT NULL,
    Salary DECIMAL(10,2) CHECK (Salary >= 0),
    JobType ENUM('Full-time', 'Part-time', 'Contract') NOT NULL,
    PostedDate DATETIME NOT NULL,
    CONSTRAINT fk_jobs_company FOREIGN KEY (CompanyID)
REFERENCES Companies(CompanyID) ON DELETE CASCADE
);
DESC Jobs;
```

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

Field	Type	Null	Key	Default	Extra
JobID	int	NO	PRI	NULL	auto_increment
CompanyID	int	NO	MUL	NULL	
JobTitle	varchar(255)	NO		NULL	
JobDescription	text	NO		NULL	
JobLocation	varchar(255)	NO		NULL	
Salary	decimal(10,2)	YES		NULL	
JobType	enum('Full-time','Part-time','Contract')	NO		NULL	
PostedDate	datetime	NO		NULL	

```
CREATE TABLE Applicants (
    ApplicantID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(100) NOT NULL,
    LastName VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Phone VARCHAR(20) UNIQUE NOT NULL,
    Resume TEXT NOT NULL
);
DESC Applicants;
```

Field	Type	Null	Key	Default	Extra
ApplicantID	int	NO	PRI	NULL	auto_increment
FirstName	varchar(100)	NO		NULL	
LastName	varchar(100)	NO		NULL	
Email	varchar(255)	NO	UNI	NULL	
Phone	varchar(20)	NO	UNI	NULL	
Resume	text	NO		NULL	
Applicants_City	varchar(100)	NO		NULL	
Applicants_State	varchar(100)	NO		NULL	
ExperienceYears	int	YES		NULL	

```
CREATE TABLE Applications (
   ApplicationID INT AUTO_INCREMENT PRIMARY KEY,
   JobID INT NOT NULL,
   ApplicantID INT NOT NULL,
   ApplicationDate DATETIME NOT NULL,
   CoverLetter TEXT NOT NULL,
   CONSTRAINT fk_applications_job FOREIGN KEY (JobID)
REFERENCES Jobs(JobID) ON DELETE CASCADE,
   CONSTRAINT fk_applications_applicant FOREIGN KEY
(ApplicantID) REFERENCES Applicants(ApplicantID) ON DELETE
CASCADE
);
```

For efficiently answering the queries I have added 3 more columns:

```
ALTER TABLE Applicants
ADD COLUMN Applicants_City VARCHAR(100) NOT NULL,
ADD COLUMN Applicants_State VARCHAR(100) NOT NULL,
ADD COLUMN ExperienceYears INT CHECK (ExperienceYears >= 0);
Desc Applications;
```

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

Field	Type	Null	Key	Default	Extra
ApplicationID	int	NO	PRI	NULL	auto_increment
JobID	int	NO	MUL	NULL	
ApplicantID	int	NO	MUL	NULL	
ApplicationDate	datetime	NO		NULL	
CoverLetter	text	NO		NULL	

3. **Define appropriate primary keys, foreign keys, and constraints.**

To display Primary, Unique and Foreign Keys of each table:

```
SHOW KEYS FROM Companies;
```

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
companies	0	PRIMARY	1	CompanyID	A	0	NULL	NULL		BTREE			YES	NULL

```
desc companies;
```

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

Field	Type	Null	Key	Default	Extra
CompanyID	int	NO	PRI	NULL	auto_increment
CompanyName	varchar(255)	NO		NULL	
Location	varchar(255)	NO		NULL	
City	varchar(100)	NO		NULL	
State	varchar(100)	NO		NULL	

SHOW KEYS FROM Jobs;

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
jobs	0	PRIMARY	1	JobID	A	0	NULL	NULL		BTREE			YES	NULL
jobs	1	fk_jobs_company	1	CompanyID	A	0	NULL	NULL		BTREE			YES	NULL

Desc Jobs; (MUL Indicates Foreign key)

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

Field	Type	Null	Key	Default	Extra
JobID	int	NO	PRI	NULL	auto_increment
CompanyID	int	NO	MUL	NULL	
JobTitle	varchar(255)	NO		NULL	
JobDescription	text	NO		NULL	
JobLocation	varchar(255)	NO		NULL	
Salary	decimal(10,2)	YES		NULL	
JobType	enum('Full-time','Part-time','Contract')	NO		NULL	
PostedDate	datetime	NO		NULL	

SHOW KEYS FROM Applicants;

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
applicants	0	PRIMARY	1	ApplicantID	A	0	NULL	NULL		BTREE			YES	NULL
applicants	0	Email	1	Email	A	0	NULL	NULL		BTREE			YES	NULL
applicants	0	Phone	1	Phone	A	0	NULL	NULL		BTREE			YES	NULL

Desc Applicants;

Field	Type	Null	Key	Default	Extra
ApplicantID	int	NO	PRI	NULL	auto_increment
FirstName	varchar(100)	NO		NULL	
LastName	varchar(100)	NO		NULL	
Email	varchar(255)	NO	UNI	NULL	
Phone	varchar(20)	NO	UNI	NULL	
Resume	text	NO		NULL	
Applicants_City	varchar(100)	NO		NULL	
Applicants_State	varchar(100)	NO		NULL	
ExperienceYears	int	YES		NULL	

SHOW KEYS FROM Applications;

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
applications	0	PRIMARY	1	ApplicationID	A	0	NULL	NULL		BTREE			YES	NULL
applications	1	fk_applications_job	1	JobID	A	0	NULL	NULL		BTREE			YES	NULL
applications	1	fk_applications_applicant	1	ApplicantID	A	0	NULL	NULL		BTREE			YES	NULL

Desc Applications; (MUL Indicates Foreign key)

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

Field	Type	Null	Key	Default	Extra
ApplicationID	int	NO	PRI	NULL	auto_increment
JobID	int	NO	MUL	NULL	
ApplicantID	int	NO	MUL	NULL	
ApplicationDate	datetime	NO		NULL	
CoverLetter	text	NO		NULL	

To display all Constraints:(In more Simpler way)

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'Companies'
AND TABLE_SCHEMA = 'CareerHub';
```

Result Grid | Filter Rows: | Export:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
PRIMARY	PRIMARY KEY	companies

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'Jobs'
AND TABLE_SCHEMA = 'CareerHub';
```

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

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
PRIMARY	PRIMARY KEY	jobs
fk_jobs_company	FOREIGN KEY	jobs
jobs_chk_1	CHECK	jobs

jobs

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'Applications'
AND TABLE_SCHEMA = 'CareerHub';
```

Result Grid | Filter Rows: | Export:

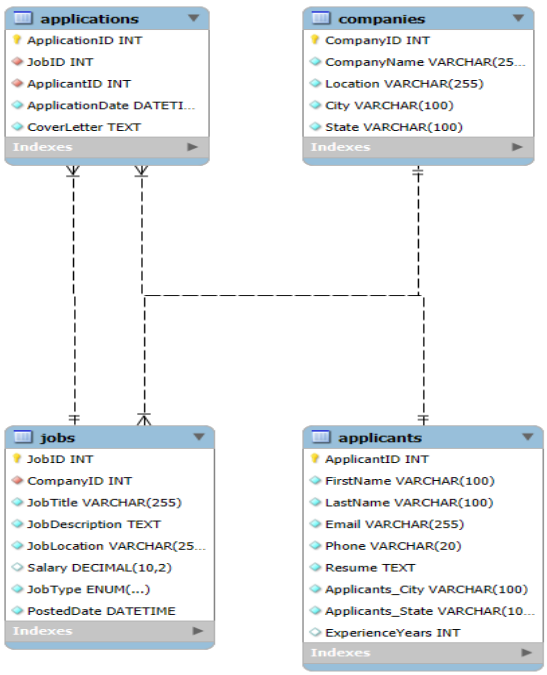
CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
PRIMARY	PRIMARY KEY	applications
fk_applications_applicant	FOREIGN KEY	applications
fk_applications_job	FOREIGN KEY	applications

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,
TABLE_NAME
FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE TABLE_NAME = 'Applicants'
AND TABLE_SCHEMA = 'CareerHub';
```

Result Grid | Filter Rows: | Export:

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
Email	UNIQUE	applicants
Phone	UNIQUE	applicants
PRIMARY	PRIMARY KEY	applicants
applicants_chk_1	CHECK	applicants

ER DIAGRAM:



4. **Ensure the script handles potential errors, such as if the database or tables already exist.**
   I am not having any database or tables with the similar name I had not faced any potential errors but to handle these sort of issues this can be used but to handle these sorts of issues, the IF NOT EXISTS clause can be used when creating databases and tables.

Table with the inserted values

Companies:

Result Grid | Filter Rows: | Edit: | Export/Import:

CompanyID	CompanyName	Location	City	State
1	Google	1600 Amphitheatre Parkway	Mountain View	California
2	Amazon	410 Terry Ave North	Seattle	Washington
3	Microsoft	One Microsoft Way	Redmond	Washington
4	Facebook	1 Hacker Way	Menlo Park	California
5	Tesla	13101 Tesla Road	Austin	Texas
6	Netflix	100 Winchester Circle	Los Gatos	California
NULL	NULL	NULL	NULL	NULL

Jobs:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

JobID	CompanyID	JobTitle	JobDescription	JobLocation	Salary	JobType	PostedDate
1	1	Software Engineer	Develop and maintain applications	Mountain View	90000.00	Full-time	2025-03-01 10:00:00
2	2	Data Scientist	Analyze large datasets and build models	Seattle	110000.00	Full-time	2025-03-02 11:30:00
3	3	Product Manager	Oversee product development	Redmond	120000.00	Full-time	2025-03-03 14:00:00
4	4	Web Developer	Build and maintain websites	Menlo Park	75000.00	Contract	2025-03-04 09:15:00
5	5	Mechanical Engineer	Design and test automotive components	Austin	85000.00	Full-time	2025-03-05 12:45:00
6	6	DevOps Engineer	Manage infrastructure and CI/CD pipelines	Los Gatos	95000.00	Full-time	2025-03-06 15:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Applicants:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ApplicantID	FirstName	LastName	Email	Phone	Resume	Applicants_City	Applicants_State	ExperienceYears
1	Alice	Johnson	alice@example.com	1234567890	Experienced software engineer.	Chennai	Tamil Nadu	4
2	Bob	Smith	bob@example.com	9876543210	Data scientist with AI expertise.	Bangalore	Karnataka	2
3	Charlie	Brown	charlie@example.com	4567891230	Product manager in fintech.	Mumbai	Maharashtra	3
4	David	Williams	david@example.com	7891234560	Web developer skilled in React.	Delhi	Delhi	1
5	Eve	Taylor	eve@example.com	3216549870	Mechanical engineer with auto experience.	CityX	Unknown	3
6	Frank	Miller	frank@example.com	6549873210	DevOps engineer with AWS knowledge.	Hyderabad	Telangana	5
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Applications:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

ApplicationID	JobID	ApplicantID	ApplicationDate	CoverLetter
1	1	1	2025-03-07 10:30:00	Applying for Software Engineer position.
2	2	2	2025-03-07 11:00:00	Applying for Data Scientist position.
3	3	3	2025-03-07 12:15:00	Applying for Product Manager role.
4	4	4	2025-03-07 13:45:00	Applying for Web Developer role.
5	5	5	2025-03-07 14:20:00	Applying for Mechanical Engineer at Tesla.
6	6	6	2025-03-07 15:30:00	Applying for DevOps Engineer at Netflix.
NULL	NULL	NULL	NULL	NULL

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.JobTitle, COUNT(a.ApplicationID) AS ApplicationCount FROM Jobs j LEFT JOIN Applications a ON j.JobID = a.JobID GROUP BY j.JobTitle;

Result Grid | Filter Rows:

JobTitle	ApplicationCount
Software Engineer	1
Data Scientist	1
Product Manager	1
Web Developer	1
Mechanical Engineer	1
DevOps Engineer	1

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.

SET @MinSalary = 80000;
SET @MaxSalary = 100000;
SELECT j.JobTitle, c.CompanyName, j.JobLocation, j.Salary FROM Jobs j JOIN Companies c ON j.CompanyID = c.CompanyID WHERE j.Salary BETWEEN @MinSalary AND @MaxSalary;

Result Grid | Filter Rows: | Export: | Wra

JobTitle	CompanyName	JobLocation	Salary
Software Engineer	Google	Mountain View	90000.00
Mechanical Engineer	Tesla	Austin	85000.00
DevOps Engineer	Netflix	Los Gatos	95000.00

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.

```
SET @ApplicantID = 3;
SELECT j.JobTitle, c.CompanyName, a.ApplicationDate FROM Applications a JOIN Jobs j ON a.JobID = j.JobID JOIN Companies c ON j.CompanyID = c.CompanyID WHERE a.ApplicantID = @ApplicantID;
```

Result Grid | Filter Rows: | Export:

JobTitle	CompanyName	ApplicationDate
Product Manager	Microsoft	2025-03-07 12:15:00

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 AVG(Salary) AS Average_Salary FROM Jobs WHERE Salary > 0;
```

Result Grid | Filter Rov

Average_Salary
95833.333333

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.CompanyName, COUNT(j.JobID) AS JobCount
FROM Jobs j
JOIN Companies c ON j.CompanyID = c.CompanyID
GROUP BY c.CompanyName
HAVING COUNT(j.JobID) = (
   SELECT MAX(JobCount)
   FROM (
      SELECT COUNT(JobID) AS JobCount
      FROM Jobs
      GROUP BY CompanyID
   ) AS JobCounts
);
```

CompanyName	JobCount
Google	1
Amazon	1
Microsoft	1
Facebook	1
Tesla	1
Netflix	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.ApplicantID, a.FirstName,a.LastName, a.ExperienceYears
FROM Applicants a
JOIN Applications app ON a.ApplicantID = app.ApplicantID
JOIN Jobs j ON app.JobID = j.JobID
JOIN Companies c ON j.CompanyID = c.CompanyID
WHERE c.city = 'Austin'
AND a.ExperienceYears >= 3;
```

Result Grid | Filter Rows: | Export:

ApplicantID	FirstName	LastName	ExperienceYears
5	Eve	Taylor	3

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

```
SELECT DISTINCT JobId,JobTitle,Salary FROM Jobs WHERE Salary
BETWEEN 80000 AND 100000;
```

Result Grid | Filter Rows:

JobId	JobTitle	Salary
1	Software Engineer	90000.00
5	Mechanical Engineer	85000.00
6	DevOps Engineer	95000.00

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

```
SELECT j.JobID, j.JobTitle
FROM Jobs j
LEFT JOIN Applications a ON j.JobID = a.JobID
WHERE a.JobID IS NULL;
```

(Since every jobs has received application my tables is empty)

Result Grid | Filter

	JobID	JobTitle

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

```
SELECT a.ApplicantID, a.FirstName, a.LastName, c.CompanyName,
j.JobTitle
FROM Applicants a
JOIN Applications app ON a.ApplicantID = app.ApplicantID
JOIN Jobs j ON app.JobID = j.JobID
JOIN Companies c ON j.CompanyID = c.CompanyID;
```

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

ApplicantID	FirstName	LastName	CompanyName	JobTitle
1	Alice	Johnson	Google	Software Engineer
2	Bob	Smith	Amazon	Data Scientist
3	Charlie	Brown	Microsoft	Product Manager
4	David	Williams	Facebook	Web Developer
5	Eve	Taylor	Tesla	Mechanical Engineer
6	Frank	Miller	Netflix	DevOps Engineer

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.CompanyID, c.CompanyName, COUNT(j.JobID) AS
JobCount
FROM Companies c
LEFT JOIN Jobs j ON c.CompanyID = j.CompanyID
GROUP BY c.CompanyID, c.CompanyName;
```

Result Grid | Filter Rows:

CompanyID	CompanyName	JobCount
1	Google	1
2	Amazon	1
3	Microsoft	1
4	Facebook	1
5	Tesla	1
6	Netflix	1

15. List all applicants along with the companies and positions they have applied for, including those who have not applied.

```
SELECT a.ApplicantID, a.FirstName, a.LastName,
COALESCE(c.CompanyName, 'No Application') AS CompanyName,
    COALESCE(j.JobTitle, 'No Application') AS JobTitle
FROM Applicants a
LEFT JOIN Applications app ON a.ApplicantID = app.ApplicantID
LEFT JOIN Jobs j ON app.JobID = j.JobID
LEFT JOIN Companies c ON j.CompanyID = c.CompanyID;
```

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

ApplicantID	FirstName	LastName	CompanyName	JobTitle
1	Alice	Johnson	Google	Software Engineer
2	Bob	Smith	Amazon	Data Scientist
3	Charlie	Brown	Microsoft	Product Manager
4	David	Williams	Facebook	Web Developer
5	Eve	Taylor	Tesla	Mechanical Engineer
6	Frank	Miller	Netflix	DevOps Engineer

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

```
SELECT c.CompanyID, c.CompanyName
FROM Companies c
JOIN Jobs j ON c.CompanyID = j.CompanyID
WHERE j.Salary > (SELECT AVG(Salary) FROM Jobs);
```

Result Grid | Filter Rows:

CompanyID	CompanyName
2	Amazon
3	Microsoft

17. Display a list of applicants with their names and a concatenated string of their city and state.

```
SELECT ApplicantID, Firstname,Lastname, CONCAT(Applicants_City,
', ', Applicants_State) AS Location
FROM Applicants;
```

Result Grid | Filter Rows: | Export: | Wr

ApplicantID	Firstname	Lastname	Location
1	Alice	Johnson	Chennai, Tamil Nadu
2	Bob	Smith	Bangalore, Karnataka
3	Charlie	Brown	Mumbai, Maharashtra
4	David	Williams	Delhi, Delhi
5	Eve	Taylor	CityX, Unknown
6	Frank	Miller	Hyderabad, Telangana

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

```
SELECT JobID, JobTitle
FROM Jobs
WHERE JobTitle LIKE '%Developer%' OR JobTitle LIKE
'%Engineer%';
```

Result Grid | Filter Rows:

JobID	JobTitle
1	Software Engineer
4	Web Developer
5	Mechanical Engineer
6	DevOps Engineer
NULL	NULL

Solved 18 questions as per Sir Guidelines
Questions left: 19,20