Applicant.py:

```
from .Jobapplication import jobapplication
from datetime import datetime

class applicant:
    def __init__(self,applicant_id,first_name,last_name,email,phone_number,resume):
        self.applicant_id = applicant_id
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.phone_number = phone_number
        self.resume = resume

    def create_profile(self, db):
        if "@" not in self.email or "." not in self.email.split("@")[-1]:
            raise ValueError("Invalid email format")
        db.insert_applicant(self)

    def apply_for_a_job(self, db, job_id, coverLetter):
        job_application = jobapplication(None, job_id, self.applicant_id, datetime.now(), coverLetter)
        db.insert_job_application(job_application)
```

company.py

```
from entity.Joblistings import Joblistings
from datetime import datetime
class company:
    def __init__(self,company_id,company_name,location):
        self.company_id = company_id
        self.company_name = company_name
        self.location = location

    def post_job(self,db,job_title,job_description,job_location,salary,job_type):
        job =
Joblistings(None,self.company_id,job_title,job_description,job_location,salary,job_type,datetime.no
w())
        db.insert_joblisting(job)

    def get_job(self, db):
        return db.GetJobsByCompany(self.company_id)
```

Jobapplication:

```
class jobapplication:
    def __init__(self,application_id,job_id,applicant_id,application_date,coverLetter):
        self.application_id = application_id
        self.job_id = job_id
        self.applicant_id = applicant_id
        self.application_date = application_date
        self.coverLetter = coverLetter
```

Joblistings.py:

```
from entity.Jobapplication import jobapplication
from datetime import datetime

class Joblistings:
    def
__init__(self,job_id,company_id,job_title,job_description,job_location,salary,job_type,posted_date):
        self.job_id = job_id
        self.company_id = company_id
        self.job_title = job_title
        self.job_description = job_description
        self.job_location = job_location
        self.salary = salary
        self.job_type = job_type
        self.posted_date = posted_date

    def Apply(self, db, applicant_id, cover_letter):
        application = jobapplication(None, self.job_id, applicant_id, datetime.now(), cover_letter)
        db.insert_job_application(application)

    def GetApplicants(self, db):
        return db.GetApplicationsForJob(self.job_id)


    def Apply(self, db, applicant_id, cover_letter):
        application = jobapplication(None, self.job_id, applicant_id, datetime.now(), cover_letter)
        db.insert_job_application(application)

    def GetApplicants(self, db):
        return db.GetApplicationsForJob(self.job_id)
```

Databasemanager.py

```
import mysql.connector
from mysql.connector import Error



class Databasemanager:

    def __init__(self):
        try:
            self.con = mysql.connector.connect(
                host = 'localhost',
                user = 'root',
                password = 'Zuhi743#',
                database = 'careerhub'
            )
            self.cursor = self.con.cursor(dictionary=True)
        except Error as e:
            print(f"Database connection error:{e}")

    def insert_company(self,company):
        try:
            self.cursor.execute("insert into company(company_id,company_name,location) values
(%s,%s,%s)",
                                (company.company_id,company.company_name,company.location))
            self.con.commit()
        except Error as e:
            print(f"Error inserting company:{e}")

    def insert_joblisting(self,Joblistings):
        try:
            if Joblistings.salary < 0:
                raise ValueError("Salary cannot be negative")
            self.cursor.execute(
                    """
                    insert into joblistings(company_id,job_title,
                    job_description,job_location,salary,job_type,posted_date)
                    values(%s,%s,%s,%s,%s,%s,%s)

""",(Joblistings.company_id,Joblistings.job_title,Joblistings.job_description,Joblistings.job_location,Jo
blistings.salary,Joblistings.job_type,Joblistings.posted_date))
            self.con.commit()
        except Error as e:
            print(f"Error inserting job:{e}")

    def insert_applicant(self,applicant):
```

```
        try:
            self.cursor.execute(
                """
            insert into applicant(applicant_id,first_name,last_name,email,phone_number,resume)
            values(%s,%s,%s,%s,%s,%s)""",
(applicant.applicant_id,applicant.first_name,applicant.last_name,
                                applicant.email,
                                applicant.phone_number,applicant.resume))
            self.con.commit()
        except Error as e:
            print(f"Error inserting applicant:{e}")

    def insert_jobapplication(self,jobapplication):
        try:
            self.cursor.execute(
                """
                insert into jobapplication(job_id,applicant_id,application_date,coverletter)
                values(%s,%s,%s,%s)

""",(jobapplication.job_id,jobapplication.applicant_id,jobapplication.application_date,jobapplication.
coverletter))
            self.con.commit()
        except Error as e:
            print(f"Error inserting application:{e}")

    def get_job_listings(self):
        self.cursor.execute(
            """
            select j.job_title,c.company_name,j.salary
            from joblistings j
            join company c on
            j.company_id = c.company_id
            """)
        return self.cursor.fetchall()

    def get_applicants(self):
        self.cursor.execute("select*from applicant")
        return self.cursor.fetchall()

    def get_companies(self):
        self.cursor.execute("select*from company")
        self.cursor.fetchall()

    def get_applicants_for_job(self,job_id):
        self.cursor.execute("select*from joblistings where job_id = %s",(job_id,))
        self.cursor.fetchall()

    def salary_range(self,min_salary,max_salary):
```

```
        try:
            self.cursor.execute(
                """
                select j.job_title,c.company_name,j.salary
                from joblistings j
                join company c on
                j.company_id = c.company_id
                where j.salary between %s and %s
                 """,(min_salary,max_salary))
            return self.cursor.fetchall()
        except Error as e:
            print(f"Error:{e}")

    def get_company_by_id(self,company_id):
        try:
            cursor = self.con.cursor(dictionary=True)
            cursor.execute("select company_name from company where company_id =
%s",(company_id,))
            result = cursor.fetchone()
            return result['company_name'] if result else None
        except Exception as e:
            print("Error:{e}")
            return None

    def get_applicant_name_byid(self,applicant_id):
        try:
            cursor=self.con.cursor(dictionary=True)
            cursor.execute("select first_name,last_name from applicant where applicant_id = %s",
                        (applicant_id,))
            result = cursor.fetchone()
            return (result['first_name'],result['last_name'])if result \
            else None
        except Exception as e:
            print("Error:{e}")
            return None

    def upload_resume(self,file_path):
        try:
            if not file_path.endswith('.pdf'):
                raise ValueError("Unsupported file format. Only PDF allowed.")
            with open(file_path, 'rb') as f:
                data = f.read()
                if len(data) > 5 * 1024 * 1024:
                    raise ValueError("File size exceeds limit (5MB).")
                return "Resume uploaded successfully"
        except FileNotFoundError:
            print("Resume file not found.")
        except ValueError as ve:
```

```
 print(ve)
```

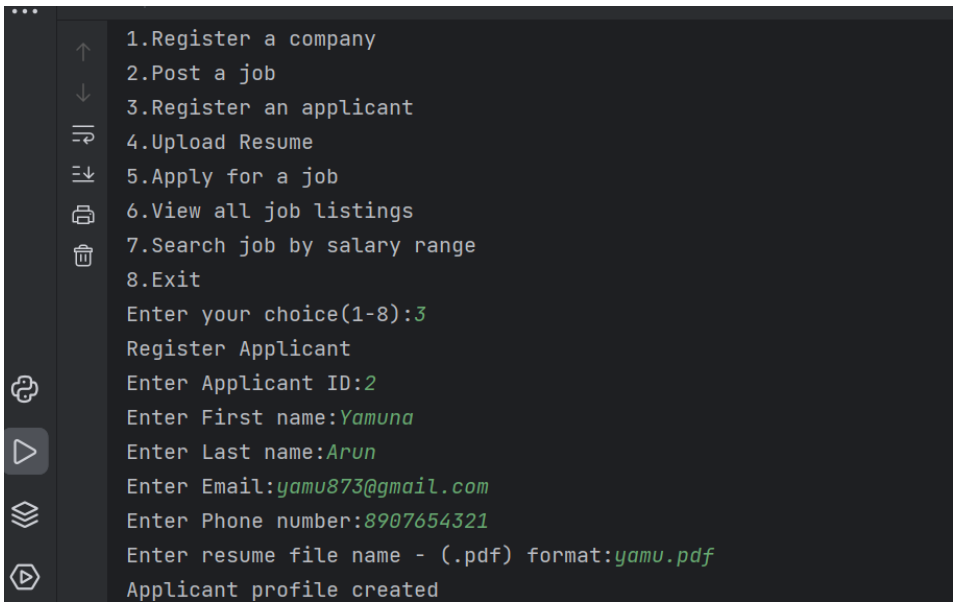
Main.py

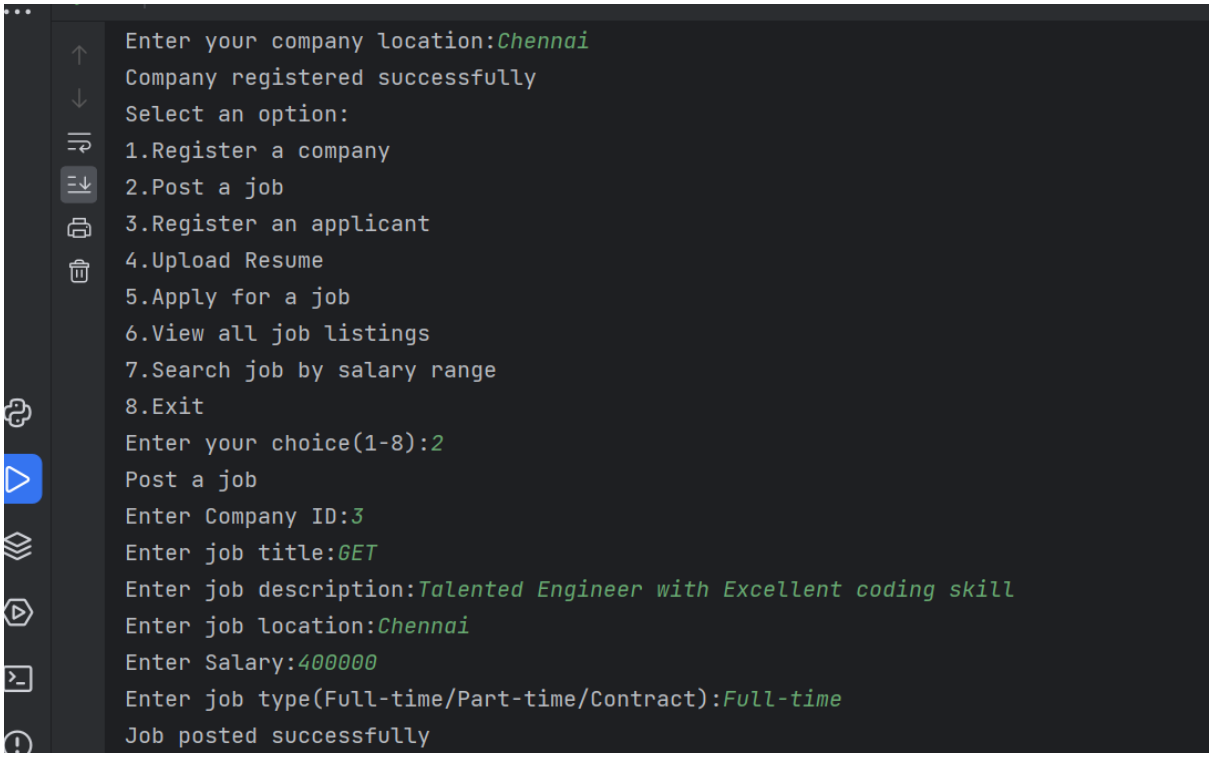
```
from entity.Applicant import applicant

from entity.Company import company

from entity.databasemanager import Databasemanager

def main():
    db = Databasemanager()
    print("Welcome to CareerHub")

    while True:
        print("Select an option:")
        print("1.Register a company")
        print("2.Post a job")
        print("3.Register an applicant")
        print("4.Upload Resume")
        print("5.Apply for a job")
        print("6.View all job listings")
        print("7.Search job by salary range")
        print("8.Exit")

        choice = int(input("Enter your choice(1-8):"))

        try:
            if choice == 1:
                print("Register Company")
                company_id = int(input("Enter company_id:"))
                company_name = input("Enter your company name:")
                location = input("Enter your company location:")
                data = company(company_id,company_name,location)
                db.insert_company(data)
                print("Company registered successfully")

            elif choice == 2:
                print("Post a job")
                company_id = int(input("Enter Company ID:"))
                company_name = db.get_company_by_id(company_id)
                if not company_name:
                    print("Company not found")
                    continue
                job_title = input("Enter job title:")
                job_desc = input("Enter job description:")
```

```
            job_location = input("Enter job location:")
            salary = float(input("Enter Salary:"))
            job_type = input("Enter job type(Full-time/Part-time/Contract):")
            data = company(company_id,company_name,"")
            data.post_job(db,job_title,job_desc,job_location,salary,job_type)
            print("Job posted successfully")

        elif choice == 3:
            print("Register Applicant")
            applicant_id = int(input("Enter Applicant ID:"))
            first_name = input("Enter First name:")
            last_name = input("Enter Last name:")
            email = input("Enter Email:")
            phone_number = input("Enter Phone number:")
            resume = input("Enter resume file name - (.pdf) format:")
            app = applicant(applicant_id,first_name,last_name,email,phone_number,resume)
            app.create_profile(db)
            print("Applicant profile created")

        elif choice == 4:
            print("Upload Resume")
            resume_pdf = input("Enter Resume file name-(.pdf) format:")
            upload_resume(resume_pdf)
            print("Resume uploaded Successfully")

        elif choice == 5:
            print("Apply for a job")
            applicant_id = int(input("Enter your applicant ID:"))
            job_id = int(input("Enter Job ID to apply:"))
            cover_letter = input("Enter cover letter:")
            applicant_name = db.get_applicant_name_byid(applicant_id)
            if not applicant_name:
                print("Application not found")
                continue
            resume = "resume.pdf"
            app = applicant(applicant_id,applicant_name[0],applicant_name[1],"","",resume)
            app.apply_for_a_job(db,job_id,"")
            print("Application submitted")

        elif choice == 6:
            print("Job Listings")
            jobs = db.get_job_listings()
            if not jobs:
                print("No jobs available")
            for j in jobs:
                print(f"{j['job_id']}:{j['job_title']}-
{j['company_name']}|{j['job_type']}|{j['salary']}|{j['location']}")
```

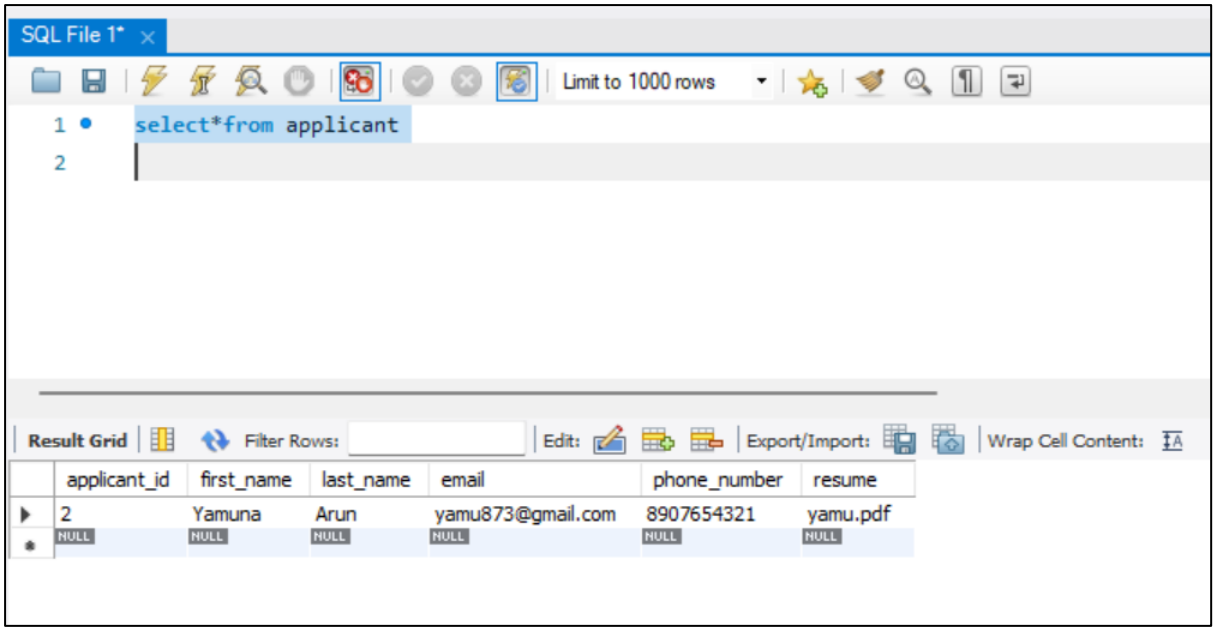
```
        elif choice == 7:
            print("Search jobs by salary range")
            min_salary = float(input("Enter minimum salary:"))
            max_salary = float(input("Enter maximum salary:"))
            jobs = db.salary_range(min_salary,max_salary)
            if not jobs:
                print("No jobs found in the above range")
            for j in jobs:
                print(f"{j['job_id']}:{j['job_title']},{j['company_name']},{j['salary']}")

        elif choice == 8:
            print("Thank you for using Careerhub,Have a bright Future!")
            break;
        else:
            print("Invalid choice,enter number btw 1-8")

    except ValueError as e:
        print(f"Error:{e}")
    except FileNotFoundError as f:
        print(f"Error:{f}")
    except Exception as e:
        print(f"Unexpected Error:{e}")

if __name__ == "__main__":
    main()
```

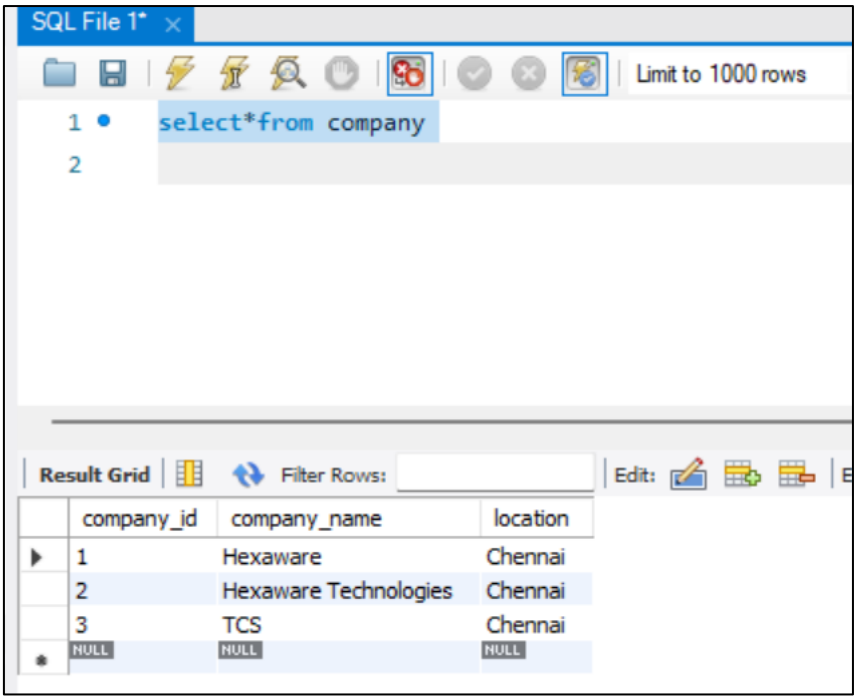
OUTPUT:

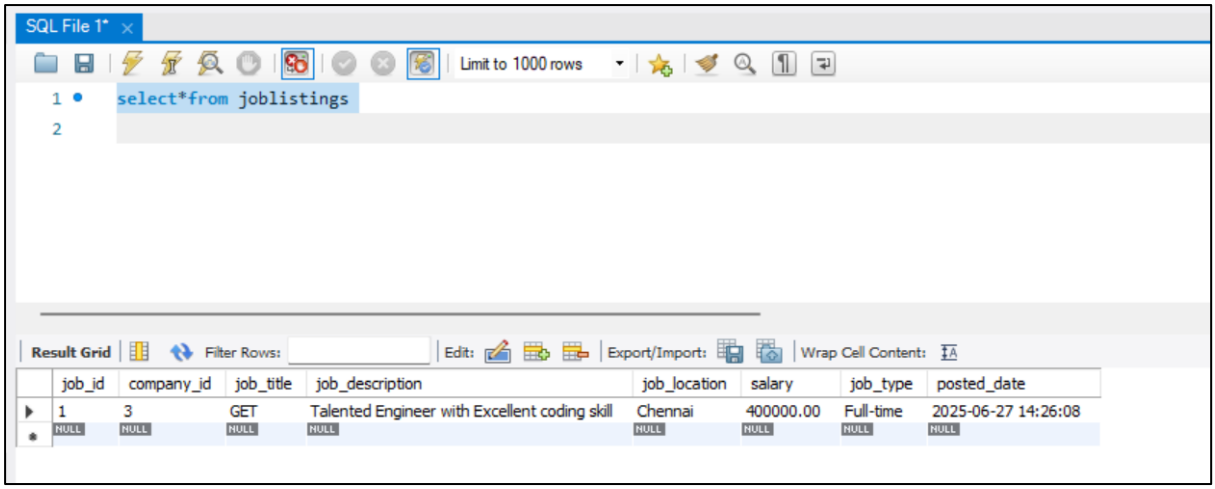



Enter your company location:Chennai
Company registered successfully
Select an option:
1.Register a company
2.Post a job
3.Register an applicant
4.Upload Resume
5.Apply for a job
6.View all job listings
7.Search job by salary range
8.Exit
Enter your choice(1-8):2
Post a job
Enter Company ID:3
Enter job title:GET
Enter job description:Talented Engineer with Excellent coding skill
Enter job location:Chennai
Enter Salary:400000
Enter job type(Full-time/Part-time/Contract):Full-time
Job posted successfully


D:\Python_Codingchallenge\venv\Scripts\python.exe D:\Python_Codingchallenge\main.py
Welcome to CareerHub
Select an option:
1.Register a company
2.Post a job
3.Register an applicant
4.Upload Resume
5.Apply for a job
6.View all job listings
7.Search job by salary range
8.Exit
Enter your choice(1-8):1
Register Company
Enter company_id:3
Enter your company name:TCS
Enter your company location:Chennai
Company registered successfully


SQL File 1*
Limit to 1000 rows
1 select*from applicant
2
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
applicant_id first_name last_name email phone_number resume
2 Yamuna Arun yamu873@gmail.com 8907654321 yamu.pdf
NULL NULL NULL NULL NULL NULL


SQL File 1*
Limit to 1000 rows
1 select*from company
2
Result Grid
Filter Rows:
Edit:
company_id company_name location
1 Hexaware Chennai
2 Hexaware Technologies Chennai
3 TCS Chennai
NULL NULL NULL


SQL File 1*
Limit to 1000 rows
1 select*from joblistings
2
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
job_id company_id job_title job_description job_location salary job_type posted_date
1 3 GET Talented Engineer with Excellent coding skill Chennai 400000.00 Full-time 2025-06-27 14:26:08
NULL NULL NULL NULL NULL NULL NULL NULL