Description:
This database manages information related to hospitals, positions available, candidates applying for those positions, and their skills. It tracks interviews between candidates and hospitals for specific positions, recording interview details such as interview dates. It also maintains a record of skills that the candidates have and also maintains a record for required skills for each position.

Hospitals:	Stores information about hospitals
Candidates:	Stores details about the candidates
Positions:	Keeps track of available positions, linking them to hospitals offering those positions
Interviews:	Records interview conducted, linking positions, candidates, hospitals and their interview details
InterviewDates:	Stores interview dates associated with interviews
Skills:	Available list of Skills
CandidateSkills:	Links Candidates and Skills they have
PositionSkills:	Links Positions and Skills they require

Assumptions:
Basic information: These tables only contain basic information while in reality there may be more relevant information that could be taken into account. For example candidates include names, addresses, and contact details but it doesn't include additional information that hospitals may find relevant such as education background, work experience, or other additional certificates.

Simplified Interview Dates: The design separates interview dates into a separate table (InterviewDates) linked to interviews. Separation allows for potential management of multiple interview dates for a single interview session if needed.

No Salary or Offer Details: The schema doesn't include salary offered, contract details, salary negotiations.

Foreign key Constraints are used to maintain referential integrity across tables so that relationships between tables are maintained.

Single interview date per interview. If interviews span multiple days or have varying time slots, this schema might not capture the interview schedule adequately.

Singular hospital for each position - it assumes each position is directly associated with one hospital. Some positions may span across multiple hospitals or be shared among them, which this schema doesn't capture.

Candidate offer status is in binary, true or false, however there could be multiple stages such as pending decisions, negotiation stages, etc.

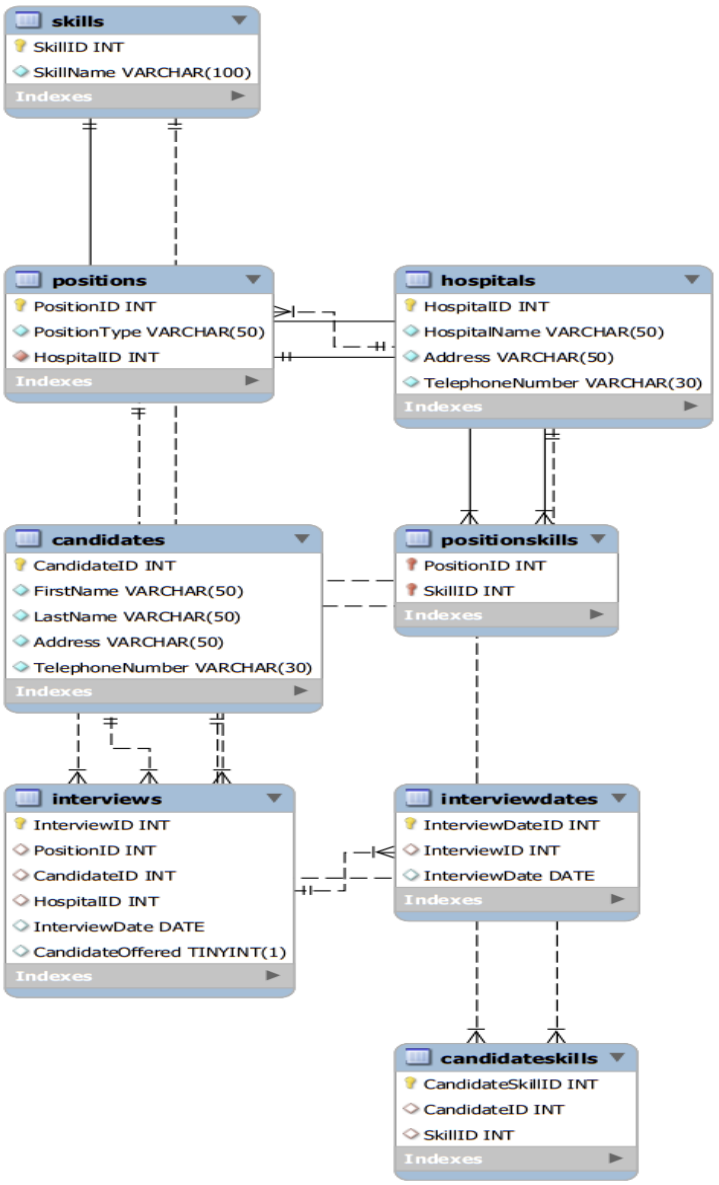
No position Status Tracking - if they are open, closed, filled within the schema.

Reaction policies:
Foreign Key Constraints: These policies ensure referential integrity between tables. For instance, in the schema, foreign key constraints are used to link Positions, Candidates, Interviews, Skills, and Hospitals tables. They prevent actions that would violate these relationships, such as deleting a hospital while positions are still associated with it.

I set up the database using just one type of rule called foreign key constraints. I didn't add any extra tricks or complicated methods to react to changes. This keeps things simple and makes sure that all the connections between different parts of the database stay reliable. By sticking to these basic rules, I avoid making things more complicated or harder to manage. This way, the database stays easy to handle and understand, especially when I don't need fancy reactions to data changes.

Sample Date:

Once I finished setting up how things should look in my database (tables) and after creating the stored procedures for it to insert rows for each table, I asked ChatGPT to help me fill it with pretend information. I gave ChatGPT examples of what the different parts of the database should look like, like hospitals, candidates, and interviews. ChatGPT then came up with make-believe names, addresses, and details that fit how real hospitals and candidates might be. I tweaked my questions to ChatGPT based on what I needed, like asking for different skills a candidate might have or when interviews could happen. Once I got all this pretend data, I used the stored procedures I made earlier in the database to put this data in the right places. These special ways I set up before helped make sure the pretend data fit nicely into the database without causing any problems. This teamwork between ChatGPT and the ways I made for the database let me fill it up with lots of pretend information that I used to test out step 4 in the assignment.


skills
SkillID INT
SkillName VARCHAR(100)
Indexes
positions
PositionID INT
PositionType VARCHAR(50)
HospitalID INT
Indexes
hospitals
HospitalID INT
HospitalName VARCHAR(50)
Address VARCHAR(50)
TelephoneNumber VARCHAR(30)
Indexes
candidates
CandidateID INT
FirstName VARCHAR(50)
LastName VARCHAR(50)
Address VARCHAR(50)
TelephoneNumber VARCHAR(30)
Indexes
positionskills
PositionID INT
SkillID INT
Indexes
interviews
InterviewID INT
PositionID INT
CandidateID INT
HospitalID INT
InterviewDate DATE
CandidateOffered TINYINT(1)
Indexes
interviewdates
InterviewDateID INT
InterviewID INT
InterviewDate DATE
Indexes
candidateskills
CandidateSkillID INT
CandidateID INT
SkillID INT
Indexes