# DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES1201800131 Anirudh Haritas Murali

This project aims to demonstrate a working database system for the domain of company interviews. The data model contains representations for all of the major participants in an interview process, such as candidates, interviewers, questions and more. The model has been designed keeping in mind good database design practices, adhering to the normal forms and enforcing appropriate constraints as necessary. The queries are aimed at returning useful information about the current status of the various entities, as well as notable statistics about the overall procedure. To demonstrate the model, a simple command-line interface has been built in Python with access to the essential functions required to operate the model .

**NOTE: All queries and scripts used here can be found at https://github.com/anihm136/DBMS_Project/**

## Introduction

The chosen domain of discourse is the interview process in a company. Some of the salient features of the data model are -

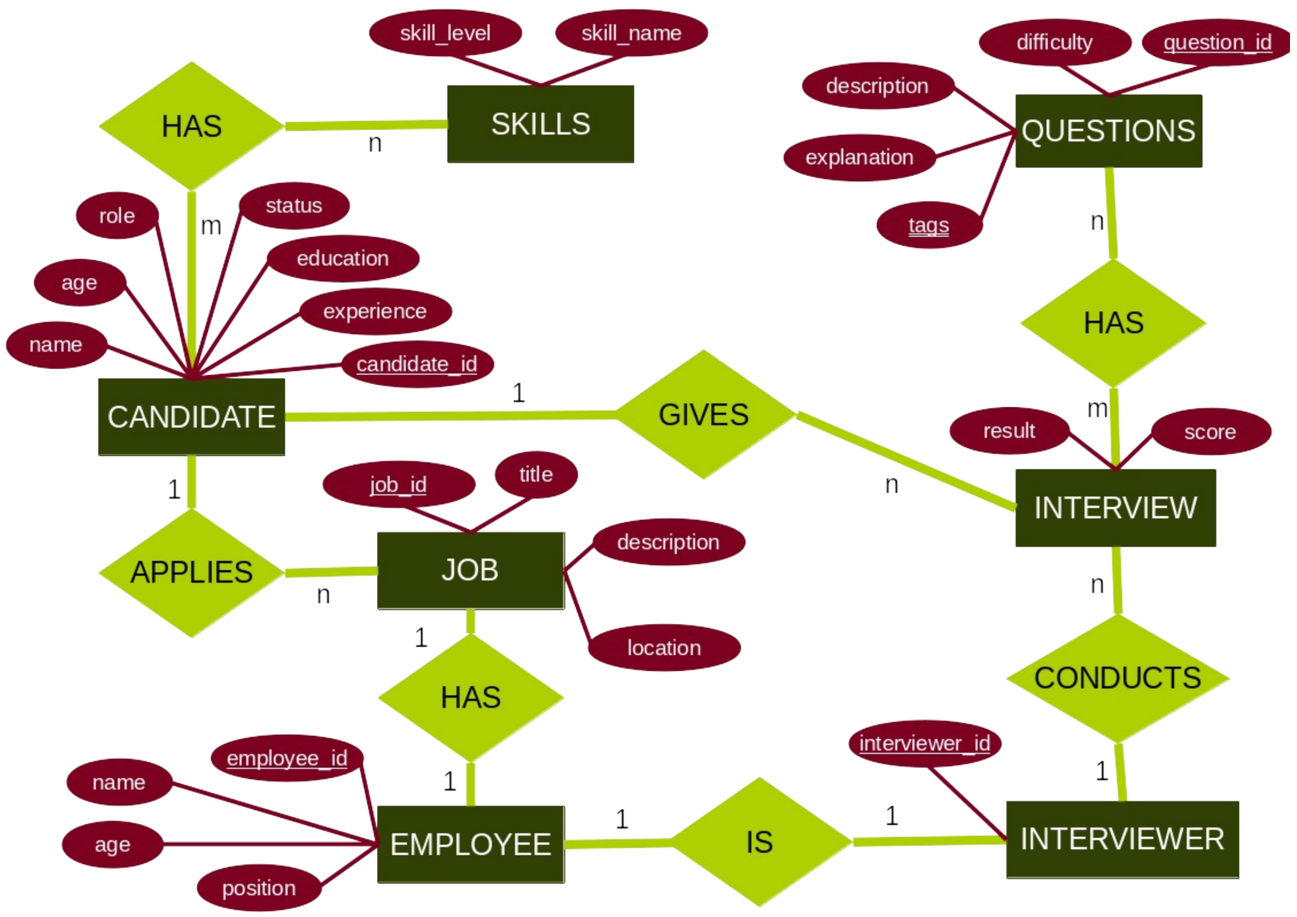
**ENTITIES**

- Candidates, with their associated information
- Employees, with their associated information (to show the link between accepted candidates and current employees, as well as obtain information about the interviewers without duplication)
- Interviewers, with minimal information (to be accessed from employee record)
- Interviews, with information on the participants, questions asked and results
- Questions, with the description, difficulty, related domains and explanations
- Skills, to associate candidates with specific skills and aptitudes in those skills
- Question tags, to associate questions with particular domains
- Job, with details about it to keep track of filled positions and applicants

**RELATIONS**

- Candidates are related to skills as per their aptitude
- Candidates and interviewers are related to interviews, as participants
- Interviewers are related to employees, as all interviewers must be employees
- Questions are related to tags, to denote the domain of the question
- Interviews are related to questions, to keep track of all questions asked in each interview
- Employees are related to jobs as having a job
- Candidates are related to jobs as applying for a job

## Data Model



In most cases, a surrogate key (id) has been added to simplify the resolution of functional dependencies and to reduce memory use for indexing the keys. Some cases require compound keys.

## FD and Normalization

- Candidate : all colums are dependent on candidate_id
- Interviewer: employee_id is dependent on interviewer_id, employee_id is a candidate key
- Interview: All columns are dependent on interview_id
- Employee: All attributes are dependent on employee_id
- Skills: Skill_level is dependent on compound key of candidate_id and skill_name
- Tags: Tag_name is dependent on tag_id

- Question description, explanation, difficulty: All attributes dependent on question_id
- Map: Both colums of the table form compound primary key
- Job_details,job_role: All columns are dependent on the job_id

Some places where normal forms were violated when constructing the schema diagram from the ER diagram -

- Question id, description, explanation and difficulty were initially in a single table, violating 2NF due to dependency on non-key attribute
- Employee details were duplicated in interviewer table, violating 3NF due to transitive dependency on non primary-key employee_id
- Tags and map table were created to avoid violation of 2NF when creating the relation with foreign keys, due to dependency on non-key attributes

## DDL

Tables are created kepping constraints and relations in mind. The entire data description language can be found here : https://github.com/anihm136/DBMS_Project/blob/f1bbab5a5315c66ecde56183cc37a9c8c7a86b43/create.sql#L15

## Triggers

Three primary trigger conditions have been identified -

1. When a candidate is accepted in an interview, he/she becomes an employee. Thus, a trigger is created to transfer the candidate's relevant details to the employee table when he/she is accepted in an interview
2. When an employee is added to the employee table, it is inferred that they are starting at the current date. Thus, a trigger is created to automatically fill the start_date filed with the current date when an employee is added
3. When a candidate is added, their status will always be in the initial state i.e, 'entry received'. Thus, a trigger is written to fill in the initial status value when a candidate is added to the candidate table

The sql for creating triggers can be found here : https://github.com/anihm136/DBMS_Project/blob/f1bbab5a5315c66ecde56183cc37a9c8c7a86b43/create.sql#L96

## SQL Queries

A large number of SQL queries have been written and tested to extract useful information from the database. Following is a list of implemented queries (in the UI)-

1. Retrieve number of jobs in each location
2. Find number of locations in which each job title exists
3. Retrieve all details of a job based on job id
4. Retrieve all job listings for a specific location
5. List number of available jobs
6. Check status of current applications

7. List number of applicants in each location
8. Retrieve details of a candidate based on candidate id
9. Retrieve list of candidates based on age group (lower and upper bound)
10. Find all skills and respective aptitudes of a candidate
11. Find all candidates with a specific skill (with a specified minimum aptitude)
12. Retrieve employee details based on employee id
13. Find number of employees at different corporate positions
14. Retrieve interviewer details by interviewer id
15. List candidates interviewed by a specific interviewer
16. Find the distribution of difficulty of questions asked by the interviewer
17. Show question details for question id
18. List all interviews and respective results of a specific candidate
19. List all interviews and respective results conducted by an interviewer
20. List all scheduled (but not yet conducted) interviews
21. Retrieve the distribution of candidate results
22. Filter candidates by average interview score
23. Find the distribution of questions asked in interviews
24. Find the difficulty score of an interview
25. Create a summary of a candidate (details, interviews, results)
26. Summarize a job title (details, employees, skills)
27. Summarize an interviewer (details, interviews, results)

All implemented queries can be found here :
https://github.com/anihm136/DBMS_Project/blob/master/queries.sql

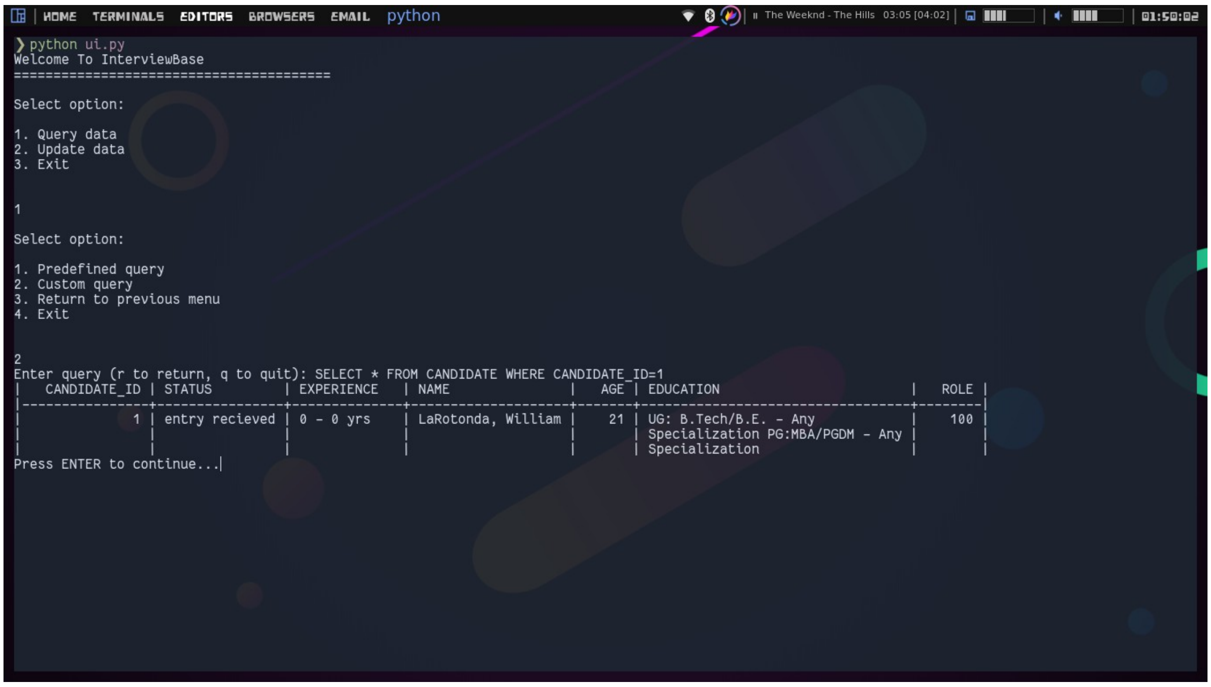
For demonstration, a python script has been written to populate the database with relevant data. Using this, the structure and constraints of the model can be shown, while verifying the accuracy of the queries

## Conclusion

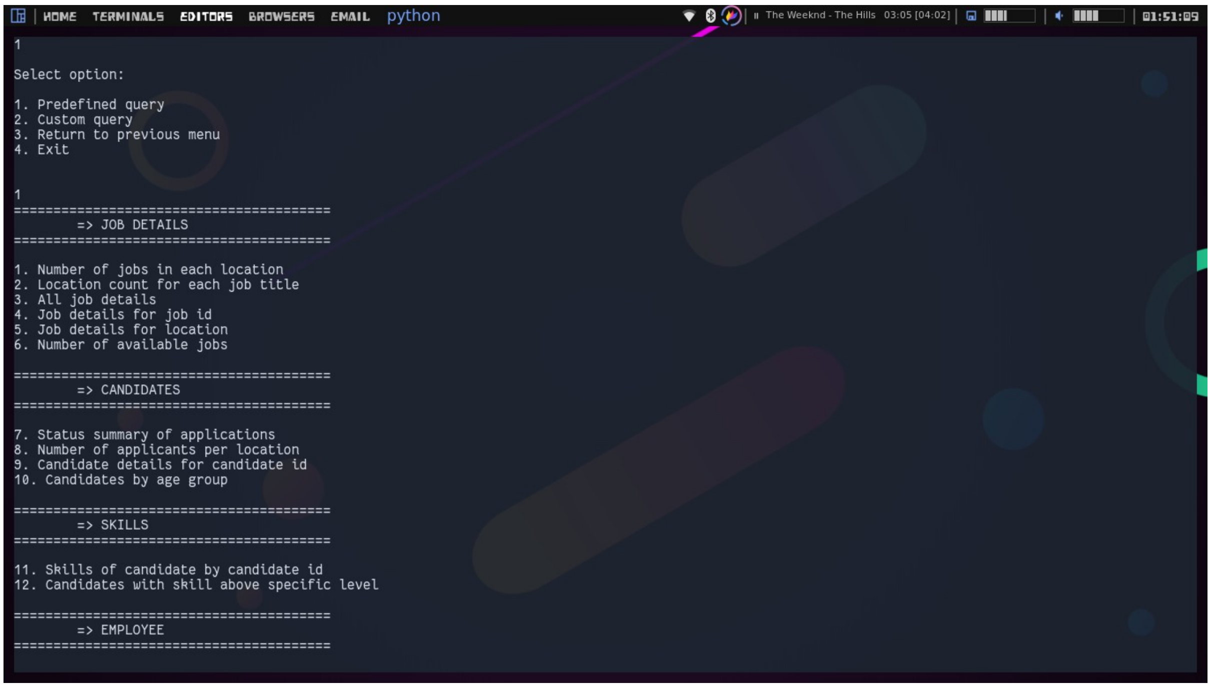
The above described database system is a rudimentary data management system for a company's interview management. It has all the basic necessities for storing relevant information about the participants and results, and can be easily customised based on individual needs.
Further work can be done on adding relevant queries, adding and integrating more relevant entities to expand upon the miniworld in order to provide a real enterprise solution, and to improve the user interface for working with it.

# Appendix

A simple command-line based user interface has been developed using Python in order to easily access the database. It has functionality to easily perform common, useful queries, execute custom queries and perform quick insertion and updation of common entities.


HOME TERMINALS EDITORS BROWSERS EMAIL python
The Weeknd - The Hills 03:05 [04:02]
01:50:02

```
❯ python ui.py
Welcome To InterviewBase
========================================

Select option:

1. Query data
2. Update data
3. Exit


1

Select option:

1. Predefined query
2. Custom query
3. Return to previous menu
4. Exit


2
Enter query (r to return, q to quit): SELECT * FROM CANDIDATE WHERE CANDIDATE_ID=1
|    CANDIDATE_ID | STATUS          | EXPERIENCE    | NAME                |   AGE | EDUCATION                         |   ROLE |
|-----------------+-----------------+---------------+---------------------+-------+-----------------------------------+--------|
|               1 | entry recieved  | 0 - 0 yrs     | LaRotonda, William  |    21 | UG: B.Tech/B.E. - Any             |    100 |
|                 |                 |               |                     |       | Specialization PG:MBA/PGDM - Any  |        |
|                 |                 |               |                     |       | Specialization                    |        |
Press ENTER to continue...
```


HOME TERMINALS EDITORS BROWSERS EMAIL python
The Weeknd - The Hills 03:05 [04:02]
01:51:09

```
1

Select option:

1. Predefined query
2. Custom query
3. Return to previous menu
4. Exit


1
========================================
        => JOB DETAILS
========================================

1. Number of jobs in each location
2. Location count for each job title
3. All job details
4. Job details for job id
5. Job details for location
6. Number of available jobs

========================================
        => CANDIDATES
========================================

7. Status summary of applications
8. Number of applicants per location
9. Candidate details for candidate id
10. Candidates by age group

========================================
        => SKILLS
========================================

11. Skills of candidate by candidate id
12. Candidates with skill above specific level

========================================
        => EMPLOYEE
========================================
```