MSCI 346

Project Submission 2

Ayser Choudhury 20603805

Victoria Li

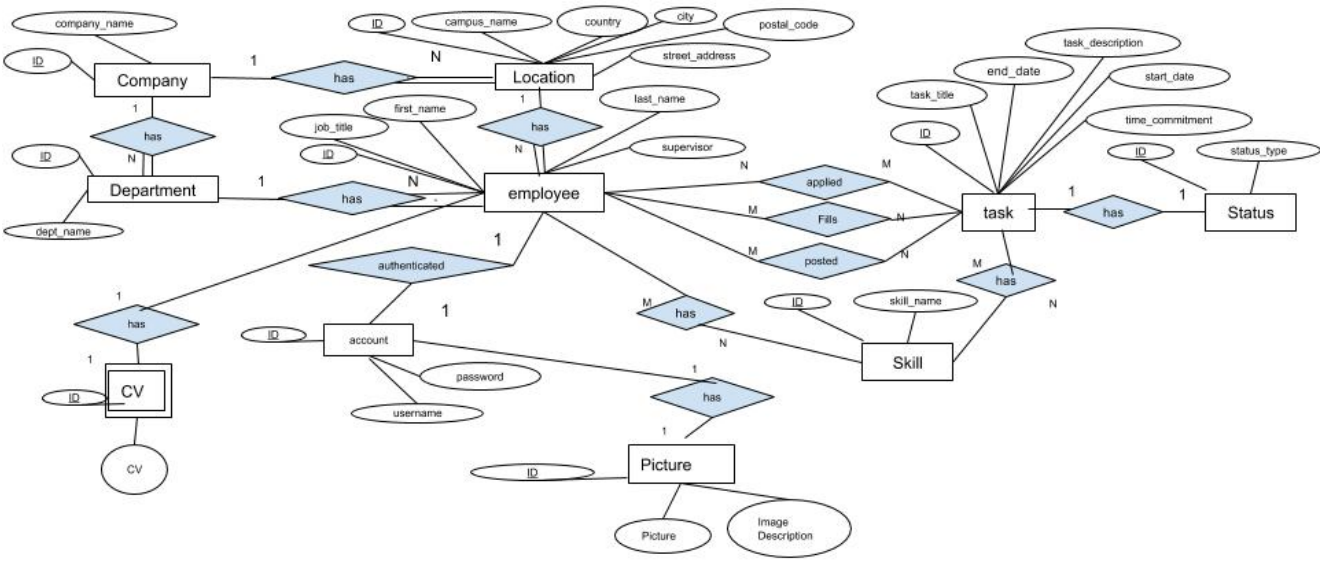
Aaron Yim

TABLE OF CONTENTS

Changes from previous ER Diagram	1
Explanation of Each Table and Attribute with Sample Data	3
How Data is Collected Normally	15
How Data Was Created for The Project	18
Example of VBA Code used for Creating Employee Information	18
Example of Python Code used to Create Task - Job Titles	22
Queries	24

Moving forward from the previous submission, the team decided to make a few changes to the data in order to optimize the space requirements of the database as well as the speed of the queries. In order to do this, a few changes were made to the structure of the database, and this can be seen in the update ER diagram below, and are elaborated on below. The specifics of what each entity and attribute are elaborated on later.

Changes from previous ER Diagram



- Company:
  - Introducing ID as Primary Key, This will speed up querying, optimize searches, and allow companies of the same name to subscribe to the service
  - Examples include “Apple Inc” (Creators of the smartphone) and “Apple Corps” (Media company started by the the Beatles”
- Location:
  - Introducing ID As Primary Key. This will speed up querying, optimize searches and allow different companies to have the same campus name. Example : Manulife and Facebook can now both have a campus names “Mars Discovery District”

    - “Address” attribute No longer exists. Country, City, Postal Code and Street Address now each exist as an attribute of a location. The purpose of this is to optimize storage.
- Department:
    - Introducing ID as Primary Key. This will speed up querying, optimize searches and does not force companies to adopt a system to number their departments in the over -all database. ID numbers will differentiate departments between and within companies.
    - Department is no longer connected to a location. This is because an increasing number of companies are starting to have cross-geographical departments, and this is to accommodate these type of multinational companies.
- Employee:
    - An employee now has both a department and a location. Since Departments no longer belong to a location, an employee needs to belong to belong to a given location which acts as their “base”
    - Skills are no longer multivalued attributes, but a separate entity. This is due to space optimization, and is elaborated on later.
    - Instead of an SSN as the primary key, ID is now the primary key. This is in order to protect the privacy of each individual employee
    - There is also now a relationship called “Filled” Tasks which are used to determine which employee has fulfilled a given task
- Task:
    - Similar to Employees, Tasks no longer have Skills. Skills are a separate entity, and this is elaborated on later.
    - Task_id is changed to ID in order to adhere to a naming convention
    - Task Status is now is its own entity, in order for space optimization
- Status:
    - In order to improve space optimization, Status is used. As each task would have a status of either “Filled” or “Available”, this would save space in the long term, as each task can now have. a foreign key attribute instead.

- Skill:
  - As mentioned earlier, Skills is now its own entity. This is done in order to optimize the database both in terms of speed and memory.
  - As many employees, and tasks will request the same skills, but not always with the same other set of skills (example R in one job posting and R and VBA in another job posting) and there will be a large overlap between skills of different employees and different tasks, each skill is given its own ID for space optimization
  - When tasks or employees are being queried for specific skills, the skills are faster to find and offer a less expensive operation; this sort of operation makes queries to the database faster.
- Account
  - Accounts no longer have a profile picture url. Instead, they have a relationship with another entity that is used to store
- CV and Pictures:
  - Resume has been renamed to CV
  - These are now separate tables that are created specifically for the purpose of file storage. This is done for space and speed optimization

Explanation of Each Table and Attribute with Sample Data

Company:

Each of the other tables all connect back to “Company” either directly or indirectly. For each company in our database, there are Departments, Employees, and Locations, which each also tie into other tables and entities. For the sake of simplicity for the presentation of the tool. only one company has been added to the database. Typically this sort of information would be stored on a database locally, on the individual company’s servers or in the cloud, and no member of a company would have access to the information posted about another company**.**

A snapshot of the data in the company table is given below, along with the structure of the table is given below:

Table Structure for Table: Company

Column	Type	Null	Default
ID	int(11)	No	
company_name	varchar(30)	No	

Data for Table: Company

Company ID	Company Name
1	FACEBOOK INC.

Attributes :

- **ID** : Unique Identifier / Primary Key for each company. It is autogenerated on addition to the database
- **company_name** : Name of the company, this is stored on the database for when the website needs to display the name of the company.

Location :

Each company can have multiple locations. Locations are determined by the company by whichever standards they themselves choose and are differentiated colloquially using the campus name, and identified in the database with ID, the primary key. The company has 4 locations. This allows different companies to have campus names that are the same. The structure and an over view of the data in the Location Table in the database are displayed in the side, followed by an explanation of each attribute:

Table Structure for Table Location

Column	Type	Null	Default
ID	int(11)	No	
campus_name	varchar(30)	Yes	NULL
company_id	int(11)	Yes	NULL
city	varchar(30)	Yes	NULL
country	varchar(30)	Yes	NULL
street_address	varchar(30)	Yes	NULL
postal_code	varchar(30)	Yes	NULL

Data for Table Location

ID	Campus	Company_Id	city	country	Street_address	Postal_code
1	HEAD QUARTERS	1	MENLO PARK	UNITED STATES OF AMERICA	1 HACKER WAY	94025
2	MARS DISCOVERY DISTRICT	1	TORONTO	CANADA	661 University Ave #1201	M5G 1M1
3	ROTHSCHILD BOULEVARD	1	TEL AVIV	ISRAEL	22 ROTHSCHILD BOULEVARD	281651
4	INTERNET CITY	1	DUBAI	UNITED ARAB EMIRATES	SHEIKH ZAYED ROAD	53777

Attributres :

- **ID:** Unique Identifier / Primary Key for each location. It is autogenerated on addition to the database
- **Campus_name:** The name by which employees refer to the different locations of the same company
- **Company_id :** This is a foreign key to the company the campus_name/location belongs to
- **City :** the city to which a given location belongs to
- **Country:** the country to which a given location belongs to
- **Street_address:** the address of the location in its given city and country
- **Postal_code :** the postal code of the location

Department:

Each company can have multiple locations. A location is not relation to a department. A department can exist across multiple locations and a location can have multiple departments. The company has 15 departments. A quick summary of the department table structure and a snap-shot of the data are given below, followed by an explanation of the attributes.

Table Structure for Table: Department

Column	Type	Null	Default
ID	int (11)	No	
company_id	int (11)	Yes	NULL
department name	varchar(30)	Yes	NULL

Data for Table Department

ID	Company_ID	Department Name
1	1	EXECUTIVE BOARD
2	1	OPERATIONS
3	1	CORPORATE STRATEGY
4	1	RESEARCH AND DEVELOPMENT

Attributes:

- **ID:** Unique Identifier / Primary Key for each department. It is autogenerated on addition to the database
- **Company_id:** This is a foreign key to the company the department belongs to

- **Department_name:** Name employees in the company will identify the department with

Employee

An employee belongs to a given location and given department and has several of their own attributes. There are a total of 501 employees given in the database, spread across the 15 departments. A quick summary of the employee table is given below, and is followed immediately by an explanation of the attributes:

Data for Table: Employee

ID	Job Title	First Name	Last Name	Location_id	Dep ID	Supervisor ID
1	CHIEF EXECUTIVE OFFICER	MARK	ZUCKERBERG	1	1	NULL
2	CHIEF OPERATIONS OFFICER	SHERYL	SANDBERG	1	1	1
8	BACK END DEVELOPER	AYSER	CHOUDHURY	1	1	1
9	FULL STACK DEVELOPER	VICTORIA	LI	1	1	1
10	PROGRAMMER	CINDY	KLEIN	1	1	1
11	PRODUCT MANAGER	AARON	YIM	1	1	1
18	CO-OP/INTERN	ABEL	AAMODT	1	1	1
19	MANAGER	ABIGAIL	AAMOT	3	9	1
247	DIRECTOR	ANGELLA	ACEITUNO	2	8	133
266	VICE PRESIDENT	ANJELICA	ACHENBACH	3	10	1
285	PRESIDENT	ANNIE	ACKERLY	2	4	1
501	CONTROLLER	TIMOTHY	PARK	1	1	7

Attributes:

- **ID:** Unique Identifier / Primary Key for each employee. It is autogenerated on addition to the database
- **Job Title:** Job position / role of the employee in the company
- **First name:** Given name of the employee

- **Last name:** surname of the employee
- **Location_id:** This is a foreign key to the location the employee belongs to
- **dep_id:** This is a foreign key to the department the employee belongs to
- **Supervisor_id:** This is a foreign key to the employee that the employee reports to

Account

Each user has an account. An account is created to separate sensitive information from readily available information regarding an employee. There is an account for every employee in the company, and no more or no less. A snapshot of the company's account table is given below, followed immediately by an explanation of the attributes.

Table Structure for Table

Column	Type	Null	Default
ID	int(11)	No	
username	varchar(30)	Yes	NULL
pass	varchar(30)	Yes	NULL
employee ID	int(11)	Yes	NULL

Account Data for Table: Account

ID	username	password	employee_id
1	MERG1	1ERG1M1	1
2	SERG2	2ERG2S2	2
3	DIRO3	3IRO3D3	3
4	JIKH4	4IKH4J4	4
5	MFER5	5FER5M5	5

Attributes:

- **ID :** Unique Identifier / Primary Key for each account. It is autogenerated on addition to the database
- **Username :** username of each employee, used for verification, security and logins. These are automatically generated based off of their first and last names, along with their employee id
- **password:** the password for the employee to log in to the "Task Trader" site. These are automatically generated.
- **Employee_id :** This is a foreign key for the account to refer to the corresponding employee

Status:

Status of tasks can be either “Available” or “Filled”. Status is kept separate from the Task Table in order to optimize space utility, as every task has a status and the status can only be one of two options. A snapshot of the Status table is given below followed immediately by the explanation of the attributes.

Table Structure for Table: Status_Type

Column	Type	Null	Default
ID	int(11)	No	
status_type	varchar(30)	Yes	NULL

Data for Table: Status_Type

ID	Status Type
1	AVAILABLE
2	FILLED

Attributes:

- **ID :** Unique Identifier / Primary Key for each status. It is autogenerated on addition to the database
- **Status_Type :** A task can only have either a status of “Available” or “Filled”

Task

A task is the main focus of this site. Tasks can be posted and applied to, and will eventually be filled. Whenever there is excess work in a department, or an lack of specific skills on a team, or a shortage of available man hours, a task can be posted in order to redistribute skills and time throughout the company. Applicants would want to take skills in order to improve their abilities in a given task, and in order to gain exposure to different departments and locations of the company. A brief snapshot of the Tasks table is given below, followed immediately by an explanation of the attributes.

Data for Table: Task

ID	Task Title	Task Description	End-Date	Start_Date	TimeComitment	Loc_id	Dep_id	Status_id
9	PHOTOSHOP HELP NEEDED	Lorem ipsum dolor sit amet, melius assueverit accommodare ius at, utinam	2018-06-12	2018-06-09	1	2	15	1
14	EXPERT HELP NEEDED IN PYTHON	Lorem ipsum dolor sit amet, melius assueverit accommodare ius at, utinam	2017-09-12	2017-07-31	1	4	15	1

Attributes:

- **ID :** Unique Identifier / Primary Key for each Task. It is autogenerated on addition to the database
- **Task_Title:** Quick title for a task, with minimal information regarding task, which should generally be sufficient as an overview of the task
- **Task Description :** A more detailed overview of the task. Should go more in-depth to the task, and answer any questions an applicant might have regarding the task.
- **End_date:** Expected date the task needs to be finished
- **Start_date:** Expected start_date of the task
- **Time_Commitment:** Weekly time a single individual will need to commit to the task to finish the task in-between the start date and end date.
- **Loc_id:** This is a foreign key to the location the task belongs to
- **Dep_id:** This is a foreign key to the department the task belongs to
- **Status_id :** This is a foreign key to the status indicating whether the status is active or filled

CV

The “Resume” section has been renamed to CV, as “resume” is an inbuilt function in SQL, and there were some errors with creating the table “Resume” through SQL. The

purpose of this table is store the resume of any of the employees who wish to upload them. It exists as a separate table, in order to optimize the speed and memory complexity of storing the files and the related queries that would be involved with the table. An explanation of the attributes for the CV table is given below:

Attributes:

- **ID:** Unique Identifier / Primary Key for each CV. It is autogenerated on addition to the database
- **Employee_ID:** This is a foreign key to the employee the cv belongs to
- **CV:** This is of type “Blob” and is used to save resumes.

Picture

Each employee is allowed to have a profile picture on their profile. This table is used to store the profile pictures of each employee. Similar to the CV table, this table is kept separately in order to optimize space and time complexity of searches. An explanation of the attributes of the picture table is given below:

Attributes:

- **ID:** Unique Identifier / Primary Key for each picture. It is autogenerated on addition to the database
- **Account_ID:** This is a foreign key to the employee’s account, the picture belongs to
- **Picture:** This is of type “Blob” and is used to save pictures
- **Image_type:** This attribute stores the file type of the image (.jpeg etc)

Skill

Each employee and task can have multiple skills. For an employee, these represent skills that they have, and for a task, these represent skills that are required to complete the task. Since skills are very often repeated between employees and tasks, in order to optimize searching and storing skills, they are stored in a different table. There are two accompanying tables called “Task Skills” and “Employee Skills” which are used alongside the skills table for the optimization. Below is a snapshot of the data in the skills table, followed by an explanation of each of the attributes:

Table Structure for Table:

Column	Type	Null	Default
ID	int(11)	No	
skill_name	varchar(30)	Yes	NULL

Skill Data for Table: Skill

ID	Skill_Name
1	Python
2	VBA
6	PHOTOGRAPHY
7	PHOTOSHOP
8	EXCEL
9	POWERPOINT
10	EXCEL
38	SSH
39	SAP

Attributes:

- ID: Unique Identifier / Primary Key for each Skill. It is autogenerated on addition to the database

- **Skill_Name:** This is the skill that is to be added to the database, and can be related both to multiple employees and multiple tasks.

Task Skills

This is a table that ties in the relationship between tasks and skills. It consists of two columns of task and skill ids. A task id can be repeated multiple times, but a skill can be tied to a task id once and only once. A skill_id can be repeated multiple times throughout the table for different tasks. A snapshot of the information for the first tasks are given to the side, followed by an explanation of the attributes:

Table Structure for Table: Task_Skills

Column	Type	Null	Default
task_id	int(11)	Yes	NULL
skill_id	int(11)	Yes	NULL

Data for Table: Task_Skills

Task_ID	Skill_ID
1	29
1	39
2	16
2	32
3	17
3	42

Attributes:

- **Task_ID:** This is a foreign key to the task that the the skill refers to.
- **Skill_ID:** This is a foreign key to the skill that is being assigned to the given task.

Employee Skills

This is a table that ties in the relationship between employees and skills. It consists of two columns of employee and skill ids. An employee id can be repeated multiple times, but a skill can be tied to an employee once and only once. A skill_id can be repeated multiple times throughout the table for different employees. A snapshot of the information for the first three employees are given to the side, followed by an explanation of the attributes:

Table Structure for Table: Employee_Skills

Column	Type	Null	Default
employee_id	int(11)	Yes	NULL
skills_id	int(11)	Yes	NULL

Data for table Employee_Skills

Employee_ID	Skills_ID
1	16
1	17
1	29
1	32
1	39
2	1
2	3
2	39
2	42
2	44
3	21
3	23
3	43
3	47
3	52

Attributes:

- **Employee_ID:** This is a foreign key to the employee that the the skill refers to.
- **Skill_ID:** This is a foreign key to the skill that is being assigned to the given employee.

Posted Task

A task can be posted by one and only one employee. This is the employee who is in charge of finding someone to complete the task. They can choose to post a task themselves, or can be someone who is part of a larger team or project who was asked to post the task on behalf of the project or the team. This employee is the one who must give the final selection of which applicant receives the task, and it is their responsibility to reach out to the employee through email to begin the next steps. A small snapshot of the first 6 tasks posted, and employee who posted the task, or rather the relationship

between task and task poster can be viewed below, followed immediately by an explanation of the attributes of the table.

Table Structure for Table: Posted_Task

Column	Type	Null	Default
task_id	int(11)	Yes	NULL
employee_id	int(11)	Yes	NULL

Data for Table: Posted_Task

Task_ID	Employee_ID
1	353
2	267
3	290
4	145
5	151
6	388
7	8
8	381
9	408
10	355

Attributes:

- **Task_ID:** This is a foreign key to the task that is being posted by an employee.
- **Employee_ID:** This is a foreign key to the employee who is posting the given task

Applied to Task

Multiple employees across any of the company's different locations and different departments can apply for a task without any restrictions. After reading a job description, an employee can quickly apply and be put into consideration for that task. When the poster is looking at the list of applicants for a given job, they can select an applicant and view their profile. An extract of a few rows of data from the applied to task table is given on the next page, preceded by an explanation of its attributes:

Attributes:

- **Task_ID:** This is a foreign key to the task that an employee is applying to
- **Employee_ID:** This is a foreign key to the employee who is applying to a task

.

Table Structure for Table: Applied_to_Task

Column	Type	Null	Default
task_id	int(11)	Yes	NULL
employee_id	int(11)	Yes	NULL

Data for Table: Applied_to_Task

Task_ID	Employee_ID
1	332
1	382
1	282
2	263
2	313
2	213
3	282
3	332
3	232

Filled Task

After an employee is selected for a task, the task status is changed to “Filled” and and the employee who is filling task is saved in the “Filled Task” table. It is the duty of the employee who filled the task and the employee who posted the task to connect via internal communications in the company. An extract from a few rows of data from the applied to task table is given below, and is followed by an explanation of its attributes.

Table structure for table Filled_Tasks

Column	Type	Null	Default
task_id	int(11)	Yes	NULL
employee_id	int (11)	Yes	NULL

Data for table Filled_Tasks

Task_id	Employee_ID
6	353
7	267
12	290
19	145
21	151
26	388
27	8
2122	473
2124	327
2126	8
2128	331
2129	483

Attributes:

- **Task_ID:** This is a foreign key to the task that an employee is going to do
- **Employee_ID:** This is a foreign key to the employee who will be doing the task

How Data is Collected Normally

Data for “Task Trader” comes from a handful of sources. Majority of the data is collected from user input, however, some data is pre-loaded onto the site. First, let us define the two different types of users:

1. **Site Creators:** Some information is integral to the operations of the inner mechanisms of the site. This type of information is created by the site creators
2. **Regular Employee**: These are the majority of users of the Task Trader service. They are the users the site, along with the user interface is made for.
3. **Site Administrators:** These are the users who are in charge of the site maintenance. They are the “business owners” of the site and are responsible for the site.

All data on the site is created from a combination of these three users. To better understand the origin of the data, below is an explanation of how each table of data is created, and by which user.

1. Data Managed by Site Creators:

Some information is critical to the operation of the website, and needs to be created before the website can be launched at any given company. This sort of information is managed by the creators of the website

A. Status_Type:

Status_Types are created with the creation of the site. They are not inserted, updated or deleted.

B. Company:

Companies are created with the creation of the site at a given company. Creators insert. update or delete companies on an as needed basis.

2. Data Managed by Regular Employees:

This data is data that is created, updated and deleted through the usage of the site. The data is created by employees, managed by employees, deleted and updated by employees through the website.

A. Tasks:

Employees post tasks; The details of each individual task is created by an employee, an employee can choose to update information regarding the task, or can even choose to delete a task if they so wish to do so.

B. Posted Tasks:

When an employee creates a task, they automatically are create data for the posted tasks table. Their employee id and their new task's id are input into this table

C. Applied to Tasks:

When an employee decides to apply for a task, and clicks on "Apply" they're creating data for the "Applied_to_Tasks" Table. Their employee_ID, and the task ID of the task they are applying to get added to the table.

D. Filled Tasks:

When an employee decides who is going to be taking care of a task, they assign the task to that person, and add data to the "Filled_Tasks" Table. The employee id of the employee selected, and the related task are added to the table.

E. Skills:

When a skill is added to an employee's skill list or a task's skill list, it is first checked to see if the skill exists in the skill table. If it doesn't it is added ad a new skill

F. Employee_Skills:

An employee adds and deletes their skills whenever they wish to do so. They cannot be updated

G. Task_Skills:

An Employee inserts information into the task skills while creating the task. They can delete the information when they choose to edit the skills related to the task. Information on this table cannot be edited.

H. CV:

An employee uploads a CV whenever they wish to do so. They can delete a CV as well whenever they wish to do so. However, a CV's information can be updated or edited.

I. Picture:
   A. An employee can upload an image to their account whenever they wish to do so. They can delete an image as well whenever they wish to do so. However, a image's information can be updated or edited.
   B.

3. <u>Data Managed by Site Administrators:</u>

This data involves everything that is relatively fixed over a course of months and years. The tables of information created, updated and deleted by these users are:

A. Employees:

Employee information is created when an employee joins the company, or when the the site is first launched within a company. Changes to an employee's information is very rare, but when they do occur, they are carried out by site administrators. When an employee leaves the company, they are deleted from the employee database

B. Accounts:

Accounts are created alongside the employees. An employee's username and password are automatically generated on their creation. However, an employee's account's password can be updated by the employee. An account is deleted when an employee leaves the company.

C. Departments:

Departments are created, updated, and deleted by site administrators on an as needed basis. Typically this is a complicated process that involves reassigning employees to different departments, but this is not a frequent occurrence.

D. Locations:

Locations are created, updated, and deleted by site administrators on an as needed basis. Typically this is a complicated process that involves reassigning employees to different locations, but this is not a frequent occurrence.

How Data Was Created for The Project

Tables such as picture, CV, status_type, departments, skills and locations were created manually in a way that would make sense in the context of this project. Other information, such as information regarding employees, tasks, and the relations that tie into employees and tasks were all generated using either VBA code or Python Code. A sample of VBA code used to create “Task” job titles and “Employee Information” is given below. Employee’s names were taken from an online database, a few employees including the names of those involved in this project, the executive board of the company that was focused on as well as the persona used in the presentation were manually added.

Example of VBA Code used for Creating Employee Information

```
Sub Button2_Click()

    Dim manager_count As Integer
    Dim director_count As Integer
    Dim VP_Count As Integer
    Dim prez_count As Integer
    Dim manager_val As Integer
    Dim director_val As Integer
    Dim VP_val As Integer
    Dim prez_val As Integer
    Dim dep_id As Integer
    Dim loc_id As Integer
```

```
    loc_id = 1
    dep_id = 1
    manager_val = -50000
    director_val = -50000
    VP_val = -50000
    prez_val = -50000

    For i = 8 To 1000
        If i Mod 19 = 0 Then
           If VP_Count < 2 Then
               If director_count < 2 Then
                    If manager_count < 2 Then
                        manager_count = manager_count + 1
                        Sheet1.Cells(i, 2).Value = "MANAGER"
                        Sheet1.Cells(i, 7).Value = director_val
                        manager_val = i
                        loc_id = Int((4 - 1 + 1) * Rnd + 1)
                        dep_id = Int((15 - 1 + 1) * Rnd + 1)
                        Sheet1.Cells(i, 6).Value = dep_id
                        Sheet1.Cells(i, 5).Value = loc_id

                    Else
                        manager_count = 0
                        Sheet1.Cells(i, 2).Value = "DIRECTOR"
                        Sheet1.Cells(i, 7).Value = VP_val
                        Sheet1.Cells(i, 6).Value = Int((15 - 1 + 1) *
Rnd + 1)
                        Sheet1.Cells(i, 5).Value = Int((4 - 1 + 1) *
Rnd + 1)

                        director_val = i
                        director_count = director_count + 1
                    End If

               Else
                   director_count = 0
                   Sheet1.Cells(i, 2).Value = "VICE PRESIDENT"
                   Sheet1.Cells(i, 7).Value = prez_val
                   Sheet1.Cells(i, 5).Value = Int((4 - 1 + 1) * Rnd +
1)
                   Sheet1.Cells(i, 6).Value = Int((15 - 1 + 1) * Rnd +
1)

                   VP_val = i
                   VP_Count = VP_Count + 1
               End If

           Else
                VP_Count = 0
```

```
            Sheet1.Cells(i, 2).Value = "PRESIDENT"
            Sheet1.Cells(i, 7).Value = -50000
            Sheet1.Cells(i, 6).Value = Int((15 - 1 + 1) * Rnd + 1)
            Sheet1.Cells(i, 5).Value = Int((4 - 1 + 1) * Rnd + 1)
            prez_val = i
        End If

    ElseIf i Mod 19 = 1 Then
        Sheet1.Cells(i, 2).Value = "FULL STACK DEVELOPER"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 2 Then
        Sheet1.Cells(i, 2).Value = "JUNIOR ANALYST"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 3 Then
        Sheet1.Cells(i, 2).Value = "DESIGNER"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 4 Then
        Sheet1.Cells(i, 2).Value = "ENGINEER"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 5 Then
        Sheet1.Cells(i, 2).Value = "SENIOR ANALYST"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 6 Then
        Sheet1.Cells(i, 2).Value = "ENGINEER"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id

    ElseIf i Mod 19 = 7 Then
        Sheet1.Cells(i, 2).Value = "FRONT END DEVELOPER"
        Sheet1.Cells(i, 7).Value = manager_val
        Sheet1.Cells(i, 6).Value = dep_id
        Sheet1.Cells(i, 5).Value = loc_id
```

```
ElseIf i Mod 19 = 8 Then
    Sheet1.Cells(i, 2).Value = "BACK END DEVELOPER"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 9 Then
    Sheet1.Cells(i, 2).Value = "FULL STACK DEVELOPER"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 10 Then
    Sheet1.Cells(i, 2).Value = "PROGRAMMER"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 11 Then
    Sheet1.Cells(i, 2).Value = "PRODUCT MANAGER"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 12 Then
    Sheet1.Cells(i, 2).Value = "PROJECT COORDINATOR"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 13 Then
    Sheet1.Cells(i, 2).Value = "DEPARTMENT SUPPORT"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 14 Then
    Sheet1.Cells(i, 2).Value = "JUNIOR ANALYST"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 15 Then
    Sheet1.Cells(i, 2).Value = "ANALYST"
    Sheet1.Cells(i, 7).Value = manager_val
    Sheet1.Cells(i, 6).Value = dep_id
    Sheet1.Cells(i, 5).Value = loc_id

ElseIf i Mod 19 = 16 Then
```

```
            Sheet1.Cells(i, 2).Value = "QUALITY ASSURANCE"
            Sheet1.Cells(i, 7).Value = manager_val
            Sheet1.Cells(i, 6).Value = dep_id
            Sheet1.Cells(i, 5).Value = loc_id

        ElseIf i Mod 19 = 17 Then
            Sheet1.Cells(i, 2).Value = "PRODUCT DEVELOPMENT"
            Sheet1.Cells(i, 7).Value = manager_val
            Sheet1.Cells(i, 6).Value = dep_id
            Sheet1.Cells(i, 5).Value = loc_id
        ElseIf i Mod 19 = 18 Then
            Sheet1.Cells(i, 2).Value = "CO-OP/INTERN"
            Sheet1.Cells(i, 7).Value = manager_val
            Sheet1.Cells(i, 6).Value = dep_id
            Sheet1.Cells(i, 5).Value = loc_id
        End If
    Next i
End Sub
```

Example of Python Code used to Create Task - Job Titles

```
import csv
from random import randint


programs =
['PYTHON','SQL','R','PHOTOSHOP','VBA','JAVA','C#','PHOTOGRAPHY','EXCEL
',
            'POWERPOINT','PREZI','PHOTO-EDITING','VIDEO-
EDITING','PAINTING','FINANCIAL ANALYSIS','PROGRAMMING','ALGORITHMS',
            'SUMMARIZATION','PROOF READING']

PREFIX = ['HELP WITH ', 'NEED EXPERT IN  ',' URGENT HELP IN  ','EXPERT
HELP NEEDED IN ','ASSISTANCE REQUIRED WITH ','GUIDANCE NEEDED WITH
','LARGE PROJECT : NEED HELP WITH ']
SUFFIX = [' HELP NEEDED',' TASK NEEDED FOR DEPARTMENT PROJECT',' TASK
AVAILABLE',' TUTOROIALS REQUIRED']
DEPARTMENT = ['CORPORTATE STRATEGY PROJECT','PRODUCT DEVELOPMENT
PROJECT','MARKETING PROJECT','ADVERTISING PROJECT','INSTAGRAM
PROJECT','WHATSAPP PROJECT','OCULUS PROJECT']
res = []

for item in programs:
    for pre in PREFIX:
        res.append(pre+item)
    for suf in SUFFIX:
        res.append(item+suf)
    for dep in DEPARTMENT:
        res.append(dep)
```

```
res1 = []

# making list bigger
print('expanding list...')
for item in res :
    for i in range(30):
        res1.append(item)

# shuffling data
print('shuffling...')
for i in range(1000000):
    a = randint(0, len(res1)-1)
    b = randint(0,len(res1)-1)
    first =''
    second =''
    first = res1[a]
    second = res1[b]
    res1[b] = first
    res1[a] = second

print('saving to csv...')

csvfile = "/Users/ayser/Desktop/w.csv"
with open(csvfile,"w") as f:
    writer = csv.writer(f)
    for row in res1:
        writer.writerow([row])
```

Queries

Explore Page

1. Recently Posted
    a. What this does:
        i. Returns a list of unfilled postings that have a deadline in the future, their task titles, and the number of people who have already applied
        ii. Entities are sorted by ascending deadline date, which is 7 days before the start of the task
        iii. A maximum of 4 postings are displayed
    b. Why we've included it:
        i. To demonstrate JOIN, GROUP BY, HAVING, and COUNT
        ii. To demonstrate use of MySQL specific DATE_ADD functionality
    c. How:
        i. SELECT T.task_title, DATE_ADD(T.start_date, INTERVAL -7 DAY) AS deadline, COUNT(A.employee_id) AS applications, T.start_date FROM Task T LEFT JOIN Applied_to_Task A ON T.ID = A.task_id WHERE T.status_id = 1 AND DATE_ADD(T.start_date, INTERVAL -7 DAY) > CURRENT_TIMESTAMP GROUP BY T.id ORDER BY T.start_date ASC LIMIT 4
2. Recommended For You
    a. What this does:
        i. Compares an employee's list of skills to the skills required for a task
        ii. Postings are sorted by the highest % match in skillset, and then by the deadline
        iii. Displays unfilled postings' task title, deadline, number of current applications, and skill match percentage
    b. Why we've included it
        i. To demonstrate the joining of two select queries, which are then further queried
        ii. To demonstrate the use of math functions such as FLOOR to round
        iii. To demonstrate the use of GROUP BY, COUNT, DATE_ADD, INNER JOIN, LEFT JOIN, etc.
        iv. To demonstrate ORDER BY with two parameters
    c. How:

```
SELECT M.task_title, DATE_ADD(M.start_date, INTERVAL -7 DAY) AS
deadline, FLOOR((M.skill_match*100/R.required_skill_ct)) AS match_out,
COUNT(A.employee_id) AS applications
FROM (SELECT T.ID, T.task_title, T.start_date, T.status_id, COUNT(skill_id) AS
skill_match
        FROM Task T LEFT JOIN Task_Skills TS on T.id = TS.task_id
        INNER JOIN Employee_Skills ES ON TS.skill_id = ES.skills_id
WHERE ES.employee_id = 10
        GROUP BY ID) M
LEFT JOIN (SELECT T.ID, COUNT(TS.skill_id) AS required_skill_ct
            FROM Task T
            LEFT JOIN Task_Skills TS
            on T.id = TS.task_id GROUP BY T.ID) R
ON M.ID = R.ID
LEFT JOIN Applied_to_Task A ON M.ID = A.task_id
WHERE M.status_id = 1 AND DATE_ADD(M.start_date, INTERVAL -7 DAY) >
CURRENT_TIMESTAMP GROUP BY M.ID ORDER BY (M.skill_match/
R.required_skill_ct) DESC, M.start_date ASC LIMIT 4
```

3. <u>At Your Location</u>
    a. What this does:
        i. Returns a list of unfilled postings that have a deadline in the future, their task titles, and the number of people who have already applied
        ii. Entities are sorted by ascending deadline date, which is 7 days before the start of the task
        iii. A maximum of 4 postings are displayed
        iv. BASICALLY THE SAME AS #1 BUT WITH THE LOCATION CLAUSE ADDED
    b. Why we've included it:
        i. To demonstrate JOIN, GROUP BY, HAVING, and COUNT
        ii. To demonstrate use of MySQL specific DATE_ADD functionality
    c. How:
        i. SELECT T.task_title, DATE_ADD(T.start_date, INTERVAL -7 DAY) AS deadline, COUNT(A.employee_id) AS applications, T.start_date FROM Task T LEFT JOIN Applied_to_Task A ON T.ID = A.task_id WHERE T.status_id = 1

AND DATE_ADD(T.start_date, INTERVAL -7 DAY) > CURRENT_TIMESTAMP
AND T.location_id = 1 GROUP BY T.id ORDER BY T.start_date ASC LIMIT 4

Analytics

- **# of Jobs Available:** SELECT COUNT(*) FROM Task where Task.start_date > CURRENT_TIMESTAMP
    - What: display # of jobs available
    - Why: aggregate function
- # of jobs applied to
- # of people applied to their postings
- Incoming/outgoing
    - Department
    - Location