

Project Report

Employee Rating Application

Group 4 Team Members:

Mayur Mahanta

(002927614)

Moheth Muralidharan

(002199070)

Executive Summary

The project's goal is to design and build a relational database for a company called, say X. Employees across the company frequently complained about a lack of transparency in the evaluation of their performances. HR employees, on the other hand, complained about the time-consuming and manual process of performance input and evaluation. Managers and other high-level executives are unable to find data and analytics related to performance and, as a result, are unable to make data-driven decisions for employees, such as how to improve overall employee performance.

This relational database shortens the data input process by 61% and eliminates the traditional manual process. The database also includes a Python-based analytics solution with the potential for analytics to monitor trends and make employee-based decisions.

The database modeling was completed by gathering requirements from HR employees as well as input from high-level executives on the type of data analytics they would like to see. The EER and UML diagrams were modeled, and then the conceptual model was mapped to a relational model with the necessary primary and foreign keys. This database was then implemented in MySQL using the SQL Workbench 8.0 software, with a portion of it implemented in NoSQL using the MongoDB Shell Script.

The databases were successfully created and connections to Python were implemented to enhance the analytics capabilities. Employees and management alike were ecstatic to have access to the data, which aided the organization in making data-driven decisions.

1. Introduction

Business Problem:

A Manufacturing Company is planning to develop a new database system that will support its new employee rating application. The database is designed particularly to decrease biasedness and increase transparency across the company. The rating of an employee will be done on a pre-defined set of scores that will be utilized to determine their individual input percentage and the efficiency of their work deliverance. The rating will be done by a group of reviewers for all the employees based on their onsite performance. The final score will depend on multiple ratings and keeping in mind the transparency factor, maybe/will be viewed by everyone in the company.

There are a standard set of rules that needs to be followed for the designed model for the above-specified application. They are as follows -

- Each employee will need to finish their work.
- The employee will have one designation. Under certain conditions, an employee can have more than one designation.
- Every employee will belong to a department.
- The employee will be rated by the group of reviewers with scores.
- All work will be rated by reviewers individually.
- The reviewer can post the scores for every work he participated in.
- The scores can be viewed by the employees.

Design:

The design constraints or the entities for the above Employee Rating Application System are as follows –

- Employee
- Designation
- Work
- Department
- Scores
- Reviewer

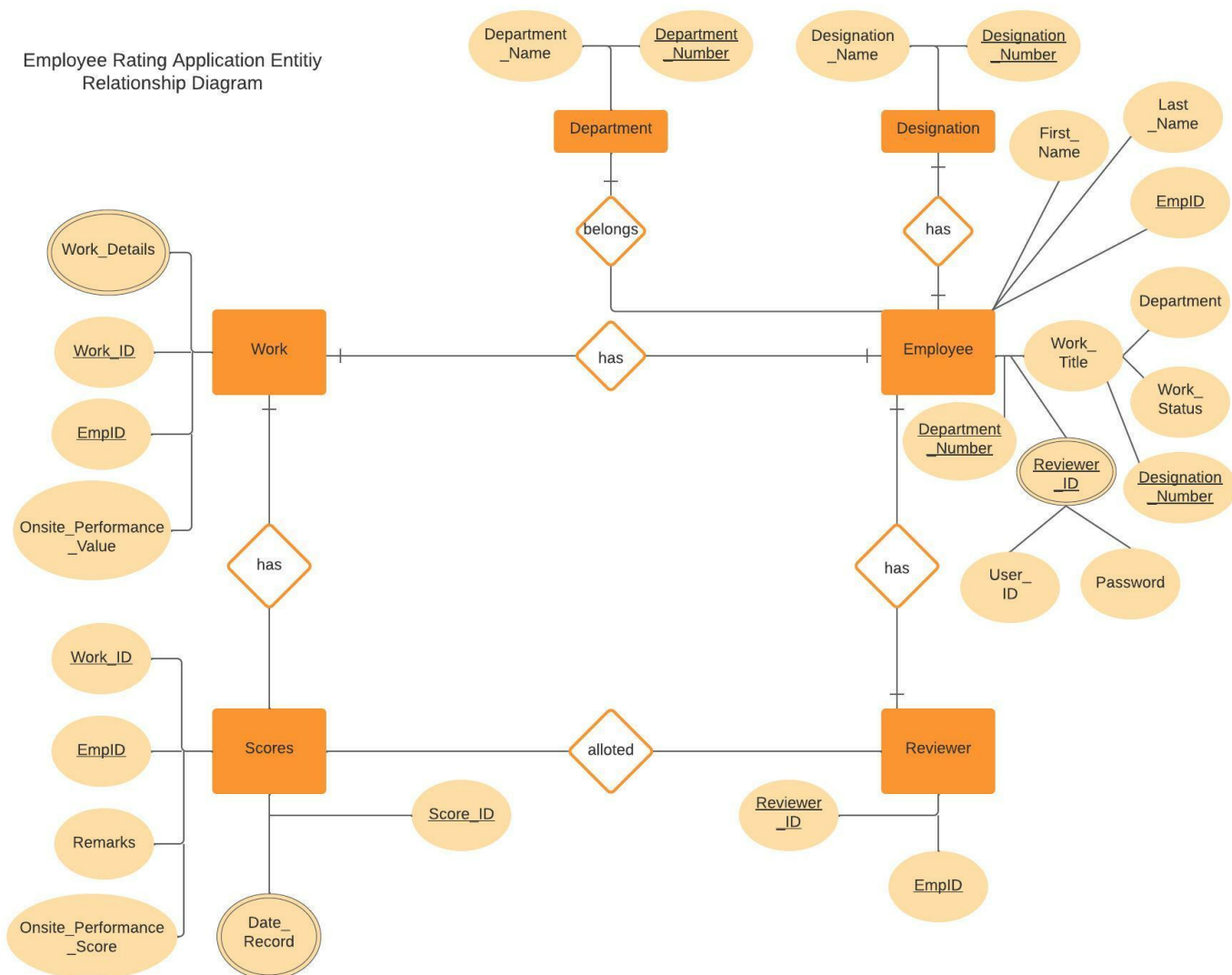
The above-given design constraints or entities will generate the tables in our database according to the design and implementation of the database schema. These tables are formed to meet the requirements and the standard set of rules defined above for our database. Firstly, the EMPLOYEE table will contain all concerned details along with its designation and department. Secondly, the DESIGNATION and DEPARTMENT tables will have a set of details that are required to review and distinguish by the reviewers while scoring. However, this set of details will not have an impact on determining the onsite performance in order to keep the transparency and unbiasedness factor intact across the company. Next, all the employees/workers will have been assigned work and the evaluation of their onsite performance will be done on the basis of these works. For the evaluation to take place for each individual employee and be added for scoring, they will be required to finish their work. Thirdly the WORK table will also contain a set of details relevant to the table which will be later used by the reviewers while evaluating and scoring each individual employee. Again, the REVIEWER table will comprise information related to the reviewer. There will be a group of reviewers who will be rating and scoring employee/work. A reviewer will rate the employee/work by allotting scores. Lastly, the SCORES table will give the ratings and evaluate each employee's onsite performance. As specified above under the standard set of rules, the scores may/can be accessed/viewed by each employee in the company.

Now, the attributes for the above listed Entities are given below –

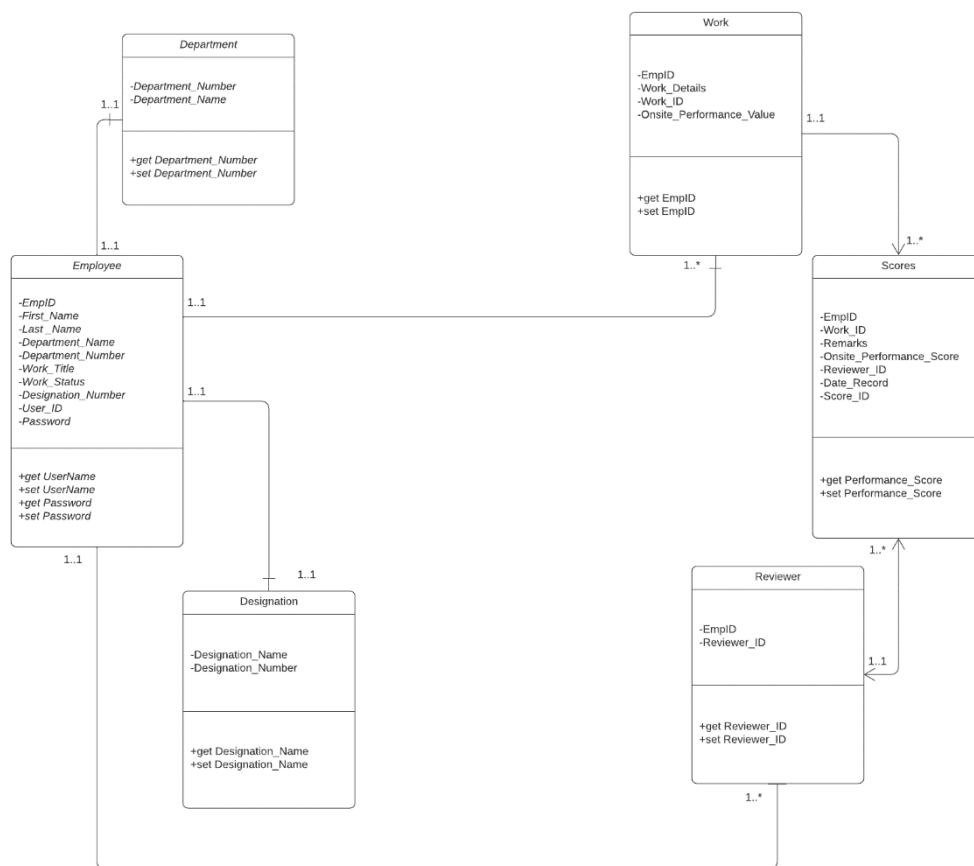
- **Employee** (EmpID, First_Name, Last_Name, User_ID, Password, Designation_Name, Designation_Number, Department_Name, Department_Number, Work_Title, Work_Status, Reviewer_ID)
- **Designation** (Designation_Name, Designation_Number)
- **Work** (Work_ID, Work_Details, EmpID, Onsite_Performance_Value)
- **Department** (Department_Name, Department_Number)
- **Scores** (Score_ID, EmpID, Work_ID, Reviewer_ID, Onsite_Performance_Score, Remarks, Date_Record)
- **Reviewer** (Reviewer_ID, EmpID)

2. (i) Conceptual Data Modeling

A) Enhanced Entity Relationship Diagram:



B) Unified Modeling Language (UML) Class Diagram:



Employee Performance UML Diagram

2. (ii) Mapping Conceptual Model to Relational Model

A) Employee (EmpID, First_Name, Last_Name, User_ID, Password, Designation_Name, Designation_Number, Department_Name, Department_Number, Work_Title, Work_Status, Reviewer_ID)

Employee: EmpID in relation to Employee: NULL not allowed, on delete/update cascade.

B) Designation (Designation_Name, Designation Number)

Here, Designation_Number is the primary key.

C) Department (Department_Name, Department Number)

Here, Department_Number is the primary key.

D) Work (Work_ID, Work_Details, EmpID, Onsite_Performance_Value)

Here, Work_ID is the primary key.

Employee: EmpID in relation to Employee: NULL not allowed, on delete/update cascade.

E) Reviewer (Reviewer_ID, EmpID)

Here, Reviewer_ID is the primary key.

Employee: EmpID in relation to Employee: NULL not allowed, on delete/update cascade.

F) Scores (Score_ID, EmpID, Work_ID, Reviewer_ID, Onsite_Performance_Score, Remarks, Date_Record)

Here, Score_ID is the primary key.

Employee: EmpID in relation to Employee: NULL not allowed, on delete/update cascade. Reviewer: Reviewer_ID in relation to Reviewer: NULL not allowed, on delete/update cascade Work: Work_ID in relation to Work: NULL not allowed, on delete/update cascade.

3. SQL Implementation

The database was created in MySQL with the schema name "*employeeeratingapplication*" and the following queries were executed using MySQL Workbench 8.0.

Query 1:

- List of Employees with Designation as "Senior Financial Analyst"

```
select * from employeeeratingapplication.employee
where Designation_Name="Senior Financial Analyst"
```

Output 1:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | EmpID | First_Name | Last_Name | User_ID | Password | Designation_Name | Designation_Number | Department_Name | Department_Number | Work_Title |
|--|-------|------------|-----------|---------|------------|--------------------------|--------------------|--------------------------|-------------------|------------|
| | 3 | Corella | Playfoot | 2696 | KfszSgimEx | Senior Financial Analyst | 27355 | Business Development | 35 | Director |
| | 41 | Gothart | O'Tierney | 3119 | SyLmRtg | Senior Financial Analyst | 16990 | Research and Development | 92 | Advisor |
| | 72 | Charlotte | Smewing | 9559 | DJNxh6io | Senior Financial Analyst | 65083 | Human Resources | 95 | Marketer |

Query 2:

- List Employees who are "Managers" (Aggregating)

```
select e.EmpID,
```

e.First_Name, e.Last_Name, w.Work_Details

from employee e, work w

where e.EmpID= w.EmpID

and w.Work_Details="Manager"

Output 2:

| | EmpID | First_Name | Last_Name | Work_Details |
|---|-------|------------|--------------|--------------|
| ▶ | 7 | Maegan | Tollerfield | Manager |
| | 12 | Melonie | Fitzackerley | Manager |
| | 20 | Karoly | Bech | Manager |
| | 27 | Fina | Kyberd | Manager |
| | 58 | Bertha | Darby | Manager |
| | 67 | Cher | Mibourne | Manager |
| | 69 | Marris | Mercik | Manager |
| | 73 | Suzanne | Lackner | Manager |
| | 89 | Breena | Yurocjhin | Manager |
| | 94 | Analise | Dungee | Manager |

Query 3:

- To find the minimum and maximum performance score of all employees (using statistical functions like min and max)

select min(Onsite_Performance_Score) as min_score,

max(Onsite_Performance_Score) as max_score

from employeeratingapplication.scores

Output 3:

| | min_score | max_score |
|---|-----------|-----------|
| ▶ | 10 | 47 |

Query 4:

- List of employees who are also reviewers (Aggregating)

select e.EmpID, e.Work_Title, e.First_Name, e.Last_Name, r.Reviewer_ID

from employee e, reviewer r

where e.EmpID=r.EmpID

Output 4:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | EmpID | Work_Title | First_Name | Last_Name | Reviewer_ID |
|--|-------|------------|------------|--------------|-------------|
| | 3 | Director | Corella | Playfoot | 1 |
| | 4 | Marketer | Amalia | Probey | 13 |
| | 5 | Director | Stephanus | Mogridge | 5 |
| | 16 | Engineer | Cris | Van der Beek | 15 |
| | 19 | Intern | Desirae | Corley | 17 |
| | 23 | Advisor | Bronnie | Moseley | 19 |
| | 23 | Advisor | Bronnie | Moseley | 8 |
| | 30 | Engineer | Ebenezer | Kitlee | 2 |
| | 34 | Marketer | Kaylyn | Roisen | 4 |
| | 39 | Typist | Genia | Coombe | 10 |
| | 40 | Director | Reena | Aldrich | 18 |
| | 41 | Advisor | Gothart | O'Tierney | 14 |
| | 45 | Human R... | Liane | Babinski | 7 |
| | 63 | Director | Adolf | Avieson | 20 |
| | 72 | Marketer | Charlotte | Smewing | 16 |
| | 72 | Marketer | Charlotte | Smewing | 6 |
| | 79 | Marketer | Jobi | Chupin | 11 |
| | 84 | Clerk | Domeniga | Howchin | 3 |
| | 85 | Advisor | Rosco | Airlie | 12 |
| | 93 | Clerk | Casi | Shasnan | 9 |

Query 5:

- Number of employees assigned to each reviewer (Reviewer_ID) for reviewing using (GROUP BY Function)

```
select count(e.EmpID), r.Reviewer_ID
from employee e, reviewer r
where e.Reviewer_ID= r.Reviewer_ID
group by Reviewer_ID
```

Output 5:

Continued below -

| Result Grid | | | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|----------------|-------------|--------------|---------|--------------------|
| | count(e.EmpID) | Reviewer_ID | | | |
| ▶ | 9 | 4 | | | |
| | 8 | 7 | | | |
| | 4 | 17 | | | |
| | 4 | 11 | | | |
| | 2 | 19 | | | |
| | 4 | 5 | | | |
| | 5 | 6 | | | |
| | 7 | 13 | | | |
| | 5 | 3 | | | |
| | 8 | 12 | | | |
| | 3 | 1 | | | |
| | 4 | 2 | | | |
| | 2 | 8 | | | |
| | 5 | 10 | | | |
| | 10 | 15 | | | |
| | 5 | 20 | | | |
| | 4 | 14 | | | |
| | 5 | 18 | | | |
| | 4 | 16 | | | |
| | 2 | 9 | | | |

Query 6:

- Listing all Employees with a performance score of more than 23 (Aggregating and using ORDER BY Function)


```

select e.EmpID, e.Work_Title, e.First_Name, e.Last_Name,
s.Onsite_Performance_Score
from employee e, scores s
where e.EmpID= s.EmpID
having Onsite_Performance_Score>=23
order by Onsite_Performance_Score desc

```

Output 6:

Continued below-

| Result Grid | | | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|----------------|------------|--------------|--------------------------|
| Filter Rows: <input type="text"/> | | | | | |
| Export:  Wrap Cell Content:  | | | | | |
| | EmpID | Work_Title | First_Name | Last_Name | Onsite_Performance_Score |
| | 10 | Consultant | Sherm | Bartlomiej | 30 |
| | 23 | Advisor | Bronnie | Moseley | 30 |
| | 86 | Janitor | Isidoro | O'Hanlon | 29 |
| | 21 | Intern | Bunni | Duggen | 27 |
| | 32 | Engineer | Kerk | Ralling | 27 |
| | 58 | Manager | Bertha | Darby | 27 |
| | 61 | Attorney | Biddy | Halligan | 27 |
| | 31 | Marketer | Allan | Risbie | 26 |
| | 93 | Clerk | Casi | Shasnan | 26 |
| | 75 | Marketer | Gertruda | Hodgins | 26 |
| | 42 | Typist | Giuditta | Telezhkin | 26 |
| | 40 | Director | Reena | Aldrich | 26 |
| | 18 | Human Resource | Veronike | Nissle | 25 |
| | 6 | Secretary | Der | O'Loughnan | 24 |
| | 55 | Human Resource | Netta | Drewery | 24 |
| | 35 | Advisor | Kasper | Chansonne | 24 |
| | 11 | Typist | Al | Walkinshaw | 23 |
| | 20 | Manager | Karoly | Bech | 23 |
| | 54 | Human Resource | Amaleta | Wesley | 23 |
| | 12 | Manager | Melonie | Fitzackerley | 23 |
| | 49 | Engineer | Sunshine | Hadenton | 23 |

Query 7:

- Listing all Employees who work as "Secretary" and has a performance score more than 15 (Aggregating based on multiple conditions)

```
select e.EmpID, e.First_Name, e.Last_Name, w.Work_Details,
s.Onsite_Performance_Score
```

```
from employee e, work w , scores s
```



```
where e.EmpID=s.EmpID
```

```
and e.EmpID=w.EmpID
```

```
having w.Work_Details= "Secretary"
```

```
and s.Onsite_Performance_Score>15
```

Output 7:

| Result Grid | | | | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|------------|------------|--------------|--------------------------|
| Filter Rows: <input type="text"/> | | | | | |
| Export:  Wrap Cell Content:  | | | | | |
| | EmpID | First_Name | Last_Name | Work_Details | Onsite_Performance_Score |
| ▶ | 6 | Der | O'Loughnan | Secretary | 24 |
| | 53 | Far | Paxeford | Secretary | 18 |

Query 8:

- Listing the employees with Work_Title "Human Resource" or is a reviewer (using UNION Function)

```
select EmpID, First_Name, Last_Name
from employee
where Work_Title= "Human Resource"
union
select e.EmpID, e.First_Name, e.Last_Name
from employee e, reviewer r
where e.EmpID=r.EmpID
```

Output 8:

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|--------------|--------------|--------------------|
| EmpID | First_Name | Last_Name | |
| 9 | Alla | Kedie | |
| 18 | Veronike | Nissle | |
| 44 | Derk | McFaden | |
| 45 | Liane | Babinski | |
| 50 | Jyoti | De Santos | |
| 54 | Amaleta | Wesley | |
| 55 | Netta | Drewery | |
| 3 | Corella | Playfoot | |
| 4 | Amalia | Probey | |
| 5 | Stephanus | Mogridge | |
| 16 | Cris | Van der B... | |
| 19 | Desirae | Corley | |
| 23 | Bronnie | Moseley | |
| 30 | Ebenezer | Kitlee | |
| 34 | Kaylyn | Roisen | |
| 39 | Genia | Coombe | |
| 40 | Reena | Aldrich | |
| 41 | Gothart | O'Tierney | |
| 63 | Adolf | Avieson | |
| 72 | Charlotte | Smewing | |
| 79 | Jobi | Chupin | |

Query 9:

- Listing the employees who are not reviewers (using Nested Query)

```
select EmpID, First_Name, Last_Name
from employee
where EmpID not in (select EmpID from reviewer)
```

Output 9 (Continued below):

| Result Grid | | | |
|-------------|------------|--------------|--------------------|
| | | Filter Rows: | |
| | | Export: | Wrap Cell Content: |
| EmpID | First_Name | Last_Name | |
| 1 | Cornie | Strowther | |
| 2 | Franklyn | Ravelus | |
| 6 | Der | O'Loughnan | |
| 7 | Maegan | Tollerfield | |
| 8 | Rinaldo | Boule | |
| 9 | Alla | Kedie | |
| 10 | Sherm | Bartlomiej | |
| 11 | Al | Walkinshaw | |
| 12 | Melonie | Fitzackerley | |
| 13 | Lemuel | Tregenza | |
| 14 | Lamont | Noen | |
| 15 | Nolana | O'Growgane | |
| 17 | Lesya | Sibthorp | |
| 18 | Veronike | Nissle | |
| 20 | Karoly | Bech | |
| 21 | Bunni | Duggen | |
| 22 | Dian | Vanin | |
| 24 | Erv | Buckthorpe | |
| 25 | Mel | Briton | |
| 26 | Riane | Matis | |
| 27 | Fina | Kyberd | |

Query 10:

- Finding average rating with respect to the departments (using statistical function avg using fields from two tables)

```

select e.Department_Number, avg(s.Onsite_Performance_Score)
from employee e, scores s
where e.EmpID=s.EmpID
group by Department_Number

```

Output 10:

| Result Grid | | |
|-------------------|---------------------------------|--------------------|
| | | Filter Rows: |
| | | Export: |
| | | Wrap Cell Content: |
| Department_Number | avg(s.Onsite_Performance_Score) | |
| 100 | 11.0000 | |
| 19 | 26.0000 | |
| 35 | 34.0000 | |
| 62 | 11.0000 | |
| 79 | 46.0000 | |
| 66 | 24.0000 | |
| 93 | 30.3333 | |
| 53 | 42.2500 | |
| 2 | 45.5000 | |
| 55 | 33.5000 | |
| 44 | 23.0000 | |
| 56 | 23.0000 | |
| 25 | 19.0000 | |
| 96 | 27.7500 | |
| 50 | 16.0000 | |
| 32 | 32.8000 | |
| 95 | 24.5000 | |
| 28 | 23.0000 | |
| 42 | 27.0000 | |
| 77 | 30.0000 | |
| 45 | 34.0000 | |

Query 11:

- Listing employee's names from the employee table and their assigned reviewer ID's from the reviewer table using INNER JOIN

```
select e.EmpID, e.First_Name, e.Last_Name, r.Reviewer_ID
from employee e
inner join reviewer r
on e.EmpID=r.EmpID
```

Output 11:

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|--------------|--------------|--------------------|
| EmpID | First_Name | Last_Name | Reviewer_ID |
| 3 | Corella | Playfoot | 1 |
| 4 | Amalia | Probey | 13 |
| 5 | Stephanus | Mogridge | 5 |
| 16 | Cris | Van der Beek | 15 |
| 19 | Desirae | Corley | 17 |
| 23 | Bronnie | Moseley | 19 |
| 23 | Bronnie | Moseley | 8 |
| 30 | Ebenezer | Kitlee | 2 |
| 34 | Kaylyn | Roisen | 4 |
| 39 | Genia | Coombe | 10 |
| 40 | Reena | Aldrich | 18 |
| 41 | Gothart | O'Tierney | 14 |
| 45 | Liane | Babinski | 7 |
| 63 | Adolf | Avieson | 20 |
| 72 | Charlotte | Smewing | 16 |
| 72 | Charlotte | Smewing | 6 |
| 79 | Jobi | Chupin | 11 |
| 84 | Domeniga | Howchin | 3 |
| 85 | Rosco | Airlie | 12 |
| 93 | Casi | Shasnan | 9 |

4. NoSQL Implementation

Three tables –

- Employee
- Scores
- Work

were created using MongoDB (Mongosh or Shell).

Schema and Table Generation:

4.1) Schema generation

use employeeratingapplication

```
db.runCommand( { buildInfo: 1 } )
```

4.2) Table Generation along with data input

```
> db.createCollection("Employee")
{ "ok" : 1 }
```

- db.Employee.insert({'EmpID':'1','First_Name':'Clark','Last_Name':'Kent','Designation_Name':'Engineer'})
- db.Employee.insert({'EmpID':'2','First_Name':'Tom','Last_Name':'Holland','Designation_Name':'Consultant'})
- db.Employee.insert({'EmpID':'3','First_Name':'Ravi','Last_Name':'Shankar','Designation_Name':'Human Resource'})
- db.Employee.insert({'EmpID':'4','First_Name':'Mark','Last_Name':'Gates','Designation_Name':'Typist'})
- db.Employee.insert({'EmpID':'5','First_Name':'Zack','Last_Name':'Snyder','Designation_Name':'Human Resource'})

```
> db.createCollection("Scores")
{ "ok" : 1 }
```

- db.Scores.insert({'EmpID':'19', 'Reviewer_ID':'10', 'Onsite_Performance_Score':'25'})
- db.Scores.insert({'EmpID':'7', 'Reviewer_ID':'11', 'Onsite_Performance_Score':'14'})
- db.Scores.insert({'EmpID':'21', 'Reviewer_ID':'14', 'Onsite_Performance_Score':'47'})
- db.Scores.insert({'EmpID':'33', 'Reviewer_ID':'18', 'Onsite_Performance_Score':'35'})
- db.Scores.insert({'EmpID':'30', 'Reviewer_ID':'16', 'Onsite_Performance_Score':'28'})

```
> db.createCollection("Work")
{ "ok" : 1 }
```

- db.Work.insert({'EmpID':'2', 'Work_ID':'1183', 'Work_Details':'Manager'})
- db.Work.insert({'EmpID':'5', 'Work_ID':'1576', 'Work_Details':'Director'})
- db.Work.insert({'EmpID':'26', 'Work_ID':'1296', 'Work_Details':'Analyst'})
- db.Work.insert({'EmpID':'39', 'Work_ID':'1991', 'Work_Details':'Manager'})
- db.Work.insert({'EmpID':'49', 'Work_ID':'1793', 'Work_Details':'Secretary'})

4.3) Queries and Outputs

The following queries were performed using the above schema, tables, and data -

Query 1:

- List all employees who have the designation “Human Resource”

```
>db.Employee.find({Designation_Name:"Human Resource"})
```

Output 1:

```
> db.Employee.find({Designation_Name:"Human Resource"})
{ "_id" : ObjectId("6268a7b087e3f82337b4342a"), "EmpID" : "3", "First_Name" : "Ravi", "Last_Name" : "Shankar", "Designation_Name" : "Human Resource" }
{ "_id" : ObjectId("6268a7b087e3f82337b4342c"), "EmpID" : "5", "First_Name" : "Zack", "Last_Name" : "Snyder", "Designation_Name" : "Human Resource" }
>
```

Query 2:

- List all employees whose EmpID > 3

```
>db.Employee.find({EmpID:{$gt:'3'}})
```

Output 2:

```
> db.Employee.find({EmpID:{$gt:'3'}})
{ "_id" : ObjectId("6268a7b087e3f82337b4342b"), "EmpID" : "4", "First_Name" : "Mark", "Last_Name" : "Gates", "Designation_Name" : "Typist" }
{ "_id" : ObjectId("6268a7b087e3f82337b4342c"), "EmpID" : "5", "First_Name" : "Zack", "Last_Name" : "Snyder", "Designation_Name" : "Human Resource" }
>
>
>
>
```

Query 3:

- List all Employees who has score >30

```
>db.Scores.find({Onsite_Performance_Score:{$gt:'30'}})
```

Output 3:

```
> db.Scores.find({Onsite_Performance_Score:{$gt:'30'}})
{ "_id" : ObjectId("6268a7d687e3f82337b4342f"), "EmpID" : "21", "Reviewer_ID" : "14", "Onsite_Performance_Score" : "47" }
{ "_id" : ObjectId("6268a7d687e3f82337b43430"), "EmpID" : "33", "Reviewer_ID" : "18", "Onsite_Performance_Score" : "35" }
>
>
>
>
```

Query 4:

- List all Employees who have score more than 30 or Employee who Emp ID = 33

Output 4:

```
> db.Scores.find({"$or": [{Onsite_Performance_Score :{$gt:30}},{EmpID :'33'}]})
{ "_id" : ObjectId("6268a7d687e3f82337b43430"), "EmpID" : "33", "Reviewer_ID" : "18", "Onsite_Performance_Score" : "35" }
>
>
>
>
```

5. Database Access via Python

The database is accessed using Python and analysis of the data is done via visualization. The code and graph are shown below –

5.1) Database Connection Code

```
import matplotlib.pyplot as plt

import mysql.connector

import pandas as pd

import numpy as np


mydb = mysql.connector.connect(host="localhost",
                               user="root",
                               password="group4",
                               database="employeeeratingapplication")

mycursor = mydb.cursor()
```

5.2) **Analysis 1:** Visualizing data of the number of employees each reviewer is responsible for

Code:

```
mycursor.execute('select count(e.EmpID), r.Reviewer_ID from employee e, reviewer r
where e.Reviewer_ID= r.Reviewer_ID group by Reviewer_ID')

result=mycursor.fetchall


no_of_emp=[]
reviewer=[]


for i in mycursor:

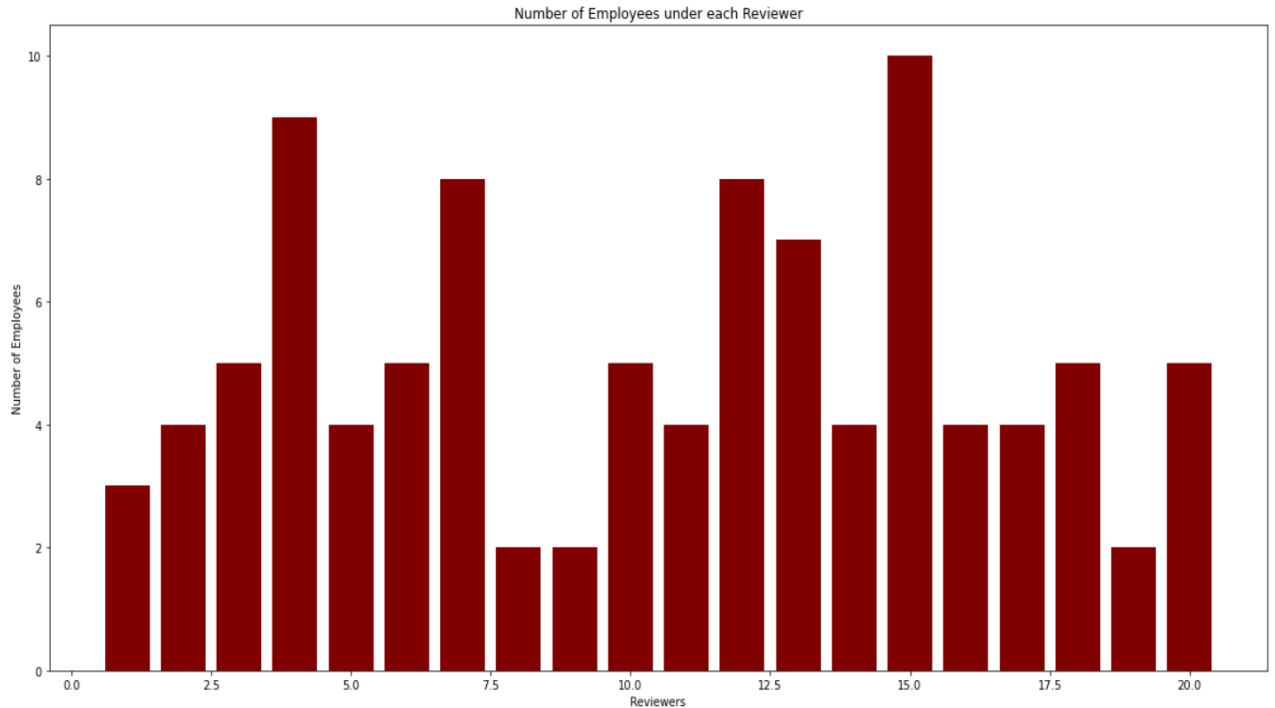
    no_of_emp.append(i[0])
    reviewer.append(i[1])


print("Number of employees=", no_of_emp)
print("Reviewers=", reviewer)
plt.bar(reviewer,no_of_emp,color='maroon')
plt.xlabel(" Reviewers")
plt.ylabel(" Number of Employees")
plt.title("Number of Employees under each Reviewer")
plt.show()
```

Graph 1:

From the bar-graph visualization here, we can derive insights on how many employees are assigned to every particular Reviewer. This is found out using the Employee IDs and Reviewer IDs

Continued below-



5.2) Analysis 2: Visualizing data of the average scores each department got rewarded with

Code:

```
mycursor.execute('select e.Department_Number, avg(s.Onsite_Performance_Score)
from employee e, scores s where e.EmplID=s.EmplID group by Department_Number')

result=mycursor.fetchall

department=[]

department_scores=[]

for i in mycursor:

    department.append(i[0])

    department_scores.append(i[1])

print("Name of Department=", department)

print("Scores=", department_scores)

plt.rcParams["figure.figsize"] = (20,10)

plt.stem(department,department_scores)

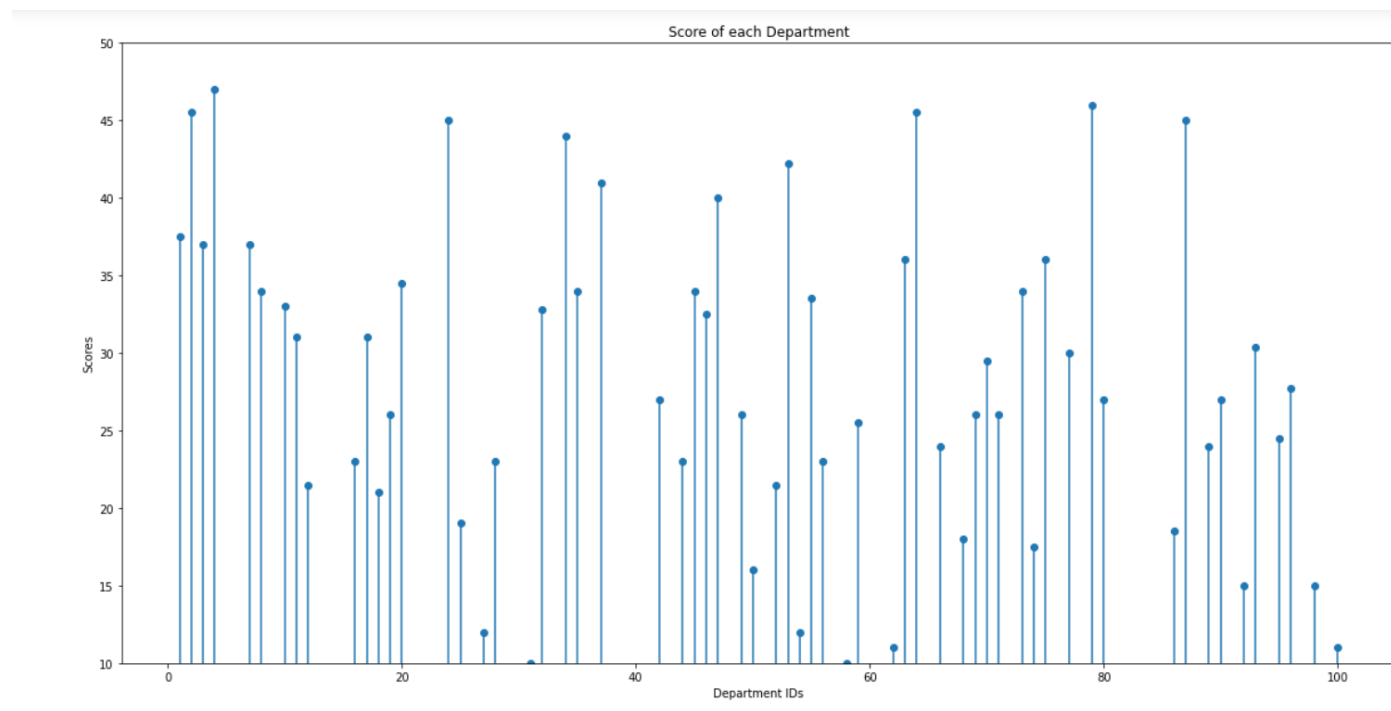
plt.ylim(10,50)

plt.xlabel("Department IDs")
```

```
plt.ylabel("Scores")
plt.title("Score of each Department")
plt.figure(figsize=(100,100))
```

Graph 2:

From the stem-pot visualization here, we can derive insights into the scores rewarded to each department. We can work on these insights to keep track of the highest performing departments and impose changes on the low-scoring departments for better results. This is found using the Department IDs and their On_Site_Performing_Score based on their Score IDS.



5.2) **Analysis 3:** Visualizing data of the number of scores each Work Title got

Code

```
mycursor.execute('select e.EmpID, e.Work_Title, e.First_Name, e.Last_Name,
s.Onsite_Performance_Score from employee e, scores s where e.EmpID= s.EmpID
having Onsite_Performance_Score>=23 order by Onsite_Performance_Score desc')

result=mycursor.fetchall

work=[]

score=[]

for i in mycursor:
    work.append(i[1])
```

```

score.append(i[4])

plt.scatter(work,score,color='r')

plt.xlabel("Work Title")

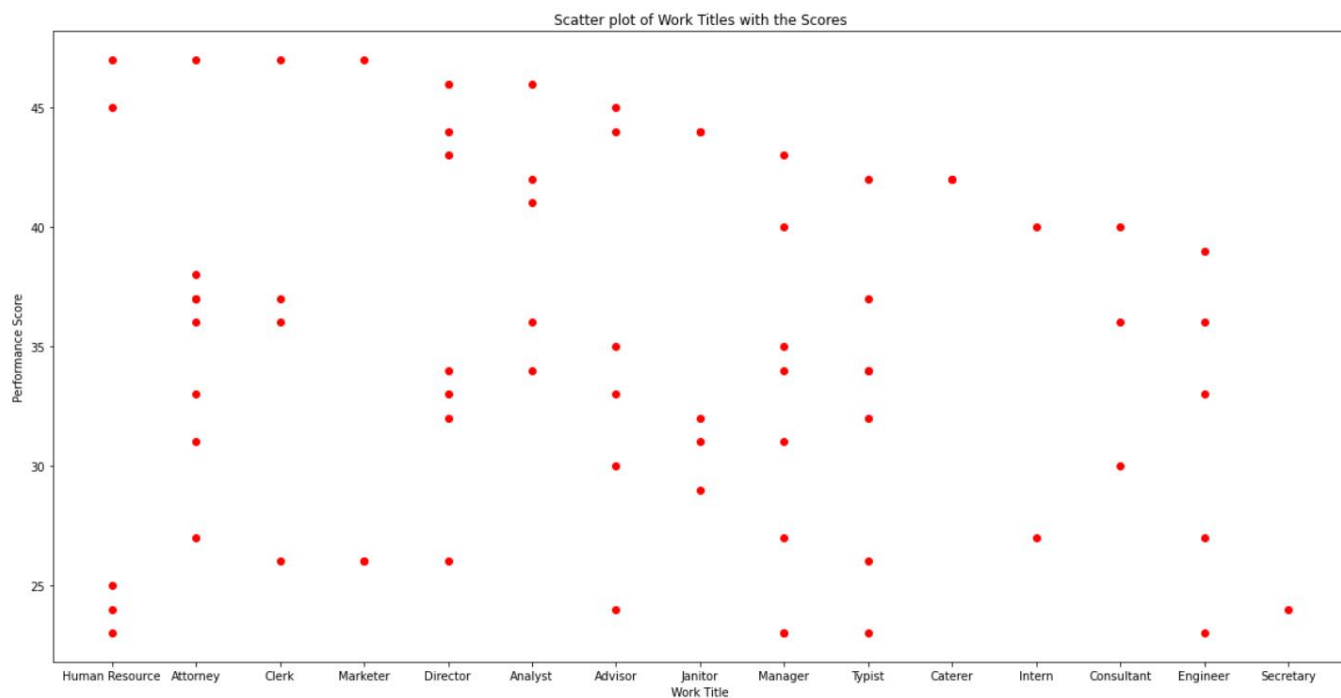
plt.ylabel("Performance Score")

plt.title('Scatter plot of Work Titles with the Scores')

```

Graph 3:

From the scatter-plot visualization here, we can derive insights into the scores rewarded for each work title. We can work on these insights to keep track of the highest performing work titles and impose changes on work strategy on the low-scoring work titles for better results. This is found using the Work Titles and their On_Site_Performing_Score based on their Score IDS.



6. Summary and Recommendation

Employee Rating Application is the new database that is ready to use and implement in any organization to support its Employee Rating System. This will help in increasing transparency and decreasing biasedness or factors tending to personal favors at the workplace and will also provide insights into the rating system using various analytical tools. This relational database shortens the data input process by 61% and eliminates the traditional manual process. The database also includes a Python-based analytics solution with the potential for analytics to monitor trends and make employee-based decisions. Both the employees and the management were thrilled to have access to the data, and it benefitted the organization to help make data-driven decisions. As the performance evaluation is publicly available for all

personnel across the company, it serves the purpose of increased transparency and invokes and sense of trust that consecutively allows for a healthy working environment.

Improvements can be done to this database by adding a panel of reviewers for all the employees based on their job performance. Along with this, a few features like shuffling the number of employees assigned to each reviewer quarterly, etc. can be implemented to improve load management on individuals along with justified performance evaluation for all.