Lewis University  
CPSC 50900: Database Systems  
Term Project

TITLE OF YOUR PROJECT:

**Employee Management System**

Proposed and submitted by:

Syed Aziz Ullah Hussaini, azizullahhussainis@lewisu.edu

Work products stored in the GitHub repository:

**https://github.com/SyedAziz21/EmployeeManagementSystem**

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 3](#_Toc85814570)

[Data Storage Alternatives 5](#_Toc85814571)

[Relational Database Design Process 7](#_Toc85814572)

[Relational Database Design 1](#_Toc85814573)3

[Data Definition Language (DDL) Scripts 1](#_Toc85814574)4

[Data Manipulation Language Scripts 1](#_Toc85814575)7

[Indexes 1](#_Toc85814576)8

[Views](#_Toc85814577) 20

[Triggers](#_Toc85814578) 21

[Transactions](#_Toc85814579) 22

[Database Security](#_Toc85814580) 24

[Locking and Concurrent Access](#_Toc85814581) 26

[Backing Up Your Database](#_Toc85814582) 28

[Python Programming](#_Toc85814583) 30

[PHP Programming](#_Toc85814584) 30

[Suggested Future Work](#_Toc85814585) 31

[Activity Log](#_Toc85814586) 33

# Initial Proposal

*Description: You will describe the data you aim to store. What data will be storing? Why are you interested in this data? Why is it important? Where will the data come from? Who will use this data? What kind of application do you plan to build with it?*

*Rubric: Your response to each of these six questions will be graded out of 3 points.*

* *3 points: clear, complete descriptions that convey the importance and meaning of your data*
* *2 points: mostly clear descriptions, although some additional data would have helped in some sections*
* *1 point: necessary details are lacking in many of your responses.*

*You will also earn 2 additional points for coming up with a descriptive title for your project.*

*Total points possible: 20*

**Employee Management System**

***Employee Management System*** is the idea behind implementation of the project which is used to keep all details of the employees of an Organization including all the personal, official, banking, experience, qualification, and identification details. All this data is very necessary for any organization to get the information of an employee at any moment of time for example if a manager wants to see the current salary of his reporting employees then he can just execute a query based on the requirement and can fetch the details required and if at all he wants to give hike to them then he can easily give it to them just by running a query instead of changing the salary for each person. My personal interest in this data is because if at all in future I got a chance to lead any team then it’ll be very helpful for me, also as I have already been a part of Project Management Team in my previous company (DXC Technology Ltd.) which led me having a good exposure to this kind of data almost for a year. Generally this data will be taken directly from the employee when he joins the organization and will be stored in the respective tables or data sources which will be managed by individual leads of each team considering an example let us say that John is going to join XYZ company then as a part of Onboarding process all the necessary details including personal, experience, address, passport/driving related, and qualification will be submitted by John whereas the other details like Branch, Team, Location will be given by manager and Salary, banking details will be given by HR, Finance Team respectively. This data will be mostly used by HR and Managers to give salary hike, if at all the manager is getting replaced then to update the same which includes team details as well. What location an employee is placed on etc will also has to be updated by respective Manager, also this will be acting as a directory to all the employees who work in the organization in which only the basic details like Name, Mobile Number, Location, Email ID, manager name, Employee ID, Address will be displayed when querying with the employee id.

The idea of the application is not yet finalized by me as there are a lot of applications that I can build using this database or data sources. Most probably I’ll be concentrating to build a website named as ‘Employee Portal’ that will act as directory to all employees and coming to HR and manager’s they’ll be updating the database based on the employee’s performance like giving hike, assigning the projects, giving promotions and all which will be reflected in the directory post the changes are done by HR and manager respectively. Depending on the tasks which we’ll be performing going further there might be some addition of data sources as well depending on the requirement which I’ll be keeping you updated with on a regular basis.

Note:

Personal, Qualification, Experience, Identification 🡪 This will be coming from the Employee that joins Organization

Banking, Official, Branch, Salary 🡪 This data will be coming from the Human Resource Manager and Technical Manager based on the project/resource requirement

**====================================================================================**

# **Data Sources**

*Description:* *Gather your data in text files. The text files may be csv, tab-delimited, xml, json, or some other custom format. Not all the files need be of the same type. Identify what each file contains by indicating where it came from, explaining in detail how it structured, and describing how you will reorganize the data into a relational database. Post your data files to your GitHub repository, and provide samples of the data in your Word doc.*

*Rubric: Your work will be graded as follows:*

* *5 points: you gathered multiple data files that contain the data that will populate your databases. If you do not use multiple data files, you will not receive credit.*
* *5 points: you described the contents of the data files in detail, including referencing their origin and explaining how they were structured.*
* *3 points: you identify which fields you plan to include in your database, including their data types and any constraints you expect to impose on the data or steps you'll have to take to clean up the data.*
* *2 points: you post the data files to your GitHub account and make it possible for me to see them.*

*Total points possible: 15*

There are various data sources we are using here in different formats like .txt, .json, .csv and .xlsx formats which we’ll need to insert into the respective tables into database the data sources and their relative information are as given below

1. ***address.json*** 🡪 This will contain the address of the employee and has the primary key as the zipcode, this data will be provided by the employee while onboarding process. It is basically needed to keep the information of employee i.e., where he currently lives in.
2. ***bankdetails.json*** 🡪 This data we are taking as the json format and will be provided by the Finance team when an employee joins basically this will contain the details of bank where the account is created for the respective employee
3. ***branchdetails.txt*** 🡪 These are the details of the work location of the employee that will be which will be shared by the manager once the projects get assigned to him. It purely depends on the client or the team that a location will be assigned to him
4. ***Employee.csv*** 🡪 This is our main file/data source which contains all the necessary information of the employee in the form of CSV, most of the fields are provided by the Employee but there are a couple of fields that’ll be determined by HR and respective project Manager for example Salary
5. ***identificationdetails.xlsx*** 🡪 These are more likely the details which are collected as the address proof from the employee which consists of a couple of Government recognized IDs that’ll be helpful in identifying the employee during background verification. We have taken Driving License and the Passport details of employees as the source of information for this purpose.
6. ***officialdetails.json*** 🡪 These details are allotted by the HR after the employee onboards this will mostly consists of the manager details the bank details, HR details etc. This will be directly managed by the HR and Manager depending on the changes what an employee goes through with the project perspective
7. ***Qualification.csv*** 🡪 These are the details of the qualifications of the employees, these will be updated by the employee itself.
8. ***WorkExperience.xlsx*** 🡪 This table/data source depicts the employee experience and address of the company where the employee is currently working and its joining date to the company.

# **Data Storage Alternatives**

*Description: We will study alternatives to storing data in a relational database. Some of the alternatives come from several decades ago, including the hierarchical and network models. Some are newer options, such as NoSQL databases that use JSON or some other encoding. Describe in detail how to store the data using two alternatives to relational databases. Be sure to describe how you would implement the alternatives and the advantages and disadvantages of each.*

*Rubric: Your work will be graded as follows*

* *5 points for clearly describing how your data could be stored using one alternative to relational databases and what the advantages and disadvantages of that approach would be.*
* *5 points for clearly describing how your data could be stored using another alternative to relational databases and what the advantages and disadvantages of that approach would be.*

*Total points possible: 10*

**Alternate Ways**

The primary database that we are using in the project is a relational database(SQL), we do have N number of tables one for storing Personal Information, one for Storing Contact Information, and so on, these tables are used for storing the data and performing CRUD operations on it. All the tables are linked to each other in one or another way.

As an alternate, we can also make use of MongoDB and Redis

1. **MongoDB**

MongoDB is a document-oriented database that stores data in JSON-like documents with dynamic schema. which makes it easier for you to store the data without thinking about the data structure to be used such as the number of fields or types of fields to store values. MongoDB documents are like JSON objects.

In MongoDB instead of creating tables we’ll be creating the collections with below syntax and will be performing similar operations like select using its related commands

* For creating collection: db.createCollection("mycol ")
* For querying collection: db.mycol.find()
* For Inserting the documents: db.mycol.insert([records we want to insert])
* For dropping the collection: db. mycol.drop()

**Advantages of using MongoDB**

* Handles large volume of data
* Developer Friendly
* Easy to deal with Cloud with zero Downtime.

**Disadvantages of using MongoDB**

* It doesn’t support transaction.
* It uses high memory for data storage.

Example showing how data is stored in SQL and Mongo DB

A record in SQL:

|  |  |  |  |
| --- | --- | --- | --- |
| IFSCCode | BankName | Location | BranchManager |
| FR0PRS | Bank of France | Paris | John Walker |

A document in Mongo DB:

{

"IFSCCode": "FR0PRS",

"BankName": "Bank of France",

"Location": "Paris",

"BranchManager": "John Walker"

}

1. **Redis**

As an Alternate, we can also make use of PostgreSQL which is open-source database also used as message broker or caching system. It has built-in replication, scripting, transactions, and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. Depending on our requirement or use case we can persist data either by periodically dumping the dataset to disk or by appending the command to disk based log

In Redis we have to setup a cluster and then it’ll be direct for us to create a database using + symbol and to load the data we’ll using the import. We’ll having to do most of the things from console in Redis.

**Advantages of using Redis**

* We can use Redis from most of the programming languages
* Redis supports almost all data structures.
* Redis uses its own hashing mechanism called Redis Hashing

**Disadvantages**

* It requires a huge ram because it is in-memory so are not supposed to use it on ram servers.
* Clients connecting to the Redis cluster should be aware of the cluster topology, causing overhead configuration on Clients.

Redis stores the data in following format

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rediskey | IFSCCode | BankName | Location | BranchManager |
| bank:1000 | FR0PRS | Bank of France | Paris | John Walker |

SQL stores as below in contrast

|  |  |  |  |
| --- | --- | --- | --- |
| IFSCCode | BankName | Location | BranchManager |
| FR0PRS | Bank of France | Paris | John Walker |

But in our case, we are using SQL database as

SQL databases provide great benefits for the data whose structure doesn’t change frequently and where data integrity is paramount. It’s also best for fast analytical queries. And in our employee management system, we have no plans for changing Database Schema, and as well data Integrity is the priority.

**====================================================================================**

# **Relational Database Design Process**

*Description: Consider the list of fields you identified in part c. Identify functional dependencies that exist among them. For each functional dependency, identify the determinants and the fields they determine. This becomes the basis for identifying your entity sets, which then become your tables. Give each entity set or table you identify in this way a unique and clear name, making sure that the names you use are singular nouns. Then list the relationships that exist among the various entity sets. For each relationship, identify its connectivity (one-to-one, one-to-many, many-to-many) and participation (optional or mandatory). Finally, make sure that none of the attributes you've assigned to each entity set are multi-valued. If they are, take the steps needed to break them down.*

*Rubric: Your work will be graded as follows:*

* *8 points for identifying all the functional dependencies, including determinants and the columns whose values they determine.*
* *2 points for naming the entity sets that make up your data with clear, easy-to-understand names.*
* *6 points for identify the relationships among the entity sets and identifying connectivity and participation for each.*
* *2 points for breaking down multi-valued attributes.*

*Total points possible: 18*

***List of Functional dependencies :***

*===========================*

***zipcode --> city***

***zipcode --> State***

Zipcode attribute from the address data determines the city, state attributes which implies that zipcode is the determinant

*===========================*

***IFSCCode --> BankName***

***IFSCCode --> BranchManager***

BankName and BranchManager are the two attributes which are determined by the IFSCCode which is in turn determined by EmployeeId

*===========================*

***branch\_id --> branch\_head***

***branch\_id --> number\_of\_employees***

By knowing the branch\_id we can determine the attributes related to it such as branch\_head, number\_of\_employees present in that branch

*===========================*

***EmployeeId, FirstName --> DateOfBirth***

***EmployeeId,FirstName,LastName --> EmailId***

EmployeeId is the main primary key which determines a lot of other attributes like IdentificationNumber, DateOfBirth etc.,

Apart from above if we take into considerstion the EmailId it is a field which will be determined by attributes EmployeeId, FirstName, LastName

*===========================*

***Identification Number --> Driving license, Passport Number***

***Passport Number --> Passport Expiry, Passport issue***

***Driving license --> Driving License Expiry***

Here Identification Number determines Driving license and Passport Number uniquely, these are the fields that in turn determines the other attributes depending on them for example the passport issue and passport expiry date

*===========================*

***EmployeeId,BranchId --> ManagerId***

***EmployeeId --> IdentificationNumber***

***IFSCcode --> AccountNumber***

IFSCCode is the determinant field which determines AccountNumber uniquely. The attributes EmployeeId,BranchId collectively determine ManagerId for that Employee

*===========================================================================*

***Entity Sets :***

The below table displays the entity sets with the respective source where they’ll be getting data from.

|  |  |
| --- | --- |
| Source | Entity Set |
| address.json | employee\_address |
| bankdetails.json | bank\_details |
| branchdetails.txt | company\_branch |
| Employee.csv | employee |
| identificationdetails.xlsx | emp\_identification |
| Officialdetails.json | official\_data |
| Qualification.csv | emp\_education |
| WorkExperience.xlsx | emp\_experience |

***Entity Relation Ship Diagrams :***

Figure 1 below shows the relationship between entities ‘employee’ and ‘bank\_details’, this is the strong relationship as both the tables has primary key present in them

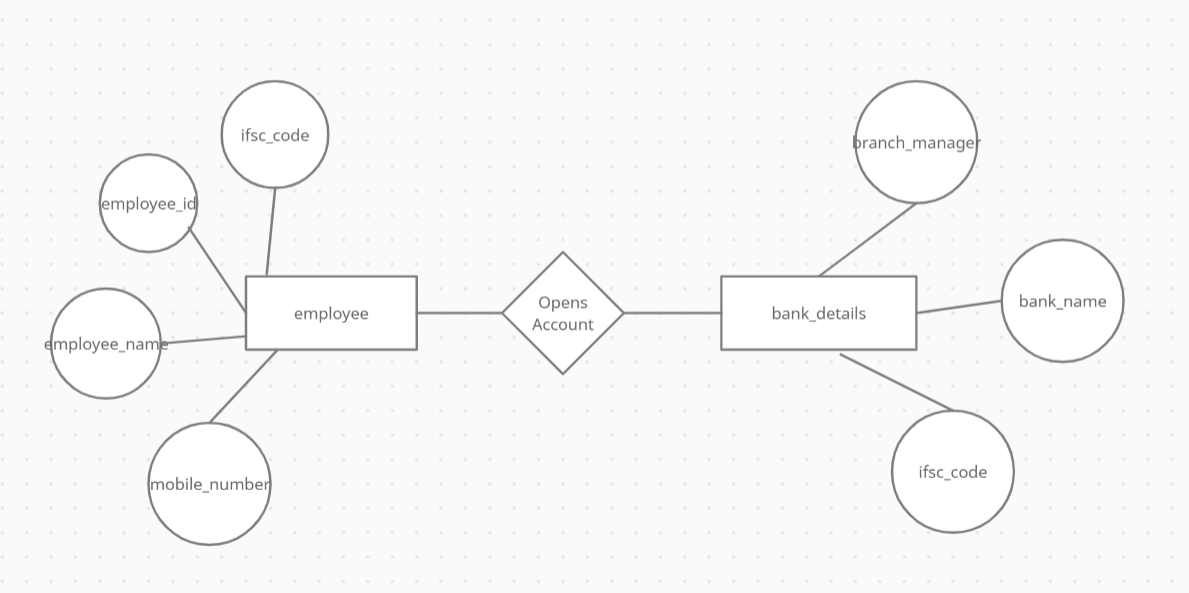
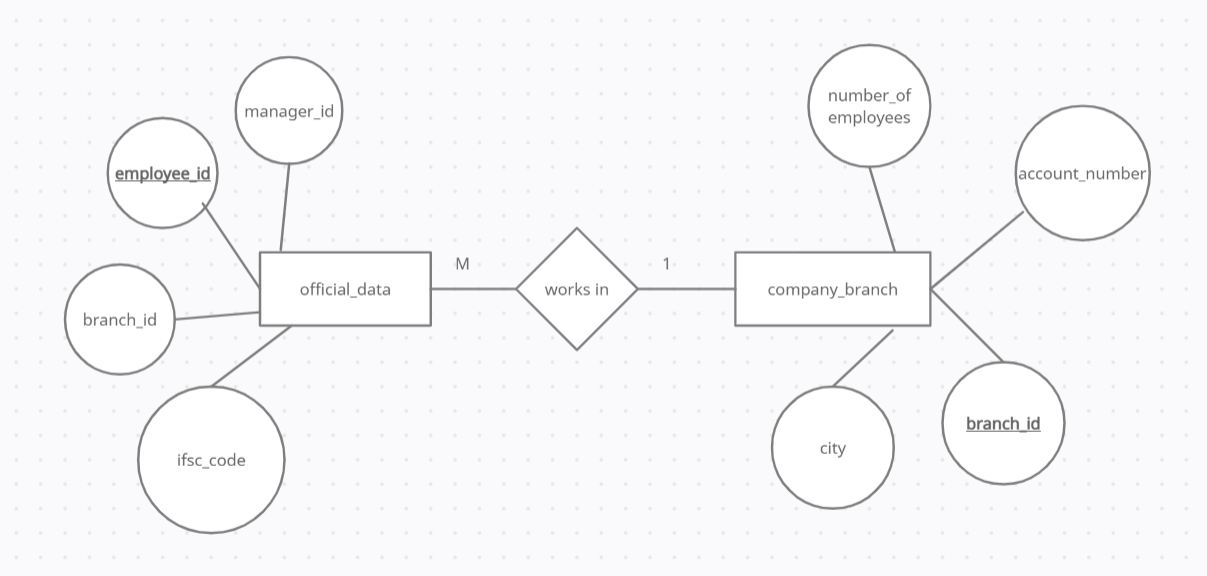
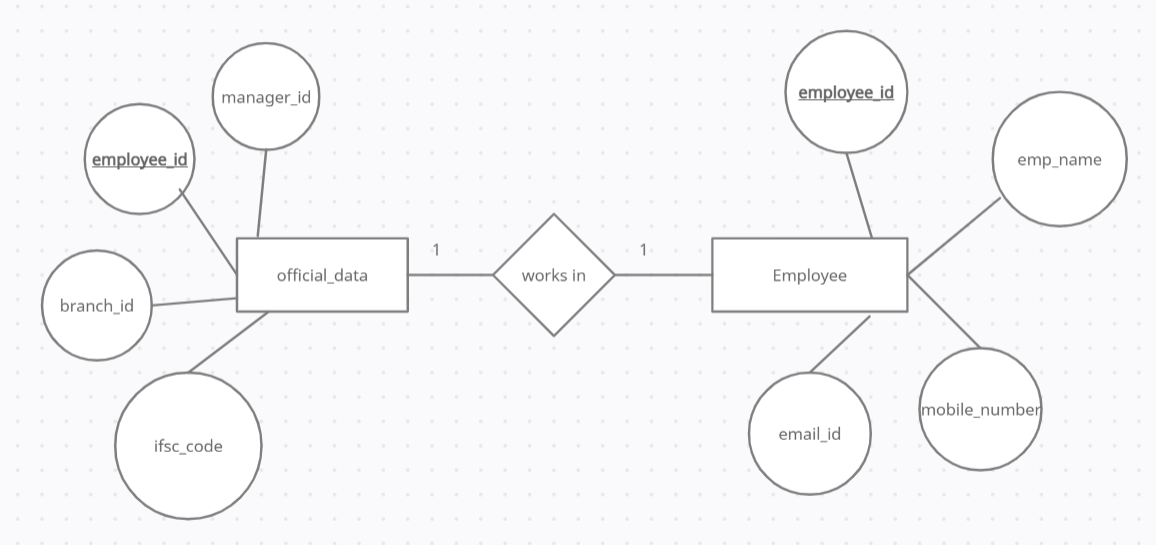


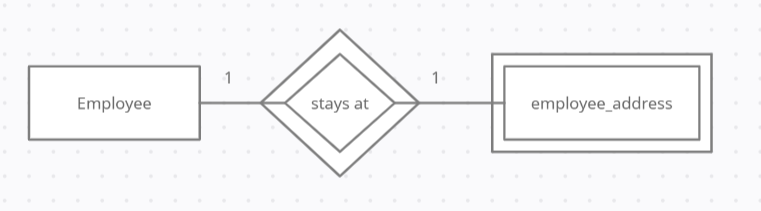
Figure 2 represents the ER Diagram between official\_data and the company\_branch which is also a strong entity relationship, it also represents the many to one relationship for the below entities as many employees can work in the same company branch. **Employee\_id, branch\_id** represents the primary keys in each tables

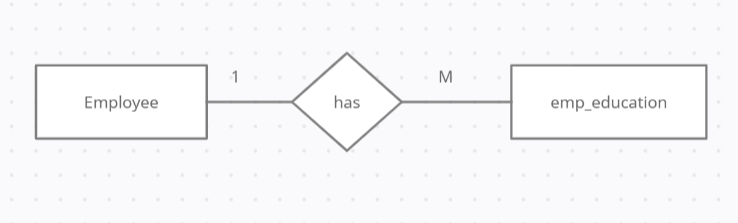


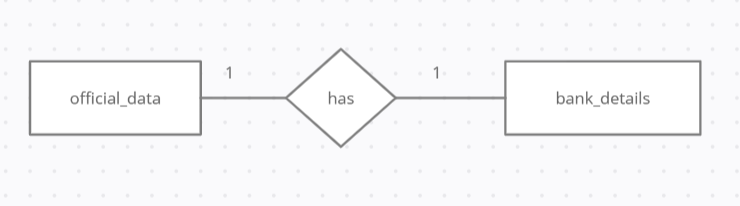
Below ER Diagram is the important one as it connects the two important tables with one another based on only one primary key i.e., employee\_id and it is only 1 to 1 relationship



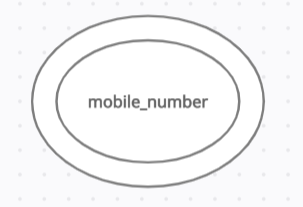
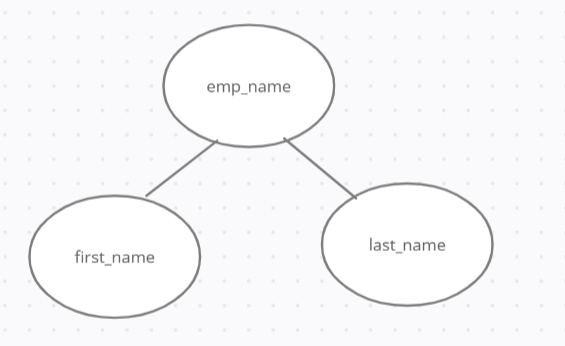
Various other entity relationships that can be established from the data that we have are as given below



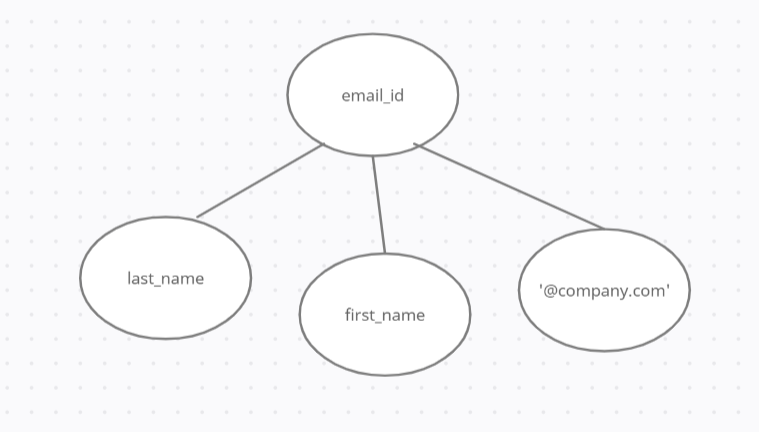




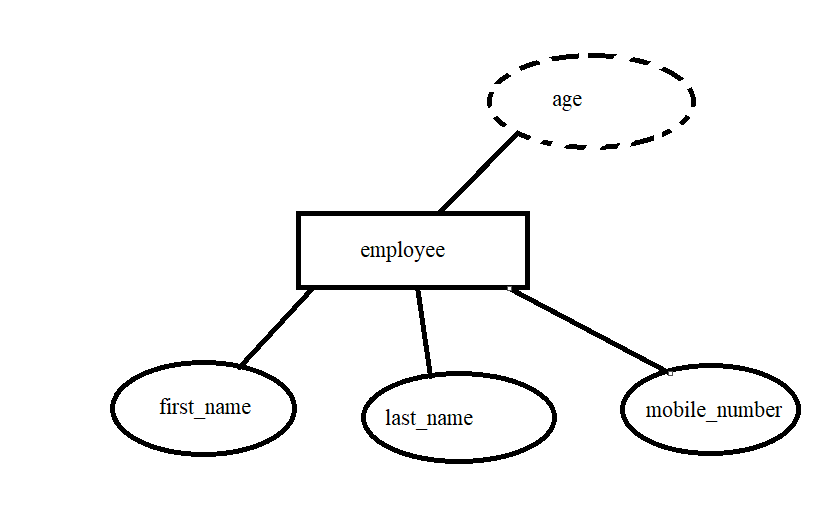
Multi Valued attributes : Attributes that take more than one values are very few in our system namely mobile\_number, whereas we have a couple of derived attributes like emp\_name which gets derived from the First\_name and last\_name, email\_id which again gets derived from first\_name and last\_name. We can have an attribute age which can be derived from the date\_of\_birth

Composite attribute :



Derived attribute :



# Relational Database Design

*Description: This is where you will complete your database design. For each of the entity sets you identified in the preceding section, analyze them to make sure they pass 2nd, 3rd, 4th, and Boyce-Codd Normal Form. If they do not, introduce additional entity sets or key changes to make sure that they do. Then, add foreign keys to connect entity sets that are related. For many-to-many relationships, introduce bridge entity sets to convert them into two one-to-many relationships. Also, consider whether you should introduce surrogate keys to create a more efficient primary key for some of your entity sets. Finally, diagram your design in Vertabello. Make sure your ER diagram correctly shows all entity sets, their primary and foreign keys, the data types for each attribute, and the connectivity and participation characteristics of each entity set. Your final Vertabello design should be something you could actually implement in a relational database management system.*

*Rubric: Your work will be graded as follows:*

* *4 points for the normalization analysis of your entity sets.*
* *3 points for introducing bridge entity sets.*
* *3 points for choosing foreign keys and perhaps more efficient surrogate keys*
* *10 points for correctly depicting your physical database model in Vertabello*

*Total points possible: 20*

**Normalization :**

We can normalize the employee table by removing the attribute address which is unnecessary as we already have that in another table emp\_address that we can join based on the zip code attribute, as far as the other entities are concerned they look normalized only as each of them are uniquely identified by a primary key and as already mentioned they have relationships defined with other entities as well.

**Bridge entity :**

To establish a many to many relationship we can include here one entity set i.e., ***project details*** where more than one employee can work on multiple projects below is the structure defined for that entity

project\_id

project\_name

client

years\_of\_contract

number\_of\_employees

project\_manager

**Foreign keys :**

Employee table has a primary key emp\_id, the same attribute emp\_id acts as foreign key in the emp\_education table referencing the employee table

Employee table has a primary key emp\_id, the same attribute emp\_id acts as foreign key in the emp\_experience table referencing the employee table as one employee can join the same organization after a span of years after resigning in that case there can be two entries of the same emp\_id

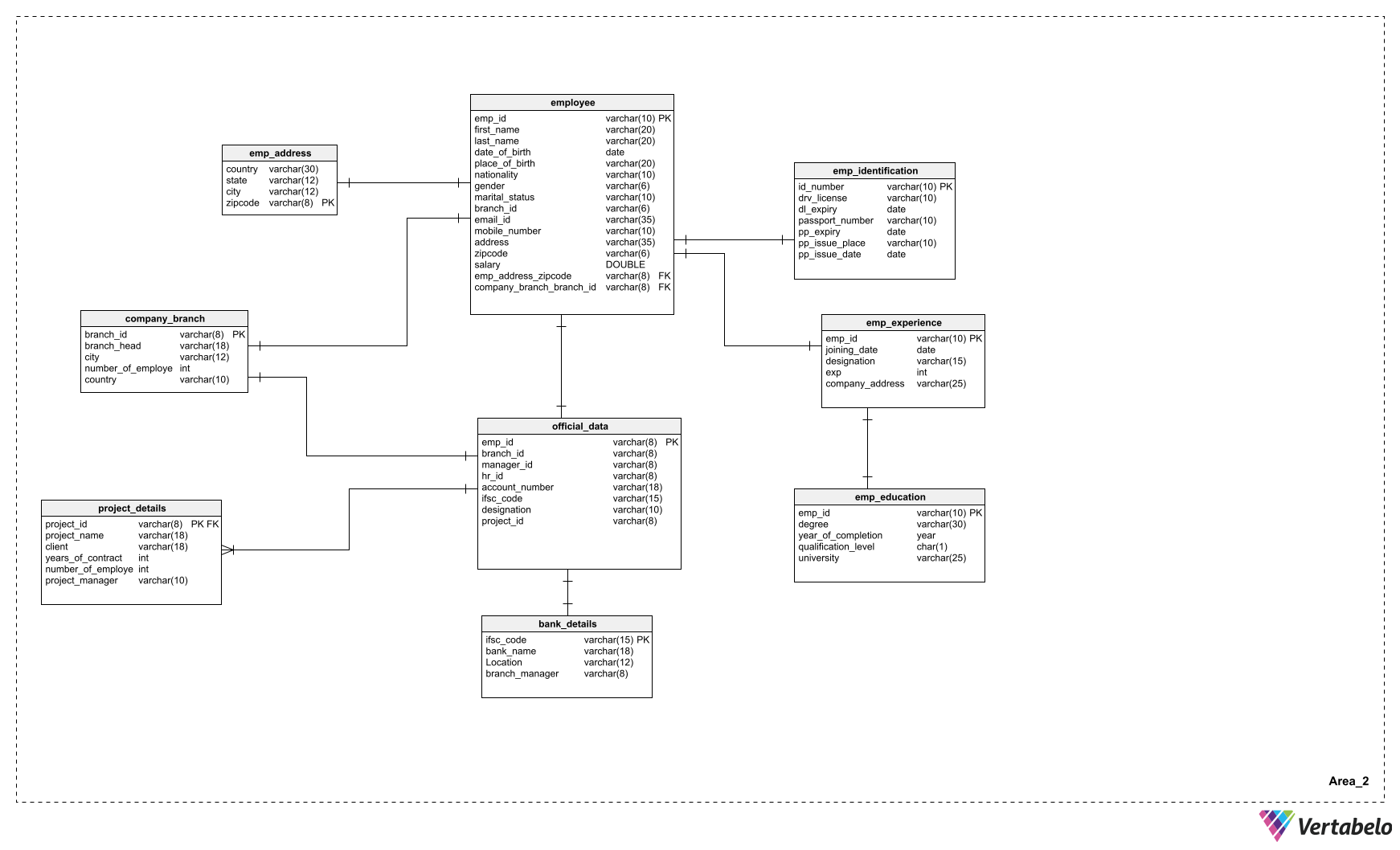
In table emp\_address the attribute zipcode acts as the primary key whereas the same attribute will act as foreign key in Employee table referencing the table emp\_address

In project\_details bridge entity the key project id is the primary key which in the official\_data acts as the foreign key

Ifsc\_code in the bank\_details is the primary key whereas the same attribute in official\_data is the foreign key,

Branch\_id in the company\_branch is the primary key whereas the same attribute in the official data acts as the foreign key

**Physical Data Model design for the Employee Management System:**

****

# Data Definition Language (DDL) Scripts

*Description: Use Vertabello to generate a script of SQL commands that build the database and its table structures. Write scripts or build Excel spreadsheets that take your data files and generate scripts of SQL insert statements from them. Use the MySQL source command to run the various scripts needed to build and populate the database in MySQL. Include the source code and / or Excel spreadsheets you use to manipulate and populate the data. Make sure all your tables have at least three records in them and that you've linked the tables through their foreign keys.*

*Rubric: Your work will be grades as follows:*

* *Database and table creation statements from Vertabello saved as an sql script file: 3 points*
* *Scripts you write or Excel spreadsheets you create to generate SQL commands for populating the tables, uploaded to GitHub: 8 points*
* *Descriptions of the scripts and Excel spreadsheets you wrote along with code excerpts included in the Word document: 5 points*
* *Screenshots of your successful attempts to use the MySQL source command to populate each table with at least three records: 4 points*

*Total points possible: 20*

*Database and table creation statements from Vertabello saved as an sql script file*

**

*Mysql successful execution of the database and table creation*

**

**Scripts to create the database and tables used:**

The below attached are the statements used to create the database ‘emp\_mgmt\_system’. The tables are created as per the requirement of the system. In order to create the tables we need to follow the order keeping the dependencies in mind as there are foreign keys that are present in other tables too. Once the tables are created I tried loading the data using the load csv command but faced several issues which I tried fixing but due to some restricted privileges I wasn’t successful in loading the data directly from csv so I have created the scripts that are used for loading./ingesting the data into the tables.

**

|  |  |
| --- | --- |
| database name: | emp\_mgmt\_system |
| Tables: |  |
| bank\_details | This table contains the details of all the banks where employees holds their accounts. It contains only 4 attributes with 1 primary key |
| company\_branch | This table contains the details of the company branches and its head's along with the  total number of employees working in that particular company branch. It has 5 attributes in total with 1 primary key |
| emp\_address | This table shows the details of the employee's address i.e in which city, state the employee lives with its zipcode Zipcode acts as the primary key for this table. Total number of attributes present are only 4 |
| emp\_education | The table has the list of educational qualifications with year of completion of all the employees,  it doesn't have any primary key present but emp\_id acts as the Foreign key in this table |
| emp\_experience | emp\_experience table keeps track of employee's date of joining and the work experience they are having emp\_id acts as the foreogn key here as well |
| emp\_identification | To identify an employee apart from the company id we need any of the government identity proof for the  employee which has been kept in this table, it'll be hvaing the details specifically of passport and  Driving License |
| employee | employee table is the main table in our entire database as our database name itself suggests that,  it'll be having several attributes which helps to keep track of all details including the personal, official and salary too emp\_id plays the role of primary key here with zipcode acting as the foreign key |
| official\_data | This table is need to identify the official details of the employee like which manager it reports to, who is HR, present working location and other stuff including the bank details  It has one primary key and three foreign keys present |
| project\_details | This is the bridge entity that we have introduced, it mainly depicts the many to many  relationship from employee to a project i.e., how many projects are currently being worked by the  employee and how many employees are working in one single project etc., It has one primary key in it |

*Below are the screenshots attached for the successful execution of table creation and data load into the tables  *

# Data Manipulation Language Scripts

*Description: Write the SQL commands for twelve queries. Two queries should be insert statements, two should update statements, one should be a delete statement, one should be a simple select statement that selects a subset of the rows and columns from one table, two should be a select statements that select data from a joining of two tables, two should use summary functions to generate statistics about the data, one should be a multi-table query, and one should be another query of your choice. Show the queries and screenshots of the results in your Word document, and save your queries in a commented sql script to GitHub.*

*Rubric: Your work will be graded as follows:*

* *1 point each for the two insert statements*
* *1 point each for the two update statements*
* *1 point for the delete statement*
* *1 point for the simple select statement*
* *2 points each for the 2 join statements*
* *2 points each for the two that use summary statements*
* *2 points for the multi-table query*
* *2 points for the query of your choice.*
* *12 points for showing the query and a screenshot of the corresponding result set back-to-back for each of these queries in your Word document.*

*Total points possible: 30*

*Attaching below the document stored with all the DML queries executed along with its successful screenshots. It contains works on joins, sub queries and a couple of built in functions namely concat() and lower()*

**

***====================================================================================***

# Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *3 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 12*

*Indexes:*

*In this database/project am defining just 4 indexes as I think these will be important.*

*The first index* ***“idx\_emp\_id”*** *will be on the employee table on emp\_id column because this is the primary key and whenever we join employee table with another the probability of using the emp\_id is 99% and when a select statement is run using the emp\_id it’ll fetch the values more faster than it was doing earlier.*

*The second index* ***“idx\_zip”*** *will be a combination of two columns {emp\_id,zipcode} as this is a combination of the keys which will be used frequently to fetch the data so whenever this index is accessed it’ll be accessing through the emp\_id first which is the primary key there by reducing the number of operations required*

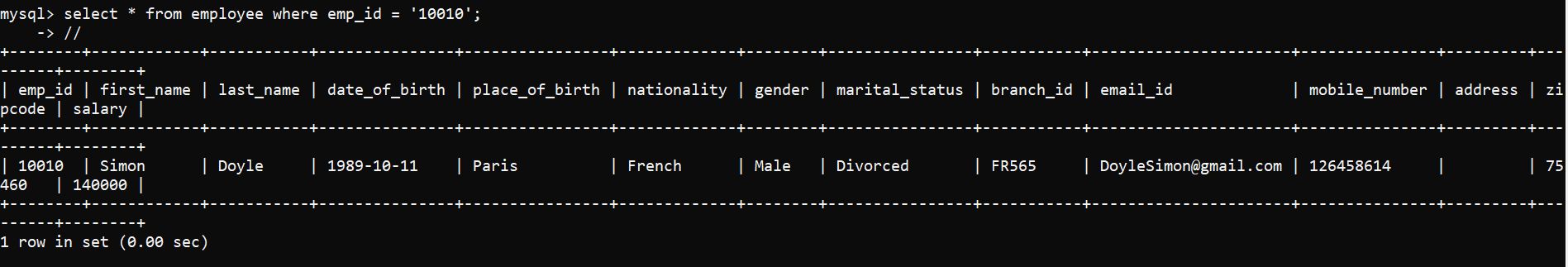
*The first index* ***“idx\_ifsc”*** *will be on the bank details table on ifsc\_code column because this is the main column which connects to other tables and is unique whenever we join bank\_details table with another it’ll fetch the values faster using ifsc\_code.*

*The first index* ***“idx\_zip\_code”*** *will be on the employee address table on zip\_code column because this is the primary key and whenever we join employee table with another, the probability of using the zip\_code is more it’ll fetch the values faster than it was doing earlier.*

***Sql needed for creating indexes:***

*create index idx\_emp\_id on employee (emp\_id);  
create index idx\_zip on employee (emp\_id, zipcode);  
create index idx\_ifsc on bank\_details (ifsc\_code);  
create index idx\_zip\_code on emp\_address (zipcode);*

*The performance of the indexes depends on the way they search the column, we have various types of indexes like clustered or non-clustered index. The mechanism to search for a column is simple as they follow, considering the employee table that am having which is indexed on emp\_id whenever the table is searched for a particular emp\_id it’ll search it in a form of B-tree i.e the emp\_id will be stored and indexed in sorted fashion having a pointer to original table thus by comparing the index value it reduces the number of operations required to fetch that value to lower when compared for the earlier approach of not indexing. Below is the example of the same*



# Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

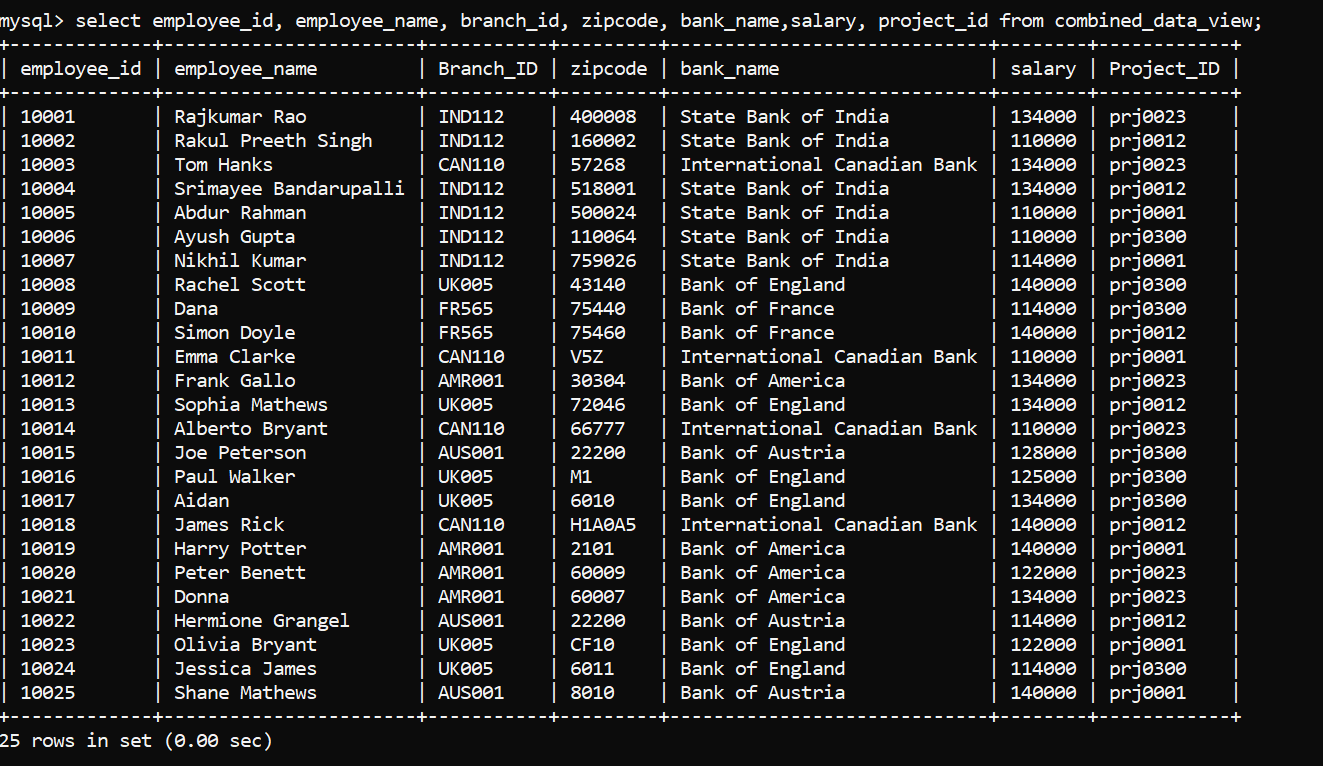
* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*
* *2 points for explaining who might benefit most from having access to each view.*

*Total points possible: 8*

**

1. combined\_data\_view

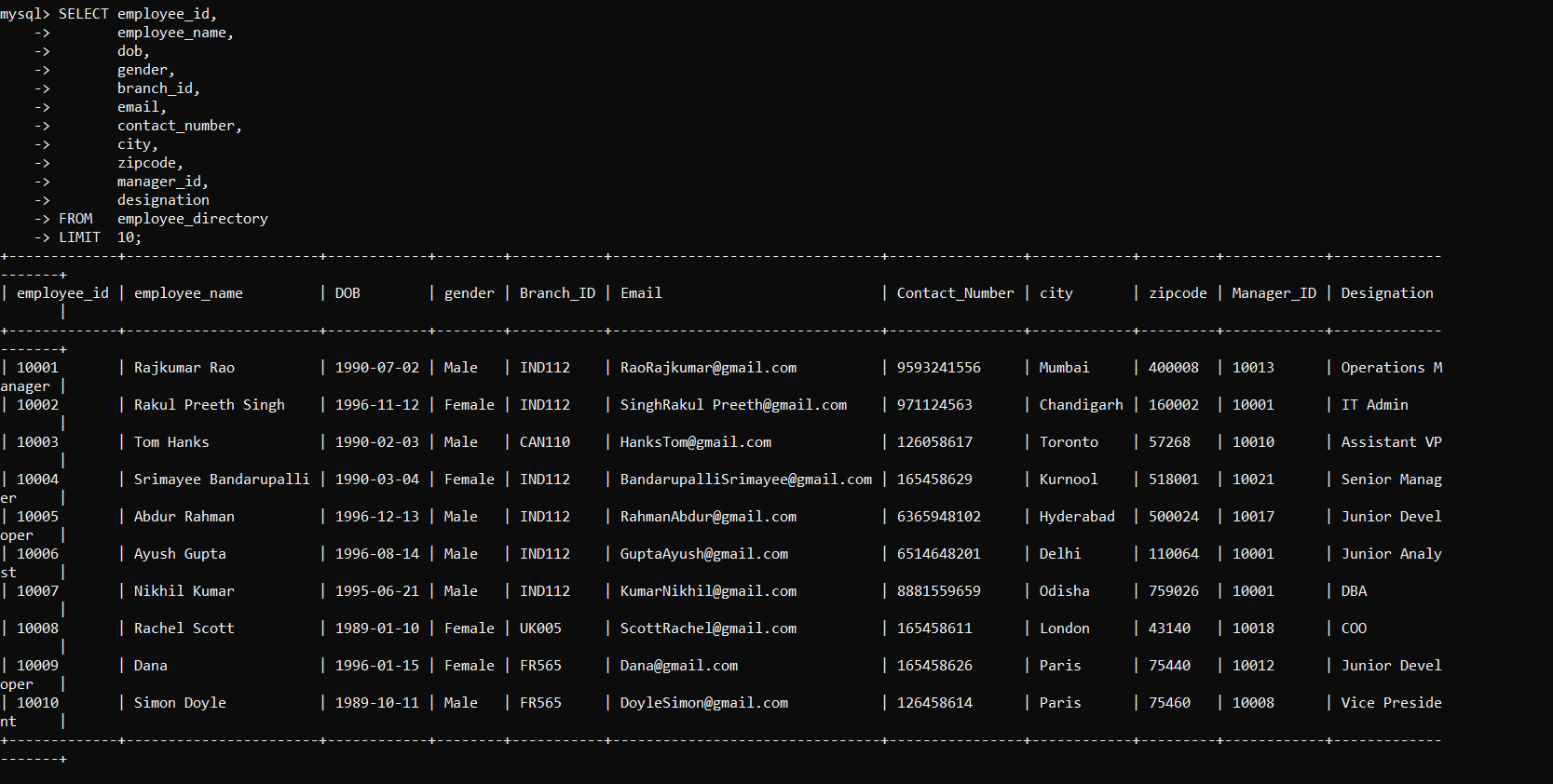
*select employee\_id, employee\_name, branch\_id, zipcode, bank\_name,salary, project\_id from combined\_data\_view;*



1. employee\_directory :

*SELECT employee\_id, employee\_name,dob,gender,branch\_id,email,*

*contact\_number,city,zipcode,manager\_id,designation FROM employee\_directory LIMIT 10;*



*There have been two views created which are beneficial for the database as if we want to create an application of the same like for employee directory, we’ll be able to use the view employee\_directory instead of running the multiple joins on table every time. This view employee\_directory can be used by anyone who is present in the organization and can have access to the general details of the employee and can reach out to them if required whereas the other view which is combined\_data\_view is used by the Manager’s and HR’s to look at the overall view of the employee details as they are authorized to do that. They can see each and every detail of the employee from that view without efforts of joining the each table every time which will in turn take a lot of time to identify also we cannot expect HR’s to run queries or join tables to see the data as they are non-technical people and will need access to this kind of utilizations from the existing things*

# 

# Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

***SQL for Trigger:***

*Delimiter //*

*Create Trigger default\_designation BEFORE INSERT ON official\_data FOR EACH ROW*

*BEGIN*

*IF NEW.designation is null THEN SET NEW.designation = 'Analyst';*

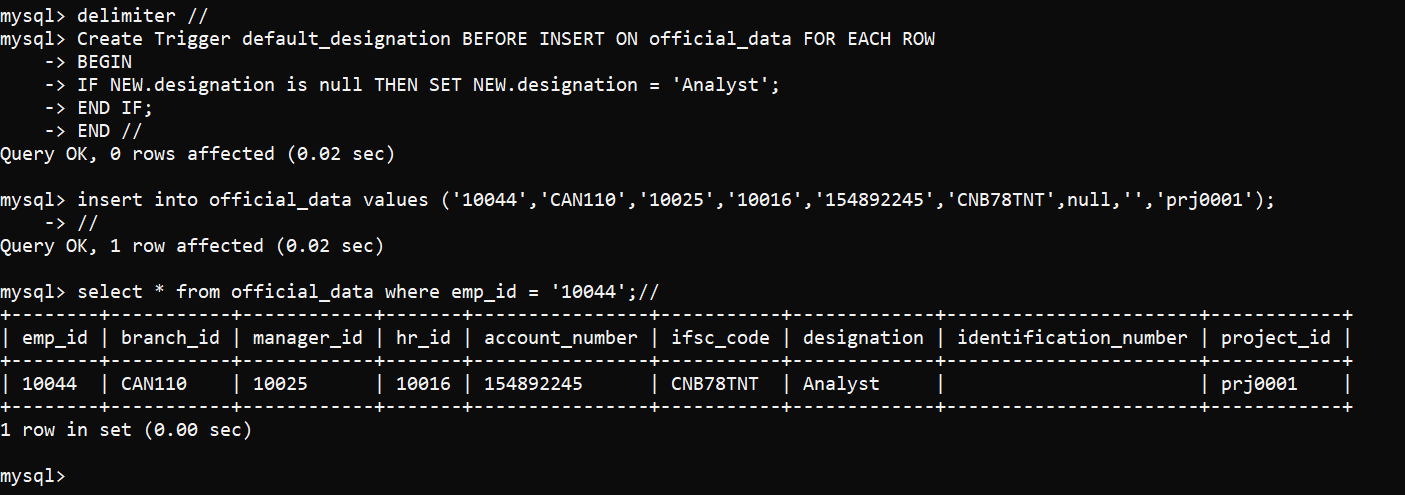
*END IF;*

*END//*

***Description for Trigger:***

*Generally, whenever a employee is hired at the company they’ll be assigned a designation so this trigger will check whenever an entry is made in the official\_data for any employee but its designation is null then on this event it’ll by default assign the designation to that employee as ‘Analyst’ as that will be the position allotted to any employee who has no experience or is joining as a fresher*

***Trigger in Action:***



# Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

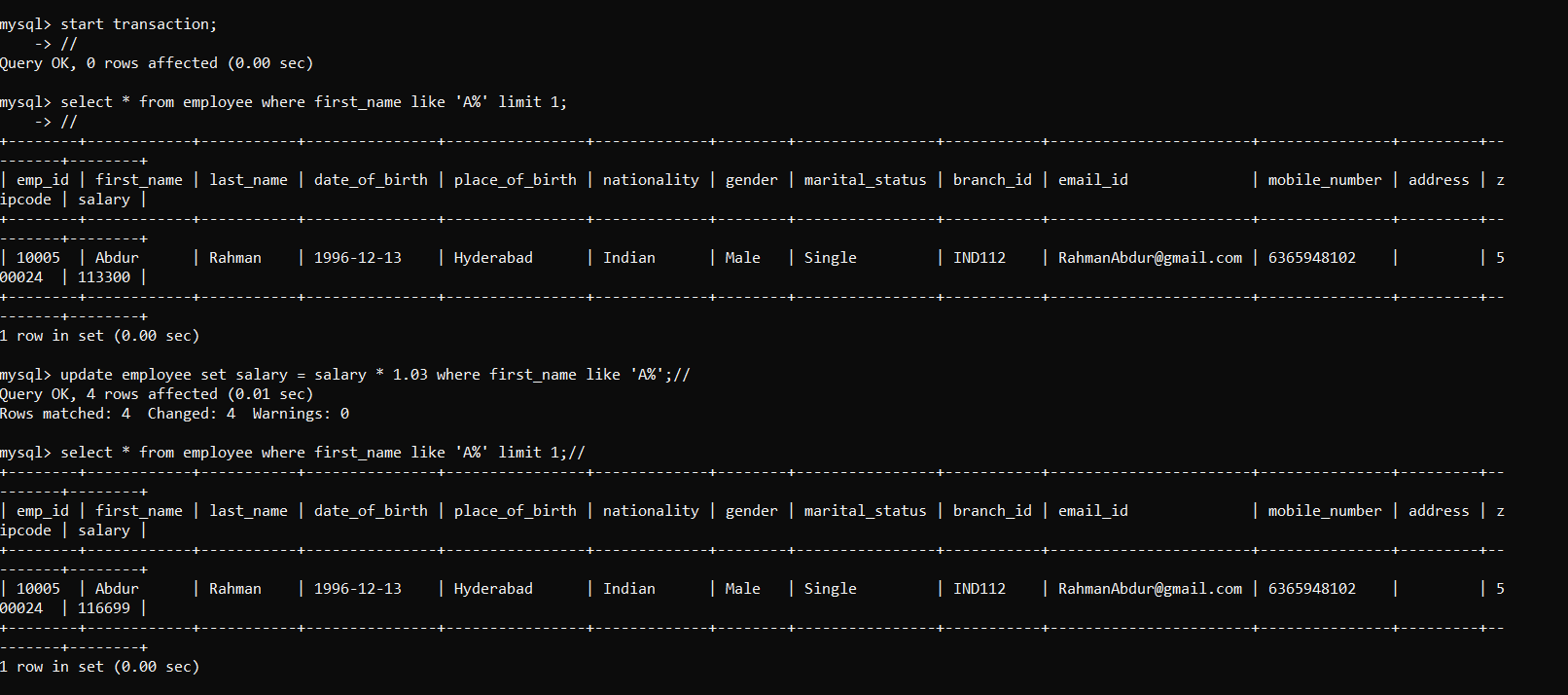
* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

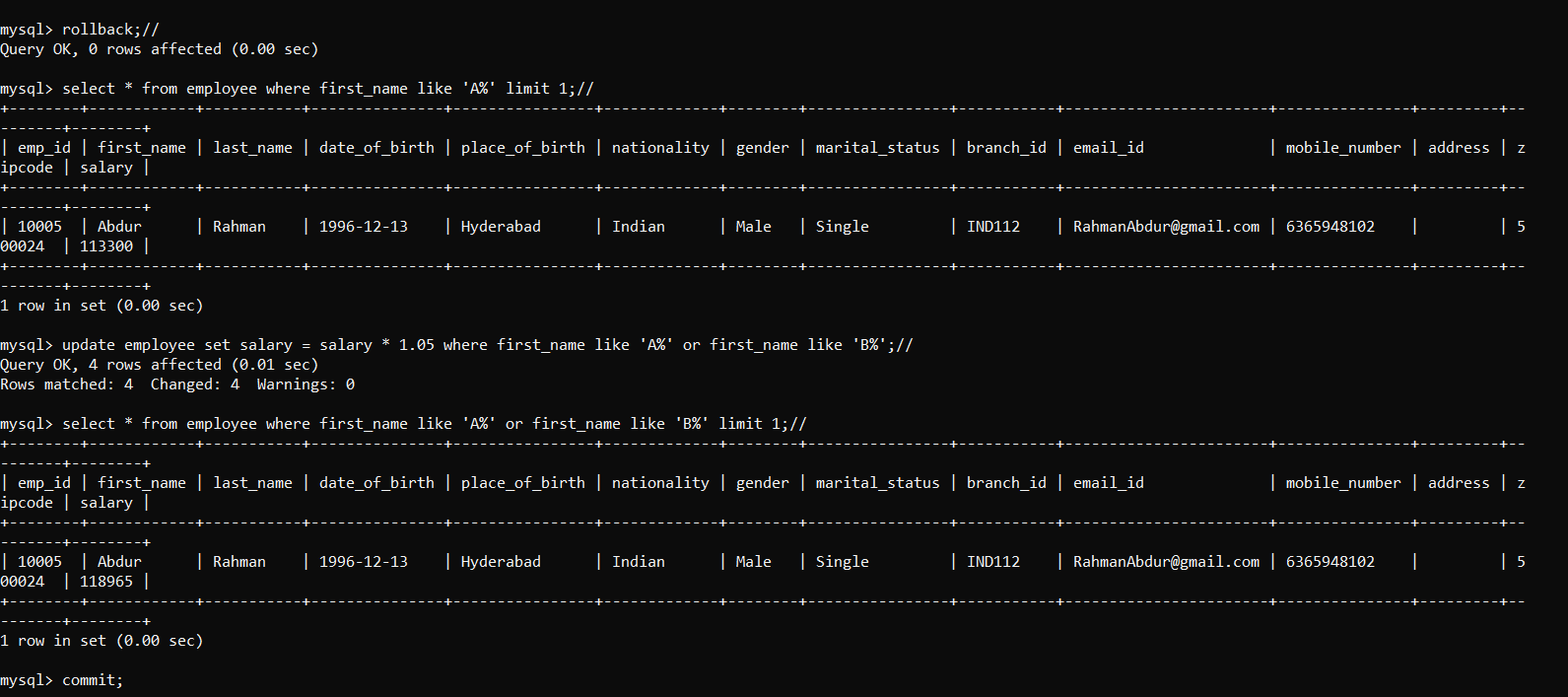
*Total points possible: 6*

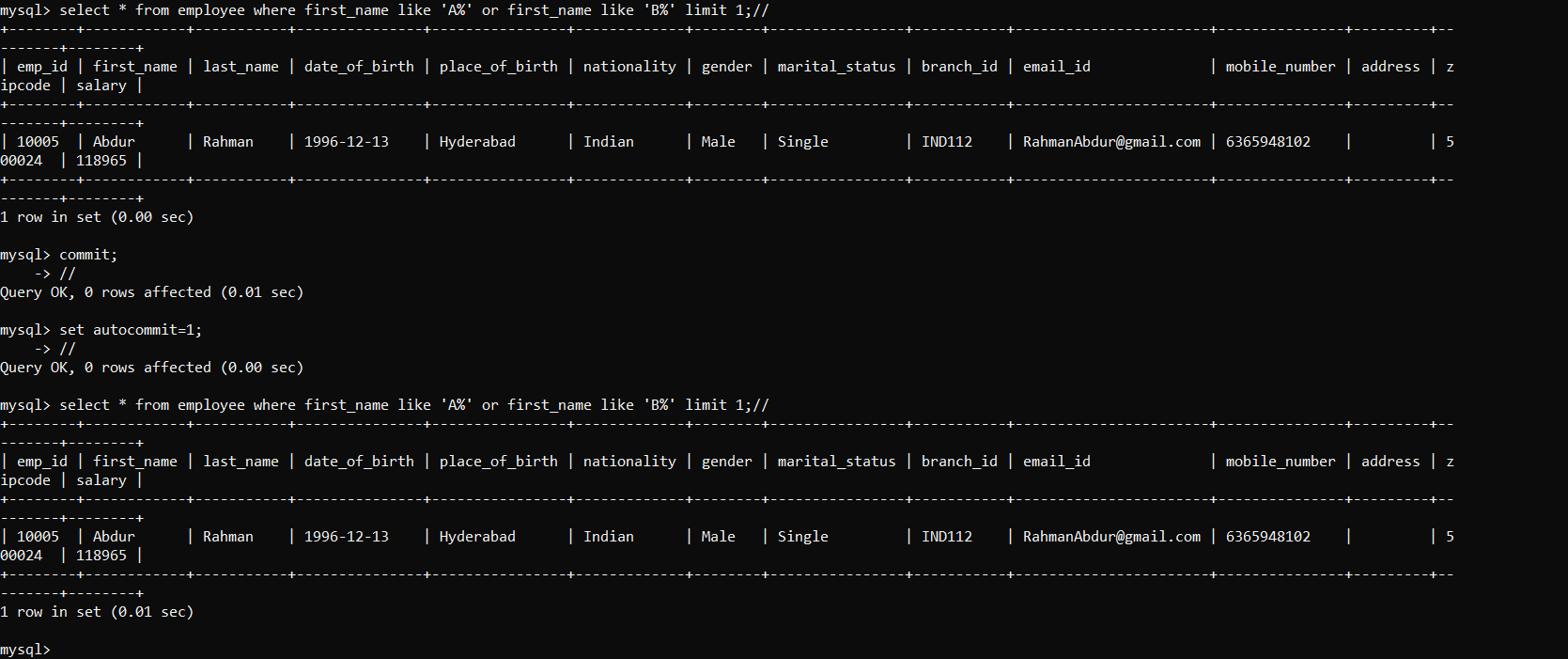
*Transactions are that important aspect of data base that holds on to a stuff that is being performed together sequentially and it can either be committed or rolled back to the point where it was earlier, during performing a transaction we can have the save points as well where we can rollback to instead of rolling back the entire transaction. We need to use this to make sure all our transactions are atomic, for example I have promoted an employee to senior position and later something happens and depending on that it feels like he needs to be taken back to the position he was working on initially then we can do that easily*

*Below am demonstrating the transaction property on my employee table for the column salary*

*I have given 3% hike to all employees whose name starts with A later I have decided to given them 5% hike instead of 3% and also to employees whose name starts with B then I’ll roll back the transaction done and will do the changes required post that I’ll verify and commit*







# Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 14*

*Various users that access the database are as follows:*

*All Employees will be having the read access over employee\_directory view but not on the tables  
All HR’s will be having access to read access to all Tables, views and write access over the below tables*

* + - 1. *Employee*
      2. *Emp\_address*
      3. *Emp\_education*
      4. *Official\_data*

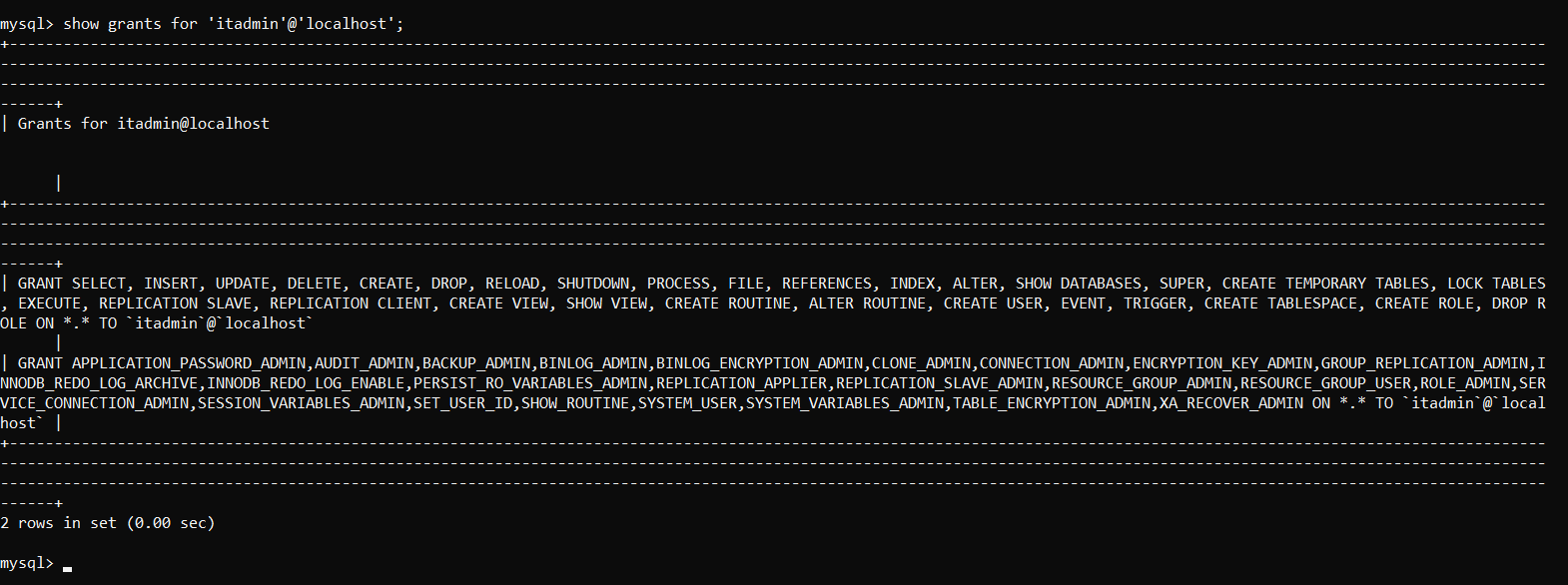
*Manager’s will be having full read access over database and write access over company branch, project\_details*

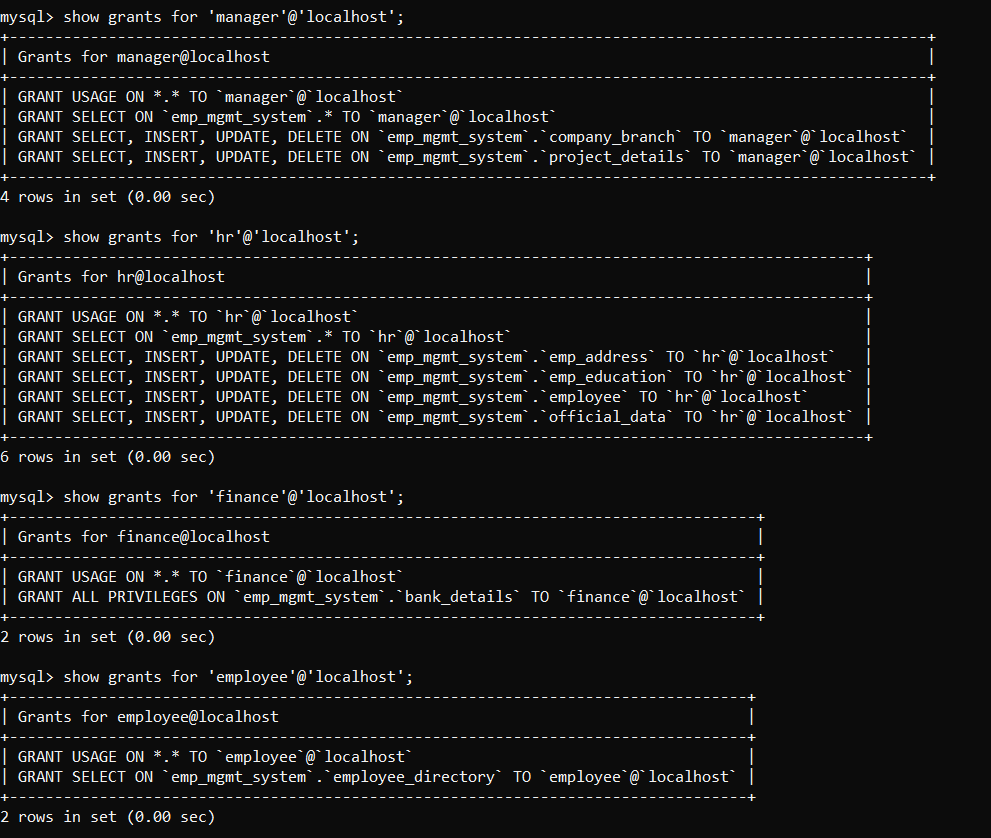
*Finance Team will be having full access over bank\_details*

*ITAdmin will be having full access over all the tables and views*

*Below file contains the user creation and grants commands for the users mentioned above and the screenshots depicts the tasks that a user can perform over the respective tables*

**





# Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

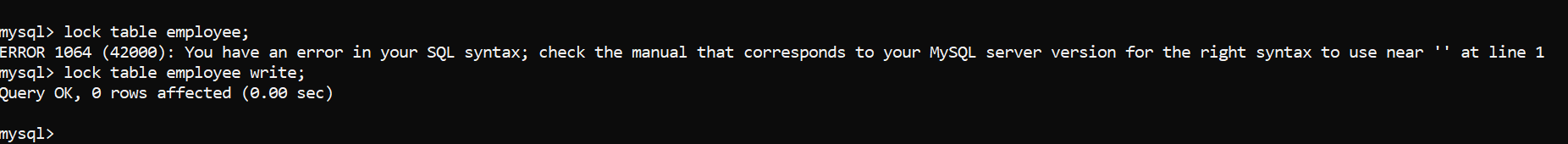
*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *3 points for providing a screenshot and accompanying explanation of locking tables.*

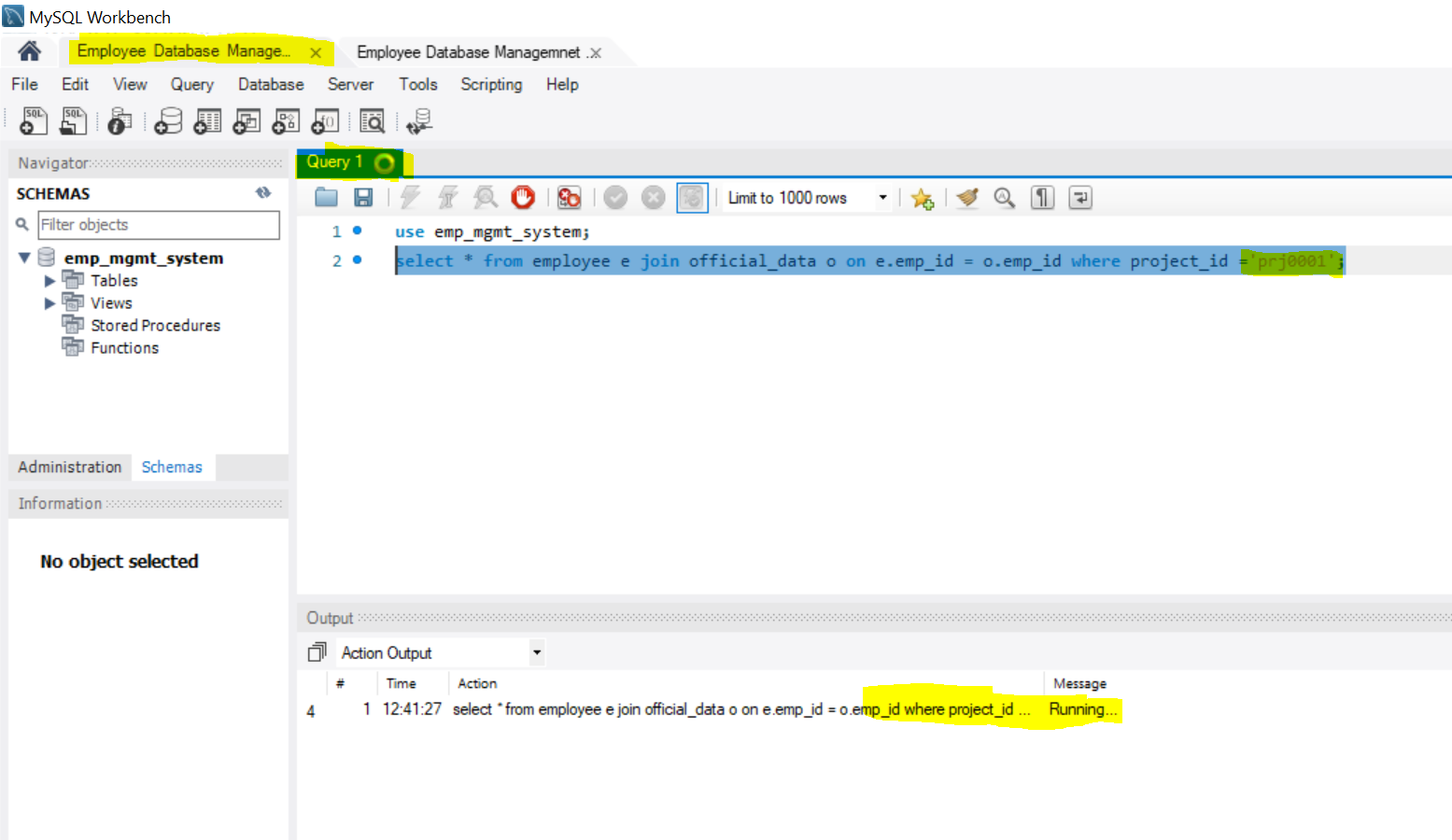
*Total points possible: 6*

*To best describe the Locking and concurrent access situation in my database system I’ll take one simple example consider there’s a hike cycle going on for all the employee’s and HR is updating the employee table with latest salaries for all employee so HR will lock the employee table now the manager’s of two different projects want to check if the salary hike that they have given to their respective employees has been properly reflected or not and tries to access the employee table but since the table is having a lock both the managers queries will get stuck as those are in wait state and once the lock is released then it’ll fetch the output. Suppose if the lock would have not been done by the HR then the managers would have been able to fetch the data which will be inconsistent as the table is still getting updated by HR. So we could conclude on this that how we can avoid showing the inconsistent data to the users by locking the important tables while doing the updates.*

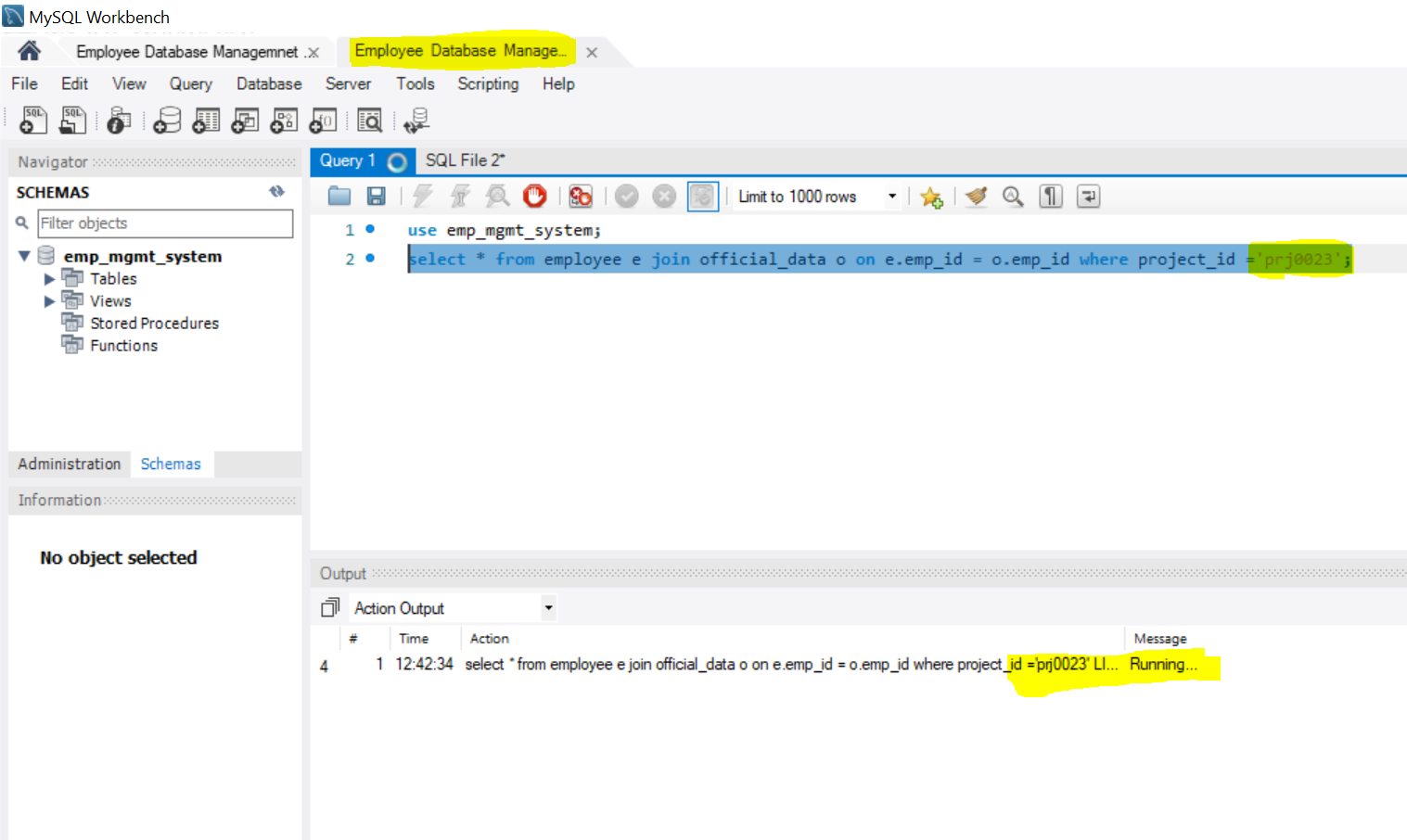
*Above scenario I have explained below using the screenshots i.e., we need to open three sessions one for HR and two for manager. Now the HR locks the table employee as it wants to update the salary*



*In below the Manager1 is accessing the employee table to see if salary change is reflected but it query stuck in running state as its waiting for lock to be released*



*In below the Manager2 in another concurrent session is accessing the employee table to see if salary change is reflected but its query also gets stuck in running state as its waiting for lock to be released*



# Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

* *12 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *3 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 15*

*Coming to the database backup strategy we have two methods that we can go for i.e., using “MySQL Enterprise Backup” which will take the physical backup of the database also we have mysqldumps which created the online logical backup and stores in .sql file which contains a set of insert statements which can be used to load the data at a later time. This operation performs a global read lock on all tables we can take incremental as well as the full backups but taking full back is not convenient always as it produces huge amount of data but incremental backups use the binary logs generated, I would prefer the mysqldump method as that’s very easy and can also be ran from command line and as our data is small we are going for full backup.*

***Command used for the backup:*** *mysqldump -u itadmin -p emp\_mgmt\_system -r "C:\Users\shussaini7\MySqlBackup\_EMS\backup.sql";*

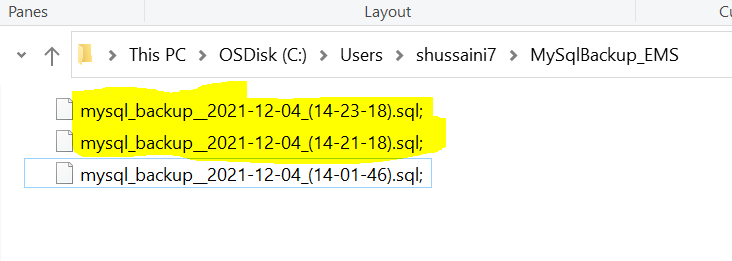
***Loaction for the backup storage:*** *C:\Users\shussaini7\MySqlBackup\_EMS\*

***Frequency of Backup to Demonstrate****: every 2 mins*

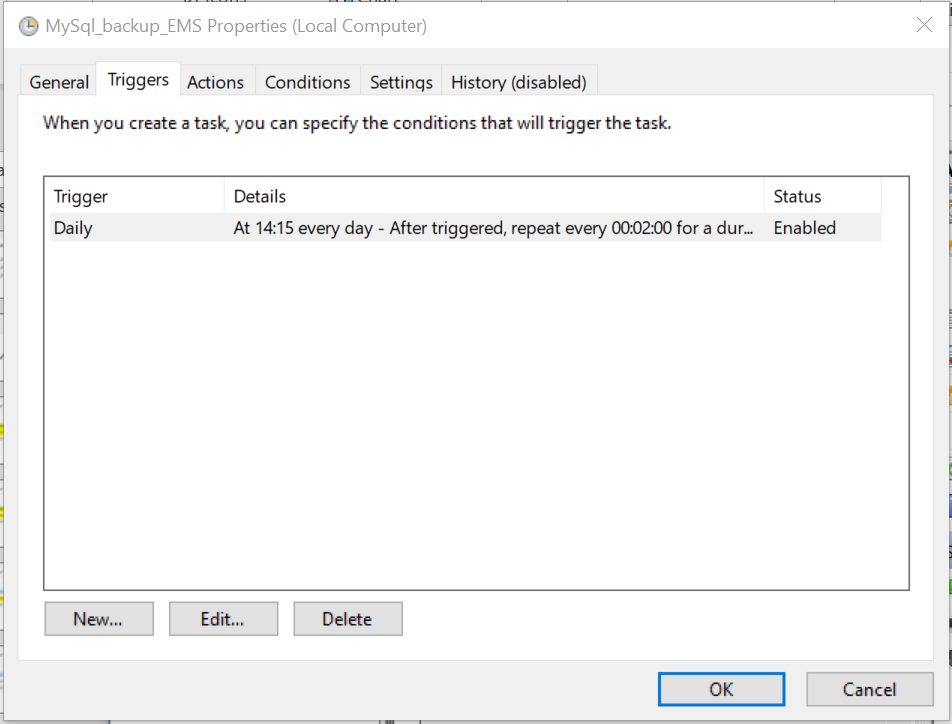
***Scheduling method used :*** *Windows task scheduler*

*Coming to the frequency of backup it generally depends on the kind of project you are handling or amount of data that you are receiving. For a real time projects the ideal backup time depends on* ***“How critical or sensitive the data is?”*** *We could generally go for a daily simple backup which will run every 24 hours or if the data is more critical then we could go for every 6 hours backup schedule*

*Here to demonstrate am going to schedule the backup for every 2 mins and the location that I’ll be using is my local disk only (****C:\Users\shussaini7\MySqlBackup\_EMS****\). If the database is larger then I would opt to go for backup in cloud environment like AWS as it’ll be convenient to do the backup in that. To schedule the automated backups, I’ll be creating a batch file i.e., “mysqlbackup\_script.bat” which will run on the frequency which we have set it to. There are several ways again to schedule the backups automatically like task manager, system scheduler etc., I’ll be using the task scheduler instead of installing any other third-party software. Using the task scheduler, I’ll be running the backup for the interval of 2 mins to display that my program for taking backup is running successfully. Below is the e screenshot attached with the details of the backups taken and also attaching the script used to take backup.*



*Below is the screenshot of the task scheduler depicting the scheduled run of the program*



* ( change the extension to .bat and place in appropriate location for running the backup on schedule)*

# Python Programming

*Description: Write a Python program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *12 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *4 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON DATABASE PROGRAMMING WORK HERE

# PHP Programming

*Description: Build an HTML form that enables the user to specify criteria to search by. Use PHP to show the results of the query on a resulting web page. Make sure you include protections against an SQL injection attack. Include your HTML and PHP code in your Word document, and also post the files to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *4 points for writing an HTML form the user will use to enter search criteria*
* *8 points for a PHP script that uses the search criteria and returns results*
* *4 points for an HTML page that shows the results*
* *4 points for explaining what SQL injection might be run on your website and explaining how you prevented it.*
* *4 points for providing screen shots of your PHP website in action.*
* *2 points for posting your code to GitHub*

*Total points possible: 26*

ENTER YOUR PHP DATABASE APP PROGRAMMING WORK HERE

# Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the* *advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

***Limitations of database:***

*The database that am using is a simple MySQL data base having very few tables and couple of views, this database files I mean the size of the table is determined by the operating system which is a constraint that we can have only limited data in it. If we are going for large organization having thousands of employees, we can face a lot of issues. Apart from this there’s also a limit on the identifier lengths that we can ignore as it’s not that big issue for us, also we can have maximum rows in MySQL up to 65,535 bytes only. If at all we have huge data and huge memory is required to run the query, then MySQL is very bad in restricting to memory as it’ll try to allocate all the memory required to run the query suppose if your system runs out of memory then the MySQL Crashes instead of terminating the query. Backup & restore becomes an issue as it may require us to freeze a master for a backup.*

***Solutions to the shortcomings along with data migration to Cloud:***

*To Improve the design of the existing database that I have created I think if we put a thought through the entire project again, we may optimize the tables furthermore like by making few of the columns derived and after doing that tables can further be normalized. If we think at the granular level considering a major organization in our mind, then we may get a lot of more attributes and the data which is with respect to employees and company that we could take up and work on. It totally depends and matters how our thinking works considering the data given with respect to the organizations size.*

*In order to overcome the drawbacks of the architectural level and its limitations we may go for various other databases like NoSQL (MongoDB, Redis, AWS Aurora etc.,) which provides us sufficient memory for both storage and processing of the queries and if we take the AWS Aurora or RDS on AWS we can easily migrate our data and the schema to that. Even if we must do any schema changes that seems to be difficult then AWS is having SCT (Schema Conversion Tool) which can take care of the schema conversion and data will be loaded smoothly also it provides various number of advantages to us like we don’t need to worry about our software anymore as all the upgrades, patches, maintenance will be taken care by Amazon itself. The database will be very faster due to high availability and scalability with higher security than what we have in ordinary db. As the AWS RDS/AWS Aurora is too compatible with Mysql, PostgreSQL it’ll be very easy for us to migrate the data and fix all the issues that we have faced.*

***Advantages and disadvantages of storing data in a document-based NoSQL format:***

*If we are using the NoSQL document DB then for sure we’ll be having a lot number of advantages starting with its schema less feature which can store any number of fields varying from one document to another. There’ll be no complex joins, will be very easy to scale as well.*

*For the working sets it’ll be using the in-memory which makes it faster while executing the queries and providing the facility of deep querying. The document DB will be almost 100 times faster than traditional RDBMS*

*Coming to the disadvantages of using documentDB the very first point which I want to highlight is that its not at all compatible with the SQL so all the design and architecture has to be re organized and re architect. When we compare this to RDBMS, these will be having very less built in functionalities which may require us to have our own code for example to tackle the ACID properties we need to have our own code written as the NoSQL databases basically doesn’t support ACID properties. We cannot run several keys simultaneously restricting our query limited to only one key at that moment.*

*For highly interconnected data, the document model is awkward whereas the RDBMS will be acceptable*

# Activity Log

*Description: As an appendix, the team will keep a daily diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since we have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

|  |  |  |
| --- | --- | --- |
| S. No | Activity | Week |
| 1 | Idea Evaluation | Week 1 and 2 |
| 2 | Initial Proposal |
| 3 | Data source identification |
| 4 | data gathering with the perspective of employee |
| 5 | data gathering with perspective of HR and manager |
| 6 | Identification of alternate ways to store data |
| 7 | Identification of Functional dependencies | Week 3 and 4 |
| 8 | Relationship between entity sets |
| 9 | Creation of ER Diagrams |
| 10 | Identification of various types of attributes |
| 11 | Understanding the concept of Normalization and implementing it |
| 12 | Creation of source for bridge entity set ( project\_details ) |
| 13 | Identification of Foreign keys in the entities |
| 14 | Designing the Physical Data Model using Vertabelo |
| 15 | Data definition language scripts using Vertabelo |
| 16 | Data manipulation language scripts using Vertabelo |
| 17 | DDL execution |
| 18 | DML Execution |
| 19 | Indexes | Week 5 .and 6 |
| 20 | View Creation |
| 21 | Trigger |
| 22 | Transactions |
| 23 | Security |
| 24 | Locking and concurrency |
| 25 | Backup and Scheduling |
| 26 | Suggested Future Work |