

A DATABASE DESIGN FOR FORTUNE X

by

MOHAMAD SABER DARWISH

MOHAMAD TAHA

ALI HARKOUS

A REPORT

Submitted to DR. RAMZI HARATY in partial fulfillment of the requirements

for the course Database Management Systems in Computer Science

Meta X – Phase IV

April 29, 2022

*This page intentionally left blank*

**Table of Contents:**

[II-Introduction: 6](#_Toc102090670)

[III- System Description and Constraints: 7](#_Toc102090671)

[IV-ER Diagram: 9](#_Toc102090672)

[V- Entity Types: 10](#_Toc102090673)

[1) Guest 10](#_Toc102090674)

[2) Employee 11](#_Toc102090675)

[3) Branch 12](#_Toc102090676)

[4) Department 13](#_Toc102090677)

[5) Room 14](#_Toc102090678)

[6) Service 15](#_Toc102090679)

[7) Privilege 16](#_Toc102090680)

[8) Invoice 17](#_Toc102090681)

[9) Expense 18](#_Toc102090682)

[10) Dependent 19](#_Toc102090683)

[VI- Relationships: 20](#_Toc102090684)

[1) Creates 20](#_Toc102090685)

[2) Works\_For 20](#_Toc102090686)

[3) Manages 21](#_Toc102090687)

[4) Has A1 21](#_Toc102090688)

[5) Has A2 22](#_Toc102090689)

[6) Supervision 22](#_Toc102090690)

[7) Works\_At 23](#_Toc102090691)

[8) Pays 23](#_Toc102090692)

[9) Attends 24](#_Toc102090693)

[10) Books 24](#_Toc102090694)

[11) Belongs\_To 25](#_Toc102090695)

[12) Has A4 25](#_Toc102090696)

[13) Has A3 26](#_Toc102090697)

[14) Dependents\_Of 26](#_Toc102090698)

[VII- Conclusion: 27](#_Toc102090699)

[VIII- ER to Relational Mapping Algorithms 28](#_Toc102090700)

[STEP 1: Mapping of Regular Entity Types 29](#_Toc102090701)

[STEP 2: Mapping of Weak Entity Types 32](#_Toc102090702)

[STEP 3: Mapping of Binary 1:1 Relation Types 32](#_Toc102090703)

[STEP 4: Mapping of Binary 1:M Relationship Types 34](#_Toc102090704)

[STEP 5: Mapping of Binary M:N Relationship Types 37](#_Toc102090705)

[STEP 6: Mapping of Multivalued Attributes 38](#_Toc102090706)

[STEP 7: Mapping of N-ary relationships 38](#_Toc102090707)

[FINAL STEP: Final Displays 39](#_Toc102090708)

[IX- Table Structure for Fortune X Hotel 41](#_Toc102090709)

[1- EMPLOYEE: 41](#_Toc102090710)

[2- GUEST: 42](#_Toc102090711)

[3- DEPARTMENT: 43](#_Toc102090712)

[4- ROOM: 43](#_Toc102090713)

[5- BRANCH: 44](#_Toc102090714)

[6- DEPENDENT: 44](#_Toc102090715)

[7- EXPENSE: 44](#_Toc102090716)

[8- INVOICE: 45](#_Toc102090717)

[9- HAS\_A3: 45](#_Toc102090718)

[10- HAS\_A4: 46](#_Toc102090719)

[11- PHONE\_NUMBER: 46](#_Toc102090720)

[12- HAS\_A1: 47](#_Toc102090721)

[13- SERVICE 47](#_Toc102090722)

[14- PRIVILEGE: 48](#_Toc102090723)

[X- Table Description: 49](#_Toc102090724)

[1- EMPLOYEE: 50](#_Toc102090725)

[2- GUEST: 51](#_Toc102090726)

[3- DEPARTMENT: 51](#_Toc102090727)

[4- ROOM: 52](#_Toc102090728)

[5- BRANCH: 52](#_Toc102090729)

[6- DEPENDENT: 52](#_Toc102090730)

[7- EXPENSE: 53](#_Toc102090731)

[8- INVOICE: 53](#_Toc102090732)

[9- HAS\_A3: 53](#_Toc102090733)

[10- HAS\_A4: 54](#_Toc102090734)

[11- PHONE\_NUMBER: 54](#_Toc102090735)

[12- HAS\_A1: 54](#_Toc102090736)

[13- SERVICE: 54](#_Toc102090737)

[14- PRIVILEGE: 55](#_Toc102090738)

[XI- Inserting Data: 55](#_Toc102090739)

[1- EMPLOYEE: 55](#_Toc102090740)

[2- GUEST: 58](#_Toc102090741)

[3- DEPARTMENT: 59](#_Toc102090742)

[4- ROOM: 60](#_Toc102090743)

[5- BRANCH: 61](#_Toc102090744)

[6- DEPENDENT: 62](#_Toc102090745)

[7- EXPENSE: 63](#_Toc102090746)

[8- INVOICE: 64](#_Toc102090747)

[9- HAS\_A3: 65](#_Toc102090748)

[10- HAS\_A4: 65](#_Toc102090749)

[11- PHONE\_NUMBER: 67](#_Toc102090750)

[12- HAS\_A1: 68](#_Toc102090751)

[13- SERVICE: 69](#_Toc102090752)

[14- PRIVILEGE: 69](#_Toc102090753)

[XII- Final Tables State: 71](#_Toc102090754)

[1- EMPLOYEE: 71](#_Toc102090755)

[2- GUEST: 71](#_Toc102090756)

[3- DEPARTMENT: 72](#_Toc102090757)

[4- ROOM: 72](#_Toc102090758)

[5- BRANCH: 73](#_Toc102090759)

[6- DEPENDENT: 73](#_Toc102090760)

[7- EXPENSE: 74](#_Toc102090761)

[8- INVOICE: 74](#_Toc102090762)

[9- HAS\_A3: 75](#_Toc102090763)

[10- HAS\_A4: 75](#_Toc102090764)

[11- PHONE\_NUMBER: 76](#_Toc102090765)

[12- HAS\_A1: 76](#_Toc102090766)

[13- SERVICE: 77](#_Toc102090767)

[14- PRIVILEGE: 77](#_Toc102090768)

[XIII- Queries: 78](#_Toc102090769)

[Query 1: 78](#_Toc102090770)

[Query 2: 79](#_Toc102090771)

[Query 3: 79](#_Toc102090772)

[Query 4: 80](#_Toc102090773)

[Query 5: 81](#_Toc102090774)

[Query 6: 81](#_Toc102090775)

[Query 7: 82](#_Toc102090776)

[Query 8: 82](#_Toc102090777)

[Query 9: 83](#_Toc102090778)

[Query 10: 84](#_Toc102090779)

[XIII- Normalization Up to The BCNF Normal Form: 85](#_Toc102090780)

[XIV- Conclusion: 94](#_Toc102090781)

[XV- Instructor’s Comments and Evaluation: 95](#_Toc102090782)

# II-Introduction:

“Meta X” is a company created by Mohamad Saber Darwish, Mohamad Taha, and Ali Harkous. This company offers a database design for a hotel and aims to satisfy customer’s preferences and requirements. Therefore, an ER diagram was implemented to build and maintain the database. In order to construct the ER diagram, we used draw.io which is a free and open-source cross-platform graph drawing software.

This report aims to discuss the entity types, relationships, and attributes constructed in the ER diagram, furthermore every entity type will be discussed separately with its relationships and attributes.

# III- System Description and Constraints:

Nowadays, people all around the world travel to other countries for several purposes including work, education, and vacation. These people need a place to sleep in and rest which is what hotels aims for. Hotels play a huge part in every country’s economy by generating income from tourists and creating many job opportunities.

Fortune X has a total of 7 branches located in different places in Lebanon as follows:

• 2 in Beirut

• 1 in Saida

• 1 in Sour

• 1 in Zahlé

• 1 in Baalbek

• 1 in West Beqaa

Each branch has its own departments, employees, and services.

As for departments, the hotel consists of 9 departments across all branches which are:

• Front Office

• Housekeeping

• Food and Beverage Service

• Kitchen

• Engineering and Maintenance

• Security

• HR

• Sales and Marketing

• Purchase and Store

The hotel’s staff consists of hundreds of employees across all branches and departments including:

• Managers

• Maintenance

• Kitchen and wait

• Front desk officers

• House Keeping

• Security members

As for the rooms, rooms consist different types such as single rooms, double rooms, and queen rooms including different numbers of beds in each. Each room is identified by a three numbered id based on the location of the room. The first number identifies the floor the room is located at while the last two numbers identify each room on the floor.

The hotel also provides several services that defer between branches including:

• Gyms

• Restaurants

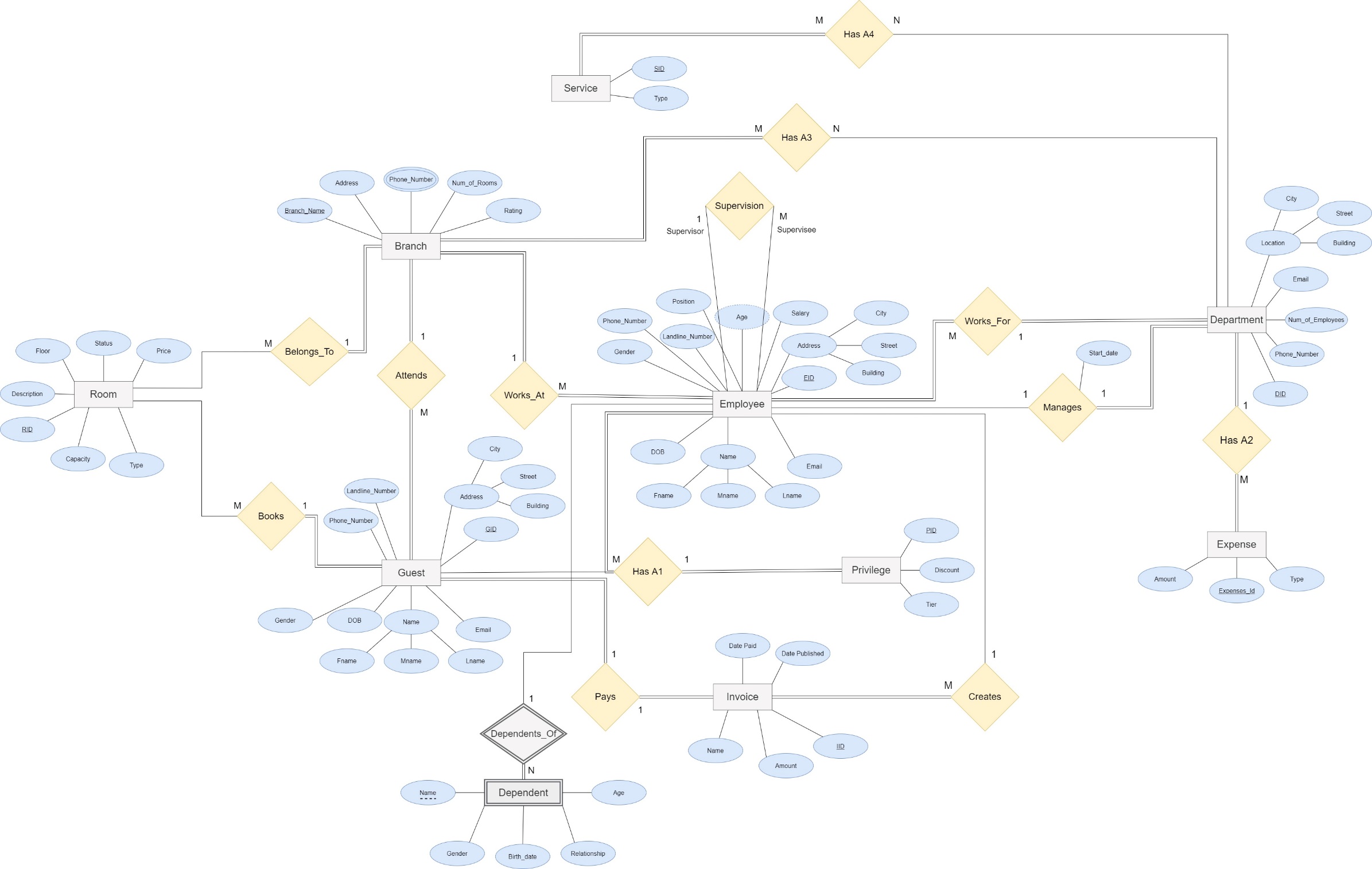
• Spa

• Pools

• Parking

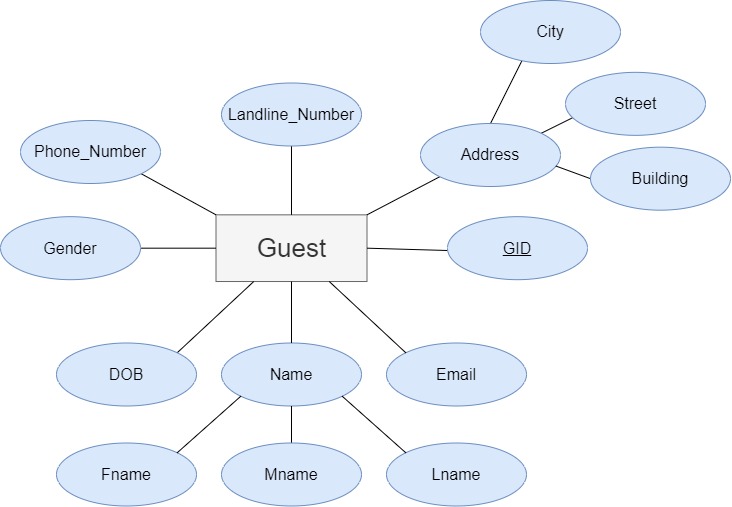
• Room Service

# IV-ER Diagram:



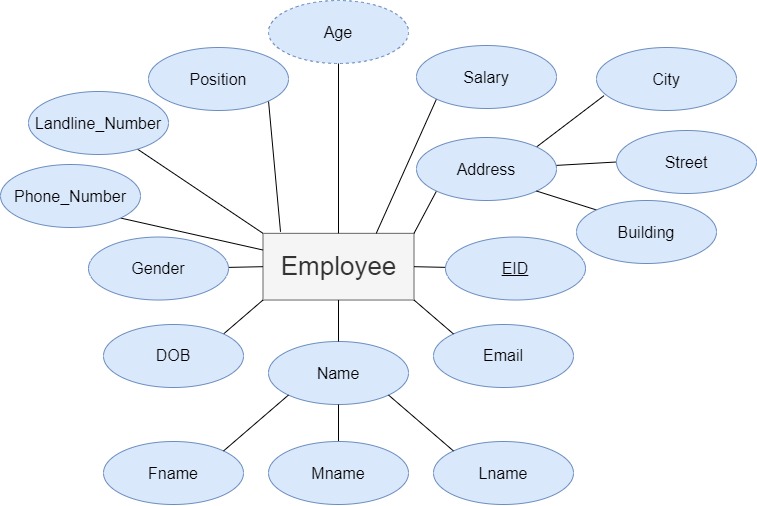
# V- Entity Types:

## Guest



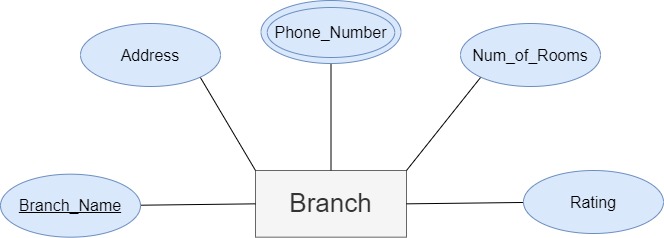
In this database, each guest has a unique id, GID which is considered a key attribute in the **Guest** entity. In addition, we save the **Guest**’s Name as a composite attribute: Fname, Mname, and Lname, and the **Guest**’s Address as composite (City, Street, and Building). This database helps the hotel to contact the guest by Email, Phone\_Number, Landline\_Number. The **Guest** entity has many other attributes such as the date of birth saved as DOB and the Gender.

## Employee



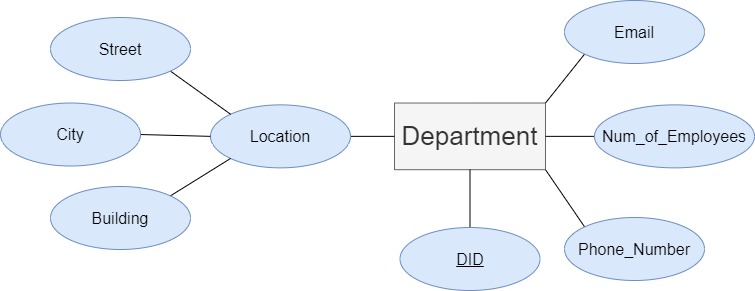
In this database, each **Employee** has a unique id, EID, which is considered a key attribute in the **Employee** entity. In addition, we save the **Employee**’s Name as a composite attribute: Fname, Mname, and Lname. The **Employee** entity type has many other attributes such as date of birth saved as DOB and Age which is derived from DOB, and of course we have the Gender of the employee. Moreover, another composite attribute for **Employee** collected is the employee’s Address, composed from it are City, Street, and Building. Furthermore, the hotel can access the salary and the position of the employee through their respective attributes designated as Salary and Position, respectively. This database helps the hotel to contact the employee using the attributes in-charge of their contact information (Email, Phone Number, Landline Number) which are Email, Phone\_Number, Landline\_Number of the employee.

## Branch



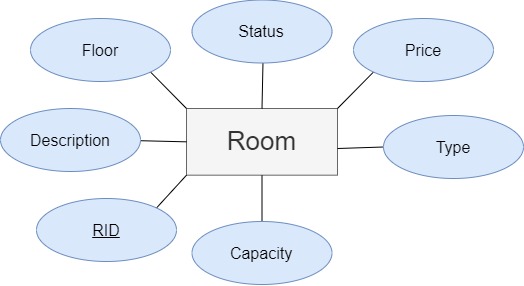
The hotel consists of several branches located all around Lebanon each defined by a Branch\_Name as a key attribute. Moreover, each entity **Branch** has an Address that shows where the branch is located, a Multivalued attribute Phone\_Number used to contact each location, and an official Rating by the Ministry of Tourism. Each branch also contains an attribute Num\_of\_Rooms that identifies the number of rooms in each branch.

## Department



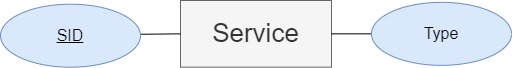
There are several departments needed to build a well-structured hotel. Thus, a **Department** entity type is needed. A department is identified by a key attribute DID, which is an ID that defers between every department in the hotel. Each department also has a Phone\_Number and Email used for contact as well as a Location which is a composite attribute composed of Street, City, and Building. The department also has a Num\_of\_Employees attribute that shows the number of employees that work at each department.

## Room



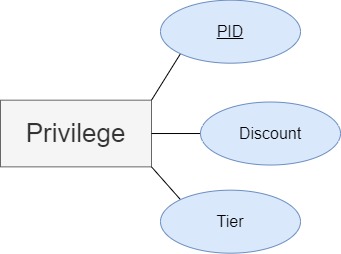
Every hotel has multiple rooms available to book by the guests. Each **Room** has a unique ID RID, which is considered a key attribute in the **Room** entity. The entity **Room** has many attributes such as Floor and a detailed Description of the room. In addition, to simplify the search of rooms for the guests, two attributes are saved under Type and Capacity. Moreover, before booking a room, it needs to be available, which is saved in the Status attribute, and the guest should check the Price as well before booking the room.

## Service



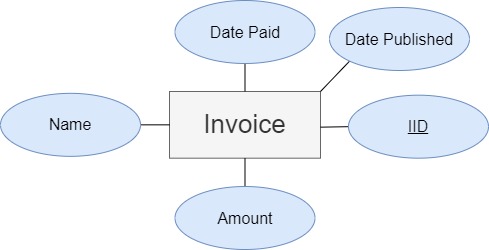
Each hotel offers several services to its guests to use throughout the day. The **Service** entity is composed of a key attribute SID, which is an ID that defers between the services, and a Type that identifies what the service is.

## Privilege



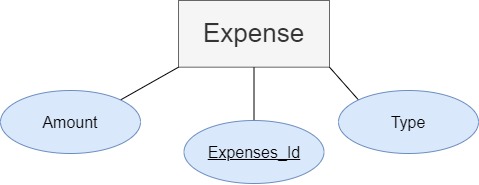
Like in every hotel, a regular guest gets more privileges from a hotel than other guests. Each Tier (silver, gold, platinum) has a unique ID, PID, which is considered a key attribute in the **Privilege** entity type, linked to a Discount.

## Invoice



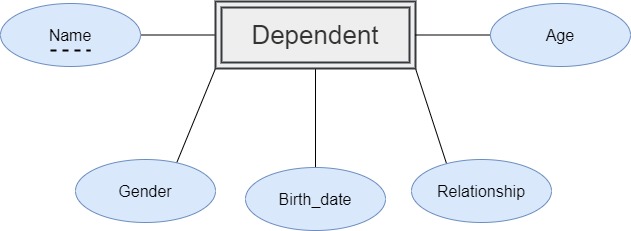
The **Invoice** entity type is used to track the payments the guests spend in their stay at the hotel. Each **Invoice** has a key attribute IID, which is an ID for each invoice. Invoices are labeled by a Name and Amount as well as a Date Published and Date Paid used to track each invoice.

## Expense



An **Expense** entity type is used to track the number of payments each department needs to pay. The **Expense** entity type is identified by an Expenses\_id that defers each expense from the other. Expenses also have an Amount that shows the cost of each expense and a Type that identifies the type of expense.

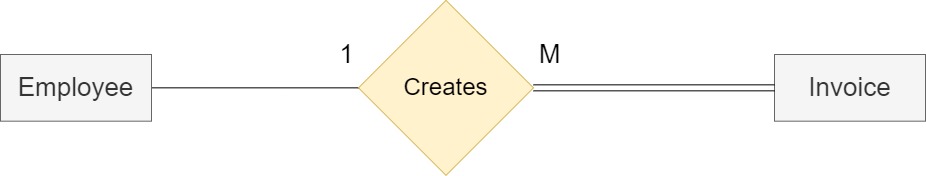
## Dependent



Each employee and guest have a dependent who the hotel can contact in case of emergency. The **Dependent** entity type is weak because it doesn’t have a key attribute. However, the **Dependent** entity type is identified by the dependent’s Name, which is a partial key and the specific id with whom the dependent is related, through the attribute Relationship. In addition, the **Dependent** entity type also has Birth\_date, Gender, and Age as attributes.

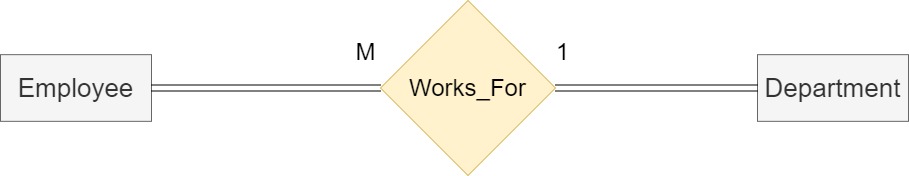
# VI- Relationships:

## Creates



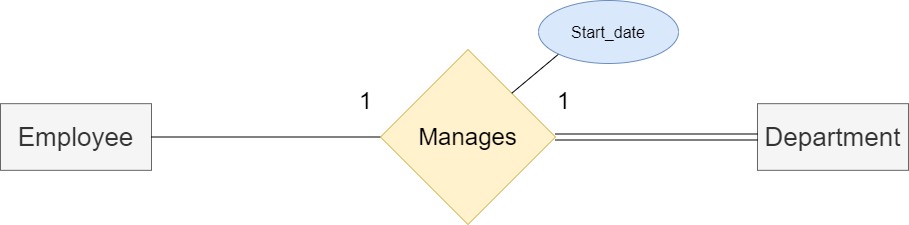
The invoices that guests pay at the hotel are created by employees. Thus, we need a Creates relationship between **Employee** and **Invoice** entities. Each employee creates more than one invoice, and each invoice can only be created by one employee. Thus, we have a one-to-many cardinality between both entities. Not all employees can create invoices, but all invoices should be created by employees. Thus, we have a partial participation from **Employee** and a total participation from **Invoice**.

## Works\_For



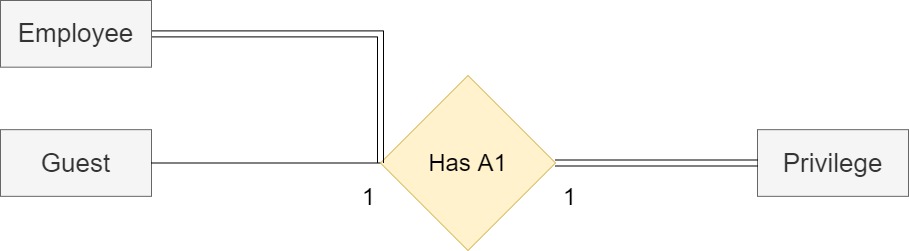
Each employee specializes in different subjects, thus a Works\_For relationship is needed between **Employee** and **Departmen**t entities should be utilized. Many employees can work at a department however no employees can work in several departments therefore we have a Many-to-one cardinality between the two entities. Since every employee needs to work at a department and every department has employees, we thus have a total participation from both entities.

## Manages



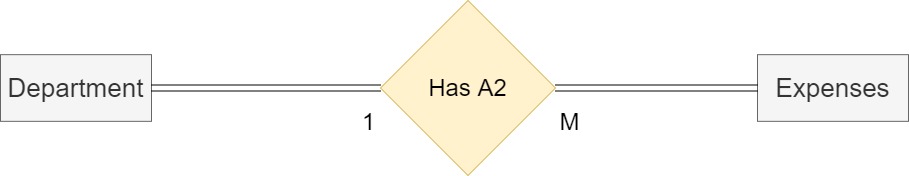
Every Department should be managed by an employee, thus a Manages relationship was needed between **Employee** and **Department** entity types. One employee is needed to manage a department and each department needs one manager. Thus, we have a one-to-one cardinality between both entities. Not all employees need to manage a department; however, all departments need to be managed by an employee thus we have a total participation from the **Department** entity type while a partial participation is needed from the **Employee** entity type. When an employee manages a department, a start date is recorded, therefore the Manages relationship has a **Start\_date** attribute.

## Has A1



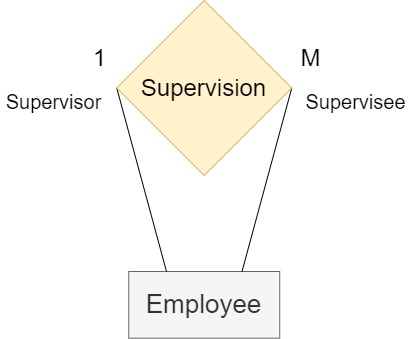
Some guests and employees should have discounts and privileges over others which leads us to a Has A1 relationship between **Guest** and **Privilege** on one hand, and **Employee** and **Privilege** on the other hand. Many employees and guests have one privilege, and each privilege has many guests and employees, thus we have a Many-to-one cardinality between all sides. All privileges have guests and employees, all employees have privileges; however, not all guests have privileges which leads us to the conclusion that there exists a total participation from **Employee** side, a partial participation from **Guest** side, and a total participation from **Privilege** side.

## Has A2



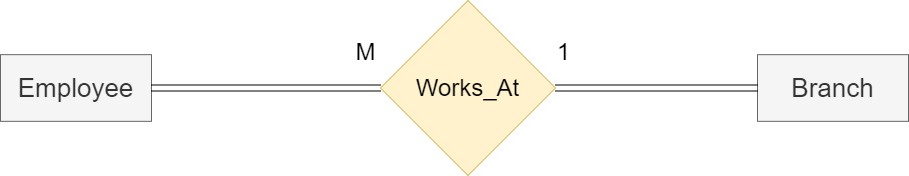
Every department has expenses. Thus, there exists a relationship Has A2 between **Department** and **Expense** entity types. Every department has many expenses, and many expenses have one department, which leads us to a one-to-Many cardinality between both entity types. All departments have expenses, and all expenses belong to departments which leads to total participation from both sides.

## Supervision



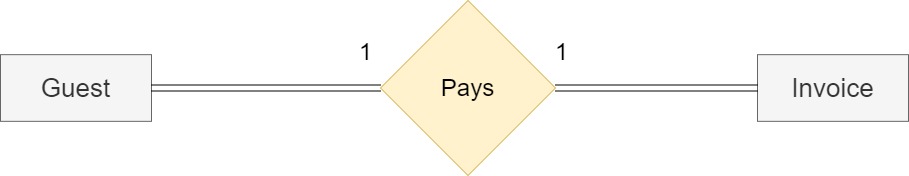
This is a recurrence relationship of entity **Employee** with itself. The employee can be supervised by another employee. One employee can supervise many employees and the employees have only one employee that supervises them, therefore we have a one-to-many cardinality. Not all employees supervise other employees and not all employees can be supervised since some of them might be in higher positions, so the relationship has partial participation from both sides.

## Works\_At



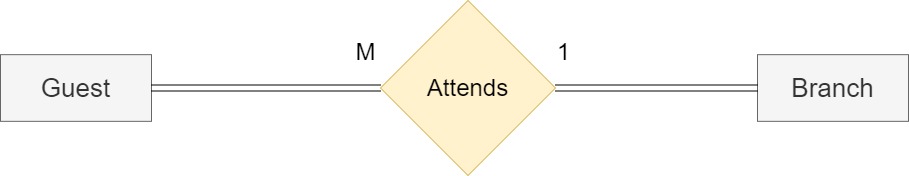
Many employees work at a branch; however, employees cannot work at more than one branch. Thus, we have a Many-to-one cardinality between **Employee** and **Branch** entity types through a Works\_At relation. All employees work at a branch and all branches also have employees. Thus, we have total participation from both sides.

## Pays



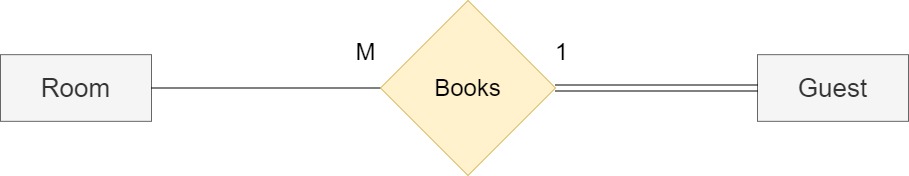
Guests should pay their bills through invoices at the hotel. Thus, there exists a Pays relationship between **Guest** and **Invoice** entity types. Each guest pays one invoice; therefore, we have a one-to-one cardinality between both entities. All Guests should pay invoices and all invoices should be paid by guest, which leads us to a total participation from both sides.

## Attends



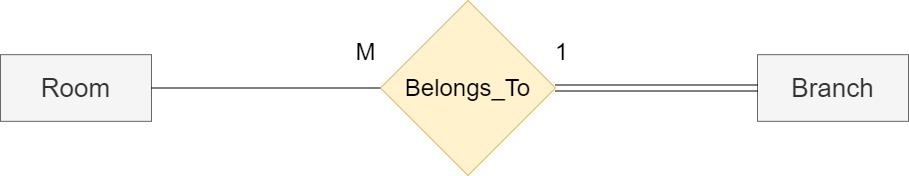
Guests should attend one of the hotel branches to spend the night therefore there exists an Attends relationship between **Guest** and **Branch** entity types. Many Guests can attend a branch of the hotel; however, same guests cannot attend more than one branch at the same time thus we have a Many-to-one cardinality between both entities. All guests should attend branches and all branches should have guests which leads us to a total participation between both sides.

## Books



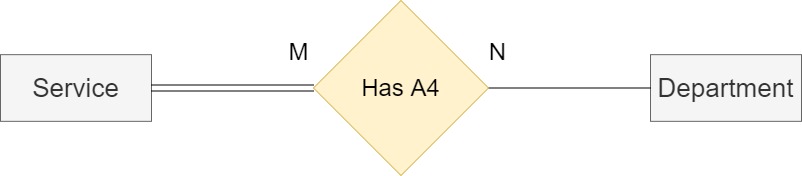
Guests need to book a room at the hotel therefore a Books relation is needed between **Room** and **Guest** entity types. Each guest can book more than one room; however, one room cannot be booked by more than one guest therefore we have a one-to-many cardinality. On the other hand, every guest should book a room; however, not all rooms should be booked at all times. Thus, we have a total participation from the **Guest** side while we have a partial participation from **Room** side.

## Belongs\_To



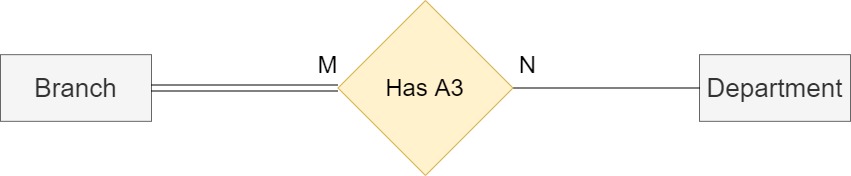
Every room belongs to a branch so a “Belongs to” relationship is needed between the **Room** entity type and **Branch** entity type. A branch has many rooms, and many rooms belong to the same branch and thus the cardinality is many-to-one. All rooms belong to a branch; however, no room belongs to more than one branch. Then the participation is total on the side of the Branch entity and partial on the side of the Room entity.

## Has A4



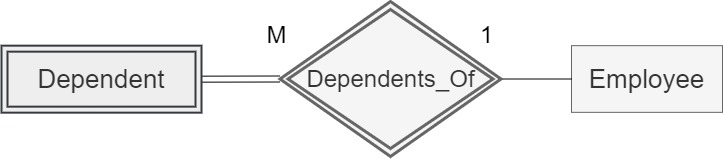
Every Hotel has services to provide for it guests thus, there exists a Has A4 relationship between **Department** and **Service** entity types. Many services are provided at departments and many departments have services which leads us to a Many-to-Many cardinality between both entity types. All services are provided in specific departments however not all departments offer services therefore we have a total participation from **Service** side and partial participation from **Department** side.

## Has A3



Departments are the main structure of every hotel; thus, the Has A3 relation between **Department** and **Branch** entity types is needed. Many departments are found at branches and many branches contain departments thus we have a Many-to-Many cardinality between both sides. All branches contain departments however not all departments are found in all branches which concludes a total participation from **Branch** side and a partial participation from **Department** side.

## Dependents\_Of



Employees have dependents therefore there exists a relationship Dependents\_Of between **Employee** and **Dependent**. Every employee can have many dependents, and every dependent has one employee. Thus, there is a one-to-Many relation between **Employee** and **Dependent.** All dependents have one employee; however, not all employees have dependents thus there is a total participation from **Dependent** entity, and partial participation from the other side.

# VII- Conclusion:

Databases are important in our daily lives. They organize and manage all the essential information that are required by different infrastructures. Hotels are one of those infrastructures that require databases to manage and store their data. This Database is essential for the optimal functioning, administration, and economic fruition of a hotel. Hotels are establishments that are designed to provide accommodation, meals, and other services for travelers and tourists. Through this database, we can show the organization of a hotel and its structure.

# VIII- ER to Relational Mapping Algorithms

After Designing the ER Diagram, the next step is to is to transform it into a Relational Model through a 7-step procedure which are:

* Step 1: Mapping of Regular Entity Types
* Step 2: Mapping of Weak Entity Types
* Step 3: Mapping of Binary 1:1 Relation Types
* Step 4: Mapping of Binary 1:N Relationship Types.
* Step 5: Mapping of Binary M:N Relationship Types.
* Step 6: Mapping of Multivalued attributes.
* Step 7: Mapping of N-ary Relationship Types

The following is a detailed explanation of the 7-step procedure applied to the ER Diagram.

## STEP 1: Mapping of Regular Entity Types

In this step, all regular entity types (Strong entity types) should be mapped into relations that include all its simple attribute in addition to a single underlined primary key. The list of our regular entity types are **Guest**, **Employee**, **Branch**, **Department**, **Room**, **Service**, **Privilege**, **Invoice**, and **Expense**.

1. Guest

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GID | Email | Fname | Mname | Lname | DOB | Gender | Phone\_Number |
| Landline\_Number | City | Street | Building |

The **Guest** entity type consists of several simple attributes, two composite attributes, and an underlined primary key GID which are all included in the relation. The composite attribute Name includes Fname, Mname, and Lname while the composite attribute Address includes City, Street, and Building. The rest of the simple attributes are Email, DOB, Gender, Phone\_Number, and Landline\_Number.

1. Employee

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EID | Email | Fname | Mname | Lname | DOB | Gender |
| Phone\_Number | Landline\_Number | Position | Salary | City | Street | Building |
| Age |

The **Employee** entity type consists of several simple attribute, two composite attributes, a derived attribute Age, and one key attribute EID which will be the primary key in the above relation. The composite attribute Name includes Fname, Mname, and Lname while the composite attribute Address includes City, Street, and Building. Email, DOB, Gender, Phone\_Number, Landline\_Number, Position, and Salary are also added as simple attributes to this relation.

1. Branch

|  |  |  |  |
| --- | --- | --- | --- |
| Branch\_Name | Address | Num\_of\_Rooms | Rating |

The **Branch** entity type only consists of a primary key Branch\_Name in addition to the simple attributes Address, Phone\_Number, Num\_of\_Rooms, and Rating which all are presented in the above relation.

1. Department

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DID | Location | Phone\_Number | Num\_of\_Employees | Email |

This entity type consists of simple attributes and a primary key DID as represented in the above relation. The simple attributes of this relation are Location, Phone\_Number, Num\_of\_Employees, and Email.

1. Room

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RID | Capacity | Type | Price | Status | Floor | Description |

In this entity, RID is the primary key while all the other attributes Capacity, Type, Price, Status, Floor, and Description are simple attributes.

1. Service

|  |  |
| --- | --- |
| SID | Type |

The **Service** entity type is the simplest entity type in our Design. It consists of one primary key SID and one simple attribute Type.

1. Privilege

|  |  |  |
| --- | --- | --- |
| PID | Discount | Tier |

This entity type has a primary key PID, and two simple attributes Discount and Tier as shown in the above relation.

1. Invoice

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IID | Amount | Data Published | Date Paid | Name |

The **Invoice** entity type consists of several simple attributes alongside a Primary Key IID.

The simple attributes are Amount, Data Published, Date Paid, and Name.

1. Expense

|  |  |  |
| --- | --- | --- |
| Expenses\_Id | Exp\_Type | Amount |

The **Expense** entity type has a primary key Expenses\_Id and two simple attributes Exp\_Type and Amount

## STEP 2: Mapping of Weak Entity Types

In this step, the weak entity should be mapped into relations that includes all its simple attributes. Its primary key is the combination of the primary keys of the owner and the partial key of the weak entity type. The weak entity type in this ER diagram is **Dependent**.

1. Dependent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EID | Name | Gender | Birth\_date | Relationship |

The **Dependent** entity type consists of several simple attributes: Name, Gender, Birth\_date and Relationship. The EID, the primary key of the entity type **Employee**, is included. EID and Name are combined to represent the primary key of the relation.

## STEP 3: Mapping of Binary 1:1 Relation Types

In this step, one-to-one relationships are mapped into the tables. This step includes three methods that can be followed.

* Step 1: Foreign Key (two relations) method: In this method, we add the primary key of the first entity type as a foreign key to the other entity type with the total participation side of this relationship.
* Step 2: Merged relation (one relation) method: In this method, we merge both entity types that have a total participation on both sides of the relationship into a single relation
* Step 3: Cross-reference or relationship relation (three relations) method: In this method, a third relation is created that includes the primary key of the two other relations in addition to the relationship attributes.

1. Department (*Manages*)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DID | Location | Phone\_Number | Num\_of\_Employees | Email | Start\_date |
| Employee\_ID |

The “Manages” relationship links the **Employee** entity type and the **Department** entity type. The **Department** entity type is on the total participation side. Thus, we add to its relation the foreign key which is the primary key of the **Employee** entity type and is referred to as Employee\_ID. Moreover, we add the attribute Start\_date of the relationship “Manages”, so we added it to the relation.

1. Guest (*Pays*)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GID | Email | Fname | Mname | Lname | DOB | Gender | Phone\_Number |
| Landline\_Number | City | Street | Building | Invoice\_ID |

The “Pays” relationship links the **Guest** entity type and the **Invoice** entity type. Both entity types have total participation. Thus, we choose to add to the relation the foreign key, which is the primary key IID of the **Invoice** entity type and is referred to as Invoice\_ID.

## STEP 4: Mapping of Binary 1:M Relationship Types

In this step, one-to-many relationships are mapped into tables. The primary key of the one side of the relationship is added as a foreign key to the “many” side of the relationship in addition to any simple attributes of the relationship.

1. Expense (Has A2)

|  |  |  |  |
| --- | --- | --- | --- |
| Amount | Expense\_Id | Exp\_Type | Department\_Id |

Each department has many expenses. The “Has A2” relationship links the **Expense** entity type and the **Department** entity type. The **Expense** entity type is on the “many” side. Thus, we add to its relation the foreign key DID which is the primary key of **Department** entity type, and we rename it to Department\_Id.

1. Invoice (Creates)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Amount | IID | Date\_Published | Date\_Paid | Employee\_Id |

Each employee creates many invoices. The “Creates” relationship links the **Employee** entity type and the **Invoice** entity type. The **Invoice** entity type is on the “many” side. Thus, we add to its relation the foreign key EID which is the primary key of **Employee** entity type, and we rename it to Employee\_Id.

1. Employee (Works\_For)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EID | Fname | Mname | Lname | DOB | Gender | |
| Phone\_Number | Landline\_Number | Position | Salary | City | Street | Building |
| Age | Start\_date | Email | Department\_ID |

Many employees work for a department therefore, a “Works\_For” relationship between the **Employees** and **Department** entity types is needed. Since the **Employee** in on the “many” side of the relationship, we add the primary key of the **Department** entity type as a Foreign Key to the **Employee** renamed as Department\_ID.

1. Employee (Works\_At)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EID | Email | Fname | Mname | Lname | DOB | Gender |
| Phone\_Number | Position | Salary | City | Street | Building | |
| Landline\_Number | Start\_date | Age | Department\_ID | B\_Name |

Many employees work at a branch thus, a “Works\_At” relationship between **Employee** and **Department** entity types is needed. In this relationship, **Employee** is on the “many” side, thus the primary key of the **Branch** entity is added as a foreign key to the **Employee** and was renamed to B\_Name.

1. Employee(Supervises)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EID | Email | Fname | Mname | | Lname |
| Gender | Building | Supervisor\_ID | B\_Name | | Street |
| Phone\_Number | Landline\_Number | Position | Salary | | City |
| Age | Start\_date | Department\_ID | DOB |

Many employees are supervised by another employee therefore, a “Supervises” relationship is needed to link the supervisor to the supervisee in the **Employee** entity. Since supervisee is on the “many” side of the relationship, we add a Supervisor\_ID as a foreign key which represents the EID (Employee ID) which is the primary key in the **Employee** entity type.

1. Room (Books)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RID | Capacity | Type | Price | Status | Floor | Description | Guest\_ID |

Each guest can book many rooms. The “Books” relationship links the **Room** entity type and the **Guest** entity type. The **Room** entity type is on the “many” side. Thus, we add to its relation the foreign key GID which is the primary key of **Guest** entity type, and we rename it to Guest\_ID.

## STEP 5: Mapping of Binary M:N Relationship Types

In this step, many-to-many relationships are mapped into relations. The two primary keys of both entity types will be added as foreign keys to a new relation. The combination of the two foreign keys will be the primary key of the new relation. The relationships simple attributes will also be added to the new relation.

1. Has\_A3

|  |  |
| --- | --- |
| B\_NAME | DEPT\_ID |

Many branches have many departments. The “Has\_A3” relationship links the **Branch** entity type with the **Department** entity type. The relationship “Has\_A3” includes the primary key of the **Branch** and **Department** entity types which are B\_NAME and DEPT\_ID. The combination of both attributes constitutes the primary key of this relation.

1. Has\_A4

|  |  |
| --- | --- |
| DEPARTMENT\_ID | SERVICE\_ID |

Many departments have many services. The “Has\_A4” relationship links the **Department** entity type with the **Service** entity type. The relationship “Has\_A4” includes the primary key of the **Department** and **Service** entity types which are DEPARTMENT\_ID and SERVICE\_ID. The combination of both attributes constitutes the primary key of this relation.

## STEP 6: Mapping of Multivalued Attributes

In this step, we create a relation for every multivalued attribute. The primary key of the entity in which the multivalued attribute is located will be added to the new relation alongside the multivalued attribute itself. The combination of both will be the primary key of the new relation

1. PHONE\_Number

|  |  |
| --- | --- |
| BranchName | PhoneNumber |

The multivalued attribute PhoneNumber belongs to the **Branch** entity type. Thus, we create a relation called “PHONE\_Number”. Its primary key is composed of BranchName, the primary key of the **Branch** entity type, and the attribute PhoneNumber which represents the different phone numbers of each branch.

## STEP 7: Mapping of N-ary relationships

In this step, we are going to map the N-ary Relationship types. We should create a new relation containing the primary keys of all participating entity types and any simple attributes of the relationship type.

1. Has\_A1

|  |  |  |
| --- | --- | --- |
| GuestID | EmployeeID | PrivilegeID |

The trinary relation “HAS\_A1” links the **Guest**, **Employee** and **Privilege** entity types. From each entity type, we select its primary key, GuestID from **Guest**, EmployeeID from **Employee** and PrivilegeID from **Privilege** and add them as foreign key to this relation. The combination of all three foreign key constitutes the primary key of the relation “Has\_A1”.

## FINAL STEP: Final Displays

Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EID | Email | Fname | Mname | | Lname |
| Gender | Building | Supervisor\_ID | B\_Name | | Street |
| Phone\_Number | Landline\_Number | Position | Salary | | City |
| Age | Start\_date | Department\_ID | DOB |

Guest

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GID | Email | Fname | Mname | Lname | DOB | Gender | Phone\_Number |
| Landline\_Number | City | Street | Building | Invoice\_ID |

Department

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DID | Location | Phone\_Number | Num\_of\_Employees | Email | Start\_date |
| Employee\_ID |

Room

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RID | Capacity | Type | Price | Status | Floor | Description | Guest\_ID |

Branch

|  |  |  |  |
| --- | --- | --- | --- |
| Branch\_Name | Address | Num\_of\_Rooms | Rating |

Dependent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EID | Name | Gender | Birth\_date | Relationship |

Expense (Has A2)

|  |  |  |  |
| --- | --- | --- | --- |
| Amount | Expense\_Id | Exp\_Type | Department\_Id |

Invoice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Amount | IID | Date\_Published | Date\_Paid | Employee\_Id |

Has\_A3

|  |  |
| --- | --- |
| B\_NAME | DEPT\_ID |

Has\_A4

|  |  |
| --- | --- |
| DEPARTMENT\_ID | SERVICE\_ID |

PHONE\_Number

|  |  |
| --- | --- |
| BranchName | PhoneNumber |

Has\_A1

|  |  |  |
| --- | --- | --- |
| GuestID | EmployeeID | PrivilegeID |

Service

|  |  |
| --- | --- |
| SID | Type |

Privilege

|  |  |  |
| --- | --- | --- |
| PID | Discount | Tier |

# IX- Table Structure for Fortune X Hotel

After designing the ER diagram for Fortune X Hotel and mapping this diagram into relational database design, now it is time to start creating the concrete tables for our database on the Oracle Database Server. We will begin by creating all tables and then inserting data into them. Finally, we will execute some queries to show the value of the database in a plant nursery.

## EMPLOYEE:

CREATE TABLE EMPLOYEE

(

EID INT PRIMARY KEY NOT NULL,

Email VARCHAR(50) NOT NULL,

Fname VARCHAR(25) NOT NULL,

Mname VARCHAR(25),

Lname VARCHAR(25) NOT NULL,

Gender char(1),

Building VARCHAR(50),

Supervisor\_ID INT,

B\_Name VARCHAR(50),

Street VARCHAR(25),

Phone\_Number VARCHAR(14),

Landline\_Number char(8),

Position VARCHAR(25),

Salary INT,

City VARCHAR(25),

Age INT,

Start\_date DATE,

Employee\_ID INT,

Department\_ID INT,54

DOB DATE NOT NULL

);

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(Department\_ID)

REFERENCES Department(DID);

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(B\_Name)

REFERENCES Branch(Branch\_Name);

ALTER TABLE EMPLOYEE

ADD FOREIGN KEY(Supervisor\_ID)

REFERENCES Employee(EID);

## GUEST:

CREATE TABLE GUEST

(

GID INT PRIMARY KEY,

Email VARCHAR(50) ,

Fname VARCHAR(25) NOT NULL,

Mname VARCHAR(25),

Lname VARCHAR(25) NOT NULL,

DOB DATE NOT NULL,

Gender char(1),

Phone\_Number VARCHAR(14),

Landline\_Number char(8),

City VARCHAR(25),

Street VARCHAR(25),

Building VARCHAR(25),

Invoice\_ID INT

);

ALTER TABLE GUEST

ADD FOREIGN KEY (Invoice\_ID)

REFERENCES INVOICE(IID);

## DEPARTMENT:

CREATE TABLE DEPARTMENT

(

DID INT PRIMARY KEY NOT NULL,

Location VARCHAR(50),

Phone\_Number VARCHAR(14),

Num\_of\_Employees INT,

Email VARCHAR(50),

Start\_date Date,

Employee\_ID INT

);

ALTER TABLE DEPARTMENT

ADD FOREIGN KEY(Employee\_ID)

REFERENCES Employee(EID);

## ROOM:

CREATE TABLE ROOM

(

RID INT PRIMARY KEY,

Capacity INT ,

Type VARCHAR(25),

Price INT,

Status VARCHAR(10),

Floor INT,

Description VARCHAR(50)

);

## BRANCH:

CREATE TABLE BRANCH

(

Branch\_Name VARCHAR(50) PRIMARY KEY,

Address VARCHAR(25),

Num\_of\_Rooms INT,

Rating INT

);

## DEPENDENT:

CREATE TABLE DEPENDENT

(

EID INT,

Name VARCHAR(25),

Gender char(1),

Birth\_date Date,

Relationship VARCHAR(50)

);

ALTER TABLE DEPENDENT

ADD FOREIGN KEY(EID)

REFERENCES EMPLOYEE(EID);

## EXPENSE:

CREATE TABLE Expense

(

Amount INT,

Expense\_Id INT PRIMARY KEY,

Exp\_Type VARCHAR(50),

Department\_Id INT

);

ALTER TABLE EXPENSE

ADD FOREIGN KEY(Department\_Id)

REFERENCES DEPARTMENT(DID)

## INVOICE:

CREATE TABLE Invoice

(

Name VARCHAR(25),

Amount INT,

IID INT PRIMARY KEY,

Date\_Published Date,

Date\_Paid Date,

Employee\_Id INT

);

ALTER TABLE INVOICE

ADD FOREIGN KEY(Employee\_Id)

REFERENCES Employee(EID);

## HAS\_A3:

CREATE TABLE HAS\_A3

(

B\_NAME VARCHAR(25),

DEPT\_ID INT

);

ALTER TABLE HAS\_A3

ADD FOREIGN KEY(B\_NAME)

REFERENCES Branch(Branch\_Name);

ALTER TABLE HAS\_A3

ADD FOREIGN KEY(DEPT\_ID)

REFERENCES Department(DID);

## HAS\_A4:

CREATE TABLE Has\_A4

(

DEPARTMENT\_ID INT,

SERVICE\_ID INT

);

ALTER TABLE HAS\_A4

ADD FOREIGN KEY(DEPARTMENT\_;ID)

REFERENCES Department(DID);

ALTER TABLE HAS\_A4

ADD FOREIGN KEY(SERVICE\_ID)

REFERENCES Service(SID);

## PHONE\_NUMBER:

CREATE TABLE PHONE\_NUMBER

(

BranchName VARCHAR(50),

PhoneNumber VARCHAR (14)

);

ALTER TABLE PHONE\_NUMBER

ADD FOREIGN KEY(BranchName)

REFERENCES Branch(Branch\_Name);

## HAS\_A1:

CREATE TABLE HAS\_A1

(

GuestID INT,

EmployeeID INT,

PrivilegeID INT,

FOREIGN KEY(GuestID)

REFERENCES Guest(GID),

FOREIGN KEY(EmployeeID)

REFERENCES Employee(EID)

);

ALTER TABLE HAS\_A1

FOREIGN KEY(GuestID)

REFERENCES Guest(GID);

ALTER TABLE HAS\_A1

FOREIGN KEY(EmployeeID)

REFERENCES Employee(EID);

ALTER TABLE HAS\_A1

ADD FOREIGN KEY(PrivilegeID)

REFERENCES Privilege(PID);

## SERVICE

CREATE TABLE SERIVCE

(

SID INT PRIMARY KEY,

Type VARCHAR(25)

);

## PRIVILEGE:

CREATE TABLE Privilege

(

PID INT PRIMARY KEY,

Discount VARCHAR(4),

Tier VARCHAR(25)

);

# X- Table Description:

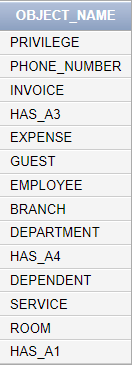
After creating all the tables on the oracle database server, we can now view the description of each table to make sure everything is working well, and no mistakes were made during the creation of tables.

In our database we have the following tables created on the oracle database server:

SQL> SELECT DISTINCT OBJECT\_NAME

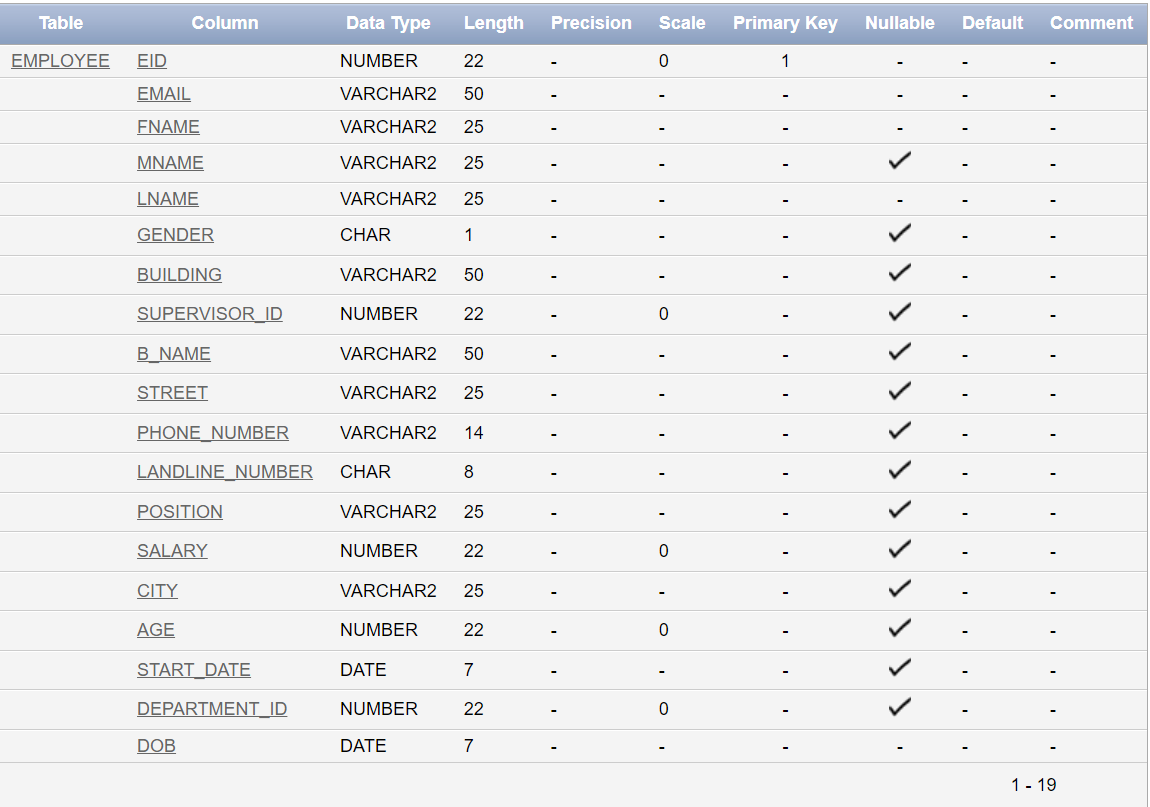
FROM USER\_OBJECTS

WHERE OBJECT\_TYPE='TABLE';



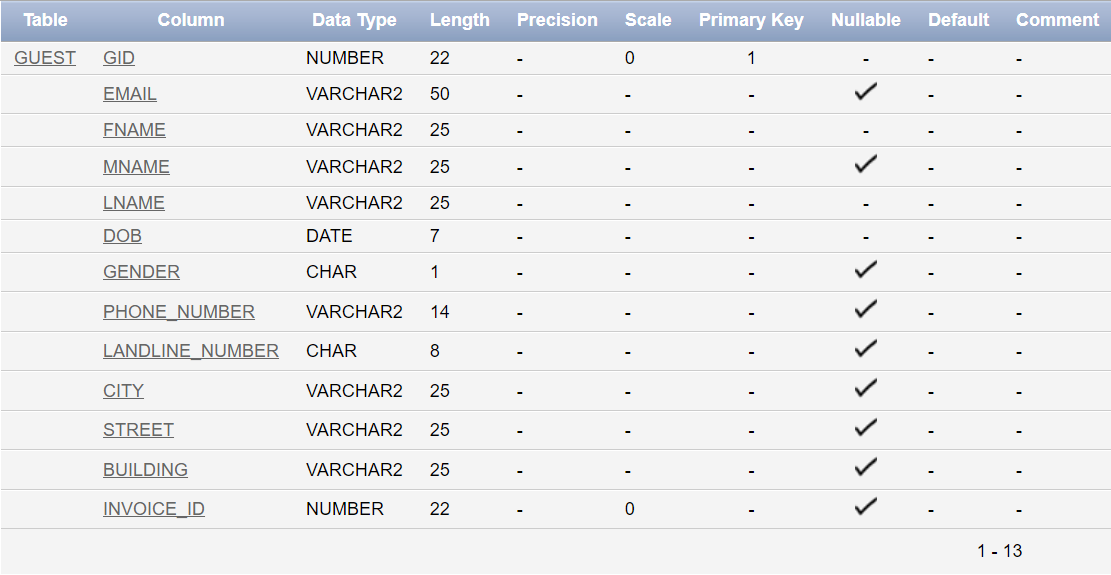
## EMPLOYEE:

SQL> DESC EMPLOYEE;



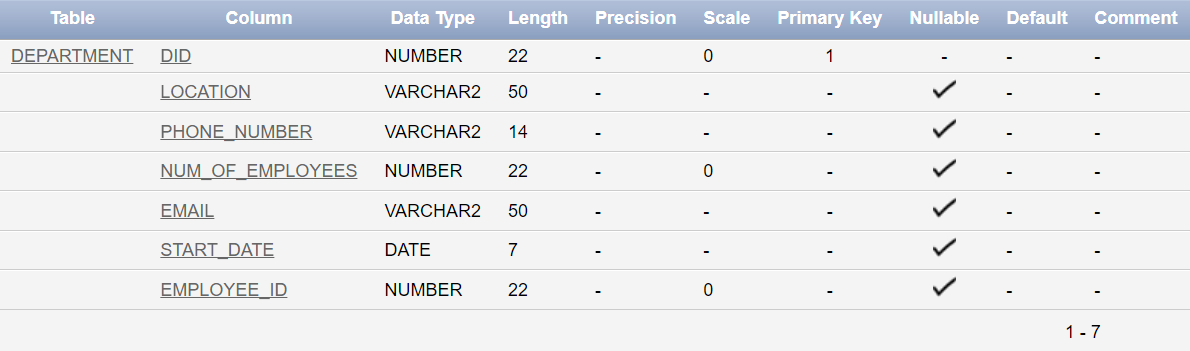
## GUEST:

SQL> DESC GUEST;



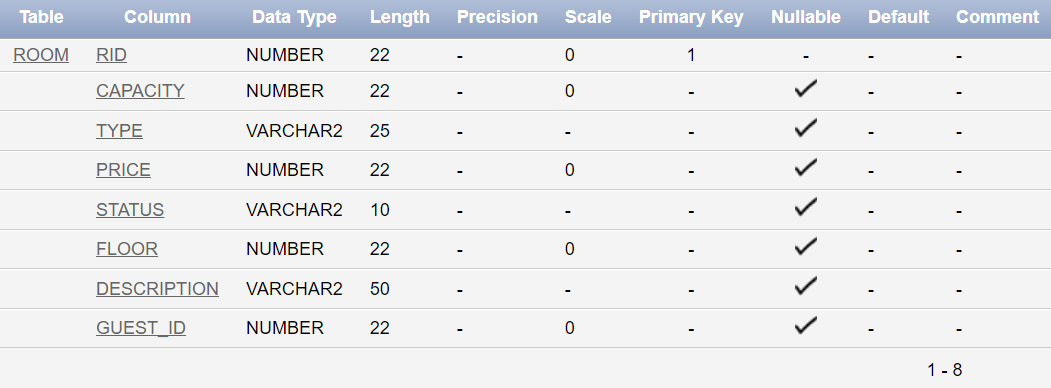
## DEPARTMENT:

SQL> DESC DEPARTMENT;



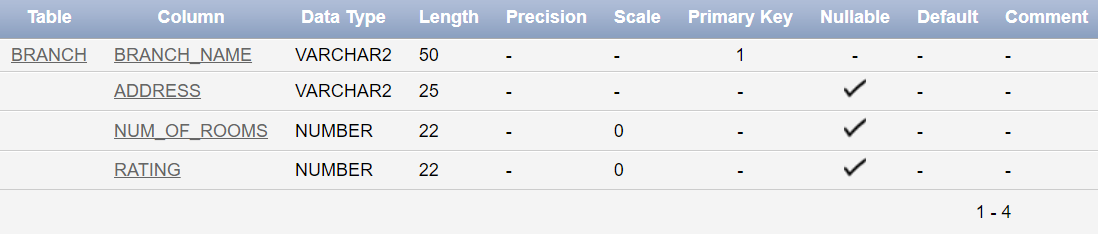
## ROOM:

SQL> DESC ROOM;



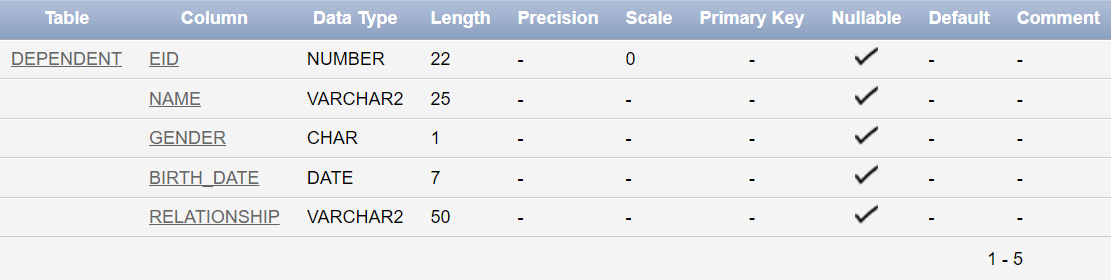
## BRANCH:

SQL> DESC BRANCH;



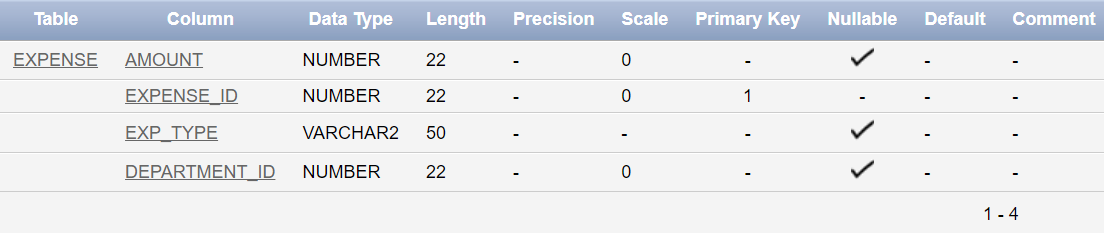
## DEPENDENT:

SQL> DESC DEPENDENT;



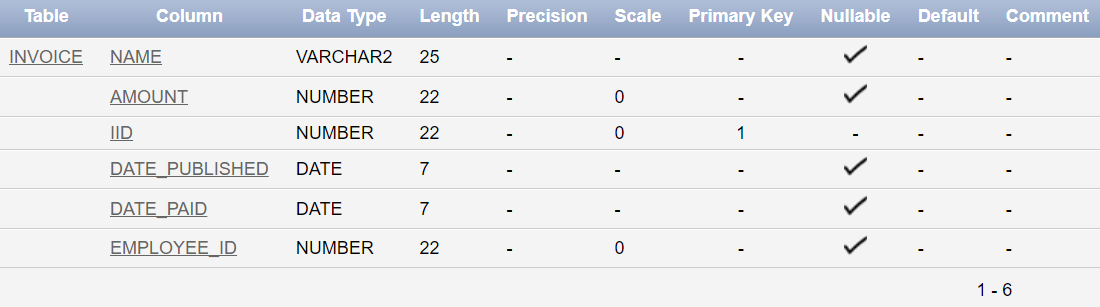
## EXPENSE:

SQL> DESC EXPENSE;



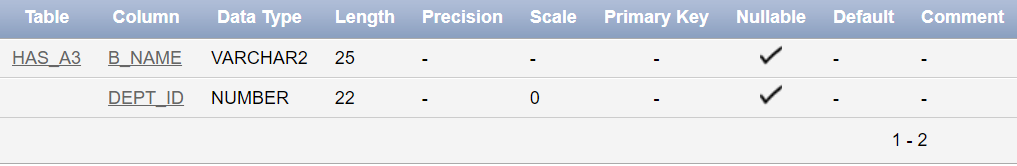
## INVOICE:

SQL> DESC INVOICE;



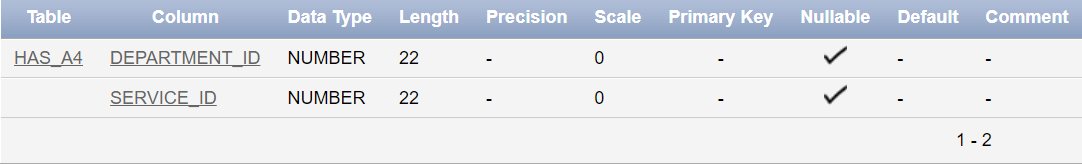
## HAS\_A3:

SQL> DESC HAS\_A3;



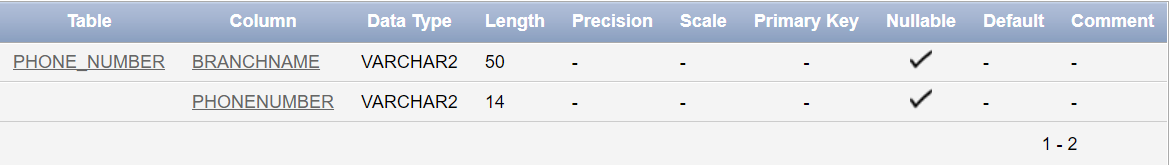
## HAS\_A4:

SQL> DESC HAS\_A4;



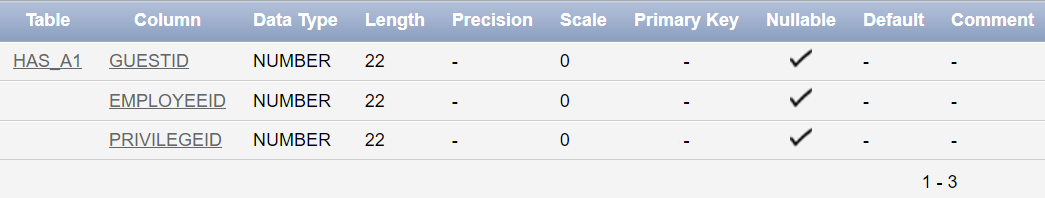
## PHONE\_NUMBER:

SQL> DESC PHONE\_NUMBER;



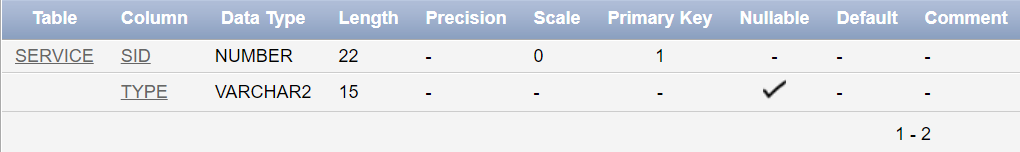
## HAS\_A1:

SQL> DESC HAS\_A1;



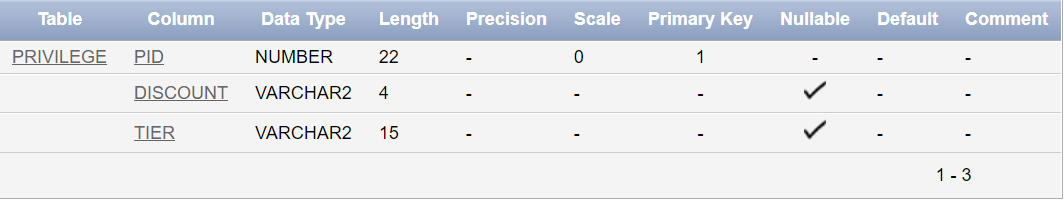
## SERVICE:

SQL> DESC SERVICE;



## PRIVILEGE:

SQL> DESC PRIVILEGE;



# XI- Inserting Data:

## EMPLOYEE:

INSERT INTO EMPLOYEE VALUES (1, 'msmd@gmail.com', 'Mohamad', null, 'Darwish', 'M', 'First building', 21, 'Branch 1', 'A street', '03461277', '01245654', 'Chef', 3000, 'beirut', 21, '12-12-2021', 1, '12-22-2000');

INSERT INTO EMPLOYEE VALUES (2, 'MohamadJ@gmail.com', 'Mohamad', null, 'Jaber', 'M', 'Second building', 21, 'Branch 2', 'B street', '03454777', '01241254', 'Chef', 2500, 'beirut', 21, '12-14-2021', 2, '12-9-2000');

INSERT INTO EMPLOYEE VALUES (3, 'MohamadK@gmail.com', 'Mohamad', 'Ali', 'Kalaaji', 'M', 'Third building', 21, 'Branch 3', 'C street', '03464787', '01245654', 'Sous-Chef', 2000, 'beirut', 22, '12-15-2021', 1, '12-12-1999');

INSERT INTO EMPLOYEE VALUES (4, 'KhalilJ@gmail.com', 'Khalil', null, 'Jaroudi', 'M', 'First building', 21, 'Branch 1', 'A street', '03464756', '01245154', 'Waiter', 1500, 'beirut', 23, '12-17-2021', 1, '12-12-1998');

INSERT INTO EMPLOYEE VALUES (5, 'NicolasS@gmail.com', 'Nicolas', null, 'Shokolata', 'M', 'First building', 21, 'Branch 1', 'A street', '03433777', '01205654', 'Waiter', 1500, 'beirut', 24, '12-12-2021', 12, '2-12-1998');  
  
INSERT INTO EMPLOYEE VALUES (6, 'LeaI@gmail.com', 'Lea', null, 'Itani', 'F', 'First building', 14, 'Branch 1', 'A street', '03460777', '01247754', 'Receptionist', 4000, 'beirut', 31, '12-19-2021', 3, '1-11-1992');

INSERT INTO EMPLOYEE VALUES (7, 'NathalieA@gmail.com', 'Nathalie', null, 'Ayoub', 'F', 'First building', 14, 'Branch 1', 'A street', '03468677', '01045654', 'Receptionist', 3500, 'beirut', 31, '12-19-2021', 12, '1-12-1992');

INSERT INTO EMPLOYEE VALUES (8, 'AyaK@gmail.com', 'Aya', null, 'Kousa', 'F', 'First building', 14, 'Branch 1', 'A street', '03464007', '01240654', 'Receptionist', 3800, 'beirut', 21, '12-12-2021', 212121, 19, '1-14-2001');  
  
INSERT INTO EMPLOYEE VALUES (9, 'DanaD@gmail.com', 'Dana', null, 'Darwish', 'F', 'Second building', 14, 'Branch 2', 'B street', '03400077', '01245054', 'Receptionist', 4100, 'beirut', 22, '12-15-2021', 4, '1-1-2000');

INSERT INTO EMPLOYEE VALUES (10, 'CharlieC@gmail.com', 'Charlie', null, 'Choplin', 'F', 'Second building', 14, 'Branch 2', 'B street', '03064777', '01245604', 'Receptionist', 3200, 'beirut', 21, '12-17-2021', 5, '1-12-2001');

INSERT INTO EMPLOYEE VALUES (11, 'LeoD@gmail.com', 'Leonardo', null, 'Davinci', 'M', 'Second building', 14, 'Branch 2', 'B street', '03464777', '01245650', 'Receptionist', 3400, 'beirut', 28, '12-14-2021', 12, '12-13-1994');  
  
INSERT INTO EMPLOYEE VALUES (12, 'NicolasT@gmail.com', 'Nicolas', null, 'Tesla', 'M', 'First building', null, 'Branch 1', 'A street', '03464770', '01005654', 'Hotel Manager', 15000, 'beirut', 29, '12-22-2020', 6, '12-2-1994');

INSERT INTO EMPLOYEE VALUES (13,'LisaS@gmail.com', 'Lisa', null, 'Sue', 'M', 'Second building', 12, 'Branch 2', 'B street', '03461177', '01240054', 'Office Manager', 10000, 'beirut', 28, '12-11-2020', 6, '11-12-1994');

INSERT INTO EMPLOYEE VALUES (14, 'JensenH@gmail.com', 'Jensen', null, 'Huan', 'M', 'Second building', 12, 'Branch 2', 'B street', '03464711', '01245600', 'Receptionist Manager', 8000, 'beirut', 29, '12-13-2020', 6, '12-12-1993');

INSERT INTO EMPLOYEE VALUES (15, 'TiaS@gmail.com', 'Tia', null, 'Sasa', 'F', 'Second building', 13, 'Branch 2', 'B street', '03464117', '01233654', 'Office Worker', 4100, 'beirut', 20, '12-12-2021', 1, '12-18-2001');

INSERT INTO EMPLOYEE VALUES (16, 'HalaS@gmail.com', 'Hala', null, 'Darwish', 'Shams', 'Third building', 13, 'Branch 3', 'C street', '03422777', '01244454', 'Office Worker', 4200, 'beirut', 20, '12-18-2021', 12, '12-12-2001');

INSERT INTO EMPLOYEE VALUES (17, 'MonaL@gmail.com', 'Mona', null, 'Lisa', 'F', 'Third building', 13, 'Branch 3', 'C street', '03433777', '01245555', 'Office Worker', 3800, 'beirut', 24, '12-22-2021', 3, '2-12-1998');

INSERT INTO EMPLOYEE VALUES (18, 'Mariah@gmail.com', 'Mariah', null, 'Mariah', 'F', 'Third building', 13, 'Branch 3', 'C street', '03444777', '01245774', 'Office Worker', 4600, 'beirut', 25, '12-16-2021', 2, '1-12-1997');

INSERT INTO EMPLOYEE VALUES (19, 'MarioC@gmail.com', 'Mario', null, 'Chehab', 'M', 'Third building', 13, 'Branch 3', 'C street', '03464667', '01247774', 'Office Worker', 4000, 'beirut', 23, '12-12-2021', 1, '12-18-1998');

INSERT INTO EMPLOYEE VALUES (20, 'JoshJ@gmail.com', 'Mohamad', 'J', 'Woodruff', 'M', 'Third building', 13, 'Branch 3', 'C street', '034699777', '01277654', 'Office Worker', 4150, 'beirut', 21, '12-28-2021', 3, '12-12-1999');

INSERT INTO EMPLOYEE VALUES (21, 'ElonM@gmail.com', 'Elon', null, 'Musk', 'M', 'Third building', 12, 'Branch 3', 'C street', '03469777', '01245694', 'Kitchen Manager', 7000, 'beirut', 29, '12-12-2020', 6, '12-12-2001');

## GUEST:

INSERT INTO GUEST VALUES (1, null, 'Mohamad', null, 'Darwish', '6-15-2001', 'M', 71450269, null, 'Beirut','Mar Elias', 'Jrab', 1);

INSERT INTO GUEST VALUES (2, null, 'Mohamad', 'Ali', 'tebeh', '6-13-2001', 'M', 71410261, null, 'Beirut','Bliss', 'karen', 2);

INSERT INTO GUEST VALUES (3, null, 'Elony', null, 'Musky', '6-28-1971', 'M', 71450759, null, null,null, null, 3);

INSERT INTO GUEST VALUES (4, 'zoro@gmail.com', 'Zoro', null, 'Roronoa', '1-12-2001', 'M', 71657269, null, 'Beirut','Hamra', 'naml', 4);

INSERT INTO GUEST VALUES (5, 'raphael@gmail.com', 'Raphael', null, 'Deen', '3-13-2002', 'M', 71457267, null, 'Beirut','Salim Salem', 'Jrab', 5);

INSERT INTO GUEST VALUES (6, null, 'Mostafa', null, 'Kousa', '3-11-2001', 'M', 71450200, 01546789, 'Beirut','Mar Elias', 'Itani', 6);

INSERT INTO GUEST VALUES (7, null, 'Sara', null, 'Shoukair', '6-11-2000', 'F', 71400269, null, 'Beirut','Mar Elias', 'Jrab', 7);

INSERT INTO GUEST VALUES (8, 'anya@gmail.com', 'Anya', null, 'Melez', '6-12-2001', 'F', 71400069, null, 'Beirut','Gemmayzeh', 'Tabara', 8);

INSERT INTO GUEST VALUES (9, null, 'Tanya', null, 'Degurechaff', '8-1-2000', 'F', null, null, null,null, null, 9);

INSERT INTO GUEST VALUES (10, 'lynn@gmail.com', 'Lynn', null, 'Tabara', '3-15-2000', 'F', 71433369, null, 'Beirut','Malla', 'Musk', 10);

INSERT INTO GUEST VALUES (11, null, 'Nemo', null, 'Abounemo', '2-25-2000', 'M', 71451289, null, 'Beirut','Mazraa', 'Darwishs', 11);

## DEPARTMENT:

INSERT INTO DEPARTMENT VALUES (12, 'Beirut barbir Majed Building', '81354267' , 15, 'MajedBuilding@gmail.com','12-4-2016', 1);  
  
INSERT INTO DEPARTMENT VALUES (6, 'Beirut barbir Darwish Building', '03276382' , 20, 'DarwishBuilding@gmail.com','1-4-2018', 2);  
  
INSERT INTO DEPARTMENT VALUES (8, 'Beirut barbir Haysam Building', '03615287' , 30, 'HaysamBuilding@gmail.com','3-8-2013', 3);  
  
INSERT INTO DEPARTMENT VALUES (4, 'Beirut ras el nabeh Aziz Building', '76621811' , 43, 'AzizBuilding@gmail.com','7-7-2020', 4);  
  
INSERT INTO DEPARTMENT VALUES (1, 'Jounieh Kaslik Taha Building', '70860212' , 23, 'TahaBuilding@gmail.com','12-4-2012', 5);  
  
INSERT INTO DEPARTMENT VALUES (7, 'Jounieh Kaslik Jamhour Building', '8135622' , 18, 'JamhourBuilding@gmail.com','12-5-2016', 6);  
  
INSERT INTO DEPARTMENT VALUES (2, 'Jounieh Kaslik Maro Building', '71217654' , 30, 'MaroBuilding@gmail.com','4-4-2022', 7);  
  
INSERT INTO DEPARTMENT VALUES (5, 'Beirut ras el nabeh Monzir Building', '71244182' , 15, 'MonzirBuilding@gmail.com','6-6-2017', 8);  
  
INSERT INTO DEPARTMENT VALUES (9, 'Beirut ras el nabeh Harkous Building', '01643082' , 28, 'HarkousBuilding@gmail.com','12-12-2016', 9);  
  
INSERT INTO DEPARTMENT VALUES (3, 'Beirut Rawshe Salem Building', '01782367' , 12, 'SalemBuilding@gmail.com','5-11-2013', 10);  
  
INSERT INTO DEPARTMENT VALUES (10, 'Beirut Rawshe Skyhome building', '81667552' , 60, 'SkyhomeBuilding@gmail.com','3-4-2018', 11);  
  
INSERT INTO DEPARTMENT VALUES (11, 'Beirut Rawshe Chbeir building', '81667552' , 60, 'ChbeirBuilding@gmail.com','3-4-2018', 12);

## ROOM:

INSERT INTO ROOM VALUES(102, 3 , 'triple bed',200 , 'available',1 , 'triple bed family deal', null);  
  
INSERT INTO ROOM VALUES(103, 5, 'suite',350 , 'available',1 , '2 rooms suite with 5 beds', null);  
  
INSERT INTO ROOM VALUES(201 ,2 , '1 bed',150 , 'occupied',2 , '1 bed room for 1 or 2 people', 1);  
  
INSERT INTO ROOM VALUES(202 ,2 , '1 bed',150 , 'available',2 , '1 bed room for 1 or 2 people', null);  
  
INSERT INTO ROOM VALUES(203, 5 , 'suite',320 , 'available',2 , '2 rooms suite with 5 beds', null);  
  
INSERT INTO ROOM VALUES(301, 3 , 'triple bed',210 , 'occupied',3 , 'triple bed family deal', 6);  
  
INSERT INTO ROOM VALUES(302, 3 , 'triple bed',210 , 'available',3 , 'triple bed family deal', null);  
  
INSERT INTO ROOM VALUES(303, 5, 'suite',360 , 'available',3 , '2 rooms suite with 5 beds', null);  
  
INSERT INTO ROOM VALUES(401 ,2 , '1 bed',160 , 'occupied',4 , '1 bed room for 1 or 2 people', 4);

INSERT INTO ROOM VALUES(402 ,2 , '1 bed',160 , 'occupied',4 , '1 bed room for 1 or 2 people', 2);  
  
INSERT INTO ROOM VALUES(403, 5 , 'suite',370 , 'occupied',4 , '2 rooms suite with 5 beds', 3);

## BRANCH:

INSERT INTO BRANCH VALUES('Branch 1','Beirut Barbir',25,5);

INSERT INTO BRANCH VALUES('Branch 2','Beirut Koraytem',40,4);

INSERT INTO BRANCH VALUES('Branch 3','Sayda Downtown',30,5);

INSERT INTO BRANCH VALUES('Branch 4','Beirut Raouche',55,5);

INSERT INTO BRANCH VALUES('Branch 5','Baalbek Shrawne',35,3);

INSERT INTO BRANCH VALUES('Branch 6','Beirut Ras El Nabeh',20,3);

INSERT INTO BRANCH VALUES('Branch 7','Jounieh Kaslik',50,5);

INSERT INTO BRANCH VALUES('Branch 8','Jbeil OLD ROAD',35,4);

INSERT INTO BRANCH VALUES('Branch 9','Sour Sea',25,5);

INSERT INTO BRANCH VALUES('Branch 10','Beirut Old Airport Road',50,4);

INSERT INTO BRANCH VALUES('Branch 11','Nabattiyeh Nmayriyeh',30,4);

INSERT INTO BRANCH VALUES('Branch 12','Beqaa West Beqaa',40,5);

## DEPENDENT:

INSERT INTO DEPENDENT VALUES( 1, 'Mohamad', 'M', '12-04-2002', 'son');

INSERT INTO DEPENDENT VALUES( 1, 'Ali', 'M', '10-15-1970', 'husband');

INSERT INTO DEPENDENT VALUES( 1, 'Mariam', 'G', '10-2-2004', 'daughter');

INSERT INTO DEPENDENT VALUES( 1, 'Sara', 'G', '1-8-2005', 'daughter');

INSERT INTO DEPENDENT VALUES( 2, 'Daniel', 'M', '6-8-2000', 'son');

INSERT INTO DEPENDENT VALUES( 2, 'Jad', 'M', '2-2-1968', 'husband');

INSERT INTO DEPENDENT VALUES( 4, 'Miguel', 'M', '4-4-1972', 'husband');

INSERT INTO DEPENDENT VALUES( 5, 'Maya', 'G', '3-12-1975', 'wife');

INSERT INTO DEPENDENT VALUES( 6, 'Jana', 'G', '4-8-1920', 'wife');

INSERT INTO DEPENDENT VALUES( 7, 'Mohamad', 'M', '12-04-2002', 'husband');

INSERT INTO DEPENDENT VALUES( 7, 'Ali', 'M', '7-3-2003', 'son');

INSERT INTO DEPENDENT VALUES( 8, 'Lea', 'G', '2-12-1978', 'wife');

## EXPENSE:

INSERT INTO EXPENSE VALUES( 2000,1232,'water',2 );

INSERT INTO EXPENSE VALUES( 1500,1234 ,'Electricity',2 );

INSERT INTO EXPENSE VALUES( 1200,1235 ,'Suplies',2 );  
  
INSERT INTO EXPENSE VALUES( 2200,1237,'water',4 );

INSERT INTO EXPENSE VALUES( 1700,1238 ,'Electricity',4 );

INSERT INTO EXPENSE VALUES( 1000,1240 ,'Suplies',4 );  
  
INSERT INTO EXPENSE VALUES( 1030,1242,'water',5 );

INSERT INTO EXPENSE VALUES( 900,1243 ,'Electricity',5 );

INSERT INTO EXPENSE VALUES( 1100,1245 ,'Suplies',5 );  
  
INSERT INTO EXPENSE VALUES( 1540,1247,'water',8 );

INSERT INTO EXPENSE VALUES( 1120,1248 ,'Electricity',8 );

INSERT INTO EXPENSE VALUES( 1330,1249 ,'Suplies',8 );

## INVOICE:

INSERT INTO INVOICE VALUES('Hotel 1 day single bed', 160, 1, '12-12-2021', '12-13-2021',15);

INSERT INTO INVOICE VALUES('Hotel 2 days single bed',300, 2, '12-12-2021', '12-13-2021',15);

INSERT INTO INVOICE VALUES('Hotel 7 days single bed', 1050, 3, '12-12-2021', '12-19-2021',16);

INSERT INTO INVOICE VALUES('Hotel 1 day suite', 350, 4, '12-12-2021', '12-13-2021',15);

INSERT INTO INVOICE VALUES('Hotel 2 days suite',720, 5, '12-12-2021', '12-13-2021',17);

INSERT INTO INVOICE VALUES('Hotel 1 day triple bed', 210, 6, '12-13-2021', '12-14-2021',15);

INSERT INTO INVOICE VALUES('Hotel 2 days triple bed', 400, 7, '12-14-2021', '12-16-2021',15);

INSERT INTO INVOICE VALUES('Hotel 7 days single bed', 1050, 8, '12-14-2021', '12-21-2021',16);

INSERT INTO INVOICE VALUES('Hotel 30 days suite', 5550, 9, '12-12-2021', '1-12-2021',16);

INSERT INTO INVOICE VALUES('Restaurant dinner', 100, 10, '12-12-2021', '12-12-2021',17);

INSERT INTO INVOICE VALUES('Hotel 2 days', 0, 11, '12-13-2021', '12-14-2021',17);

## HAS\_A3:

INSERT INTO HAS\_A3 VALUES('Branch 1',1);

INSERT INTO HAS\_A3 VALUES('Branch 1',6);

INSERT INTO HAS\_A3 VALUES('Branch 1',8);

INSERT INTO HAS\_A3 VALUES('Branch 6',4);

INSERT INTO HAS\_A3 VALUES('Branch 8',5);

INSERT INTO HAS\_A3 VALUES('Branch 6',9);

INSERT INTO HAS\_A3 VALUES('Branch 7',12);

INSERT INTO HAS\_A3 VALUES('Branch 7',2);

INSERT INTO HAS\_A3 VALUES('Branch 7',7);

INSERT INTO HAS\_A3 VALUES('Branch 4',3);

INSERT INTO HAS\_A3 VALUES('Branch 4',10);

INSERT INTO HAS\_A3 VALUES('Branch 4',11);

## HAS\_A4:

INSERT INTO HAS\_A4 VALUES(5,1);

INSERT INTO HAS\_A4 VALUES(5,2);

INSERT INTO HAS\_A4 VALUES(5,6);

INSERT INTO HAS\_A4 VALUES(5,8);      
  
INSERT INTO HAS\_A4 VALUES(1,3);

INSERT INTO HAS\_A4 VALUES(1,8);

INSERT INTO HAS\_A4 VALUES(4,1);

INSERT INTO HAS\_A4 VALUES(1,2);      
  
INSERT INTO HAS\_A4 VALUES(1,4);

INSERT INTO HAS\_A4 VALUES(1,1);

INSERT INTO HAS\_A4 VALUES(1,7);

INSERT INTO HAS\_A4 VALUES(1,10);      
  
INSERT INTO HAS\_A4 VALUES(2,8);

INSERT INTO HAS\_A4 VALUES(2,5);

INSERT INTO HAS\_A4 VALUES(2,1);      
  
INSERT INTO HAS\_A4 VALUES(3,1);

INSERT INTO HAS\_A4 VALUES(3,3);

INSERT INTO HAS\_A4 VALUES(3,5);

INSERT INTO HAS\_A4 VALUES(3,6);

INSERT INTO HAS\_A4 VALUES(3,9);

## PHONE\_NUMBER:

INSERT INTO PHONE\_NUMBER VALUES('Branch 1', 01662201);

INSERT INTO PHONE\_NUMBER VALUES('Branch 2', 01662202);

INSERT INTO PHONE\_NUMBER VALUES('Branch 3', 01662203);

INSERT INTO PHONE\_NUMBER VALUES('Branch 4', 01662204);

INSERT INTO PHONE\_NUMBER VALUES('Branch 5', 01662205);

INSERT INTO PHONE\_NUMBER VALUES('Branch 6', 01662206);

INSERT INTO PHONE\_NUMBER VALUES('Branch 7', 01662207);

INSERT INTO PHONE\_NUMBER VALUES('Branch 8', 01662208);

INSERT INTO PHONE\_NUMBER VALUES('Branch 9', 01662209);

INSERT INTO PHONE\_NUMBER VALUES('Branch 10', 01662210);

INSERT INTO PHONE\_NUMBER VALUES('Branch 11', 01662211);

INSERT INTO PHONE\_NUMBER VALUES('Branch 12', 01662212);

## HAS\_A1:

INSERT INTO HAS\_A1 VALUES (1, null, 1);

INSERT INTO HAS\_A1 VALUES (2, null, 2);

INSERT INTO HAS\_A1 VALUES (3, null, 3);

INSERT INTO HAS\_A1 VALUES (4, null, 5);

INSERT INTO HAS\_A1 VALUES (5, null, 4);

INSERT INTO HAS\_A1 VALUES (null, 1, 1);

INSERT INTO HAS\_A1 VALUES (null, 2, 1);

INSERT INTO HAS\_A1 VALUES (null, 3, 2);

INSERT INTO HAS\_A1 VALUES (null, 4, 4);

INSERT INTO HAS\_A1 VALUES (null, 5, 5);

## SERVICE:

INSERT INTO SERVICE VALUES(1,'Pool');

INSERT INTO SERVICE VALUES(2,'Restaurant');

INSERT INTO SERVICE VALUES(3,'Parking');

INSERT INTO SERVICE VALUES(4,'Wifi');

INSERT INTO SERVICE VALUES(5,'Housekeeping');

INSERT INTO SERVICE VALUES(6,'Vending Machine');

INSERT INTO SERVICE VALUES(7,'Kitchen');

INSERT INTO SERVICE VALUES(8,'Air conditioning');

INSERT INTO SERVICE VALUES(9,'Towels');

INSERT INTO SERVICE VALUES(10,'SPA');

## PRIVILEGE:

INSERT INTO PRIVILEGE VALUES( 1, '10%', 'IRON');

INSERT INTO PRIVILEGE VALUES( 2, '12%', 'BRONZE');

INSERT INTO PRIVILEGE VALUES( 3, '13%', 'SILVER');

INSERT INTO PRIVILEGE VALUES( 4, '15%', 'GOLD');

INSERT INTO PRIVILEGE VALUES( 5, '20%', 'PLATINUM');

INSERT INTO PRIVILEGE VALUES( 6, '25%', 'DIAMOND');

INSERT INTO PRIVILEGE VALUES( 7, '30%', 'MASTER');

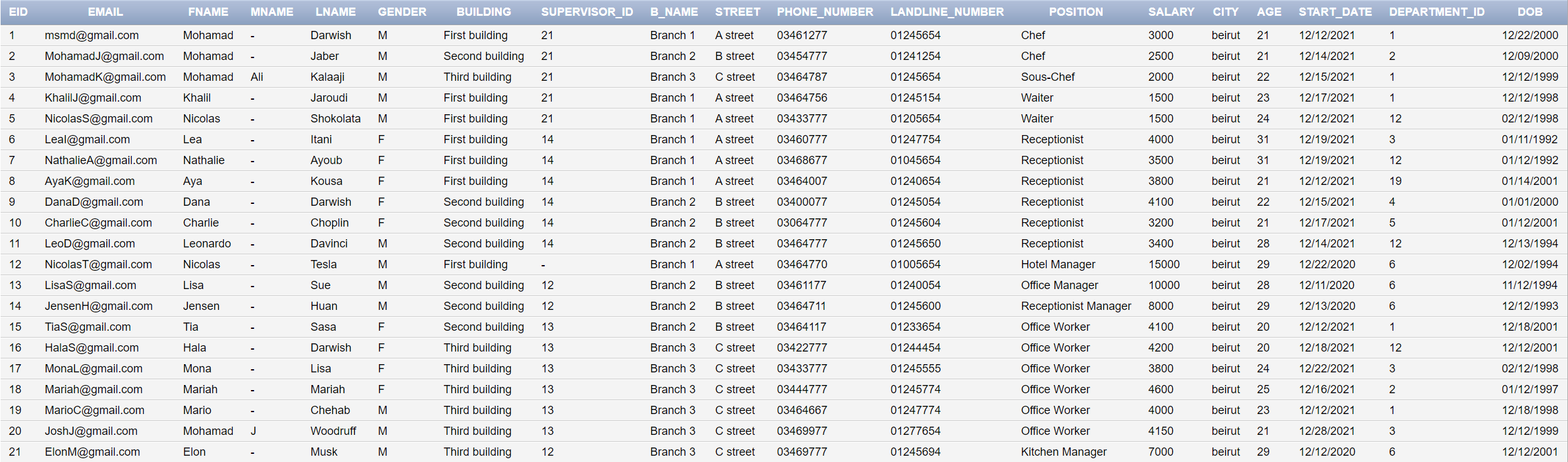
INSERT INTO PRIVILEGE VALUES( 8, '50%', 'GRAND MASTER');

INSERT INTO PRIVILEGE VALUES( 9, '70%', 'CHALLENGER');

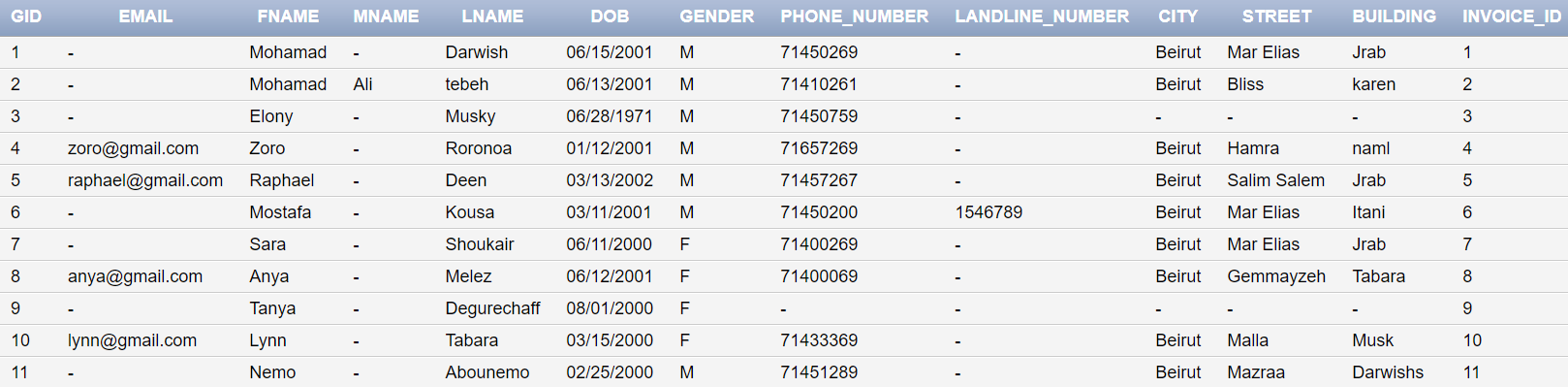
INSERT INTO PRIVILEGE VALUES( 10, '100%', 'LEGENDARY');

# XII- Final Tables State:

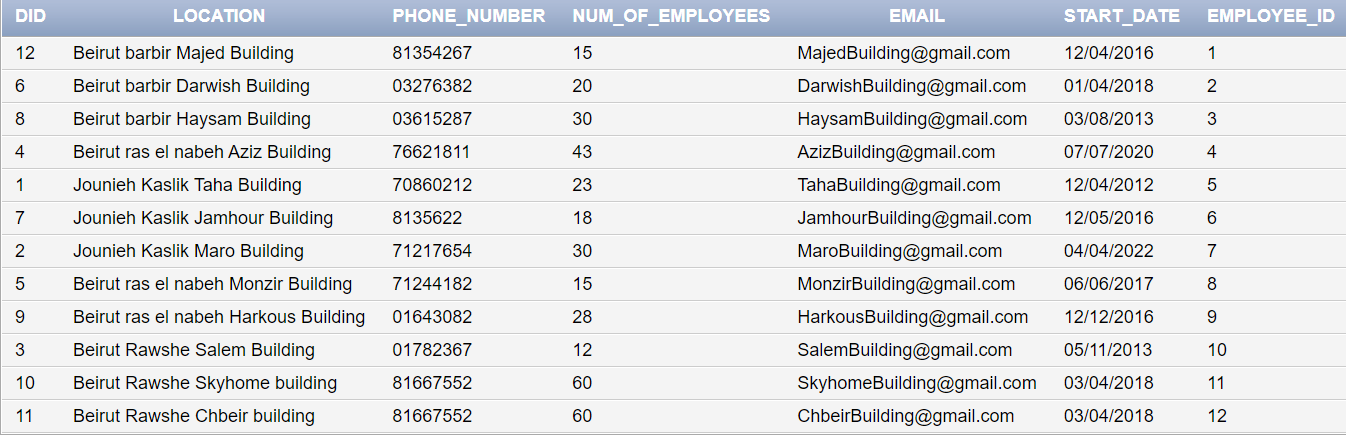
## EMPLOYEE:



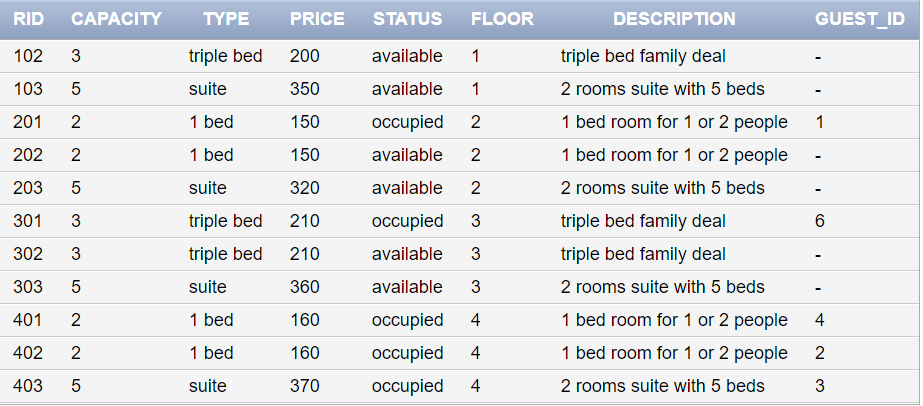
## GUEST:



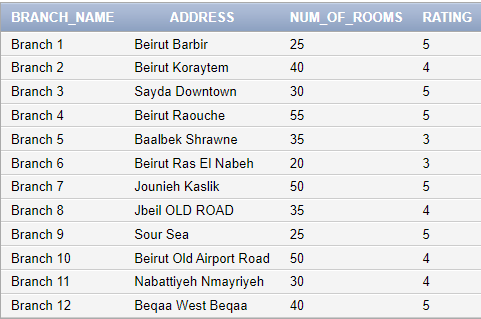
## DEPARTMENT:



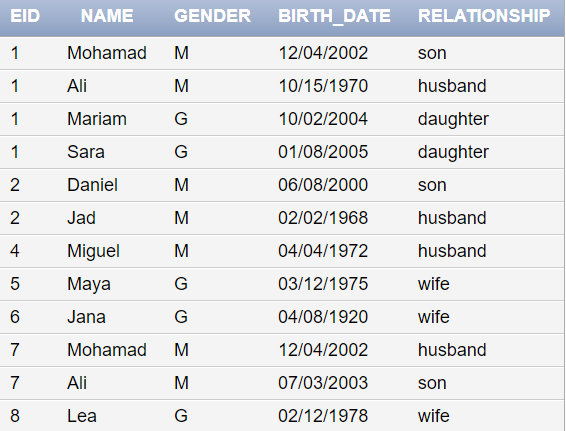
## ROOM:



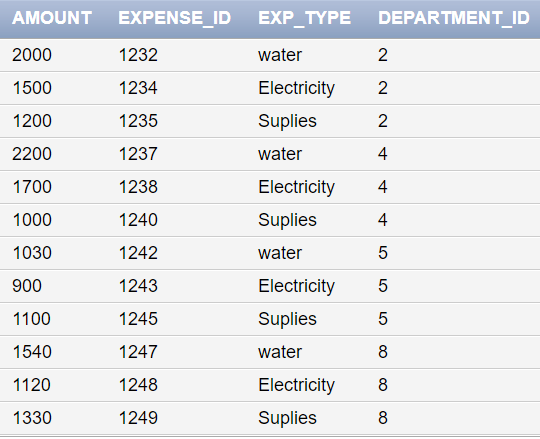
## BRANCH:



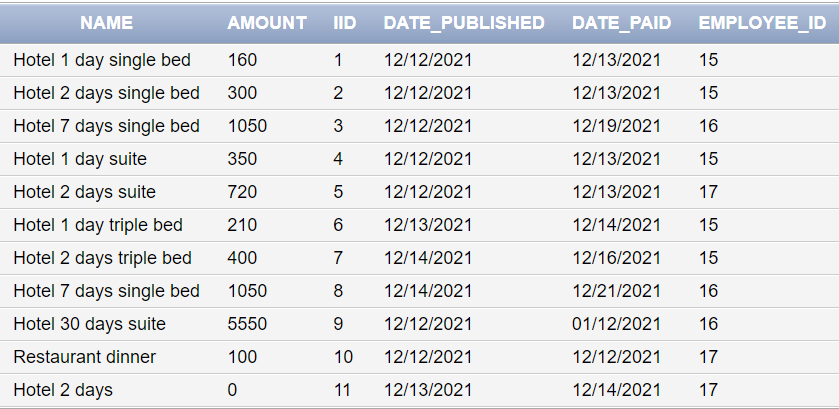
## DEPENDENT:



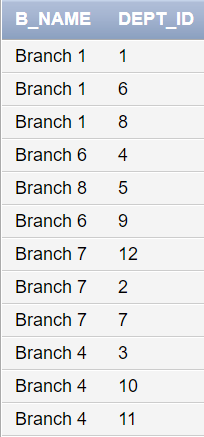
## EXPENSE:



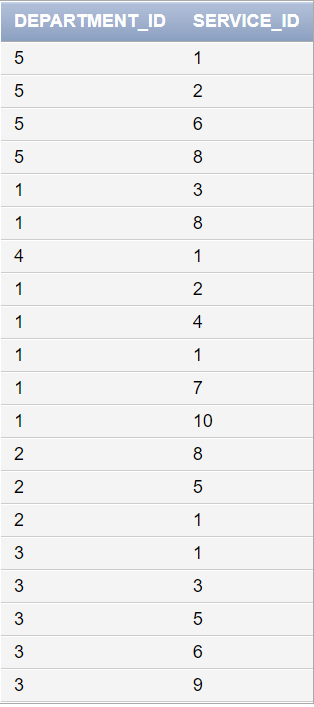
## INVOICE:



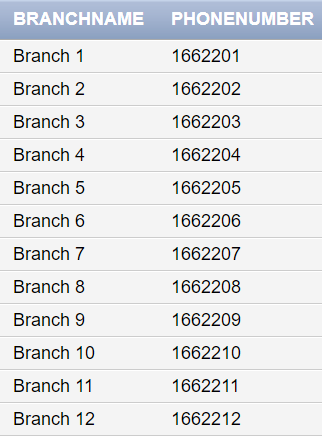
## HAS\_A3:



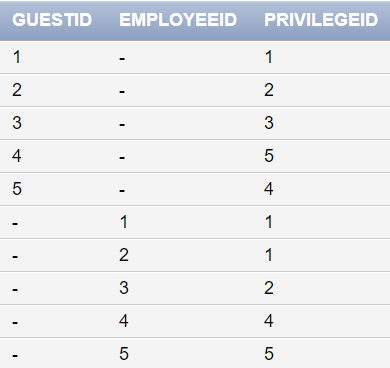
## HAS\_A4:



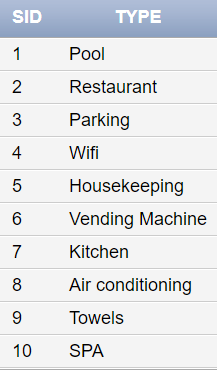
## PHONE\_NUMBER:



## HAS\_A1:



## SERVICE:



## PRIVILEGE:



# XIII- Queries:

## Query 1:

A customer that lives in Beirut wanted to know all the branches information including phone number, address, and branch name in Beirut that provide pool services.

SELECT B\_NAME, LOCATION, PHONE\_NUMBER

FROM DEPARTMENT

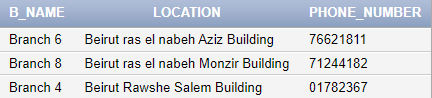
INNER JOIN HAS\_A4 ON DID=DEPARTMENT\_ID

INNER JOIN HAS\_A3 ON DEPT\_ID=DEPARTMENT\_ID

WHERE LOCATION LIKE 'Beirut%'

AND

SERVICE\_ID = 1;



## Query 2:

The same customer moves on and calls one of the branches asking if there were 2 suites available for booking. The employee moves on and runs this query to check for room availability.

SELECT COUNT(RID) FROM ROOM

WHERE TYPE = 'suite' AND STATUS = 'available';



## Query 3:

A random customer comes in saying his father '' has booked a 1 bed room in the hotel

The front desk employee moves on to check if the customers father has booked a room

to provide the customer with the room information.

SELECT RID, TYPE, FNAME, LNAME

FROM (ROOM INNER JOIN GUEST ON GID=GUEST\_ID)

WHERE FNAME = 'Mohamad' AND LNAME = 'tebeh';



## Query 4:

Many guests have sent complaints regarding the food served in the restaurant at Branch 1. Therefore, the following measures are taken:

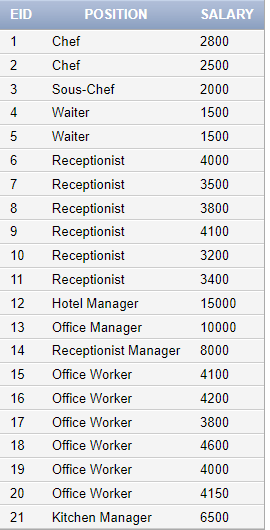
* 200$ were deducted from the chef’s salary of this particular branch
* 500$ from the Kitchen’s Manager.

UPDATE EMPLOYEE SET SALARY = SALARY - 200

WHERE POSITION = 'Chef' AND B\_NAME = 'Branch 1';

UPDATE EMPLOYEE SET SALARY = SALARY - 500

WHERE POSITION = 'Kitchen Manager';

Before and After:

## Query 5:

A loyal guest needs to know how much discount he has.

SELECT FNAME, DISCOUNT FROM HAS\_A1, GUEST, PRIVILEGE

WHERE GID = 4

AND

GID = GUESTID

AND PID= PRIVILEGEID;



## Query 6:

The Receptionist

List all the information of the current guests (first name, middle name, last name, email, phone number) in all branches.

SELECT FNAME, MNAME, LNAME, EMAIL, PHONE\_NUMBER FROM GUEST, ROOM

WHERE GID = GUEST\_ID

AND

STATUS = 'occupied';

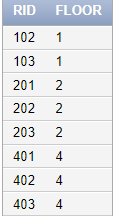


## Query 7:

A fire occurred on the third floor, we need to delete its rooms from our database.

DELETE FROM ROOM

WHERE FLOOR = 3;

Before And After:

Table

Description automatically generated

## Query 8:

Its Christmas next week! As a bonus, Branch 1 of the Hotel has decided to raise employees’ salaries based on the following:

20% raise for salaries above 3000$

15% raise for salaries between 2000 and 3000$

10% raise for salaries under 2000$

UPDATE EMPLOYEE

SET SALARY = SALARY\*1.2

WHERE SALARY >= 3000;

UPDATE EMPLOYEE

SET SALARY = SALARY\*1.15

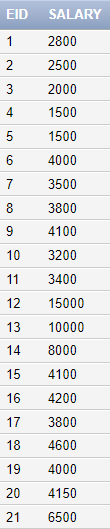
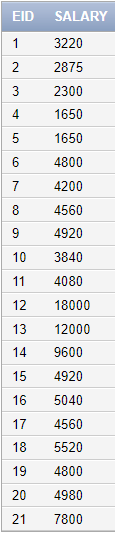
WHERE SALARY >= 2000 AND SALARY <3000;

UPDATE EMPLOYEE

SET SALARY = SALARY\*1.1

WHERE SALARY < 2000;

Before And After:



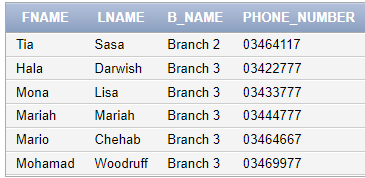
## Query 9:

The office manager, Lisa Sue, was asked by her supervisor, Nicolas Tesla to give him a list of all the employees supervised by her in all branches.

SELECT FNAME, LNAME, B\_NAME, PHONE\_NUMBER

FROM EMPLOYEE

WHERE SUPERVISOR\_ID = 13;



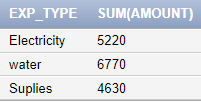
## Query 10:

The hotel manager Lisa Sue wanted to know the expenses of water, electricity, and supplies among all branches

SELECT EXP\_TYPE, SUM(AMOUNT)

FROM EXPENSE

GROUP BY EXP\_TYPE;



# XIII- Normalization Up to The BCNF Normal Form:

After creating all relations, we should improve them by normalizing according to several normal forms. Here we are going to normalize our database up to the fourth normal form which is the Boyce-Codd Normal Form. On each relation we are going to apply the four normal forms. We start with the first then second then third and at last the BCNF normal form. Let us first start by a general description to each normal form.

First Normal Form:

The First Normal Form disallows composite attributes, multivalued attributes, and their combinations to exist in a relation. The only attribute values permitted are single atomic values.

Second Normal Form:

Definitions:

● A Prime attribute must be a member of some candidate key

● A Non-prime attribute is not a prime attribute—that is, it is not a member of any candidate key.

● Functional Dependency: an FD Y → Z, where removal of any attribute from Y means the FD does not hold anymore.

The Second normal form is based on the concept of full functional dependency. A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key.

Third Normal Form:

The third normal form is based on the concept of transitive functional dependency.

Transitive Dependency: A functional dependency X →Y in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X →Z and Z →Y hold.

A relation schema R is in the third normal form if it satisfies the second normal form, and no nonprime attribute of R is transitively dependent on the primary key. For every nontrivial functional dependency X →Y either X should be a super key or Y is a prime attribute.

Boycee-Codd Normal Form:

The Boycee-Codd normal form is a stricter form than the third normal form. The BCNF differs from the definition of the third normal form in only one condition. The third normal form allows the right-hand side of the functional dependency to be a prime attribute while BCNF does not allow that.

1. Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EID | Email | Fname | Mname | | Lname |
| Gender | Building | Supervisor\_ID | B\_Name | | Street |
| Phone\_Number | Landline\_Number | Position | Salary | | City |
| Age | Start\_date | Department\_ID | DOB |

A. The **EMPLOYEE** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **EMPLOYEE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key **“EID”.**

C. The **EMPLOYEE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there is no non-prime attributes that are transitively dependent on the primary key **“EID”**.

D. The **EMPLOYEE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Guest

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GID | Email | Fname | Mname | Lname | DOB | Gender | Phone\_Number |
| Landline\_Number | City | Street | Building | Invoice\_ID |

A. The **GUEST** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **GUEST** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key **“GID”.**

C. Although the **GUEST** relation schema satisfies the 2NF, it doesn’t satisfy all conditions of the 3NF because “Email” determines “First\_Name” (FD: Email → First\_Name); however, “Email” is not a super key and “First\_Name” is not prime attributes.

D. The **GUEST** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

Normalize:

1. Guest1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GID | Mname | | Lname | | DOB | | Gender | Phone\_Number |
| Landline\_Number | City | Street | | Building | | Invoice\_ID | |

1. Guest2

|  |  |
| --- | --- |
| Email | Fname |

1. Department

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DID | Location | Phone\_Number | Num\_of\_Employees | Email | Start\_date |
| Employee\_ID |

A. The **DEPARTMENT** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **DEPARTMENT** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “DID”.

C. The **DEPARTMENT** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “DID”.

D. The **DEPARTMENT** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Room

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RID | Capacity | Type | Price | Status | Floor | Description | Guest\_ID |

A. The **ROOM** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **ROOM** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “RID”.

C. The **ROOM** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “RID”.

D. The **ROOM** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Branch

|  |  |  |  |
| --- | --- | --- | --- |
| Branch\_Name | Address | Num\_of\_Rooms | Rating |

A. The **BRANCH** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **BRANCH** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “Branch\_Name”.

C. The **BRANCH** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “Branch\_Name”.

D. The **BRANCH** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Dependent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EID | Name | Gender | Birth\_date | Relationship |

A. The **DEPENDENT** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **DEPENDENT** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “EID”.

C. The **DEPENDENT** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “EID”.

D. The **DEPENDENT** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Expense

|  |  |  |  |
| --- | --- | --- | --- |
| Amount | Expense\_Id | Exp\_Type | Department\_Id |

A. The **EXPENSE** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **EXPENSE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “Expense\_Id”.

C. The **EXPENSE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “Expense\_Id”.

D. The **EXPENSE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Invoice

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Amount | IID | Date\_Published | Date\_Paid | Employee\_Id |

A. The **INVOICE** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **INVOICE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “IID”.

C. The **INVOICE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “IID”.

D. The **INVOICE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Service

|  |  |
| --- | --- |
| SID | Type |

A. The **SERVICE** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **SERVICE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “SID”.

C. The **SERVICE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “SID”.

D. The **SERVICE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

1. Privilege

|  |  |  |
| --- | --- | --- |
| PID | Discount | Tier |

A. The **SERVICE** relation schema satisfies all conditions of the 1NF because it contains single and atomic attributes. This relation has neither multivalued attributes nor composite attributes.

B. The **SERVICE** relation schema satisfies all conditions of the 2NF because every non-prime attribute is fully functionally dependent on the primary key “PID”.

C. The **SERVICE** relation schema satisfies all conditions of the 3NF because it satisfies the 2NF and there are no non-prime attributes that are transitively dependent on the primary key “PID”.

D. The **SERVICE** relation schema satisfies all conditions of the BCNF because there exists no functional dependency X→A where X is not a super key or A is a prime attribute and X not a super key.

Relational Schemas Without non-prime attributes:

1. Has\_A3

|  |  |
| --- | --- |
| B\_NAME | DEPT\_ID |

1. Has\_A4

|  |  |
| --- | --- |
| DEPARTMENT\_ID | SERVICE\_ID |

1. PHONE\_Number

|  |  |
| --- | --- |
| BranchName | PhoneNumber |

1. Has\_A1

|  |  |  |
| --- | --- | --- |
| GuestID | EmployeeID | PrivilegeID |

# XIV- Conclusion:

Databases are crucial establishments for the functioning and operation of an organization, containing heaps of information and data. A hotel is one of those organizations that make use of a Database by saving a considerable amount of data on it. This database will be important for the administration and functioning of a hotel. It will help keep track of salaries, employees, customers, guests, available rooms, and so on. Hotels are convinent establishments that provide paid lodging and facilities on a short-term basis. Hotels are especially important in tourism areas where business would be booming which would help the economy thrive. Through this database, we can show the organization of a hotel and its structure.

# XV- Instructor’s Comments and Evaluation:

This page should be filled by our instructor DR. RAMZI HARATY to write any comments, improvements, and feedback for our database system. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_