MILESTONE 1

DATABASE DESGINS

23311

SUPHAL SHARMA

TABLE OF CONTENT

1. ERD

● Introduction

● ER Diagram

● Report

2. Normalisation

● Introduction

● Diagram

● Textual Form (Schema)

● Report

3. Relational Schema

● Introduction

● Schema

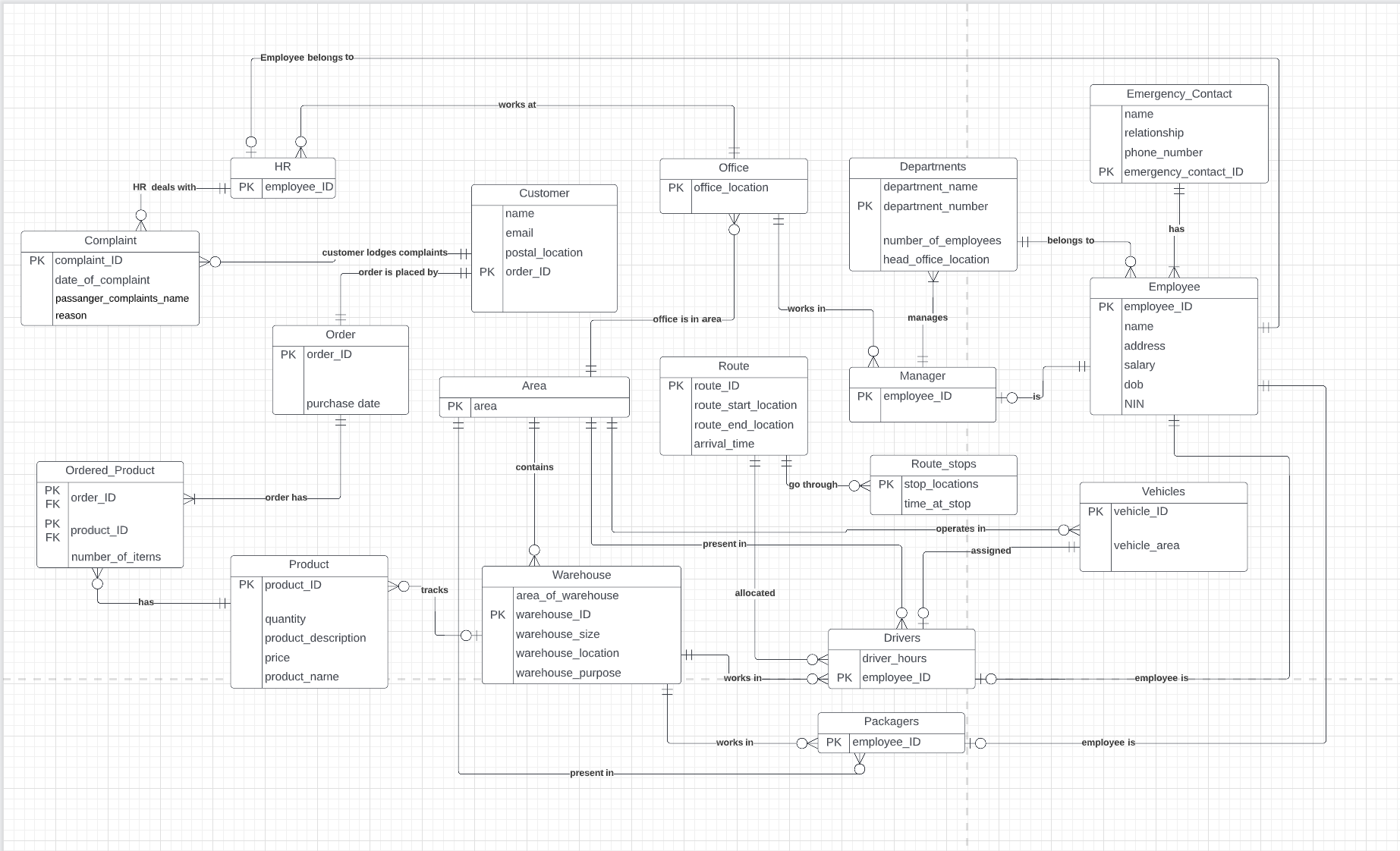
● Report

1)ERD

**Introduction**

As the Database Engineer of the company Kilburnazon. We need to create a Entity Relation Diagram of a Databases in this section. I have included PR also in each table.

**DIAGRAM**



**REPORT**

For this task I have made 18 tables. The first table is Employee with employee\_ID as the primary key. Each Employee must have one and only Emergency\_Contact and must be in one and only one department. Each Employee will be either a HR, driver, manager or packager.

Next, I have the Departments table with department\_number as the primary key. A department can have no or many employees. A department will have one and only one manager.

Next, I have Emergency\_Contact with emergency\_contact\_ID as the primary key. There must be at least 1 employee for each emergency contact.

Next, I have the Manager table with employee\_ID as the primary key. A manager is only a single employee and works in a single office. A manager will overlook at least one department.

Next, I have the Packager table with employee\_ID as the primary key. A packager is only a single employee and will operate in one and only one area and the warehouse in the area.

Next, I have the HR table with employee\_ID as the primary key. A packager is only a single employee and works in one and only one office. HR can have no or many complaints.

Next, I have Driver table with employee\_ID as the primary key. A driver is only a single employee who can work in only one warehouse of the area he/she is in. The driver is assigned one and only one route and car each day.

Next, I have the Area table with area as the primary key. An area can have no or many offices, warehouses, drivers, packagers and vehicles.

Next, I have the Office table with office\_location as the primary key. An office will always be in a single area but can have no or many HR and Manager.

Next, I have the Warehouse with warehouse\_ID as the primary key. A warehouse will only be in a single area but can have no or many products, drivers and packagers.

Next, I have the Product table with product\_ID as the primary key. A product may be in the ordered products many times or never. Product will either be or not be in the warehouse.

Next, I have the Order table with the order\_ID as the primary key. An order can only be placed by a single customer. It has at least one product which is ordered.

Next, I have the Ordered\_Product table with order\_ID and Product\_ID as the primary keys. The ordered products should only link to one order and each ordered product should be present in products.

Next, I have the Customer table with order\_ID as the primary key. Customer can have no or many complaints but every order has a separate customer’s details linked.

Next, I have the Complaint table with complaint\_ID as the primary key. A complaint is specific to only one customer and is assigned to only one HR.

Next, I have the Vehicles table with vehicle\_ID as the primary key. A vehicle is only from one area and it may or may not be assigned to a driver.

Next, I have the Route table with route\_ID as the primary key. A route can have no or many stops and may be assigned to no driver or many drivers.

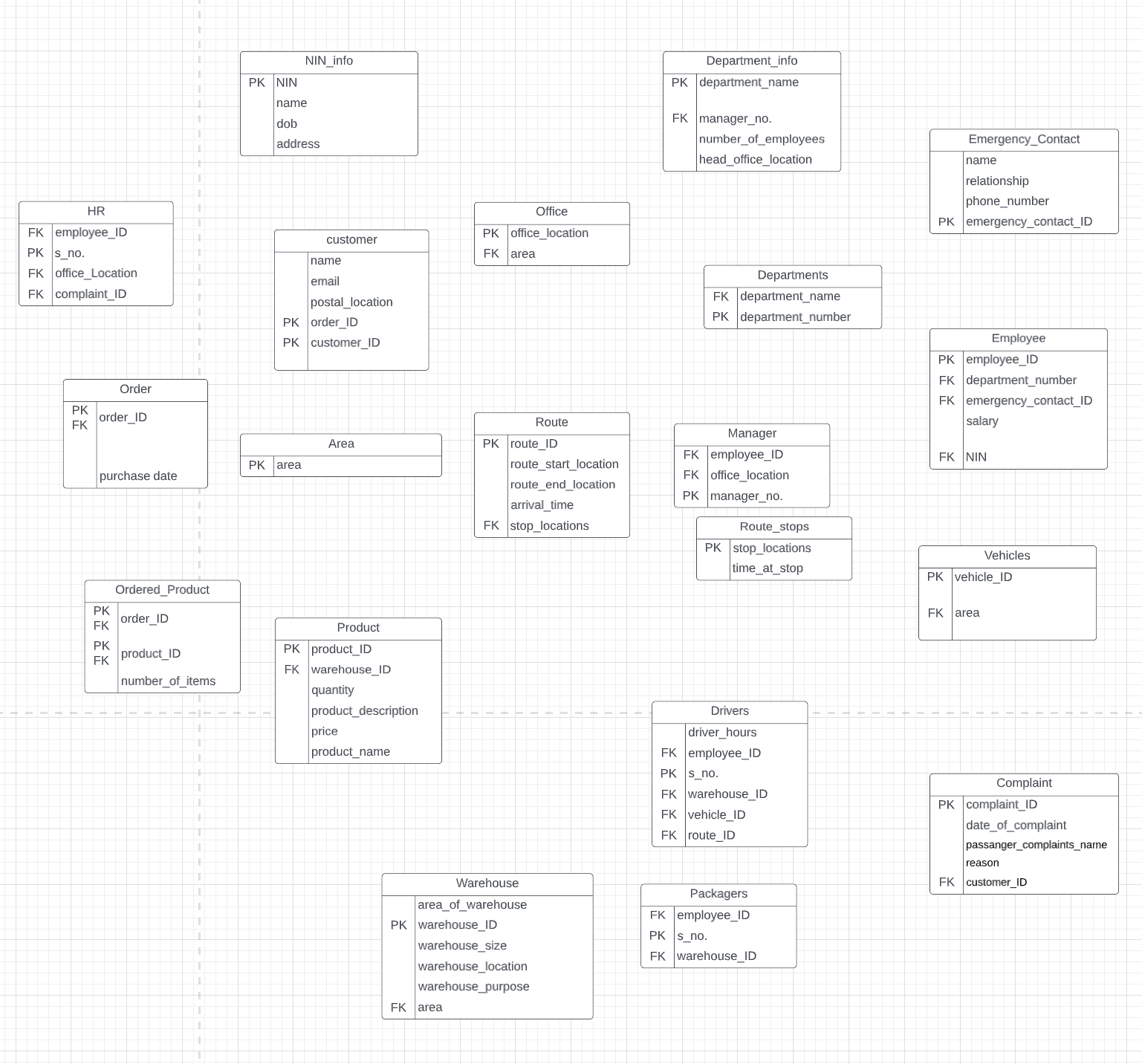
At last, I have the Route\_stops table with stop\_location as the primary key. Every stop will have one route.

2)NORMALISATION

**INTRODUCTION**

After creating ERD for the databases. Now we need to check for any redundancy in our tables to make it more organized by minimising redundancy which will be implemented making changes in our tables and making it follow 1NF,2NF, and 3NF. In order to show relations between various tables I have added foreign keys.

**DIAGRAM**



**RELATIONAL SCHEMA**

  Emergency Contact  (   Emergency\_Contact\_ID   ,   Relationship\_to\_employee   ,   Phone\_Number   ,  Name. );

  Area  (   area   );

  Vehicles  (   vehicle\_ID  );

FK area → Area(area)

  Route   (   route\_ID   ,   route\_start\_location   ,   route\_end\_location   ,   arrival\_time  );

FK stop\_locations → Route\_stops(stop\_locations)

  Route\_stops  (   stop\_locations   ,   time\_at\_stop   );

  Department\_info  (   department\_name   ,   number\_of\_employees   ,    Head\_Office\_Location   );

FK manager\_number → Manager(manager\_number)

  Warehouse  (   warehouse\_ID   ,  area\_of\_warehouse  ,   warehouse\_size   ,   warehouse\_location   ,   warehouse\_purose  );

FK area → Area(area)

  Drivers  (   s\_no.  ,  driver\_hours);

FK employee\_ID → Employee(employee\_ID)

FK warehouse\_ID → Warehouse(warehouse\_ID)

FK vehicle\_ID → Vehicles(vehicle\_ID)

FK route\_ID → Route(route\_ID)

  Packagers  (   s\_no.   );

FK employee\_ID → Employee(employee\_ID)

FK warehouse\_ID → Warehouse(warehouse\_ID)

  Product  (   product\_ID   ,  quantity   ,   product\_description   ,   price   ,   product\_name   );

FK warehouse\_ID → Warehouse(warehouse\_ID)

  Order  (   order\_ID   ,  purchase date );

FK order\_ID → Order(order\_ID)

  Ordered\_Product  (   order\_ID   ,   product\_ID   ,   number\_of\_items   );

FK order\_ID → Order(order\_ID)

FK product\_ID → Product(product\_ID)

  Manager  (   manager\_no. );

FK employee\_ID → Employee(employee\_ID)

FK office\_Location → Office(office\_Location)

  customer  (   order\_ID   ,   customer\_ID   ,   postal\_location   ,  name   ,  email);

  Complaint  (   complaint\_ID   ,   date\_of\_complaint   ,   passanger\_complaints\_name   ,   reason);

FK customer\_ID → customer(customer\_ID)

  Office  (   office\_Location  ,);

FK area → Area(area)

  HR  (   s\_no.   );

FK employee\_ID → Employee(employee\_ID)

FK office\_location → Office(office\_location)

FK complaint\_ID → Complaint(complaint\_ID)

  Departments (   department\_number   );

FK department\_name → Department\_info(department\_name)

  NIN\_info  (   NIN,   name   ,   address   ,   dob   );

  Employee  (   employee\_ID   ,    salary   );

FK department\_number → Departments(department\_number)

FK  emergency\_contact\_ID → Emergency\_Contact(emergency\_contact\_ID)

FK NIN → NIN\_info(NIN)

**REPORT**

1NF

I first checked that the tables are in 1NFwhich states that there should not be a repeating group and each attribute should be atomic. All the tables were indeed in 1NF.

For an instance if I would have the Route\_stops as an attribute to Route table this would have made the Route\_stops a multivalued attribute but we can see it has been created separately.

So, we conclude that this is in 1NF and move on to check for 2NF.

2NF

For this to be in 2NF it should 1st be in 1NF which we checked in the previous step. After that we need to check for partial dependencies and remove if any.

We go through all the tables and can clearly see that there exists no such table in which there are non-key attributes with partial dependencies.

So, we conclude the tables are in 2NF and move on to check on the 3NF.

3NF

For this to be in 3NF it should 1st be in 2NF which we have checked in the previous step. After that we need to make sure that there are no transitive dependencies within each table and remove if any.

We see that in the table Departments that two attributes, department\_name and department\_number, dependent on each other and the rest of the attributes (number\_of\_employees, head\_office\_location etc) are dependent on both of them. This creates a transitive dependency. To break this, I have split the table into two tables, Departments and Department\_Info with primary keys as department\_number and department\_name respectively. I have used department\_name as the foreign key in Departments. I have added all the other attributes like number\_of\_employees in the Department\_info.

We can also see that in the table Employee that two attributes, employee\_ID and NIN, dependent on each other and the some of the attributes (name, dob etc) are dependent on both of them. This creates a transitive dependency. To break this, I have split the table into two tables, Employee and NIN\_info with primary keys employee\_ID and NIN respectively. I have used NIN as the foreign key in Employee. I have used the attributes name, dob, address in NIN\_info .

Rest of the tables don’t have transitive dependencies.

Hence, we conclude that this is in 3NF.

**3)Relational Schema**

**INTRODUCTION**

I have represented all the connections after Normalising in the textual form shown below. Schema is a blueprint of our whole database.

**RELATIONAL SCHEMA**

  Emergency Contact  (   Emergency\_Contact\_ID   ,   Relationship\_to\_employee   ,   Phone\_Number   ,  Name. );

  Area  (   area   );

  Vehicles  (   vehicle\_ID  );

FK area → Area(area)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Route   (   route\_ID   ,   route\_start\_location   ,   route\_end\_location   ,   arrival\_time  );

FK stop\_locations → Route\_stops(stop\_locations)

ON DELETE SET NULL, ON UPDATE CASCADE

  Route\_stops  (   stop\_locations   ,   time\_at\_stop   );

  Department\_info  (   department\_name   ,   number\_of\_employees   ,    Head\_Office\_Location   );

FK manager\_number → Manager(manager\_number)

ON DELETE SET NULL, ON UPDATE CASCADE

  Warehouse  (   warehouse\_ID   ,  area\_of\_warehouse  ,   warehouse\_size   ,   warehouse\_location   ,   warehouse\_purose  );

FK area → Area(area)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Drivers  (   s\_no.  ,  driver\_hours);

FK employee\_ID → Employee(employee\_ID)

ON DELETE SET NULL, ON UPDATE CASCADE

FK warehouse\_ID → Warehouse(warehouse\_ID)

ON DELETE SET NULL, ON UPDATE CASCADE

FK vehicle\_ID → Vehicles(vehicle\_ID)

ON DELETE SET NULL, ON UPDATE CASCADE

FK route\_ID → Route(route\_ID)

ON DELETE SET NULL, ON UPDATE CASCADE

  Packagers  (   s\_no.   );

FK employee\_ID → Employee(employee\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK warehouse\_ID → Warehouse(warehouse\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Product  (   product\_ID   ,  quantity   ,   product\_description   ,   price   ,   product\_name   );

FK warehouse\_ID → Warehouse(warehouse\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Order  (   order\_ID   ,  purchase date );

FK order\_ID → Order(order\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Ordered\_Product  (   order\_ID   ,   product\_ID   ,   number\_of\_items   );

FK order\_ID → Order(order\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK product\_ID → Product(product\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Manager  (   manager\_no. );

FK employee\_ID → Employee(employee\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK office\_Location → Office(office\_Location)

ON DELETE NO ACTION, ON UPDATE CASCADE

  customer  (   order\_ID   ,   customer\_ID   ,   postal\_location   ,  name   ,  email);

  Complaint  (   complaint\_ID   ,   date\_of\_complaint   ,   passanger\_complaints\_name   ,   reason);

FK customer\_ID → customer(customer\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Office  (   office\_Location  );

FK area → Area(area)

ON DELETE NO ACTION, ON UPDATE CASCADE

  HR  (   s\_no.   );

FK employee\_ID → Employee(employee\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK office\_location → Office(office\_location)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK complaint\_ID → Complaint(complaint\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

  Departments (   department\_number   );

FK department\_name → Department\_info(department\_name)

ON DELETE NO ACTION, ON UPDATE CASCADE

  NIN\_info  (   NIN,   name   ,   address   ,   dob   );

  Employee  (   employee\_ID   ,    salary   );

FK department\_number → Departments(department\_number)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK  emergency\_contact\_ID → Emergency\_Contact(emergency\_contact\_ID)

ON DELETE NO ACTION, ON UPDATE CASCADE

FK NIN → NIN\_info(NIN)

ON DELETE NO ACTION, ON UPDATE CASCADE

**REPORT**

In order to prevent any data loss due to deletion in parent table and therefore FK can't be null in this case and always need to have some value in a table, we use NO ACTION so that if the FK is attempting to get deleted from its parent table, then action won’t be achieved. We use this in all except Department\_info, Routes and Driver where we use SET NULL constraint because if we delete the FK value in parent table then the FK value in the child table is automatically set to NULL. CASCADE is being used in all tables because if the primary key which is foreign key of the child table, gets updated then wherever the key has been used as foreign key also gets updated automatically.