**Database Development**

Airline Company assessment 1

University of Derby

100614702

Table of Contents

[1. Introduction 3](#_Toc119709580)

[2. Normalization 3](#_Toc119709581)

[2.1. Initial database 3](#_Toc119709582)

[2.2. Database Normalization 3](#_Toc119709583)

[2.3. Tables 6](#_Toc119709584)

[3. Entity Diagram 11](#_Toc119709585)

[3.1. Entity Diagram 11](#_Toc119709586)

[3.2. Entity Diagram from Mysql Workbench 11](#_Toc119709587)

[3.3. Entity Diagram from MS Access 13](#_Toc119709588)

[4. Function Demonstration 13](#_Toc119709589)

[4.1. Create 13](#_Toc119709590)

[4.2. Insert 14](#_Toc119709591)

[4.3. Delete 14](#_Toc119709592)

[4.4. Update 15](#_Toc119709593)

[4.5. Select 16](#_Toc119709594)

[5. Evaluation 16](#_Toc119709595)

[6. Reference 17](#_Toc119709596)

# Introduction

The development of this project requires the knowledge of flowchart, Entity Relationship diagram and how to normalize the table according to the 1NF, 2NF, 3NF and so on if required. This project gives the introduction of how professional databases are build and what all complexity can be faced while building up.

In the long run of history, there is always need of something to store information and computing history do dire need of database. Throughout the history there appeared vison of database development and one of the first practical databases is relational database. This database is so popular that it is still getting used by larger companies. To implement relational database there are many software such as MySql, SQL server that build to main data stably and effectively with lots of tools assist the programmer.

In this assignment about the Airline Company database design, before developing database there needs to be through analyse the relationship and connections of the data given to us, understanding what all things are required and what not. After that, next steps come to highlight all the entities and its attributes for the Entity-Relationship diagram. Here this ER diagram is unnormalized and needs to process by applying all Normal Forms ( 1NF, 2NF and 3NF). Further down comes preview of all the tables build in MySQL Workbench. Finally, demonstration of CRUD operation are shown with codes and screenshots.

# Normalization

## Initial database

The given data needs to be normalized before designing it to database software.

|  |  |
| --- | --- |
| **Table** | **Columns** |
| Flight | Flight\_Id, Pilot\_Id, Date, Origin, Dest, Arr\_time, Dep\_time |
| Passenger | Passenger\_Id, Name, Surname, Address, Phone |
| Booking | Flight\_Id, Passenger\_Id, Charge, Type |
| Staff | Emp\_Id, Name, Surname, Address, Phone, Salary |
| Crew | Crew\_Id, Emp\_Id, Flight\_Id |
| Pilot | Pilot\_Id, Rating, Emp\_Id, Airplane\_Id |
| Airplane | Airplane\_Id, Manufacturing, Model\_Number, Type |

## Database Normalization

Normalization is used to organise the data in a database. This includes creating tables and connection between those tables following the rules to protect the data and make the database more flexible by removing redundancy and inconsistent dependency[1].

Redundant data wastes storage space and create maintenance problems. If specific data exist at more than one place and that need to change then it need to change at all stored places . Inconsistency comes with redundant data as some data may remain unchanged due to unnormalized that created some of the records to be inconsistent with same record stored at different place.

**1st Normal Form**

First Normal Form signifies that if a relation contains a mutli-valued attributes then it violates the first normal form. A database is said to be in First Normal if :-

* + 1. There are only single valued attributes.
    2. Attribute domain does not change.
    3. There is a unique name for every column.
    4. The order in which data is stores not matter.

In Flight table, arrival time and departure time will have multiple values as it can have one or more stops. So, Arr\_time and Dep\_time needs to be separated from Flight table.

For rest of tables, there is no need to change anything. As every attribute are atomic.

|  |  |
| --- | --- |
| **Table** | **Columns** |
| Flight | Flight\_Id, Pilot\_Id, Date, Origin, Dest |
| Stop | Stop\_Id, Flight\_Id, Place, Arr\_time, Dep\_time |
| Passenger | Passenger\_Id, Name, Surname, Address, Phone |
| Booking | Flight\_Id, Passenger\_Id, Charge, Type |
| Staff | Emp\_Id, Name, Surname, Address, Phone, Salary |
| Crew | Crew\_Id, Emp\_Id, Flight\_Id |
| Pilot | Pilot\_Id, Rating, Emp\_Id, Airplane\_Id |
| Airplane | Airplane\_Id, Manufacturing, Model\_Number, Type |

**2nd Normal Form**

First Normal form doesn’t remove redundancy, but it removes repeating groups[2].

To make table second normal, table should:-

1. Follow 1st Normal form.
2. There should not be a partial dependency i.e no non-prime attribute is dependent on any proper subset of any key of the table.

In **Flight** table, there are more than one pilots that are assigned to same flight. So, we create separate table named FlightPilot to separate related information of a flight from pilot information.

In the **Booking** table, composite key based on Flight\_Id and Passenger\_Id is dropped and **Booking\_Id** is assigned primary key. The attributes charge and Seat type are dependent on flight\_Id, it doesn’t fully dependent on booking\_Id. So this information will be stored in separate table called **Ticket** table and Ticket\_Id is added to **reservation** table.

Airplane table is assigned a unique Airplane\_Id.

|  |  |
| --- | --- |
| **Table** | **Columns** |
| Flight | Flight\_Id, Date, Origin, Dest |
| Flight\_Pilot | FlightPilot\_Id, Flight\_Id, Pilot\_Id |
| Stop | Stop\_Id, Flight\_Id, Place, Arr\_time, Dep\_time |
| Passenger | Passenger\_Id, Name, Surname, Address, Phone |
| Booking | Booking\_Id, Passenger\_Id, Ticket\_Id |
| Ticket | Ticket\_Id, Flight\_Id, Charge, Type |
| Staff | Emp\_Id, Name, Surname, Address, Phone, Salary |
| Crew | Crew\_Id, Emp\_Id, Flight\_Id |
| Pilot | Pilot\_Id, Rating, Emp\_Id, Airplane\_Id |
| Airplane | Airplane\_Id, Manufacturing, Model\_Number, Type |

**3rd Normal Form**

A relation is in third normal form if there is no transitive dependency for non- prime attributes as well as it is in second normal form.

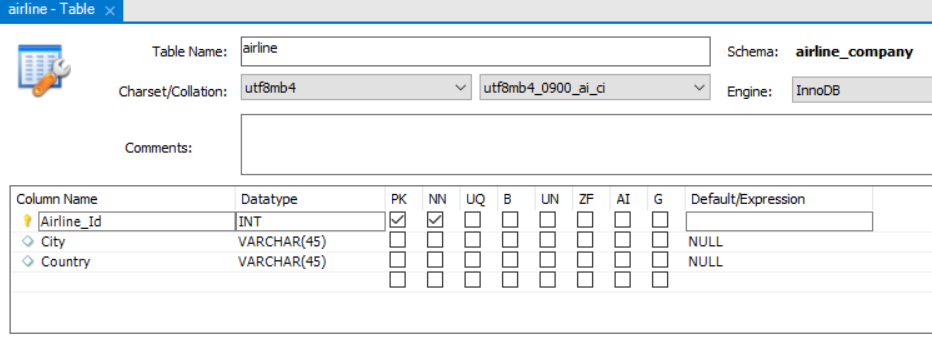
The origin and destination of a **Flight** table depends on airplane and not on any specific flight instead it is independent entity. The repetition of location is highly likely, so it needs to be stored in different table called **Airline** and used as a foreign key in the flight table.

Pilot fly airplane based on its rating. So, NumSer in Pilot table is dependent on the rating instead of Pilot\_Id. To remove this transitive dependency a separate table named **Rating** is created to store and type of planes that pilots can fly with those ratings.

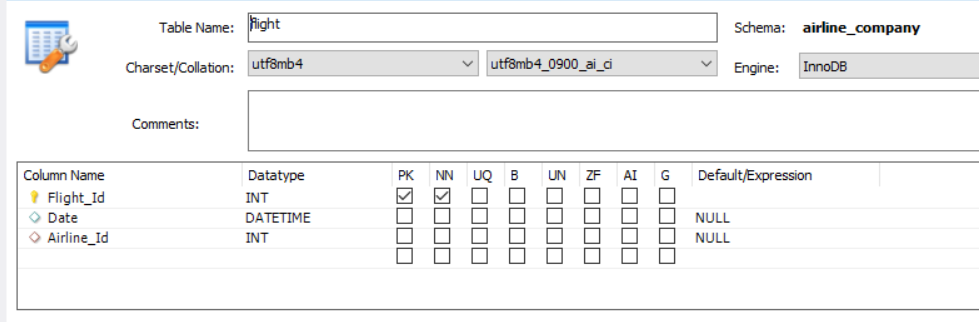
|  |  |
| --- | --- |
| **Table** | **Columns** |
| Airline | Airline\_Id, City, Country |
| Flight | Flight\_Id, Date, Airline\_Id |
| Flight\_Pilot | FlightPilot\_Id, Flight\_Id, Pilot\_Id |
| Stop | Stop\_Id, Flight\_Id, Airline\_Id, Arr\_time, Dep\_time |
| Passenger | Passenger\_Id, Name, Surname, Address, Phone |
| Booking | Booking\_Id, Passenger\_Id, Ticket\_Id |
| Ticket | Ticket\_Id, Flight\_Id, Charge, Type |
| Staff | Emp\_Id, Name, Surname, Address, Phone, Salary |
| Crew | Crew\_Id, Emp\_Id, Flight\_Id |
| Pilot | Pilot\_Id, Rating\_Id, Emp\_Id |
| Rating | Rating\_Id, Rating, Airplane\_Id |
| Airplane | Airplane\_Id, Manufacturing, Model\_Number, Type |

## Tables

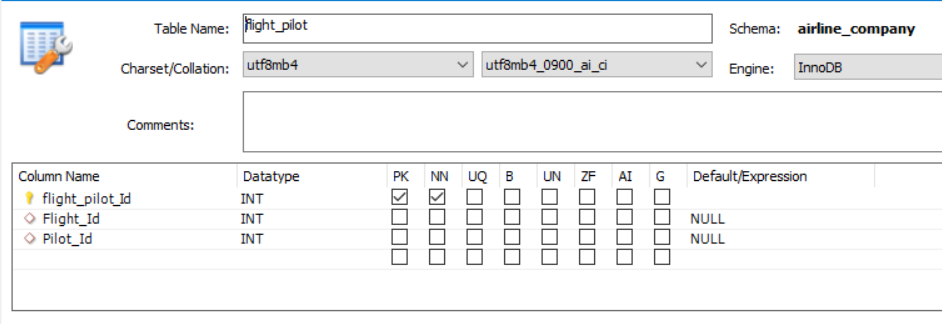
1. Airline



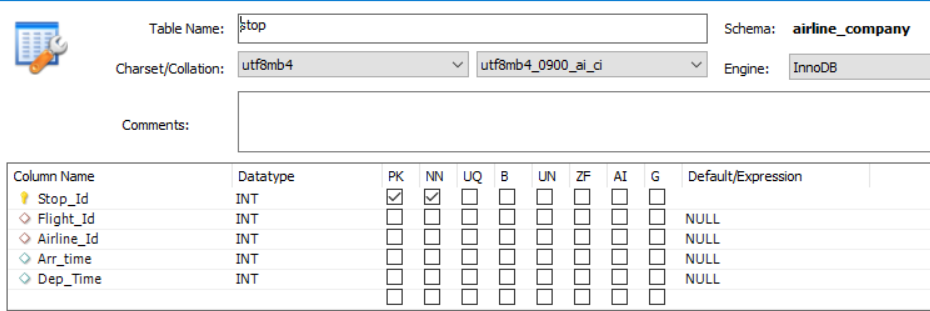
1. Flight



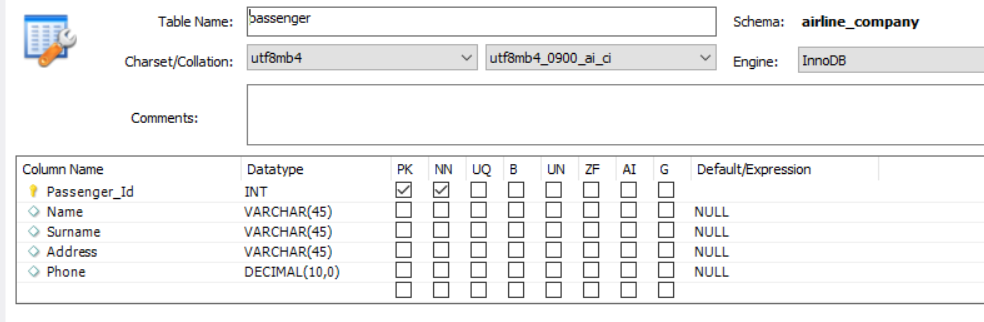
1. Flight\_Pilot



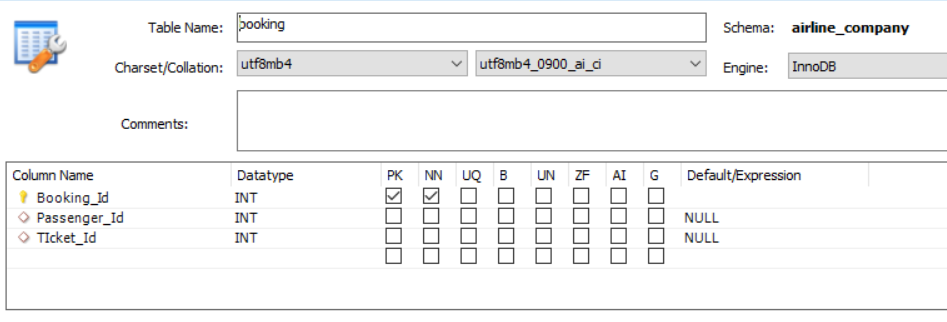
1. Stop



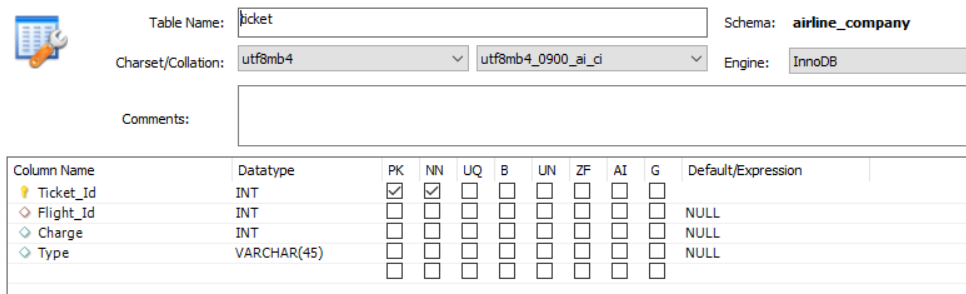
1. Passenger



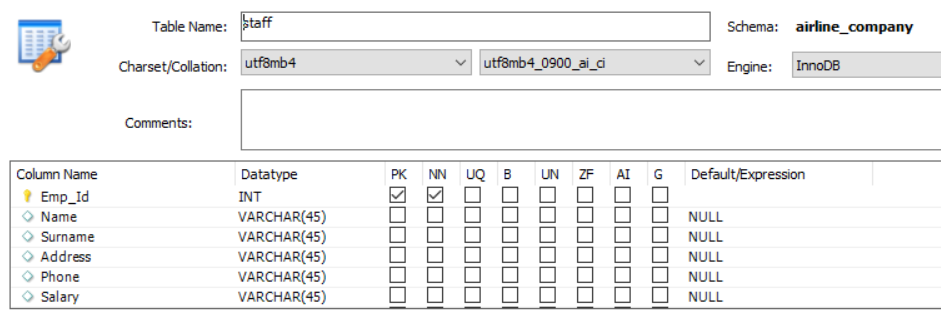
1. Booking



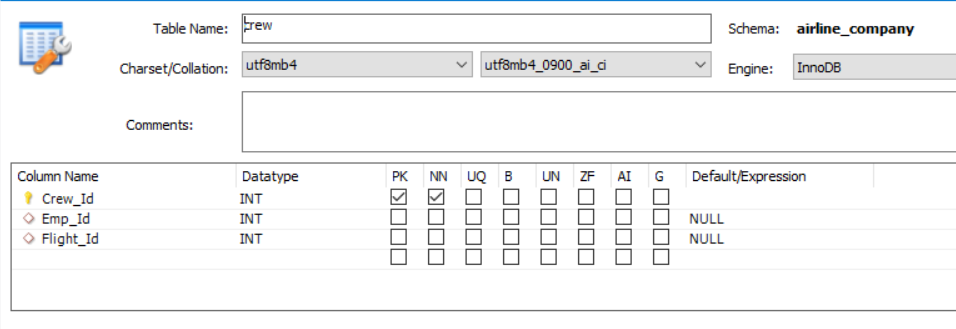
1. Ticket



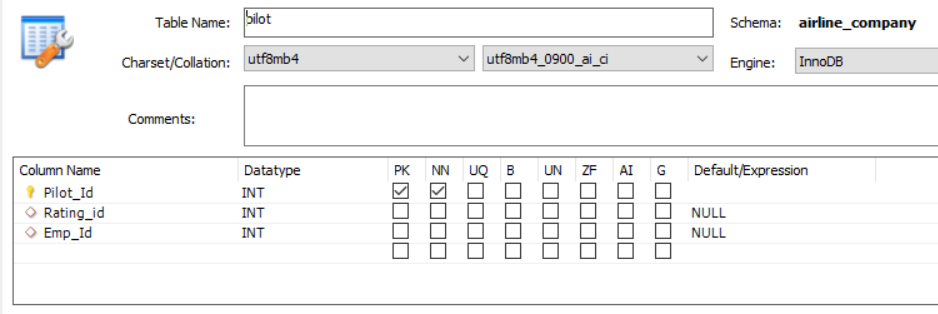
1. Staff



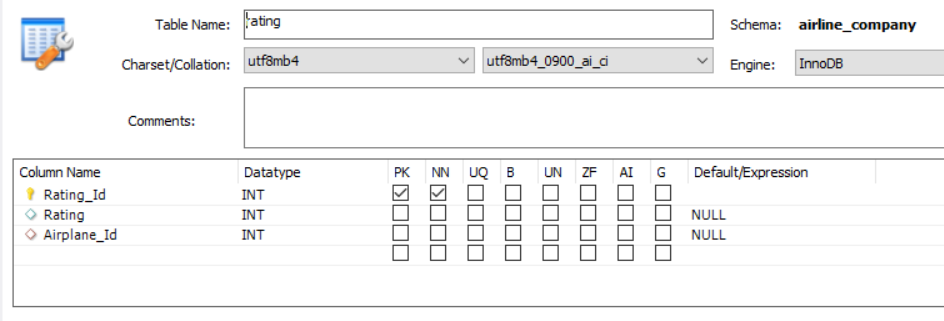
1. Crew



1. Pilot



1. Rating



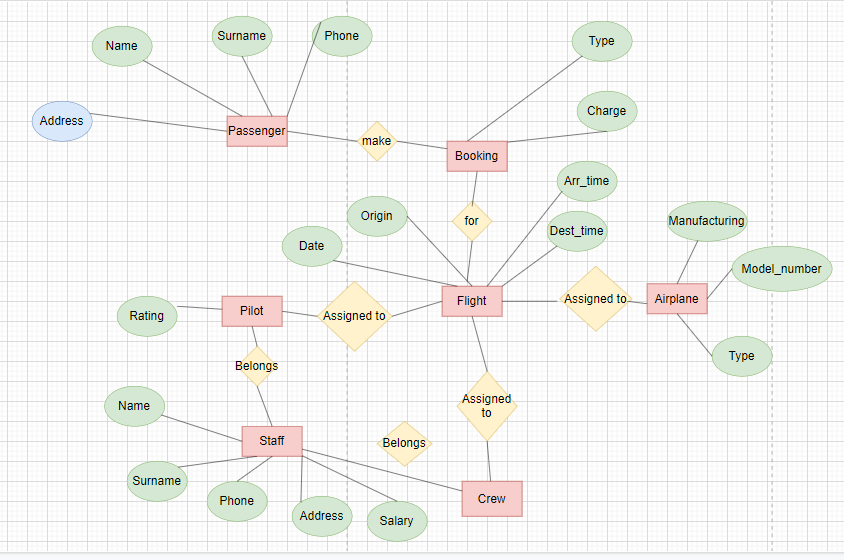
1. Airplane

Table

Description automatically generated

# Entity Diagram

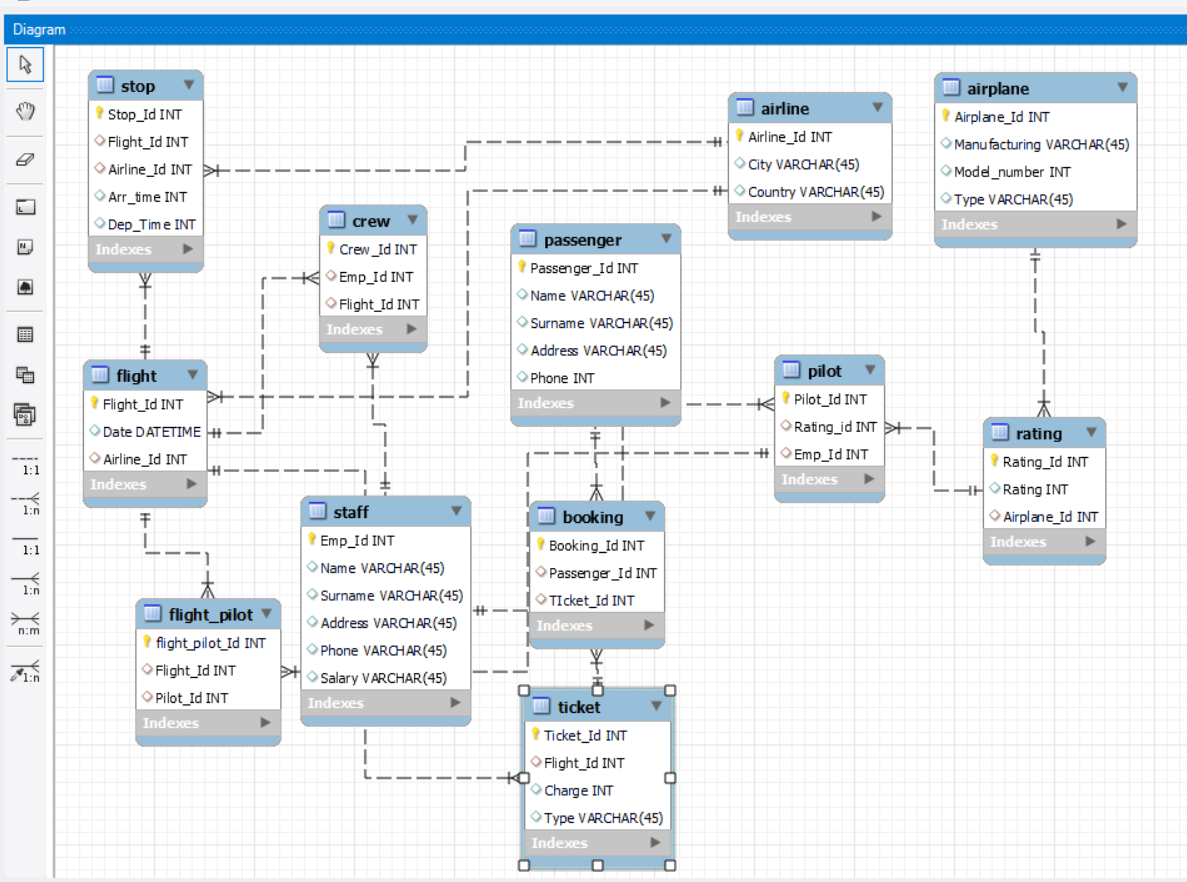
## Entity Diagram



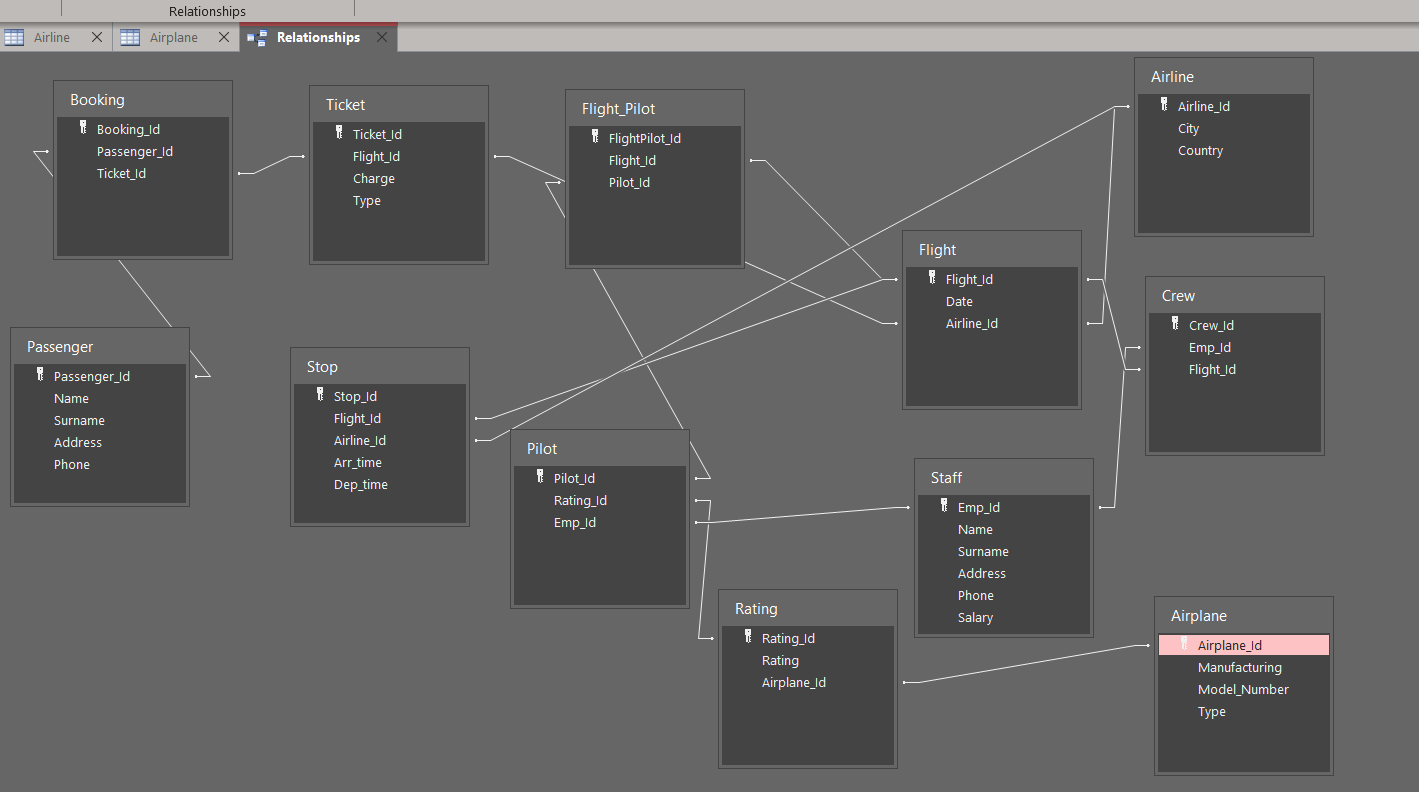
This E-R diagram describes the connection and relationship among tables and attributes.

The flow can be seen that Passenger with all its details makes booking with providing charge for the booking and selecting the type of seats eg normal, premium, business class. After that Booking get made with flight of airplane mentioning its date, origin, destination and time of arrival and departure for number of its stops. Pilot is assigned to the flight with all crew member present in the plane. There is rating of pilot which tells about what all airplane pilot can fly with its rating.

## Entity Diagram from Mysql Workbench



## Entity Diagram from MS Access



# Function Demonstration

These examples are to show CRUD operation based on Mysql workbench database :-

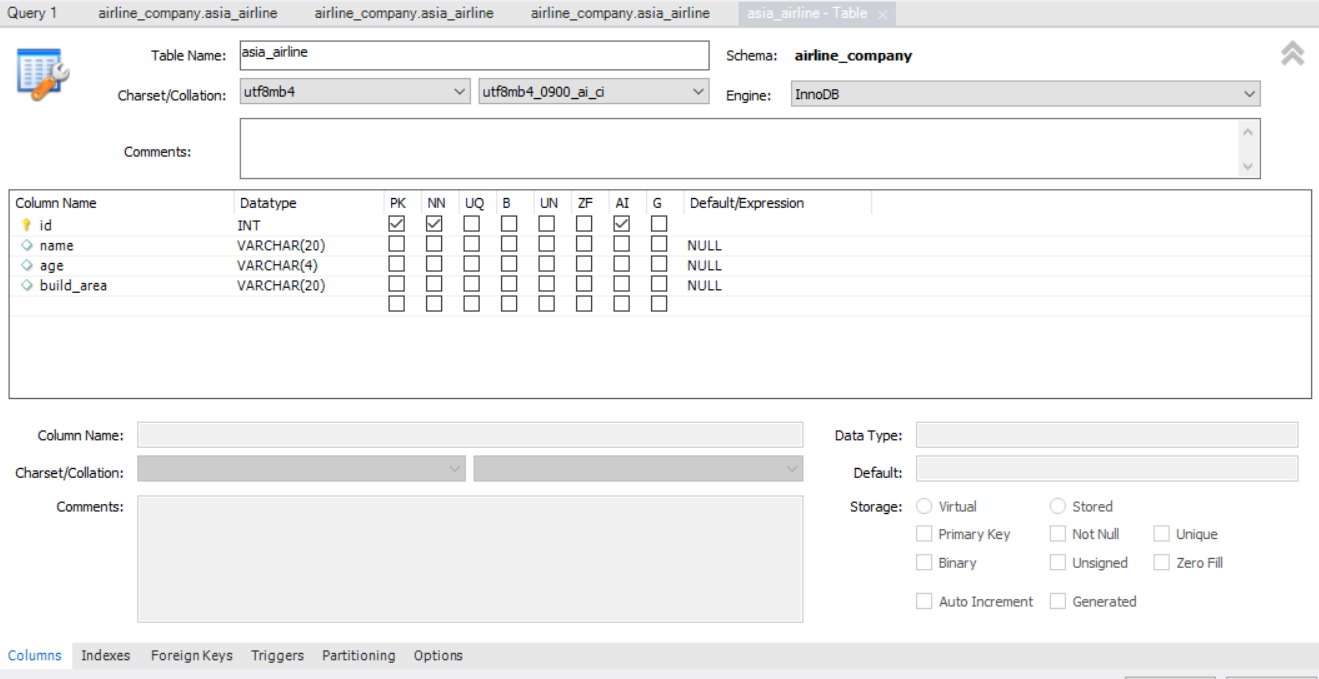
## Create

Create a table called asia\_airline and add its attributes. These sql code aims to create a table. Graphical user interface, text, application, chat or text message

Description automatically generated

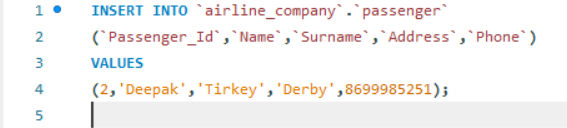
Information given by Mysql

The table has been created succssfully, below is the screenshot



## Insert

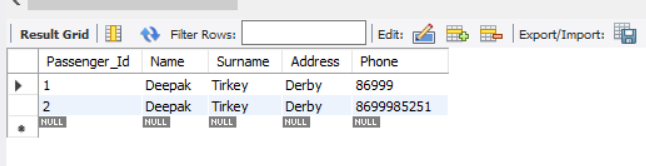
This part is where a piece of data is inserted to passenger table. Below is the code for inserting function



Operation information for the insert function.



This table shows the data has been inserted to this table.



## Delete

Deleting the piece of data from table. Here is the preview of data.

Graphical user interface, application

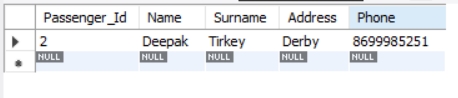
Description automatically generated

Sql code for delete function. Graphical user interface, text, application, email

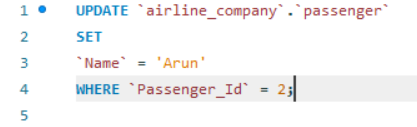
Description automatically generated

## Update

Update function help to change the value of the data inserted in the database. The table below is data before update.



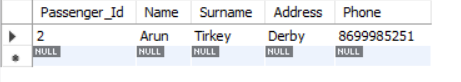
Here is the code for details of update.



Mysql action completed notification.

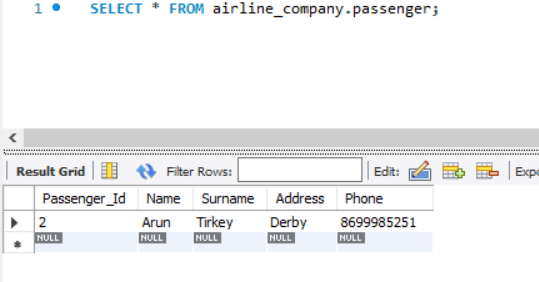


This table below shows the data has been updated with inputted data.



## Select

Select function is used to show all data or required data of a table in the database. Picture below shows the code and the display of the code.



# Evaluation

Creating database in database software is very simple given, we fulfill all the requirement such as providing primary keys for all the tables and data types.

Only problem faced while normalizing the tables, it is difficult to normalize all tables considering very few detailed examples were given on internet.

Creating database in MySQL is straight forward with user friendly GUI that assist with all the requirement already provided on the screen.

SQL implementation in MS Access is little difficult as compared to MySQL workbench as there is not automatic ER diagram generator, it needs to be created by dragging all the tables and make connections between the tables.

# 6. Reference

[1]. Microsoft Article 2007, “Description of the database normalization basics”, last visited 18/1/2022,

<https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description>

[2]. https://www.geeksforgeeks.org/