Generation EasyJet: A flight booking system and how it operates

Damien McGloin

40000631

MSC Software Development

December 2020

Contents

Introduction

The goal of this project was to reverse engineer EasyJet's web database. The first step was the creation of an ER diagram. This approach was chosen as the entity-relationship model often reflects the real world more accurately than a hierarchical model. This step was a team effort though. Group 1, which includes me and seven other classmates, took part in the process of entity discovery. Working individually, we each searched the EasyJet website for entities and their relationships and then combined our findings. As expected, there was a great deal of overlap, but we were able to cover the website quite thoroughly. The next step was drawing the first draft of our ER diagram which you can see in the appendix [1].

After completing our first draft there were several issues. Many of the tables were missing unique keys and foreign keys which are necessary for running SQL queries and connecting tables with foreign key constraints. Some of the attributes we included such as 'medical certificate' would only be presented at the airport and do not play a role in booking a flight. We had also decided to normalise out dates into a separate 'Calendar' table, but it was discovered that using the 'date' data type was much more efficient, so the 'Calendar' table was removed entirely. Similarly, age was normalised out to its own table to preserve data integrity. However, normalisation will be discussed in greater detail in its own section. A final draft of our group's ER diagram is also included in the appendix [2].

After the completion of our final group draft several noteworthy alterations had been made. Each table had been assigned a primary key which would auto increment. This would ensure each table had an identifier which could be considered unique on a global scale as even a passport number might not be unique across the world or a flight number could be changed causing issues with the database. Cardinality constraints were also implemented. They are used to highlight not only the relationships between tables but also the kind of relationship that exists for example this could be a one to many or many to many relationship. Although there is a school of thought that many to many relationships can be considered bad database design it was considered the best method of allowing certain actions and this will be explained in greater detail later in the report. The next step in this project was the construction of our own individual databases on myPhpAdmin. One of my goals with this project was to use real world data where possible. This includes accurate pricing information as well as real flights, dates, times, seating plans and flight codes. The passenger list includes the names of my fellow classmates with variations to age and travel reasons to show variety as can be seen in image 3 below [3].

Title	PassengerFirstName ▲ 1	PassengerSurname	Age	PassengerType	TravelReason
Mr	Adam	Gavigan	18+	Adult	Business
Mr	Aidan	McGowan	18+	Adult	Leisure
Mr	Alexander	O'Neill	18+	Adult	Leisure
Miss	Amy-Lee	Connon	15	Child	Leisure
Mr	Aravinth	Ravichandiran	18+	Adult	Business
Miss	Caitlin	O'Hara	17	Adult	Leisure
Miss	Carla	Rush	18+	Adult	Leisure
Mr	Chris	Forsythe	17	Adult	Business
Mr	Connor	Lennon	14	Child	Leisure
Mr	Conor	Loughran	18+	Adult	Leisure
Mr	Damien	McGloin	18+	Adult	Leisure

Image 1: Some of the passenger information present in the database

The booker information includes the names and information of several staff members from the QUB school of EEECS such as my current lecturers. However, for obvious security reasons all card information or passwords included in the database are entirely fictitious. Booking reference codes are also fake as it would not be possible to use real codes in this case. An example of some of the booker information can be seen in image 4 below [4].

BookerFirstName	BookerSurname	EmailAddress	AES_DECRYPT(`UserPassword`,'mySecretKey')
Neil	Anderson	n.anderson@qub.ac.uk	ILoveDatabases28
Moira	Watson	m.watson@qub.ac.uk	ComputingFoundationsRocks10
Aidan	McGowan	aidan.mcgowan@qub.ac.uk	Programming4Life
John	Busch	j.a.busch@qub.ac.uk	WebDevAllTheWay15
Matthew	Collins	m.collins@qub.ac.uk	random48
Michael	Cregan	m.cregan@qub.ac.uk	random97
Andrew	McDowell	andrew.mcdowell@qub.ac.uk	random90
Ian	O'Neill	i.oneill@qub.ac.uk	random13
Darryl	Stewart	Dw.Stewart@qub.ac.uk	random37
Paul	Sage	P.Sage@qub.ac.uk	random28
Maire	O'Neill	m.oneill@ecit.qub.ac.uk	random19

Image 2: Booker details including decrypted password information

2. Scope of the project & overall design

During the early stages of planning this project it was decided that certain elements of the flight booking system such as hotel selection, cars and insurance would be deemed outside the scope of this database. While the EasyJet flight booking system does allow the booker to select elements other than flights the data required for this process would be stored and handled by one of EasyJet's partner companies such as booking.com or Zurich. Although a real-world EasyJet database designer would most likely record a booker's decision to select one of these options, I decided to focus solely on the other elements of the flight booking system.

For this project I had five main concerns:

1. To demonstrate how someone would be able to book a flight with the possibility of selecting multiple seats and bag types.
2. The booker should be able to request special assistance and purchase items from the website's digital catalogue such as a meal voucher.
3. To reflect the website's ability to update as new flights, flight times, prices and other information is updated.
4. To allow EasyJet staff the ability to find helpful information which would enable them to perform their job better.
5. To limit the possibility of inconsistent data i.e., data that is updated in one table but not in others.

The ability to update prices and other information will be covered in section 4 alongside several other design choices I made. The last two of these concerns will be discussed in greater detail in section 5 below which focuses on normalisation. The following section will discuss how the first two concerns were addressed by examining the process of booking a flight.

3. A walkthrough of the booking system

The process of booking a flight begins with the booker selecting a departure and arrival point. The booker will also select dates for departure and or return along with the number of passengers who are classified by three different age types. It is important to note that EasyJet will allow an online booking for up to forty people with the additional caveat of one infant per each adult traveling. Also, a maximum of ten children can travel with one adult. This makes the process of booking more complex than it seems at first glance. However, it is assumed that many of these conditions would be handled by other developers and not the database designer.

My database reflects this process firstly by searching the database for flights which match the search criteria. An SQL query [5] returns the following option when searching for a flight from Belfast International to Alicante which leaves on December 19th and returns on December 26th. A search result for this criteria can be found below.

FlightID	FlightNumber	FlightDeptDate	FlightDeptTime	FlightArrDate	FlightArrTime	DepartureAirport	ArrivalAirport	FlightPrice
1	EZY6703	2020-12-19	15:15:00	2020-12-19	19:15:00	Belfast International	Alicante	228.99
2	EZY6704	2020-12-26	17:55:00	2020-12-26	20:05:00	Alicante	Belfast International	23.99

Image 3: Search results for flights from Belfast to Alicante on specified date

Next a database query [6] determines the remaining seats available in the seating table by comparing the assigned seats with the unassigned seats. As we can see in the image below the plane to Alicante has six remaining seats, but the return flight is fully booked. The booker will find they can proceed with the departing flight but for their selected return date there are no available flights.

COUNT(SeatID) - COUNT(PassengerID)	COUNT(SeatID) - COUNT(PassengerID)
6	0

Image 4: Remaining seats for the planes with flightID 1 and flightID 2

If the booker proceeds, they can select seats from an interactive image showing the seat layout of the plane used for the departing flight. An SQL join query [7] can be used to show the available seats on the plane as well as the type of seats available and other information such as their additional benefits. A description of the flight range is also included here based on the assumption that seats on a short-range flight are cheaper than those on a long-range flight. However, flight range is not visible to the booker during the process of seat selection. This information is only to assist EasyJet in determining seat prices on each flight.

RowNumber	SeatLetter	SeatingTypeName	AdditionalBenefits	SeatPrice	FlightDesc
13	C	Extra Legroom	2 cabin bags, dedicated bag drop, speedy boarding	24.99	Mid Range
13	D	Extra Legroom	2 cabin bags, dedicated bag drop, speedy boarding	24.99	Mid Range
30	A	Standard	Choose where you want to sit, window, middle or ai...	5.99	Mid Range
30	B	Standard	Choose where you want to sit, window, middle or ai...	5.99	Mid Range
31	C	Standard	Choose where you want to sit, window, middle or ai...	5.99	Mid Range
31	D	Standard	Choose where you want to sit, window, middle or ai...	5.99	Mid Range

Image 5: Available seats on Flight EZY6703 including price, seat type and benefits

The next step considered in the flight booking process is baggage selection. Again, it is important to note that passengers are limited to a maximum of three hold items, but it is assumed that enforcing this limitation on a booker is outside of the concern of the database designer. While baggage prices will vary from flight to flight it is again assumed that it will be more expensive on a flight traveling a longer distance so once again flight range is used in assigning a price. It was also important that passengers can select multiple bags and sports equipment. For this reason, I chose to employ many to many tables. The image below shows how these tables were designed.

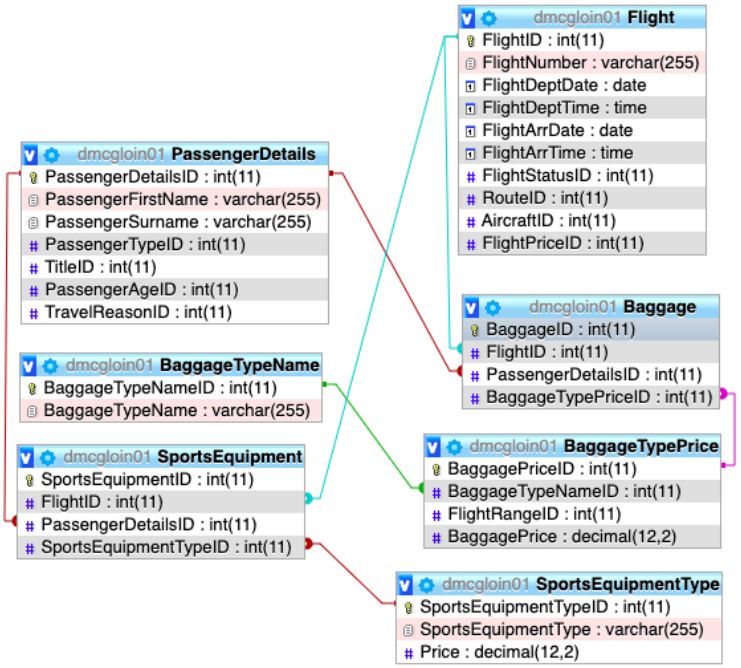


Image 6: Designer view demonstrating implementation of many to many tables for baggage and sports equipment selection

In the above image we can see that the 'Baggage' table has been designed so that information such as the names and prices of different baggage types which appear once in their respective tables can appear many times in the 'Baggage' table and be assigned to multiple different passengers. The 'SportsEquipment' table was designed in an identical fashion so this means the booker can select multiple bags and sports equipment of different types on behalf of the passengers. Similarly, a 'SpecialAssistance' table was

designed to allow a booker to select multiple different assistance requirements such as a nut allergy. The final steps involved in the booking process are entering booker details, passenger details and the confirmation of payment.

In order to store a record of the items purchased a 'CostLineItems' table was created. This can be seen in the image below. This functions as a receipt table, independent of any fixed prices featured in the database. This means any update to the pricing of flights, seats, or additional items after the date of booking will not affect this record of purchase and allows the booker to attain a correct refund if it is necessary.

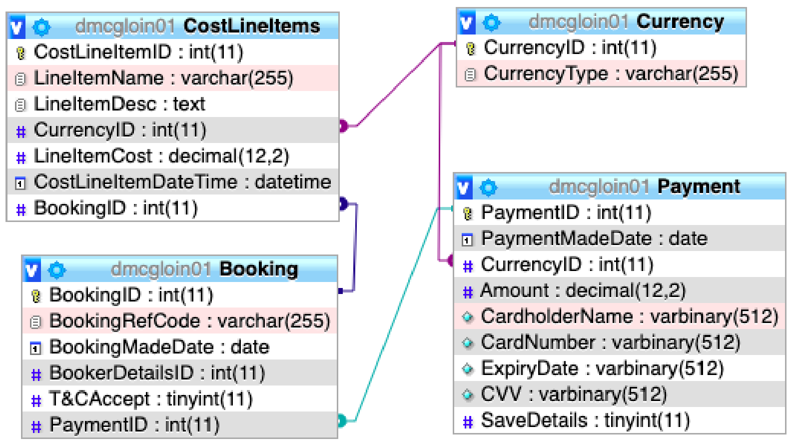


Image 7: Structure of database tables for recording purchases of each booker

An SQL join query [8] allows for a summary of all purchases made by each booker. An example of this can be seen below with Mr. Neil Anderson who kindly purchased four tickets to Alicante for some of his students alongside bags and meal vouchers.

Title	BookerFirstName	BookerSurname	LineItemName	LineItemDesc	CurrencyType	LineItemCost	CostLineItemDateTime	BookingRefCode
Mr	Neil	Anderson	Adult Ticket	Belfast International to Alicante Dec 19th	GBP British Pounds	228.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Adult Ticket	Belfast International to Alicante Dec 19th	GBP British Pounds	228.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Adult Ticket	Belfast International to Alicante Dec 19th	GBP British Pounds	228.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Adult Ticket	Belfast International to Alicante Dec 19th	GBP British Pounds	228.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	23kg bag	Mid range flight	GBP British Pounds	14.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	23kg bag	Mid range flight	GBP British Pounds	14.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	29kg bag	Mid range flight	GBP British Pounds	38.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	26kg bag	Mid range flight	GBP British Pounds	26.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Meal Voucher		GBP British Pounds	4.00	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Meal Voucher		GBP British Pounds	4.00	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Meal Voucher		GBP British Pounds	4.00	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Meal Voucher		GBP British Pounds	4.00	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Extra legroom seat	Mid range flight	GBP British Pounds	24.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Up front seat	Mid range flight	GBP British Pounds	15.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Up front seat	Mid range flight	GBP British Pounds	15.99	2020-11-25 17:23:00	EL5C20G
Mr	Neil	Anderson	Up front seat	Mid range flight	GBP British Pounds	15.99	2020-11-25 17:23:00	EL5C20G

Image 8: Record of purchases made by Mr. Neil Anderson

With this I have demonstrated how someone could book a flight, make additional purchases and request special assistance for any health requirements they might have. The next section will focus on some of the difficult choices faced while designing the database including how to update some data and store personal information.

4. Justifying the design

Initially when considering the issue of seating I intended to connect each booking to the number of seats purchased as it appeared that the actual seat assignment did not occur at this stage. Further research showed that seat allocation does occur at this stage and as a result my database was revised to reflect this. To ensure that staff and bookers can query the available seats on a flight the database was designed to store individual seats for each flight. However, it would not be possible to accurately reflect the EasyJet website as a single flight could hold up to 235 passengers. Instead, a total of 78 seats were stored in the database with 26 of those seats being allocated to flight EZY6703 traveling from Belfast to Alicante.

For the issue of pricing, it was important to represent not just the price of flights shown on the day that I researched them but also represent how they might be updated in real time. For this I created a 'FixedFlightPrices' table which stores the price of each flight as well as the date it was priced. This could be updated with an SQL query [9] to reflect how prices change based on seat availability and how long remains until the departure day. A second concern with regards to pricing is how a child and infant's ticket price would be calculated and stored in the system. However, the EasyJet policy with regards to the pricing of a child's ticket makes this simple. If leaving from a U.K airport a child's ticket is 13 pounds cheaper (or the equivalent in another currency e.g., 15.33 cents). However, if leaving from an airport outside the U.K the price of an adult's ticket and a child's ticket is the same. Therefore, I designed my database with the assumption that rather than storing separate prices for child tickets the website's pricing system would be coded to automatically subtract 13 pounds from an adult ticket if the flight is departing from a U.K airport. There is a flat 25-pound charge for an infant even if an adult's ticket is cheaper than this, so this price was stored alongside several other fixed prices such as the meal voucher in a 'CatalogueType' table.

At first, I intended to store prices in multiple currencies as I believed this would reflect EasyJet's actual website more accurately. While storing so much data for prices is entirely possible the main issue would be updating this regularly. Not only would flight prices require updating but also seat prices as well. However, it seems EasyJet also recognized this as an issue and employ an algorithm for converting the cost. We can see an indication of this in the image below.

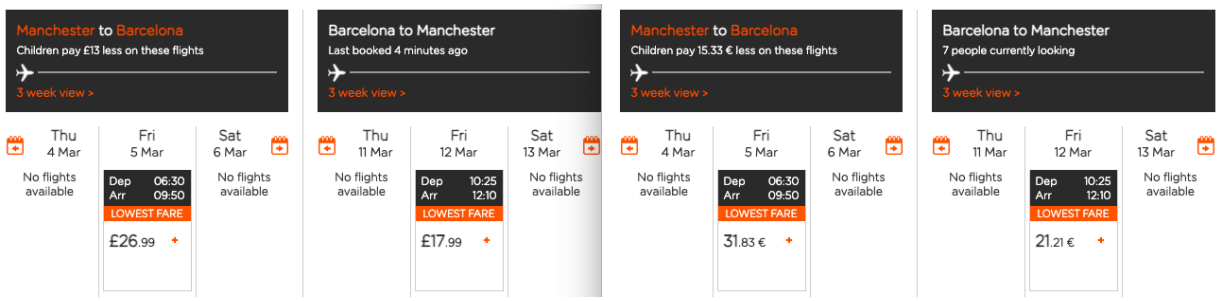


Image 9: Screenshot of pound to euro price comparison

We can see on the left a typical pricing format of 26.99 and 17.99 but on the right the euro pricing is stored with the less appealing numbers 31.83 and 21.21. Several flight searches using other currencies such as dollars yielded similar results. Therefore, a currency conversion algorithm is most likely employed to simplify the process of updating multiple prices simultaneously on the website. My database was designed to be consistent with this. The total amount stored in the payment table and the price stored in

the ‘CostLineItem’ table contain the currency and the amount the customer would see on the actual EasyJet website.

In order to more accurately reflect a real-world database, it was important to also consider encryption. The decision to store any personal information, especially card information, is not a decision to take lightly. I decided to use the built-in AES_ENCRYPT function in MySQL but it’s important to note that this is a low-level form of encryption and a real-world database designer would use a different language such as PHP for encryption. An example update query can be seen in the appendix. It is important to note the inclusion of a secret key, in this case ‘mySecretKey’. While a different secret key could be used for each booker the same key was employed in each case for the sake of simplicity. This is the key for decrypting the database. An example of the decrypted card information for some of the bookers can be seen in the image below [10].

PaymentMadeDate	CurrencyType	Amount	AES_DECRYPT(CardholderName, 'mySecretKey')	AES_DECRYPT(CardNumber, 'mySecretKey')	AES_DECRYPT(ExpiryDate, 'mySecretKey')	AES_DECRYPT(CVV, 'mySecretKey')
2020-11-25	GBP British Pounds	1100.88	Neil Anderson	3474014867541489	2024-07-20	400
2020-07-23	GBP British Pounds	1613.82	Moira Watson	6763435592724739	2024-04-25	156
2020-09-26	GBP British Pounds	955.92	Aidan McGowan	5105662654624451	2024-10-03	938
2020-11-14	GBP British Pounds	1109.50	John Busch	3562154722883880	2023-06-15	362
2020-04-01	GBP British Pounds	382.93	Matthew Collins	6762091800137717	2023-09-17	114

Image 10: Some of the booker information in the database including decrypted fake card details

Lastly, I’d like to draw attention to the inclusion of some of the more unique date types. Some of the tables featured such as ‘BookerDetails’ contain columns with yes or no values for example ‘EasyJetOffersAndUpdates’. The data type selected was tinyint. Although the bit data type which only has a possible value of 0 or 1 could be a viable option there are merits to each data type and the decision seems to be largely a matter of preference. For columns which feature a yes or no selection tinyint was selected with 1 being chosen to represent yes and 0 representing no.

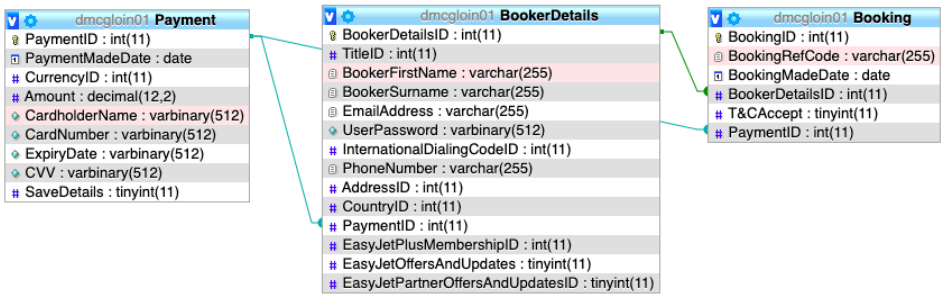


Image 11: Example of tinyint and varbinary datatypes

5. Normalisation

One of the goals of normalisation is minimizing redundancy where possible. With this in mind the address table was normalised out. As one person with the same address could make the same booking multiple times this would lead to a lot of repetition in the ‘BookerDetails’ table. Similarly, some attributes were removed if they could be derived with SQL functions. An example of this would be flight duration which could be found by using the TIMEDIFF function with an SQL query [11]. Another example of minimizing

redundancy can be found with the 'SeatingType' table. My initial design allowed me to represent multiple potential seat prices based on the distance the plane traveled and the type of seat but there was a lot of repeating data which violated the first form of normalisation. The initial and final versions of this table can be seen in the image below.

SeatingTypeID	SeatingType	AdditionalBenefits	FlightType	SeatPrice
1	Extra legroom	2 cabin bags, dedicated bag drop, speedy boarding	Short range	15.99
2	Up front	2 cabin bags, dedicated bag drop, speedy boarding	Short range	8.99
4	standard		Short range	2.99
5	Extra legroom	2 cabin bags, dedicated bag drop, speedy boarding	Mid range	24.99
6	Up front	2 cabin bags, dedicated bag drop, speedy boarding	Mid range	15.99
7	Standard		Mid range	5.99
8	Extra legroom	2 cabin bags, dedicated bag drop, speedy boarding	Long range	39.99
9	Up front	2 cabin bags, dedicated bag drop, speedy boarding	Long range	28.99
10	Standard		Long range	8.99

SeatingTypeID	SeatingTypeNameID	AdditionalBenefitsID	FlightRangeID	SeatPrice
1	1	1	1	15.99
2	2	1	1	8.99
4	3	2	1	2.99
5	1	1	2	24.99
6	2	1	2	15.99
7	3	2	2	5.99
8	1	1	3	39.99
9	2	1	3	28.99
10	3	2	3	8.99

Image 12: Initial and final design of 'SeatingType' table

A key concern as previously mentioned was the possibility of data inconsistency so normalisation was employed to limit this as much as possible. An example of this could be an alteration to a route. Flight numbers such as EZY6703 are typically reused for the same journeys again and again but if EasyJet decided to reallocate this flight number this could cause a great deal of confusion with the computer system potentially providing passengers with the wrong flight information. To prevent an issue such as this the route table was normalised out meaning that any updates to a route could be completed while maintaining data integrity.

	RouteID	DepartureAirport	ArrivalAirport
Edit Copy Delete	1	Belfast International	Alicante
Edit Copy Delete	2	Alicante	Belfast International
Edit Copy Delete	3	Belfast International	Krakow
Edit Copy Delete	4	Krakow	Belfast International
Edit Copy Delete	5	Edinburgh	Berlin Brandenburg

Image 13: Some of the routes present in the database

Other concerns relating to data integrity led to tables such as title and age being normalised. This allows staff to query the age of passengers much more easily. If a passenger had entered 'ten' while another had entered '10' this would make it difficult to accurately track this information which is important for both seat allocation and ensuring that all children are seated with a guardian. Normalising out certain tables such as age or sporting equipment not only reflects the websites drop down menu system and preserves data integrity but also assists staff in carrying out essential tasks at check in. For example, a search of passengers on flight EJU2277 with sports equipment [12] reveals that the passenger Natasha McCabe has booked a snowboard and a sporting firearm. Allowing staff to be aware of this in advance of the flight ensures they are ready to check she has completed the necessary firearms declaration form and stored her equipment in accordance with the safety procedures listed on the website.

FlightNumber	FlightDeptDate	FlightDeptTime	PassengerFirstName	PassengerSurname	SportsEquipmentType	Price
EJU2277	2020-12-15	12:00:00	Natasha	McCabe	Snowboard	37.00
EJU2277	2020-12-15	12:00:00	Natasha	McCabe	Sporting firearm	37.00

Image 14: Passengers on Flight EJU2277 carrying sports equipment

Other examples of this include the special assistance table which can be searched [13] to reveal that some passengers such as Conor Loughran and Caitlin O'Hara on flight EZY6703 have a nut allergy. While easyjet states their inability to guarantee a 100% allergy free environment it would be helpful for staff to be aware of passengers with health requirements so they can assist or aid them if possible.

FlightNumber	FlightDeptDate	FlightDeptTime	PassengerFirstName	PassengerSurname	SpecialAssistanceType
EZY6703	2020-12-19	15:15:00	Caitlin	O'Hara	Nut allergy
EZY6703	2020-12-19	15:15:00	Conor	Loughran	Nut allergy

Image 15: Passengers on Flight EZY6703 with special assistance requirements

6. Improvements

There are several areas of this project which I feel could be improved. First, there is the scale of the project. EasyJet offers an enormous number of flights and routes to many bookers and passengers. My database features a very small amount of data comparatively. Additional data could assist in not only better representing a functional database but in highlighting flaws for example the existence of data repetition. However, working with a smaller dataset allowed for the database to be edited and altered without too much difficulty.

A second concern is the database's inability to limit certain actions. A good example of this is with seat selection. While each flight has a certain number of seats and it is possible to query both the maximum capacity as well as the remaining seats available there is no restriction in place to prevent a seat from being assigned to multiple passengers. It is possible that a developer would code the website in a way which prevents this from happening but from my perspective it seems like a failing of the database's functionality.

A third limitation with my database is its inability to showcase how it might change over time as more and more data is added and the process of data warehousing becomes a requirement. Related to this is the difficulty of updating elements, particularly flight prices as these can change often and across such a vast number of flights it would be difficult to ensure the accuracy of the information in my database.

A fourth limitation relates to the search function of the website. With my database it is possible to query flights based on their departure date, but it is not possible to check the available seats on these flights without performing a second SQL query. It is my belief that completing this task with one sql query would be more efficient and would resemble the actual website more closely.

A final limitation found is the database's ability to appropriately assign correct pricing to certain elements. I designed my database so that flights would be classified as either short, mid or long range and baggage and seat prices would reflect this. However, while this does show some attempt to show the difference in these prices it is likely that EasyJet has a more complex method of judging the seat and baggage prices on separate flights.

7.Conclusion

The goal of reverse engineering EasyJet's web database was largely successful. My database allows a booker to check the availability of flights as well as their prices. A booker is also able to select one or more seats as well as a range of baggage options, notify staff of special assistance requirements and purchase additional items. EasyJet staff can update flight information including dates, times, prices, and flight status. They are also able to query key information about passengers so they can ensure the process of check-in occurs without issue for each passenger and they can ideally provide a better experience. Maintaining data integrity across multiple connected tables seems to have been successful although it is difficult to fully measure success in this area when working with a small amount of demo data. A screenshot of the database in designer mode can be seen below showcasing the final design of this project.

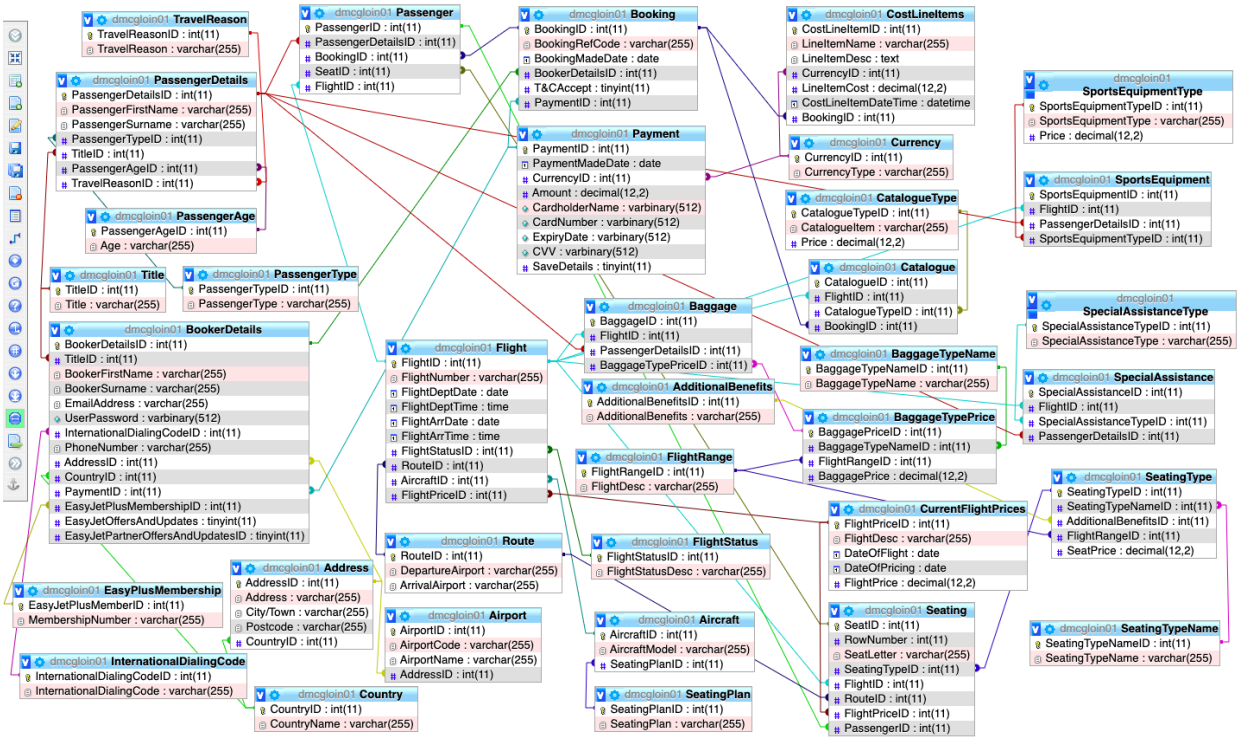
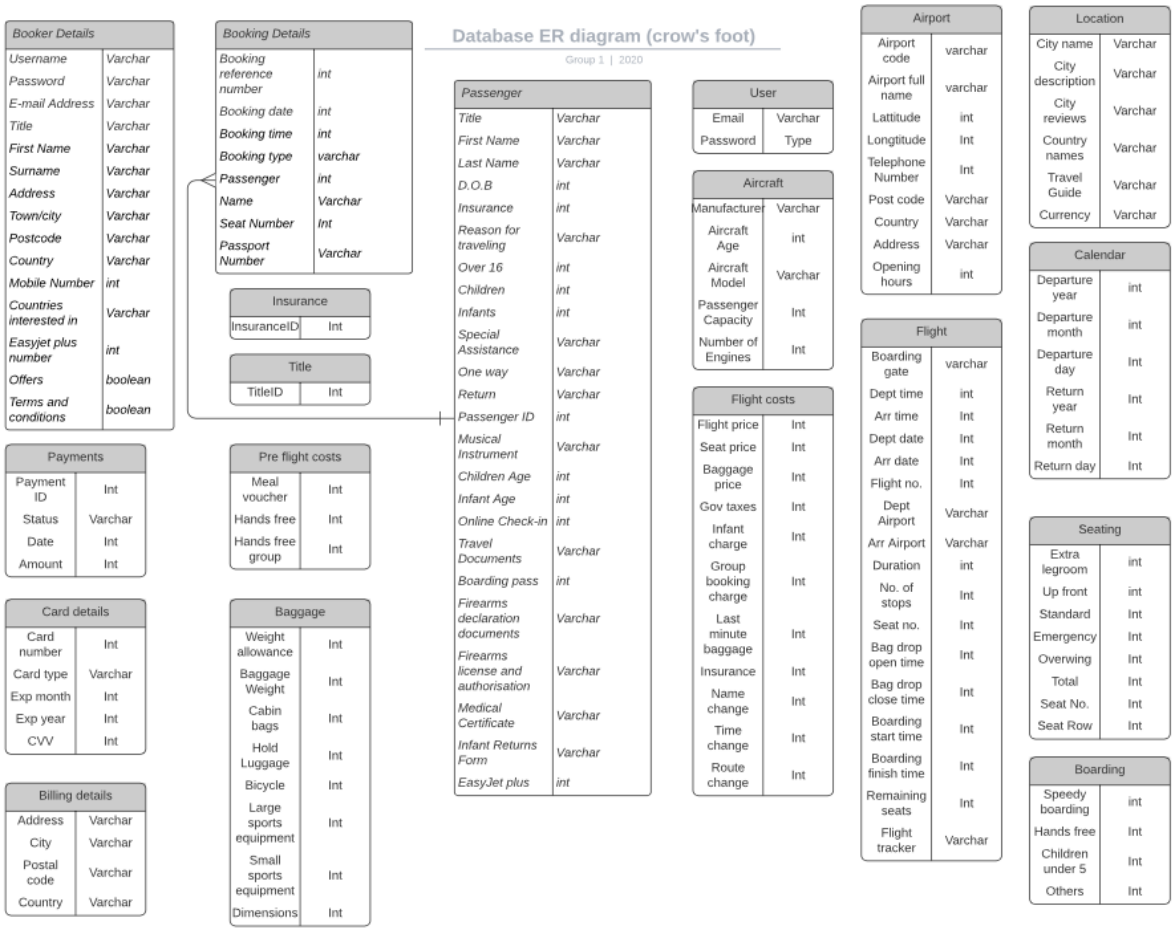


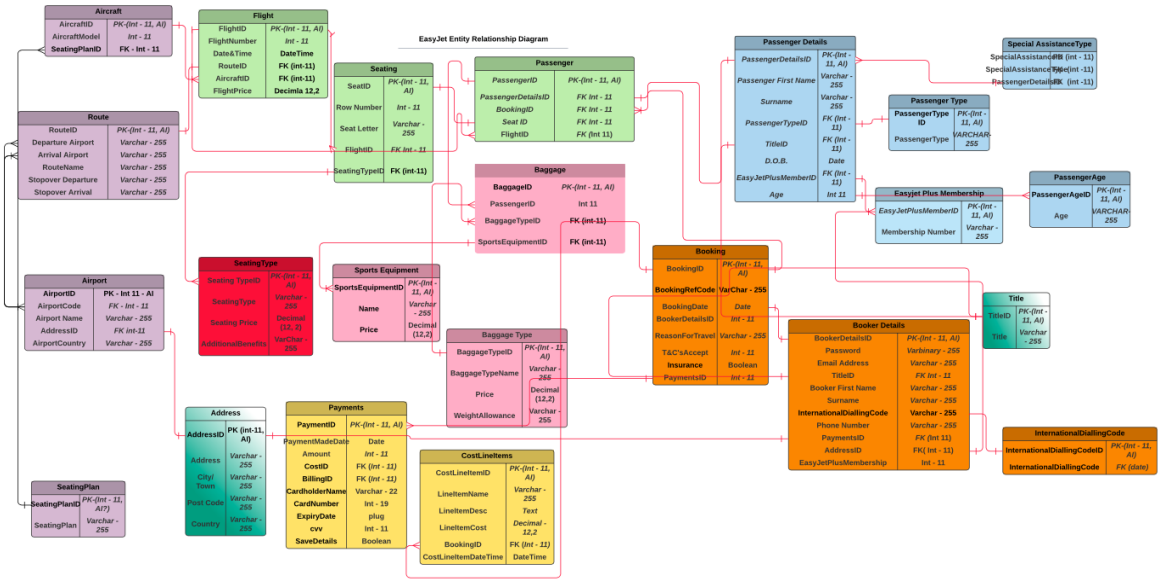
Image 16: Designer view of flight booking system consisting of 36 tables

Appendix

[1] First draft of group 1 ER Diagram



[2] Final draft of group 1 ER Diagram



[3] SQL query for showing passenger details information

SELECT Title, PassengerFirstName, PassengerSurname, Age, PassengerType, TravelReason

FROM PassengerDetails

INNER JOIN PassengerType ON PassengerDetails.PassengerTypeID =

PassengerType.PassengerTypeID

INNER JOIN Title ON PassengerDetails.TitleID =

Title.TitleID

INNER JOIN PassengerAge ON PassengerDetails.PassengerAgeID =

PassengerAge.PassengerAgeID

INNER JOIN TravelReason ON PassengerDetails.TravelReasonID =

TravelReason.TravelReasonID

[4] SQL query for referencing booker details with decrypted password

SELECT `BookerFirstName`, `BookerSurname`, EmailAddress, AES_DECRYPT(`UserPassword`,'mySecretKey') FROM `BookerDetails`

[5] SQL query for searching return flight information based on specified departure and arrival points and dates

SELECT FlightNumber, FlightDeptDate, FlightDeptTime, FlightArrDate, FlightArrTime, DepartureAirport, ArrivalAirport, FlightPrice

FROM Flight

```
INNER JOIN Route ON Flight.RouteID =
Route.RouteID
INNER JOIN CurrentFlightPrices ON Flight.FlightPriceID =
CurrentFlightPrices.FlightPriceID
WHERE FlightDeptDate = '2020-12-19'
AND DepartureAirport = 'Belfast International'
AND ArrivalAirport = 'Alicante'
OR FlightDeptDate = '2020-12-26'
AND DepartureAirport = 'Alicante'
AND ArrivalAirport = 'Belfast International'
```

[6] SQL query to determine if there are enough seats remaining on a specified flight

```
SELECT COUNT(SeatID) - COUNT(PassengerID)
FROM Seating
WHERE FlightID = '1'
```

[7] SQL query to determine seat information including seat availability for a specified flight

```
SELECT RowNumber, SeatLetter, SeatingTypeName, AdditionalBenefits, SeatPrice, FlightDesc
FROM Seating
JOIN SeatingType ON Seating.SeatingTypeID =
SeatingType.SeatingTypeID
JOIN SeatingTypeName ON SeatingType.SeatingTypeNameID =
SeatingTypeName.SeatingTypeNameID
JOIN AdditionalBenefits ON SeatingType.AdditionalBenefitsID =
AdditionalBenefits.AdditionalBenefitsID
JOIN FlightRange ON SeatingType.FlightRangeID =
FlightRange.FlightRangeID
WHERE FlightID = '1'
AND PassengerID IS NULL
```

[8] SQL query to show purchase information based on specified name and booking reference code

```
SELECT Title, BookerFirstName, BookerSurname, LineItemName, LineItemDesc, CurrencyType,
LineItemCost, CostLineItemDateTime, BookingRefCode

FROM CostLineItems

JOIN Booking ON CostLineItems.BookingID =

Booking.BookingID

JOIN BookerDetails ON Booking.BookerDetailsID =

BookerDetails.BookerDetailsID

JOIN Payment ON BookerDetails.PaymentID =

Payment.PaymentID

JOIN Title ON BookerDetails.TitleID =

Title.TitleID

JOIN Currency ON CostLineItems.CurrencyID =

Currency.CurrencyID

WHERE BookerFirstName = 'Neil'

AND BookerSurname = 'Anderson'

AND BookingRefCode = 'EL5C20G'
```

[9] SQL query to update flight price and date of pricing for specified flight from 64.99 to 65.99

```
UPDATE CurrentFlightPrices

SET DateOfPricing = '2020-11-30', FlightPrice = '65.99'

WHERE FlightPriceID = '13'
```

[10] SQL query to show payment information including decrypted card information

```
SELECT PaymentMadeDate, CurrencyType, Amount, AES_DECRYPT(CardholderName, 'mySecretKey'),
AES_DECRYPT(CardNumber, 'mySecretKey'),

AES_DECRYPT(ExpiryDate, 'mySecretKey'), AES_DECRYPT(CVV, 'mySecretKey')

FROM Payment

INNER JOIN Currency ON Payment.CurrencyID =

Currency.CurrencyID
```

[11] SQL query to show fight duration for specified flight

```
SELECT TIMEDIFF (`FlightDeptTime`, `FlightArrTime`)
```

```
FROM Flight

WHERE FlightID = 1
```

[12] SQL query to check specified flight for passengers with sports equipment

```
SELECT FlightNumber, FlightDeptDate, FlightDeptTime, PassengerFirstName, PassengerSurname,
SportsEquipmentType, Price

FROM SportsEquipment

INNER JOIN Flight ON SportsEquipment.FlightID =

Flight.FlightID

INNER JOIN PassengerDetails ON SportsEquipment.PassengerDetailsID =
PassengerDetails.PassengerDetailsID

INNER JOIN SportsEquipmentType ON SportsEquipment.SportsEquipmentTypeID =

SportsEquipmentType.SportsEquipmentTypeID

WHERE FlightNumber = 'EJU2277'
```

[13] SQL query to check specified flight for passengers with special assistance requirements

```
SELECT FlightNumber, FlightDeptDate, FlightDeptTime, PassengerFirstName, PassengerSurname,
SpecialAssistanceType

FROM SpecialAssistance

INNER JOIN Flight ON SpecialAssistance.FlightID =

Flight.FlightID

INNER JOIN PassengerDetails ON SpecialAssistance.PassengerDetailsID =

PassengerDetails.PassengerDetailsID

INNER JOIN SpecialAssistanceType ON SpecialAssistance.SpecialAssistanceTypeID =

SpecialAssistanceType.SpecialAssistanceTypeID

WHERE FlightNumber = 'EZY6703'
```