CSC7052 Database Assignment

Student Name: Neill Calvert

Student Number: 40150199

Degree Pathway: MSc Software Development

Supervisor: Dr Neil Anderson

School of Electronics, Electrical Engineering &
Computer Science

Queen’s University Belfast

A signed and completed cover sheet must accompany the submission of the Individual Software Development dissertation submitted for assessment.

Work submitted without a cover sheet will **NOT** be marked.

Declaration of Academic Integrity

Before signing the declaration below please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.
6. Software and files are submitted on a memory stick.
7. Journal has been submitted.

I declare that I have read both the University and the School of Electronics, Electrical Engineering and Computer Science guidelines on plagiarism - http://www.qub.ac.uk/schools/eeecs/Education/StudentStudyInformation/Plagiarism/ - and that the attached submission is my own original work. No part of it has been submitted for any other assignment and I have acknowledged in my notes and bibliography all written and electronic sources used.

I am aware of the disciplinary consequences of failing to abide and follow the School and Queen’s University Regulations on Plagiarism.

Name: (BLOCK CAPITALS)	**MATTHEW NEILL ARMOUR CALVERT**
Student Number:	**40150199**

Student’s signature Matthew Neill Armour Calvert *Date of submission* 23/11/2018

Table of Contents

1.0 Introduction

1.1 Aims and Scope

The project entailed reverse engineering a database from the EasyJet website using MySQL. The main requirement for database functionality was to demonstrate user ability to book one or more passengers on a flight from one airport to another while taking aspects such as luggage and special assistance into account.

Due to the aim of the project the site was investigated prior to the production of the entity-relationship diagram and relevant sections were selected. Sections of the website including hotel bookings, car hire and others that required third party support were deemed out of scope due to time constraints. In keeping with the aims of the project, most effort was focused on the facility to create a new account for the EasyJet website and the flight booking process.

1.2 Database Concept and Proposal

As one of the largest low-budget airlines in the U.K. with over 200 aircraft, EasyJet offers a variety of routes across Europe with a range of facilities and addons that customers may utilise. To be successful, the database must be able to maintain this information accurately, while the data should be easily usable by both customers and staff.

Due to the nature of airline regulations and customer expectations it is important the database does not contain inconsistent or erroneous data. This means the database must be defined by the principles of normalisation to minimise redundancy and the potential for user errors.

The design focuses on the tables required to create a user account, create a passenger(s), book a seat on a flight on a specific route and book luggage on that flight. Importantly, the design assumes that passengers exist solely on one flight and when a passenger is booked onto a return or connecting flight, they are a new passenger. The design also assumes that EasyJet have no partner airlines and that car hire and EasyJet hotel bookings are handled by a third party.

This report includes a designer view of the database (accepted as the entity-relationship diagram) to show an overview of the database design, while information on important primary and foreign keys will also be provided. Important decisions made when designing the database are also discussed regarding normalisation, business intelligence and improvements that could be made. Lastly, areas where this individual database differed from other group members will be highlighted and discussed.

To demonstrate how meaningful data can be retrieved from the database sample, queries will also be provided in the appendix and throughout the report including their expected result using sample data.

2.0 An Overview of the Database Design and Entity-Relationship Diagram

2.1 Entity-Relationship Diagram

Before the construction of the database using MySQL, a suitable entity-relationship diagram was created. This allowed each business concept to be termed an entity and placed inside a box with relevant attributes stemming from these boxes. This process began during the group work element of the project in which efforts were concentrated on the most important attributes of each entity.

These newly created entities were then linked to each other via the relationships between them such as one-to-one or many-to-one.

Upon creation of the ER-diagram a more informative design diagram was created where entities and attributes were placed inside tables linked using crows' feet notation to other tables. Figure 1 is a final design diagram with ER-principles showing the relationship-types between tables.

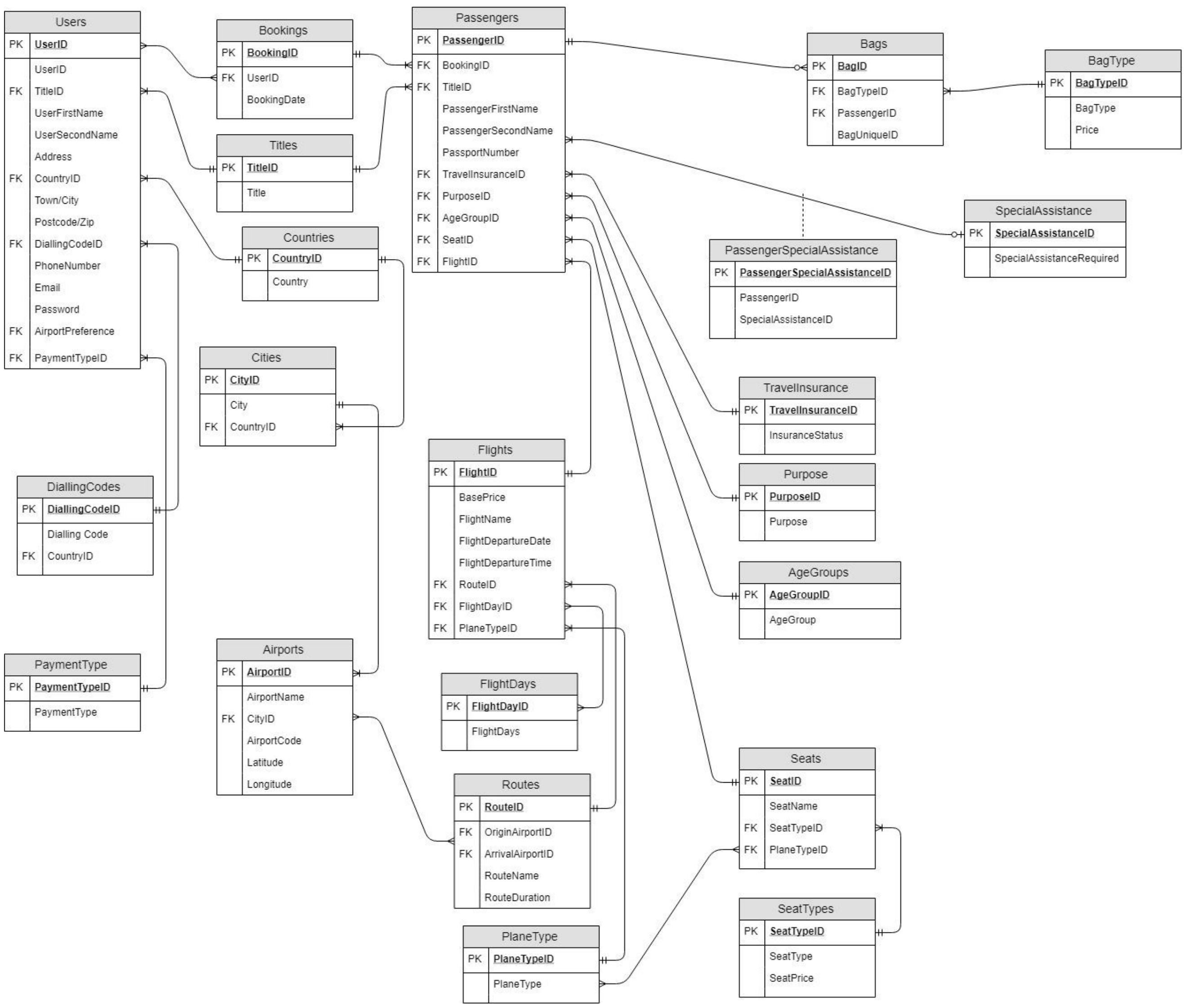


Figure 1. Figure showing the design of the database and the relationships between tables in crows' feet notation where Primary keys are abbreviated to 'PK' and Foreign Keys are abbreviated to 'FK'.

The process began with the user creation component of the website including the important information customers must provide when using an account such as their name, address, telephone number and email. This was followed by the creation of routes with origin and arrival airports linked to flights that users may book passengers onto.

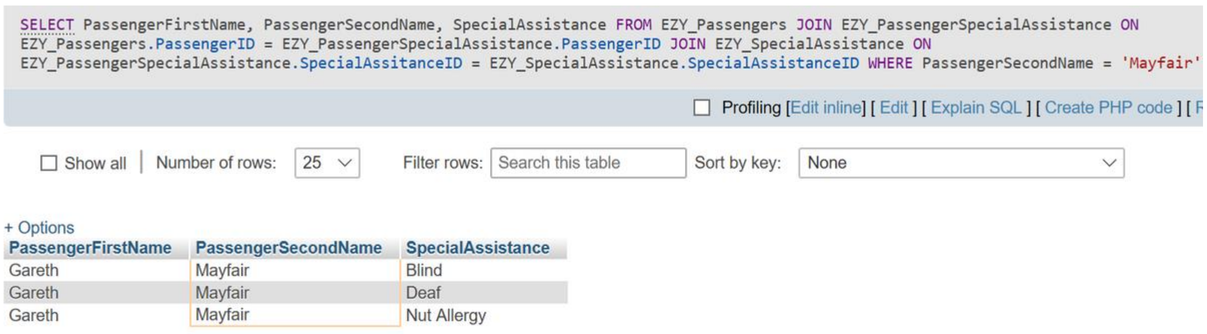
At this point the diagram became more complicated and options the user may select from, were placed into more tables. This included information such as the type of seat passengers may book including extra leg room or upfront. They also included demographic information such as their age group and country of residence coupled with further data such as insurance status. In general, the passengers table was linked to these additional tables via a many-to-one relationship as shown in Figure 1. Importantly however, an optional many-to-many relationship was realised when theorising the special assistance table, in which many passengers can have no, one or many forms of special assistance as shown in Figure 1.

2.2 Linking Tables and Deviations from the Group Model

When constructing the individual ER-diagram there was a key disagreement in the handling of special assistance compared to the group model.
The original group diagram included the important optional many-to-many relationship in which many passengers could be linked to many or no types of special assistance.

While many-to-many relationships can exist in databases it is often regarded as poor database design. It is not enough to simply understand that a many-to-many relationship exists, but the database must be able to store and express the many combinations of data that EasyJet provides (Bagui and Earp, 2011).

To ensure the data could be reflected, a linking table was inserted noted by a dash line to the main cardinality constraint between passengers and special assistance as shown in Figure 1. This table (PassengerSpecialAssistance) contained a SpecialAssistanceID column linked via foreign key constraint to the primary key of the SpecialAssistance table. PassengerID in the PassengerSpecialAssistance table was also linked via foreign key constraint to the primary key of the Passengers table as shown in Figure 1. This allowed for one passenger to select more than one disability and that information to be queried effectively as shown in Figure 2 below.



```
SELECT PassengerFirstName, PassengerSecondName, SpecialAssistance FROM EZY_Passengers JOIN EZY_PassengerSpecialAssistance ON
EZY_Passengers.PassengerID = EZY_PassengerSpecialAssistance.PassengerID JOIN EZY_SpecialAssistance ON
EZY_PassengerSpecialAssistance.SpecialAssitanceID = EZY_SpecialAssistance.SpecialAssistanceID WHERE PassengerSecondName = 'Mayfair'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [F
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

PassengerFirstName	PassengerSecondName	SpecialAssistance
Gareth	Mayfair	Blind
Gareth	Mayfair	Deaf
Gareth	Mayfair	Nut Allergy

Figure 2. MySQL query showing a passenger with more than one special assistance requirement

These key differences from the group ER-diagram provide a true reflection of the services EasyJet provide and mitigate the circumstances of the many-to-many relationship.

3.0 Information on the Primary Keys, Foreign Key Constraints and Data Types Used

3.1 Primary Keys

In relational databases, the Primary Key is an important column used to uniquely identify record sets. To fulfil its purpose the Primary Key must be unique to each row and may not contain null values (Harrington, 2016). In order to satisfy this requirement, all tables within the database have a primary key that has been set to the type, 'Int' with a length of 10 and can be identified as the 'table name' with the postfix: 'ID', such as 'UserID'. These values have also been set to automatically increment upon entry of a new row of data to ensure that each new row is assigned a unique primary key every time. It was important that these primary keys have closely related foreign key constraints in order to link tables throughout the database and retrieve precise information (Page, 1990).

3.2 Foreign Key Constraints

The Foreign Key Constraints are shown in Figure 1 with the abbreviation 'FK'. These foreign key constraints are important to the effectiveness of the database and ensure the data of each table is not isolated (Nixon, 2018). In this database almost, all foreign keys have the 'restrict' action enabled meaning the data in the parent table referenced by this foreign key in child tables cannot be deleted or updated.

There was an exception to this in the 'PassengerBookingID' foreign key constraint in the passengers table that was set to cascade on deletion. This means upon deletion of a booking the passengers connected to this booking would also be deleted from the database as expected when cancelling an EasyJet booking.

An important example is the Titles table linked via foreign key constraint to the Users and Passengers tables. Columns that rely on foreign key constraints like titleID allow users to select their title from a suitable list to minimise errors that would otherwise arise when typing in this data manually. These constraints are shown in Figure 3 below and support the principles of normalisation.

3.3 Important Datatypes Used

In most cases where an integer value was required the data length was set to 10 and where a variable character (VarChar) field was required, the data length was set to 255. These standardised datatypes maintained consistency throughout the database and are shown in Figure 3 below.

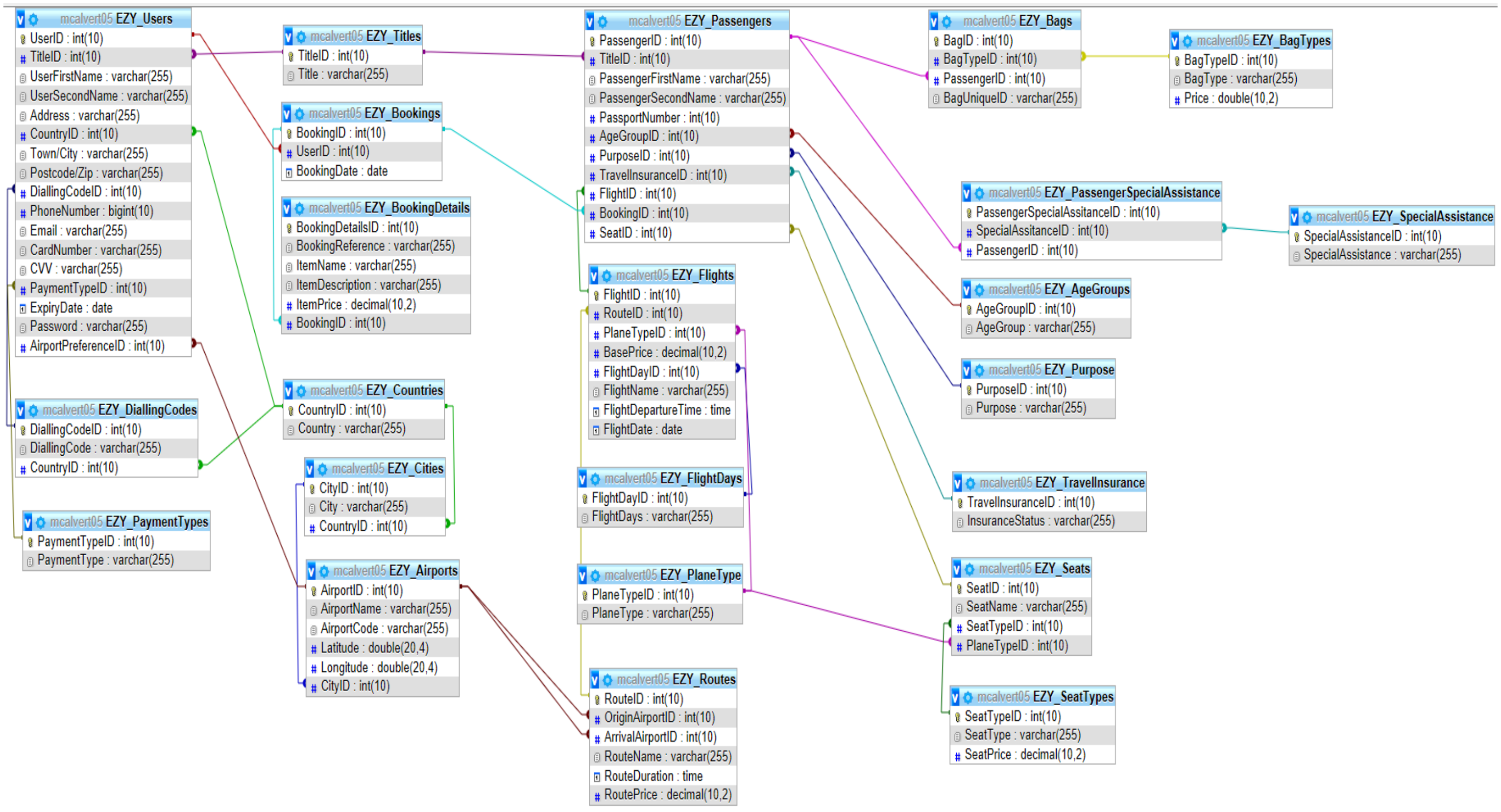


Figure 3. Figure showing the designer view of the completed database in MySQL including the data types and lengths of each variable

Numerous tables including Routes, BagTypes, SeatTypes and BookingDetails shown in Figure 3 contain a price column that can be used to calculate the full booking cost for the user. To ensure the prices were presented accurately it was important these datatypes were set to 2 decimal places to avoid incorrect pricing.

Secondly, the date datatype was used throughout the database including the ExpiryDate in the Users table, the FlightDate in the Flights table and the BookingDate in the Bookings table. These datatypes were kept consistent throughout each table in the standard dd:mm:yy form to avoid confusion and errors. The same principle was implemented when using the time datatype for FlightDepartureTime and FlightDuration in which the standard hh:mm:ss form was used.

Date and time formats were kept consistent to avoid confusion to users and to ensure the data could be queried easily. For example, maintaining departure time and duration in the same format allows for the calculation of an estimated arrival time using a sum query of the two columns.

Lastly, it is important to highlight the use of the varchar data type of length 255 for CardNumber and CVV columns in the Users table instead of the expected int types. For security reasons, it was key that card numbers and card security codes were encrypted to uphold consumer confidence. To encrypt using advanced encryption standard the data type must be set to a varchar with a minimum 128-bit key length.

The frontend development team would prevent users entering card numbers and CVV codes with characters or the incorrect number of digits to mitigate the circumstances of these datatypes.

4.0 Discussion and Justification of Important Design Decisions

4.1 Encryption of Sensitive Information

The database allows for a password facility in which a user may have a password of their choosing in order to access their account. It is important that information like this is encrypted to block other consumers or potential attackers from accessing password data.

Secondly, user accounts include a single set of card details for one of 10 card types EasyJet allows. For security reasons it is important that this type of data including the 16-digit card number and 3-digit card security code (CVV) are encrypted to minimise risk to users.

MySQL has three main built-in encryption algorithms with varying levels of complexity that may be used including 'Data Encryption Standard' (DES), 'MD5 message-digest algorithm' and 'Advanced Encryption Standard' (AES).

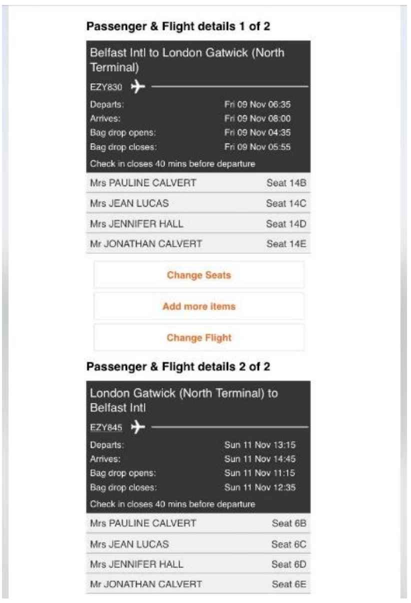
DES is an early encryption type designed in the 1970s and relies on a symmetric key algorithm with a 56-bit key size. While it was possible to use this encryption method in the database it is a very insecure option, mainly due to the small key size that would allow for passwords and card details to be easily decrypted (Foundation, 1999).

MD5 is a more complicated encryption algorithm available for use in MySQL, however the algorithm has some major vulnerabilities. A major requirement of hash algorithms is that two separate inputs should not generate the same digest. MD5 fails in this regard and was therefore found to be more vulnerable to attacks (Chen and Jin, 2009). Many resources are also available for the decryption of MD5 online that would be severely damaging to consumer perception of EasyJet should the database be compromised.

Due to these risks it was decided that AES encryption was sufficient due to its more secure encryption method and wide use across commercial and governmental organisations. While attacks are still possible, the likelihood of data decryption is greatly reduced compared to MD5 and DES encryption methods.

4.2 Booking a Return or Connecting Flight

EasyJet provide a connecting and return flights service to users in which several flights may be booked for the same passenger in a single order. Analysis of an EasyJet receipt available in Figure 4 below revealed a possible solution to this issue in which passenger details for one flight are distinct from another, despite passenger identity remaining the same.


Passenger & Flight details 1 of 2

Belfast Intl to London Gatwick (North Terminal)

EZY830

Departs:	Fri 09 Nov 06:35
Arrives:	Fri 09 Nov 08:00
Bag drop opens:	Fri 09 Nov 04:35
Bag drop closes:	Fri 09 Nov 05:55

Check in closes 40 mins before departure

Mrs PAULINE CALVERT	Seat 14B
Mrs JEAN LUCAS	Seat 14C
Mrs JENNIFER HALL	Seat 14D
Mr JONATHAN CALVERT	Seat 14E

Passenger & Flight details 2 of 2

London Gatwick (North Terminal) to Belfast Intl

EZY845

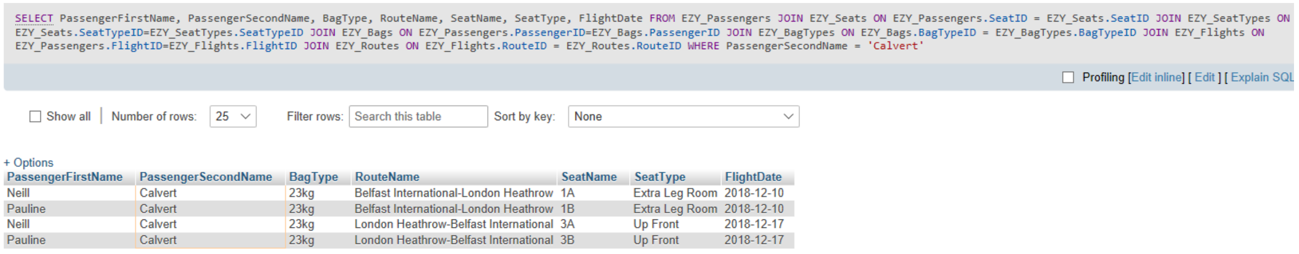
Departs:	Sun 11 Nov 13:15
Arrives:	Sun 11 Nov 14:45
Bag drop opens:	Sun 11 Nov 11:15
Bag drop closes:	Sun 11 Nov 12:35

Check in closes 40 mins before departure

Mrs PAULINE CALVERT	Seat 6B
Mrs JEAN LUCAS	Seat 6C
Mrs JENNIFER HALL	Seat 6D
Mr JONATHAN CALVERT	Seat 6E

Figure 4. Figure showing the EasyJet receipt available to users showing passengers with the same name booked on two separate flights (Easyjet.com, 2018)

To book a return flight on this database, passenger names are re-entered into the passenger table as new passengers. This means they may select seats and bags for a different flight while maintaining the same bookingID. This also allows passengers on return flights to return with more or less bags and in different seat types as EasyJet currently allows. A query is shown below demonstrating these options in Figure 5 in which passengers with the last name, Calvert return from London Heathrow after one week in different seat types.


```
SELECT PassengerFirstName, PassengerSecondName, BagType, RouteName, SeatName, SeatType, FlightDate FROM EZY_Passengers JOIN EZY_Seats ON EZY_Passengers.SeatID = EZY_Seats.SeatID JOIN EZY_SeatTypes ON
EZY_Seats.SeatTypeID=EZY_SeatTypes.SeatTypeID JOIN EZY_Bags ON EZY_Passengers.PassengerID=EZY_Bags.PassengerID JOIN EZY_BagTypes ON EZY_Bags.BagTypeID = EZY_BagTypes.BagTypeID JOIN EZY_Flights ON
EZY_Passengers.FlightID=EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID WHERE PassengerSecondName = 'Calvert'
```

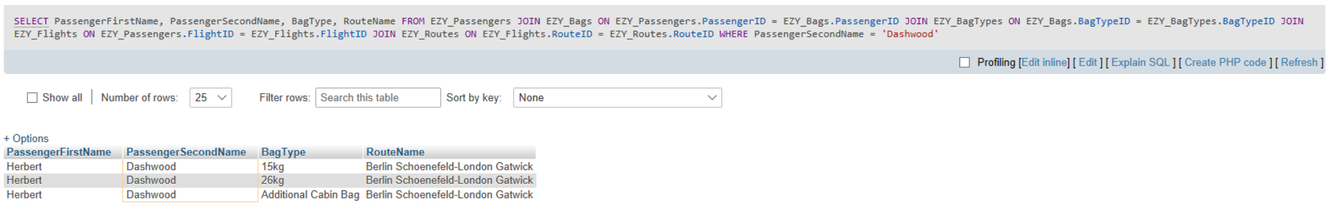
Profiling [Edit inline] [Edit] [Explain SQL
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

PassengerFirstName	PassengerSecondName	BagType	RouteName	SeatName	SeatType	FlightDate
Neill	Calvert	23kg	Belfast International-London Heathrow	1A	Extra Leg Room	2018-12-10
Pauline	Calvert	23kg	Belfast International-London Heathrow	1B	Extra Leg Room	2018-12-10
Neill	Calvert	23kg	London Heathrow-Belfast International	3A	Up Front	2018-12-17
Pauline	Calvert	23kg	London Heathrow-Belfast International	3B	Up Front	2018-12-17

Figure 5. Figure showing a select query in MySQL to present passenger names, route, seats and date of flight

4.3 Ability of One Passenger to Book Numerous Bags

EasyJet allows one carryon bag per passenger included in the flight fee and additional carryon or hold luggage at an extra cost. This was reflected in the database in which the passenger may select a bag type and add it to their passengerID. Unlike the seatID column in the Passengers table, the addition of a PassengerBag table linked by foreign key constraints to the PassengerID and BagTypeID of a separate bag types table allowed passengers to select many bags. This ability for one passenger to book numerous bags on one flight is demonstrated below in Figure 6.



```
SELECT PassengerFirstName, PassengerSecondName, BagType, RouteName FROM EZY_Passengers JOIN EZY_Bags ON EZY_Passengers.PassengerID = EZY_Bags.PassengerID JOIN EZY_BagTypes ON EZY_Bags.BagTypeID = EZY_BagTypes.BagTypeID JOIN
EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID WHERE PassengerSecondName = 'Dashwood'
```

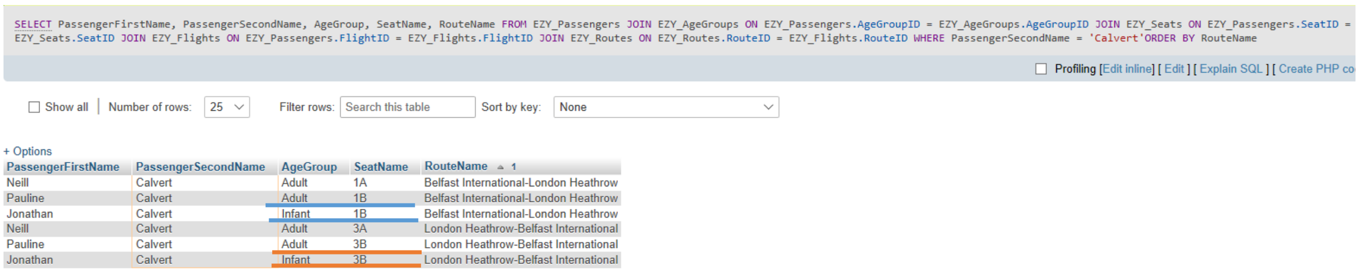
PassengerFirstName	PassengerSecondName	BagType	RouteName
Herbert	Dashwood	15kg	Berlin Schoenefeld-London Gatwick
Herbert	Dashwood	26kg	Berlin Schoenefeld-London Gatwick
Herbert	Dashwood	Additional Cabin Bag	Berlin Schoenefeld-London Gatwick

Figure 6. MySQL query demonstrating ability of a passenger to have numerous bags on a single flight

4.4 Infants Sharing Seats with an Adult

EasyJet does not offer seats to infants and instead allows parents to pay a fee for travel with the option for infants to sit in a cot or on the passenger's lap. To ensure this service was reflected in the database and infants can be booked on a flight, an age group table was created. This table included the options 'Adult', 'Child' and 'Infant' available for users to select. Importantly, the 'SeatID' on the Passengers table linked to the Seats table via foreign key allowed infants to be assigned to the same seat as an adult. The frontend development team could prevent other passengers who are not infants from selecting the same seat, possibly using an if statement.

The result of a query showing an infant sharing a seat with an adult is presented below in Figure 7.



```
SELECT PassengerFirstName, PassengerSecondName, AgeGroup, SeatName, RouteName FROM EZY_Passengers JOIN EZY_AgeGroups ON EZY_Passengers.AgeGroupID = EZY_AgeGroups.AgeGroupID JOIN EZY_Seats ON EZY_Passengers.SeatID =
EZY_Seats.SeatID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Routes.RouteID = EZY_Flights.RouteID WHERE PassengerSecondName = 'Calvert'ORDER BY RouteName
```

PassengerFirstName	PassengerSecondName	AgeGroup	SeatName	RouteName ▲ 1
Neill	Calvert	Adult	1A	Belfast International-London Heathrow
Pauline	Calvert	Adult	1B	Belfast International-London Heathrow
Jonathan	Calvert	Infant	1B	Belfast International-London Heathrow
Neill	Calvert	Adult	3A	London Heathrow-Belfast International
Pauline	Calvert	Adult	3B	London Heathrow-Belfast International
Jonathan	Calvert	Infant	3B	London Heathrow-Belfast International

Figure 7. MySQL query showing the infant 'Jonathan Calvert' assigned to the same seat as the adult 'Pauline Calvert'

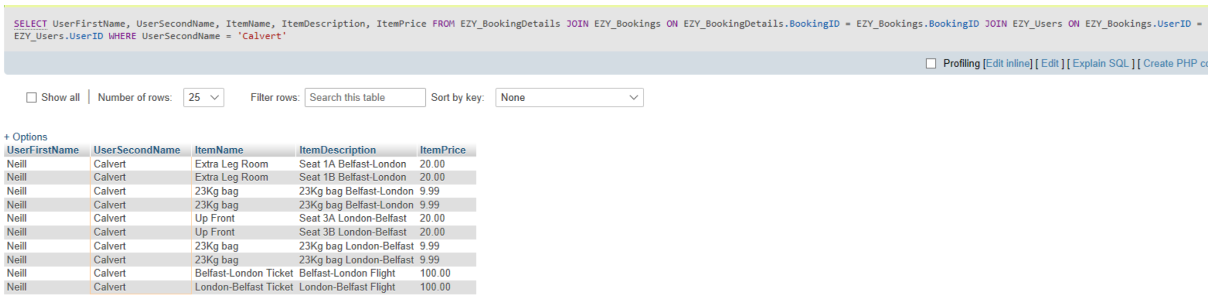
Lastly, allowing infants to share seats in the Passengers table means the distinct seatIDs must be counted instead of the number of passengers on the plane to present the correct number of occupied seats as the EasyJet website allows.

4.5 Booking Details Table

A key difference in this individual database compared to the early group model is the addition of a booking details table. This table is responsible for holding the name of each item, an item description and pricing information of these items. This table serves as an invoice for the customer to see a break-down of their booking. While some members of the group linked the prices to a foreign key this led to a major issue in which already existing bookings would be updated upon item price changes. This would mean prices could go up or down and impact

the booking already made by a user. To avoid this situation data is entered manually into the table, however this is still not ideal and requires improvements at the frontend.
Despite these issues the table has potential for a business intelligence strategy in showing the most popular products users buy and products that undersell.

To improve upon this table the frontend developers should automate a sum of the item prices for each user to show the total price the customer must pay. This has been represented by two simple queries in Figure 8 that shows an invoice and Figure 9, which shows a sum of that invoice

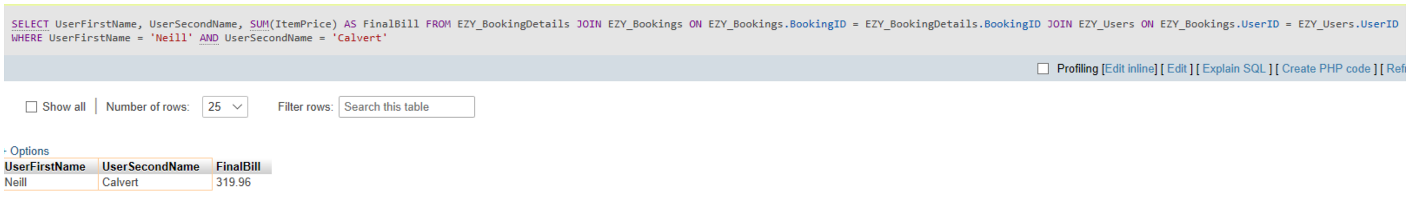


```sql
SELECT UserFirstName, UserSecondName, ItemName, ItemDescription, ItemPrice FROM EZY_BookingDetails JOIN EZY_Bookings ON EZY_BookingDetails.BookingID = EZY_Bookings.BookingID JOIN EZY_Users ON EZY_Bookings.UserID =
EZY_Users.UserID WHERE UserSecondName = 'Calvert'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP c
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

UserFirstName	UserSecondName	ItemName	ItemDescription	ItemPrice
Neill	Calvert	Extra Leg Room	Seat 1A Belfast-London	20.00
Neill	Calvert	Extra Leg Room	Seat 1B Belfast-London	20.00
Neill	Calvert	23Kg bag	23Kg bag Belfast-London	9.99
Neill	Calvert	23Kg bag	23Kg bag Belfast-London	9.99
Neill	Calvert	Up Front	Seat 3A London-Belfast	20.00
Neill	Calvert	Up Front	Seat 3B London-Belfast	20.00
Neill	Calvert	23Kg bag	23Kg bag London-Belfast	9.99
Neill	Calvert	23Kg bag	23Kg bag London-Belfast	9.99
Neill	Calvert	Belfast-London Ticket	Belfast-London Flight	100.00
Neill	Calvert	London-Belfast Ticket	London-Belfast Flight	100.00

Figure 8. MySQL query showing the invoice of the user, 'Neill Calvert'



```sql
SELECT UserFirstName, UserSecondName, SUM(ItemPrice) AS FinalBill FROM EZY_BookingDetails JOIN EZY_Bookings ON EZY_Bookings.BookingID = EZY_BookingDetails.BookingID JOIN EZY_Users ON EZY_Bookings.UserID = EZY_Users.UserID
WHERE UserFirstName = 'Neill' AND UserSecondName = 'Calvert'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Ref
Show all
Number of rows: 25
Filter rows: Search this table
- Options

UserFirstName	UserSecondName	FinalBill
Neill	Calvert	319.96

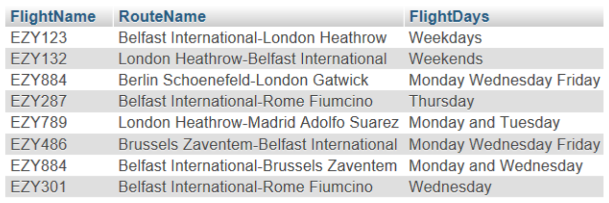
Figure 9. MySQL query calculating a total of all item prices of a specific user and presenting them as a final bill

4.6 Addition of Travel Insurance and Purpose Tables

Unlike the original group model, the individual database included a travel insurance table and passenger purpose table. The passenger table contained InsuranceID and PurposeID under foreign key constraints to their respective tables. This means that a passenger must select that they are either insured or uninsured and whether they are travelling for business or leisure. While this information may seem trivial it has potential to create business intelligence reports that could entail how many passengers are insured on EasyJet flights and identify possible marketing strategies. This is also the case for business and leisure passengers in which a report could be created to identify routes more popular for business or leisure.

4.7 Addition of a Flight Days Table

A final key difference between the group ER-Diagram and the individual database was the addition of a flight days table. This table acted as a schedule for flights containing a list of individual days of the week and combinations of these days such as Monday, Tuesday and Wednesday. A FlightDaysID was used as the primary key for this table and as a foreign key constraint to the selectable FlightDaysID in the flights table. This means this table can used to show the days each flight will operate and allows this information to be viewed by potential customers to help plan their holidays. This also means individual flights can be queried to show the days in which they operate as shown in Figure 10 below reflecting the EasyJet website.



FlightName	RouteName	FlightDays
EZY123	Belfast International-London Heathrow	Weekdays
EZY132	London Heathrow-Belfast International	Weekends
EZY884	Berlin Schoenefeld-London Gatwick	Monday Wednesday Friday
EZY287	Belfast International-Rome Fiumcino	Thursday
EZY789	London Heathrow-Madrid Adolfo Suarez	Monday and Tuesday
EZY486	Brussels Zaventem-Belfast International	Monday Wednesday Friday
EZY884	Belfast International-Brussels Zaventem	Monday and Wednesday
EZY301	Belfast International-Rome Fiumcino	Wednesday

Figure 10. MySQL query showing the flight name, route and days of operation

5.0 Discussion and Justification of Important Normalisation Decisions

5.1 Importance of Normalisation in Relational Databases

The main aim of normalisation in relational databases is to minimise redundancy and prevent needless repetition of data that could cause potential errors. This aim ensured that tables that require repetitive input or tables that reflect the same data were eliminated, maintaining atomicity.

During construction of the database first, second and third normal forms were adhered to. Tables avoided repetition according to first normal form and it was ensured that when adding new data to the database the schema remained unchanged. No tables contain partial dependencies and there are no dependencies between non-key fields in accordance with the second and third normal forms (Shoval and Even-Chaime, 1987).

5.2 Normalisation of the Users Table

The Users table contained a large amount of data specified by the user when creating an account on the EasyJet website. To create an account the user requires unique data such as their email address, first name, second name, home address, postcode/zip and their mobile phone number.

Additional information required to create an account threatened to violate the normal forms if entered into the database unchecked. These fields included title, telephone dialling code, country, payment card type and the airport the user is most interested in. Relying on the user to enter this data into the table was likely to lead to inconsistencies or errors.

Secondly, without normalising these fields the table was open to repetitive inputs in violation of the first normal form.

To minimise the threat of these fields to the normal forms, new tables were created to store this data and linked to the users table via foreign key constraints. The column headings and foreign key constraints critical for normalisation are shown below in Table 1.

Table 1. Table showing the column heading/field of the user table and any key constraints of the users table that are important for normalisation.

Column Heading/Field of the User table	Key Constraints
UserID	Primary Key that is unique to each user
TitleID	Foreign Key Constraint based on TitleID in the 'Titles' table

CountryID	Foreign Key Constraint based on CountryID in the 'Countries' table
DiallingCodeID	Foreign Key Constraint based on DiallingCodeID in the 'DiallingCodes' table
PaymentTypeID	Foreign Key Constraint based on PaymentTypeID in the 'PaymentType' table
AirportPreferenceID	Foreign Key Constraint based on AirportID in the 'Airports' table

As shown in Table 1 columns that have the potential to contain repetitive and erroneous data are linked by foreign key constraints to tables from which the user may select an option. For example, the user may only select Mr, Mrs or Miss from the Titles table removing the possibility of entering inconsistent data such as 'mister'.

5.3 Normalisation of the Passengers Table

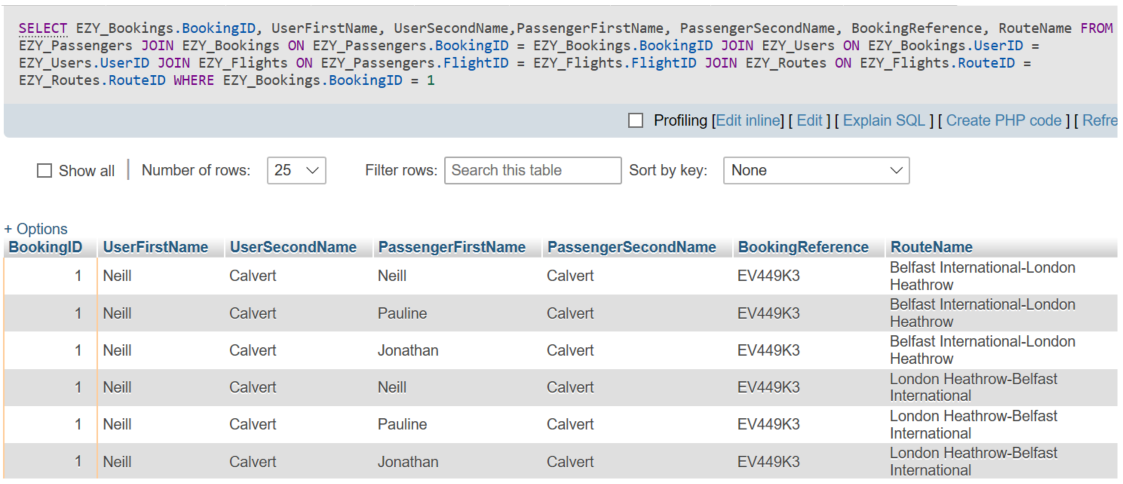
A second important table in terms of normalisation decisions was the passengers table. Table 2 below shows the important columns that have been normalised into new tables and the foreign key constraints linking these tables together.

Table 2. Table showing the column heading/field of the passengers table and any key constraints of the passengers table that are important for normalisation.

Column Heading/Field of the Passenger table	Key Constraints
Passenger ID	Primary Key that is unique to each passenger
TitleID	Foreign Key Constraint based on TitleID in the 'Titles' table
AgeGroupID	Foreign Key Constraint based on AgeGroupID in the 'AgeGroups' table
PurposeID	Foreign Key Constraint based on PurposeID in the 'Purpose' table
TravelInsuranceID	Foreign Key Constraint based on TravelInsuranceID in the 'TravelInsurance' table
BookingID	Foreign Key Constraint based on BookingID in the 'Bookings' table

As with the users table, passenger titles have been normalised, preventing input of inconsistent spelling or titles that do not specify a gender, such as Doctor. Likewise, users may not input their own purpose for travel that could be difficult to analyse. Instead passengers' purpose has been normalised into a usable form to a separate table in which users may select business or leisure. The same process was carried out for travel insurance, normalising the column into another table allowing users to select insured or uninsured. Following the principles of normalisation these decisions help minimise redundant or repetitive input while ensuring each new data entry is atomic.

Lastly, preliminary database design used the users second name as a bookingID allowing numerous passengers on a plane to be linked to one user via this last name. This proved inefficient and threatened normalisation of the database, with repetitive user input of their last name when adding new passengers to their account. Instead, a booking table was created as shown in Figure 3 in which a user may create a booking with a unique bookingID linked to several passengers. The use of this bookingID to link a single user to multiple passengers on a booking is demonstrated in Figure 11 below.



```
SELECT EZY_Bookings.BookingID, UserFirstName, UserSecondName,PassengerFirstName, PassengerSecondName, BookingReference, RouteName FROM
EZY_Passengers JOIN EZY_Bookings ON EZY_Passengers.BookingID = EZY_Bookings.BookingID JOIN EZY_Users ON EZY_Bookings.UserID =
EZY_Users.UserID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID =
EZY_Routes.RouteID WHERE EZY_Bookings.BookingID = 1
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refre
Show all | Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

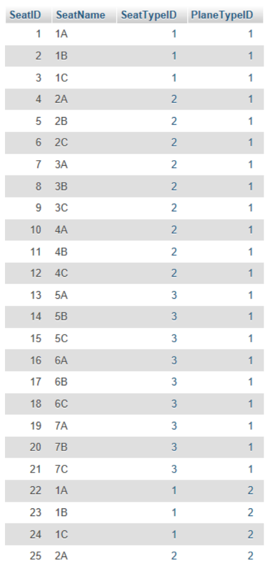
BookingID	UserFirstName	UserSecondName	PassengerFirstName	PassengerSecondName	BookingReference	RouteName
1	Neill	Calvert	Neill	Calvert	EV449K3	Belfast International-London Heathrow
1	Neill	Calvert	Pauline	Calvert	EV449K3	Belfast International-London Heathrow
1	Neill	Calvert	Jonathan	Calvert	EV449K3	Belfast International-London Heathrow
1	Neill	Calvert	Neill	Calvert	EV449K3	London Heathrow-Belfast International
1	Neill	Calvert	Pauline	Calvert	EV449K3	London Heathrow-Belfast International
1	Neill	Calvert	Jonathan	Calvert	EV449K3	London Heathrow-Belfast International

Figure 11. MySQL query demonstrating a single user with multiple passengers linked by a single booking

5.4 Normalisation of the Seats Table

Initially the seats table was complex, containing a rowID column linked to a separate primary key in a rows table and letterID column linked to a primary key in a letters table. This proved inefficient, creating a complex relationship between numerous tables and would be complicated for the customer to use.

Instead, the table now lists the seat names EasyJet offers, coupled with a SeatTypeID and a PlaneTypeID linked to a seat type table and plane type table respectively via foreign keys. The contents of this table are shown in Figure 12 below.



SeatID	SeatName	SeatTypeID	PlaneTypeID
1	1A	1	1
2	1B	1	1
3	1C	1	1
4	2A	2	1
5	2B	2	1
6	2C	2	1
7	3A	2	1
8	3B	2	1
9	3C	2	1
10	4A	2	1
11	4B	2	1
12	4C	2	1
13	5A	3	1
14	5B	3	1
15	5C	3	1
16	6A	3	1
17	6B	3	1
18	6C	3	1
19	7A	3	1
20	7B	3	1
21	7C	3	1
22	1A	1	2
23	1B	1	2
24	1C	1	2
25	2A	2	2

Figure 12. Figure showing a screenshot of the Seats table

The table allows users to select exactly which seat they wish to book depending on the type of plane their flight uses such as A319 or A320. This ensures that each row is atomic in the table based on the seatID and PlanetypeID despite the seatnames 1A, 1B, 1C and 2A being repeated.

<u>6.0 Discussion and Justification of some Improvements that could be made</u>

6.1 Booking Details Table

The Booking Details table is simplified and requires data to be entered manually including important information like price. Should this be entered wrongly into the system this could lead to significant long-term problems and loss of consumer confidence. Additionally, the Booking Details table does not account for information on currency used or the tax amount in different areas users may be purchasing flights from. These shortfalls have great potential risk to EasyJet should they implement the database and could lead to incorrect payments being made.

To improve upon this table, it is recommended frontend developers implement invoicing software or use a third-party option. An example of this third-party software is 'WorldPay' used by Queen's University Belfast to process transactions for student fees. The service is used by over 300,000 businesses in the U.K. providing a final bill to the customer in a currency of their choosing with relevant tax information accounted for (Business.worldpay.com, 2018).

6.2 Expected Arrival Time

While the Flights table contains a departure time and the Routes table contains a flight duration, an expected arrival time is missing from the database. When booking a flight, EasyJet allows customers to view the departure and expected arrival times in the correct time zones.

Flight duration may be added to the flight departure time to query the expected arrival time in this database, however this should be readily available to the customer before booking as a separate column. This issue could be solved by frontend development in which flight duration is updated live and is added to the flight departure time via a programme in order to display the estimated arrival time. Finally, this arrival time may be converted to a local time of arrival considering the various time zones EasyJet operates in.

6.3 EasyJet Membership and Fare Types

EasyJet offers several membership programmes and fare types to customers that affect the price of flights, seats and bags that proved difficult to reflect in the database. EasyJet offers flexi fares guaranteeing a bag type of 23Kg without charge that would prove difficult to implement in the database without a frontend development team.

EasyJet also offers various types of Plus membership in which the passenger may ignore the cost of seat types, bring an additional free bag on board, receive a bistro voucher and other items difficult to reflect in the database (Easyjet.com, 2018).

To improve upon the database a membership type table could be added in which the user may select the type of membership they want such as Plus or Flight Club. Additionally, the specific service they wish to utilise on the flight such as free additional bag or bistro could be linked to a primary key like passenger membership ID and linked to the Passengers table via foreign key constraint.

Information presented from these membership tables would be excellent for a business intelligence report to show popular services that customers utilise and services that may require more advertising.

7.0 Discussion and Justification for a Database (Business Intelligence) Report that could be Created, Along with the Proposed SQL for the Report

7.1 Business Intelligence and Potential Benefits

Business intelligence involves the utilisation of various applications for the collection and analysis of data to help businesses make better decisions (Watson, 2009). In terms of this EasyJet relational database, this process would involve querying for information that will reflect areas of business success and areas where improvements can be made.

Business intelligence has numerous benefits and is implemented in businesses worldwide using specialised software. These benefits can include cost savings, greater understanding of customer needs and increased sales if correct and informative data is collected (Mariani et al, 2018).

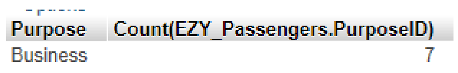
The collection of this data generally follows a business event or business period like a financial year or implementation of a new service or product before the data is stored, analysed and presented in a meaningful way, after which action can be taken. As this database has not been implemented, business intelligence reports that could be created are described below including possible queries that could be used to collect this data.

7.2 Business and Leisure Bookings Example

To predict future customer behaviour, it is important to understand the number of business and leisure bookings at different times of the year. For example, larger numbers of passengers flying for leisure may occur close to the summer or beforehand while the number of passengers flying for business may take place at a steady rate. This information could help forecasting for the next financial year or pinpoint key areas where advertising revenue should be focused at specific periods.

Two examples of these queries are shown below counting the number of passengers flying for business and leisure in November.

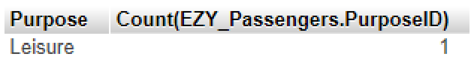
```
SELECT Purpose, Count(EZY_Passengers.PurposeID) FROM EZY_Passengers JOIN EZY_Purpose ON EZY_Passengers.PurposeID = EZY_Purpose.PurposeID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID WHERE EZY_Passengers.PurposeID = 1 AND (FlightDate BETWEEN '2018-11-01' AND '2018-11-30')
```



Purpose	Count(EZY_Passengers.PurposeID)
Business	7

Figure 13. Screenshot showing a sample result of the number of business passengers in the month of November

```
SELECT Purpose, Count(EZY_Passengers.PurposeID) FROM EZY_Passengers JOIN EZY_Purpose ON EZY_Passengers.PurposeID = EZY_Purpose.PurposeID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID WHERE EZY_Passengers.PurposeID = 2 AND (FlightDate BETWEEN '2018-11-01' AND '2018-11-30')
```

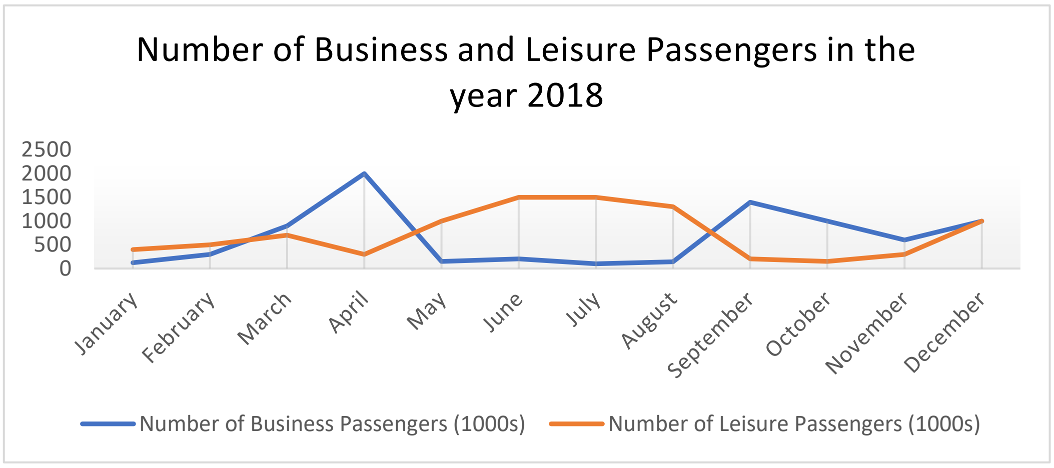


Purpose	Count(EZY_Passengers.PurposeID)
Leisure	1

Figure 14. Screenshot showing a sample result of the number of leisure passengers in the month of November

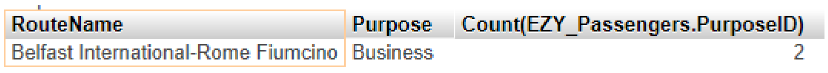
These queries can be used to present graphs like Graph 1 below that shows the number of business and leisure passengers in an understandable way. Importantly this data would be much more usable to executive teams within EasyJet to set targets for business or leisure passengers for the future.

Graph 1: Exemplar graph showing the business and leisure passengers in the year 2018



To ensure data is more specific and presents a worldwide overview of routes or airports popular with business passengers two queries can be used. The first query shown below shows a selected route coupled with the number of business passengers between two dates. This information can be reinforced by the second query showing the number of business passengers departing from a particular airport between two dates.

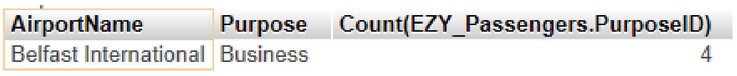
SELECT RouteName, Purpose, Count(EZY_Passengers.PurposeID) FROM EZY_Passengers JOIN EZY_Purpose ON EZY_Passengers.PurposeID = EZY_Purpose.PurposeID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID WHERE EZY_Passengers.PurposeID = 1 AND RouteName = 'Belfast International-Rome Fiumcino' AND (FlightDate BETWEEN '2018-11-01' AND '2018-11-30')



RouteName	Purpose	Count(EZY_Passengers.PurposeID)
Belfast International-Rome Fiumcino	Business	2

Figure 15. Screenshot showing the result of a query selecting the route name and number of business passengers

SELECT AirportName, Purpose, Count(EZY_Passengers.PurposeID) FROM EZY_Passengers JOIN EZY_Purpose ON EZY_Passengers.PurposeID = EZY_Purpose.PurposeID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID JOIN EZY_Airports ON EZY_Routes.OriginAirportID = EZY_Airports.AirportID WHERE EZY_Passengers.PurposeID = 1 AND AirportName = 'Belfast International' AND (FlightDate BETWEEN '2018-11-01' AND '2018-11-30')

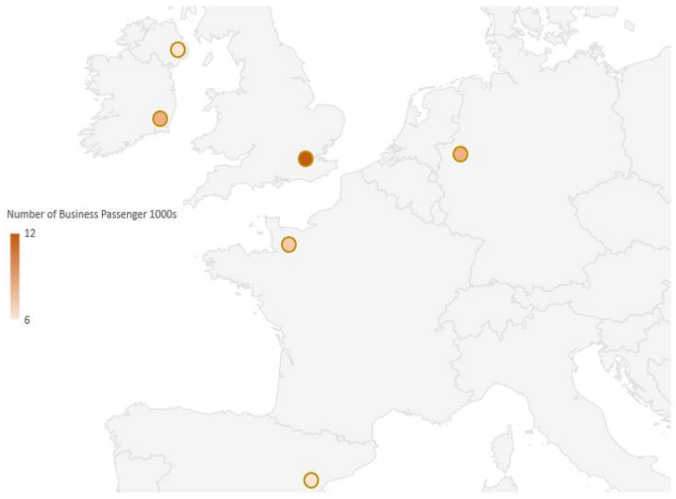


AirportName	Purpose	Count(EZY_Passengers.PurposeID)
Belfast International	Business	4

Figure 16. Screenshot showing the result of a query selecting origin airport and the number of business passengers departing from this airport

Information obtained from these queries will be important for future business decisions. For example, EasyJet offers a FLEXI fare programme designed to save business passengers time that may be more popular in routes with a greater number of these passengers. Information presented from these queries above on origin airports and routes could help forecasts for the next business year and supply an excellent advertising target for the FLEXI programme. Importantly, these queries can help construct geo charts to present easily understandable information to boards in charge of these decisions as shown in Graph 2 showing popular origin airports for business passengers.

Graph 2. Sample Graph showing a map of western Europe with the origin airports of business passengers(1000s) made with GoogleCharts



8.0 Conclusion

In conclusion, the EasyJet database created for this project exhibits the key services the airline offers to customers. This database takes into account connecting and return flights, coupled with the ability of passengers to have numerous bags and select special assistance based on a disability.

Suitable primary keys and foreign key constraints have been created and allow for the successful querying of numerous connected tables at once using joins. Primary keys in the database serve as unique identifiers as intended and foreign key constraints prevent entry of invalid or inconsistent data.

Normalisation and the three normal forms have been adhered to during the creation of the database to minimise input errors and redundancy, while ensuring that new data entered into the database is atomic.

Lastly, suitable improvements have been outlined including ideas for the frontend development team and a suitable proposal for a business intelligence report has been presented, coupled with supporting queries.

Tables, Graphs and Figures

Tables

Graphs

Figures

References

1. Bagui, S. and Earp, R. (2011). *Database Design Using Entity-Relationship Diagrams, Second Edition*. 2nd ed. Hoboken: CRC Press, pp.15-17.

2. Business.worldpay.com. (2018). *Merchant Services – Worldpay*. [online] Available at: https://business.worldpay.com/lp/Merchant_Services_Brand?msclkid=e21ba308882616135a2ef8f9d831799d&utm_source=bing&utm_medium=cpc&utm_campaign=_WP-SME-Brand_Terms-Core-Exact&utm_term=worldpay&utm_content=Core%20WorldPay%20Brand [Accessed 18 Nov. 2018].

3. CHEN, S. and JIN, C. (2009). Set of Necessary and Sufficient Conditions in Collision Attacks on MD5. *Journal of Software*, [online] 20(6), pp.1617-1624. Available at: https://www.researchgate.net/publication/251043472_Set_of_Necessary_and_Sufficient_Conditions_in_Collision_Attacks_on_MD5_Set_of_Necessary_and_Sufficient_Conditions_in_Collision_Attacks_on_MD5 [Accessed 13 Nov. 2018].

4. Easyjet.com. (2018). *Our routes, fares and products | easyJet*. [online] Available at: http://www.easyjet.com/en/help/booking-and-check-in/our-routes-fares-and-products [Accessed 18 Nov. 2018].

5. Easyjet.com. (2018). *Sign In - Manage bookings - easyJet.com*. [online] Available at: https://www.easyjet.com/EN/secure/MyEasyJet.mvc/AllBookings [Accessed 5 Nov. 2018].

6. Foundation, E. (1999). Cracking DES [the Data Encryption Standard]: Secrets of Encryption Research Policies, Wiretap Politics & [Micro]Chip Design: How [U.S. Government] Federal Agencies Subvert Privacy. *EDPACS*, [online] 27(5), pp.1-2. Available at: https://www.tandfonline.com/doi/abs/10.1201/1079/43251.27.5.19991101/30285.5 [Accessed 15 Nov. 2018].

7. Harrington, J. (2016). *Relational database design and implementation*. 4th ed. Amsterdam: Morgan Kaufmann/Elsevier, pp.211-215.

8. Mariani, M., Baggio, R., Fuchs, M. and Höepken, W. (2018). Business intelligence and big data in hospitality and tourism: a systematic literature review. *International Journal of Contemporary Hospitality Management*, [online] 17(9), p.21. Available at: https://www.emeraldinsight.com/doi/abs/10.1108/IJCHM-07-2017-0461 [Accessed 19 Nov. 2018].

9. Nixon, R. (2018). *Learning PHP, MySQL, and JavaScript*. 27th ed. Beijing: O'Reilly, pp.350-356.

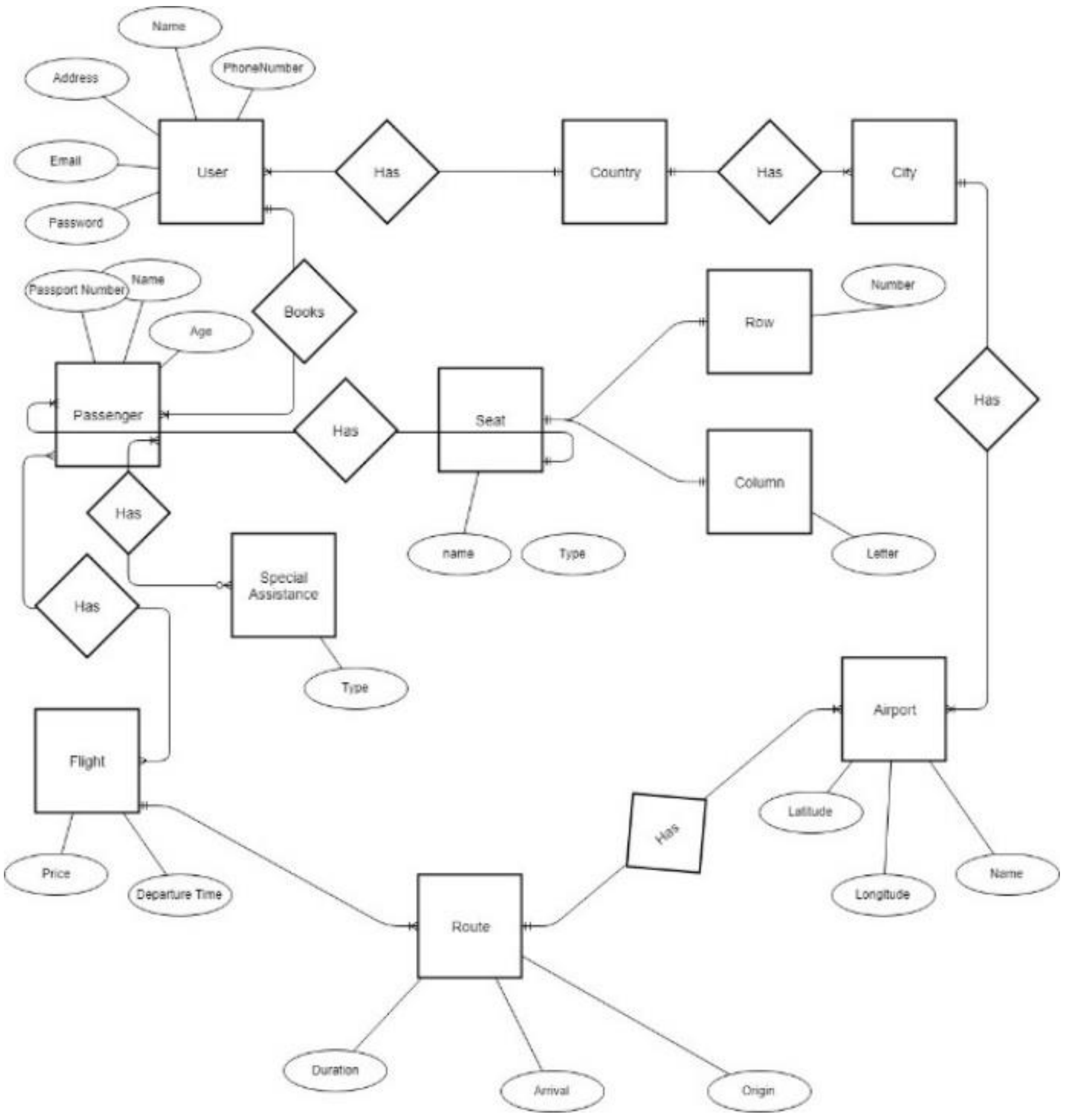
10. Page, A. (1990). *Relational databases*. 1st ed. Wilmslow: Sigma Press, p.35.

11. Shoval, P. and Even-Chaime, M. (1987). Database schema design: an experimental comparison between normalization and information analysis. *ACM SIGMIS Database*, [online] 18(3), pp.30-39. Available at: https://www.researchgate.net/publication/234797241_Database_schema_design_An_experimental_comparison_between_normalization_and_information_analysis [Accessed 12 Nov. 2018].

12. Watson, H. (2009). Tutorial: Business Intelligence – Past, Present, and Future. *Communications of the Association for Information Systems*, [online] 25(39), pp.11-15. Available at: https://www.researchgate.net/publication/234797241_Database_schema_design_An_experimental_comparison_between_normalization_and_information_analysis [Accessed 18 Nov. 2018].

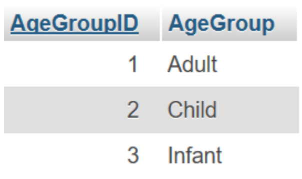
Appendix A

Initial basic group ER-diagram



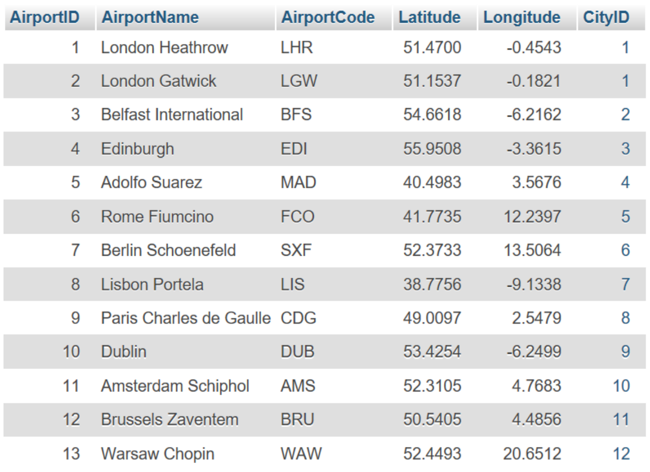
Appendix B

Screenshots of important tables mentioned in the report including sample data.



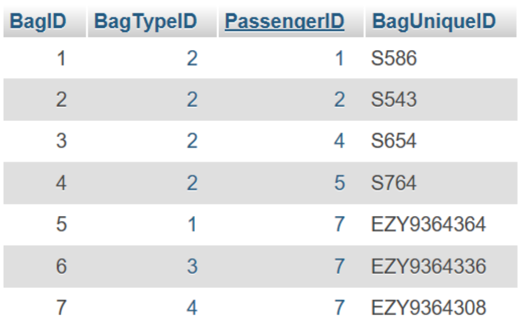
AgeGroupID	AgeGroup
1	Adult
2	Child
3	Infant

Appendix A 1. Age Group Table



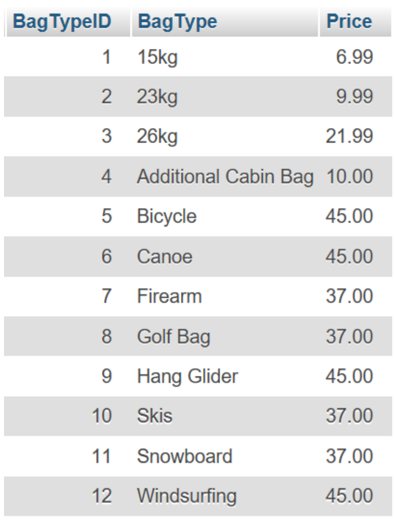
AirportID	AirportName	AirportCode	Latitude	Longitude	CityID
1	London Heathrow	LHR	51.4700	-0.4543	1
2	London Gatwick	LGW	51.1537	-0.1821	1
3	Belfast International	BFS	54.6618	-6.2162	2
4	Edinburgh	EDI	55.9508	-3.3615	3
5	Adolfo Suarez	MAD	40.4983	3.5676	4
6	Rome Fiumcino	FCO	41.7735	12.2397	5
7	Berlin Schoenefeld	SXF	52.3733	13.5064	6
8	Lisbon Portela	LIS	38.7756	-9.1338	7
9	Paris Charles de Gaulle	CDG	49.0097	2.5479	8
10	Dublin	DUB	53.4254	-6.2499	9
11	Amsterdam Schiphol	AMS	52.3105	4.7683	10
12	Brussels Zaventem	BRU	50.5405	4.4856	11
13	Warsaw Chopin	WAW	52.4493	20.6512	12

Appendix A 2. Airport Table



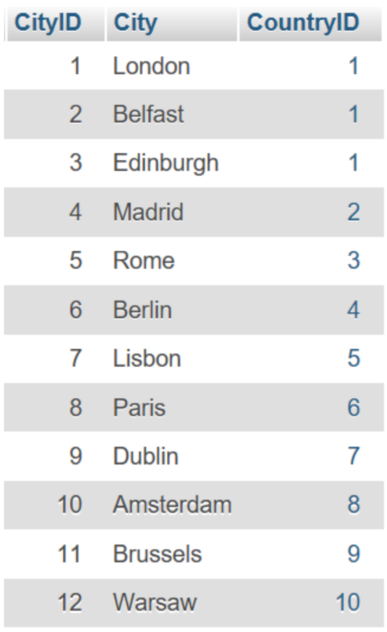
BagID	BagTypeID	PassengerID	BagUniqueID
1	2	1	S586
2	2	2	S543
3	2	4	S654
4	2	5	S764
5	1	7	EZY9364364
6	3	7	EZY9364336
7	4	7	EZY9364308

Appendix A 3. Bags Table



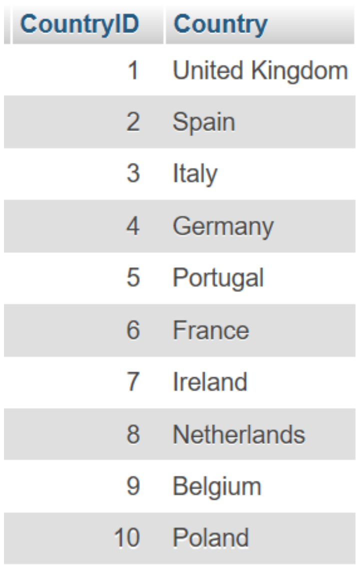
BagTypeID	BagType	Price
1	15kg	6.99
2	23kg	9.99
3	26kg	21.99
4	Additional Cabin Bag	10.00
5	Bicycle	45.00
6	Canoe	45.00
7	Firearm	37.00
8	Golf Bag	37.00
9	Hang Glider	45.00
10	Skis	37.00
11	Snowboard	37.00
12	Windsurfing	45.00

Appendix A 4. BagType Table



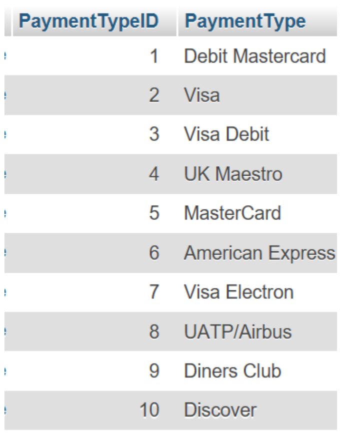
CityID	City	CountryID
1	London	1
2	Belfast	1
3	Edinburgh	1
4	Madrid	2
5	Rome	3
6	Berlin	4
7	Lisbon	5
8	Paris	6
9	Dublin	7
10	Amsterdam	8
11	Brussels	9
12	Warsaw	10

Appendix A 5. Cities Table



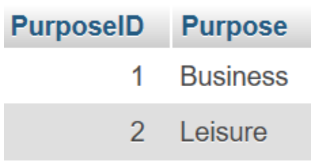
CountryID	Country
1	United Kingdom
2	Spain
3	Italy
4	Germany
5	Portugal
6	France
7	Ireland
8	Netherlands
9	Belgium
10	Poland

Appendix A 4. Countries Table



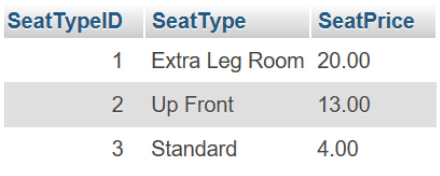
PaymentTypeID	PaymentType
1	Debit Mastercard
2	Visa
3	Visa Debit
4	UK Maestro
5	MasterCard
6	American Express
7	Visa Electron
8	UATP/Airbus
9	Diners Club
10	Discover

Appendix A 7. Payment Type Table



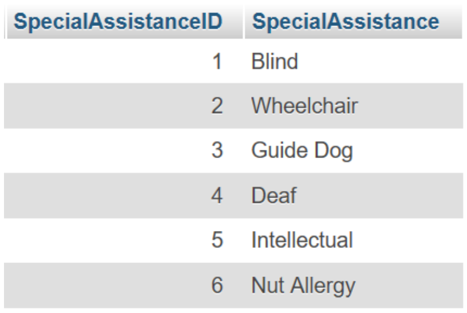
PurposeID	Purpose
1	Business
2	Leisure

Appendix A 8. Purpose Table



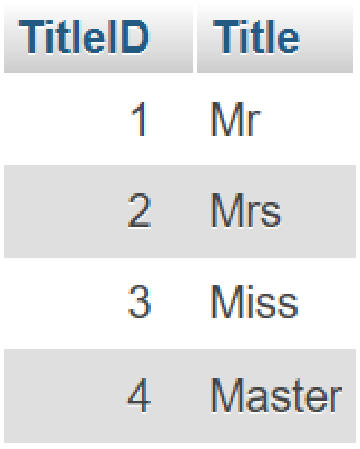
SeatTypeID	SeatType	SeatPrice
1	Extra Leg Room	20.00
2	Up Front	13.00
3	Standard	4.00

Appendix A 9. Seat Type Table



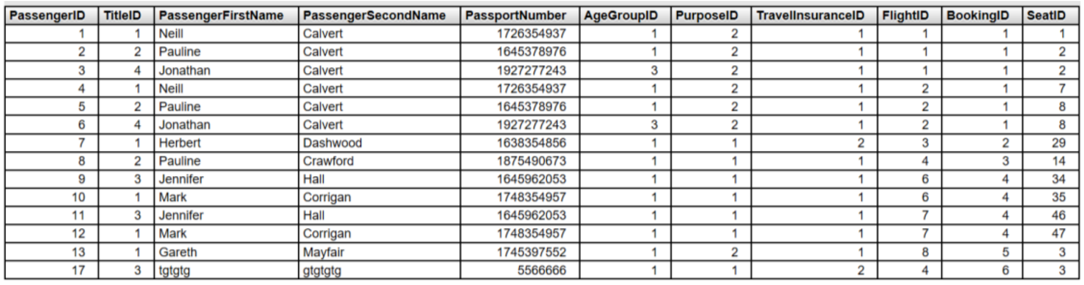
SpecialAssistanceID	SpecialAssistance
1	Blind
2	Wheelchair
3	Guide Dog
4	Deaf
5	Intellectual
6	Nut Allergy

Appendix A 10. Special Assistance Table



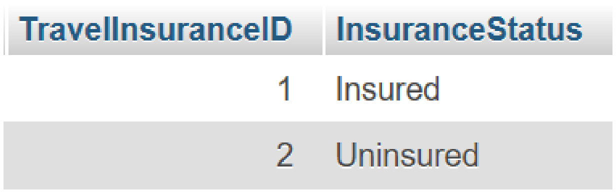
TitleID	Title
1	Mr
2	Mrs
3	Miss
4	Master

Appendix A 11. Titles Table



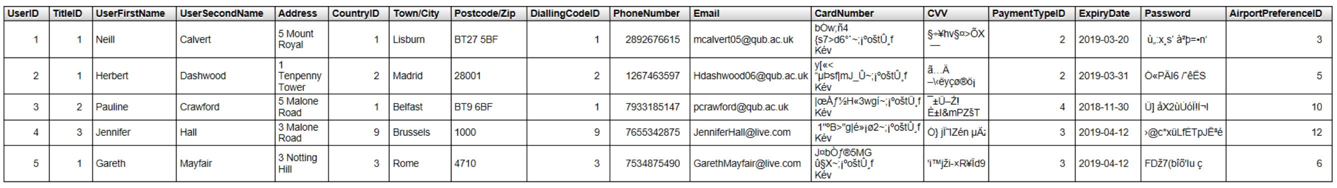
PassengerID	TitleID	PassengerFirstName	PassengerSecondName	PassportNumber	AgeGroupID	PurposeID	TravelInsuranceID	FlightID	BookingID	SeatID
1	1	Neill	Calvert	1726354937	1	2	1	1	1	1
2	2	Pauline	Calvert	1645378976	1	2	1	1	1	2
3	4	Jonathan	Calvert	1927277243	3	2	1	1	1	2
4	1	Neill	Calvert	1726354937	1	2	1	2	1	7
5	2	Pauline	Calvert	1645378976	1	2	1	2	1	8
6	4	Jonathan	Calvert	1927277243	3	2	1	2	1	8
7	1	Herbert	Dashwood	1638354856	1	1	2	3	2	29
8	2	Pauline	Crawford	1875490673	1	1	1	4	3	14
9	3	Jennifer	Hall	1645962053	1	1	1	6	4	34
10	1	Mark	Corrigan	1748354957	1	1	1	6	4	35
11	3	Jennifer	Hall	1645962053	1	1	1	7	4	46
12	1	Mark	Corrigan	1748354957	1	1	1	7	4	47
13	1	Gareth	Mayfair	1745397552	1	2	1	8	5	3
17	3	tgtgtg	gtgtgtg	5566666	1	1	2	4	6	3

Appendix A12. Passengers Table



TravelInsuranceID	InsuranceStatus
1	Insured
2	Uninsured

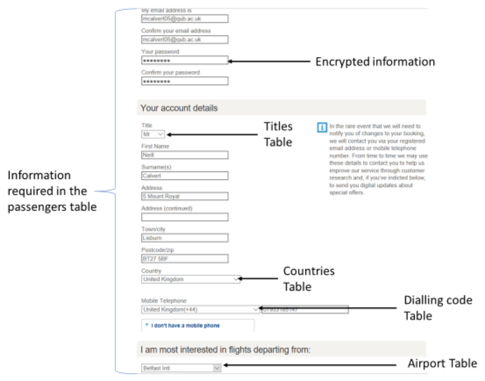
Appendix A 13. Travel Insurance Table

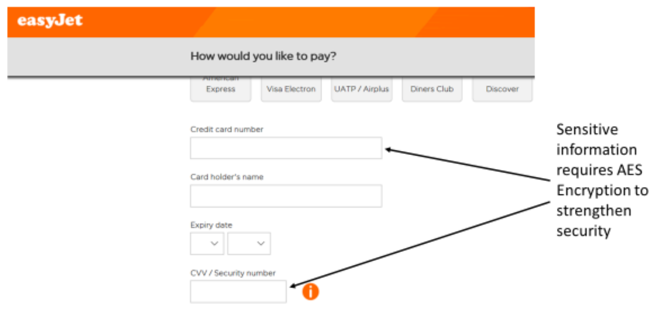


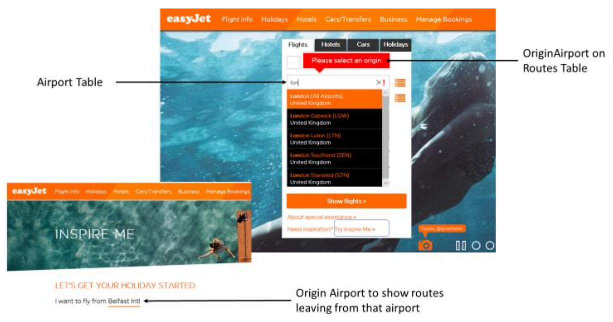
UserID	TitleID	UserFirstName	UserSecondName	Address	CountryID	Town/City	Postcode/Zip	DiallingCodeID	PhoneNumber	Email	CardNumber	CVV	PaymentTypeID	ExpiryDate	Password	AirportPreferenceID
1	1	Neill	Calvert	5 Mount Royal	1	Lisburn	BT27 5BF	1	2892676615	mcalvert05@qub.ac.uk	bÒw;ñ4 {s7>d6°`~;¡ºoštÛ¸f Kév	§÷¥hv§¤>ÕX —	2	2019-03-20	ù„:x¸s' à²þ=•n'	3
2	1	Herbert	Dashwood	1 Tenpenny Tower	2	Madrid	28001	2	1267463597	Hdashwood06@qub.ac.uk	y[«< "µÞsf|mJ_Û~;¡ºoštÛ¸f Kév	ã…Ä –\‹ëyçø®ö¡	2	2019-03-31	Ò«PÄl6 /ˆêËS	5
3	2	Pauline	Crawford	5 Malone Road	1	Belfast	BT9 6BF	1	7933185147	pcrawford@qub.ac.uk	|œÅƒ½H«3wgí~;¡ºoštÛ¸f Kév	¯±Ü–Ż! È±l&mPZšT	4	2018-11-30	Ü] åX2ùÙóÏ!Ì¬l	10
4	3	Jennifer	Hall	3 Malone Road	9	Brussels	1000	9	7655342875	JenniferHall@live.com	1"ºB>"g|é»¡ø2~;¡ºoštÛ¸f Kév	Ò} jÏ˜IZén µÄ;	3	2019-04-12	›@c*xüLfÉTpJË³é	12
5	1	Gareth	Mayfair	3 Notting Hill	3	Rome	4710	3	7534875490	GarethMayfair@live.com	J¤bÒƒ®5MG û§X~;¡ºoštÛ¸f Kév	'i™jži-×R¥Ïd9	3	2019-04-12	FDž7(bîõ'lu ç	6

Appendix A 14. Users Table

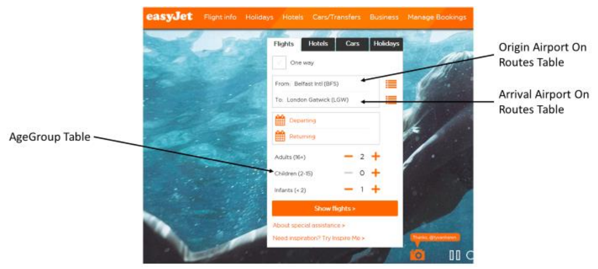
Appendix C



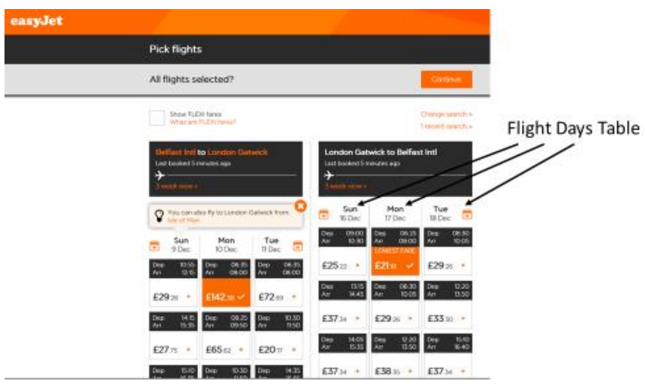


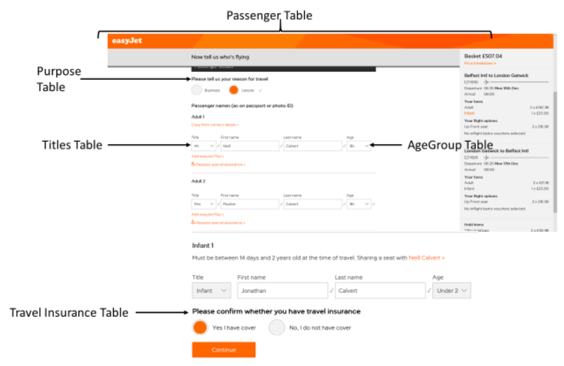


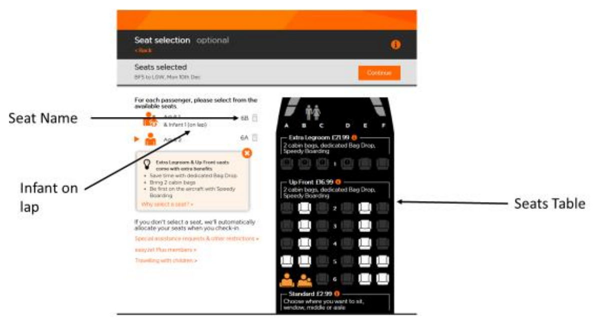
easyJet
Airport Table
Please select an origin
OriginAirport on Routes Table
Show flights
INSPIRE ME
LET'S GET YOUR HOLIDAY STARTED
I want to fly from Belfast Intl
Origin Airport to show routes leaving from that airport

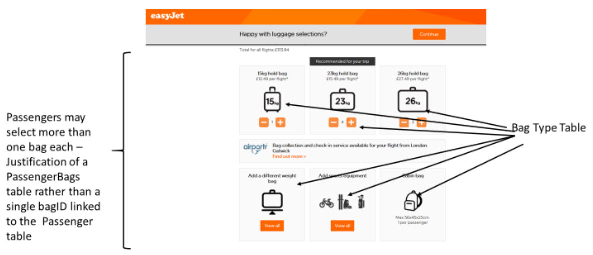


easyJet
Origin Airport On Routes Table
Arrival Airport On Routes Table
AgeGroup Table
Show flights



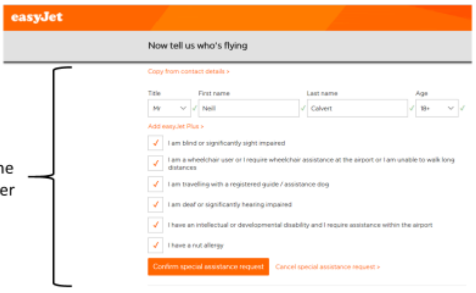






Passengers may select more than one bag each – Justification of a PassengerBags table rather than a single bagID linked to the Passenger table

Ability to choose more than one special assistance per passenger



easyJet

Now tell us who's flying

Copy from contact details >

Title	First name	Last name	Age
Mr	Neill	Calvert	18+

Add easyJet Plus >

✓ I am blind or significantly sight impaired

✓ I am a wheelchair user or I require wheelchair assistance at the airport or I am unable to walk long distances

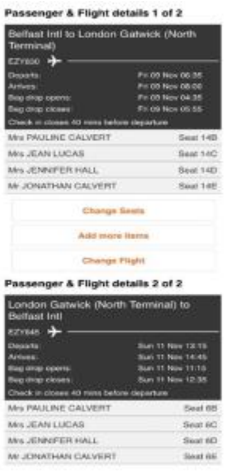
✓ I am travelling with a registered guide / assistance dog

✓ I am deaf or significantly hearing impaired

✓ I have an intellectual or developmental disability and I require assistance within the airport

✓ I have a nut allergy

Confirm special assistance request   Cancel special assistance request >



Passenger & Flight details 1 of 2

Belfast Intl to London Gatwick (North Terminal)

EZY800

Departs:	Fri 09 Nov 06:35
Arrives:	Fri 09 Nov 08:00
Bag drop opens:	Fri 09 Nov 04:35
Bag drop closes:	Fri 09 Nov 05:55

Check in closes 40 mins before departure

Mrs PAULINE CALVERT	Seat 14B
Mrs JEAN LUCAS	Seat 14C
Mrs JENNIFER HALL	Seat 14D
Mr JONATHAN CALVERT	Seat 14E

Change Seats

Add more items

Change Flight

Passenger & Flight details 2 of 2

London Gatwick (North Terminal) to Belfast Intl

EZY846

Departs:	Sun 11 Nov 13:15
Arrives:	Sun 11 Nov 14:45
Bag drop opens:	Sun 11 Nov 11:15
Bag drop closes:	Sun 11 Nov 12:35

Check in closes 40 mins before departure

Mrs PAULINE CALVERT	Seat 6B
Mrs JEAN LUCAS	Seat 6C
Mrs JENNIFER HALL	Seat 6D
Mr JONATHAN CALVERT	Seat 6E

Transaction showing rollback of a user insertion

START TRANSACTION;

SAVEPOINT S1;

INSERT INTO `EZY_Users` (`UserID`, `TitleID`, `UserFirstName`, `UserSecondName`, `Address`, `CountryID`, `Town/City`, `Postcode/Zip`, `DiallingCodeID`, `PhoneNumber`, `Email`, `CardNumber`, `CVV`, `PaymentTypeID`, `ExpiryDate`, `Password`, `AirportPreferenceID`) VALUES (NULL, '3', 'Jennifer', 'Hall', '3 Malone Road', '9', 'Brussels', '1000', '9', '07655342875', 'JenniferHall@live.com', AES_ENCRYPT('2538564539152637','cardKey'), AES_ENCRYPT('548','cvvKey'), '3', '2019-04-12', AES_ENCRYPT('hallsy','passwordKey'), '12');

ROLLBACK TO S1 ;

COMMIT

Decryption of AES encrypted columns

SELECT UserFirstName, UserSecondName, AES_DECRYPT(CardNumber, 'cardKey'), AES_DECRYPT(CVV, 'cvvKey'), AES_DECRYPT(Password, 'passwordKey') FROM EZY_Users

Select origin airports beginning with 'Lon' and the routes flying from these airports using a wildcard like query
SELECT RouteName FROM EZY_Routes INNER JOIN EZY_Airports ON EZY_Routes.OriginAirportID = EZY_Airports.AirportID WHERE AirportName LIKE 'Lon%'

Select origin airports beginning with 'Lon' and the routes flying from these airports under 5 hours using a wildcard like query
SELECT RouteName, RouteDuration FROM EZY_Routes JOIN EZY_Airports ON EZY_Routes.OriginAirportID = EZY_Airports.AirportID WHERE AirportName LIKE 'Lon%' AND RouteDuration <= '05:00:00'

Inner Join showing the flight name, route name and the days these flights travel

SELECT FlightName, RouteName, FlightDays, FlightDepartureTime FROM EZY_Flights JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID JOIN EZY_FlightDays ON EZY_Flights.FlightDayID = EZY_FlightDays.FlightDayID

SELECT FlightName, RouteName, FlightDays, FlightDepartureTime FROM EZY_Flights JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID JOIN EZY_FlightDays ON EZY_Flights.FlightDayID = EZY_FlightDays.FlightDayID WHERE RouteName = 'Belfast International-London Heathrow'

SELECT FlightName, RouteName, FlightDays, FlightDepartureTime, RouteDuration FROM EZY_Flights JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID JOIN EZY_FlightDays ON EZY_Flights.FlightDayID = EZY_FlightDays.FlightDayID WHERE RouteName = 'Belfast International-London Heathrow'

SELECT FlightName, RouteName, FlightDays, FlightDepartureTime, RouteDuration, FlightBasePrice FROM EZY_Flights JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID JOIN EZY_FlightDays ON EZY_Flights.FlightDayID = EZY_FlightDays.FlightDayID WHERE RouteName = 'Belfast International-London Heathrow'

Insertion of a passenger into the passenger table

INSERT INTO `EZY_Passengers` (`PassengerID`, `TitleID`, `PassengerFirstName`, `PassengerSecondName`, `PassportNumber`, `AgeGroupID`, `PurposeID`, `TravelInsuranceID`, `FlightID`, `BookingID`, `SeatID`) VALUES (NULL, '3', 'Jenny', 'Calvert', '1647858749', '1', '2', '1', '2', '1', '9');

Show passengers on all seats on all plane types

SELECT * FROM EZY_Seats LEFT JOIN EZY_Passengers ON EZY_Seats.SeatID = EZY_Passengers.SeatID WHERE PlaneTypeID = 1

Counting the number of business passengers on a flight

SELECT FlightName, RouteName, Purpose, Count(EZY_Passengers.PurposeID) AS 'Number of Leisure Passengers' FROM EZY_Passengers JOIN EZY_Purpose ON EZY_Passengers.PurposeID = EZY_Purpose.PurposeID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID WHERE EZY_Flights.FlightID = 1 AND Purpose = 'Leisure'

Left join showing all flights with any passengers booked on those flights (note the null where a flight is empty)

SELECT * FROM EZY_Flights LEFT JOIN EZY_Passengers ON EZY_Flights.FlightID = EZY_Passengers.FlightID

Count distinct query that presents the number of seats taken on a flight while taking into account the fact an infant shares a seat

SELECT FlightName, COUNT(DISTINCT EZY_Seats.SeatID) AS 'Number of seats taken' FROM EZY_Seats JOIN EZY_Passengers ON EZY_Seats.SeatID = EZY_Passengers.SeatID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID WHERE EZY_Flights.FlightID = 1

INNER JOIN SHOWING MULTIPLE BAGS CONNECTED TO A SINGLE PASSENGER

SELECT PassengerFirstName, PassengerSecondName, BagType FROM EZY_Passengers JOIN EZY_Bags ON EZY_Passengers.PassengerID = EZY_Bags.PassengerID JOIN EZY_BagTypes ON EZY_Bags.BagTypeID = EZY_BagTypes.BagTypeID

Inner join showing multiple bags connected to a single passenger limiting the return to 7
SELECT PassengerFirstName, PassengerSecondName, BagType FROM EZY_Passengers JOIN EZY_Bags ON EZY_Passengers.PassengerID = EZY_Bags.PassengerID JOIN EZY_BagTypes ON EZY_Bags.BagTypeID = EZY_BagTypes.BagTypeID LIMIT 7

Passenger Table JOIN TO SHOW ONE USER WITH NUMEROUS PASSENGERS AND RETURN FLIGHTS ON A SINGLE BOOKING

SELECT UserFirstName, UserSecondName, BookingReference, PassengerFirstName, PassengerSecondName, FlightName, RouteName FROM EZY_Users JOIN EZY_Bookings ON EZY_Users.UserID = EZY_Bookings.UserID JOIN EZY_Passengers ON EZY_Passengers.BookingID = EZY_Bookings.BookingID JOIN EZY_Flights ON EZY_Passengers.FlightID = EZY_Flights.FlightID JOIN EZY_Routes ON EZY_Flights.RouteID = EZY_Routes.RouteID WHERE EZY_Bookings.BookingID = 1

Left join of special assistance to show all type of special assistance and any passengers

SELECT * FROM EZY_SpecialAssistance LEFT JOIN EZY_PassengerSpecialAssistance ON EZY_SpecialAssistance.SpecialAssistanceID = EZY_PassengerSpecialAssistance.SpecialAssitanceID LEFT JOIN EZY_Passengers ON EZY_PassengerSpecialAssistance.PassengerID = EZY_Passengers.PassengerID

Sum of the booking details page

SELECT SUM(ItemPrice) FROM EZY_BookingDetails JOIN EZY_Bookings ON EZY_Bookings.BookingID = EZY_BookingDetails.BookingID JOIN EZY_Users ON EZY_Bookings.UserID = EZY_Users.UserID

Sum of the booking details page with 20% tax to 2 two decimal places

SELECT ROUND (SUM(ItemPrice) *1.2, 2) FROM EZY_BookingDetails JOIN EZY_Bookings ON EZY_Bookings.BookingID = EZY_BookingDetails.BookingID JOIN EZY_Users ON EZY_Bookings.UserID = EZY_Users.UserID

Delete a booking from the passengers table showing a cascade delete

DELETE FROM EZY_Bookings WHERE BookingID = 6

Delete passenger from a booking

DELETE FROM EZY_Passengers WHERE PassengerID = 15