CT004-3-3-ADVBS

ADVANCED DATABASE SYSTEMS

UC3F1812SE / UC3F1812IT

LUA HAU ZHENG (TP038936)
LAU ZHEN XIAN (TP038767)
FOONG WING YAN (TP038550)

ABDALLAH S.M. ALNATSHA

Workload Matrix

No	Name	Intake	Student ID
1	Lua Hau Zheng	UC3F1812SE	TP038936
2	Lau Zhen Xian	UC3F1812SE	TP038767
3	Foong Wing Yan	UC3F1812IT	TP038550

No	Chapter	Student Name		
		Hau Zheng	Zhen Xian	Wing Yan
1	Chapter 1	33%	33%	33%
2	Chapter 2	33%	33%	33%
3	Chapter 3	33%	33%	33%
4	Chapter 4	33%	33%	33%
5	Chapter 5	33%	33%	33%
6	Chapter 6	33%	33%	33%

Signature		
Student Name		
Lua Hau Zheng (Leader)	Lau Zhen Xian	Foong Wing Yan

TABLE OF CONTENTS

INTRODUCTION

The Database life cycle has six step which included requirement gathering, analysis, logical design, implementation, realizing the design and populating the database. Requirement gathering is the first step of database life cycle. During this step, the database designers have to understand the requirement of the system and document the data such as functional requirements. For example, database designer needs to collect the information of Travel Safe International company and this data requirements document is used for reference in order to more understanding. Hence, the result of this step is a document that includes the detailed requirements [CITATION ADR19 \l 1033].

Second step of database life cycle is data analysis which start with the statement of data requirements and then produces a conceptual data model. The purpose of analysis is to obtain a detailed description of the data that will suitable requirements [CITATION ADR19 \l 1033]. These include properties such as the possible range of values that can be permitted for attributes. For example, database designer needs to know the flight code, flight name, flight fare and more for Travel Safe International company. Hence, a conceptual data model is concerned with the meaning and structure of data and not with the details affecting how they are implemented [CITATION ADR19 \l 1033].

Third step which is logical design and it begin with a conceptual data model and procedures a specification of a logical schema. Then, this will identify the specific type of database system such as relational that is required. But the relational representation is still independent of any specific Database Management System (DBMS) and it is another conceptual model. Database designer can adopt a relational representation of the conceptual data model as input to the logical design process [CITATION ADR19 \l 1033]. For example, database designer can determine all the tables which needed for Travel Safe International company. Hence, the output of this stage is a detailed relational specification, the logical schema, of all the tables and constraints needed to satisfy the description of the data in the conceptual data model [CITATION ADR19 \l 1033].

Next which is the implementation part and it is the fourth step in database life cycle. It involves the construction of a database according to the specification of a logical schema. Implementation also heavily influenced by the choice of available DBMS's database tools and operating environment because there are additional tasks beyond simply creating a database schema such as data must be entered into the tables [CITATION ADR19 \l 1033].

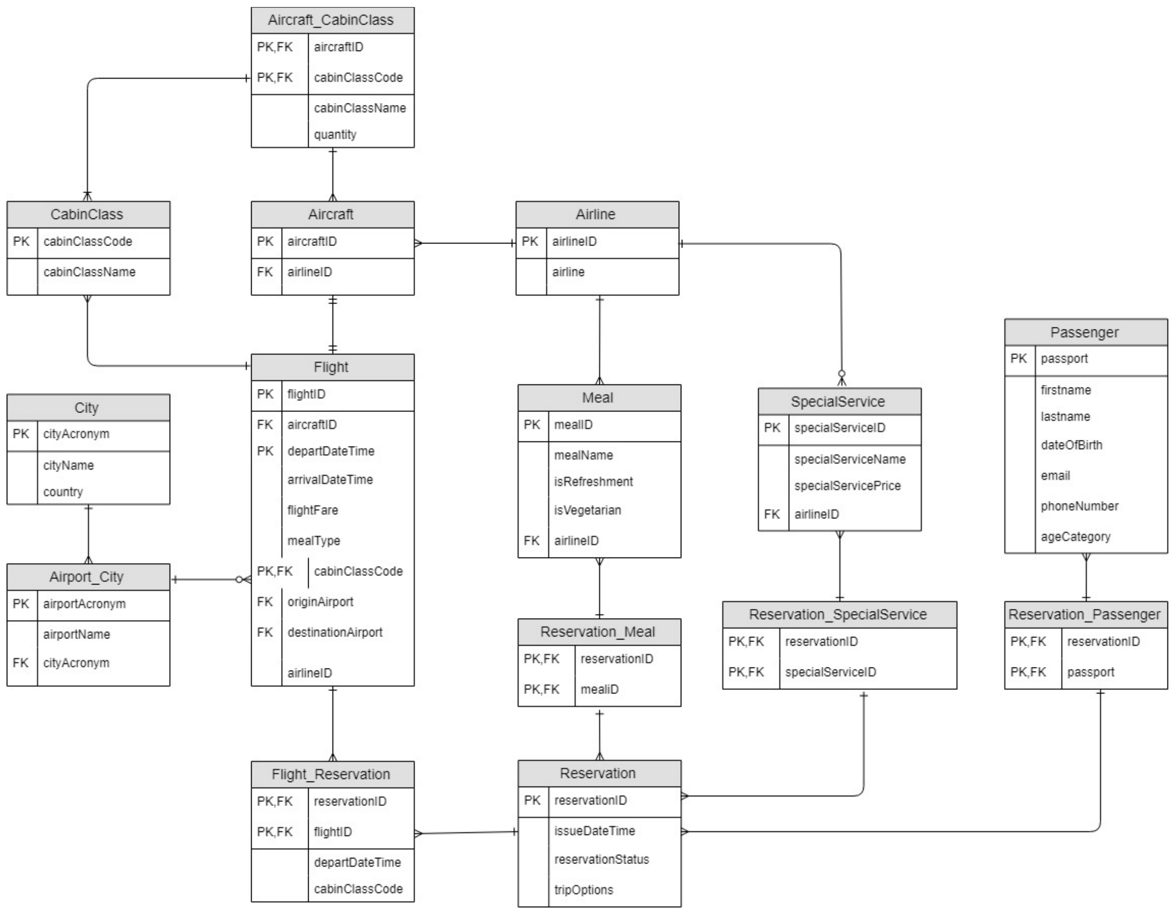
In database terms, this might involve choosing vendor products with DBMS and SQL variants most suitable to the database we need to implement [CITATION ADR19 \l 1033]. Therefore, implementation can involve additional flexing of the design to overcome any software or hardware limitations.

Fifth step for database life cycle is realizing the design. In this step, database designer needs their database to be created according to the definitions they have produced after the logical design has been created. For an implementation with a relational DBMS, this will probably involve the use of SQL to create tables and constraints that satisfy the logical schema description and the choice of appropriate storage schema [CITATION ADR19 \l 1033]. For example, database designer needs to enter the column name for each table and also identify the datatype for each data. One way to achieve this is to write the appropriate SQL DDL statements into a file that can be executed by a DBMS so that there is an independent record, a text file, of the SQL statements defining the database [CITATION ADR19 \l 1033]. Hence, at here, database designer needs to write a query statement so that SQL can able to working successfully.

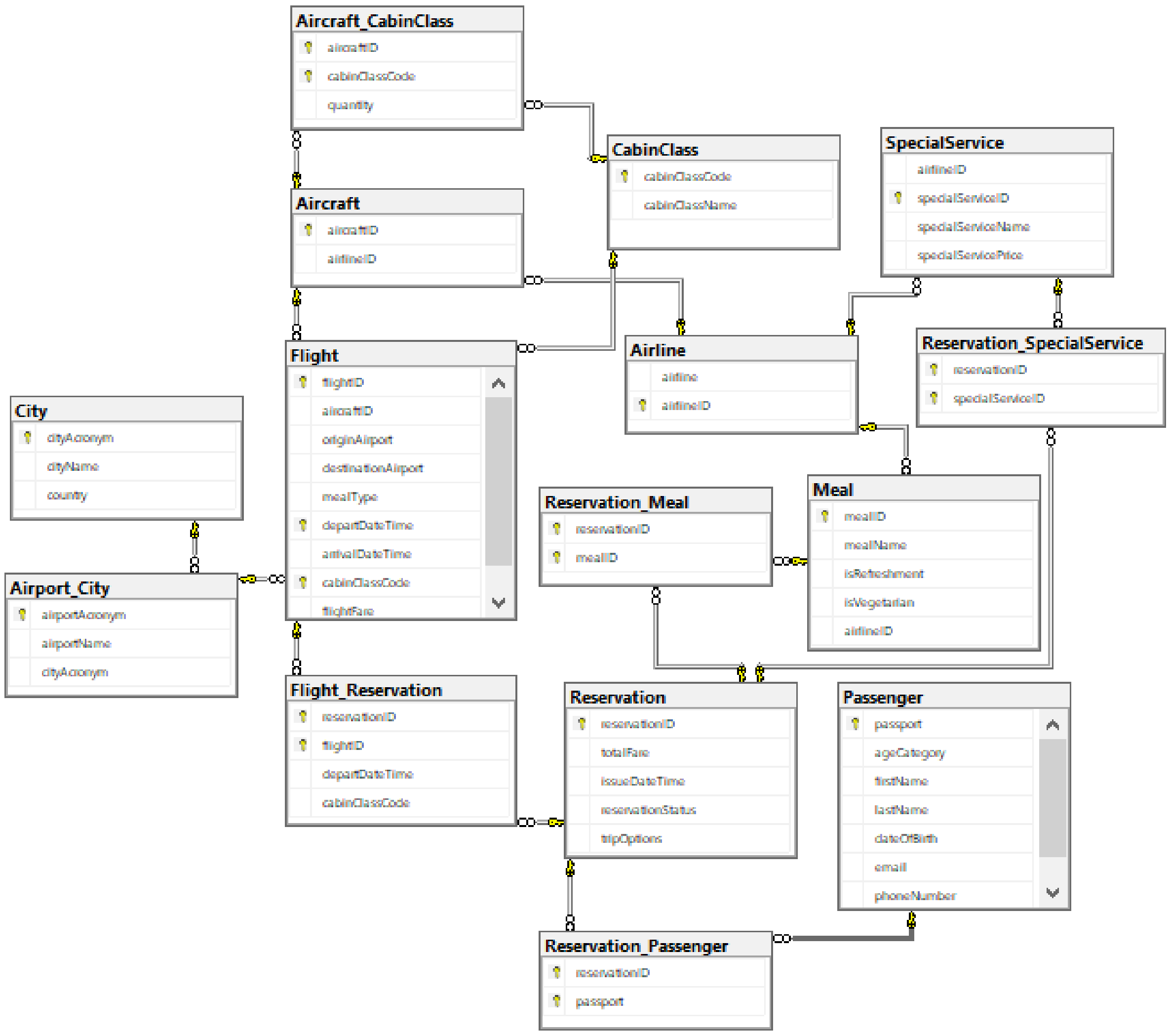
Sixth step which also the last step of database life cycle which is populating the database. After the database has been created, there are two ways of populating the tables such as either from existing data or through the use of the user applications developed for the database [CITATION ADR19 \l 1033]. Therefore, database life cycle able to let the proposed system successful.

CHAPTER 1.0
DESIGN

1.1 Entity Relationship Diagram (Crow's foot notation)



1.2 Entity Relation Model



CHAPTER 2.0
OPTIMIZATION STRATEGY

2.1 LUA HAU ZHENG (TP038936)

```sql
create table CabinClass(
    cabinClassCode varchar(5),
    cabinClassName varchar(255),
    primary key(cabinClassCode)
);

create table Aircraft_CabinClass (
    aircraftID varchar(5),
    cabinClassCode varchar(5),
    ...
    primary key(aircraftID, cabinClassCode),
    foreign key(aircraftID) references Aircraft(aircraftID),
    foreign key(cabinClassCode) references CabinClass(CabinClassCode),
);
```

The query above shows the creation of normalisation table. If a developer wishes to retrieve the cabinClassName based on the cabinClassCode, he/she needs to inner join with the CabinClass table. However, through denormalization the table by adding a cabinClassName to the AirCraft_CabinClass table, it can reduce the join that is needed to retrieve the cabinClassName and increasing its processing speed. This also decreases the table amount in the database.

```sql
create table Aircraft_CabinClass (
    aircraftID varchar(5),
    cabinClassCode varchar(5),
    cabinClassName varchar(255),
    primary key(aircraftID, cabinClassCode),
    foreign key(aircraftID) references Aircraft(aircraftID),
    foreign key(cabinClassCode) references CabinClass(CabinClassCode),
);
```

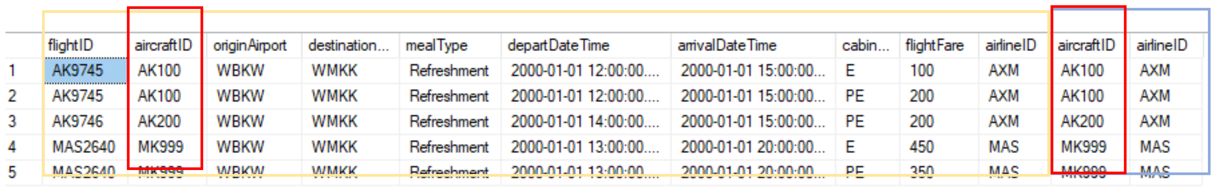
Code above shows the creation the Aircraft_CabinClass table with full details of cabin class in denormalization form.

2.2 LAU ZHEN XIAN (TP038767)

```sql
select *
from Flight
inner join Aircraft
on Flight.aircraftID = Aircraft.aircraftD
```

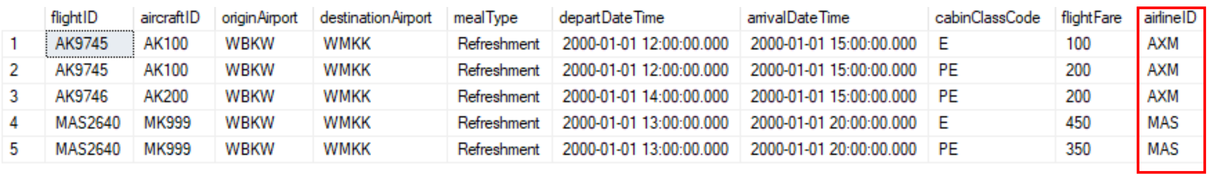
The query above is used to acquire the flight information along with its corresponding airline company ID. In order to acquire the airline company ID, the developer will be required to inner join two tables which are Flight and Aircraft. This situation occurred because Flight table do not store airlineID, whereas the Aircraft table has the data of the airlineID. Both tables need to be linked together based on their aircraftID to get the airlineID. The table below is represented in normalised form and the frame colour in the below diagram indicates:

Colour	Indication
Yellow	Flight table
Blue	Aircraft table
Red	Same column of Flight and Aircraft table (aircraftID)



	flightID	aircraftID	originAirport	destination...	mealType	departDateTime	arrivalDateTime	cabin...	flightFare	airlineID	aircraftID	airlineID
1	AK9745	AK100	WBKW	WMKK	Refreshment	2000-01-01 12:00:00....	2000-01-01 15:00:00...	E	100	AXM	AK100	AXM
2	AK9745	AK100	WBKW	WMKK	Refreshment	2000-01-01 12:00:00....	2000-01-01 15:00:00...	PE	200	AXM	AK100	AXM
3	AK9746	AK200	WBKW	WMKK	Refreshment	2000-01-01 14:00:00....	2000-01-01 15:00:00...	PE	200	AXM	AK200	AXM
4	MAS2640	MK999	WBKW	WMKK	Refreshment	2000-01-01 13:00:00....	2000-01-01 20:00:00...	E	450	MAS	MK999	MAS
5	MAS2640	MK999	WBKW	WMKK	Refreshment	2000-01-01 13:00:00....	2000-01-01 20:00:00...	PE	350	MAS	MK999	MAS

Denormalization can be done to decrease the join of data by adding an airlineID column in the Flight table. By doing so, the retrieval of airlineID can be done without joining other tables, thus speeding up the retrieval time.



	flightID	aircraftID	originAirport	destinationAirport	mealType	departDateTime	arrivalDateTime	cabinClassCode	flightFare	airlineID
1	AK9745	AK100	WBKW	WMKK	Refreshment	2000-01-01 12:00:00.000	2000-01-01 15:00:00.000	E	100	AXM
2	AK9745	AK100	WBKW	WMKK	Refreshment	2000-01-01 12:00:00.000	2000-01-01 15:00:00.000	PE	200	AXM
3	AK9746	AK200	WBKW	WMKK	Refreshment	2000-01-01 14:00:00.000	2000-01-01 15:00:00.000	PE	200	AXM
4	MAS2640	MK999	WBKW	WMKK	Refreshment	2000-01-01 13:00:00.000	2000-01-01 20:00:00.000	E	450	MAS
5	MAS2640	MK999	WBKW	WMKK	Refreshment	2000-01-01 13:00:00.000	2000-01-01 20:00:00.000	PE	350	MAS

The query above shows the SQL to acquire the flight information along with its corresponding airline company ID. As shown in the query, the developer can now acquire the airline company ID without the needs of inner joining extra table. This is because the table has been added a duplicate column to speed up the performance to obtain the airline company ID.

2.3 FOONG WING YAN (TP038550)

```sql
create table Meal(
    mealID varchar(10),
    ...
    primary key (mealID)
);

create table Meal_Airline(
    mealID varchar(10),
    airlineID varchar(5),
    ...
    primary key (mealID, airlineID)
);
```

The query above displays the normalisation of meal that is available for different kind of airline company. However, through adding the airlineID into the Meal table, it allows for the data to be retrieved easily without joining the Meal_Aircraft table.

```sql
create table Meal(
    mealID varchar(10),
    ...
    airlineID varchar(5)
    foreign key (airlineID) references Airline(airlineID),
    primary key (mealID)
);
```

The query above is the denormalization form of Meal table by adding a new airlineID column. In this way, the retrieval of meal that is provided by different airline can be program in one line of code.

```sql
select *
from meal
```

CHAPTER 3.0
CONSTRAINT

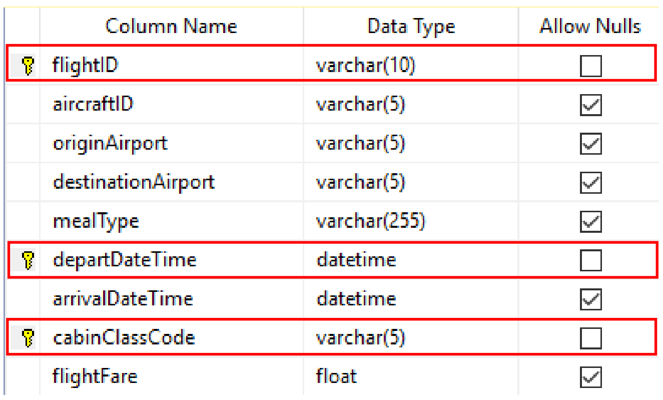
Constraint in SQL is used to limit the user input into the system, which can benefit the database through enhancing the data integrity, accuracy and reliability. Constraint can be separated into table constraint where the constraint will be applied to the entire table and column constraint which will be applied to the columns in table. If there are any violation between the input with the constraint, the input will be aborted and require new input from the user[CITATION w3s19 \l 2057]. Below will discuss several types of constraints.

3.1 LUA HAU ZHENG (TP038936)

PRIMARY KEY is a constraint combined of both NOT NULL constraint and UNIQUE constraint to eliminate the chances of NULL and duplicate value within the same column. In the code below, flightID, departDateTime and cabinClassCode are assigned as the PRIMARY KEYS for Flight table.

```
create table Flight(
    flightID varchar(10),
    departDateTime datetime,
    cabinClassCode varchar(5),
    ...
    primary key (flightID, departDateTime, cabinClassCode),
);
```

In the table design view that is displayed in below, the PRIMARY KEY can be identified by the yellow keys that is placed beside the column name. These three (3) column also do not have tick in the "Allow Nulls" column in table design view because PRIMARY KEY cannot be NULL.



	Column Name	Data Type	Allow Nulls
🔑	flightID	varchar(10)	☐
	aircraftID	varchar(5)	✓
	originAirport	varchar(5)	✓
	destinationAirport	varchar(5)	✓
	mealType	varchar(255)	✓
🔑	departDateTime	datetime	☐
	arrivalDateTime	datetime	✓
🔑	cabinClassCode	varchar(5)	☐
	flightFare	float	✓

3.2 LAU ZHEN XIAN (TP038767)

```sql
--check to ensure the departureDateTime must before arrivalDateTime
ALTER TABLE Flight
ADD CHECK (DATEDIFF(MINUTE, Flight.departDateTime, Flight.arrivalDateTime) >
0);
```

CHECK constraint in the above diagram ensure the time between the departure datetime and arrival datetime is always positive using the DATEDIFF (interval, startDatetime, endDatetime). The code above will return the minutes of the difference between the arrival datetime minus and the depart datetime. If the return value is negative, an error will be invoked and it indicates the flight arrived before it departure, which is invalid.

3.3 FOONG WING YAN (TP038550)

```sql
--email need to be in [email_name]@[domain_name].com format
ALTER TABLE Passenger
ADD CHECK (email LIKE '%@%.com');
```

CHECK constraint above is used to validate the email that is provided by the user. It ensure the email is in the format of [email_name]@[domain_name].com through LIKE operator combine with Wildcard. If the email input from the user does not fulfil the condition, the input will be aborted and require another new input from the user.

CHAPTER 4.0
TRIGGER

4.1 LUA HAU ZHENG (TP038936)

```sql
Create Trigger setTripOptions
On Flight_Reservation
After Insert
As
Begin
    If ((Select Count(*) From Flight_Reservation
         Inner Join inserted
            On Flight_Reservation.reservationID = inserted.reservationID) = 1)
        Begin
            UPDATE Reservation Set tripOptions='One-Way' Where Reservation.reservationID=(Select reservationID From inserted)
        End
    ElSE IF (
        (Select Count(*) From Flight_Reservation Inner Join inserted On Flight_Reservation.reservationID=inserted.reservationID) = 2
    AND
    (
        (Select location.destinationAirport From
            (
            Select Flight.flightID, Flight.destinationAirport, Flight.originAirport
            From  Flight_Reservation Inner Join Flight On Flight_Reservation.flightID = Flight.flightID
            WHERE Flight_Reservation.reservationID = (Select reservationID From inserted)
            )as location
        Order by location.flightID asc
        Offset 0 rows
        Fetch Next 1 rows only
    )
    =
    (Select location.originAirport
        From
            (
            Select Flight.flightID, Flight.destinationAirport,Flight.originAirport
            From Flight_Reservation Inner Join Flight On Flight_Reservation.flightID=Flight.flightID
            WHERE Flight_Reservation.reservationID = (Select reservationID From inserted)
            )as location
        Order by location.flightID asc
        Offset 1 rows
        Fetch Next 1 rows only
    )
    )
    )
        Begin
            UPDATE Reservation Set tripOptions ='Round-trip' Where Reservation.reservationID = (Select reservationID From inserted)
        End
    ELSE
        Begin
            UPDATE Reservation Set tripOptions='Multi-city' Where Reservation.reservationID = (Select reservationID From inserted)
        End
```

```
End
```

The above trigger will add trip options into the tripOptions column in Reservation table based on the total number of flights that is booked by the user. The concept behind this trigger is find the total count that a user booked in one reservation.

For the total count found in one reservation is equal to one, it means the flight is a one-way flight, thus it will update the tripOptions column in Reservation table to “One-Way”. However, to verify the trip option is “Round-Trip”, it needs to fulfil two requirements, which are exact two flight needs to be found in one reservation, and the departure location of the first flight is the same as the arrival location of the second flight. “Multi Trip” is assigned if it is neither “One-Way” nor “Round-Trip”.

4.2 LAU ZHEN XIAN (TP038767)

```
create trigger setMealType
on Flight
After insert
as
begin

    declare @duration int;
    set @duration =  DATEDIFF(HOUR,(Select arrivalDateTime from inserted),
(Select departDateTime From inserted));
    if ( @duration <= 1)
        UPDATE Flight Set mealType = 'Refreshment' Where flightID = (Select
flightID From inserted)
    else if ( @duration <=3)
        UPDATE Flight Set mealType = 'Single Meal' Where flightID = (Select
flightID From inserted)
    else
        UPDATE Flight Set mealType = 'Multi Meal' Where flightID = (Select f
lightID From inserted)
end
```

The above trigger will add meal type into the mealType column in Flight table based on the duration of the flight. DATEDIFF (interval, startDatetime, endDatetime) is used to compare the hour difference between departure date and arrival date. The type of meal served based on the flight’s duration is displayed in the table below.

Duration	Meal Type
Below 1 hour	Refreshment
1 to 3 hours	Single Meal
Above 3 hours	Multi Meal

4.3 FOONG WING YAN (TP038550)

```
Create Trigger setAgeCategory
On Passenger
After Insert
As
```

```
Begin
    Declare @age Int;
    SET @age = DATEDIFF(YEAR,(Select dateOfBirth From inserted),GETDATE());
    IF (@age <= 2)
        UPDATE Passenger Set ageCategory = 'Infant' Where passport= (Select passport From inserted)
    ELSE IF (@age <= 11)
        UPDATE Passenger Set ageCategory='Child' Where passport=(Select passport From inserted)
    ELSE IF (@age <= 17)
        UPDATE Passenger Set ageCategory='Youth' Where passport=(Select passport From inserted)
    ELSE IF (@age <= 64)
        UPDATE Passenger Set ageCategory='Adult' Where passport=(Select passport From inserted)
    ELSE
        UPDATE Passenger Set ageCategory='Senior' Where passport=(Select passport From inserted)
End
```

The above trigger will add data into the ageCategory column in Passenger table by differentiating each passenger into different age category by comparing their date of birth with the current date. DATEDIFF (interval, startDatetime, endDatetime) is a built-in function that use primary to find out the gap between two different datetime and GETDATE() will return the current date. The relationship between different age and age category is displayed in the table below.

Age	**Age Category**
Below 2	Infant
2 – 11	Child
12 – 17	Youth
18 – 64	Adult
Above 65	Senior

CHAPTER 5.0
PROCEDURE

5.1 LUA HAU ZHENG (TP038936)

```sql
create procedure ViewTotalMeal
@AirlineID varchar(5)
as
select Meal.airlineID, Meal.mealName, Count(Reservation_Meal.mealID) AS totalNumber
from Reservation_Meal
inner join Meal on Reservation_Meal.mealID = Meal.mealID
where Meal.airlineID = @AirlineID
group by  rollup (Meal.airlineID, Meal.mealName)
order by totalNumber asc
```

5.2 LAU ZHEN XIAN (TP038767)

```sql
create procedure SearchForFlight
@TravelOrigin varchar(255),
@TravelDestination varchar(255),
@DepartDateTime datetime
As
Select *
from
(
    select Airline.airline,Flight.flightID,
        (select Airport_City.cityAcronym from Airport_City where Airport_City.airportAcronym = Flight.originAirport) as originCity,
        (select Airport_City.cityAcronym from Airport_City where Airport_City.airportAcronym = Flight.destinationAirport) as destinationCity
        from Flight
        inner join Aircraft
        on Flight.aircraftID = Aircraft.aircraftID
        inner join Airline
        on Aircraft.airlineID = Airline.airlineID
        group by Airline.airline, Flight.flightID, Flight.originAirport, Flight.destinationAirport
) as TravelLocation
inner join Flight
on Flight.flightID = TravelLocation.flightID
where TravelLocation.originCity = @TravelOrigin
  AND TravelLocation.destinationCity = @TravelDestination
  AND Flight.departDateTime = @DepartDateTime;
```

5.3 FOONG WING YAN (TP038550)

```sql
create procedure RevenueForEachClass
@Airline varchar(255)
as
select Flight.*, Aircraft_CabinClass.quantity,(Flight.flightFare * Aircraft_CabinClass.quantity) as expectedRevenue
```

```sql
from Flight
inner join Aircraft_CabinClass
on Flight.aircraftID = Aircraft_CabinClass.aircraftID
inner join Aircraft
on Aircraft_CabinClass.aircraftID = Aircraft.aircraftID
where Flight.cabinClassCode = Aircraft_CabinClass.cabinClassCode AND Aircraft.airlineID = @Airline
```

CHAPTER 6.0
QUERY

6.1 FOONG WING YAN (TP038550)

i. Create a query which shows direct flights only for given dates, source& destination.

```
select Flight.aircraftID,Flight.flightID,Flight.departDateTime,Flight.arrivalDateTime,Flight.originAirport,Flight.destinationAirport
from Flight
inner join Flight_Reservation
on Flight.flightID = Flight_Reservation.flightID
inner join Reservation
on Flight_Reservation.reservationID = Reservation.reservationID AND Reservation.totalFare = Flight.flightFare
where Reservation.tripOptions = 'Direct-Way'
  AND Flight.departDateTime = '2000-02-03 08:00:00.000'
  AND Flight.originAirport = 'WMKK'
  AND Flight.destinationAirport = 'WBKW'
```

ii. Create a query which shows aircraft code, class code, and expected revenue for each class code, along with the total revenue of each aircraft for a given airline in a single journey.

```
select Flight.*, Aircraft_CabinClass.quantity,(Flight.flightFare * Aircraft_CabinClass.quantity) as expectedRevenue
from Flight
inner join Aircraft_CabinClass
on Flight.aircraftID = Aircraft_CabinClass.aircraftID
inner join Aircraft
on Aircraft_CabinClass.aircraftID = Aircraft.aircraftID
where Flight.cabinClassCode = Aircraft_CabinClass.cabinClassCode AND Aircraft.airlineID = 'AXM'
```

iii. Create a query which shows all passenger numbers with their corresponding descriptions of reservation status for a specific airline.

```
select Reservation.reservationStatus, count(Distinct Passenger.passport) as NumberOfPassenger
from Passenger
inner join Reservation_Passenger
on Passenger.passport = Reservation_Passenger.passport
inner join Reservation
on Reservation_Passenger.reservationID = Reservation.reservationID
inner join Flight_Reservation
on Flight_Reservation.reservationID = Reservation.reservationID
inner join Flight
on Flight.flightID = Flight_Reservation.flightID and Flight_Reservation.cabinClassCode = Flight.cabinClassCode
inner join Aircraft
on Flight.aircraftID = Aircraft.aircraftID
```

```sql
inner join Airline
on Airline.airlineID = Aircraft.airlineID
where Airline.airlineID = 'AXM'
group by Reservation.reservationStatus
```

iv. Create a query which shows the name of airline that has been most frequently travelled through by the passengers for specified source and destination in given range of dates.

```sql
SELECT Airline.airline,Flight.originAirport, Flight.destinationAirport, Flight_Reservation.departDateTime, Flight.arrivalDateTime, MAX(Flight_Reservation.flightID) AS FlightNumber
FROM Flight_Reservation
LEFT JOIN Flight ON Flight_Reservation.flightID = Flight.flightID
Left join Aircraft ON Flight.aircraftID = Aircraft.aircraftID
Left join Airline ON Aircraft.airlineID = Airline.airlineID
GROUP BY airline, originAirport, destinationAirport, Flight_Reservation.departDateTime, Flight.arrivalDateTime;
```

v. Create a query which provides, for each age category of passengers which the total number of infants, children, youths, adults & seniors travelling through specified flight in a single journey operated by a specified airline in given date.

```sql
SELECT
COALESCE(ageCategory, 'Total Number of People') AS AgeCategory,
Count(ageCategory) AS NumberOfAgeCategory
FROM
Passenger
GROUP BY ROLLUP (ageCategory);
```

vi. Create a query which shows the airline name offering maximum number of journey routes along with names of source and destination.

```sql
SELECT Airline.airline,Flight.originAirport, Flight.destinationAirport, MAX(Flight_Reservation.flightID) AS FlightNumber FROM Flight_Reservation
LEFT JOIN Flight ON Flight_Reservation.flightID = Flight.flightID
Left join Aircraft ON Flight.aircraftID = Aircraft.aircraftID
Left join Airline ON Aircraft.airlineID = Airline.airlineID
GROUP BY airline, originAirport, destinationAirport;
```

vii. Develop one additional query of your own which provides information that would be useful for the business.

```sql
SELECT
firstName,
lastName,
CASE
WHEN totalFare < 300 THEN 'Low'
WHEN totalFare >= 600 AND totalFare <= 600 THEN 'Average'
WHEN totalFare > 1000 THEN 'High'
END evaluation
FROM
Reservation, Passenger
```

6.2 LUA HAU ZHENG (TP038936)

viii. Create a query which displays flight details, such as, the aircraft code, regular fare, and discounted fare for the first class. A 25% discount is being offered. Label the columns as Aircraft, Regular First-Class fare, and Discounted First Class fare.

```
select * , (flightFare * 0.75) AS discountedFare
from Flight
where cabinClassCode = 'E';
```

ix. Create a query which displays the sorted details of flights to given city code with the least duration flight displayed first.

```
select *
from (select Flight.*, DATEDIFF(HOUR, Flight.departDateTime, Flight.arrivalDateTime) as [Duration (hour)],
        (select Airport_City.cityAcronym from Airport_City where Airport_City.airportAcronym = Flight.originAirport) as originCity,
        (select Airport_City.cityAcronym from Airport_City where Airport_City.airportAcronym = Flight.destinationAirport) as destinationCity
        from Flight)
as Flight_Airport_City
where Flight_Airport_City.originCity='TWU' And Flight_Airport_City.destinationCity='KUL'
order by Flight_Airport_City.[Duration (hour)] asc
```

x. Create a query which displays the types of non-vegetarian meals offered on flights.

```
select distinct Flight.flightID, Meal.mealName
from Meal
INNER JOIN Aircraft
ON Meal.airlineID = Aircraft.airlineID
inner join Flight
on Aircraft.aircraftID = Flight.aircraftID
where (Flight.mealType = 'Single Meal' or Flight.mealType = 'Multi Meal') AND (Meal.isRefreshment = '0' AND Meal.isVegetarian = '0');
```

xi. Create a query which shows the names of countries to which TSI provides flight reservations. Ensure that duplicate country names are eliminated from the list.

```
select distinct country
from City
inner join Airport_City
    on City.cityAcronym = Airport_City.cityAcronym
```

```sql
inner join Flight
    on Airport_City.airportAcronym = Flight.originAirport or Airport_City.airportAcronym = Flight.destinationAirport;
```

xii. Create a query which provides, for each airline, the following information: The total number of flights scheduled in a given date. Result should contain both detailed breakup & summary for flights for each airline along with overall summary. Hint: you may wish to use rollup or cubes statements with a query. Some marks will be awarded for the query structure, even if you cannot generate the totals.

```sql
DECLARE @d DATETIME = '2000-01-01 00:00:000';
DECLARE @a DATETIME ='2000-01-01 23:59:59.999';

SELECT a.airline, a.aircraftID ,a.departDateTime, Count (*) AS totalFlightInOneDay
FROM   (SELECT DISTINCT Flight.flightID,
                        Flight.aircraftID,
                        Flight.departDateTime,
                        Airline.airline
        FROM   Flight
               INNER JOIN Aircraft
                       ON Flight.aircraftid = Aircraft.aircraftid
               INNER JOIN Airline
                       ON Aircraft.airlineid = Airline.airlineid
        WHERE  ( Flight.departdatetime >= @d
                 AND Flight.arrivaldatetime <= @a ))AS a
GROUP  BY rollup ( a.airline, a.aircraftID ,a.departDateTime)
```

xiii. Create a query which shows the names of the meal options available on the given airline.

```sql
select *
from Meal
where airlineID = 'AXM';
```

xiv. Develop one additional query of your own which provides information that would be useful for the business. Marks will be awarded depending on the technical skills shown and the relevance of the query.

```sql
select Meal.airlineID, Meal.mealName, Count(Reservation_Meal.mealID) AS totalNumber
from Reservation_Meal
inner join Meal on Reservation_Meal.mealID = Meal.mealID
group by  rollup (Meal.airlineID, Meal.mealName)
order by totalNumber asc
```

6.3 LAU ZHEN XIAN (TP038767)

xv. Create a query which shows the minimum, maximum, and average journey hours for flights to given city code. Display column headings as, Minimum duration, Maximum duration, and Average duration respectively.

```
select *
from (
    select min(DATEDIFF(HOUR, Flight.departDateTime, Flight.arrivalDateTime)
) as MinimumHour,
           max(DATEDIFF(HOUR, Flight.departDateTime, Flight.arrivalDateTime)
) as MaximumHour,
           avg(DATEDIFF(HOUR, Flight.departDateTime, Flight.arrivalDateTime)
) as AverageHour,
           (select Airport_City.cityAcronym from Airport_City where Airport_
City.airportAcronym = Flight.originAirport) as originCity,
           (select Airport_City.cityAcronym from Airport_City where Airport_
City.airportAcronym = Flight.destinationAirport) as destinationCity
    from Flight
    group by Flight.originAirport, Flight.destinationAirport
) as P
where P.originCity='KUL'
  And P.destinationCity='TWU';
```

xvi. Create a query which shows the journey date, number of booked seats, and class name for given passenger.

```
select count(Passenger.passport) as NumberofBookedSeats,
       Flight_Reservation.cabinClassCode as CabinClassCode,
       Flight_Reservation.departDateTime as departDateTime
from Flight_Reservation
inner join Reservation_Passenger
on Flight_Reservation.reservationID = Reservation_Passenger.reservationID
inner join Passenger
on Passenger.passport = Reservation_Passenger.passport
where Passenger.passport = '111111-11-1111'
group by Flight_Reservation.departDateTime, Flight_Reservation.cabinClassCod
e
```

xvii. Create a query which shows the names of meals not requested by any passenger.

```
Select Meal.mealID, Meal.mealName as 'Not Requested Meal'
From Meal
Where Not Exists (Select Reservation_Meal.mealID From Reservation_Meal where
Reservation_Meal.mealID=Meal.mealID)
```

xviii. Create a query which shows the details of passengers booked through a specified airline in a given date for multi-city flights.

```
select Passenger.passport,
       Passenger.firstName,
       Passenger.lastName,
       Passenger.dateOfBirth,
       Passenger.phoneNumber,
       Passenger.email
from Passenger
inner join Reservation_Passenger
on Passenger.passport = Reservation_Passenger.passport
inner join Reservation
on Reservation_Passenger.reservationID = Reservation.reservationID
inner join Flight_Reservation
on Reservation.reservationID = Flight_Reservation.reservationID
inner join Flight
on Flight_Reservation.flightID = Flight.flightID and Flight_Reservation.cabinClassCode = Flight.cabinClassCode
inner join Aircraft
on Flight.aircraftID = Aircraft.aircraftID
inner join Airline
on Aircraft.airlineID = Airline.airlineID
where Airline.airlineID = 'AXM'
  AND Flight.departDateTime = '2000-01-01 12:00:00.000'
  AND Reservation.tripOptions = 'Multi-City'
```

xix. Create a query which provides, for each airline, the following information: The total number of unaccompanied children travelling in a given date. Result should contain both detailed breakup &summary for unaccompanied children for each airline along with overall summary. Hint: you may wish to use rollup or cube statements with a query. Some marks will be awarded for the query structure, even if you cannot generate the totals.

```
select Distinct Reservation.reservationID ,Passenger.passport, SpecialService.specialServiceName, count(Passenger.ageCategory) as NumberofUnaccompaniedChildren
from Passenger
inner join Reservation_Passenger
on Passenger.passport = Reservation_Passenger.passport
inner join Reservation
on Reservation_Passenger.reservationID = Reservation.reservationID
inner join Reservation_SpecialService
on Reservation.reservationID = Reservation_SpecialService.reservationID
inner join SpecialService
on Reservation_SpecialService.specialServiceID = SpecialService.specialServiceID
inner join Flight_Reservation
on Reservation.reservationID = Flight_Reservation.reservationID
inner join Flight
on Flight_Reservation.flightID = Flight.flightID and Flight_Reservation.cabinClassCode = Flight.cabinClassCode
where Flight.departDateTime = '2000-01-01 12:00:00.000'
  AND Passenger.ageCategory = 'Child'
  AND Reservation_SpecialService.specialServiceID = 'AXM-UMC'
group by rollup( Reservation.reservationID, Passenger.passport,SpecialService.specialServiceName,Passenger.ageCategory)
```

xx. Create a query which shows the details of passengers who have availed any extra services for a given flight on specified date.

```
select distinct Passenger.passport,
       Passenger.firstName,
       Passenger.lastName,
       Passenger.dateOfBirth,
```

```sql
       Passenger.phoneNumber,
       Passenger.email,
       SpecialService.specialServiceName
from Passenger
inner join Reservation_Passenger
on Passenger.passport = Reservation_Passenger.passport
inner join Reservation
on Reservation_Passenger.reservationID = Reservation.reservationID
inner join Reservation_SpecialService
on Reservation.reservationID = Reservation_SpecialService.reservationID
inner join SpecialService
on Reservation_SpecialService.specialServiceID = SpecialService.specialServiceID
inner join Flight_Reservation
on Reservation.reservationID = Flight_Reservation.reservationID
inner join Flight
on Flight_Reservation.flightID = Flight.flightID and Flight_Reservation.cabinClassCode = Flight.cabinClassCode
where Flight.departDateTime = '2000-01-01 14:00:00.000'
  AND Flight.flightID = 'AK9746'
```

xxi. Develop one additional query of your own which provides information that would be useful for the business. Marks will be awarded depending on the technical skills shown and the relevance of the query.

```sql
select Airline.airline, count(Passenger.passport)as NumberOfPassengers
from Passenger
inner join Reservation_Passenger
on Passenger.passport = Reservation_Passenger.passport
inner join Reservation
on Reservation_Passenger.reservationID = Reservation.reservationID
inner join Flight_Reservation
on Reservation.reservationID = Flight_Reservation.reservationID
inner join Flight
on Flight_Reservation.flightID = Flight.flightID and Flight_Reservation.cabinClassCode = Flight.cabinClassCode
inner join Aircraft
on Flight.aircraftID = Aircraft.aircraftID
inner join Airline
on Aircraft.airlineID = Airline.airlineID
group by Airline.airline
```

CHAPTER 7.0
PERSONAL REFLECTION

7.1 LUA HAU ZHENG (TP038936)

During the development of this assignment, I have learnt a lot about database design and the correct way each kind of data. I am honoured to be chosen as a leader to carry out this assignment. I learnt a lot about the responsibility and the way to divide the task based on the group member capability. Sharing, teaching and learning and commutating with my group members is also an important experience that I learnt throughout finishing this assignment. However, some obstruct are met in the process of developing this assignment.

One of the obstruct is the different intake code between team members. This situation causes the difficulty in organising a meeting because the class time are different from each other. Moreover, since the group members are from different intakes, they also have their different subject’s assignments. The meeting is harder to construct especially when the meeting time is near one of the member’s other subject’s assignment due date.

Another difficulty is the lack of transportation between team members. One of our members do not live near anywhere the university or the university hostel. Although the meeting can be conducted online, but it decreases the efficiency of communication thus causing misunderstanding of the database design. In addition, since most of the query require group discussion, the sharing and teaching session between group members become very inefficient and ineffective.

7.2 LAU ZHEN XIAN (TP038767)

At the end of this assignment, I have gotten a huge insight relevant to the database development. Based on the assignment question, the group members will be required to filter out all the vital points that can affect the database design. In facts, this is one of the hardest parts for the database development process as there are many special conditions will be required to read and understand thoroughly. Furthermore, after filtering out the special conditions, we also require separating the conditions to determine whether it should be done by Trigger or by manually insert the data. Apart from that, when doing the query section, it also provides a huge insight for me towards the SQL. This is because I had very least experience in writing the query especially when the software engineering cost is mostly focus on software development language like JAVA and C++.

During working on the assignment, I have spent a lot of efforts in discussion with the group leader – Lua Hau Zheng and another group member – Foong Wing Yan, on many aspects such as designing the database, entities that should be included in the system, primary key and foreign key for each table so that data can be obtained in correct way as well as how to draw the ERD diagram correctly. In addition, we also held many meetings at university even though we do not have class at the day. Besides, while working on the queries in the individual part, we also ask opinions from each other to ensure we have understood the question correctly. Upon completing the 7 queries, we will check with other members so that the answers are checked and agreed by all the members.

7.3 FOONG WING YAN (TP038550)

In fulfilment of the requirement for this assignment, the database designer has come to acknowledge the significance of database life cycle. In light of this, it is evident that this assignment was not easy mission to complete as the challenging tasks the database designer encountered were on implementing the query which is SQL Server language and explain the query and results in the group part which generated by Microsoft SQL Server Management Studio. In the initial stage, the database designer faced many computational errors due to the SQL Server language was indeed fairly new to the database designer. Thus, requiring more time and in-depth studying through lab tutorials and so on.

Throughout the duration of this assignment, the database designer had faced strugglers and triumphs in attempts for completing the task of the implementing the queries for this database system. Then, database designer start searches on the web in W3School to learnt more about the SQL tutorials. From those searching, database designer able to understand more the statements and the functions. Therefore, the database designer has gained insights on how statement or functions is conducted and how it can be applied into query. Apart from that, gaining a new skill on how to operate Microsoft SQL Server Management Studio and in essence how to create SQL language such as query has opened up the database designer’s mind to a possible career venture into database administrator in the near future.

Having reached to the end of this assignment, the database designer can proudly say that new knowledge and skills have most definitely been gained not only in relation to database but also on a personal note too. Other than that, extra skill that the database designer had gain is Teamwork skill which is the most important skill gain in the group part because teamwork skill able to make the database system and assignment successful. On a personal reflection, the most prominent takeaway was the database designer’s determination to continuously attempt to solve the compilations and computational errors that surfaced. The database designer learned the importance of trial-and-errors in helping ensure the aim of the assignment is successfully achieved in the end.