

A ■ P ■ U

**ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION**

GROUP ASSIGNMENT

Student Names : Alexander Ho YingHan (TP022858)
Teh Chun Wei (TP027353)
Ahmed I-Mon (TP030577)

Intake Code : UC3F1402SE

Subject : Advanced Database Systems

Project Title : Group Assignment

Lecturer : ABDALLAH S.M. ALNATSA

Submission Date : 30th May 2013

1 TABLE OF CONTENTS

2	Introduction.....	5
3	Assumptions.....	6
4	Entity Relationship Modelling.....	7
4.1	Entity Relationship Diagram.....	7
4.2	Data Table Structure.....	8
4.3	Normalization.....	12
4.3.1	UNF.....	12
4.3.2	1NF	12
4.3.3	2NF	13
4.3.4	3NF	14
5	Optimization Strategy	15
5.1	De-Normalization.....	15
5.2	Duplicating None-key Attributes in One to Many Relationship.....	17
5.3	SQL Optimization Technique	19
5.4	Proper Indexing in Table Columns	20
6	Constraints and Triggers	21
6.1	Constraints.....	21
6.1.1	Primary Key	21
6.1.2	Foreign Key	23
6.1.3	NOT NULL.....	24
6.2	Triggers	25
6.2.1	Reservation Cancellation	25
6.2.2	Flight Delay	27
6.2.3	Reschedule Reservation	28

6.3	Stored Procedure	30
6.3.1	Insert Passenger and Meal Information to a New Temporary Table	30
6.3.2	Customer Info Enquiry via Passport Number	31
6.3.3	Insert New Reservation	32
7	SQL Queries.....	33
7.1	Member 1: Alexander Ho YingHan	33
7.1.1	Query (i).....	33
7.1.2	Queries (ii)	34
7.1.3	Queries (iii)	35
7.1.4	Queries (iv)	36
7.1.5	Queries (v)	37
7.1.6	Queries (vi)	38
7.1.7	Queries (vii)	39
7.2	Member 2: Teh Chun Wei.....	40
7.2.1	Query (viii).....	40
7.2.2	Query (ix).....	41
7.2.3	Query (x).....	42
7.2.4	Query (xi).....	43
7.2.5	Query (xii).....	44
7.2.6	Query (xiii).....	45
7.2.7	Query (xiv).....	46
7.3	Member 3: Ahmed I-Mon	47
7.3.1	Query (xv).....	47
7.3.2	Query (xvi).....	48
7.3.3	Query (xvii).....	49

7.3.4	Query (xviii).....	50
7.3.5	Query (xix).....	51
7.3.6	Query (xx).....	52
7.3.7	Query (xxi).....	53
8	Personal Reflection Report	54
8.1	Alexander Ho YingHan TP022858	54
8.2	Teh Chun Wei (TP027353).....	55
8.3	Ahmed I-Mon (TP030577).....	56
9	Workload Matrix.....	57
10	References	58

2 INTRODUCTION

Travel Safe International (TSI) is one of the leading companies in the business of providing global distribution systems regarding selling tickets for multiple airlines. TSI basically provide main features that required by booking system such as flight booking, customer registration and flight rescheduling. With some requirements given, a database is required to create for storing different information related to flight booking, customer registration and others. Since rules and regulation of each perspective airline need to be taken into account, queries will be designed to fulfill the business requirements and provide additional functions to maintain TSI for long term operations.

Entity Relationship model will be created to show all the entity in a clear view by listing down all the attributes and relationship with other entities. Constraints, stored procedure and triggers will be created in this database system to fulfill the business requirement and maintain the database in a logical and structural manner. Other than that, optimization strategies will also be implemented to enhance the database performance.

3 ASSUMPTIONS

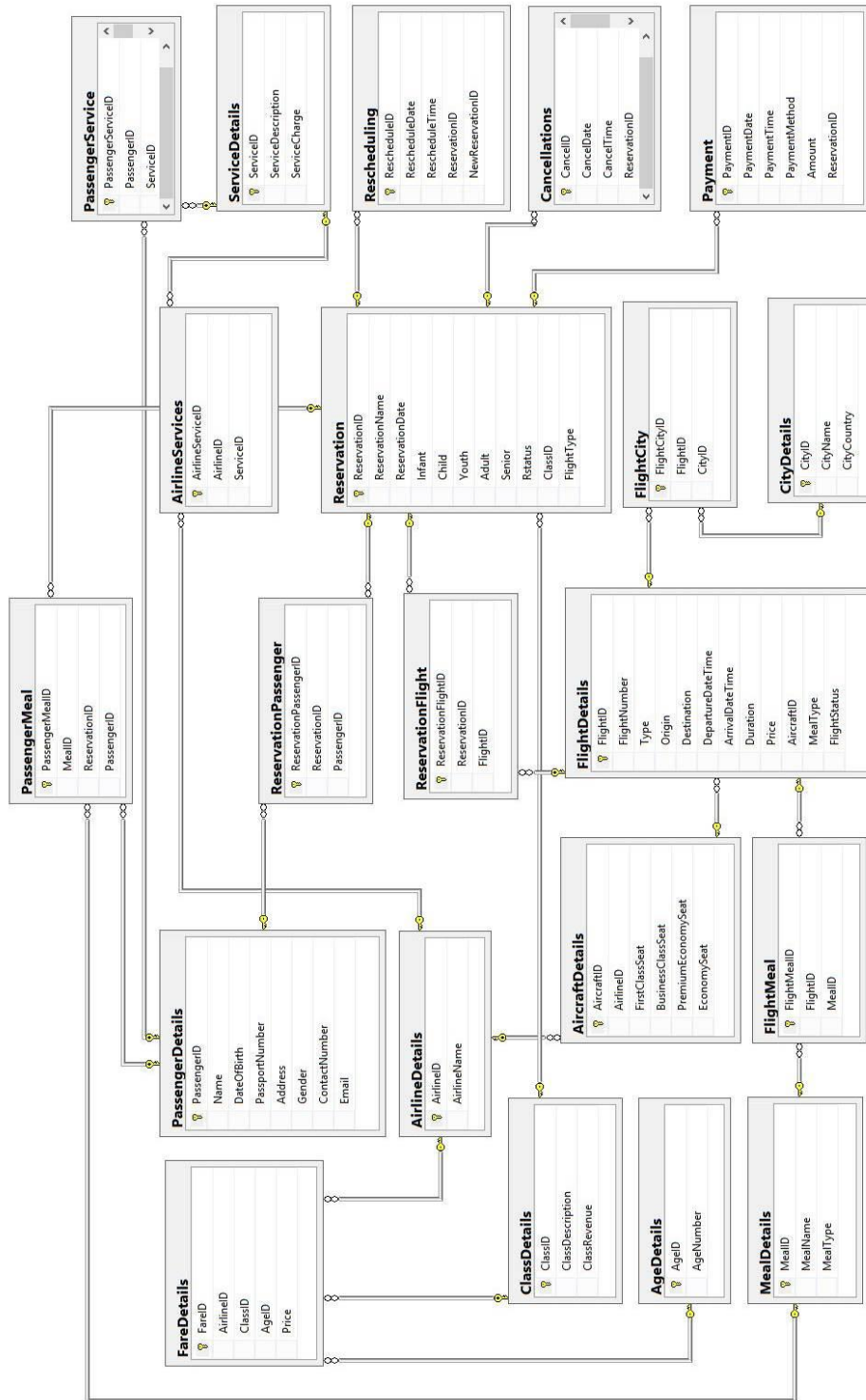
Basic business requirements are given for constructing a database but not clearly described. Numbers of assumptions had brought up to make the database in a more structural and maintainable view. Assumption brought up are listed as below such as.

- Some airlines limit only 4 passengers per booking.
- Most of the airline provide several services such as unaccompanied minor services, wheel chair or child care.
- Each flight serve different meal plan based on the duration of flight
- Every aircraft have different number of seat based on the class type.

Every booking of flight can differentiate into 3 category which is One-way that reach destination directly, round trip that allow passengers to depart and return in 1 booking, and last option which is multi-city that let flight transit and change flight on the specific destination

4 ENTITY RELATIONSHIP MODELLING

4.1 ENTITY RELATIONSHIP DIAGRAM



4.2 DATA TABLE STRUCTURE

Table Name	Attribute Name	Data Type	Allow NULL	Primary Key	Foreign Key
AgeDetails	AgeID	varchar(50)	No	Yes	No
	AgeNumber	int	No	No	No
AircraftDetails	AircraftID	nvarchar(20)	No	Yes	No
	AirlineID	nvarchar(20)	No	No	Yes
	FirstClassSeat	int	No	No	No
	BusinessClassSeat	int	No	No	No
	PremiumEconomySeat	int	No	No	No
	EconomySeat	int	No	No	No
AirlineDetails	AirlineID	nvarchar(20)	No	Yes	No
	AirlineName	varchar(50)	No	No	No
AirlineServices	AirlineServiceID	nvarchar(50)	No	Yes	No
	AirlineID	nvarchar(20)	No	No	Yes
	ServiceID	nvarchar(50)	No	No	Yes
Cancellations	CancelID	nvarchar(50)	No	Yes	No
	CancelDate	date	No	No	No
	ReservationID	nvarchar(50)	No	No	Yes
CityDetails	CityID	varchar(50)	No	Yes	No
	CityName	varchar(50)	No	No	No
	CityCountry	varchar(50)	No	No	No
ClassDetails	ClassID	varchar(50)	No	Yes	No
	ClassDescription	varchar(50)	No	No	No
	ClassRevenue	int	No	No	No
FareDetails	FareID	nchar(10)	No	Yes	NO
	AirlineID	nvarchar(20)	No	No	Yes
	ClassID	varchar(50)	No	No	Yes
	AgeID	varchar(50)	No	No	Yes
	Price	decimal(18,0)	No	No	No

FlightDetails	FlightID	nvarchar(50)	No	Yes	No
	FlightNunmber	nvarchar(50)	No	No	No
	Type	varchar(50)	No	No	No
	Origin	nvarchar(50)	No	No	No
	Destination	nvarchar(50)	No	No	No
	DepartureDateTime	datetime	No	No	No
	ArrivalDateTime	datetime	No	No	No
	Duration	datetime	No	No	No
	Price	decimal(18,0)	No	No	No
	AircraftID	nvarchar(20)	No	No	Yes
	MealType	varchar(50)	No	No	No
	FlightStatus	varchar(50)	No	No	No
FlightCity	FlightCityID	nvarchar(50)	No	YES	NO
	FlightID	nvarchar(50)	No	No	Yes
	CityID	varchar(50)	No	No	Yes
FlightMeal	FlightMealID	nvarchar(50)	No	Yes	No
	FlightID	nvarchar(50)	No	No	Yes
	MealID	nvarchar(50)	No	No	Yes
MealDetails	MealID	nvarchar(50)	No	Yes	No
	MealName	varchar(50)	No	No	No
	MealType	varchar(50)	No	No	No
PassengerDetails	PassengerID	nvarchar(50)	No	Yes	No
	Name	varchar(50)	No	No	No
	DateOfBirth	date	No	No	No
	PassportNumber	nvarchar(50)	No	No	No
	Address	nvarchar(max)	No	No	No
	Gender	varchar(50)	No	No	No
	ContactNumber	numeric(18,0)	No	No	No
	Email	nvarchar(50)	No	No	No

PassengerMeal	PassengerMealID	nvarchar(50)	No	Yes	No
	MealID	nvarchar(50)	No	No	YES
	ReservationID	nvarchar(50)	No	No	Yes
	PassengerID	nvarchar(50)	No	No	Yes
PassengerService	PassengerServiceID	nvarchar(50)	No	No	Yes
	PassengerID	nvarchar(50)	No	No	Yes
	ServiceID	nvarchar(50)	No	No	Yes
Payment	PaymentID	nvarchar(50)	No	Yes	No
	PaymentDate	date	No	No	No
	PaymentTime	time(7)	No	No	No
	PaymentMethod	varchar(50)	No	No	No
	Amount	decimal(18,0)	No	No	No
	ReservationID	nvarchar(50)	No	No	Yes
Rescheduling	RescheduleID	nvarchar(50)	No	Yes	No
	RescheduleDate	date	No	No	No
	RescheduleTime	time(7)	No	No	No
	ReservationID	nvarchar(50)	No	No	Yes
	NewReservationID	nvarchar(50)	No	No	No
Reservation	ReservationID	nvarchar(50)	No	Yes	No
	ReservationName	varchar(50)	No	No	No
	ReservationDate	date	No	No	No
	Infant	int	No	No	No
	Child	int	No	No	No
	Youth	int	No	No	No
	Adult	int	No	No	No
	Senior	int	No	No	No
	RStatus	varchar(50)	No	No	No
	ClassID	varchar(50)	No	No	Yes
	FlightType	varchar(50)	No	No	No

ReservationPassenger	ReservationPassengerID	nvarchar(50)	No	Yes	No
	ReservationID	nvarchar(50)	No	No	Yes
	PassengerID	nvarchar(50)	No	No	Yes
ReserveFlight	ReservationFlightID	nvarchar(50)	No	Yes	No
	ReservationID	nvarchar(50)	No	No	Yes
	FlightID	nvarchar(50)	No	No	Yes
ServiceDetails	ServiceID	nvarchar(50)	No	Yes	No
	ServiceDescription	varchar(50)	No	No	No
	ServiceCharge	decimal(18,0)	No	No	No

Table 1: Structure of Data Tables

4.3 NORMALIZATION

4.3.1 UNF

AirlineID, AirlineName, AircraftID, FirstClassSeat, BusinessClassSeat, PremiumEconomySeat, EconomySeat, FareID, ClassID, AgeID, AgeNumber, FlightID, FlightNumber, Type, Origin, Destination, DepartureDateTime, ArrivalDateTime, Duration, MealType, Price, FlightMealID, MealID, MealName, PassengerMealID, ReservationID, PassengerID, ClassDescription, ClassRevenue, ReservationName, ReservationDate, Infant, Child, Youth, Adult, Senior, RStatus, FlightType, ReservationPassengerID, Name, DateOfBirth, PassportNumber, Address, Gender, ContactNumber, Email, ServiceID, ServiceDescription, ServiceCharge, ReservationFlightID, CityID, CityName, CityCountry, PaymentID, PaymentDate, PaymentTime, PaymentMethod, Amount, AirlineServiceID, RescheduleID, RescheduleDate, RescheduleTime, NewReservationID, CancelID, CancelDate, CancelTime, FlightStatus, FlightCityID

4.3.2 1NF

AirlineID, FareID, AgeID, AircraftID, FlightMealID, FlightID, ClassID, ReservationID, ReservationPassengerID, MealID, PassengerMealID, PassengerID, ReserveFlight, PaymentID, CityID, ServiceID, AirlineServiceID, RescheduleID, CancelID, FlightCityID	AirlineName, FirstClassSeat, BusinessClassSeat, PremiumEconomySeat, EconomySeat, AgeNumber, FlightNumber, Type, Origin, Destination, DepartureDateTime, ArrivalDateTime, Duration, MealType, Price, MealName, ClassDescription, ClassRevenue, ReservationName, ReservationDate, Infant, Child, Youth, Adult, Senior, RStatus, Type, Name, DateOfBirth, PassportNumber, Address, Gender, ContactNumber, Email, ServiceDescription, ServiceCharge, CityName, CityCountry, PaymentDate, PaymentTime, PaymentMethod, Amount, RescheduleDate, RescheduleTime, NewReservationID, CancelDate, CancelTime,
---	---

	FlightStatus
--	--------------

4.3.3 2NF

AirlineID	AirlineName
AgeID	AgeNumber
AirlineID, ServiceID	AirlineServiceID
AircraftID, AirlineID	FirstClassSeat, BusinessClassSeat, PremiumEconomySeat, EconomySeat
AirlineID, ClassID, AgeID	FareID, Price
FlightID, MealID	FlightMealID
FlightID, CityID	FlightCityID
FlightID, AircraftID	FlightNumber, Type, Origin, Destination, DepartureDateTime, ArrivalDateTime, Duration, MealType, Price, FlightStatus
MealID	MealName, MealType
ClassID	ClassDescription, ClassRevenue
RID, ClassCode	RName, RDate, Infant, Child, Youth, Adult, Senior, RStatus. FlightType
MealCode, RID, PID	PassengerMealCode
RID, PID	ReservationPassengerCode
PID, ServiceCode	PName, DOB, PassportNo, Address, Gender, ContactNo, Email
ServiceCode	ServiceDesc, ServiceCharge
RID, FlightCode	ReserveFlight
RID, PaymentID	PaymentDate, PaymentTime, PaymentMethod, Amount
RescheduleID, RID	RescheduleDate, RescheduleTime, NewRID
CancelID, RID	CancelDate, CancelTime

4.3.4 3NF

FareID	Price
FlightNumber	Type, Origin, Destination, DepartureDateTime, ArrivalDateTime, Duration, MealType, Price

5 OPTIMIZATION STRATEGY

5.1 DE-NORMALIZATION

De-normalization is a process of attempting to optimize the read performance of a database by adding redundant or by grouping data. It is used to combine several database data and generate a new group view of data. Below will show example of de-normalization such as combining two Many-to-many relationship tables. For example:

- ❖ Many Passenger can choose many PassengerMeal
- ❖ PassengerMeal can have more than one Meal

Below shows the many-to-many relationship diagram, in order to get access for data from Passenger and Meal. An inner join statement is considered.

PassengerDetails	PassengerMeal	MealDetails
PassengerID (PK)	PassengerMealID(PK)	MealID(PK)
Name	MealID(FK)	MealName
DateOfBirth	ReservationID(FK)	MealType
PassportNumber	PassengerID(FK)	
Address		
Gender		
ContactNumber		
Email		

Table 2: Many to Many Relationship Diagram

PassengerDetail*
PassengerID(PK)
Name
DateOfBirth
PassportNumber
Address
Gender
ContactNumber
Email
MealName
MealType

Table 3: De-normalization Diagram

From above table, when the de-normalization is applied where both detail of Passenger and Meal will be merged together as shown in table 2. The benefits of using the de-normalization are improved performance, the need for fewer joins, and the ability to maintain history information. It allows staff of aircraft easier to arrange food before flight and distribute the food on plane. It can also be used as a reference when there are some mistakes on food distribution to avoid any misunderstanding. In other view on this system, de-normalization enables the means to maintain function of history look-up on previous transaction and it will helps the system to be easier to manage.

5.2 DUPLICATING NONE-KEY ATTRIBUTES IN ONE TO MANY RELATIONSHIP

A one-to-many relationship is two table that share common attributes that perform a relationship. A relationship should be a parent table that involves an attribute from the child entity.

Based on TSI, the Passenger table should include the ServiceID as foreign key in PassengerDetails table. The information from Service table, have to use join statement to obtain data through the parent table.

PassengerDetails	ServiceDetails
PassengerID (PK)	ServiceID (PK)
ServiceID	ServiceDescription
Name	ServiceCharge
DateOfBirth	
PassportNumber	
Address	
Gender	
ContactNumber	
Email	

Table 4: One to Many Relationship

PassengerDetails
PassengerID (PK)
Name
DateOfBirth
PassportNumber
Address
Gender
ContactNumber
Email
ServiceDescription
ServiceCharge

Table 5: Avoid Duplication of key

Using above table 4, it reduces the JOIN statements used in database which will enhance the database performance greatly and through the simplicity of the query statement. It saves more database space and the simplicity of the system can be learnt by any staff in a short amount of time.

5.3 SQL OPTIMIZATION TECHNIQUE

SQL statements basically are used to obtain data from database. Several way of writing queries that can obtain the same results. It is important to know that the performance is not fastest due to common mistaken done by developer which the way to write the query is incorrect. The list below will show how to increase the performance which that can be optimized.

1. Avoid full table scan - SQL query does not include primary key or index which the full scan occur.

Example:

```
SELECT PassengerID, DateOfBirth, Name from Passenger where Name =  
'Lim Hong Jun'
```

5.4 PROPER INDEXING IN TABLE COLUMNS

Proper indexing on table column will greatly optimize the database performance on executing the queries. The most effective ways for indexing is assigning primary key to each and every table that can be found in the database. Thus, data can be retrieve rapidly on the sorting operations owing to the tables are physically sorted based on its primary key field.

Index should only be assign to related tables where information will be share among the tables. For example, FlightDetails table will be using specific information such as ReservationFlightID, AircraftID, FlightCityID and FlightMealID from related tables in order to perform various business operations.

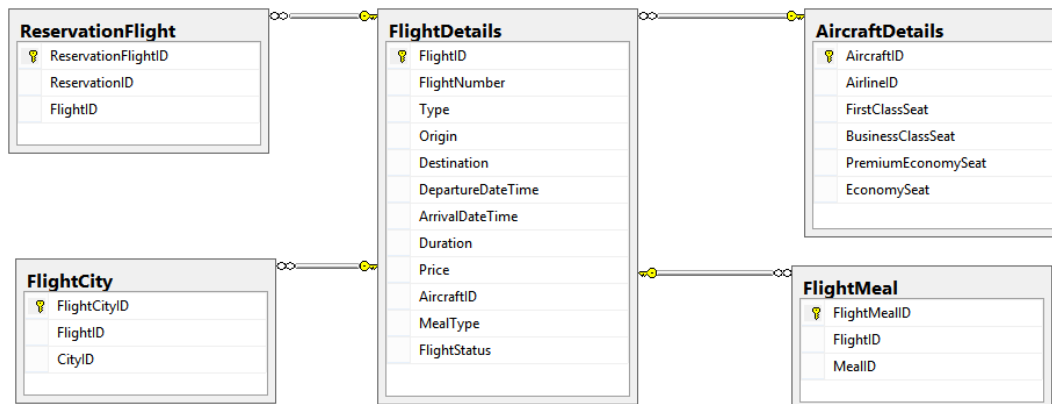


Figure 2: Proper indexing in table columns

Although creating indexes may enhance the overall performance of the database but side effect may occurs if there are too many indexes. Developers may have to inner join all related table in order to execute a simple query. Thus, it would be real menace when the database size is huge.

6 CONSTRAINTS AND TRIGGERS

6.1 CONSTRAINTS

Constraints allow user to enforce rules in database. These rules may affect business logic, database integrity and table structure. There are several constraint to be discussed.

6.1.1 Primary Key

Primary key constraint is Not Null and is considered a Unique Constraint in the sense that it must contain data inside and does not repeat the same value. It helps the user to easily and quickly identify a specific data in a table.

```
--/Flight Details Table
Create Table FlightDetails
(
    FlightID nvarchar (20) not null PRIMARY KEY,
    FlightNumber nvarchar (50) not null,
    Type varchar(50) not null,
    Origin nvarchar (50) not null,
    Destination nvarchar (50) not null,
    DepartureDateTime datetime not null,
    ArrivalDateTime datetime not null,
    Duration datetime not null,
    Price decimal not null,
    AircraftID nvarchar (20) not null FOREIGN KEY REFERENCES AircraftDetails(AircraftID),
    MealType varchar (50) not null,
    FlightStatus varchar (50) not null,
)
```

Figure 3: Primary Key

Above figure shows, the primary key is set in FlightDetails table. This FlightID will have foreign key references in other tables which will use it to access the FlightDetails table to obtain data. An example would be that a different table would be able to access a specific row of data in FlightDetails via the unique primary key, ensuring that only a single type of result will be listed since there would be no duplicate primary key fields.

6.1.2 Foreign Key

Foreign key constraint is to ensure the referential integrity of the data in one table to match values in another table (W3Schools, 2014). Foreign key itself is a primary in other table that store their own data.

```
--/Flight Details Table
Create Table FlightDetails
(
    FlightID nvarchar (20) not null PRIMARY KEY,
    FlightNumber nvarchar (50) not null,
    Type varchar(50) not null,
    Origin nvarchar (50) not null,
    Destination nvarchar (50) not null,
    DepartureDateTime datetime not null,
    ArrivalDateTime datetime not null,
    Duration datetime not null,
    Price decimal not null,
    AircraftID nvarchar (20) not null FOREIGN KEY REFERENCES AircraftDetails(AircraftID),
    MealType varchar (50) not null,
    FlightStatus varchar (50) not null,
)
```

Figure 4: Foreign Key

The figure above shows a foreign key constraint has been added to the AircraftID in the FlightDetails table, which will link the field to its reference which is the AircraftID primary key inside the AircraftDetails table. This enables functions such as referential integrity where the DBMS is able to check whether a foreign key column in a table contains the same data as the ones found in the primary key column in the original table. For this example shown here, AircraftID is used to determine which aircraft will be used for the created flight.

6.1.3 NOT NULL

NOT NULL constraints are used to enforce a rule that a column must always contain valid and applicable data- in short, that a column may not contain NULL. (Codeidol.com, 2012) It will automatically alert developers if the database found that there's an empty column haven't filled. Adopting NOT NULL constraints will not just help a developer to maintain a database with increased ease, it prevents events such as the developer forgetting to input necessary data as it will generate an alert to developer. It also prevents errors such as linking of data that will lead to no result show as NULL foreign key. Thus, most of the time NOT NULL constraints are needed to avoid unnecessary errors.

```
--/Flight Details Table
Create Table FlightDetails
(
    FlightID nvarchar (20) not null PRIMARY KEY,
    FlightNumber nvarchar (50) not null,
    Type varchar(50) not null,
    Origin nvarchar (50) not null,
    Destination nvarchar (50) not null,
    DepartureDateTime datetime not null,
    ArrivalDateTime datetime not null,
    Duration datetime not null,
    Price decimal not null,
    AircraftID nvarchar (20) not null FOREIGN KEY REFERENCES AircraftDetails(AircraftID),
    MealType varchar (50) not null,
    FlightStatus varchar (50) not null,
)
```

Figure 3: NOT NULL statements

The query above shows the usage of NOT NULL in FlightDetails table. This is to ensure that all the column data are filled so that other tables and queries can refer to the required data.

6.2 TRIGGERS

6.2.1 Reservation Cancellation

```

CREATE TRIGGER Cancel_Reservation
on Reservation
instead of delete
as

    declare @Reservation_ID as nvarchar(50)
    declare @Flight_Schedule as datetime
    declare @Cancel as nvarchar(50)
    declare @CancelNumber as integer
    set @Cancel = (select top 1 CancelID from Cancellations order by CancelID DESC)
    set @Reservation_ID = (select ReservationID from deleted)
    set @Flight_Schedule =
        (select f.DepartureDateTime from FlightDetails f
         inner join ReservationFlight rf on rf.FlightID = f.FlightID
         inner join Reservation r on rf.ReservationID = r.ReservationID where r.ReservationID = @Reservation_ID)
    set @CancelNumber = SUBSTRING(@Cancel,4,8)

if(select DATEDIFF(hour, CURRENT_TIMESTAMP, @Flight_Schedule))>3
begin
    update Reservation
    SET RStatus = 'Invalid'
    Where @Reservation_ID = ReservationID

    insert into Cancellations(CancelID, CancelDate, CancelTime, ReservationID)
    values('CAN'+ (CAST(@CancelNumber + 1 as nvarchar)), (CAST(GETDATE()as DATE)), (CAST(GETDATE()as time)),@Reservation_ID)

    print 'Reservation ID : ' + @Reservation_ID
    print 'Flight on : ' + Convert(varchar, @Flight_Schedule)
    print 'Reservation has been Cancelled'
    print 'Fine Charges : N/A'
End
else
Begin
    update Reservation
    SET RStatus = 'Invalid'
    Where @Reservation_ID = ReservationID

    insert into Cancellations(CancelID, CancelDate, CancelTime, ReservationID)
    values('CAN'+ (CAST(@CancelNumber + 1 as nvarchar)), (CAST(GETDATE()as DATE)), (CAST(GETDATE()as time)),@Reservation_ID)

    print 'Reservation ID : ' + @Reservation_ID
    print 'Flight on : ' + Convert(varchar, @Flight_Schedule)
    print 'Reservation has been Cancelled'
    print 'Fine Charges : RM 200'
End

--- Show the Result ---

Select * from Reservation where ReservationID = @Reservation_ID
Select * from Cancellations where ReservationID = @Reservation_ID
Go

```

Figure 4: Reservation Cancellation Trigger

Output:

Results Messages											
	ReservationID	ReservationName	ReservationDate	Infant	Child	Youth	Adult	Senior	Rstatus	ClassID	FlightType
1	R00001	Alexander	2014-05-01	1	0	0	1	0	Invalid	C0003	One Way

	CancelID	CancelDate	CancelTime	ReservationID
1	CAN00001	2014-05-02	12:00:00.0000000	R00001
2	CAN2	2014-05-30	01:14:59.2630000	R00001

Figure 7: Result of Reservation Cancellation (1)

Results Messages	
(1 row(s) affected)	
(1 row(s) affected)	
Reservation ID : R00001	
Flight on : May 10 2014 10:00AM	
Reservation has been Cancelled	
Fine Charges : RM 200	
(1 row(s) affected)	
(2 row(s) affected)	
(1 row(s) affected)	

Figure 5: Result of Reservation Cancellation (2)

The Reservation Cancellation Trigger was implemented to prevent incorrect data update on reservation records. When a reservation is cancelled by user, the status in reservation table will be modified to 'Invalid' to show that the particular reservation is unavailable. At same time, the deleted record will generate in Cancellations table to prove that the user is no longer reserving this particular ticket. Other than that, according to the business strategy, those people who cancelled their reservation 3 hours before the flight will cause the ticket to lose its value and penalty charges will be charged. The trigger will perform checking in flight schedule to confirm the date is matched to reservation ID selected. The system will then display messages showing that the data is completely modified.

6.2.2 Flight Delay

```

create trigger Flight_Delay
on FlightDetails
instead of delete
as
begin
    RAISERROR('Flight cant proceed as usual. Flight will be postpone instead.',16,10)
    select * from FlightDetails
    update FlightDetails
    set FlightStatus = 'Postpone'
    from FlightDetails f INNER JOIN Deleted d on f.FlightID = d.FlightID
END
GO

```

Output:

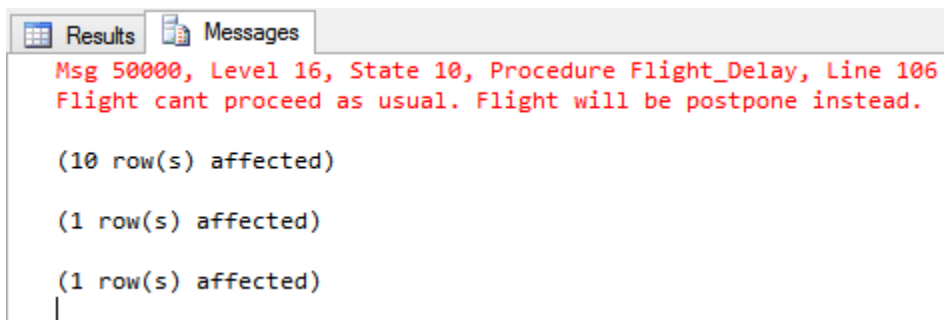


Figure 6: Result of Flight Delay (1)

	FlightID	FlightNumber	Type	Origin	Destination	DepartureDateTime	ArrivalDateTime	Duration	Price	AircraftID	MealType	FlightStatus
1	F00001	MH1238	DIRECT	KUL	HKG	2014-05-10 10:00:00.000	2014-05-10 12:00:00.000	2014-05-28 02:00:00.000	700	A0007	SINGLE-MEAL	POSTPONED
2	F00002	MH2304	TRANSIT	DPS	BRT	2014-05-11 10:00:00.000	2014-05-11 12:00:00.000	2014-05-28 02:00:00.000	600	A0005	MULTI-MEAL	POSTPONED
3	F00003	AK2348	DIRECT	MIA	BKK	2014-05-12 10:00:00.000	2014-05-12 12:00:00.000	2014-05-28 02:00:00.000	1800	A0003	SINGLE-MEAL	ON-GOING
4	F00004	AK8374	DIRECT	MEL	SUL	2014-05-13 10:00:00.000	2014-05-13 12:00:00.000	2014-05-28 02:00:00.000	750	A0008	MULTI-MEAL	ON-GOING
5	F00005	TS9203	DIRECT	PEK	SIN	2014-05-14 10:00:00.000	2014-05-14 12:00:00.000	2014-05-28 02:00:00.000	960	A0002	SINGLE-MEAL	ON-GOING
6	F00006	TS8849	DIRECT	SIN	PEK	2014-05-15 10:00:00.000	2014-05-15 12:00:00.000	2014-05-28 02:00:00.000	350	A0009	SINGLE-MEAL	ON-GOING
7	F00007	ZE0939	TRANSIT	SUL	MEL	2014-05-16 10:00:00.000	2014-05-16 12:00:00.000	2014-05-28 02:00:00.000	1600	A0004	MULTI-MEAL	ON-GOING
8	F00008	UL2323	TRANSIT	BKK	MIA	2014-05-17 10:00:00.000	2014-05-17 12:00:00.000	2014-05-28 02:00:00.000	850	A0001	SINGLE-MEAL	POSTPONED
9	F00009	UL9929	DIRECT	BRT	DPS	2014-05-18 10:00:00.000	2014-05-18 12:00:00.000	2014-05-28 02:00:00.000	450	A0006	SINGLE-MEAL	ON-GOING
10	F00010	JR7384	TRANSIT	HKG	KUL	2014-05-19 10:00:00.000	2014-05-19 12:00:00.000	2014-05-28 02:00:00.000	660	A0010	MULTI-MEAL	ON-GOING

Figure 7: Result of Flight Delay (2)

This Trigger was implemented to prevent any flight records from being removed accidentally or intentionally. If there any unforeseen circumstances appearing such as Heavy Snowfall or Lightning shower where flight has to be delayed or postponed, this delete command will take action to updating the status from on-going to postpone instead. This allows staff and passengers to quickly identify the status of certain flights.

6.2.3 Reschedule Reservation

```

CREATE TRIGGER reschedule_reservation
ON Rescheduling
AFTER INSERT
AS
IF EXISTS(SELECT RStatus from Reservation where RStatus = 'Reissued')
BEGIN
    PRINT 'The Reservation has reissued, cannot reissue again.'
    rollback
END
ELSE IF EXISTS(SELECT RStatus from Reservation where RStatus = 'Invalid')
BEGIN
    PRINT 'The Reservation ID is Invalid!'
    rollback
END
ELSE
    DECLARE @newReservationID nvarchar(50)
    DECLARE @oldReservationID nvarchar(50)
    DECLARE @newDate date
    DECLARE @newRName varchar(50)
    DECLARE @newInfant int
    DECLARE @newChild int
    DECLARE @newYouth int
    DECLARE @newAdult int
    DECLARE @newSenior int
    DECLARE @newClassID varchar(50)
    DECLARE @newFlightType varchar(50)
    BEGIN
        SET @newReservationID = (SELECT NewReservationID FROM inserted)
        SET @oldReservationID = (SELECT ReservationID FROM inserted)
        SET @newDate = (SELECT RescheduleDate from inserted)
        SET @newRName = (SELECT ReservationName FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newInfant = (SELECT Infant FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newChild = (SELECT Child FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newYouth = (SELECT Youth FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newAdult = (SELECT Adult FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newSenior = (SELECT Senior FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newClassID = (SELECT ClassID FROM Reservation WHERE ReservationID = @oldReservationID)
        SET @newFlightType = (SELECT FlightType FROM Reservation WHERE ReservationID = @oldReservationID)

        PRINT 'The reservation has been successfully rescheduled. New Reservation ID is ' + @newReservationID

        INSERT INTO Reservation
        VALUES (@newReservationID, @newRName, @newDate, @newInfant, @newChild, @newYouth, @newAdult, @newSenior, 'Booked', @newClassID, @newFlightType)

        UPDATE Reservation
        SET RStatus = 'Reissued'
        WHERE ReservationID = @oldReservationID
        UPDATE ReservationPassenger
        SET ReservationID = @newReservationID
        WHERE ReservationID = @oldReservationID
    END
END

```

Figure 8: Trigger of Reschedule Reservation (1)

The trigger that is implanted in `reschedule_reservation` was designed to meet the business requirements which is to reschedule the Reservation ID if any passengers want to made changes to his/her reservation. The trigger will be triggered as soon as the data is inserted to the “Rescheduling” table. The trigger will simply check whether the reservation status linked to the “ReservationID” given by the passenger has been reissued. If the Reservation status is indicates “Reissued” or “Invalid”, the data would not be inserted into the database, instead, a rollback will be performed and an error message will be displayed.

In opposite, if the Reservation status indicates “Booked”, the data will be inserted into “Rescheduling” table, and it will update the existing reservation status to “Reissued” based on the “ReservationID” provided by passenger. After that, a new ReservationID will be generate and inserted to “ReservationDetails” table as well as update “ReservationPassenger” table to ensure the new “ReservationID” is attach with the “PassengerID” that passengers provide. In the meantime, the existing “ReservationID” that has been reissued will become invalid.

6.3 STORED PROCEDURE

6.3.1 Insert Passenger and Meal Information to a New Temporary Table

```
create procedure Passenger_Meal
@p_id nvarchar(50), @p_Name varchar(50), @Meal_Detail varchar(50)
as
begin
declare @temp as table
(PassengeReservationID nvarchar(50), PassengerName varchar(50), MealName varchar(50))
insert into @temp(PassengeReservationID, PassengerName, MealName) values (@p_id, @p_Name, @Meal_Detail)
end
go
```

Figure 9: Passenger_Meal Procedure

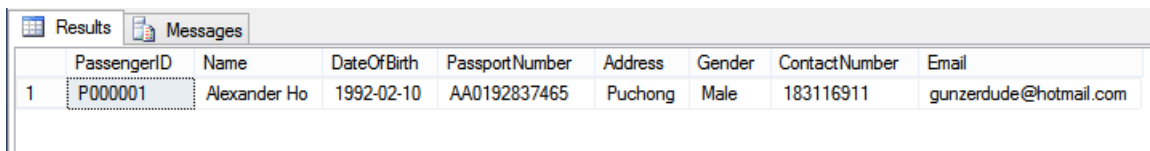
Passenger_Meal procedure is used to insert necessary record from Passenger and Meal table as new record and store data into a temporary table which is used to show out the necessary output. It is used to show which passengers had ordered what meal in a specific flight.

6.3.2 Customer Info Enquiry via Passport Number

```
-- Create proc PassportNumber| @PassportNumber nvarchar(15)
as
-- select *
from PassengerDetails
where PassportNumber = @PassportNumber

exec PassportNumber 'AA0192837465'
```

Output:



	PassengerID	Name	DateOfBirth	PassportNumber	Address	Gender	ContactNumber	Email
1	P000001	Alexander Ho	1992-02-10	AA0192837465	Puchong	Male	183116911	gunzerdude@hotmail.com

Figure 11: Results of customer info enquiry via passport number

The customer enquiry via passport was implemented to enable staffs to quickly acquire customer information via passport number. Since Passport numbers can uniquely identify a person so it can be used as a secondary primary key. Basic customer information will be request to input during registration for ticket reservation for differentiation process purposes in the system. The system will verify the input passport number and search the passenger database for this particular customer.

6.3.3 Insert New Reservation

```

CREATE Procedure new_reservation
@reserveID nvarchar(50),
@reserveName varchar(50),
@reserveDate date,
@numInfant int,
@numChild int,
@numYouth int,
@numAdult int,
@numSenior int,
@cID varchar(50),
@FType varchar(50),
@passengerReservationID nvarchar(50),
@passengerName varchar(50),
@birthDate date,
@PassNo nvarchar(50),
@pAddress nvarchar(255),
@pGender varchar(50),
@contact numeric(18,0),
@email nvarchar(50),
@rpID nvarchar(50),
@rfID nvarchar(50),
@fID nvarchar(50),
@payID nvarchar(50),
@payDate date,
@payTime time(7),
@payMethod varchar(50),
@payAmount decimal(18,0)
AS
BEGIN
    DECLARE @totalSeat int
    SET
        @totalSeat = @numInfant+@numChild+@numYouth+@numAdult+@numSenior
    PRINT 'Total Number of Seat: ' + convert (varchar,@totalSeat)

    IF @totalSeat > 4
        BEGIN
            RAISERROR ('Maximum of 4 passenger(s) per booking', 16,10)
        END
    ELSE
        BEGIN
            INSERT INTO Reservation
            (ReservationID, ReservationName, ReservationDate, Infant, Child, Youth, Adult, Senior, RStatus, ClassID, FlightType)
            values
            (@reserveID, @reserveName, @reserveDate, @numInfant, @numChild, @numYouth, @numAdult, @numSenior, 'Booked', @cID, @FType)
            INSERT INTO PassengerDetails
            (PassengerID, Name, DateOfBirth, PassportNumber, Address, Gender, ContactNumber, Email)
            values
            (@passengerReservationID, @passengerName, @birthDate, @PassNo, @pAddress, @pGender, @contact, @email)
            Insert INTO ReservationPassenger
            (ReservationPassengerID, ReservationID, PassengerID)
            values
            (@rpID, @reserveID, @passengerReservationID)
            INSERT INTO ReservationFlight
            (ReservationFlightID, ReservationID, FlightID)
            values
            (@rfID, @reserveID, @fID)
            INSERT INTO Payment
            (PaymentID, PaymentDate, PaymentTime, PaymentMethod, Amount, ReservationID)
            values
            (@payID, @payDate, @payTime, @payMethod, @payAmount, @reserveID)
        END
    END

```

Figure 12: Stored Procedure new_reservation (3)

The stored procedure for new_reservation will be used to insert a new reservation record into “Reservation” table, “PassengerDetails” table, “ReservationPassenger” table, “ReservationFlight” table, and “Payment” table. Before insertion to related tables, the stored procedure will check the total number of passenger to ensure the number of passenger is not more than 4 passengers. If this limit is exceeded, the new record would not be inserted into tables and an error messaged will be displayed. On the flipside, when number of passengers is less than the limit, the record will be stored into related tables.

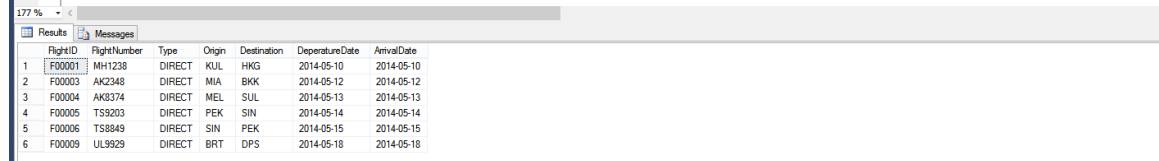
7 SQL QUERIES

7.1 MEMBER 1: ALEXANDER HO YINGHAN

7.1.1 Query (i)

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

```
--No.i
select FlightID,FlightNumber,Type,Origin,Destination, (CAST(DepartureDateTime as Date))as DeperatureDate,
(CAST(ArrivalDateTime as Date))as ArrivalDate
from FlightDetails
where Type='Direct' |
```



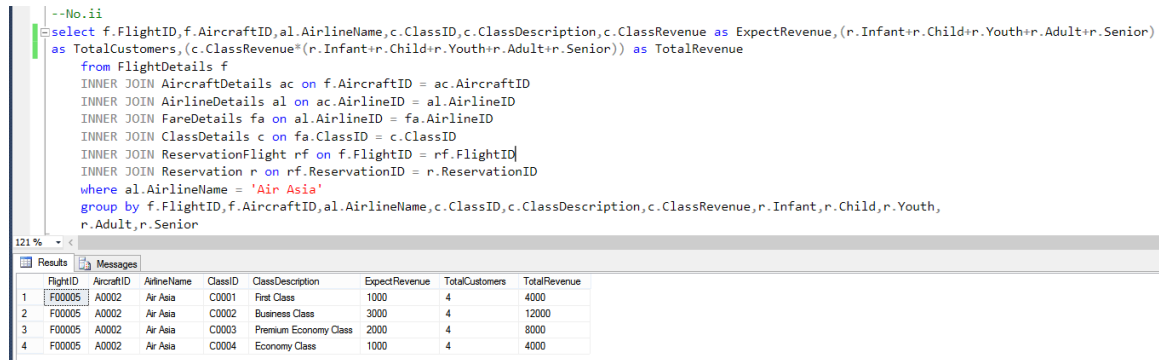
	FlightID	FlightNumber	Type	Origin	Destination	DeperatureDate	ArrivalDate
1	F00001	MH1238	DIRECT	KUL	HKG	2014-05-10	2014-05-10
2	F00003	AK2348	DIRECT	MIA	BKK	2014-05-12	2014-05-12
3	F00004	AK8374	DIRECT	MEL	SUL	2014-05-13	2014-05-13
4	F00005	TS9203	DIRECT	PEK	SIN	2014-05-14	2014-05-14
5	F00006	TS8849	DIRECT	SIN	PEK	2014-05-15	2014-05-15
6	F00009	UL9929	DIRECT	BRT	DPS	2014-05-18	2014-05-18

Figure 13: Solution and result of query (i)

This query will show all flight information regarding direct flight.

7.1.2 Queries (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.



The screenshot shows a SQL query in a text editor and its results in a table. The query is as follows:

```
--No.ii
select f.FlightID,f.AircraftID,al.AirlineName,c.ClassID,c.ClassDescription,c.ClassRevenue as ExpectRevenue,(r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
as TotalCustomers,(c.ClassRevenue*(r.Infant+r.Child+r.Youth+r.Adult+r.Senior)) as TotalRevenue
from FlightDetails f
INNER JOIN AircraftDetails ac on f.AircraftID = ac.AircraftID
INNER JOIN AirlineDetails al on ac.AirlineID = al.AirlineID
INNER JOIN FareDetails fa on al.AirlineID = fa.AirlineID
INNER JOIN ClassDetails c on fa.ClassID = c.ClassID
INNER JOIN ReservationFlight rf on f.FlightID = rf.FlightID
INNER JOIN Reservation r on rf.ReservationID = r.ReservationID
where al.AirlineName = 'Air Asia'
group by f.FlightID,f.AircraftID,al.AirlineName,c.ClassID,c.ClassDescription,c.ClassRevenue,r.Infant,r.Child,r.Youth,
r.Adult,r.Senior
```

The results table shows the following data:

	FlightID	AircraftID	AirlineName	ClassID	ClassDescription	ExpectRevenue	TotalCustomers	TotalRevenue
1	F00005	A0002	Air Asia	C0001	First Class	1000	4	4000
2	F00005	A0002	Air Asia	C0002	Business Class	3000	4	12000
3	F00005	A0002	Air Asia	C0003	Premium Economy Class	2000	4	8000
4	F00005	A0002	Air Asia	C0004	Economy Class	1000	4	4000

Figure 14: Solution and result of query (ii)

Expect Revenue for First Class is 10000, Business Class is 8000, Premium Economy Class is 5000 and Economy Class is 3000. The results will be varied and the total revenue will be based on how many total customers purchase the ticket.

7.1.3 Queries (iii)

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

```
--No.iii
select p.PassengerID,r.*,al.AirlineName
from PassengerDetails p
INNER JOIN ReservationPassenger rp on p.PassengerID = rp.PassengerID
INNER JOIN Reservation r on rp.ReservationID = r.ReservationID
INNER JOIN ReservationFlight rf on r.ReservationID = rf.ReservationID
INNER JOIN FlightDetails f on rf.FlightID = f.FlightID
INNER JOIN AircraftDetails ac on f.AircraftID = ac.AircraftID
INNER JOIN AirlineDetails al on ac.AirlineID = al.AirlineID
```

	PassengerID	ReservationID	ReservationName	ReservationDate	Infant	Child	Youth	Adult	Senior	Rstatus	ClassID	FlightType	AirlineName
1	P000001	R00001	Alexander	2014-05-01	1	0	0	1	0	Invalid	C0003	One Way	Malaysia Airline System
2	P000002	R00002	Jun Wei	2014-05-02	0	1	1	1	1	Booked	C0004	One Way	Cathay Airlines
3	P000003	R00003	Ahmed I-mon	2014-05-03	1	0	1	2	0	Booked	C0002	Multi Way	Air Asia
4	P000006	R00004	Alicia	2014-05-04	0	3	0	1	0	Booked	C0003	Multi Way	Cathay Airlines
5	P000007	R00005	Stephanie	2014-05-05	2	0	1	1	0	Booked	C0002	One Way	Singapore Airlines
6	P000009	R00006	Rebecca	2014-05-06	0	2	0	2	0	Booked	C0001	Multi Way	Malaysia Airline System
7	P000004	R00007	Simon	2014-05-07	1	0	0	1	1	Booked	C0002	One Way	Eva Airlines
8	P000008	R00008	Chin Chooi	2014-05-08	0	1	1	2	0	Booked	C0004	Multi Way	Singapore Airlines

Figure 15: Solution and result of query (iii)

This query will allow user to look for all reservation information regarding what airline company chosen, how many passengers, types of flight and the reservation status.

7.1.4 Queries (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.

```
--No.iv
Select count(rf.FlightID) as NoOfTravelled,al.AirlineName, f.Origin, f.Destination,
(CAST(f.DepartureDateTime as Date))as DepartureDate,(CAST(f.ArrivalDateTime as Date))as ArrivalDate
from ReservationFlight rf
INNER JOIN FlightDetails f on rf.FlightID = f.FlightID
INNER JOIN AircraftDetails ac on f.AircraftID = ac.AircraftID
INNER JOIN AirlineDetails al on ac.AirlineID = al.AirlineID
group by rf.FlightID,al.AirlineName,f.Origin,f.Destination,f.DepartureDateTime,f.ArrivalDateTime |
```

	NoOfTravelled	AirlineName	Origin	Destination	DepartureDate	ArrivalDate
1	1	Air Asia	PEK	SIN	2014-05-14	2014-05-14
2	1	Cathay Airlines	MIA	BKK	2014-05-12	2014-05-12
3	1	Cathay Airlines	SUL	MEL	2014-05-16	2014-05-16
4	1	Eva Airlines	BRT	DPS	2014-05-18	2014-05-18
5	1	Malaysia Airline System	KUL	HKG	2014-05-10	2014-05-10
6	1	Malaysia Airline System	MEL	SUL	2014-05-13	2014-05-13
7	1	Singapore Airlines	SIN	PEK	2014-05-15	2014-05-15
8	1	Singapore Airlines	HKG	KUL	2014-05-19	2014-05-19

Figure 16: Solution and result of query (iv)

This query will show user regarding the number of travelled for specific airline company for specific journey routes.

7.1.5 Queries (v)

Create a query which provides, for each age category of passengers, the following information:

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. Result should contain both detailed breakup & summary for above mentioned categories 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.

```
--No.v
select r.ReservationID,f.FlightID, (Cast(f.DepartureDateTime as Date))as DepatureDate, (Cast(f.ArrivalDateTime as Date))as ArrivalDate,
f.Origin,f.Destination, sum(r.Infant) as Infant, sum(r.Child) as Child, sum(r.Youth) as Youth, sum(r.Adult) as Adults,
sum(r.Senior) as Seniors
from Reservation r
INNER JOIN ReservationFlight rf on r.ReservationID = rf.ReservationID
INNER JOIN FlightDetails f on rf.FlightID = f.FlightID
group by cube (r.ReservationID,f.FlightID,f.DepartureDateTime,f.ArrivalDateTime,f.Origin,f.Destination)
```

121 %

Results

Messages

	ReservationID	FlightID	DepatureDate	ArrvalDate	Origin	Destination	Infant	Child	Youth	Adults	Seniors
40	NULL	NULL	NULL	2014-05-14	PEK	SIN	1	0	1	2	0
41	NULL	NULL	NULL	NULL	PEK	SIN	1	0	1	2	0
42	NULL	NULL	NULL	NULL	NULL	SIN	1	0	1	2	0
43	R00006	F00004	2014-05-13	2014-05-13	MEL	SUL	0	2	0	2	0
44	NULL	F00004	2014-05-13	2014-05-13	MEL	SUL	0	2	0	2	0
45	NULL	NULL	2014-05-13	2014-05-13	MEL	SUL	0	2	0	2	0
46	NULL	NULL	NULL	2014-05-13	MEL	SUL	0	2	0	2	0
47	NULL	NULL	NULL	NULL	MEL	SUL	0	2	0	2	0
48	NULL	NULL	NULL	NULL	NULL	SUL	0	2	0	2	0
49	NULL	NULL	NULL	NULL	NULL	NULL	5	7	4	11	2

Figure 17: Solution and results of query (v)

As it can be seen that the third highlighted row is the sum of first and second highlighted row.

7.1.6 Queries (vi)

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

```
--No.vi
select count(f.AircraftID) as MaximumNoOfJourney, al.AirlineName, f.Origin,f.Destination
from FlightDetails f
INNER JOIN AircraftDetails ac on f.AircraftID = ac.AircraftID
INNER JOIN AirlineDetails al on ac.AirlineID = al.AirlineID
group by f.AircraftID,al.AirlineName,f.Origin,f.Destination
```

	MaximumNoOfJourney	AirlineName	Origin	Destination
1	1	Air Asia	BKK	MIA
2	1	Air Asia	PEK	SIN
3	1	Cathay Airlines	MIA	BKK
4	1	Cathay Airlines	SUL	MEL
5	1	Eva Airlines	DPS	BRT
6	1	Eva Airlines	BRT	DPS
7	1	Malaysia Airline System	KUL	HKG
8	1	Malaysia Airline System	MEL	SUL
9	1	Singapore Airlines	SIN	PEK
10	1	Singapore Airlines	HKG	KUL

Figure 18: Solution and results of query (vi)

This query will show the user regarding the maximum number of journey which are offered by several airline company with specific journey routes.

7.1.7 Queries (vii)

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.

```
--No.vii
select f.FlightNumber,f.AircraftID,al.AirlineName,r.ClassID,c.ClassDescription, sum(r.Infant) as InfantBookedSeat,
sum(r.Child) as ChildBookedSeat, sum(r.Youth) as YouthBookedSeat, sum(r.Adult) as AdultBookedSeat,
sum(r.Senior) as SeniorBookedSeat, (r.Infant+r.Child+r.Youth+r.Adult+r.Senior) as TotalBookedSeat,
CASE when r.ClassID='C0001' THEN ac.FirstClassSeat - (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
when r.ClassID='C0002' then ac.BusinessClassSeat - (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
when r.ClassID='C0003' then ac.PremiumEconomySeat - (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
when r.ClassID='C0004' then ac.EconomySeat - (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
else 0 END as [Remaining Seats]
from FlightDetails f
inner join ReservationFlight rf on f.FlightID = rf.FlightID
inner join Reservation r on rf.ReservationID = r.ReservationID
inner join ClassDetails c on r.ClassID = c.ClassID
inner join AircraftDetails ac on f.AircraftID = ac.AircraftID
inner join AirlineDetails al on ac.AirlineID = al.AirlineID
group by f.FlightNumber,f.AircraftID,al.AirlineName,r.ClassID,c.ClassDescription,r.Infant,r.Child,r.Youth,r.Adult,r.Senior,
ac.FirstClassSeat,ac.BusinessClassSeat,ac.PremiumEconomySeat,ac.EconomySeat
```

	FlightNumber	AircraftID	AirlineName	ClassID	ClassDescription	InfantBookedSeat	ChildBookedSeat	YouthBookedSeat	AdultBookedSeat	SeniorBookedSeat	TotalBookedSeat	Remaining Seats
1	AK2348	A0003	Cathay Airlines	C0003	Premium Economy Class	0	3	0	1	0	4	96
2	AK8374	A0008	Malaysia Airline System	C0001	First Class	0	2	0	2	0	4	46
3	JR7384	A0010	Singapore Airlines	C0004	Economy Class	0	1	1	2	0	4	96
4	MH1238	A0007	Malaysia Airline System	C0003	Premium Economy Class	1	0	0	1	0	2	98
5	TS8849	A0009	Singapore Airlines	C0002	Business Class	2	0	1	1	0	4	46
6	TS9203	A0002	Air Asia	C0002	Business Class	1	0	1	2	0	4	46
7	UL9929	A0006	Eva Airlines	C0002	Business Class	1	0	0	1	1	3	47
8	ZE0939	A0004	Cathay Airlines	C0004	Economy Class	0	1	1	1	1	4	96

Figure 19: Solution and results of query (vii)

For the own design queries part, I have decided to show the remaining seats for specific flight where information including InfantBookedSeat, ChildBookedSeat and etc will be shown. The actual seats sizes for all aircrafts are set at as follows; first class 50, business class 50, premium economy seat 100 and economy 100. The results for remaining seats column will be based on the subtraction of total booked seat for all age category.

7.2 MEMBER 2: TEH CHUN WEI

7.2.1 Query (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 column as Aircraft, Regular First Class fare, and Discounted First Class fare.

Solution:

```
--viii--
SELECT f.FlightID, a.AircraftID, fa.AgeID AS Category, fa.Price as CategoryPrice, f.Price,
fa.Price+f.Price AS Regular_First_Class_fare, (f.Price+fa.Price)*0.75 AS Discounted_First_Class_fare
FROM FlightDetails f INNER JOIN AircraftDetails a
ON f.AircraftID = a.AircraftID
INNER JOIN FareDetails fa
ON a.AirlineID=fa.AirlineID
WHERE fa.ClassID='C0001'
ORDER BY FlightID
```

Figure 20: Solution of query (viii)

Output:

	FlightID	AircraftID	Category	CategoryPrice	Price	Regular_First_Class_fare	Discounted_First_Class_fare
1	F00005	A0002	INFANT	0	960	960	720.00
2	F00005	A0002	CHILD	150	960	1110	832.50
3	F00005	A0002	YOUTH	250	960	1210	907.50
4	F00005	A0002	ADULT	350	960	1310	982.50
5	F00005	A0002	SENIOR	250	960	1210	907.50
6	F00008	A0001	INFANT	0	850	850	637.50
7	F00008	A0001	CHILD	150	850	1000	750.00
8	F00008	A0001	YOUTH	250	850	1100	825.00
9	F00008	A0001	ADULT	350	850	1200	900.00
10	F00008	A0001	SENIOR	250	850	1100	825.00

Figure 21: Results of query (viii)

The figure above shows the output of the query (viii) with the class code is “C0001”.

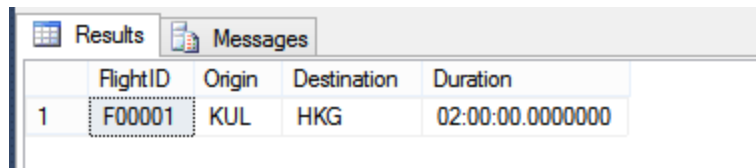
7.2.2 Query (ix)

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

Solution:

```
--ix--  
SELECT f.FlightID, f.Origin ,f.Destination, (CAST(f.Duration as time)) as Duration  
From FlightDetails f  
WHERE Origin = 'KUL'  
Order by Duration;
```

Figure 22: Solution of query (ix)

Output:

The screenshot shows a SQL Server query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

	FlightID	Origin	Destination	Duration
1	F00001	KUL	HKG	02:00:00.0000000

Figure 23: Results of query (ix)

The figure above shows the output of the query (ix) with the given city is “KUL”.

7.2.3 Query (x)

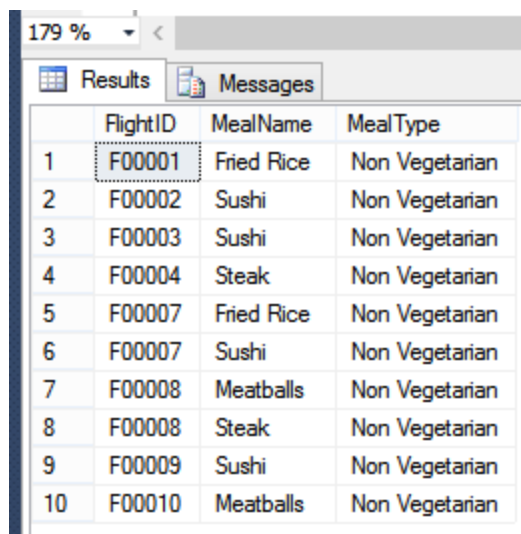
Create a query which displays the types of non-vegetarian meals offered on flights

Solution:

```
--X--  
SELECT f.FlightID, m.MealName, m.MealType  
From MealDetails m  
INNER JOIN FlightMeal fm  
ON m.MealID = fm.MealID  
INNER JOIN FlightDetails f  
ON fm.FlightID = f.FlightID  
WHERE m.MealType = 'Non Vegetarian'  
ORDER BY f.FlightID, MealName
```

Figure 24: Solution of query (x)

Output:



The screenshot shows a database query results window with a zoom level of 179%. The window has two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with four columns: FlightID, MealName, and MealType. The table contains 10 rows of data, sorted by FlightID and then MealName. The first row is highlighted with a dashed border.

	FlightID	MealName	MealType
1	F00001	Fried Rice	Non Vegetarian
2	F00002	Sushi	Non Vegetarian
3	F00003	Sushi	Non Vegetarian
4	F00004	Steak	Non Vegetarian
5	F00007	Fried Rice	Non Vegetarian
6	F00007	Sushi	Non Vegetarian
7	F00008	Meatballs	Non Vegetarian
8	F00008	Steak	Non Vegetarian
9	F00009	Sushi	Non Vegetarian
10	F00010	Meatballs	Non Vegetarian

Figure 25: Results of query (x)

The figure above shows the output of query (x) with the meal type is “Non-Vegetarian” as well as sort by FlightCode and MealName.

7.2.4 Query (xi)

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

Solution:

```
--xi--  
SELECT Distinct CityCountry  
FROM CityDetails
```

Figure 26: Solution of query (xi)

Output:

The screenshot shows a database application window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'CityCountry' and a list of 10 countries. The countries are: Australia, China, Hong Kong, Indonesia, Korea, Malaysia, Singapore, Thailand, United Kingdom, and United States. The first row, 'Australia', is highlighted with a blue background.

	CityCountry
1	Australia
2	China
3	Hong Kong
4	Indonesia
5	Korea
6	Malaysia
7	Singapore
8	Thailand
9	United Kingdom
10	United States

Figure 27: Results of query (xi)

The figure above shows the output of query (xi) with eliminated duplicate country name.

7.2.5 Query (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 cube statements with a query. Some marks will be awarded for the query structure, even if you cannot generate the totals.

Solution:

```
--xii--
SELECT count(f.FlightID) AS TotalNumberOfFlight, air.AirlineName, (CAST(f.DepartureDateTime as date)) as Departure
FROM FlightDetails f
INNER JOIN AircraftDetails a
ON f.AircraftID = a.AircraftID
INNER JOIN AirlineDetails air
ON a.AirlineID = air.AirlineID
Group by rollup(air.AirlineName, (CAST(f.DepartureDateTime as date)))
order by (CAST(f.DepartureDateTime as date))
```

Figure 28: Solution of query (xii)

Output:

Results			
	TotalNumberOfFlight	AirlineName	Departure
1	2	Air Asia	NULL
2	2	Cathay Airlines	NULL
3	2	Eva Airlines	NULL
4	2	Malaysia Airline System	NULL
5	2	Singapore Airlines	NULL
6	10	NULL	NULL
7	1	Malaysia Airline System	2014-05-10
8	1	Eva Airlines	2014-05-11
9	1	Cathay Airlines	2014-05-12
10	1	Malaysia Airline System	2014-05-13
11	1	Air Asia	2014-05-14
12	1	Singapore Airlines	2014-05-15
13	1	Cathay Airlines	2014-05-16
14	1	Air Asia	2014-05-17
15	1	Eva Airlines	2014-05-18
16	1	Singapore Airlines	2014-05-19

Figure 29: Results of query (xii)

The figure above shows the output of query (xii) with the rollup statements. First it shows the total number of flight based on Airline AirlineName and shows the total of flights. Next, it will categorize the number of flights based on the departure date and airline company name.

7.2.6 Query (xiii)

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

Solution:

```
--xiii--  
SELECT DISTINCT al.AirlineID, al.AirlineName, m.MealName, m.MealType  
FROM AirlineDetails al  
INNER JOIN AircraftDetails a  
ON al.AirlineID = a.AirlineID  
INNER JOIN FlightDetails f  
ON a.AircraftID = f.AircraftID  
INNER JOIN FlightMeal fm  
ON f.FlightID = fm.FlightID  
INNER JOIN MealDetails m  
ON fm.MealID = m.MealID  
WHERE al.AirlineID = 'AA'
```

Figure 30: Solution of query (xiii)

Output:



	AirlineID	AirlineName	MealName	MealType
1	AA	Air Asia	Fruit Salad	Vegetarian
2	AA	Air Asia	Meatballs	Non Vegetarian
3	AA	Air Asia	Sandwich	Vegetarian
4	AA	Air Asia	Steak	Non Vegetarian

Figure 31: Results of query (xiii)

The figure above shows the output of query (xiii) with the given airline is “AA”.

7.2.7 Query (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.

Solution:

```
--xiv--
SELECT f.FlightNumber, r.ReservationID as ReservationID, p.Name as Name, p.PassportNumber, s.ServiceDescription
FROM FlightDetails f
INNER JOIN ReservationFlight rf
ON f.FlightID = rf.FlightID
INNER JOIN Reservation r
ON rf.ReservationID = r.ReservationID
INNER JOIN ReservationPassenger rp
ON r.ReservationID = rp.ReservationID
INNER JOIN PassengerDetails p
ON rp.PassengerID = p.PassengerID
INNER JOIN PassengerService ps
ON p.PassengerID = ps.PassengerID
INNER JOIN ServiceDetails s
ON ps.ServiceID = s.ServiceID
WHERE ServiceDescription = 'Wheel Chair'
GROUP BY f.FlightNumber, r.ReservationID, p.Name, p.PassportNumber, s.ServiceDescription
```

Figure 32: Solution of query (xiv)

Output:

Results		Messages			
	FlightNumber	ReservationID	Name	PassportNumber	ServiceDescription
1	JR7384	R00008	Chin Chooi	AA5330241020	Wheel Chair

Figure 33: Results of query (xiv)

The figure above shows the output of query (xiv). It displays which passenger requires Service of “Wheel Chair” according to each flight.

7.3 MEMBER 3: AHMED I-MON

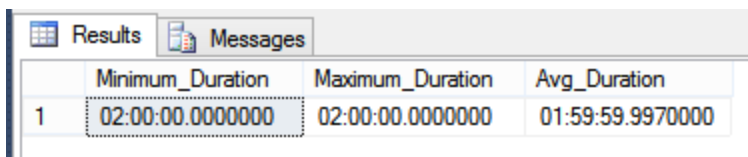
7.3.1 Query (xv)

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.

```
--Question xv

Select MIN(CAST(Duration as time)) as Minimum_Duration,
MAX(CAST(Duration as time)) as Maximum_Duration,
CAST(CAST(AVG(CAST(Duration as float)) as datetime)as time) as Avg_Duration
From FlightDetails
```

Figure 34: Solution of query (xv)



	Minimum_Duration	Maximum_Duration	Avg_Duration
1	02:00:00.0000000	02:00:00.0000000	01:59:59.9970000

Figure 35: Results of query (xv)

This are the result of minimum duration, maximum duration and average duration.

7.3.2 Query (xvi)

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

```
Select (CAST(f.DepartureDateTime as date))as DepartureDate,  
(CAST(f.ArrivalDateTime as date) )as ArrivalDate,  
(r.Infant+r.Child+r.Youth+r.Adult+r.Senior) as TotalSeat,  
c.ClassDescription,  
r.ReservationName  
From Reservation r inner join ReservationFlight rf  
ON r.ReservationID = rf.ReservationID  
inner join FlightDetails f  
ON rf.FlightID = f.FlightID  
inner join ClassDetails c  
On r.ClassID = c.ClassID  
WHERE r.ReservationName = 'Ahmed I-Mon'
```

Figure 36: Solution of query (xvi)

Results		Messages			
	DepartureDate	ArrivalDate	TotalSeat	ClassDescription	ReservationName
1	2014-05-14	2014-05-14	4	Business Class	Ahmed I-mon

Figure 37: Results of query (xvi)

The passenger has reserved a flight on specific date including the total Seat and the class seat.

7.3.3 Query (xvii)

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

```

Select m.MealName
from MealDetails m inner join PassengerMeal pm
on m.MealID = pm.MealID
inner join Reservation r
on pm.ReservationID = r.ReservationID
WHERE m.MealID not in (pm.MealID)

```

Figure 38: Solution of query (xvii)

MealName

Figure 39: Results of query (xvii)

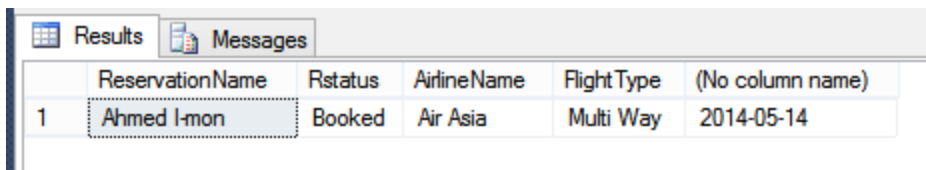
The result should show the MealName of beef, sushi, and salad but due to SQL problems the result cannot be show properly but the way of coding is correct.

7.3.4 Query (xviii)

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

```
SELECT distinct r.ReservationName,  
r.Rstatus,  
air.AirlineName,  
r.FlightType,  
(CAST(f.DepartureDateTime as Date))  
from Reservation r inner join ReservationFlight rf  
on r.ReservationID = rf.ReservationID  
inner join FlightDetails f  
on rf.FlightID = f.FlightID  
inner join AircraftDetails a  
on f.AircraftID = a.AircraftID  
inner join AirlineDetails air  
on a.AirlineID = air.AirlineID  
WHERE r.FlightType = 'Multi Way' and f.DepartureDateTime = '2014-05-14 10:00:00.000'
```

Figure 40: Solution of query (xviii)



	ReservationName	Rstatus	AirlineName	Flight Type	(No column name)
1	Ahmed I-mon	Booked	Air Asia	Multi Way	2014-05-14

Figure 41: Results of query (xviii)

The result show the by searching specific flight type data 'multi-city' and exact date for the passenger booked.

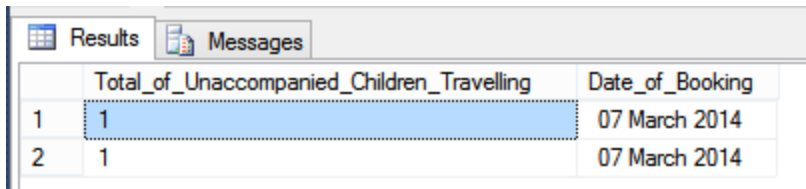
7.3.5 Query (xix)

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.

```
Select Count(p.PassengerID) as Total_of_Unaccompanied_Children_Travelling,  
      ' 07 March 2014 ' as Date_of_Booking  
from PassengerDetails p  
inner join PassengerService ps on p.PassengerID = ps.PassengerID  
inner join ServiceDetails s on ps.ServiceID = s.ServiceID  
inner join ReservationPassenger rp on p.PassengerID = rp.PassengerID  
inner join Reservation r on rp.ReservationID = r.ReservationID  
Where r.ReservationDate = '2014-05-07' AND s.ServiceID = 'S0002'  
group by CUBE(p.PassengerID)
```

Figure 42: Solution of query (xix)



	Total_of_Unaccompanied_Children_Travelling	Date_of_Booking
1	1	07 March 2014
2	1	07 March 2014

Figure 43: Results of query (xix)

The result are showing the total number of unaccompanied child travelling in specific date given. The result are three people.

7.3.6 Query (xx)

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

```

Select p.Name, p.DateOfBirth, p.PassportNumber, p.Address, p.Gender, r.ReservationDate, s.ServiceDescription
from PassengerDetails p
inner join ReservationPassenger rp on p.PassengerID = rp.PassengerID
inner join Reservation r on rp.ReservationID = r.ReservationID
inner join PassengerService ps on p.PassengerID = ps.PassengerID
inner join ServiceDetails s on ps.ServiceID = s.ServiceID
WHERE ps.ServiceID not in('S0003')
Group by p.Name, p.DateOfBirth, p.PassportNumber, p.Address, p.Gender, r.ReservationDate, s.ServiceDescription

```

Figure 44: Solution of query (xx)

Results		Messages					
	Name	DateOfBirth	PassportNumber	Address	Gender	ReservationDate	ServiceDescription
1	Chin Chooi	1992-04-23	AA5330241020	Ipoh	Female	2014-05-08	Wheel Chair
2	Simon	1978-12-21	AA4452089132	Puchong	Male	2014-05-07	Unaccompanied Minor Service

Figure 45: Results of query (xx)

This query is to show the result of people who are having extra service during their flight.

7.3.7 Query (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.

```

Select r.ReservationName, r.RStatus, r.FlightType, m.MealName, s.ServiceDescription,
(r.Infant+r.Child+r.Youth+r.Adult+r.Senior) as Total_Seat_Booked,
((CAST(r.Infant+r.Child+r.Youth+r.Adult+r.Senior as decimal))*(pay.Amount+s.ServiceCharge)) as Total_Payment_RM
from Reservation r
inner join ReservationPassenger rp on r.ReservationID = rp.ReservationID
inner join PassengerDetails p on rp.PassengerID = p.PassengerID
inner join PassengerService ps on p.PassengerID = ps.PassengerID
inner join ServiceDetails s on ps.ServiceID = s.ServiceID
inner join Payment pay on r.ReservationID = pay.ReservationID
inner join ReservationFlight rf on r.ReservationID = rf.ReservationID
inner join FlightDetails f on rf.FlightID = f.FlightID
inner join FlightMeal fm on f.FlightID = fm.FlightID
inner join MealDetails m on fm.MealID = m.MealID

```

Figure 46: Solution of query (xxi)

	ReservationName	RStatus	FlightType	MealName	ServiceDescription	Total_Seat_Booked	Total_Payment_RM
1	Alexander	Invalid	One Way	Fried Rice	No Service	2	2000
2	Alexander	Invalid	One Way	Fruit Salad	No Service	2	2000
3	Alicia	Booked	Multi Way	Pasta	No Service	4	4000
4	Alicia	Booked	Multi Way	Sushi	No Service	4	4000
5	Ahmed I-mon	Booked	Multi Way	Fruit Salad	No Service	4	4000
6	Ahmed I-mon	Booked	Multi Way	Sandwich	No Service	4	4000
7	Stephanie	Booked	One Way	Pizza	No Service	4	4000
8	Stephanie	Booked	One Way	Pasta	No Service	4	4000
9	Jun Wei	Booked	One Way	Sushi	No Service	4	4000
10	Jun Wei	Booked	One Way	Fried Rice	No Service	4	4000
11	Simon	Booked	One Way	Fruit Salad	Unaccompanied Minor Service	3	3000
12	Simon	Booked	One Way	Sushi	Unaccompanied Minor Service	3	3000
13	Chin Chooi	Booked	Multi Way	Meatballs	Wheel Chair	4	4000
14	Chin Chooi	Booked	Multi Way	Pizza	Wheel Chair	4	4000

Figure 47: Results of query (xxi)

This result is according to the business strategy, which staff need to check what are the details that should show at their monitor screen. The details are Reservation Name, the status, flight type, meal and service reserved, the number of seat booked and the payment of passenger going to

8 PERSONAL REFLECTION REPORT

8.1 ALEXANDER HO YINGHAN TP022858

For this assignment, we were required to form a group of three to develop a database system for Travel Safe International. During the course of the assignment, we have collaborated together to create diagrams such as Entity Relationship Diagram (ERD) to fulfill the business requirements of the coursework. We have also discussed many forms of optimization strategies to be used in the design of the database tables to ensure integrity and minimal redundancy.

During the actual development of the database, I faced many difficulties while creating the database but was able to get through them with discussions with my group members and also Mr. Abdallah, whom I am also very thankful for his patience in repeating his lectures or tutorial to ensure that I and the rest of my course mates are able to fully understand the advanced database techniques that he is trying to impart on us. Thanks to the knowledge I have learned from these encounters, I was able to successfully implement triggers and stored procedures into the database and have them working without any issues.

For the personal SQL queries, I was tasked with the questions allocated for member part 1. The majority of the questions proved quite challenging and some in particular required me to do additional research using the internet as well as referring to study material in the library of APU. Examples of these questions are ones which required the use of rollup or cube statements which I had trouble understanding initially, but after going through the research, I was able to understand the uses of these functions and was able to solve all the questions assigned to me.

To conclude my reflections on this assignment, I would have to say that it was a very interesting and great experience going through the ups and downs during the course of completing this database. I have learned a great deal of new advanced database functions which will certainly come in handy in the future during software development projects which makes use of databases such as my upcoming final year project.

8.2 TEH CHUN WEI (TP027353)

Several lesson been learned during the construction of the database and document. Other than tutorial and lecture class, this assignment also allow me to practice my knowledge and skill on advanced database such as triggers, store procedure and other.

To build an ERD that can understand by all team members, few meeting had organized by team leader to discuss about detail of assignment. During building of database, lots of help had been provided by team leader, Alex to me whenever I face some logical problem on relationship between tables.

3 main function such as constraints, triggers and store procedure are the next part to proceed after database been constructs. This 3 function contribute different effect to the system. Custom function above had to provide by each member for this assignment. Among this 3 function, the part that occur most error during development was triggers. I done with the help of team leader and web surfing to solve the problem where the code was unable to retrieve and store into relate table.

I would like to thanks at here for those who provided help to me during development of whole assignment.

8.3 AHMED I-MON (TP030577)

This assignment has helped me gain a lot of knowledge and information about database system. As this is a group assignment, it also helped me a lot to work in groups as well. Together working with my team mates, I learned a lot about Entity Relationship Diagram (ERD) and normalizations as well. These diagrams were based on the case study which was given to the group as an assignment.

Each member was given a set of 7 statements upon which queries had to be implemented. The 7 statements I chose had given me a lot of knowledge and gained much information about database systems and queries. I appreciate my group members for their time and effort and most of all, their help to complete this assignment. I have learned a lot from them and new things such as rollup and cube statements.

Apart from what the lecturer taught us in the classes, this group assignment enlightened me with the importance of standard procedures, and trigger like statements to a database. Stored procedures provides a new table to let a user store some important attributes. Trigger is a rule that allows the user to access, insert, update or delete some data. In addition, one really important command I learned is the CAST command which helps in situations like changing datetime data type to a float, decimal or an integer.

The project has been a success and on behalf of my team mates, I would like to express my gratitude towards Mr. Abdallah S.M Alnatsha for his kind help throughout the module.

9 WORKLOAD MATRIX

No	Phases	Ahmed I-Mon	Teh Chun Wei	Alexander Ho YingHan
	Documentation			
1	Workload Matrix		100%	
2	Introduction			100%
3	Assumptions	33%	33%	33%
4	Personal Reflection	33%	33%	33%
	Design			
5	EERD	33%	33%	33%
6	Structure of Table		100%	
7	Normalization			100%
8	Optimization Strategy	33%	33%	33%
9	Constraint	33%	33%	33%
10	Trigger	33%	33%	33%
11	Store Procedure	33%	33%	33%
	Implementation			
12	Database Development	33%	33%	33%
13	Member 1 Query			100%
14	Member 2 Query		100%	
15	Member 3 Query	100%		
	Signature			

10 REFERENCES

Codeidol.com, 2012. *Codeidol.com*. [Online]

Available at: <http://codeidol.com/sql/sql-performance-tuning/Constraints/NOT-NULL/>
[Accessed 16 2 2014].

Microsoft Developer Network, 2012. *CREATE TRIGGER (Transact-SQL)*. [Online]

Available at: <http://msdn.microsoft.com/en-us/library/ms189799.aspx>
[Accessed 14 February 2014].

Microsoft Technet, 2014. *PRINT (Transact-SQL)*. [Online]

Available at: <http://technet.microsoft.com/en-us/library/ms176047.asp>
[Accessed 13 February 2014].

Microsoft Technet, 2014. *Using RAISERROR*. [Online]

Available at: [http://technet.microsoft.com/en-us/library/ms177497\(v=sql.105\).aspx](http://technet.microsoft.com/en-us/library/ms177497(v=sql.105).aspx)
[Accessed 13 February 2014].

W3Schools, 2014. *SQL Constraints*. [Online]

Available at: http://www.w3schools.com/sql/sql_constraints.asp
[Accessed 10 February 2014].

W3Schools, 2014. *SQL PRIMARY KEY Constraint*. [Online]

Available at: http://www.w3schools.com/sql/sql_primarykey.asp
[Accessed 11 February 2014].

W3Schools, 2014. *SQL Server CONVERT() Function*. [Online]

Available at: http://www.w3schools.com/sql/func_convert.asp
[Accessed 11 February 2014].