U·C·T·I
ASIA PACIFIC UNIVERSITY
COLLEGE of TECHNOLOGY & INNOVATION

GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT004-3-3-ADVBS

ADVANCED DATABASE SYSTEMS

APD3F2205IT(DT) / APD3F2205IT / UC3F2205SE / APD3F2205SE / APU3F2205IT(CC) / APU3F2205SE / UC3F2205IT(CC)

HAND OUT DATE: 31 OCTOBER 2022

HAND IN DATE: 20 DECEMBER 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 Submit your assignment at the administrative counter

2 Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing)

3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld

4 Cases of plagiarism will be penalized

5 The assignment should be bound in an appropriate style (comb bound or stapled).

6 Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.

7 You must obtain 50% overall to pass this module.

Workload Matrix

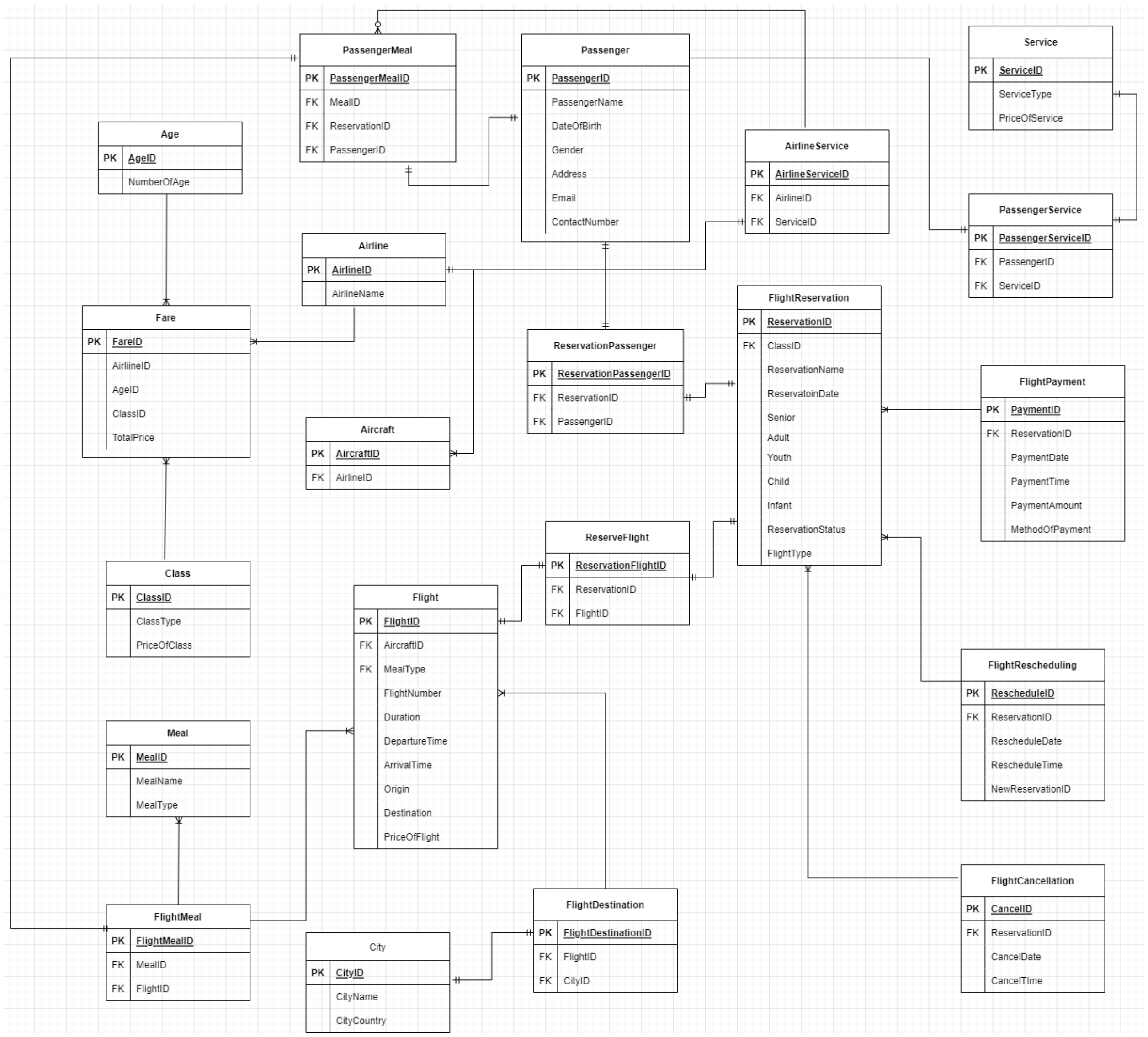
Group 1	Chong Zhan Wei TP046181	Yap Jing Hoong TP046254	Ricky Kee Shi Kit TP046842
Introduction	33%	33%	33%
ERD	33%	33%	33%
ERM	33%	33%	33%
Normalization	33%	33%	33%
Constraints	33%	33%	33%
Triggers	32%	35%	33%
Stored Procedure	33%	33%	33%
Optimization Strategy	33%	33%	33%
SQL Queries			
Student 1	100%		
Student 2		100%	
Student 3			100%

Table of Contents

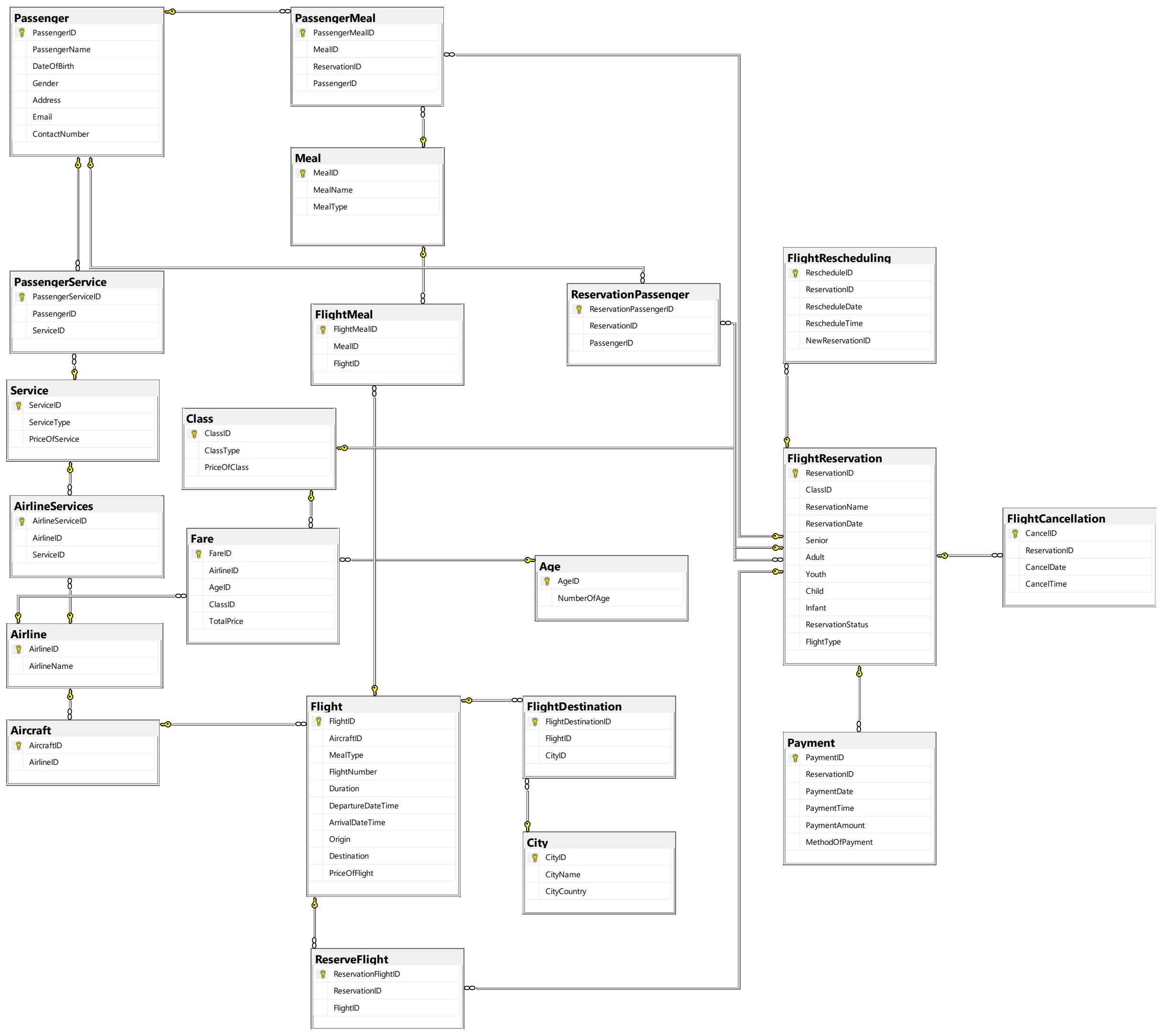
1.0 Introduction

Travel Safe International (TSI), one of the leading companies in this industry, is a participant in global distribution systems and sells tickets for a number of different airlines. TSI, in its most basic form, offers the primary functionality that are required of a booking system. These features include the ability to book flights, register customers, and reschedule flights. You are required to construct a database in accordance with certain rules in order to hold information pertaining to the booking of flights, the registration of clients, and other topics. In light of the fact that the policies and procedures of each airline need to be taken into account, enquiries will be made to satisfy the requirements of the business and provide additional features in order to keep TSI operational over the long term. Entity Relationship Model will be developed to exhibit all the entities in an understandable manner by identifying all of their attributes and their relationships with other entities. This will be done in order to make Entity Relationship Model. In order to fulfil the requirements of the company and maintain the database in a logical and organised fashion, this database management system will include constraints, stored procedures, and triggers. In addition to that, several optimization tactics will be implemented in an effort to increase the performance of the database.

2.0 Entity Relationship Diagram



3.0 Entity Relationship Model



4.0 Normalization

4.1 1NF

AirlineID, AirlineName

AircraftID

AgeID, NumberOfAge

ClassID, ClassType, PriceOfClass

FareID, TotalPrice

MealID, MealName, MealType

FlightMealID

FlightID, FlightNumber, Duration, DepartureTime, ArrivalTime, Origin, Destination, PriceOfFlight

FlightDestinationID

CityID, CityName, CityCountry

ReservationID, ReservationName, ReservationDate, Senior, Adult, Youth, Child, Infant, Reservation Status, FlightType

ReservationFlightID

PassengerID, PassengerName, DateOfBirth, Gender, Address, Email, ContactNumber

PassengerMealID

ReservationPassengerID

ServiceID, ServiceType, PriceOfService

AirlineServiceID

PassengerServiceID

PaymentID, PaymentDate, PaymentTime, PaymentAmount, MethodOfPayment

RescheduleID, RescheduleDate, RescheduleTime, NewReservationID

CancelID, CancelDate, CancelTime

4.2 2NF

AirlineID, AirlineName

AgeID, NumberOfAge

ClassID, ClassType, PriceOfClass

MealID, MealName, MealType

CityID, CityName, CityCountry

MealID, FlightID, FlightMealID

FlightID, CityID, FlightDestinationID

FlightID, AircraftID, FlightNumber, Duration, DepartureTime, ArrivalTime, Origin, Destination, PriceOfFlight

MealID, ReservationID, PassengerID, PassengerMealID

PassengerID, PassengerName, DateOfBirth, Gender, Address, Email, ContactNumber

ReservationID, PassengerID, ReservationPassengerID

ReservationID, FlightID, ReservationFlightID

ReservationID, ClassID, ReservationName, ReservationDate, Senior, Adult, Youth, Child, Infant, Reservation Status, FlightType

AirlineID, ServiceID, AirlineServiceID

ServiceID, ServiceType, PriceOfService

PassengerID, ServiceID, PassengerServiceID

PaymentID, ReservationID, PaymentDate, PaymentTime, PaymentAmount, MethodOfPayment

RescheduleID, ReservationID, RescheduleDate, RescheduleTime, NewReservationID

CancelID, ReservationID, CancelDate, CancelTime

4.3 3NF

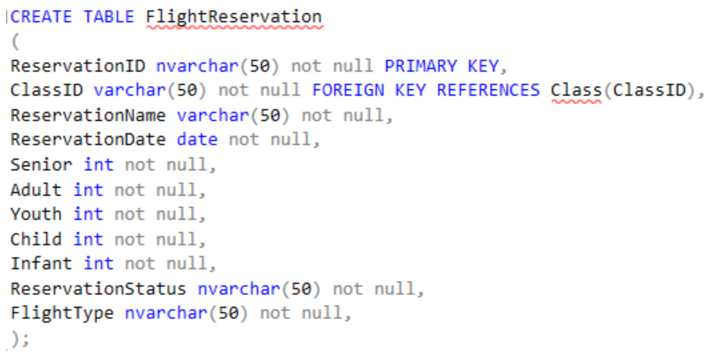
FareID, TotalPrice

5.0 Constraints

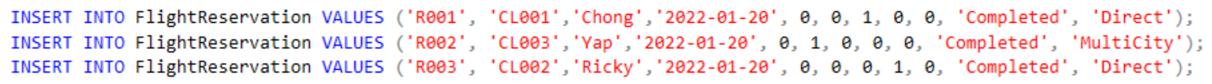
Constraints in SQL are usually used to enforce rules for data in a database. By using constraints, it helps user to ensure the reliability and accuracy of data in the database.

5.1 Primary Key

Primary Key is one of the constraints which is Not Null and Unique. In a database table, it must have only one primary key and it does not repeat the value in the table.



```sql
CREATE TABLE FlightReservation
(
ReservationID nvarchar(50) not null PRIMARY KEY,
ClassID varchar(50) not null FOREIGN KEY REFERENCES Class(ClassID),
ReservationName varchar(50) not null,
ReservationDate date not null,
Senior int not null,
Adult int not null,
Youth int not null,
Child int not null,
Infant int not null,
ReservationStatus nvarchar(50) not null,
FlightType nvarchar(50) not null,
);
```

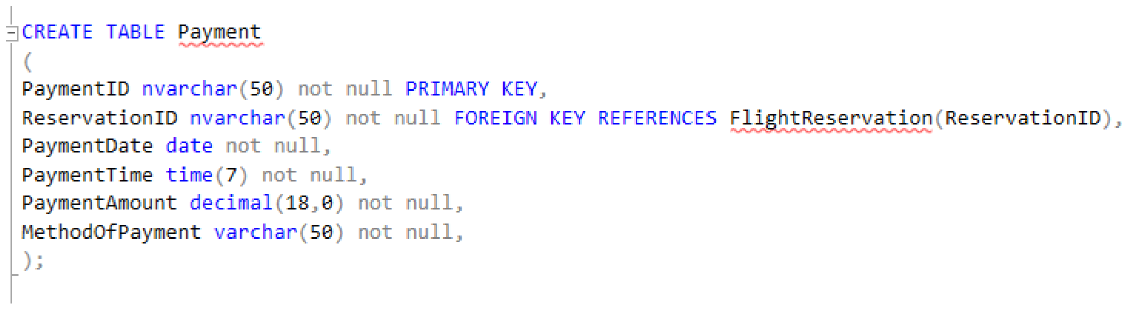


```sql
INSERT INTO FlightReservation VALUES ('R001', 'CL001','Chong','2022-01-20', 0, 0, 1, 0, 0, 'Completed', 'Direct');
INSERT INTO FlightReservation VALUES ('R002', 'CL003','Yap','2022-01-20', 0, 1, 0, 0, 0, 'Completed', 'MultiCity');
INSERT INTO FlightReservation VALUES ('R003', 'CL002','Ricky','2022-01-20', 0, 0, 0, 1, 0, 'Completed', 'Direct');
```

The figure above shows the example use of Primary Key in FlightReservation table. As from the figure above, it shown that ReservationID was declared as a Primary key. Thus, user able to use it to join FlightReservataion Table with other related table to obtain valuable data information.

5.2 Foreign Key

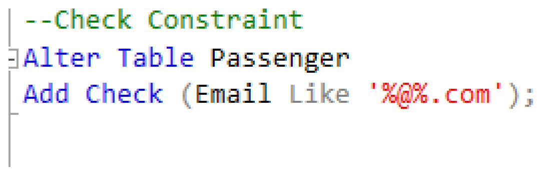
Foreign Key is one of the constraints which is used to prevent invalid data from being inserted to the key column because it has the values contained in its parent’s table.


```
CREATE TABLE Payment
(
PaymentID nvarchar(50) not null PRIMARY KEY,
ReservationID nvarchar(50) not null FOREIGN KEY REFERENCES FlightReservation(ReservationID),
PaymentDate date not null,
PaymentTime time(7) not null,
PaymentAmount decimal(18,0) not null,
MethodOfPayment varchar(50) not null,
);
```

The figure above shows the uses of Foreign Key in Payment Table. As from the figure above, it shown that the ReservationID was a Foreign key, and it references from the FlightReservation Table which is its parent table.

5.3 Check

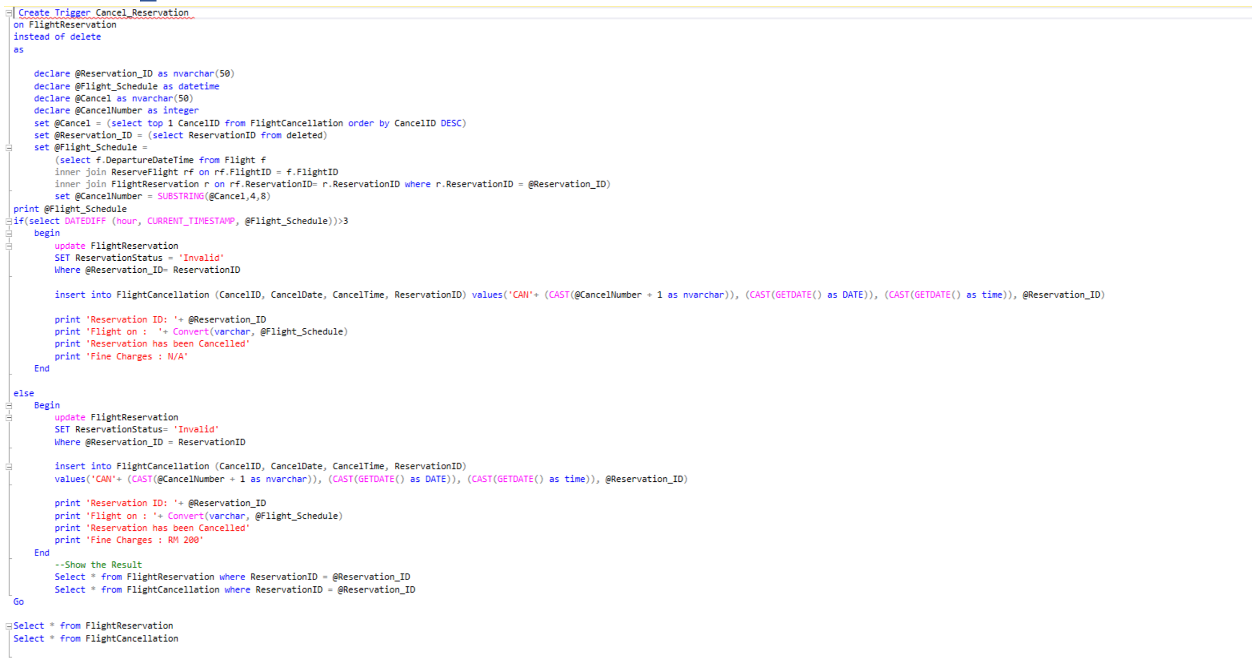
Check is a constraint which is used to limit the value range in a table column. User able to define the value in certain column to ensure it follows the 'rule'.


```
--Check Constraint
Alter Table Passenger
Add Check (Email Like '%@%.com');
```

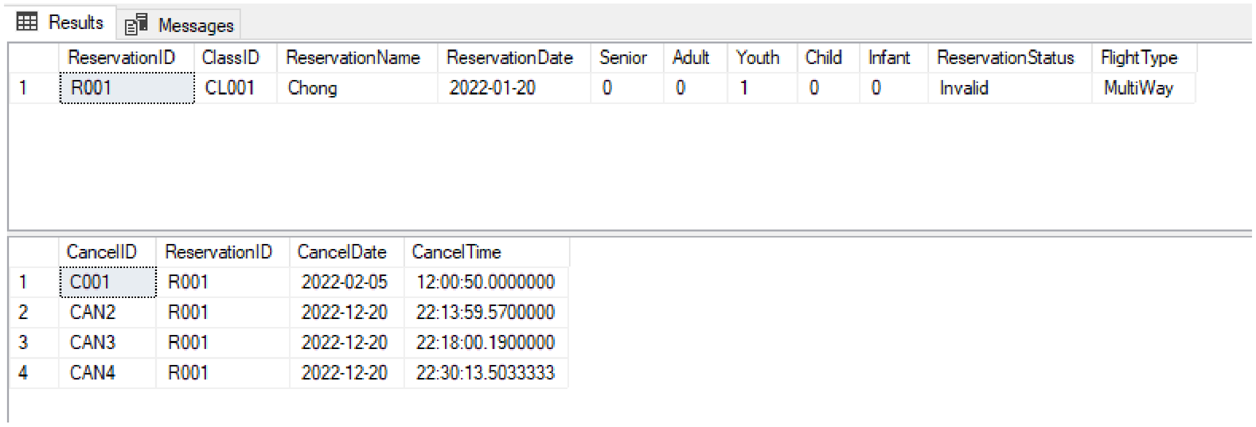
The figure above shows the example use of Check constraint. As from the table above, it shown that the email of passenger must be using the format of an email which is '%@%.com'.

6.0 Trigger

6.1 Cancel_Reservation



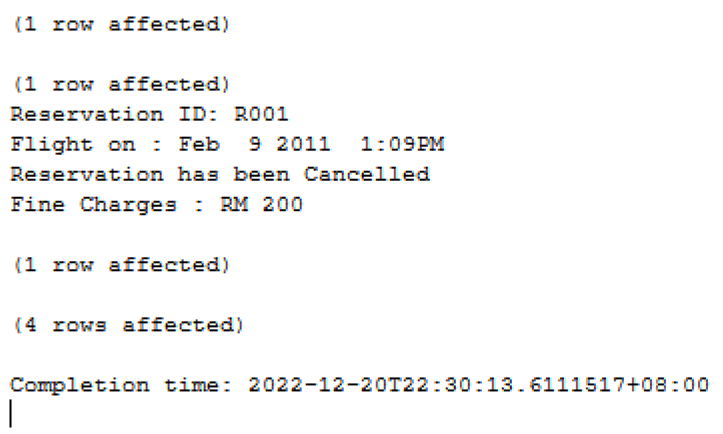
```sql
Create Trigger Cancel_Reservation
on FlightReservation
instead of delete
as

    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 FlightCancellation order by CancelID DESC)
    set @Reservation_ID = (select ReservationID from deleted)
    set @Flight_Schedule =
        (select f.DepartureDateTime from Flight f
        inner join ReserveFlight rf on rf.FlightID = f.FlightID
        inner join FlightReservation r on rf.ReservationID= r.ReservationID where r.ReservationID = @Reservation_ID)
        set @CancelNumber = SUBSTRING(@Cancel,4,8)
print @Flight_Schedule
if(select DATEDIFF (hour, CURRENT_TIMESTAMP, @Flight_Schedule))>3
    begin
        update FlightReservation
        SET ReservationStatus = 'Invalid'
        Where @Reservation_ID= ReservationID

        insert into FlightCancellation (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 FlightReservation
        SET ReservationStatus= 'Invalid'
        Where @Reservation_ID = ReservationID

        insert into FlightCancellation (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 FlightReservation where ReservationID = @Reservation_ID
        Select * from FlightCancellation where ReservationID = @Reservation_ID
Go

Select * from FlightReservation
Select * from FlightCancellation
```



Results | Messages

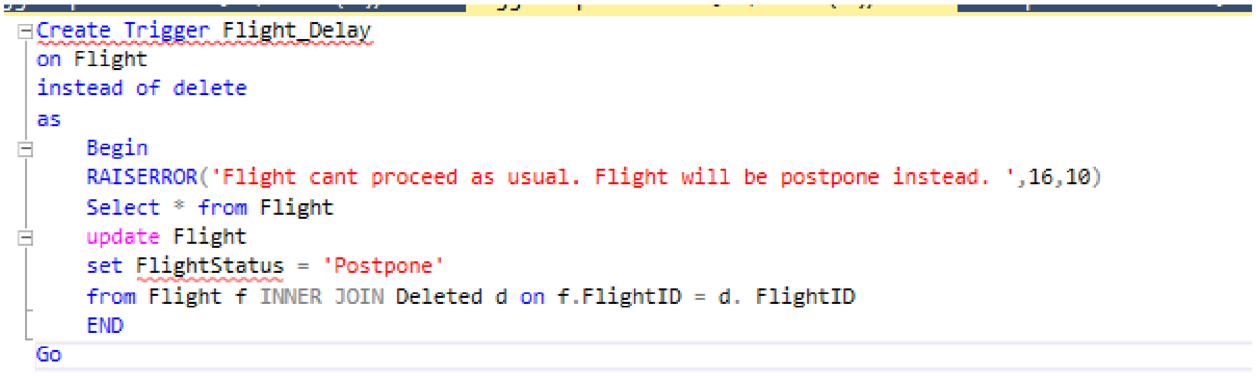
	ReservationID	ClassID	ReservationName	ReservationDate	Senior	Adult	Youth	Child	Infant	ReservationStatus	FlightType
1	R001	CL001	Chong	2022-01-20	0	0	1	0	0	Invalid	MultiWay

	CancelID	ReservationID	CancelDate	CancelTime
1	C001	R001	2022-02-05	12:00:50.0000000
2	CAN2	R001	2022-12-20	22:13:59.5700000
3	CAN3	R001	2022-12-20	22:18:00.1900000
4	CAN4	R001	2022-12-20	22:30:13.5033333

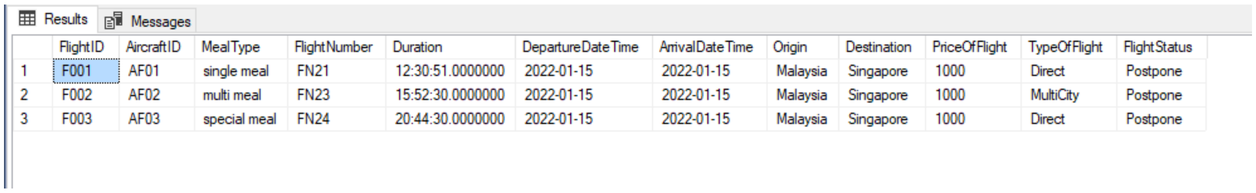


```
(1 row affected)

(1 row affected)
Reservation ID: R001
Flight on : Feb  9 2011  1:09PM
Reservation has been Cancelled
Fine Charges : RM 200

(1 row affected)

(4 rows affected)

Completion time: 2022-12-20T22:30:13.6111517+08:00
```

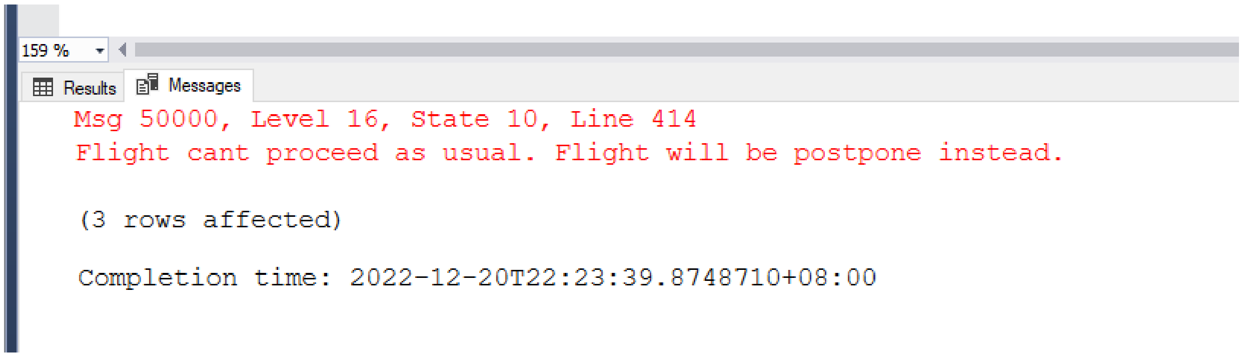
6.2 Flight_Delay



```
Create Trigger Flight_Delay
on Flight
instead of delete
as
    Begin
    RAISERROR('Flight cant proceed as usual. Flight will be postpone instead. ',16,10)
    Select * from Flight
    update Flight
    set FlightStatus = 'Postpone'
    from Flight f INNER JOIN Deleted d on f.FlightID = d. FlightID
    END
Go
```

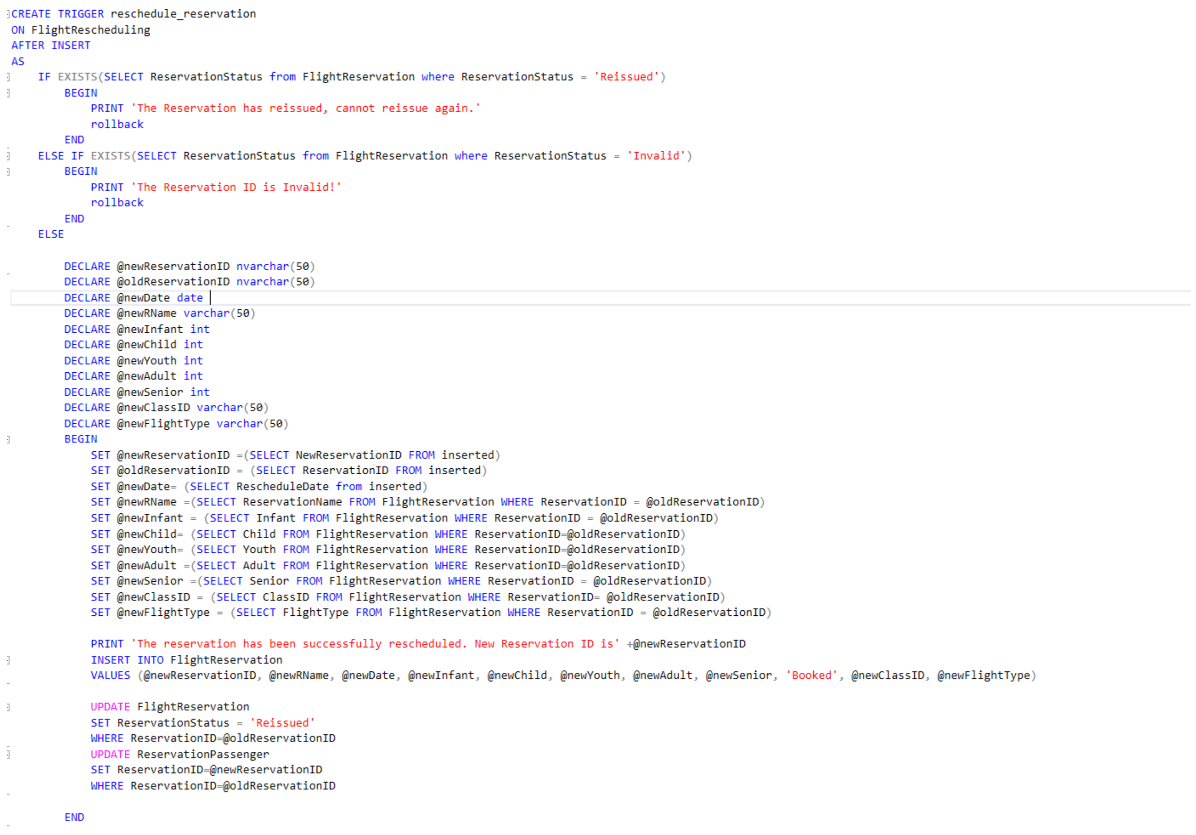


	FlightID	AircraftID	MealType	FlightNumber	Duration	DepartureDateTime	ArrivalDateTime	Origin	Destination	PriceOfFlight	TypeOfFlight	FlightStatus
1	F001	AF01	single meal	FN21	12:30:51.0000000	2022-01-15	2022-01-15	Malaysia	Singapore	1000	Direct	Postpone
2	F002	AF02	multi meal	FN23	15:52:30.0000000	2022-01-15	2022-01-15	Malaysia	Singapore	1000	MultiCity	Postpone
3	F003	AF03	special meal	FN24	20:44:30.0000000	2022-01-15	2022-01-15	Malaysia	Singapore	1000	Direct	Postpone



```
Msg 50000, Level 16, State 10, Line 414
Flight cant proceed as usual. Flight will be postpone instead.

(3 rows affected)

Completion time: 2022-12-20T22:23:39.8748710+08:00
```

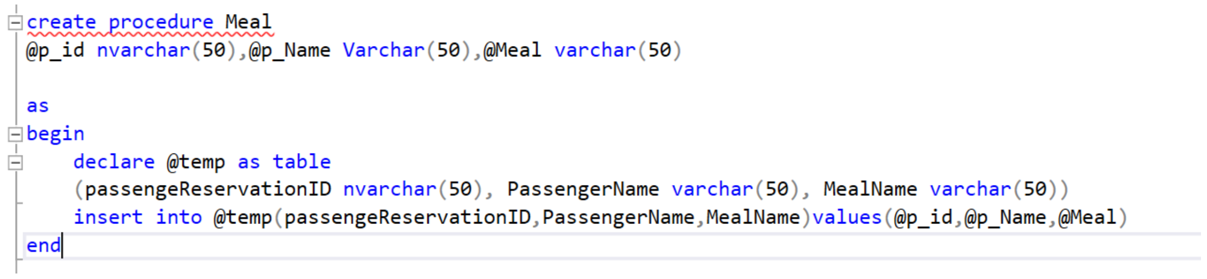
6.3 Reschedule_Reservation



```
CREATE TRIGGER reschedule_reservation
ON FlightRescheduling
AFTER INSERT
AS
    IF EXISTS(SELECT ReservationStatus from FlightReservation where ReservationStatus = 'Reissued')
        BEGIN
            PRINT 'The Reservation has reissued, cannot reissue again.'
            rollback
        END
    ELSE IF EXISTS(SELECT ReservationStatus from FlightReservation where ReservationStatus = '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 FlightReservation WHERE ReservationID = @oldReservationID)
            SET @newInfant = (SELECT Infant FROM FlightReservation WHERE ReservationID = @oldReservationID)
            SET @newChild= (SELECT Child FROM FlightReservation WHERE ReservationID=@oldReservationID)
            SET @newYouth= (SELECT Youth FROM FlightReservation WHERE ReservationID=@oldReservationID)
            SET @newAdult =(SELECT Adult FROM FlightReservation WHERE ReservationID=@oldReservationID)
            SET @newSenior =(SELECT Senior FROM FlightReservation WHERE ReservationID = @oldReservationID)
            SET @newClassID = (SELECT ClassID FROM FlightReservation WHERE ReservationID= @oldReservationID)
            SET @newFlightType = (SELECT FlightType FROM FlightReservation WHERE ReservationID = @oldReservationID)

            PRINT 'The reservation has been successfully rescheduled. New Reservation ID is' +@newReservationID
            INSERT INTO FlightReservation
            VALUES (@newReservationID, @newRName, @newDate, @newInfant, @newChild, @newYouth, @newAdult, @newSenior, 'Booked', @newClassID, @newFlightType)

            UPDATE FlightReservation
            SET ReservationStatus = 'Reissued'
            WHERE ReservationID=@oldReservationID
            UPDATE ReservationPassenger
            SET ReservationID=@newReservationID
            WHERE ReservationID=@oldReservationID

        END
```

7.0 Stored Procedure

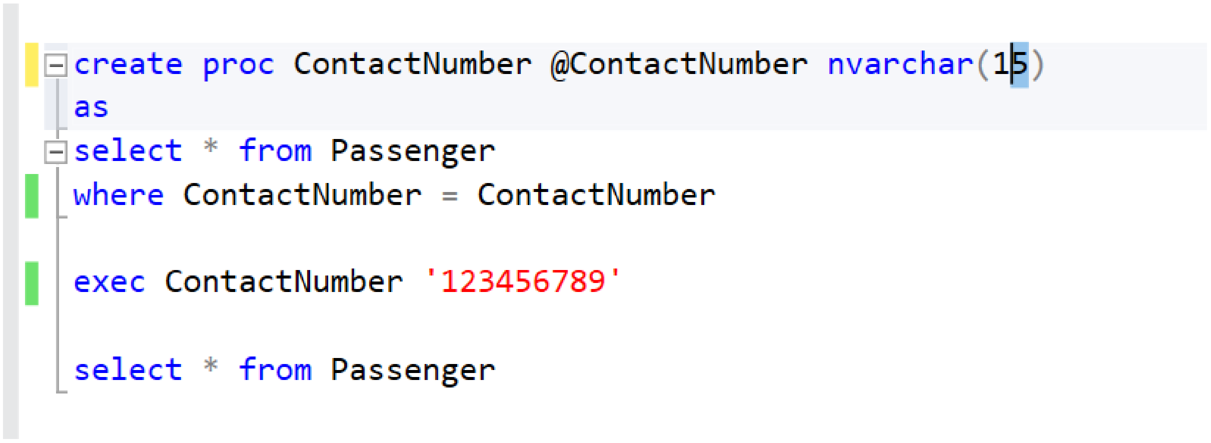
7.1 Insert in stored procedure



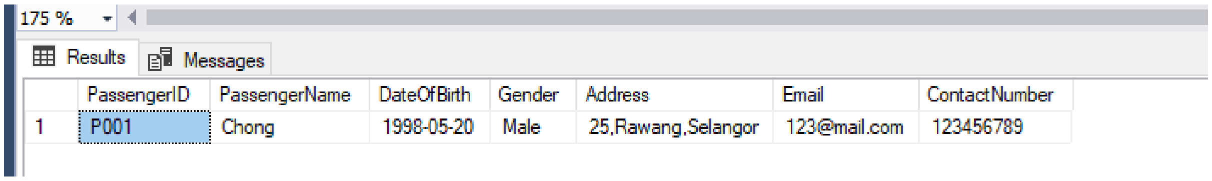
```
create procedure Meal
@p_id nvarchar(50),@p_Name Varchar(50),@Meal varchar(50)

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)
end
```

The relevant records from the Passenger and Meal tables are added as new entries using the meal method, and data is stored in a temporary table to provide the appropriate result. It displays of passengers on a particular aircraft have ordered respective meal.

7.2 Info Enquiry of Customer via Contact_Number



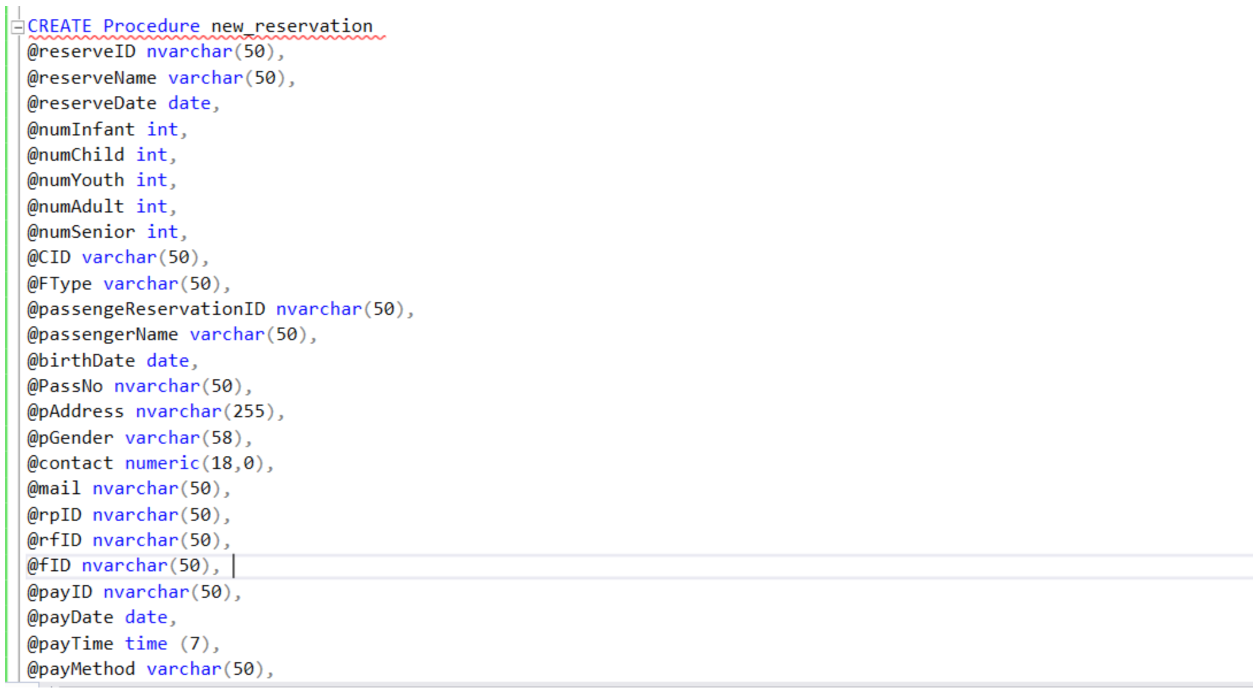
```
create proc ContactNumber @ContactNumber nvarchar(15)
as
select * from Passenger
where ContactNumber = ContactNumber

exec ContactNumber '123456789'

select * from Passenger
```



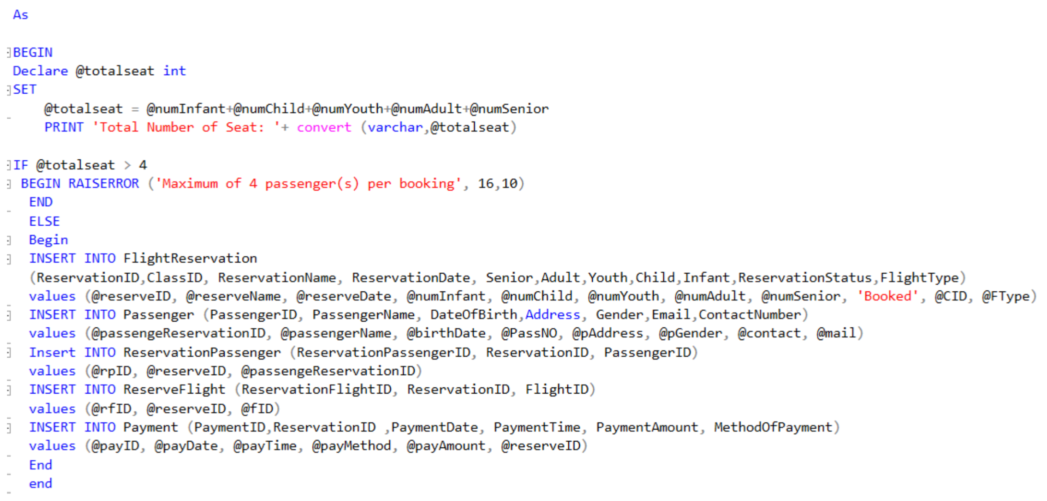
	PassengerID	PassengerName	DateOfBirth	Gender	Address	Email	ContactNumber
1	P001	Chong	1998-05-20	Male	25,Rawang,Selangor	123@mail.com	123456789

The implementation of the consumer inquiry through contact number allowed employees to swiftly get customer information. Contact numbers can be used as a supplementary primary key since they can uniquely identify a person. When enrolling for flight reservations, staff will ask for basic consumer information in order to differentiate the process. Furthermore, the system will check related contact number which entered and will look up this specific customer in the passenger table of database.

7.3 Insert New Reservation in stored procedure



```
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),
@passengeReservationID nvarchar(50),
@passengerName varchar(50),
@birthDate date,
@PassNo nvarchar(50),
@pAddress nvarchar(255),
@pGender varchar(58),
@contact numeric(18,0),
@mail nvarchar(50),
@rpID nvarchar(50),
@rfID nvarchar(50),
@fID nvarchar(50),
@payID nvarchar(50),
@payDate date,
@payTime time (7),
@payMethod varchar(50),
```



```
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 FlightReservation
  (ReservationID,ClassID, ReservationName, ReservationDate, Senior,Adult,Youth,Child,Infant,ReservationStatus,FlightType)
  values (@reserveID, @reserveName, @reserveDate, @numInfant, @numChild, @numYouth, @numAdult, @numSenior, 'Booked', @CID, @FType)
  INSERT INTO Passenger (PassengerID, PassengerName, DateOfBirth,Address, Gender,Email,ContactNumber)
  values (@passengeReservationID, @passengerName, @birthDate, @PassNO, @pAddress, @pGender, @contact, @mail)
  Insert INTO ReservationPassenger (ReservationPassengerID, ReservationID, PassengerID)
  values (@rpID, @reserveID, @passengeReservationID)
  INSERT INTO ReserveFlight (ReservationFlightID, ReservationID, FlightID)
  values (@rfID, @reserveID, @fID)
  INSERT INTO Payment (PaymentID,ReservationID ,PaymentDate, PaymentTime, PaymentAmount, MethodOfPayment)
  values (@payID, @payDate, @payTime, @payMethod, @payAmount, @reserveID)
  End
  end
```

In order to add a new flight reservation record into the “Reservation” table, “Passenger” table, “Reservation” table, “ReserveFlight” table, and “Payment” table, stored procedure from new_Flight_reservation is selected. Stored procedure will check number of total passengers to confirm passengers does not exceed 4 passengers before insertion to related tables. If limitation is surpassed, the latest record will not be added into tables and an error message will appear. On the other hand, record will be added into related tables when passengers is within the limit.

8.0 Optimization Strategy

8.1 Optimization Technique SQL

SQL statements are mostly used to get information out of a database. There are different ways to write queries that all work the same. It's important to know that the performance isn't the fastest because developers often make the mistake of writing queries in the wrong way. The list below shows how to improve the performance of things that can be optimized.

Many SQL developers use SELECT *, which means "select all," as a shorthand to query all the data in a table when doing exploratory queries. But if a table has a lot of fields and rows, it uses up a lot of database resources by asking for a lot of data that doesn't need to be looked up.

With the SELECT statement, you can tell the database to only look for the data you need to meet business needs. Here's what I mean.

Inefficient:

```
SELECT * FROM Passenger
```

This query could also get other information from the Passenger table, like DateOfBirth, Address, and ContactNumber.

Efficient:

```
SELECT PassengerID, PassengerName, DateOfBirth, Gender, Address, Email, ContactNumber

FROM Passenger
```

This query is much more organised and only pulls the information that is needed for Passenger data.

8.2 In a one-to-many relationship, duplicating none-key attributes

An one-to-many relationship is composed of two tables which with identical properties thtat will be connected together. A connection is a parent table that makes use of an attribute from a child object.

FlightReservation	**Class**
ReservationID	ClassID
ClassID	ClassType
ReservationName	PriceOfClass
ReservationDate	
Senior	
Adult	
Youth	
Child	
Infant	
ReservationStatus	
FlightType	

Based on the TSI, the ClassID should be added as a foreign key to the FlightReservation table. Use a "join" statement to get information from the Class table through the parent table.

FlightReservation
ReservationID
ReservationName
ReservationDate
Senior
Adult
Youth
Child

Infant
ReservationStatus
FlightType
ClassType
PriceOfClass

As the picture above shows, it cuts down on the number of JOIN statements used in databases. This makes the database run much faster and makes the query statement much simpler. It saves more space in the database and is simple enough that any staff member can learn it quickly.

8.3 De-Normalization

De-normalization is the method to improve the speed of the database to read by including redundant data or sorting data together. This is helpful to gather data from different databases and make a new data. Examples of denormalization are combining two Many-to-many relationship tables. For instance:

• PassengerMeal can have several Meals.

• Many Passengers can select from a variety of PassengerMeals.

The many-to-many relationship diagram below depicts how to obtain data from the Passenger and Meal tables.

Passenger	**PassangerMeal**	**Meal**
PassengerID	PassengerMealID	MealID
PassengerName	MealID	MealName
DateOfBirth	ReservationID	MealType
Gender	PassengerID	
Address		
Email		
ContactNumber		

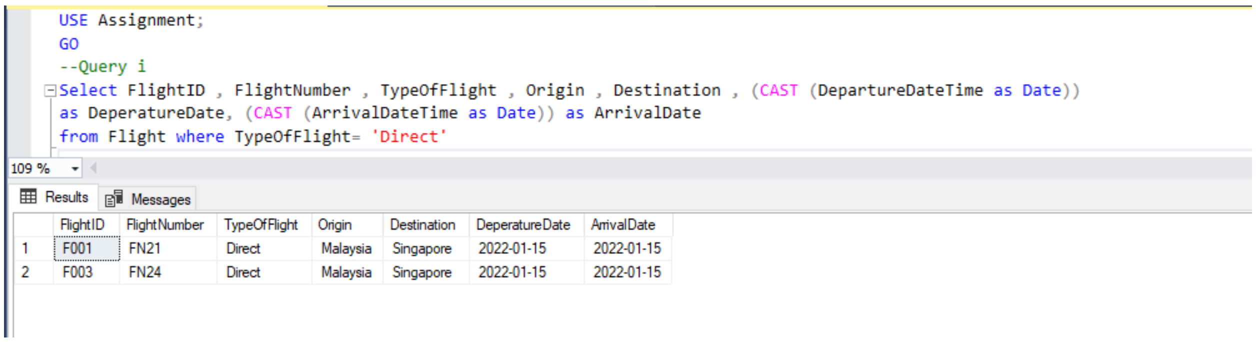
FlightReservation
PassengerID
PassengerName
DateOfBirth
Gender
Address
Email
ContactNumber
MealName
MealType

Based on first table, when denormalization is used, details of both the passenger and the meal will be combined, as shown in table above. When you use de normalisation, performance is better, you need fewer joins, and you can keep track of information about the past. It makes it easier for plane staff to set up food before a flight and pass it out on the plane. It is also a guide when errors occur such as how food is given out so that no one gets confused. In another way to look at this system, de-normalization makes it possible to keep the function of looking up the history of past transactions. This will make the system easier to manage.

9.0 Query

9.1 Student1: Chong Zhan Wei

9.1.1 Query i



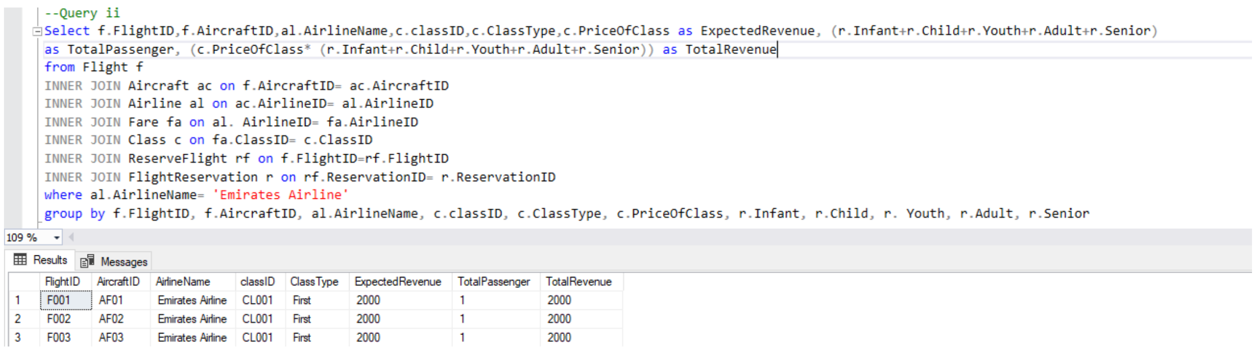
```sql
USE Assignment;
GO
--Query i
Select FlightID , FlightNumber , TypeOfFlight , Origin , Destination , (CAST (DepartureDateTime as Date))
as DeperatureDate, (CAST (ArrivalDateTime as Date)) as ArrivalDate
from Flight where TypeOfFlight= 'Direct'
```

109 %

Results | Messages

	FlightID	FlightNumber	TypeOfFlight	Origin	Destination	DeperatureDate	ArrivalDate
1	F001	FN21	Direct	Malaysia	Singapore	2022-01-15	2022-01-15
2	F003	FN24	Direct	Malaysia	Singapore	2022-01-15	2022-01-15

9.1.2 Query ii



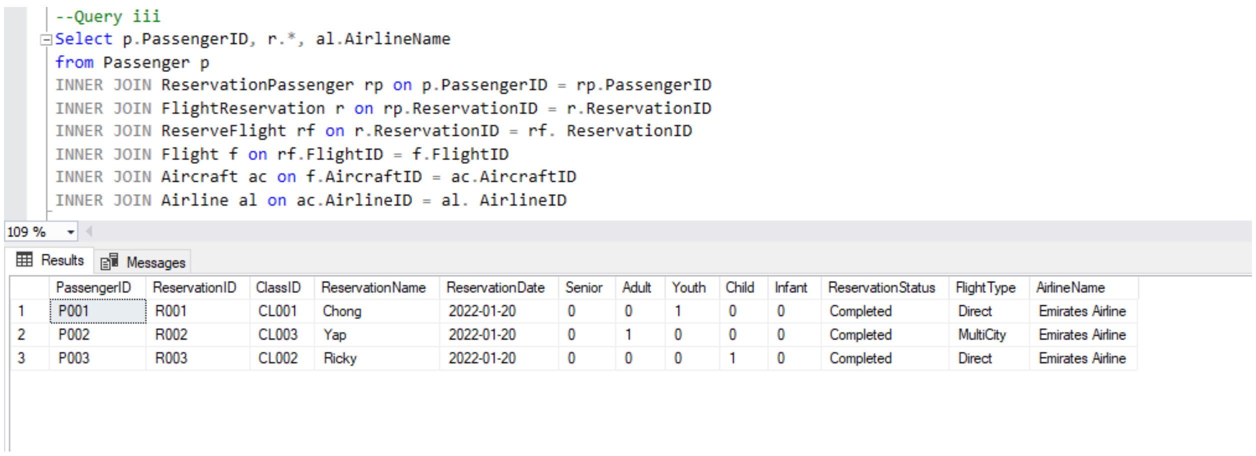
```sql
--Query ii
Select f.FlightID,f.AircraftID,al.AirlineName,c.classID,c.ClassType,c.PriceOfClass as ExpectedRevenue, (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)
as TotalPassenger, (c.PriceOfClass* (r.Infant+r.Child+r.Youth+r.Adult+r.Senior)) as TotalRevenue
from Flight f
INNER JOIN Aircraft ac on f.AircraftID= ac.AircraftID
INNER JOIN Airline al on ac.AirlineID= al.AirlineID
INNER JOIN Fare fa on al. AirlineID= fa.AirlineID
INNER JOIN Class c on fa.ClassID= c.ClassID
INNER JOIN ReserveFlight rf on f.FlightID=rf.FlightID
INNER JOIN FlightReservation r on rf.ReservationID= r.ReservationID
where al.AirlineName= 'Emirates Airline'
group by f.FlightID, f.AircraftID, al.AirlineName, c.classID, c.ClassType, c.PriceOfClass, r.Infant, r.Child, r. Youth, r.Adult, r.Senior
```

109 %

Results | Messages

	FlightID	AircraftID	AirlineName	classID	ClassType	ExpectedRevenue	TotalPassenger	TotalRevenue
1	F001	AF01	Emirates Airline	CL001	First	2000	1	2000
2	F002	AF02	Emirates Airline	CL001	First	2000	1	2000
3	F003	AF03	Emirates Airline	CL001	First	2000	1	2000

9.1.3 Query iii



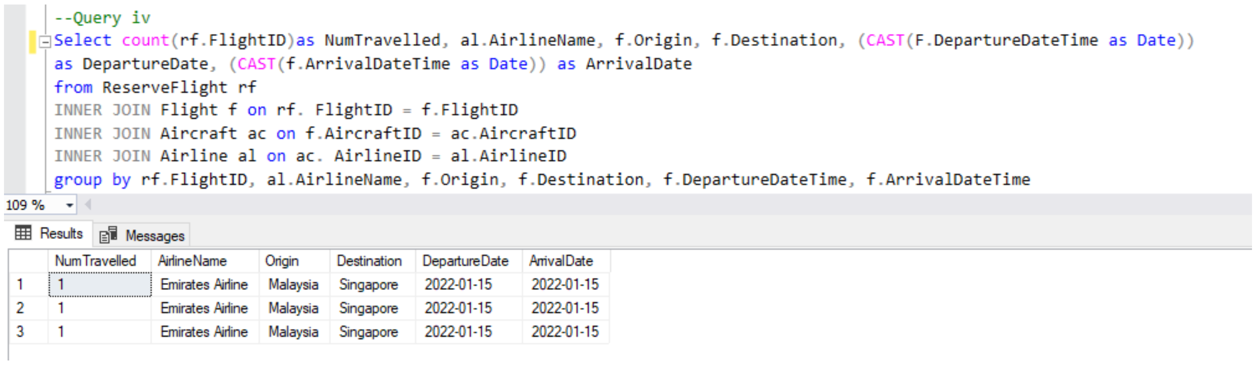
```sql
--Query iii
Select p.PassengerID, r.*, al.AirlineName
from Passenger p
INNER JOIN ReservationPassenger rp on p.PassengerID = rp.PassengerID
INNER JOIN FlightReservation r on rp.ReservationID = r.ReservationID
INNER JOIN ReserveFlight rf on r.ReservationID = rf. ReservationID
INNER JOIN Flight f on rf.FlightID = f.FlightID
INNER JOIN Aircraft ac on f.AircraftID = ac.AircraftID
INNER JOIN Airline al on ac.AirlineID = al. AirlineID
```

109 %

Results | Messages

	PassengerID	ReservationID	ClassID	ReservationName	ReservationDate	Senior	Adult	Youth	Child	Infant	ReservationStatus	FlightType	AirlineName
1	P001	R001	CL001	Chong	2022-01-20	0	0	1	0	0	Completed	Direct	Emirates Airline
2	P002	R002	CL003	Yap	2022-01-20	0	1	0	0	0	Completed	MultiCity	Emirates Airline
3	P003	R003	CL002	Ricky	2022-01-20	0	0	0	1	0	Completed	Direct	Emirates Airline

9.1.4 Query iv



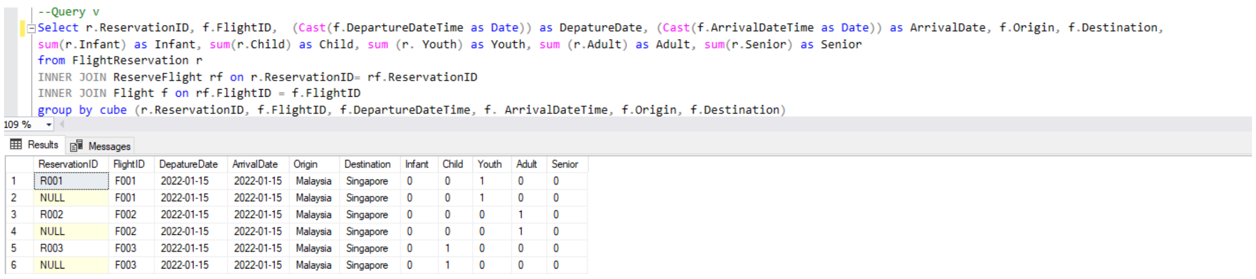
```sql
--Query iv
Select count(rf.FlightID)as NumTravelled, al.AirlineName, f.Origin, f.Destination, (CAST(F.DepartureDateTime as Date))
as DepartureDate, (CAST(f.ArrivalDateTime as Date)) as ArrivalDate
from ReserveFlight rf
INNER JOIN Flight f on rf. FlightID = f.FlightID
INNER JOIN Aircraft ac on f.AircraftID = ac.AircraftID
INNER JOIN Airline al on ac. AirlineID = al.AirlineID
group by rf.FlightID, al.AirlineName, f.Origin, f.Destination, f.DepartureDateTime, f.ArrivalDateTime
```

109 %

Results | Messages

	NumTravelled	AirlineName	Origin	Destination	DepartureDate	ArrivalDate
1	1	Emirates Airline	Malaysia	Singapore	2022-01-15	2022-01-15
2	1	Emirates Airline	Malaysia	Singapore	2022-01-15	2022-01-15
3	1	Emirates Airline	Malaysia	Singapore	2022-01-15	2022-01-15

9.1.5 Query v



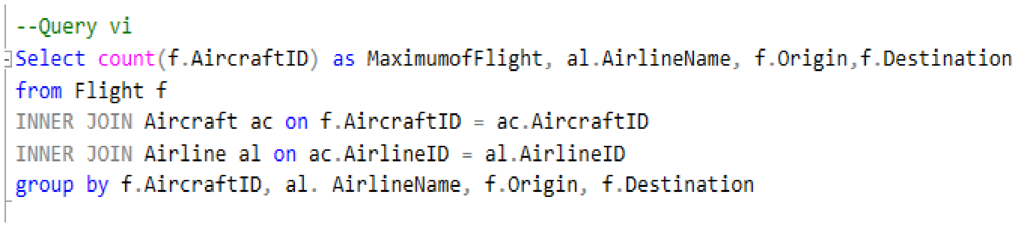
```sql
--Query v
Select r.ReservationID, f.FlightID,  (Cast(f.DepartureDateTime as Date)) as DepatureDate, (Cast(f.ArrivalDateTime as Date)) as ArrivalDate, f.Origin, f.Destination,
sum(r.Infant) as Infant, sum(r.Child) as Child, sum (r. Youth) as Youth, sum (r.Adult) as Adult, sum(r.Senior) as Senior
from FlightReservation r
INNER JOIN ReserveFlight rf on r.ReservationID= rf.ReservationID
INNER JOIN Flight f on rf.FlightID = f.FlightID
group by cube (r.ReservationID, f.FlightID, f.DepartureDateTime, f. ArrivalDateTime, f.Origin, f.Destination)
```

109 %

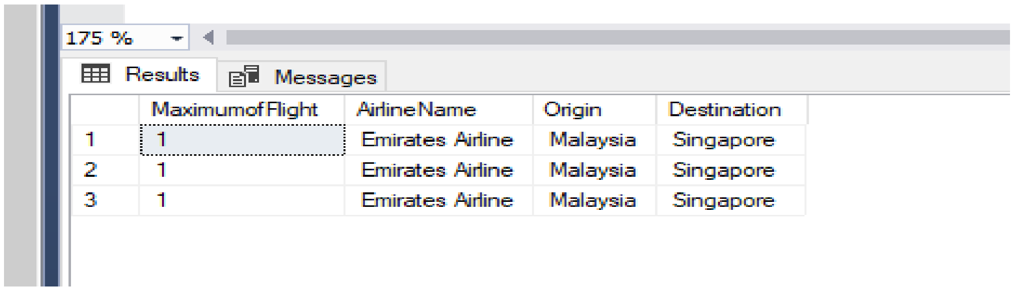
Results | Messages

	ReservationID	FlightID	DepatureDate	ArrivalDate	Origin	Destination	Infant	Child	Youth	Adult	Senior
1	R001	F001	2022-01-15	2022-01-15	Malaysia	Singapore	0	0	1	0	0
2	NULL	F001	2022-01-15	2022-01-15	Malaysia	Singapore	0	0	1	0	0
3	R002	F002	2022-01-15	2022-01-15	Malaysia	Singapore	0	0	0	1	0
4	NULL	F002	2022-01-15	2022-01-15	Malaysia	Singapore	0	0	0	1	0
5	R003	F003	2022-01-15	2022-01-15	Malaysia	Singapore	0	1	0	0	0
6	NULL	F003	2022-01-15	2022-01-15	Malaysia	Singapore	0	1	0	0	0

9.1.6 Query vi



```sql
--Query vi
Select count(f.AircraftID) as MaximumofFlight, al.AirlineName, f.Origin,f.Destination
from Flight f
INNER JOIN Aircraft ac on f.AircraftID = ac.AircraftID
INNER JOIN Airline al on ac.AirlineID = al.AirlineID
group by f.AircraftID, al. AirlineName, f.Origin, f.Destination
```



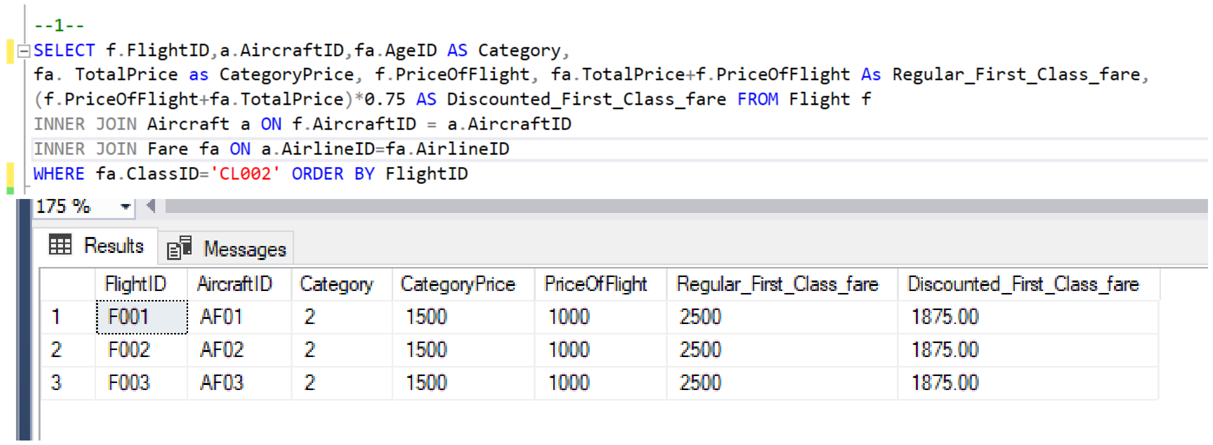
175 %

Results | Messages

	MaximumofFlight	AirlineName	Origin	Destination
1	1	Emirates Airline	Malaysia	Singapore
2	1	Emirates Airline	Malaysia	Singapore
3	1	Emirates Airline	Malaysia	Singapore

9.2 Student2: Yap Jing Hoong

9.2.1 Query i



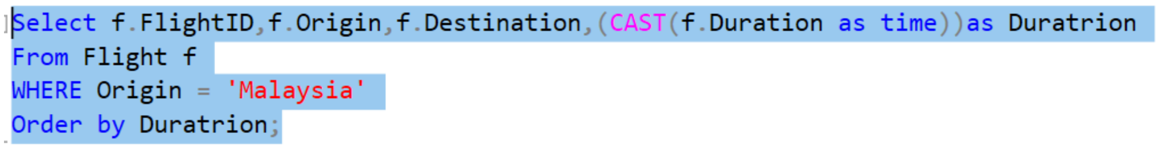
```
--1--
SELECT f.FlightID,a.AircraftID,fa.AgeID AS Category,
fa. TotalPrice as CategoryPrice, f.PriceOfFlight, fa.TotalPrice+f.PriceOfFlight As Regular_First_Class_fare,
(f.PriceOfFlight+fa.TotalPrice)*0.75 AS Discounted_First_Class_fare FROM Flight f
INNER JOIN Aircraft a ON f.AircraftID = a.AircraftID
INNER JOIN Fare fa ON a.AirlineID=fa.AirlineID
WHERE fa.ClassID='CL002' ORDER BY FlightID
```

175 %

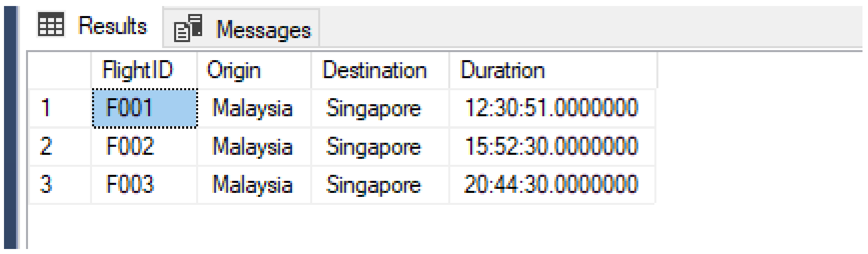
Results | Messages

	FlightID	AircraftID	Category	CategoryPrice	PriceOfFlight	Regular_First_Class_fare	Discounted_First_Class_fare
1	F001	AF01	2	1500	1000	2500	1875.00
2	F002	AF02	2	1500	1000	2500	1875.00
3	F003	AF03	2	1500	1000	2500	1875.00

9.2.2 Query ii



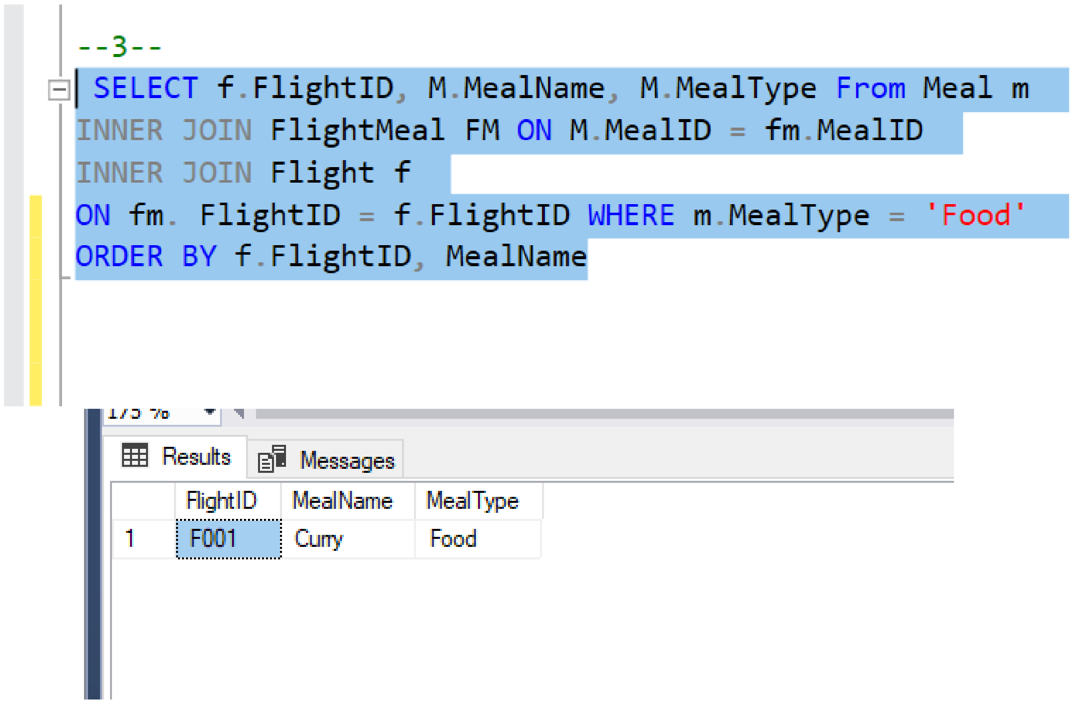
```
Select f.FlightID,f.Origin,f.Destination,(CAST(f.Duration as time))as Duratrion
From Flight f
WHERE Origin = 'Malaysia'
Order by Duratrion;
```



Results | Messages

	FlightID	Origin	Destination	Duratrion
1	F001	Malaysia	Singapore	12:30:51.0000000
2	F002	Malaysia	Singapore	15:52:30.0000000
3	F003	Malaysia	Singapore	20:44:30.0000000

9.2.3 Query iii



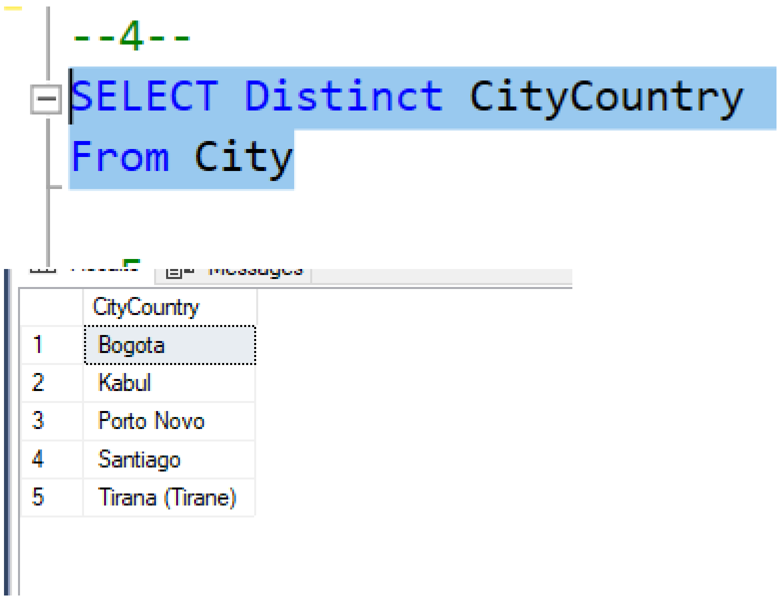
```
--3--
SELECT f.FlightID, M.MealName, M.MealType From Meal m
INNER JOIN FlightMeal FM ON M.MealID = fm.MealID
INNER JOIN Flight f
ON fm. FlightID = f.FlightID WHERE m.MealType = 'Food'
ORDER BY f.FlightID, MealName
```

175 %

Results | Messages

	FlightID	MealName	MealType
1	F001	Curry	Food

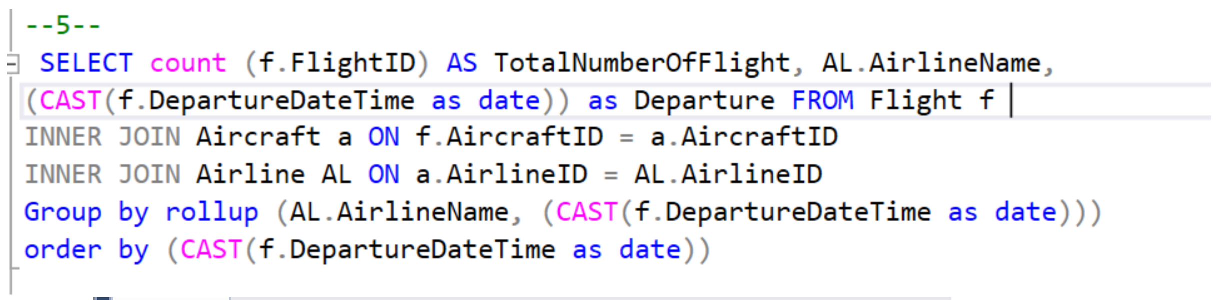
9.2.4 Query iv



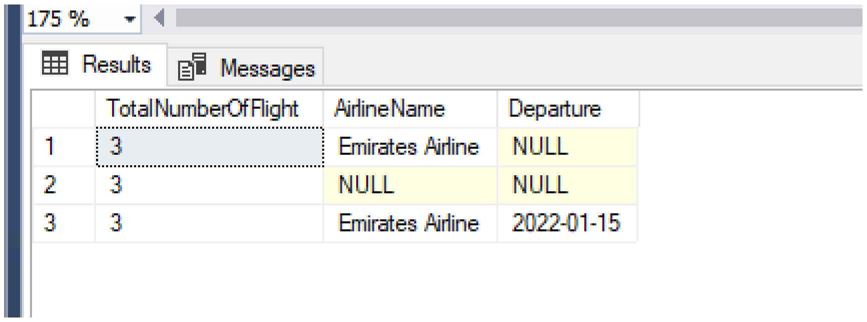
```
--4--
SELECT Distinct CityCountry
From City
```

	CityCountry
1	Bogota
2	Kabul
3	Porto Novo
4	Santiago
5	Tirana (Tirane)

9.2.5 Query v

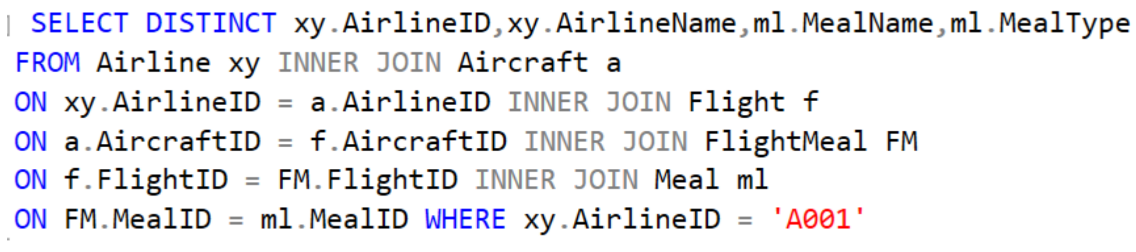


```
--5--
SELECT count (f.FlightID) AS TotalNumberOfFlight, AL.AirlineName,
(CAST(f.DepartureDateTime as date)) as Departure FROM Flight f
INNER JOIN Aircraft a ON f.AircraftID = a.AircraftID
INNER JOIN Airline AL ON a.AirlineID = AL.AirlineID
Group by rollup (AL.AirlineName, (CAST(f.DepartureDateTime as date)))
order by (CAST(f.DepartureDateTime as date))
```

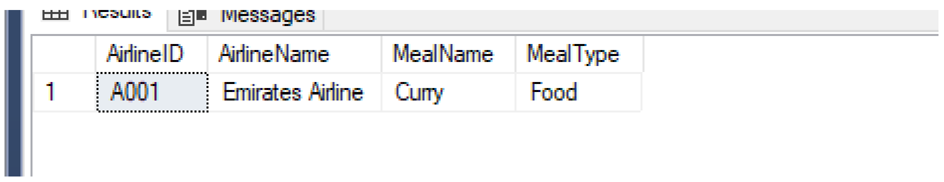


	TotalNumberOfFlight	AirlineName	Departure
1	3	Emirates Airline	NULL
2	3	NULL	NULL
3	3	Emirates Airline	2022-01-15

9.2.6 Query vi



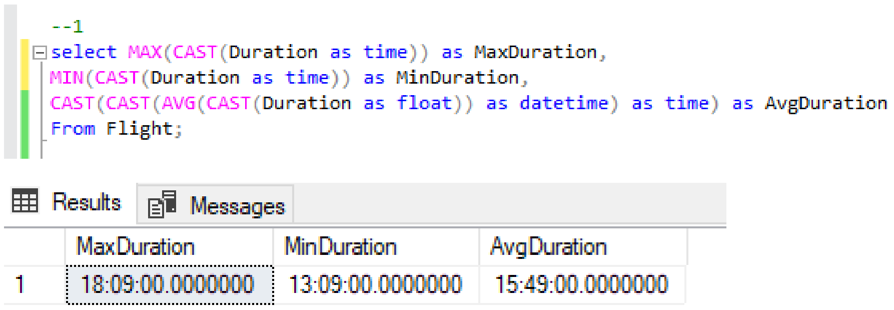
```
SELECT DISTINCT xy.AirlineID,xy.AirlineName,ml.MealName,ml.MealType
FROM Airline xy INNER JOIN Aircraft a
ON xy.AirlineID = a.AirlineID INNER JOIN Flight f
ON a.AircraftID = f.AircraftID INNER JOIN FlightMeal FM
ON f.FlightID = FM.FlightID INNER JOIN Meal ml
ON FM.MealID = ml.MealID WHERE xy.AirlineID = 'A001'
```



	AirlineID	AirlineName	MealName	MealType
1	A001	Emirates Airline	Curry	Food

9.3 Student3: Ricky Kee Shi Kit

9.3.1 Query I



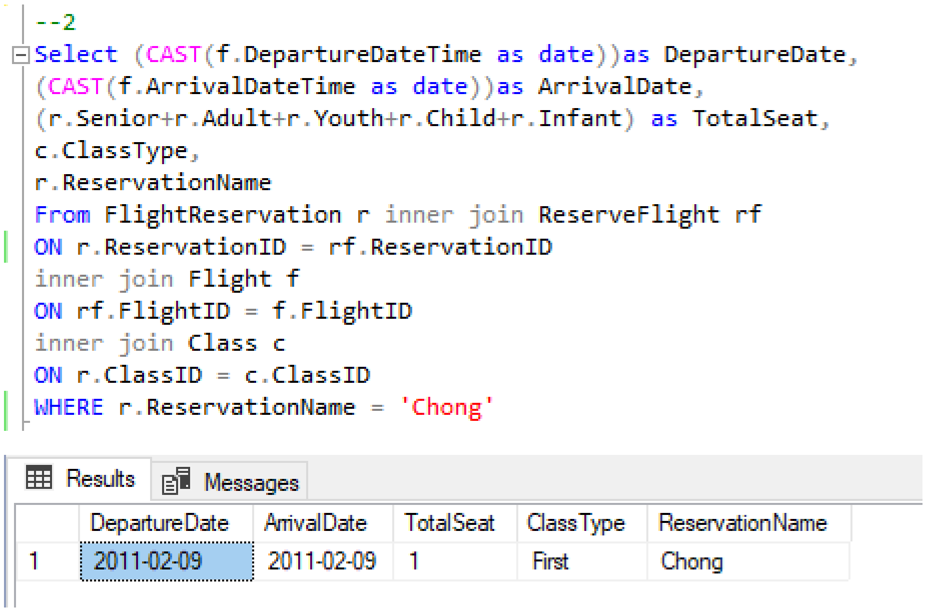
```sql
--1
select MAX(CAST(Duration as time)) as MaxDuration,
MIN(CAST(Duration as time)) as MinDuration,
CAST(CAST(AVG(CAST(Duration as float)) as datetime) as time) as AvgDuration
From Flight;
```

Results | Messages

	MaxDuration	MinDuration	AvgDuration
1	18:09:00.0000000	13:09:00.0000000	15:49:00.0000000

This is the outcome of the maximum, minimum, and average durations.

9.3.2 Query ii



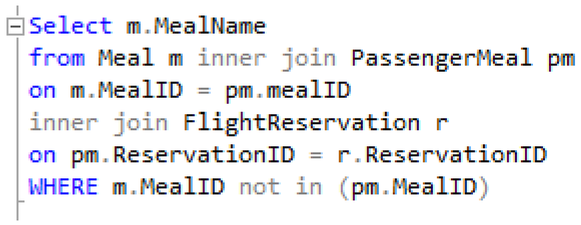
```sql
--2
Select (CAST(f.DepartureDateTime as date))as DepartureDate,
(CAST(f.ArrivalDateTime as date))as ArrivalDate,
(r.Senior+r.Adult+r.Youth+r.Child+r.Infant) as TotalSeat,
c.ClassType,
r.ReservationName
From FlightReservation r inner join ReserveFlight rf
ON r.ReservationID = rf.ReservationID
inner join Flight f
ON rf.FlightID = f.FlightID
inner join Class c
ON r.ClassID = c.ClassID
WHERE r.ReservationName = 'Chong'
```

Results | Messages

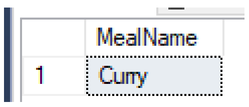
	DepartureDate	ArrivalDate	TotalSeat	ClassType	ReservationName
1	2011-02-09	2011-02-09	1	First	Chong

The passenger reserved a flight on a specific day, providing the total number of seats and seat class.

9.3.3 Query iii



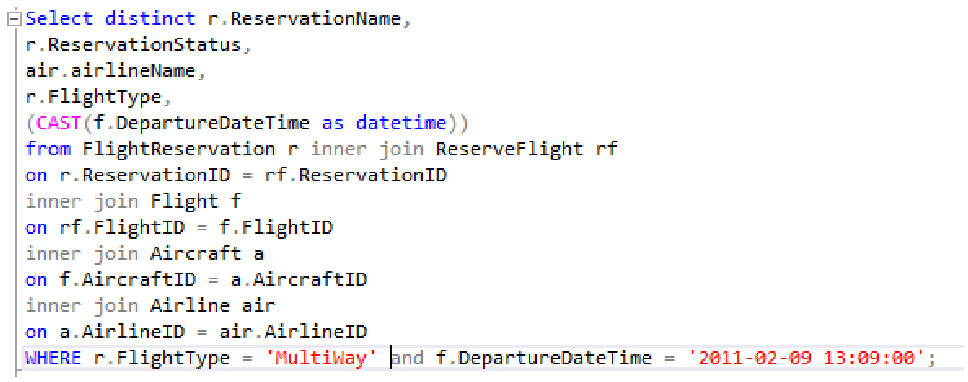
```sql
Select m.MealName
from Meal m inner join PassengerMeal pm
on m.MealID = pm.mealID
inner join FlightReservation r
on pm.ReservationID = r.ReservationID
WHERE m.MealID not in (pm.MealID)
```



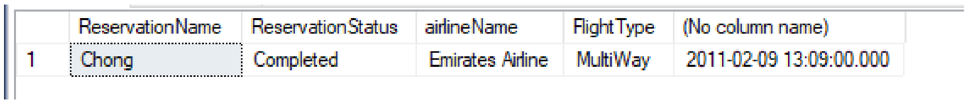
	MealName
1	Curry

The outcome displays the meals that were not requested by any passanger.

9.3.4 Query iv



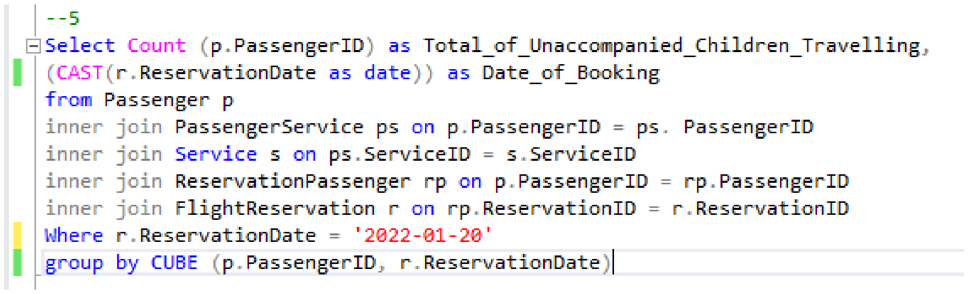
```
Select distinct r.ReservationName,
r.ReservationStatus,
air.airlineName,
r.FlightType,
(CAST(f.DepartureDateTime as datetime))
from FlightReservation r inner join ReserveFlight rf
on r.ReservationID = rf.ReservationID
inner join Flight f
on rf.FlightID = f.FlightID
inner join Aircraft a
on f.AircraftID = a.AircraftID
inner join Airline air
on a.AirlineID = air.AirlineID
WHERE r.FlightType = 'MultiWay' and f.DepartureDateTime = '2011-02-09 13:09:00';
```



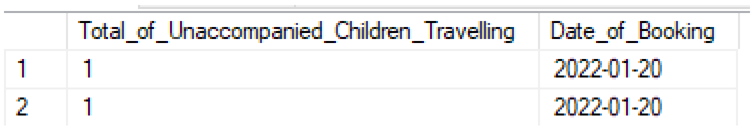
	ReservationName	ReservationStatus	airlineName	FlightType	(No column name)
1	Chong	Completed	Emirates Airline	MultiWay	2011-02-09 13:09:00.000

The findings reflect what was discovered by searching for precise flight type data (in this example, "multi-city") and the exact day and time the passenger booked.

9.3.5 Query v



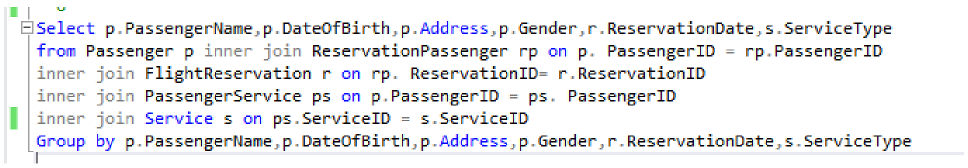
```
--5
Select Count (p.PassengerID) as Total_of_Unaccompanied_Children_Travelling,
(CAST(r.ReservationDate as date)) as Date_of_Booking
from Passenger p
inner join PassengerService ps on p.PassengerID = ps. PassengerID
inner join Service s on ps.ServiceID = s.ServiceID
inner join ReservationPassenger rp on p.PassengerID = rp.PassengerID
inner join FlightReservation r on rp.ReservationID = r.ReservationID
Where r.ReservationDate = '2022-01-20'
group by CUBE (p.PassengerID, r.ReservationDate)
```



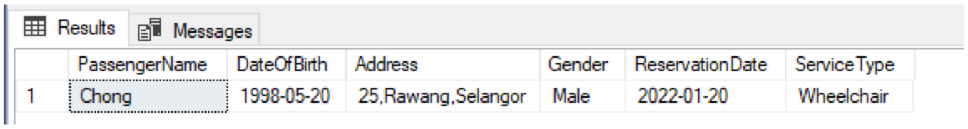
	Total_of_Unaccompanied_Children_Travelling	Date_of_Booking
1	1	2022-01-20
2	1	2022-01-20

The results show the total number of children who travelled alone on a certain date. Two people are the result.

9.3.6 Query vi



```
Select p.PassengerName,p.DateOfBirth,p.Address,p.Gender,r.ReservationDate,s.ServiceType
from Passenger p inner join ReservationPassenger rp on p. PassengerID = rp.PassengerID
inner join FlightReservation r on rp. ReservationID= r.ReservationID
inner join PassengerService ps on p.PassengerID = ps. PassengerID
inner join Service s on ps.ServiceID = s.ServiceID
Group by p.PassengerName,p.DateOfBirth,p.Address,p.Gender,r.ReservationDate,s.ServiceType
```



	PassengerName	DateOfBirth	Address	Gender	ReservationDate	ServiceType
1	Chong	1998-05-20	25,Rawang,Selangor	Male	2022-01-20	Wheelchair

This query will provide the number of passengers that are receiving additional service on their flight.

References

Evans, L. (n.d.). *SisenseBlog*. Retrieved from SisenseBlog: https://www.sisense.com/blog/8-ways-fine-tune-sql-queries-production-databases/

Samantary, S. (2017, April 20). *C#Corner*. Retrieved from C#Corner: https://www.c-sharpcorner.com/blogs/instead-of-delete-triggers-and-view#:~:text=INSTEAD%20OF%20DELETE%20TRIGGERS%20are,delete%20records%20from%20a%20view.&text=INSTEAD%20OF%20DELETE%20triggers%20are%20used%20to%20delete%20records%20from,is%20based%20on%20

W3schools. (n.d.). *W3schools SQL*. Retrieved from W3schools SQL: https://www.w3schools.com/sql/default.asp