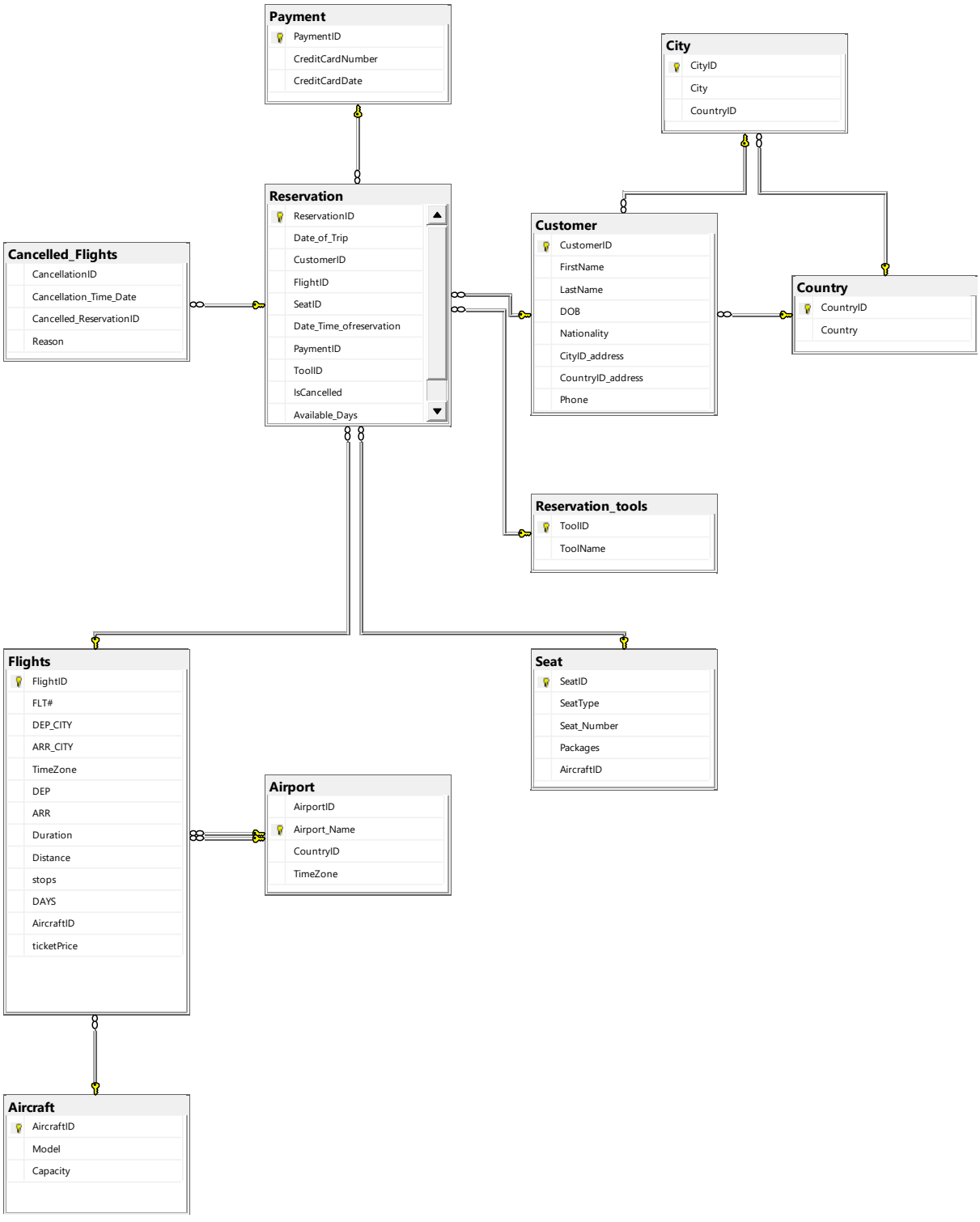
IV. Database Model

1. Diagram



2. Columns

3. Data entry

In order to simulate real-life scenarios as much as possible, so as to get some tangible insight from the database and queries, it was decided to feed our database real life data. To do so, a link was established between a data source found on www.mea.com with our database. The import process was adjusted so that each column from the source file is imported into the corresponding table column on our database.

4. Data Cleaning

SQL is a powerful to maintain a clean and well-structured data. We used some queries in order to clean the messy imported data so that it better serves our business model. The following cleaning procedures were undergone through SQL queries:

- Separating entries into two columns for further analysis. (Eg: ATHENS (ATH)+3 was separated into ATHENS (ATH) and +3 in order to identify the time zone of the airport).

```
UPDATE Sheet1$ SET TimeZone=SUBSTRING(CITY,CHARINDEX(')', CITY) + 1,LEN(CITY))
from Sheet1$;
UPDATE Sheet1$ SET TimeZone= 0 where TimeZone='UTC'
UPDATE Sheet1$ SET TimeZone= 0 where TimeZone=' UTC'
UPDATE Sheet1$ SET TimeZone=TimeZone-3
```

- The duration of the flight was deducted arithmetically by deducting the arrival time from the departure time and then factoring in the time zone difference found from the previous step.

```
UPDATE Sheet1$ SET Duration= DATEDIFF ( MINUTE , DEP , ARR )-60*TimeZone from
Flights
UPDATE Sheet1$ SET Duration= Duration+1440 where Duration<0;
```

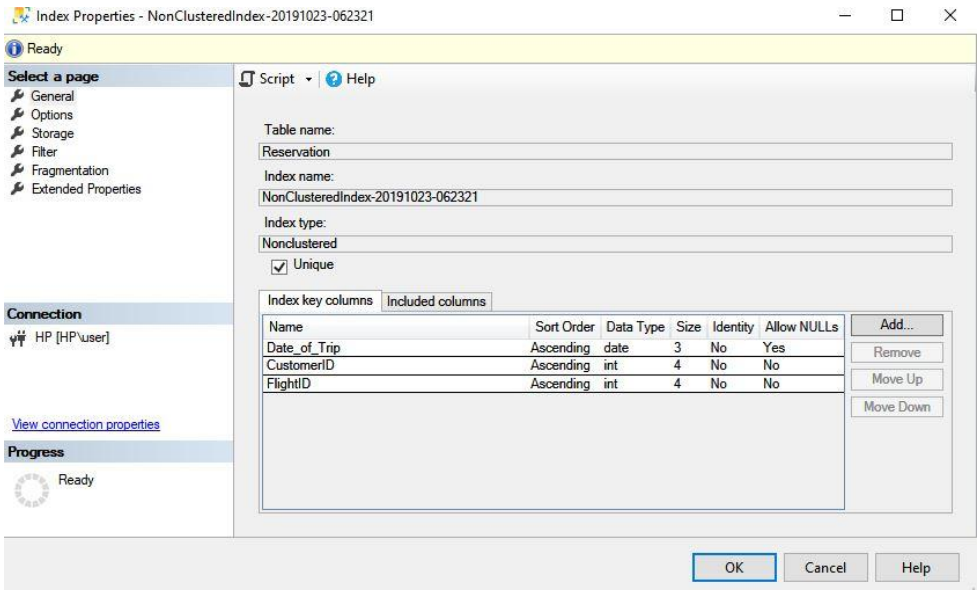
- Reservations that have been cancelled are to be identified by changing its corresponding entry in the isCancelled column to 1. The following query will copy the details of this reservation into a designated table called Cancelled Flights.

```
--Moving Reservation Data of Cancelled Flight into Cancelled_Flights Table
USE Airline_Reservation
INSERT INTO Cancelled_Flights(Cancelled_ReservationID)
SELECT ReservationID
FROM Reservation
WHERE Reservation.IsCancelled=1
```

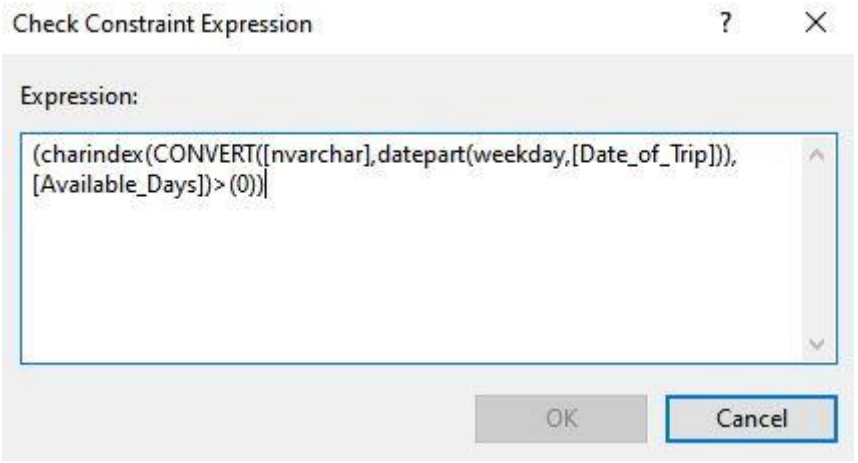
V. Database

I. Constraints

1) Non-Clustered Index: A non-clustered index was created to ensure that a customer cannot be stored as booking the same flight on the same day twice.



2) The date of departure(Date_of_Trip) cannot be a day where the flight is not offered.(i.e. if a flight is only available on Mondays, one cannot reserve a flight on 23/10/2019, which is actually a Wednesday). So, the following constraint was imposed:



DATEPART is an SQL function which return the weekday of the inputted date. Hence, when inputed a Date_of_Trip, the query will check if the corresponding Week Day is present in the string of offered flights

ReservationID	Date_of_Trip	CustomerID	FlightID	SeatID	Date_Time_ofr...	PaymentID	ToolID	IsCancelled	Available_Days
1	2019-11-11	1	12	NULL	<Binary data>	2	2	0	146
7	2019-11-11	3	1	NULL	<Binary data>	2	2	0	25
8	2019-05-06	2	1	NULL	<Binary data>	2	2	0	25
9	2019-07-07	4	22	NULL	<Binary data>	2	2	0	1234567
10	2019-11-04	2	12	NULL	<Binary data>	2	2	0	146
15	2019-05-05	1	17	NULL	<Binary data>	3	2	0	257
17	2019-08-08	1	19	NULL	<Binary data>	2	2	0	574
22	2019-10-10	4	1	NULL	<Binary data>	1	2	0	25
28	2019-07-07	11	3	NULL	<Binary data>	2	2	0	1234567

2019-10-10 was allowed as a Date_of_Trip entry only because this day happens to be a Thursday (5) which is among the list of Available days for this specific flight (Flight ID 1 is available on Mondays and Thursdays).

NOTE: Run this code to update Available_Days Column as per the corresponding entered Flight ID

```
USE Airline_Reservation
update Reservation
set Available_Days = Flights.DAYS
from Reservation inner join Flights on Reservation.FlightID = Flights.FlightID
```

2. Queries for Analysis

1) Flights per Day

```
--(1)Number of Flights From Beirut per Day
USE Airline_Reservation
Select ARR_CITY,DAYS from Flights where DAYS like '%1%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%2%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%3%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%4%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%5%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%6%' group by ARR_CITY,DAYS
Select ARR_CITY,DAYS from Flights where DAYS like '%7%' group by ARR_CITY,DAYS
```

Results Messages

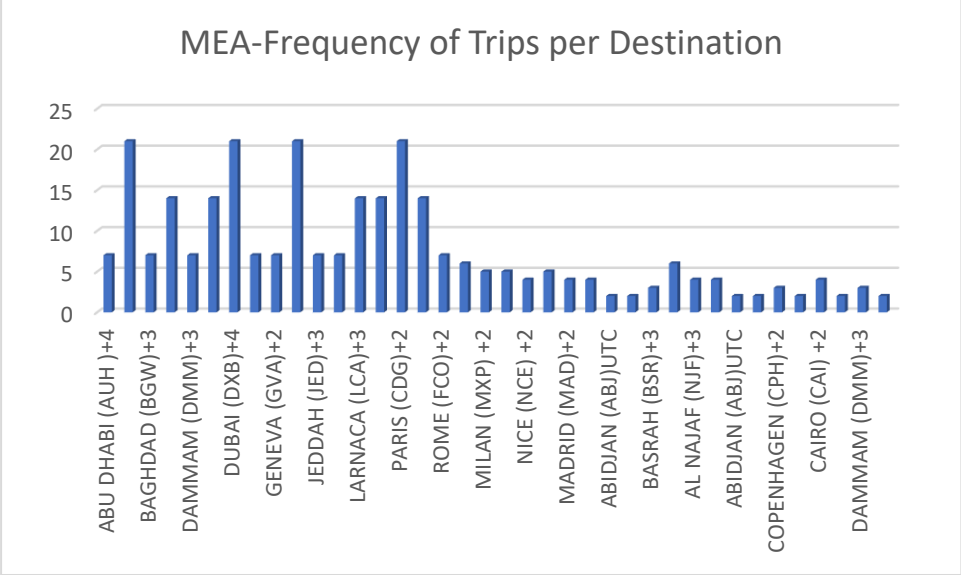
ARR_CITY	DAYS
ABIDJAN (ABJ)UTC	14
ABU DHABI (AUH)+4	1234567
ACCRA (ACC) UTC	14
AMMAN (AMM)+3	1234567
ATHENS (ATH)+3	1357
BAGHDAD (BGW)+3	1234567
BASRAH (BSR)+3	146
BRUSSELS (BRU) +2	12367
CAIRO (CAI) +2	1234567
DAMMAM (DMM)+3	1234567
DOHA (DOH) +3	1234567
DUBAI (DXB)+4	1234567

Query executed successfully. | HP (14.0 RTM) | HP\user (61) | Airline_Reservation | 00:00:00 | 27 rows

27 flights on Monday

2) Frequency of Trips per Destination

```
Select  ARR_CITY,SUM(LEN(DAYS)) from Flights  group by ARR_CITY,DAYS
```



3) Frequency of Trabelers per Nationality

```
USE Airline_Reservation
SELECT c.Nationality,count(c.Nationality) as Total
FROM Customer as c
RIGHT JOIN Reservation as r
    on r.CustomerID = c.CustomerID
GROUP BY c.Nationality
ORDER BY Total DESC;
```

	Nationality	Total
1	Lebanese	4
2	Egyptian	3
3	American	2

4) Load Factor: Percentage of occupied seats to total capacity

```
USE Airline_Reservation
SELECT f.FLT#,r.Date_of_Trip,count(r.FlightID)*100/(ac.Capacity) as SeatOccupancyRate
FROM Aircraft as ac
RIGHT JOIN Flights as f
    on f.AircraftID = ac.AircraftID
RIGHT JOIN Reservation as r
    on f.FlightID = r.FlightID
GROUP BY f.FLT# ,r.Date_of_Trip,Capacity
ORDER BY SeatOccupancyRate;
```

	FLT#	Date_of_Trip	SeatOccupancyRate
1	ME426	2019-07-07	0
2	ME444	2019-08-08	1
3	ME571	2019-05-06	1
4	ME571	2019-10-10	1
5	ME571	2019-11-11	1
6	ME225	2019-05-05	1
7	ME332	2019-11-04	1
8	ME418	2019-07-07	1
9	ME332	2019-11-11	4

You can try to decrease capacity to have more interesting Occupancy Rate

5) Passenger yield is an important parameter when it comes to measuring profitability of a line of service. NB: all ticket prices were assumed to be 500$.

```
--(5) Passenger yield: revenue earned for each passenger kilometer flown
USE Airline_Reservation
SELECT f.FLT#,SUM(f.ticketPrice)/(COUNT(r.ReservationID)*f.Distance) as
Passenger_Yield
FROM  Flights as f
RIGHT JOIN Reservation as r
      on f.FlightID = r.FlightID
GROUP BY f.FLT#,f.Distance
ORDER BY Passenger_Yield DESC;
```

	FLT#	Passenger_Yield
1	ME3...	0.410509
2	ME4...	0.306936
3	ME4...	0.237416
4	ME4...	0.233863
5	ME2...	0.166944
6	ME5...	0.096861

6) Compiling all data into one table

```
--(6)FirstName,LastName,DOB,CountryID_address
SELECT *

FROM Customer
Full outer JOIN Reservation
      on Customer.CustomerID = Reservation.CustomerID
   Full Outer JOIN Flights
    on Flights.FlightID = Reservation.FlightID
```

	CustomerID	FirstName	LastName	DOB	Nationality	CityID_address	CountryID_address	Phone	ReservationID	Date_of_Trip	CustomerID	FlightID	SeatID
1	2	Kamel	Khoury	1995-10-...	American	1	2	36543...	8	2019-05-06	2	1	NULL
2	3	Dani	Sami	1986-06-...	Lebane...	2	1	33652...	7	2019-11-11	3	1	NULL
3	4	Hiba	Salim	2000-03-...	Egyptian	3	2	34152...	22	2019-10-10	4	1	NULL
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	11	Shadi	Najjar	1966-03-...	Egyptian	3	2	33341...	28	2019-07-07	11	3	NULL
6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
9	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7) Most visited country: to adjust flight operations accordingly

```
--(7)most visted Country
SELECT Flights.ARR_CITY, Count (ReservationID) as '#of visitors'
FROM Reservation
INNER JOIN Flights ON Reservation.FlightID=Flights.FlightID
 INNER JOIN Airport ON Flights.ARR_CITY=Airport.Airport_Name
group By ARR_CITY,Airport_Name
```

	ARR_CITY	#of visitors
1	ABIDJAN (ABJ)UTC	3
2	ABU DHABI (AUH)+4	1
3	BASRAH (BSR)+3	5
4	COPENHAGEN (CPH)...	1
5	DAMMAM (DMM)+3	1
6	DUBAI (DXB)+4	1

8) Which platform are users using to reserve?

```
--(8) reservation tool used
SELECT Reservation_tools.ToolName,Count (ReservationID) as '#of customer'
FROM Reservation
INNER JOIN Reservation_tools ON Reservation.ToolID=Reservation_tools.ToolID
group By ToolName
```

9) As an airline, we are interested to know how much distance has an aircraft covered so far in order to notify technical teams to perform periodic mechanical check-ups.

```
USE Airline_Reservation
SELECT  ac.AircraftID, f.FLT# , f.Distance* count( DISTINCT r.Date_of_Trip) as
KMs_Travelled
FROM Aircraft as ac
RIGHT JOIN Flights as f
    on f.AircraftID = ac.AircraftID
RIGHT JOIN Reservation as r
 on f.FlightID = r.FlightID
GROUP BY ac.AircraftID,f.FLT# ,f.Distance
ORDER BY KMs_Travelled DESC;
```

	AircraftID	FLT#	KMs_Travelled
1	13	ME5...	15486
2	4	ME2...	2995
3	8	ME3...	2436
4	18	ME4...	2138
5	1	ME4...	2106
6	7	ME4...	1629