DATA-DEFINITION SQLs TO CREATE RELATIONAL TABLES

```
CREATE TABLE Airline (
airline_ID char(2),
airline_name varchar(100),
primary key (airline_ID));

CREATE TABLE Airport (
airport_ID char(3),
airport_name varchar(100),
city varchar(100),
country varchar (40),
primary key (airport_ID));

CREATE TABLE Flight (
airline_ID char(2),
flight_no int,
num_of_seats int,
num_of_stops int,
working_days char(7),
stop_no int,
depart_from char(3),
depart_time time,
arrive_to char(3),
arrive_time time,
arrive_day int,
fare numeric(10,2),
fare_restriction varchar(100),
primary key (airline_ID, flight_no, stop_no),
foreign key (airline_ID) references Airline(airline_ID));


CREATE TABLE Account (
account_no int,
account_type char(1), /*  'C' - customer, 'M' - manager, 'R' - customer rep
account_create_date date,
pword varchar(20),
primary key (account_no)
);

CREATE TABLE Employee (
account_no int,
```

```
account_type char(1),
employee_no varchar(20),
primary key (employee_no),
foreign key (account_no) references Account (account_no));

CREATE TABLE Customer (
account_no int,
first_name varchar(30),
last_name varchar(30),
address varchar(30),
city varchar(30),
state varchar(2),
zip_code varchar(5),
telephone varchar(10),
email varchar(30),
credit_card_no char(16),
preferences varchar(50),
/* we can indicate 'window', 'vegetarian' etc within preferences, and then search whether
preferences contains the specific terms */
primary key (account_no),
foreign key (account_no) references Account (account_no)
);

CREATE TABLE Reservation (
account_no int,
reservation_no int,
reservation_date date,
reservation_status varchar(20),
num_legs int,
passengers varchar(100),
fare_restrictions varchar(100),
total_fare numeric(10,2),
booking_fee numeric(10,2),
customer_rep varchar(20),
/* indicate customer_rep by employee_no?*/
primary key (reservation_no),
foreign key (customer_rep) references Employee (employee_no),
foreign key (account_no) references Customer (account_no)
);


CREATE TABLE Leg (
reservation_no int,
```

airline_id char(2),
flight_no int,
stop_no int,
leg_no int,
Dept_date date,
Arr_date date,
seat_num int,
class varchar(10),
meal varchar(10),
primary key (reservation_no, leg_no),
foreign key (reservation_no) references Reservation (reservation_no) ,
foreign key (airline_ID, flight_no, stop_no) references flight (airline_ID, flight_no, stop_no)
);

CUSTOMER-LEVEL FUNCTIONALITY

1. Make flight reservations: One-Way/ Round Trip / Multi-city / Domestic / International / Flexible Date time

When a customer makes a reservation from the website, the following values will be passed from the front-end UI:

ws_account_no
ws_reservation_no
ws_num_legs
ws_passengers
ws_fare_restrictions
ws_booking_fee
ws_customer_rep
ws_airline_id
ws_flight_no
ws_stop_no
ws_leg_no
ws_dept_date
ws_arr_date
ws_seat_num
ws_class
ws_meal

Values to be calculated before inserting into reservation table:

ws_reservation_date = select NOW()
ws_next_reservation_no = (select max(reservation_no) from Reservation) + 1
ws_reservation_status = ‘A’ (for active)
ws_total_fare = 0 (this will be updated into table after loading the reservation and legs table)

INSERT INTO RESERVATION VALUES (ws_account_no,
ws_reservation_no,
ws_reservation_date,
ws_reservation_status
ws_num_legs,
ws_passengers,
ws_fare_restrictions,
ws_booking_fee,
ws_total_fare,
ws_customer_rep,
ws_total_fare
)

If reservation has 3 legs (for example), 3 records will be inserted into LEG table.

INSERT INTO LEG VALUES(
ws_reservation_no,
ws_airline_id,
ws_flight_no,
ws_stop_no,
ws_leg_no,
ws_dept_date,
ws_arr_date,
ws_seat_num,
ws_class,
ws_meal,
)

Total fare calculation (ws_total_fare):

select sum(f.fare), r.fare_restriction
into ws_fare, ws_restriction
From flight f, leg L, reservation r
where L.reservation_no = “reservation_no”
And L.reservation_no = r. reservation_no
And L. airline_id = f. airline_id

```
And L.flight_no = f. flight_no
And L.stop_no = f.stop_no;
```

Below code applies various discounts / promotions based on fare_restriction:

```
If ws_restriction = 'A' (say it mean 20% discount)
        Discounted_fare = ws_fare * 0.8
Elseif ws_restriction = 'B' (flat $50 discount)
        Discounted_fare = ws_fare - 50
Else
        Discounted_fare = ws_fare
```

SQL to update total reservation fare in reservation table:

```
UPDATE RESERVATION
SET total_fare = discounted_fare
WHERE reservation_no = ws_reservation_no;
```

2. Cancel flight reservation

When a customer cancels a reservation from the website, the following values will be passed from the front-end UI:

```
ws_account_no
ws_reservation_no

UPDATE RESERVATION
SET reservation_status = 'C'
WHERE reservation_no = ws_reservation_no
AND account_no = ws_account_no;
```

Reservation_status is an indicator field in the RESERVATION table to indicate whether the reservation has been 'cancelled' or stayed 'active'.

3. Retrieve Current reservations

When a customer logs into his account and select the option to view his active reservations, the following value will be passed from the front-end UI to the back-end: ws_account_no

```
SELECT reservation_no, reservation_date, total_fare, customer_rep
FROM RESERVATION
```

WHERE account_no = ws_account_no and reservation_status = ‘A’;

If customer requires more information about a given reservation, he will select the ‘travel itinerary’ option for a that reservation.
System will be setup to ensure that customer sees reservations only under his own account and is not able to access another customer’s reservations.

4. Travel itinerary for a given reservation

Upon selecting the ‘Travel Itinerary’ option for a given reservation, the below SQL statements will execute.

```
SELECT  r.reservation_no, r.reservation_date, r.passengers, r.total_fare, L.airline_id,
L.flight_no, L.leg_no, L.seat_num, L.class, L.meal, f.depart_from, L.depart_date, f.depart_time,
f.arrive_to, (L.depart_date + f.arrive_day) as Arrive_date, f.arrive_time
FROM reservation r, leg L, flight f
WHERE r.reservation_no = ws_reservation_no
AND r.reservation_no =  L.reservation_no
AND L. airline_id = f. airline_id
AND L.flight_no = f. flight_no
AND L.stop_no = f.stop_no;
```

5. History of all past and current reservations

```
SELECT R.reservation_no, R.reservation_status, R.reservation_date, R.passengers,
R.total_fare, L.airline_id, L.flight_no, L.leg_no, L.seat_num, L.class, L.meal, F.depart_from,
L.depart_date, F.depart_time, F.arrive_to, (L.depart_date + F.arrive_day) as Arrive_date,
F.arrive_time
FROM reservation R, leg L, flight F
WHERE R.account_no = ws_account_no
AND R.reservation_no =  L.reservation_no
And L. airline_id = F. airline_id
And L.flight_no = F. flight_no
And L.stop_no = F.stop_no;
```

6. Best-seller list of flights (TOP 10)

```
SELECT T1. airline_id, T1.flight_no, T1. num_of_times_reserved
FROM (SELECT airline_id, flight_no, count(distinct(reservation_no)) as num_of_times_reserved
        FROM Leg L, reservation R
        WHERE L.reservation_no = R.reservation_no
        AND R.reservation_status not = ‘C’
```

```
        GROUP BY airline_id, flight_no) T1
ORDER BY T1. num_of_times_reserved DESC
LIMIT 10;
```

MANAGER-LEVEL FUNCTIONALITY

The manager should be able to:

1. **Add, Edit and Delete information for a customer**

Manager will have access via User Interface to login into his account and view the account numbers and names of all the customers registered on the website.
Manager will be able to select an account number using customer's first and last name and will click on the 'delete' option next to account number.

Below SQL will execute on the backend: (**To delete a customer record**)

```
DELETE FROM CUSTOMER
WHERE account_no = ws_account_no;

DELETE FROM ACCOUNT
WHERE account_no = ws_account_no;
```

Similarly, when manager pulls up a customers record and edits particular field values, below SQL will update the data for customer: (**to update the customer record**)

```
UPDATE customer
SET ..
WHERE ..
```

SQL to add a customer:

```
ws_next_account_no = (SELECT MAX(account_no) from account) + 1
ws_account_type = 'C'  /* for customer
ws_account_create_date = SELECT NOW()
ws_pword = 'default12'

INSERT INTO account VALUES (ws_next_account_no, ws_account_type,
ws_account_create_date, ws_pword);

INSERT INTO customer VALUES (
account_no,
first_name,
last_name,
address,
city,
state,
zip_code,
telephone,
email,
credit_card_no,
preferences);
```

2. Obtain a sales report for a particular month

Sales report will show the total revenue by airline:

```
SELECT  A.airline_id, A.airline_name, SUM(R.total_fare)
FROM airline A, reservation R, (SELECT DISTINCT airline_id, reservation_no FROM leg
WHERE reservation_status != 'C') L
WHERE A.airline_id = L.airline_id
AND L.reservation_no = R.reservation_no
GROUP BY A.airline_id;
```

3. Produce a comprehensive listing of all flights

```
SELECT *
FROM flight;
```

4. Produce a list of reservations by flight number or by customer name

If manager inquires using the flight number (ws_flight_no):

```
SELECT DISTINCT R.reservation_no, R.reservation_date, R.total_fare
FROM RESERVATION R, LEG L
WHERE L.flight_no = ws_flight_no
AND L.reservation_id = R.reservation_id
AND R.reservation_status = ‘A’;
```

If manager inquires using the customer name:

```
SELECT DISTINCT R.reservation_no, R.reservation_date, R.total_fare
FROM RESERVATION R, CUSTOMER C
WHERE C.first_name  = ws_first_name
AND C.last_name  = ws_last_name
AND C.account_no = R.account_no
AND R.reservation_status = ‘A’;
```

5. **Summary Listing of Revenue Generated by Flight, Destination City or Customer**

```
DELIMITER //
CREATE PROCEDURE getSummary(
IN choice char(1)
)
```

```
// When CHOICE = ‘I’, Code for selecting summary listing by flight

SELECT L.airline_id, L.flight_no, SUM(R.booking_fee / F.num_of_stops) as revenue
FROM Reservation R, Leg L, flight F
WHERE
        L.flight_id = ws_flight_no
        AND L.airline_id = F.airline_id
        AND L.flight_no = F.flight_no
        AND L.reservation_no = R.reservation_no
        AND R.reservation_status != 'C'
GROUP BY airline_id, flight_no;
```

// When CHOICE = ‘D’, code for selecting summary listing by destination city

```
SELECT F.arrive_to, A.city,SUM(R.booking_fee /  F.num_of_stops) as revenue
FROM Reservation R
JOIN Leg L ON R.reservation_no = L.reservation_no
JOIN Flight F ON L.flight_no = F.flight_no AND L.airline_id = L.airline_id
JOIN Airport A ON F.arrive_to = A.airport_ID
WHERE R.reservation_status != 'C'
GROUP BY A.city;
```

// When CHOICE = ‘C’, code for selecting summary listing by customer

```
SELECT C.account_no, C.first_name, C.last_name, SUM(R.booking_fee) as revenue
FROM Customer C, Reservation R
WHERE
        C.account_no = R.account_no AND
   R.reservation_status != 'C'
GROUP BY C.account_no


END //
DELIMITER ;
```

6. Determine which customer generated most total revenue

```
SELECT C.account_no, sum(R.booking_fee) as revenue
FROM customer C, reservation R
WHERE R.reservation_status != 'C' and C.account_no = R.account_no
GROUP BY C.account_no
ORDER BY revenue DESC
LIMIT 1;
```

7. Produce a list of most active flights from the last few months - Manager selects number of months

```
DELIMITER //
CREATE PROCEDURE get_ActiveFlights(
   IN num_of_months int)
```

```
BEGIN
SELECT T1. airline_id, T1.flight_no, T1. num_of_times_reserved

FROM (SELECT airline_id, flight_no, count(distinct(reservation_no)) as num_of_times_reserved
                FROM Leg L, reservation R
   WHERE L.reservation_no = R.reservation_no
   AND R.reservation_status != 'C'
        AND L.Dept_date >= CURRENT_DATE - INTERVAL num_of_months MONTH
                GROUP BY airline_id, flight_no) T1
ORDER BY T1. num_of_times_reserved DESC
LIMIT 10;

END //
DELIMITER ;
```

8. Produce a list of all customers who have seats reserved on a given flight

```
DELIMITER //
CREATE PROCEDURE get_CustomersOnFlight(
   IN airline Char(2),
        IN flight int)

BEGIN
SELECT C.first_name, C.last_name, C.account_no
FROM Customer C, Reservation R, Leg L
WHERE
   C.account_no = R.account_no AND
        R.reservation_no =L.reservation_no AND
        L.airline_id = airline AND
        L.flight_no = flight AND
        L.seat_num IS NOT NULL;
END //
DELIMITER ;
```

9. Produce a list of all flights departing from a given airport

```
DELIMITER //
CREATE PROCEDURE get_FlightsAtAirport(
   IN airport char(3))
```

```
BEGIN
SELECT F.*
FROM Flights F
WHERE
    F.depart_from = airport;
END //
DELIMITER ;
```