Airline Booking System

Team 5 Members

Ardian Bryan Limasarian
Kiang Siong Boon
Lee Wee Lun
Lee Jian Ann
Stanley See Chong Hua

Overview

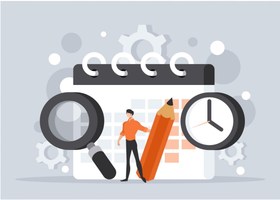
- Airline Booking System
- Entity Relationship
- Implementation of SQL
- Live Demonstration

Airline Booking System Features

Main Features

Flight Search

Flight Booking

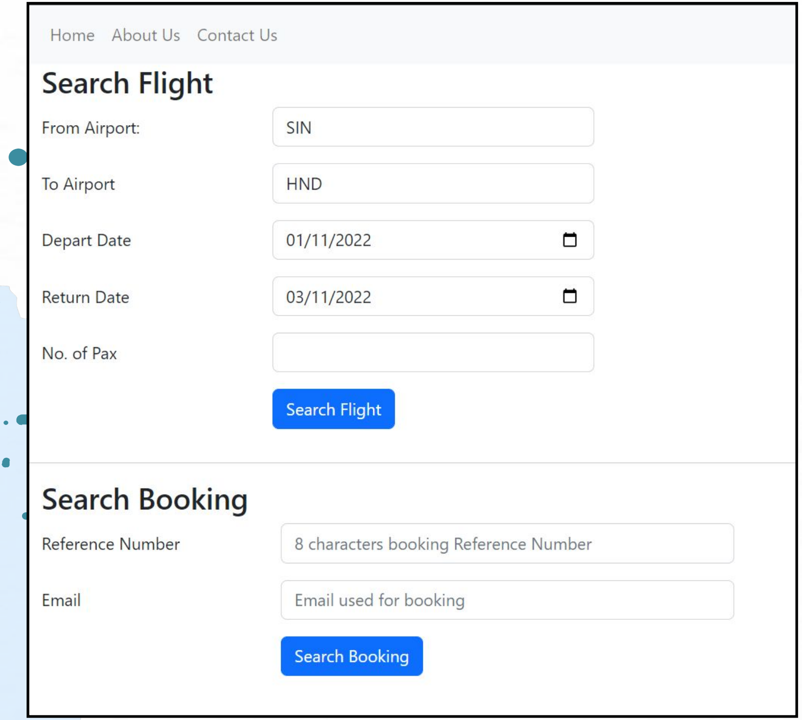


Manage Booking

Flight Administration

Booking Administration

Airline Booking System Interface

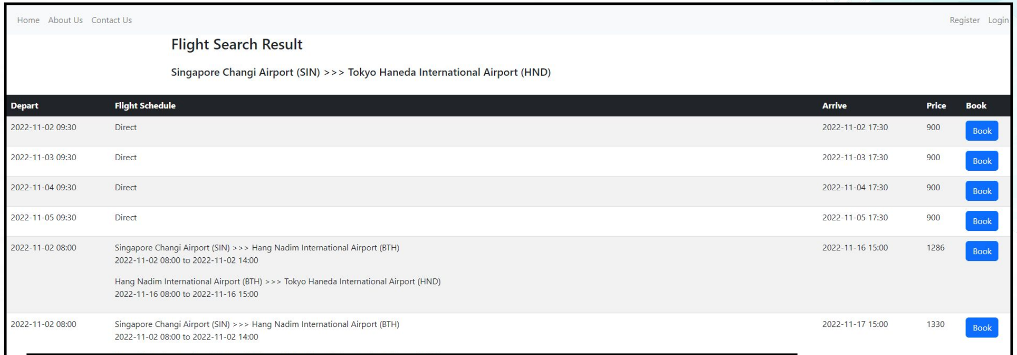


Search Flight

From Airport:	SIN
To Airport	HND
Depart Date	01/11/2022
Return Date	03/11/2022
No. of Pax	

Search Booking

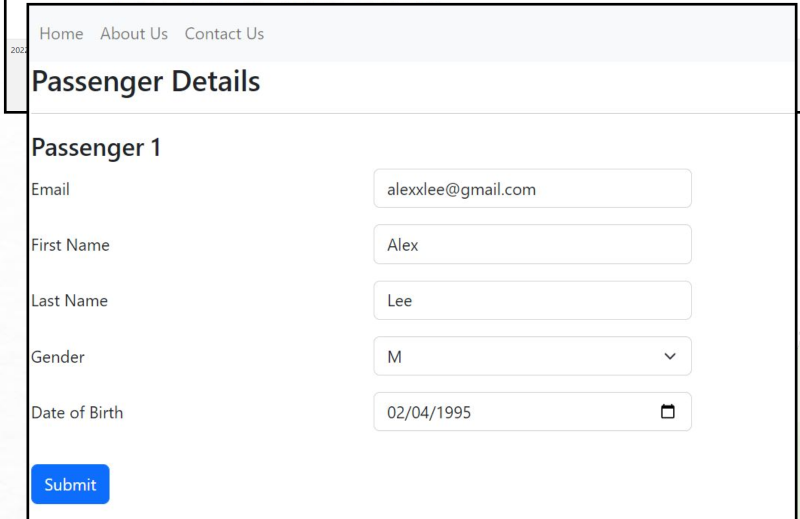
Reference Number	8 characters booking Reference Number
Email	Email used for booking



Flight Search Result

Singapore Changi Airport (SIN) >>> Tokyo Haneda International Airport (HND)

Depart	Flight Schedule	Arrive	Price	Book
2022-11-02 09:30	Direct	2022-11-02 17:30	900	Book
2022-11-03 09:30	Direct	2022-11-03 17:30	900	Book
2022-11-04 09:30	Direct	2022-11-04 17:30	900	Book
2022-11-05 09:30	Direct	2022-11-05 17:30	900	Book
2022-11-02 08:00	Singapore Changi Airport (SIN) >>> Hang Nadim International Airport (BTH) 2022-11-02 08:00 to 2022-11-02 14:00 Hang Nadim International Airport (BTH) >>> Tokyo Haneda International Airport (HND) 2022-11-16 08:00 to 2022-11-16 15:00	2022-11-16 15:00	1286	Book
2022-11-02 08:00	Singapore Changi Airport (SIN) >>> Hang Nadim International Airport (BTH) 2022-11-02 08:00 to 2022-11-02 14:00	2022-11-17 15:00	1330	Book
202		2022-11-18 15:00	1286	Book

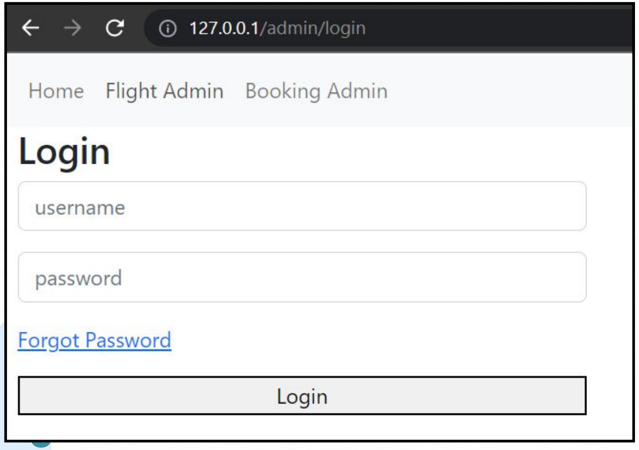


Passenger Details

Passenger 1

Email	alexxlee@gmail.com
First Name	Alex
Last Name	Lee
Gender	M
Date of Birth	02/04/1995

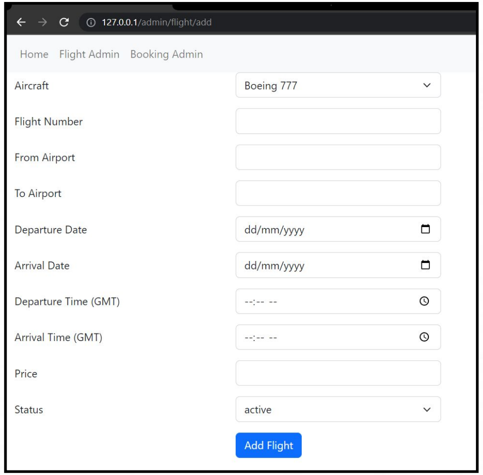
Airline Booking System Interface



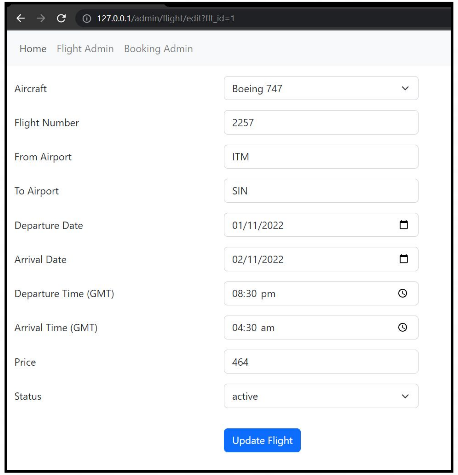
Login

username

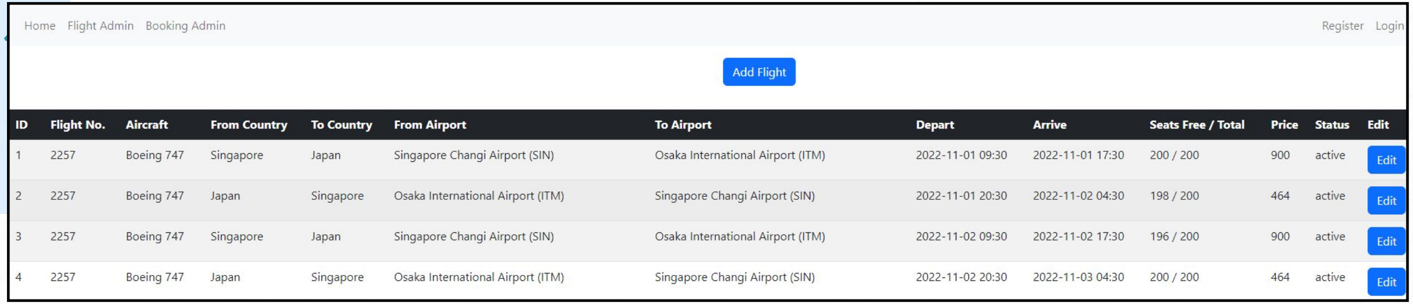
password



Aircraft	Boeing 777
Flight Number	
From Airport	
To Airport	
Departure Date	dd/mm/yyyy
Arrival Date	dd/mm/yyyy
Departure Time (GMT)	--:-- --
Arrival Time (GMT)	--:-- --
Price	
Status	active



Aircraft	Boeing 747
Flight Number	2257
From Airport	ITM
To Airport	SIN
Departure Date	01/11/2022
Arrival Date	02/11/2022
Departure Time (GMT)	08:30 pm
Arrival Time (GMT)	04:30 am
Price	464
Status	active



ID	Flight No.	Aircraft	From Country	To Country	From Airport	To Airport	Depart	Arrive	Seats Free / Total	Price	Status	Edit
1	2257	Boeing 747	Singapore	Japan	Singapore Changi Airport (SIN)	Osaka International Airport (ITM)	2022-11-01 09:30	2022-11-01 17:30	200 / 200	900	active	Edit
2	2257	Boeing 747	Japan	Singapore	Osaka International Airport (ITM)	Singapore Changi Airport (SIN)	2022-11-01 20:30	2022-11-02 04:30	198 / 200	464	active	Edit
3	2257	Boeing 747	Singapore	Japan	Singapore Changi Airport (SIN)	Osaka International Airport (ITM)	2022-11-02 09:30	2022-11-02 17:30	196 / 200	900	active	Edit
4	2257	Boeing 747	Japan	Singapore	Osaka International Airport (ITM)	Singapore Changi Airport (SIN)	2022-11-02 20:30	2022-11-03 04:30	200 / 200	464	active	Edit

Entity Relationship

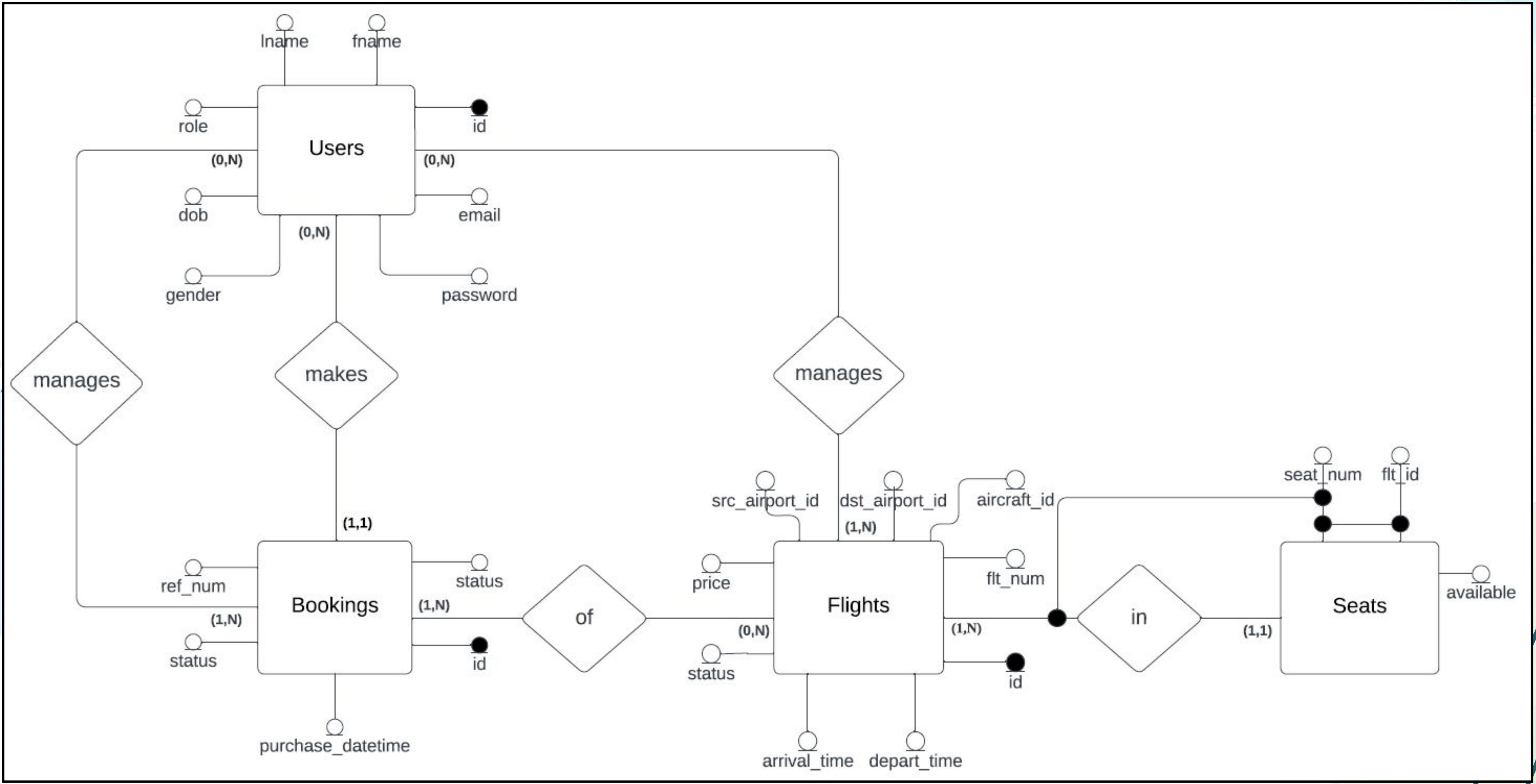
Database schema

- Bookings
- Users
- Flights
- Seats

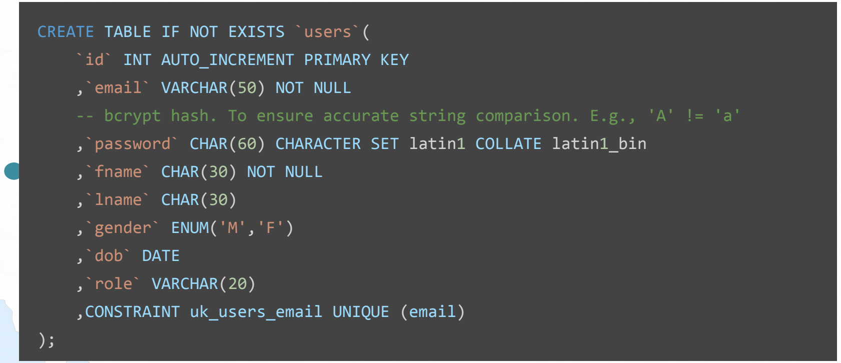
Relationship

- User - Make - Booking
- Booking - Of - Flights
- Seats - In - Flights
- User - Manage - own Booking
- User (Admin) - Manage - Bookings
- User (Admin) - Manage - Flights

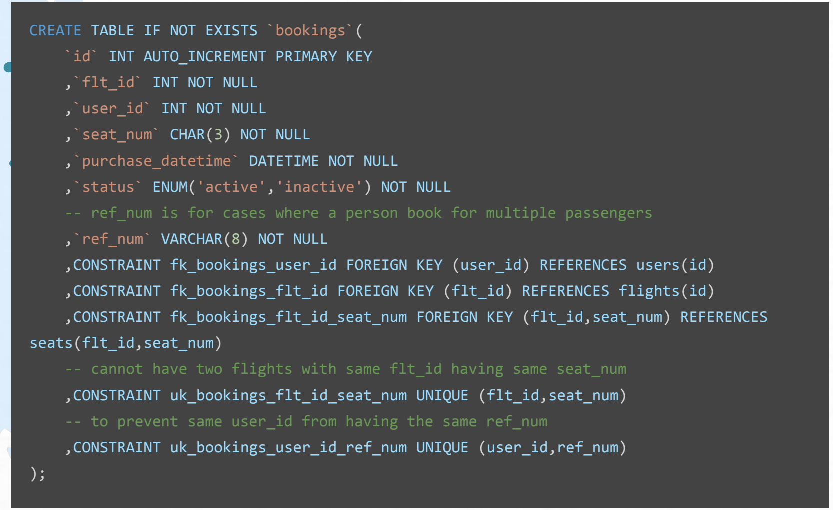
ER Diagram


lname
fname
role
id
Users
(0,N)
(0,N)
dob
email
(0,N)
gender
password
manages
makes
manages
src_airport_id
dst_airport_id
aircraft_id
seat_num
flt_id
(1,1)
(1,N)
ref_num
status
price
flt_num
available
Bookings
(1,N)
of
Flights
in
Seats
(1,N)
(0,N)
(1,N)
(1,1)
status
id
status
id
purchase_datetime
arrival_time
depart_time

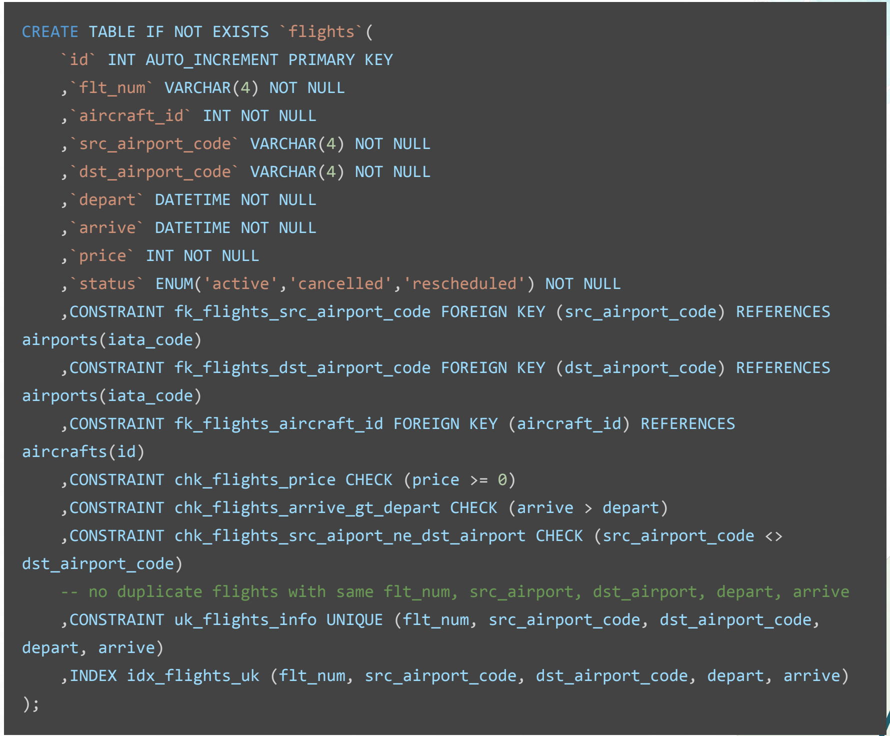
Implementation of SQL Tables



```sql
CREATE TABLE IF NOT EXISTS `users`(
    `id` INT AUTO_INCREMENT PRIMARY KEY
    ,`email` VARCHAR(50) NOT NULL
    -- bcrypt hash. To ensure accurate string comparison. E.g., 'A' != 'a'
    ,`password` CHAR(60) CHARACTER SET latin1 COLLATE latin1_bin
    ,`fname` CHAR(30) NOT NULL
    ,`lname` CHAR(30)
    ,`gender` ENUM('M','F')
    ,`dob` DATE
    ,`role` VARCHAR(20)
    ,CONSTRAINT uk_users_email UNIQUE (email)
);
```

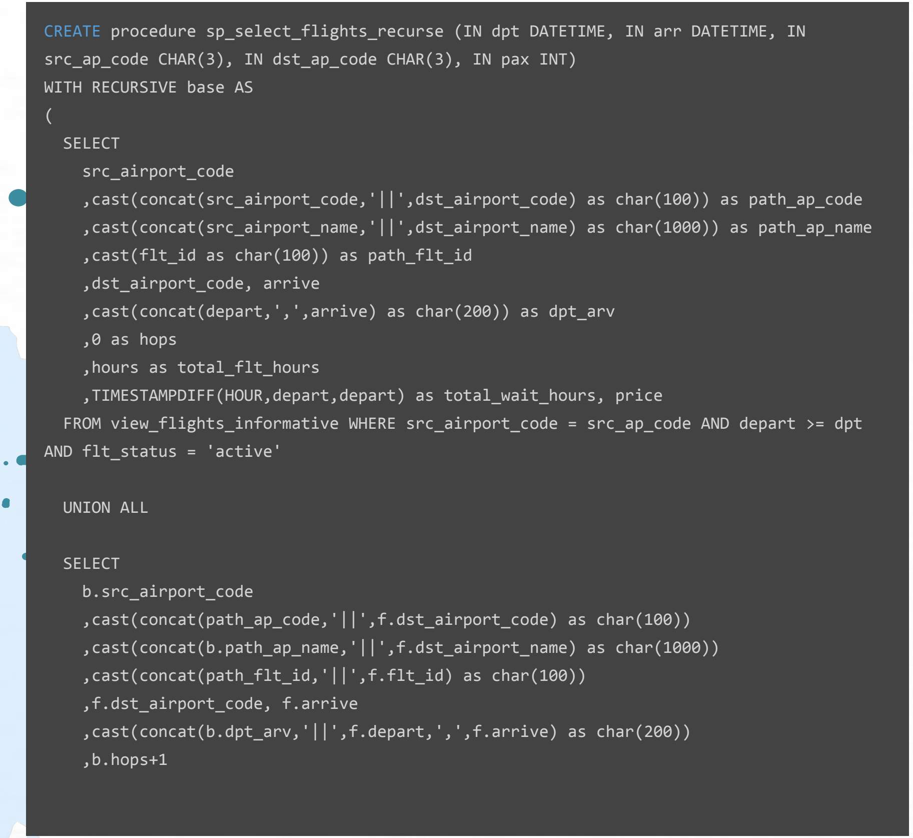


```sql
CREATE TABLE IF NOT EXISTS `bookings`(
    `id` INT AUTO_INCREMENT PRIMARY KEY
    ,`flt_id` INT NOT NULL
    ,`user_id` INT NOT NULL
    ,`seat_num` CHAR(3) NOT NULL
    ,`purchase_datetime` DATETIME NOT NULL
    ,`status` ENUM('active','inactive') NOT NULL
    -- ref_num is for cases where a person book for multiple passengers
    ,`ref_num` VARCHAR(8) NOT NULL
    ,CONSTRAINT fk_bookings_user_id FOREIGN KEY (user_id) REFERENCES users(id)
    ,CONSTRAINT fk_bookings_flt_id FOREIGN KEY (flt_id) REFERENCES flights(id)
    ,CONSTRAINT fk_bookings_flt_id_seat_num FOREIGN KEY (flt_id,seat_num) REFERENCES
seats(flt_id,seat_num)
    -- cannot have two flights with same flt_id having same seat_num
    ,CONSTRAINT uk_bookings_flt_id_seat_num UNIQUE (flt_id,seat_num)
    -- to prevent same user_id from having the same ref_num
    ,CONSTRAINT uk_bookings_user_id_ref_num UNIQUE (user_id,ref_num)
);
```

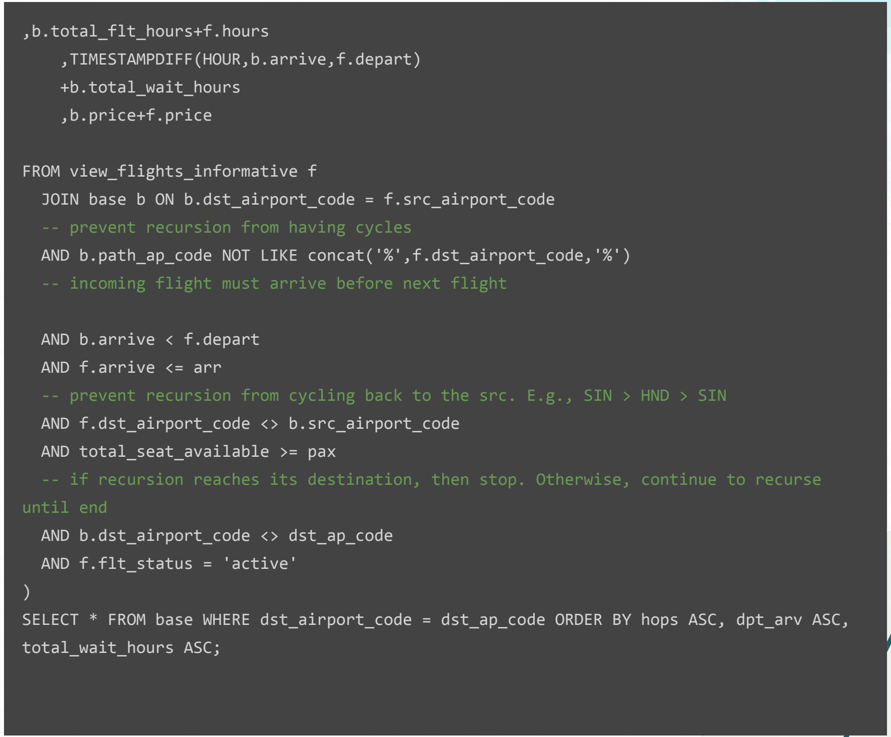


```sql
CREATE TABLE IF NOT EXISTS `flights`(
    `id` INT AUTO_INCREMENT PRIMARY KEY
    ,`flt_num` VARCHAR(4) NOT NULL
    ,`aircraft_id` INT NOT NULL
    ,`src_airport_code` VARCHAR(4) NOT NULL
    ,`dst_airport_code` VARCHAR(4) NOT NULL
    ,`depart` DATETIME NOT NULL
    ,`arrive` DATETIME NOT NULL
    ,`price` INT NOT NULL
    ,`status` ENUM('active','cancelled','rescheduled') NOT NULL
    ,CONSTRAINT fk_flights_src_airport_code FOREIGN KEY (src_airport_code) REFERENCES
airports(iata_code)
    ,CONSTRAINT fk_flights_dst_airport_code FOREIGN KEY (dst_airport_code) REFERENCES
airports(iata_code)
    ,CONSTRAINT fk_flights_aircraft_id FOREIGN KEY (aircraft_id) REFERENCES
aircrafts(id)
    ,CONSTRAINT chk_flights_price CHECK (price >= 0)
    ,CONSTRAINT chk_flights_arrive_gt_depart CHECK (arrive > depart)
    ,CONSTRAINT chk_flights_src_aiport_ne_dst_airport CHECK (src_airport_code <>
dst_airport_code)
    -- no duplicate flights with same flt_num, src_airport, dst_airport, depart, arrive
    ,CONSTRAINT uk_flights_info UNIQUE (flt_num, src_airport_code, dst_airport_code,
depart, arrive)
    ,INDEX idx_flights_uk (flt_num, src_airport_code, dst_airport_code, depart, arrive)
);
```

Implementation of Indirect Flight SQL

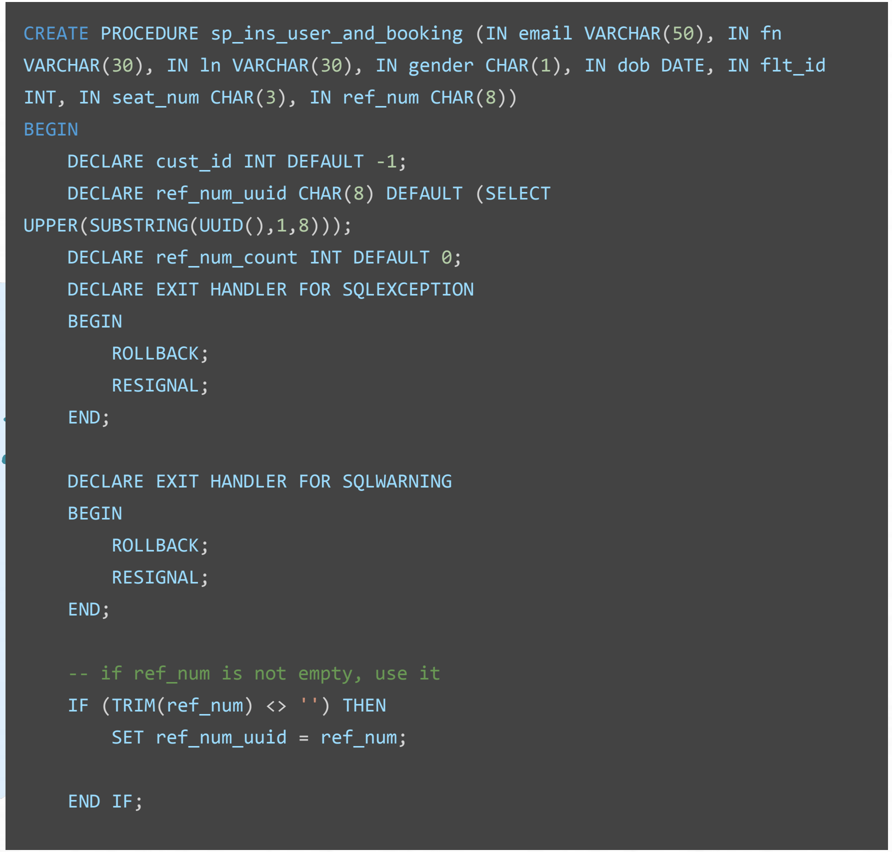


```
CREATE procedure sp_select_flights_recurse (IN dpt DATETIME, IN arr DATETIME, IN
src_ap_code CHAR(3), IN dst_ap_code CHAR(3), IN pax INT)
WITH RECURSIVE base AS
(
  SELECT
    src_airport_code
    ,cast(concat(src_airport_code,'||',dst_airport_code) as char(100)) as path_ap_code
    ,cast(concat(src_airport_name,'||',dst_airport_name) as char(1000)) as path_ap_name
    ,cast(flt_id as char(100)) as path_flt_id
    ,dst_airport_code, arrive
    ,cast(concat(depart,',',arrive) as char(200)) as dpt_arv
    ,0 as hops
    ,hours as total_flt_hours
    ,TIMESTAMPDIFF(HOUR,depart,depart) as total_wait_hours, price
  FROM view_flights_informative WHERE src_airport_code = src_ap_code AND depart >= dpt
AND flt_status = 'active'

  UNION ALL

  SELECT
    b.src_airport_code
    ,cast(concat(path_ap_code,'||',f.dst_airport_code) as char(100))
    ,cast(concat(b.path_ap_name,'||',f.dst_airport_name) as char(1000))
    ,cast(concat(path_flt_id,'||',f.flt_id) as char(100))
    ,f.dst_airport_code, f.arrive
    ,cast(concat(b.dpt_arv,'||',f.depart,',',f.arrive) as char(200))
    ,b.hops+1
```

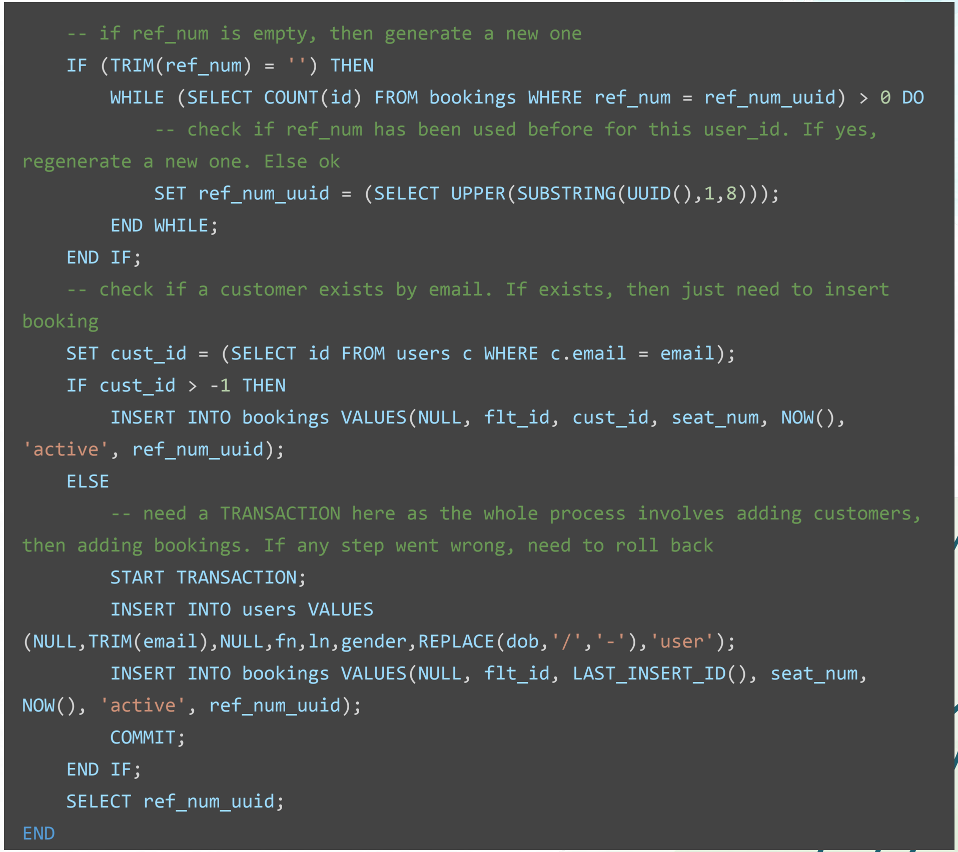


```
,b.total_flt_hours+f.hours
    ,TIMESTAMPDIFF(HOUR,b.arrive,f.depart)
    +b.total_wait_hours
    ,b.price+f.price

FROM view_flights_informative f
  JOIN base b ON b.dst_airport_code = f.src_airport_code
  -- prevent recursion from having cycles
  AND b.path_ap_code NOT LIKE concat('%',f.dst_airport_code,'%')
  -- incoming flight must arrive before next flight

  AND b.arrive < f.depart
  AND f.arrive <= arr
  -- prevent recursion from cycling back to the src. E.g., SIN > HND > SIN
  AND f.dst_airport_code <> b.src_airport_code
  AND total_seat_available >= pax
  -- if recursion reaches its destination, then stop. Otherwise, continue to recurse
until end
  AND b.dst_airport_code <> dst_ap_code
  AND f.flt_status = 'active'
)
SELECT * FROM base WHERE dst_airport_code = dst_ap_code ORDER BY hops ASC, dpt_arv ASC,
total_wait_hours ASC;
```

Implementation of Customer and Booking SQL

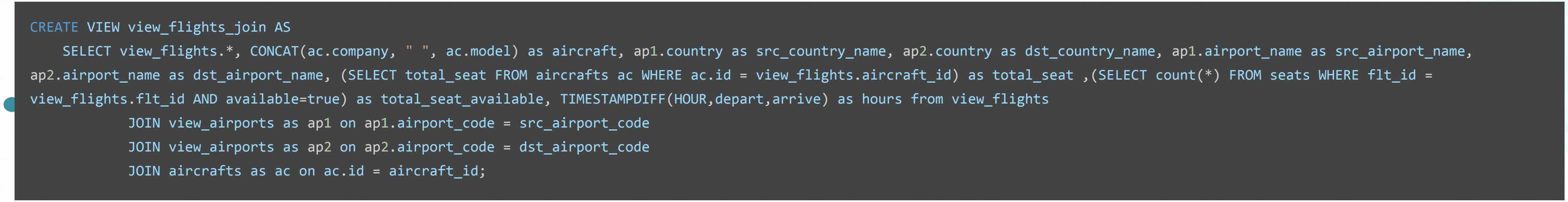


```
CREATE PROCEDURE sp_ins_user_and_booking (IN email VARCHAR(50), IN fn
VARCHAR(30), IN ln VARCHAR(30), IN gender CHAR(1), IN dob DATE, IN flt_id
INT, IN seat_num CHAR(3), IN ref_num CHAR(8))
BEGIN
    DECLARE cust_id INT DEFAULT -1;
    DECLARE ref_num_uuid CHAR(8) DEFAULT (SELECT
UPPER(SUBSTRING(UUID(),1,8)));
    DECLARE ref_num_count INT DEFAULT 0;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    DECLARE EXIT HANDLER FOR SQLWARNING
    BEGIN
        ROLLBACK;
        RESIGNAL;
    END;

    -- if ref_num is not empty, use it
    IF (TRIM(ref_num) <> '') THEN
        SET ref_num_uuid = ref_num;

    END IF;
```

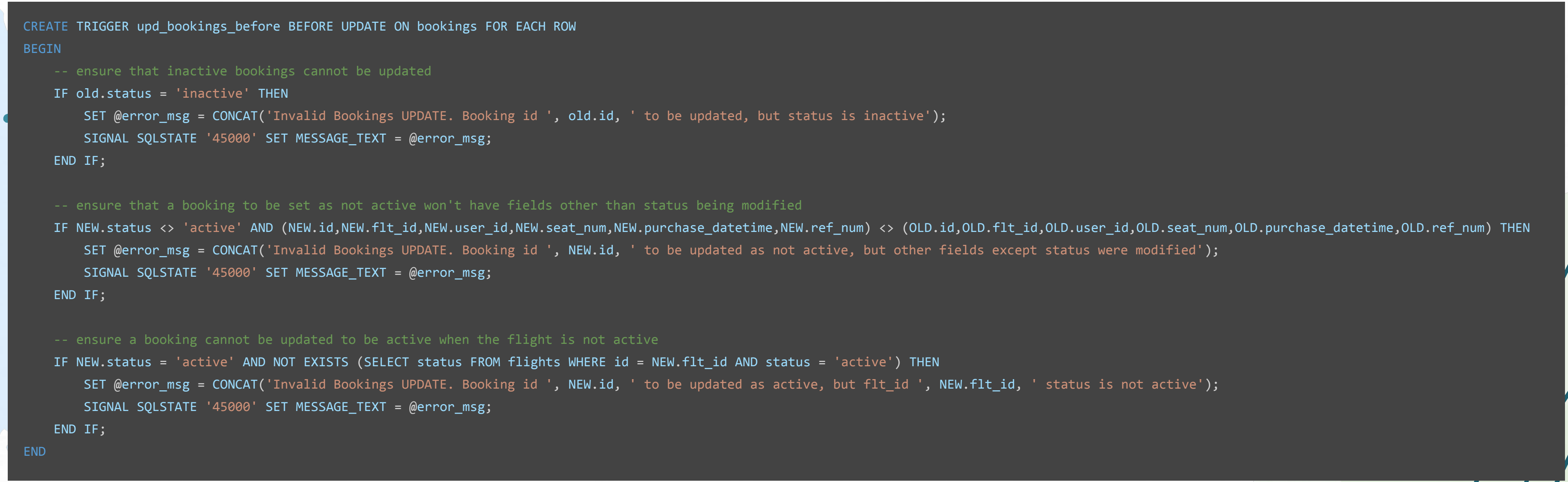


```
    -- if ref_num is empty, then generate a new one
    IF (TRIM(ref_num) = '') THEN
        WHILE (SELECT COUNT(id) FROM bookings WHERE ref_num = ref_num_uuid) > 0 DO
            -- check if ref_num has been used before for this user_id. If yes,
regenerate a new one. Else ok
            SET ref_num_uuid = (SELECT UPPER(SUBSTRING(UUID(),1,8)));
        END WHILE;
    END IF;
    -- check if a customer exists by email. If exists, then just need to insert
booking
    SET cust_id = (SELECT id FROM users c WHERE c.email = email);
    IF cust_id > -1 THEN
        INSERT INTO bookings VALUES(NULL, flt_id, cust_id, seat_num, NOW(),
'active', ref_num_uuid);
    ELSE
        -- need a TRANSACTION here as the whole process involves adding customers,
then adding bookings. If any step went wrong, need to roll back
        START TRANSACTION;
        INSERT INTO users VALUES
(NULL,TRIM(email),NULL,fn,ln,gender,REPLACE(dob,'/','-'),'user');
        INSERT INTO bookings VALUES(NULL, flt_id, LAST_INSERT_ID(), seat_num,
NOW(), 'active', ref_num_uuid);
        COMMIT;
    END IF;
    SELECT ref_num_uuid;
END
```

Implementation of Flight View and Constraints SQL



```sql
CREATE VIEW view_flights_join AS
    SELECT view_flights.*, CONCAT(ac.company, " ", ac.model) as aircraft, ap1.country as src_country_name, ap2.country as dst_country_name, ap1.airport_name as src_airport_name,
ap2.airport_name as dst_airport_name, (SELECT total_seat FROM aircrafts ac WHERE ac.id = view_flights.aircraft_id) as total_seat ,(SELECT count(*) FROM seats WHERE flt_id =
view_flights.flt_id AND available=true) as total_seat_available, TIMESTAMPDIFF(HOUR,depart,arrive) as hours from view_flights
            JOIN view_airports as ap1 on ap1.airport_code = src_airport_code
            JOIN view_airports as ap2 on ap2.airport_code = dst_airport_code
            JOIN aircrafts as ac on ac.id = aircraft_id;
```



```sql
CREATE TRIGGER upd_bookings_before BEFORE UPDATE ON bookings FOR EACH ROW
BEGIN
    -- ensure that inactive bookings cannot be updated
    IF old.status = 'inactive' THEN
        SET @error_msg = CONCAT('Invalid Bookings UPDATE. Booking id ', old.id, ' to be updated, but status is inactive');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @error_msg;
    END IF;

    -- ensure that a booking to be set as not active won't have fields other than status being modified
    IF NEW.status <> 'active' AND (NEW.id,NEW.flt_id,NEW.user_id,NEW.seat_num,NEW.purchase_datetime,NEW.ref_num) <> (OLD.id,OLD.flt_id,OLD.user_id,OLD.seat_num,OLD.purchase_datetime,OLD.ref_num) THEN
        SET @error_msg = CONCAT('Invalid Bookings UPDATE. Booking id ', NEW.id, ' to be updated as not active, but other fields except status were modified');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @error_msg;
    END IF;

    -- ensure a booking cannot be updated to be active when the flight is not active
    IF NEW.status = 'active' AND NOT EXISTS (SELECT status FROM flights WHERE id = NEW.flt_id AND status = 'active') THEN
        SET @error_msg = CONCAT('Invalid Bookings UPDATE. Booking id ', NEW.id, ' to be updated as active, but flt_id ', NEW.flt_id, ' status is not active');
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @error_msg;
    END IF;
END
```

Additional Implementation SQL

Top 10 Popular Travel Destinations

```
SELECT COUNT(booking_id) AS Bookings, dst_country_name AS Destination FROM view_bookings_join GROUP BY
dst_country_name ORDER BY Bookings DESC LIMIT 10;
```

Number of Bookings by Season

```
SELECT COUNT(b.id), FLOOR((MONTH(b.purchase_datetime) % 12) / 3) AS season FROM bookings b GROUP BY
season;
```

Top Revenue by Destination per Year

```
SELECT SUM(price) AS Revenue, dst_country_name AS Destination, YEAR(purchase_datetime) AS Year FROM
view_bookings_join vbj1 GROUP BY Year, Destination HAVING SUM(price) >= ALL (SELECT sum(vbj2.price)
FROM view_bookings_join vbj2 WHERE Year = YEAR(vbj2.purchase_datetime) GROUP BY
YEAR(vbj2.purchase_datetime), vbj2.dst_country_name)
```

Live Demo