# Final Project Guidelines

Here are some of the basic procedures that are required for the project: *NOTE: for this project assume that all flights are direct and one way only.*

**Customer Functions:**

1. Should be able to register by entering his/her name, address, city, state, phone number. Once successfully registered, the system should provide him an ID.

2. Should be able to search for a flight by giving the source, destination and date. The procedure should return a list of flights along with fares from the source to the destination on the given date.

3. Should be able to make a reservation by giving the number of passengers, and the flight number (based on the search made in 2 above)

4. Should be able to view all of his reservations by entering his ID.

**Admin Functions:**

1. Should be able to view all flight details from a location (given the source).

2. Should be able to change the fare of any flight (based on Flight ID).

3. Should be able to see all the reservations on a given flight

4. Should be able to view any customer details (given the ID of the customer)

5. Add a new flight

Your program should consist of the above procedures. You can use a text based menu for Customer and have the customer make a choice as to what he wants to do. For example the customer should see:

```
Customer Menu
1. Register
2. Search for a flight
3. Make Reservation
4. View your Reservation
5. Exit
```

Similarly the admin should see the following menu:

```
Admin Menu
1. View flight details by entering the source location
2. Change the fare of any flight
3. Add a flight
4. View all reservations for a given flight
5. View Customer details (by entering the ID of the customer)
6. Exit
```

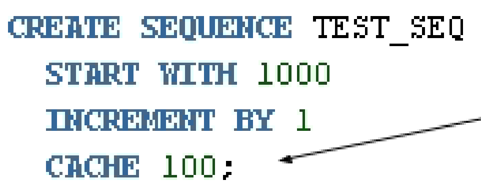
**Extra Credit**

If the flight is full, the customer should be notified and not allowed to make a reservation. After each successful reservation, the number of seats in the flight should be reduced by the number of passengers.

**Hints and Suggestions:**

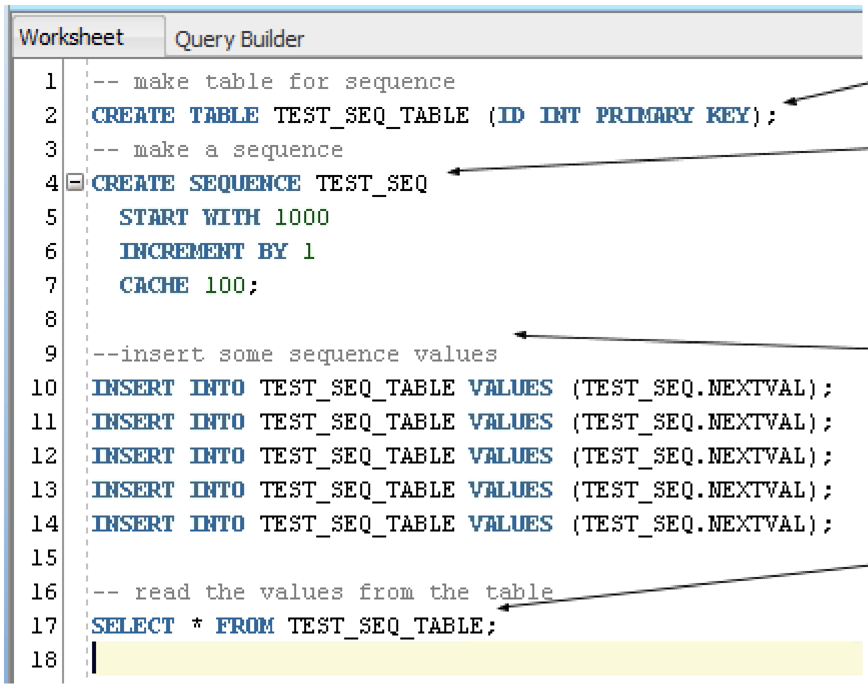
1. Use nextval for automatically generating the ID of the customer (during the registration process). That ID should be given to the customer after successful registration.

Here's an example sequence I created in SQL Developer (you'll get to try this later on)



```
CREATE SEQUENCE TEST_SEQ
  START WITH 1000
  INCREMENT BY 1
  CACHE 100;
```

Here is an example of a Sequence for creating unique numbers for an Order table's primary key. Notice that I cached 100 sequence numbers for performance reasons.



```
-- make table for sequence
CREATE TABLE TEST_SEQ_TABLE (ID INT PRIMARY KEY);
-- make a sequence
CREATE SEQUENCE TEST_SEQ
  START WITH 1000
  INCREMENT BY 1
  CACHE 100;

--insert some sequence values
INSERT INTO TEST_SEQ_TABLE VALUES (TEST_SEQ.NEXTVAL);
INSERT INTO TEST_SEQ_TABLE VALUES (TEST_SEQ.NEXTVAL);
INSERT INTO TEST_SEQ_TABLE VALUES (TEST_SEQ.NEXTVAL);
INSERT INTO TEST_SEQ_TABLE VALUES (TEST_SEQ.NEXTVAL);
INSERT INTO TEST_SEQ_TABLE VALUES (TEST_SEQ.NEXTVAL);

-- read the values from the table
SELECT * FROM TEST_SEQ_TABLE;
```

Make a table for the sequence.

Make the sequence

Use the sequence in 5 inserts

See the sequence numbers users

2. Use the execute procedure command to test each of the above procedures.

For example:

1. **execute register_customer (‘Ashwin S’, ‘300 Jay st’, ‘Brooklyn’, ‘NY’, ‘111-222-3333’);**
   Should display the following statements:
   **The new customer has been successfully registered, and his Customer ID is: 1111;**
   We should then be able to view this customer by typing the following:
   **select * from customer;**

2. **execute find_flight (‘jfk’, ‘chicago’, ’10-MAY-2017’);**
   **should display something like below:**

| FlightID | Source | Destination | Fare | Date | Number of Seats remaining |
|---|---|---|---|---|---|
| 1111 | Jfk | Chicago | $300 | 10-MAY-2017 | 10 |
| 2222 | Jfk | Chicago | $400 | 10-MAY-2017 | 40 |
| 3333 | Jfk | Chicago | $200 | 10-MAY-2017 | 2 |
| 4444 | Jfk | Chicago | $100 | 10-MAY-2017 | 56 |

3. **execute make_a_reservation (1111, 2222, 2)**
   should return
   **The customer 1111 has made the reservation on flight 2222 with 2 seats confirmed.**

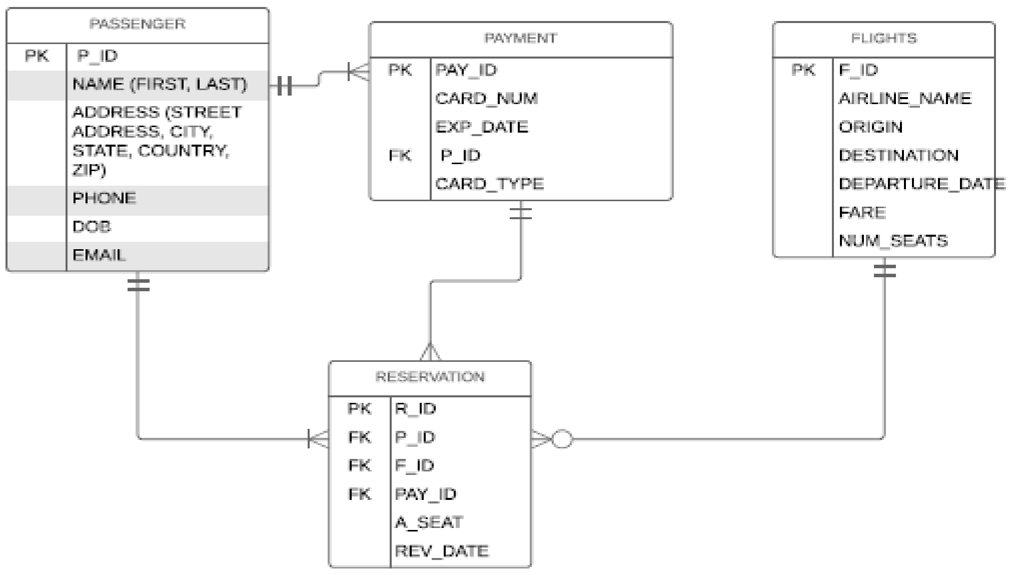
**And so on..**

**Trigger for generating the ID of the inserted movie:**

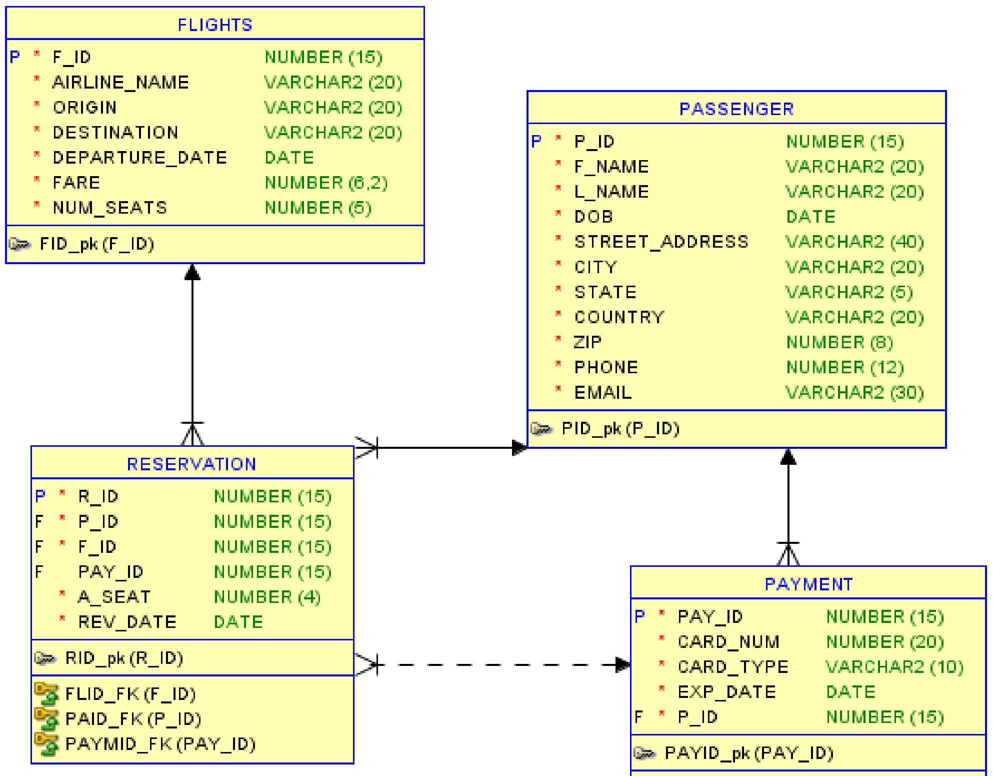
```
create or replace trigger generateID
before insert or update of id on movies
for each row
begin
  dbms_output.put_line('The id of the movie inserted is: ' || :NEW.id);
end;
```

**Airline Booking System: Airline Booking System:** Created an Entity Relationship Diagram model from a business requirement. The goal was to create an airline booking system by following the requirements. First, created a physical model and then implemented all data into Oracle 11g server. Pulled out data using SQL query. Any user can log in to the airline's booking system using their credential and book, update and cancel flights

**ERD DIAGRAM:**

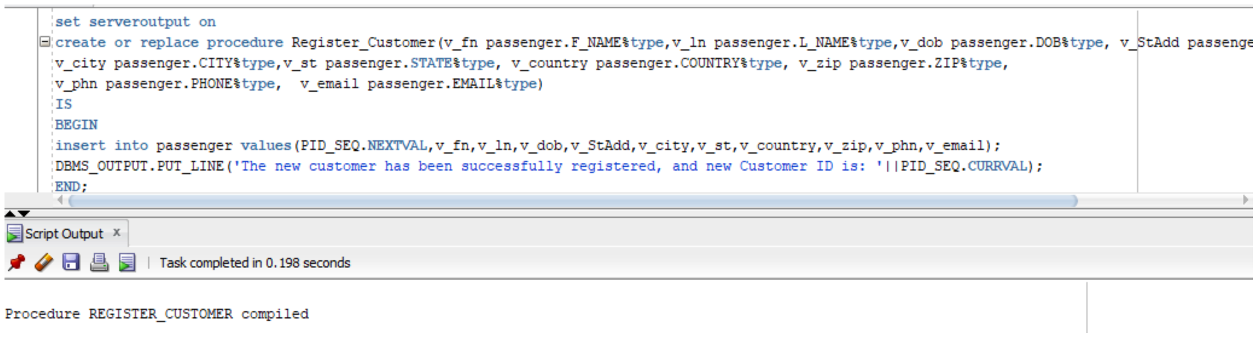


**Physical Model**



## CUSTOMER FUNCTIONS

1. Should be able to register by entering his/her name, address, city, state, phone number. Once successfully registered, the system should provide him an ID.



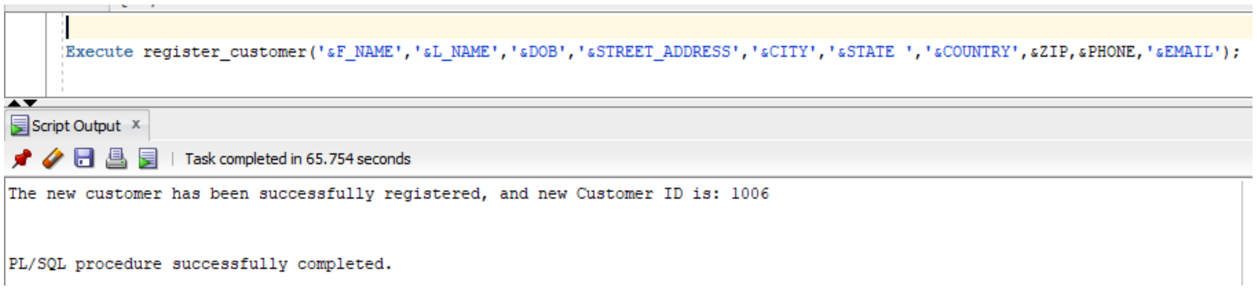
```
set serveroutput on
create or replace procedure Register_Customer(v_fn passenger.F_NAME%type,v_ln passenger.L_NAME%type,v_dob passenger.DOB%type, v_StAdd passenge
v_city passenger.CITY%type,v_st passenger.STATE%type, v_country passenger.COUNTRY%type, v_zip passenger.ZIP%type,
v_phn passenger.PHONE%type,  v_email passenger.EMAIL%type)
IS
BEGIN
insert into passenger values(PID_SEQ.NEXTVAL,v_fn,v_ln,v_dob,v_StAdd,v_city,v_st,v_country,v_zip,v_phn,v_email);
DBMS_OUTPUT.PUT_LINE('The new customer has been successfully registered, and new Customer ID is: '||PID_SEQ.CURRVAL);
END;
```

Script Output

Task completed in 0.198 seconds

```
Procedure REGISTER_CUSTOMER compiled
```

**Output:**



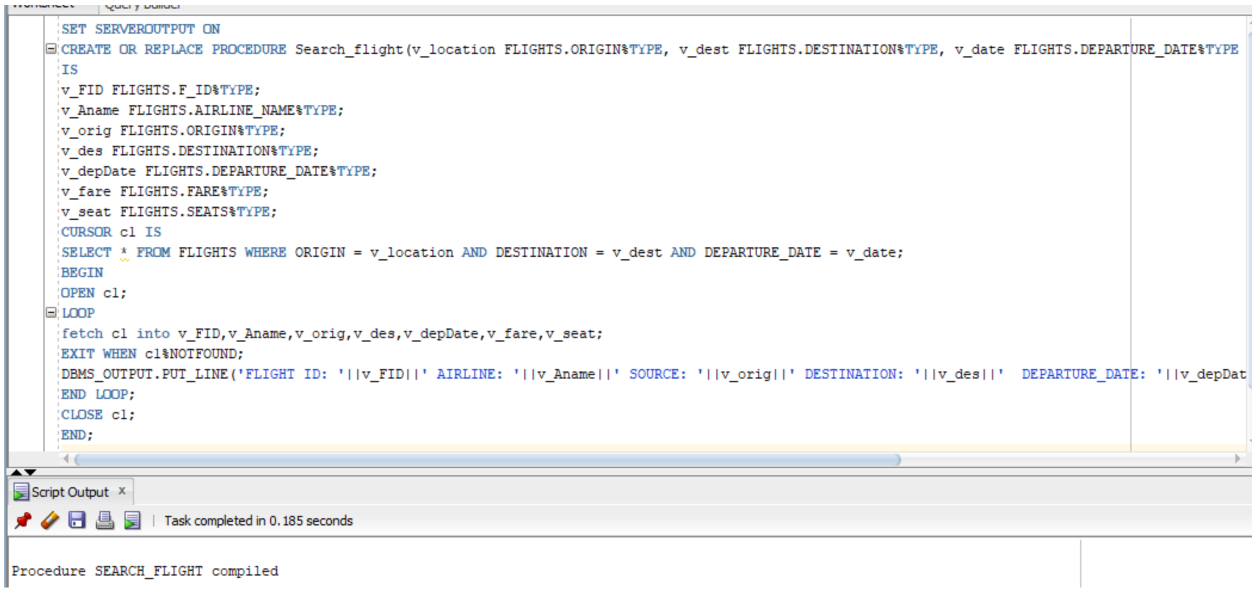
```
Execute register_customer('&F_NAME','&L_NAME','&DOB','&STREET_ADDRESS','&CITY','&STATE ','&COUNTRY',&ZIP,&PHONE,'&EMAIL');
```

Script Output

Task completed in 65.754 seconds

```
The new customer has been successfully registered, and new Customer ID is: 1006


PL/SQL procedure successfully completed.
```

2. Should be able to search for a flight by giving the source, destination and date. The procedure should return a list of flights along with fares from the source to the destination on the given date.



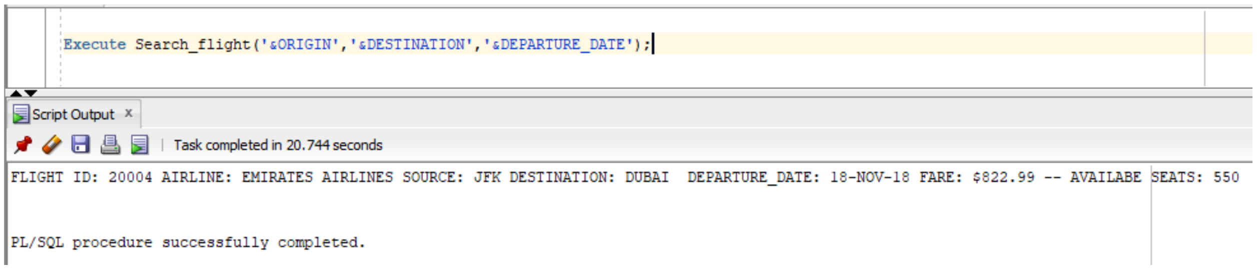
```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE Search_flight(v_location FLIGHTS.ORIGIN%TYPE, v_dest FLIGHTS.DESTINATION%TYPE, v_date FLIGHTS.DEPARTURE_DATE%TYPE
IS
v_FID FLIGHTS.F_ID%TYPE;
v_Aname FLIGHTS.AIRLINE_NAME%TYPE;
v_orig FLIGHTS.ORIGIN%TYPE;
v_des FLIGHTS.DESTINATION%TYPE;
v_depDate FLIGHTS.DEPARTURE_DATE%TYPE;
v_fare FLIGHTS.FARE%TYPE;
v_seat FLIGHTS.SEATS%TYPE;
CURSOR cl IS
SELECT * FROM FLIGHTS WHERE ORIGIN = v_location AND DESTINATION = v_dest AND DEPARTURE_DATE = v_date;
BEGIN
OPEN cl;
LOOP
fetch cl into v_FID,v_Aname,v_orig,v_des,v_depDate,v_fare,v_seat;
EXIT WHEN cl%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('FLIGHT ID: '||v_FID||' AIRLINE: '||v_Aname||' SOURCE: '||v_orig||' DESTINATION: '||v_des||'  DEPARTURE_DATE: '||v_depDat
END LOOP;
CLOSE cl;
END;
```

Script Output

Task completed in 0.185 seconds

```
Procedure SEARCH_FLIGHT compiled
```

**Output:**



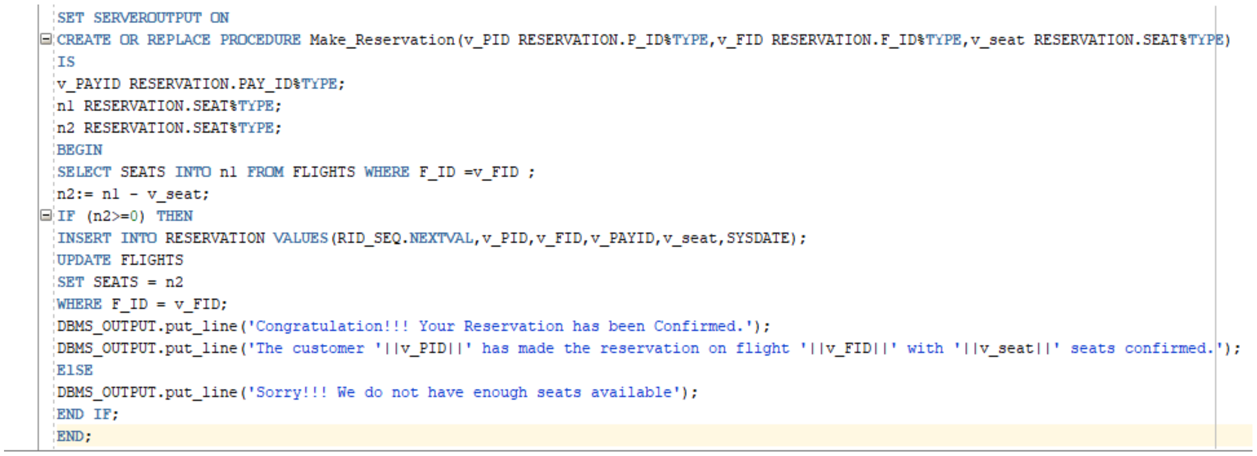
```
Execute Search_flight('&ORIGIN','&DESTINATION','&DEPARTURE_DATE');
```

Script Output ×

Task completed in 20.744 seconds

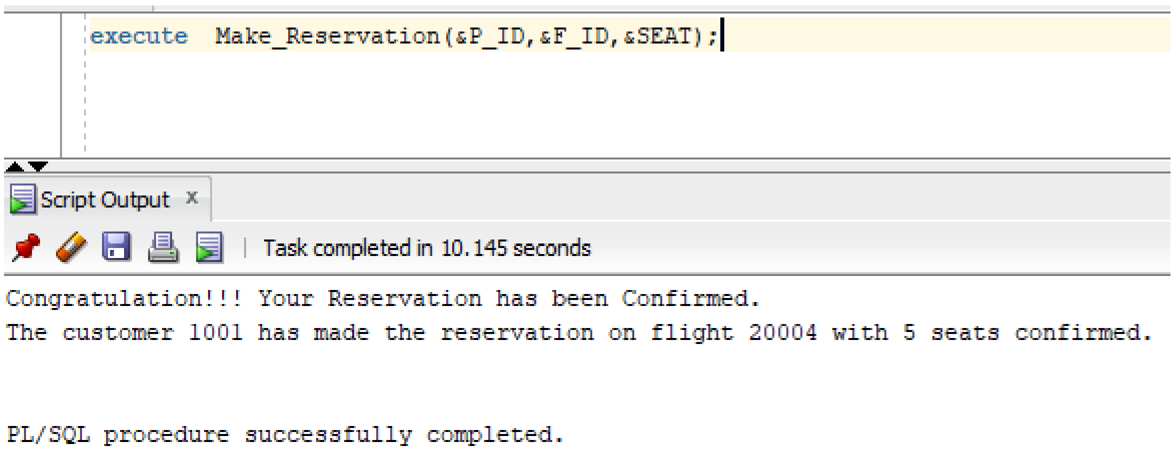
```
FLIGHT ID: 20004 AIRLINE: EMIRATES AIRLINES SOURCE: JFK DESTINATION: DUBAI  DEPARTURE_DATE: 18-NOV-18 FARE: $822.99 -- AVAILABE SEATS: 550


PL/SQL procedure successfully completed.
```

3. Should be able to make a reservation by giving the number of passengers, and the flight number (based on the search made in 2 above)



```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE Make_Reservation(v_PID RESERVATION.P_ID%TYPE,v_FID RESERVATION.F_ID%TYPE,v_seat RESERVATION.SEAT%TYPE)
IS
v_PAYID RESERVATION.PAY_ID%TYPE;
n1 RESERVATION.SEAT%TYPE;
n2 RESERVATION.SEAT%TYPE;
BEGIN
SELECT SEATS INTO n1 FROM FLIGHTS WHERE F_ID =v_FID ;
n2:= n1 - v_seat;
IF (n2>=0) THEN
INSERT INTO RESERVATION VALUES(RID_SEQ.NEXTVAL,v_PID,v_FID,v_PAYID,v_seat,SYSDATE);
UPDATE FLIGHTS
SET SEATS = n2
WHERE F_ID = v_FID;
DBMS_OUTPUT.put_line('Congratulation!!! Your Reservation has been Confirmed.');
DBMS_OUTPUT.put_line('The customer '||v_PID||' has made the reservation on flight '||v_FID||' with '||v_seat||' seats confirmed.');
ElSE
DBMS_OUTPUT.put_line('Sorry!!! We do not have enough seats available');
END IF;
END;
```

**Output:**



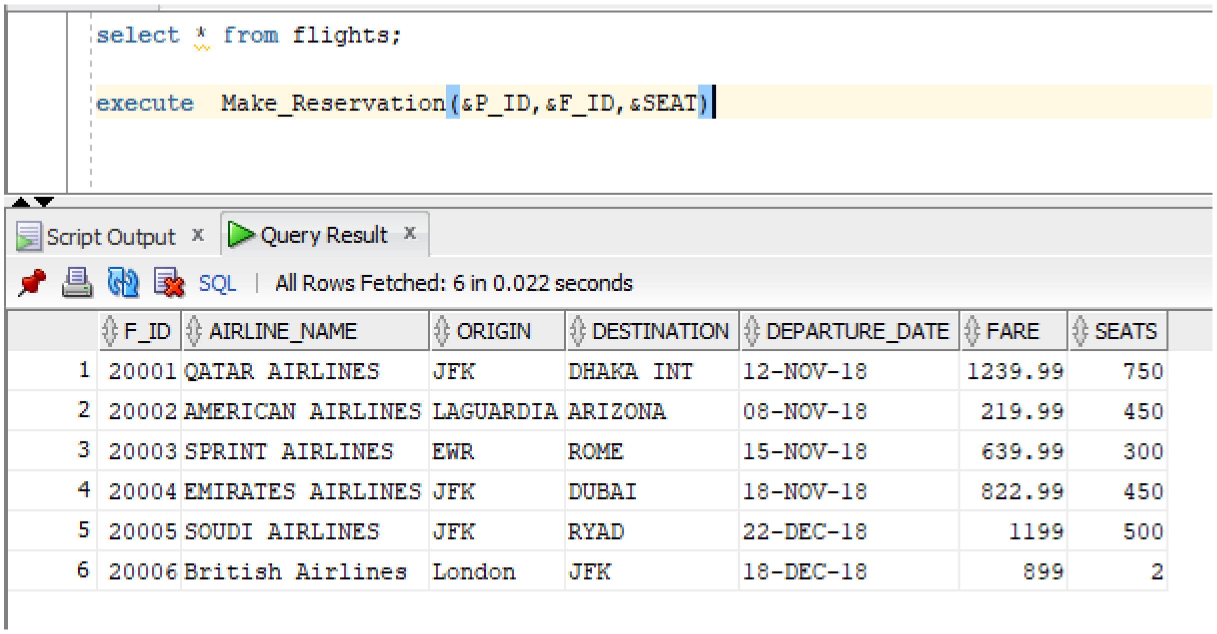
```
execute  Make_Reservation(&P_ID,&F_ID,&SEAT);
```

Script Output ×

Task completed in 10.145 seconds

```
Congratulation!!! Your Reservation has been Confirmed.
The customer 1001 has made the reservation on flight 20004 with 5 seats confirmed.


PL/SQL procedure successfully completed.
```

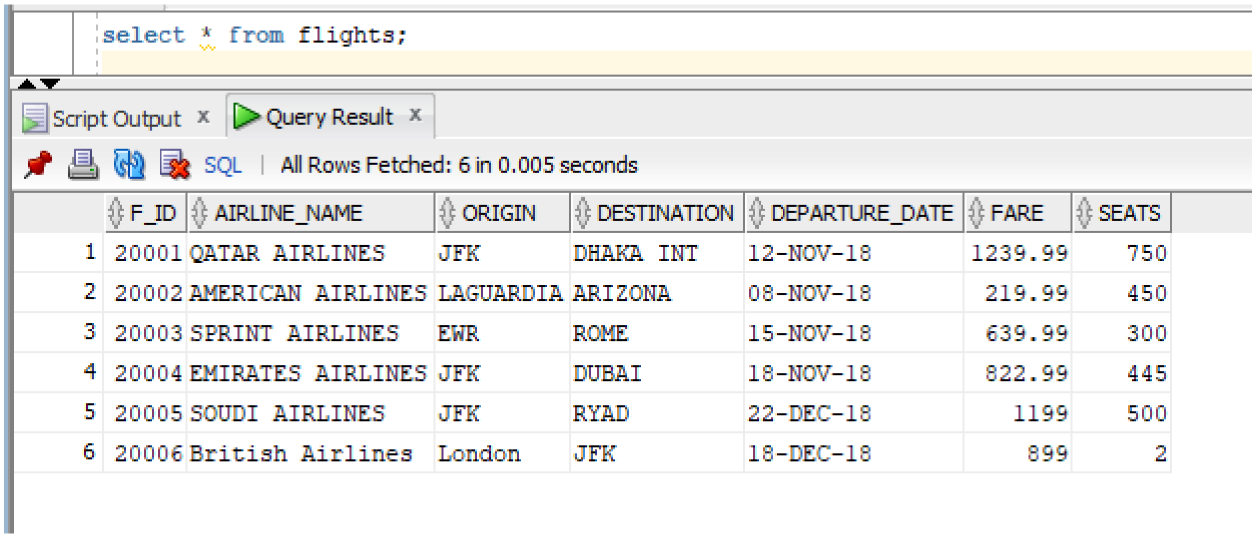
Here you can see Flight#2004 has 450 seats.



```
select * from flights;

execute  Make_Reservation(&P_ID,&F_ID,&SEAT)
```

| | F_ID | AIRLINE_NAME | ORIGIN | DESTINATION | DEPARTURE_DATE | FARE | SEATS |
|---|---|---|---|---|---|---|---|
| 1 | 20001 | QATAR AIRLINES | JFK | DHAKA INT | 12-NOV-18 | 1239.99 | 750 |
| 2 | 20002 | AMERICAN AIRLINES | LAGUARDIA | ARIZONA | 08-NOV-18 | 219.99 | 450 |
| 3 | 20003 | SPRINT AIRLINES | EWR | ROME | 15-NOV-18 | 639.99 | 300 |
| 4 | 20004 | EMIRATES AIRLINES | JFK | DUBAI | 18-NOV-18 | 822.99 | 450 |
| 5 | 20005 | SOUDI AIRLINES | JFK | RYAD | 22-DEC-18 | 1199 | 500 |
| 6 | 20006 | British Airlines | London | JFK | 18-DEC-18 | 899 | 2 |

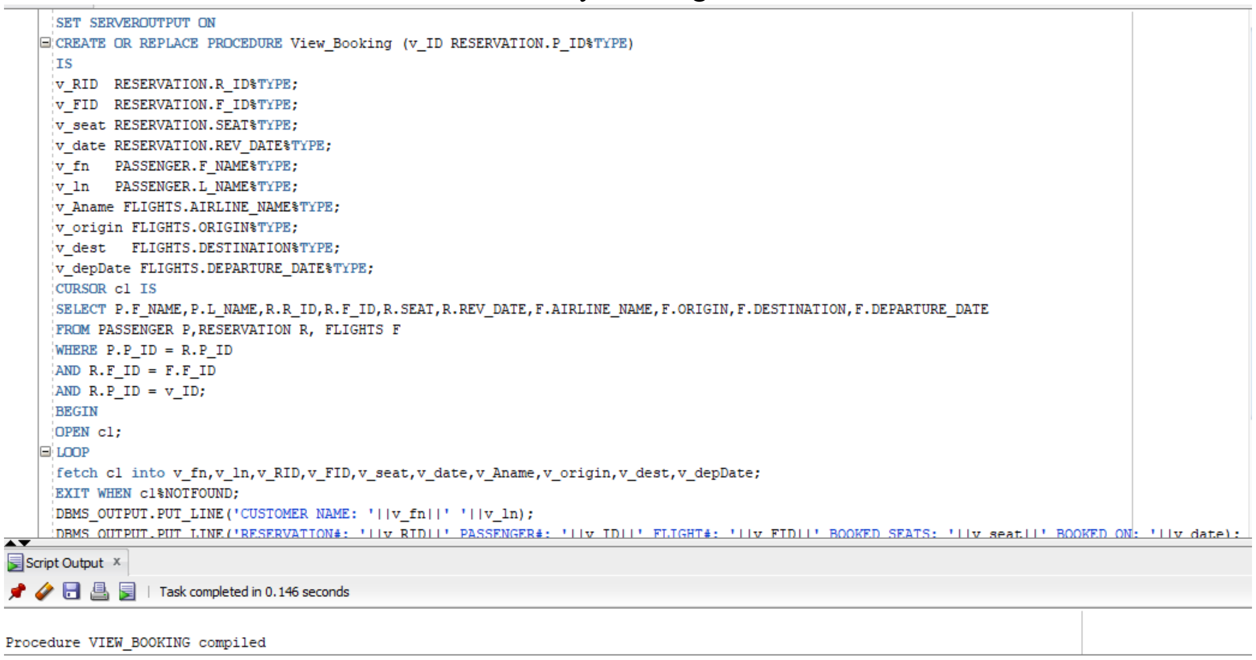
After booking 5 seats, now Flight#20004 has 445 seats. The number of booking seats has been deducted from total seats number.



```
select * from flights;
```

| | F_ID | AIRLINE_NAME | ORIGIN | DESTINATION | DEPARTURE_DATE | FARE | SEATS |
|---|---|---|---|---|---|---|---|
| 1 | 20001 | QATAR AIRLINES | JFK | DHAKA INT | 12-NOV-18 | 1239.99 | 750 |
| 2 | 20002 | AMERICAN AIRLINES | LAGUARDIA | ARIZONA | 08-NOV-18 | 219.99 | 450 |
| 3 | 20003 | SPRINT AIRLINES | EWR | ROME | 15-NOV-18 | 639.99 | 300 |
| 4 | 20004 | EMIRATES AIRLINES | JFK | DUBAI | 18-NOV-18 | 822.99 | 445 |
| 5 | 20005 | SOUDI AIRLINES | JFK | RYAD | 22-DEC-18 | 1199 | 500 |
| 6 | 20006 | British Airlines | London | JFK | 18-DEC-18 | 899 | 2 |

4. Should be able to view all of his reservations by entering his ID.



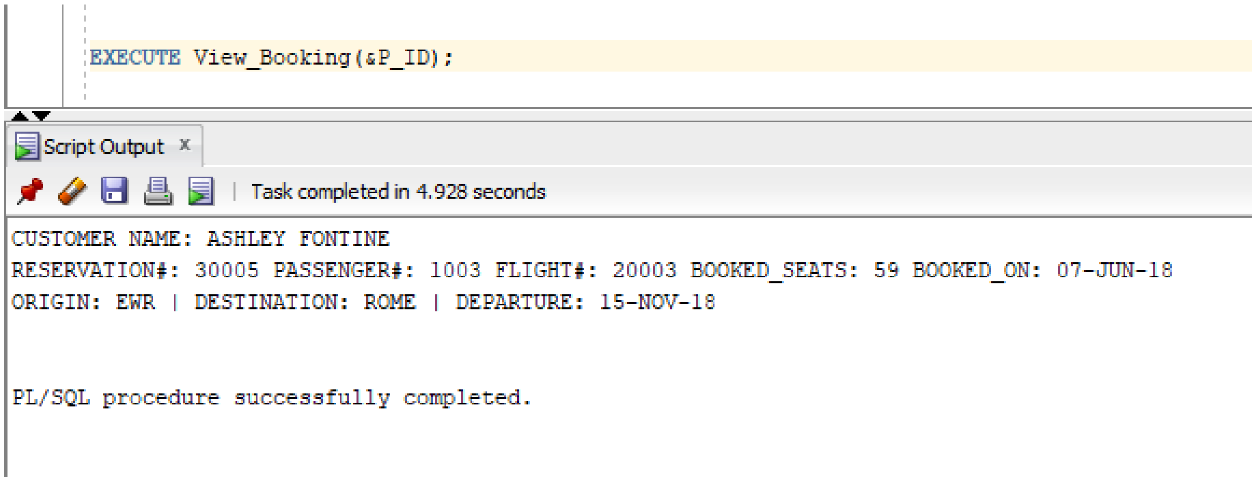
```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE View_Booking (v_ID RESERVATION.P_ID%TYPE)
IS
v_RID  RESERVATION.R_ID%TYPE;
v_FID  RESERVATION.F_ID%TYPE;
v_seat RESERVATION.SEAT%TYPE;
v_date RESERVATION.REV_DATE%TYPE;
v_fn   PASSENGER.F_NAME%TYPE;
v_ln   PASSENGER.L_NAME%TYPE;
v_Aname FLIGHTS.AIRLINE_NAME%TYPE;
v_origin FLIGHTS.ORIGIN%TYPE;
v_dest   FLIGHTS.DESTINATION%TYPE;
v_depDate FLIGHTS.DEPARTURE_DATE%TYPE;
CURSOR c1 IS
SELECT P.F_NAME,P.L_NAME,R.R_ID,R.F_ID,R.SEAT,R.REV_DATE,F.AIRLINE_NAME,F.ORIGIN,F.DESTINATION,F.DEPARTURE_DATE
FROM PASSENGER P,RESERVATION R, FLIGHTS F
WHERE P.P_ID = R.P_ID
AND R.F_ID = F.F_ID
AND R.P_ID = v_ID;
BEGIN
OPEN c1;
LOOP
fetch c1 into v_fn,v_ln,v_RID,v_FID,v_seat,v_date,v_Aname,v_origin,v_dest,v_depDate;
EXIT WHEN c1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('CUSTOMER NAME: '||v_fn||' '||v_ln);
DBMS_OUTPUT.PUT_LINE('RESERVATION#: '||v_RID||' PASSENGER#: '||v_ID||' FLIGHT#: '||v_FID||' BOOKED_SEATS: '||v_seat||' BOOKED_ON: '||v_date);
```

Script Output ×

Task completed in 0.146 seconds

```
Procedure VIEW_BOOKING compiled
```

**OUTPUT:**



```
EXECUTE View_Booking(&P_ID);
```

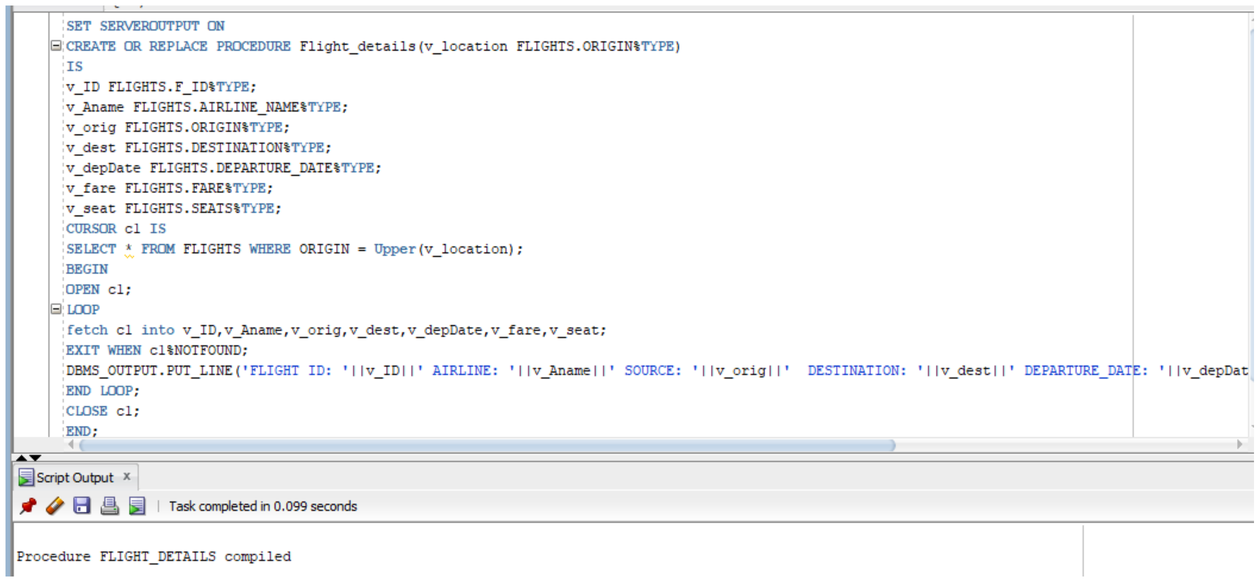
Script Output ×

Task completed in 4.928 seconds

```
CUSTOMER NAME: ASHLEY FONTINE
RESERVATION#: 30005 PASSENGER#: 1003 FLIGHT#: 20003 BOOKED_SEATS: 59 BOOKED_ON: 07-JUN-18
ORIGIN: EWR | DESTINATION: ROME | DEPARTURE: 15-NOV-18


PL/SQL procedure successfully completed.
```

**Admin Functions:**

1. Should be able to view all flight details from a location (given the source).



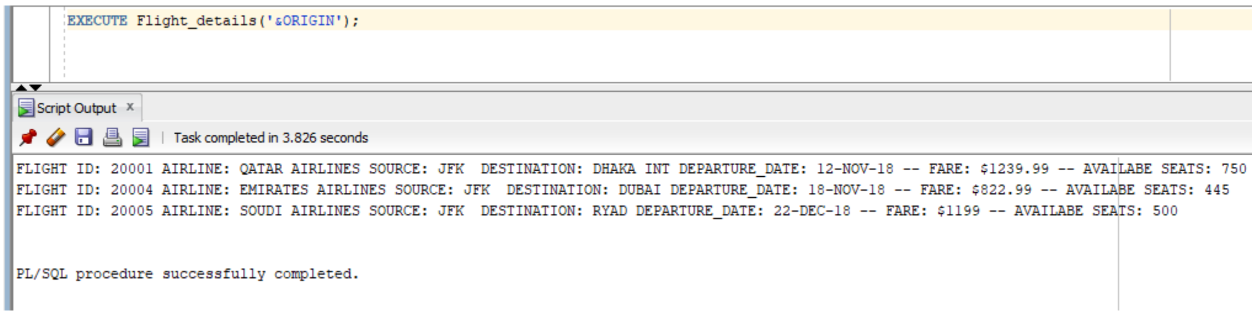
```
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE Flight_details(v_location FLIGHTS.ORIGIN%TYPE)
IS
v_ID FLIGHTS.F_ID%TYPE;
v_Aname FLIGHTS.AIRLINE_NAME%TYPE;
v_orig FLIGHTS.ORIGIN%TYPE;
v_dest FLIGHTS.DESTINATION%TYPE;
v_depDate FLIGHTS.DEPARTURE_DATE%TYPE;
v_fare FLIGHTS.FARE%TYPE;
v_seat FLIGHTS.SEATS%TYPE;
CURSOR c1 IS
SELECT * FROM FLIGHTS WHERE ORIGIN = Upper(v_location);
BEGIN
OPEN c1;
LOOP
fetch c1 into v_ID,v_Aname,v_orig,v_dest,v_depDate,v_fare,v_seat;
EXIT WHEN c1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('FLIGHT ID: '||v_ID||' AIRLINE: '||v_Aname||' SOURCE: '||v_orig||'  DESTINATION: '||v_dest||' DEPARTURE_DATE: '||v_depDat
END LOOP;
CLOSE c1;
END;
```

Script Output

Task completed in 0.099 seconds

```
Procedure FLIGHT_DETAILS compiled
```

**OUTPUT:**



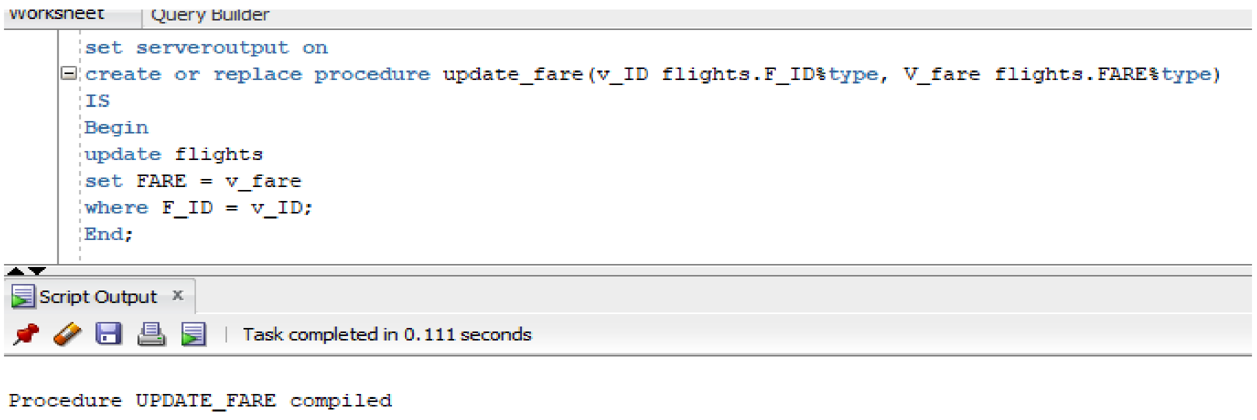
```
EXECUTE Flight_details('&ORIGIN');
```

Script Output

Task completed in 3.826 seconds

```
FLIGHT ID: 20001 AIRLINE: QATAR AIRLINES SOURCE: JFK  DESTINATION: DHAKA INT DEPARTURE_DATE: 12-NOV-18 -- FARE: $1239.99 -- AVAILABE SEATS: 750
FLIGHT ID: 20004 AIRLINE: EMIRATES AIRLINES SOURCE: JFK  DESTINATION: DUBAI DEPARTURE_DATE: 18-NOV-18 -- FARE: $822.99 -- AVAILABE SEATS: 445
FLIGHT ID: 20005 AIRLINE: SOUDI AIRLINES SOURCE: JFK  DESTINATION: RYAD DEPARTURE_DATE: 22-DEC-18 -- FARE: $1199 -- AVAILABE SEATS: 500


PL/SQL procedure successfully completed.
```

2. Should be able to change the fare of any flight (based on Flight ID).



Worksheet | Query Builder

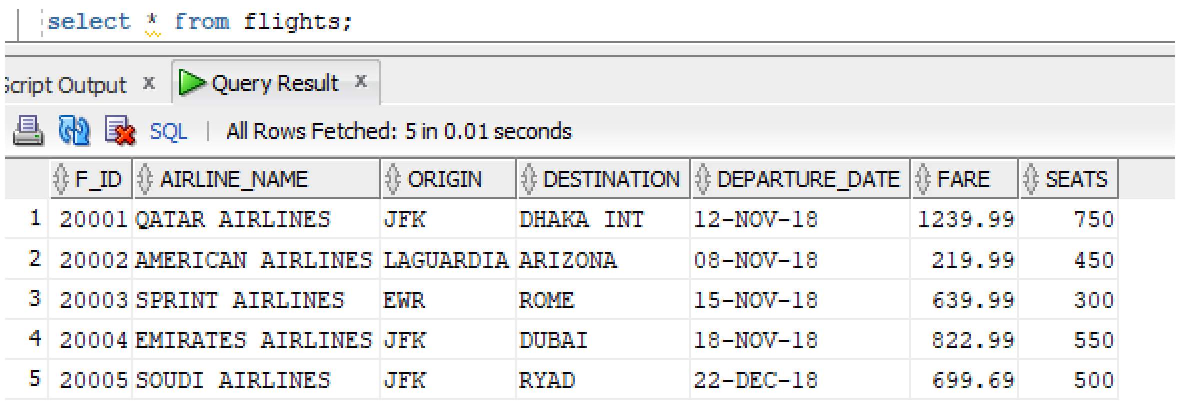
```
set serveroutput on
create or replace procedure update_fare(v_ID flights.F_ID%type, V_fare flights.FARE%type)
IS
Begin
update flights
set FARE = v_fare
where F_ID = v_ID;
End;
```

Script Output ×

Task completed in 0.111 seconds

```
Procedure UPDATE_FARE compiled
```

**OUTPUT:** This screenshot displays the fare of flights.



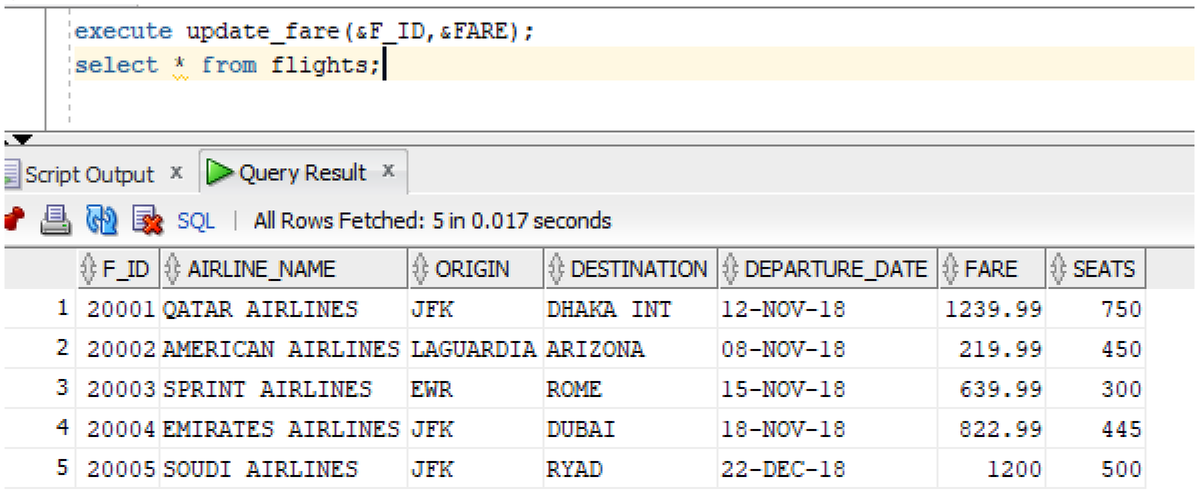
```
select * from flights;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.01 seconds

| | F_ID | AIRLINE_NAME | ORIGIN | DESTINATION | DEPARTURE_DATE | FARE | SEATS |
|---|---|---|---|---|---|---|---|
| 1 | 20001 | QATAR AIRLINES | JFK | DHAKA INT | 12-NOV-18 | 1239.99 | 750 |
| 2 | 20002 | AMERICAN AIRLINES | LAGUARDIA | ARIZONA | 08-NOV-18 | 219.99 | 450 |
| 3 | 20003 | SPRINT AIRLINES | EWR | ROME | 15-NOV-18 | 639.99 | 300 |
| 4 | 20004 | EMIRATES AIRLINES | JFK | DUBAI | 18-NOV-18 | 822.99 | 550 |
| 5 | 20005 | SOUDI AIRLINES | JFK | RYAD | 22-DEC-18 | 699.69 | 500 |

Flight#20005 Fare has been updated.



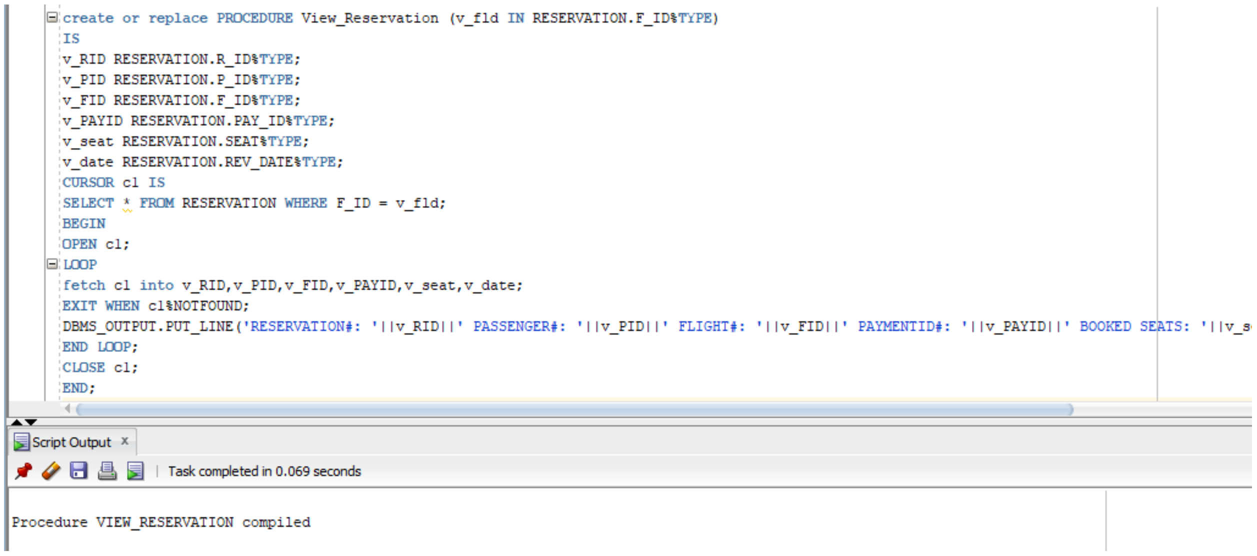
```
execute update_fare(&F_ID,&FARE);
select * from flights;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.017 seconds

| | F_ID | AIRLINE_NAME | ORIGIN | DESTINATION | DEPARTURE_DATE | FARE | SEATS |
|---|---|---|---|---|---|---|---|
| 1 | 20001 | QATAR AIRLINES | JFK | DHAKA INT | 12-NOV-18 | 1239.99 | 750 |
| 2 | 20002 | AMERICAN AIRLINES | LAGUARDIA | ARIZONA | 08-NOV-18 | 219.99 | 450 |
| 3 | 20003 | SPRINT AIRLINES | EWR | ROME | 15-NOV-18 | 639.99 | 300 |
| 4 | 20004 | EMIRATES AIRLINES | JFK | DUBAI | 18-NOV-18 | 822.99 | 445 |
| 5 | 20005 | SOUDI AIRLINES | JFK | RYAD | 22-DEC-18 | 1200 | 500 |

3. Should be able to see all the reservations on a given flight.



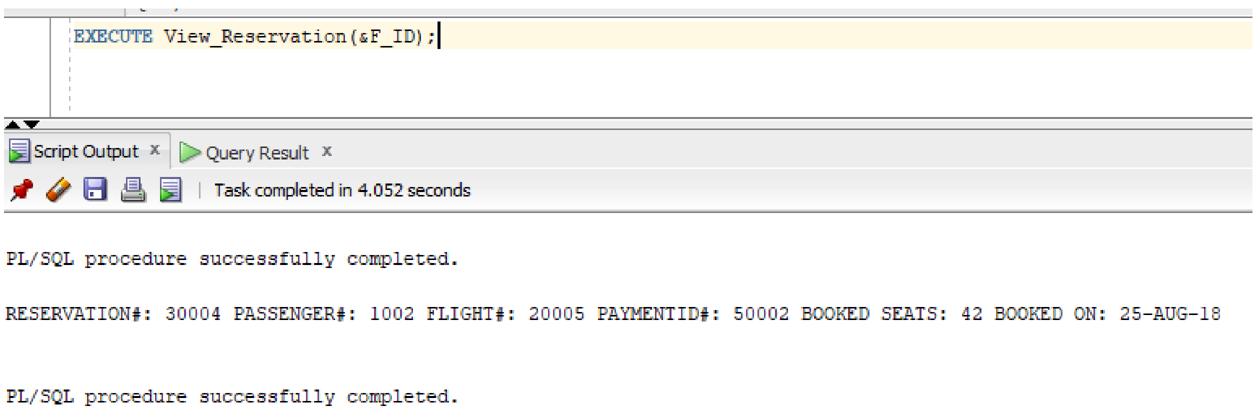
```
create or replace PROCEDURE View_Reservation (v_fId IN RESERVATION.F_ID%TYPE)
IS
v_RID RESERVATION.R_ID%TYPE;
v_PID RESERVATION.P_ID%TYPE;
v_FID RESERVATION.F_ID%TYPE;
v_PAYID RESERVATION.PAY_ID%TYPE;
v_seat RESERVATION.SEAT%TYPE;
v_date RESERVATION.REV_DATE%TYPE;
CURSOR c1 IS
SELECT * FROM RESERVATION WHERE F_ID = v_fId;
BEGIN
OPEN c1;
LOOP
fetch c1 into v_RID,v_PID,v_FID,v_PAYID,v_seat,v_date;
EXIT WHEN c1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('RESERVATION#: '||v_RID||' PASSENGER#: '||v_PID||' FLIGHT#: '||v_FID||' PAYMENTID#: '||v_PAYID||' BOOKED SEATS: '||v_s
END LOOP;
CLOSE c1;
END;
```

Script Output ×

Task completed in 0.069 seconds

```
Procedure VIEW_RESERVATION compiled
```

**OUTPUT:**



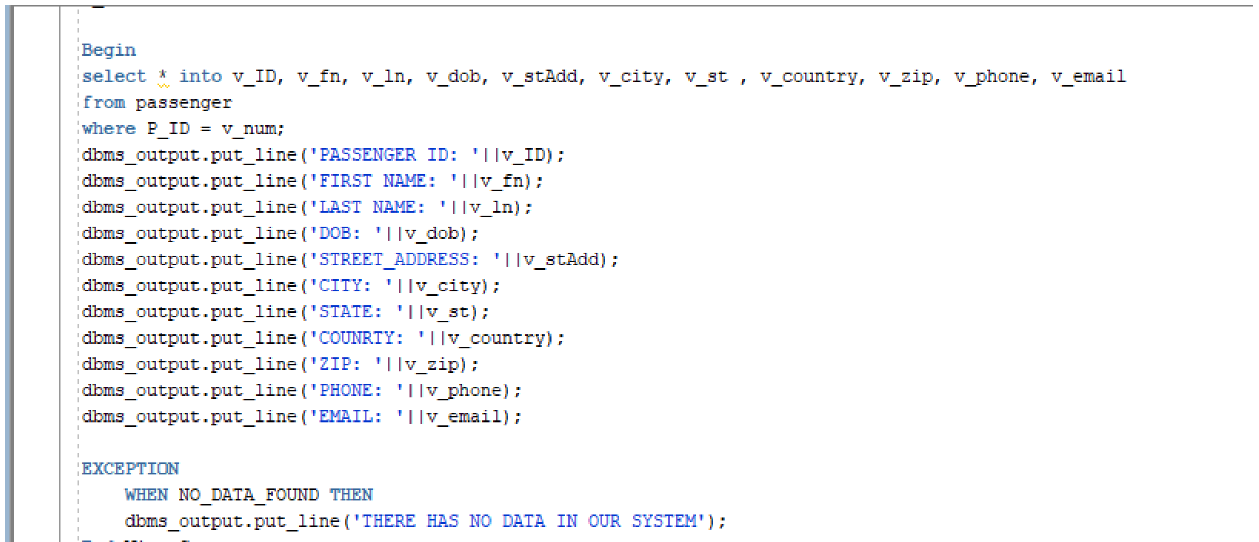
```
EXECUTE View_Reservation(&F_ID);
```

Script Output × Query Result ×

Task completed in 4.052 seconds

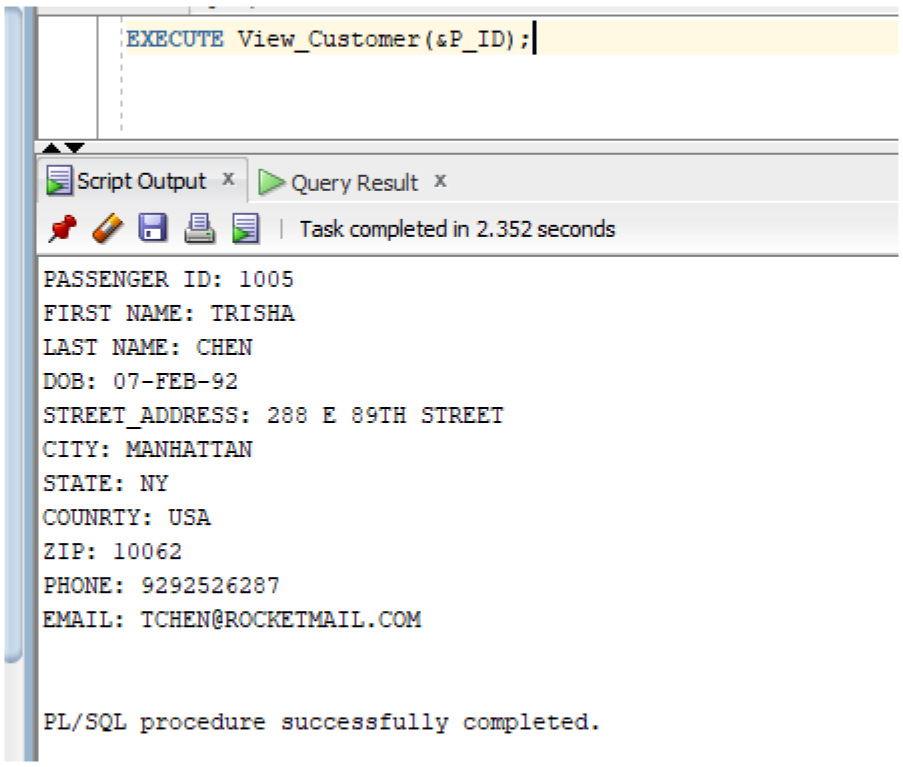
```
PL/SQL procedure successfully completed.

RESERVATION#: 30004 PASSENGER#: 1002 FLIGHT#: 20005 PAYMENTID#: 50002 BOOKED SEATS: 42 BOOKED ON: 25-AUG-18


PL/SQL procedure successfully completed.
```

4. Should be able to view any customer details (given the ID of the customer).



```
Begin
select * into v_ID, v_fn, v_ln, v_dob, v_stAdd, v_city, v_st , v_country, v_zip, v_phone, v_email
from passenger
where P_ID = v_num;
dbms_output.put_line('PASSENGER ID: '||v_ID);
dbms_output.put_line('FIRST NAME: '||v_fn);
dbms_output.put_line('LAST NAME: '||v_ln);
dbms_output.put_line('DOB: '||v_dob);
dbms_output.put_line('STREET_ADDRESS: '||v_stAdd);
dbms_output.put_line('CITY: '||v_city);
dbms_output.put_line('STATE: '||v_st);
dbms_output.put_line('COUNRTY: '||v_country);
dbms_output.put_line('ZIP: '||v_zip);
dbms_output.put_line('PHONE: '||v_phone);
dbms_output.put_line('EMAIL: '||v_email);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('THERE HAS NO DATA IN OUR SYSTEM');
```

**OUTPUT:**



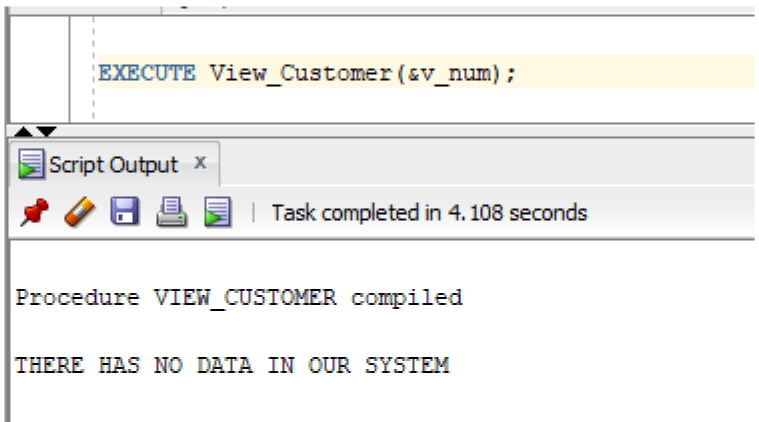
```
EXECUTE View_Customer(&P_ID);
```

Script Output × | Query Result ×

Task completed in 2.352 seconds

```
PASSENGER ID: 1005
FIRST NAME: TRISHA
LAST NAME: CHEN
DOB: 07-FEB-92
STREET_ADDRESS: 288 E 89TH STREET
CITY: MANHATTAN
STATE: NY
COUNRTY: USA
ZIP: 10062
PHONE: 9292526287
EMAIL: TCHEN@ROCKETMAIL.COM


PL/SQL procedure successfully completed.
```

I used exception in this procedure. When I enter a invalid passenger ID, then database shows there is no data in our system for this passenger.



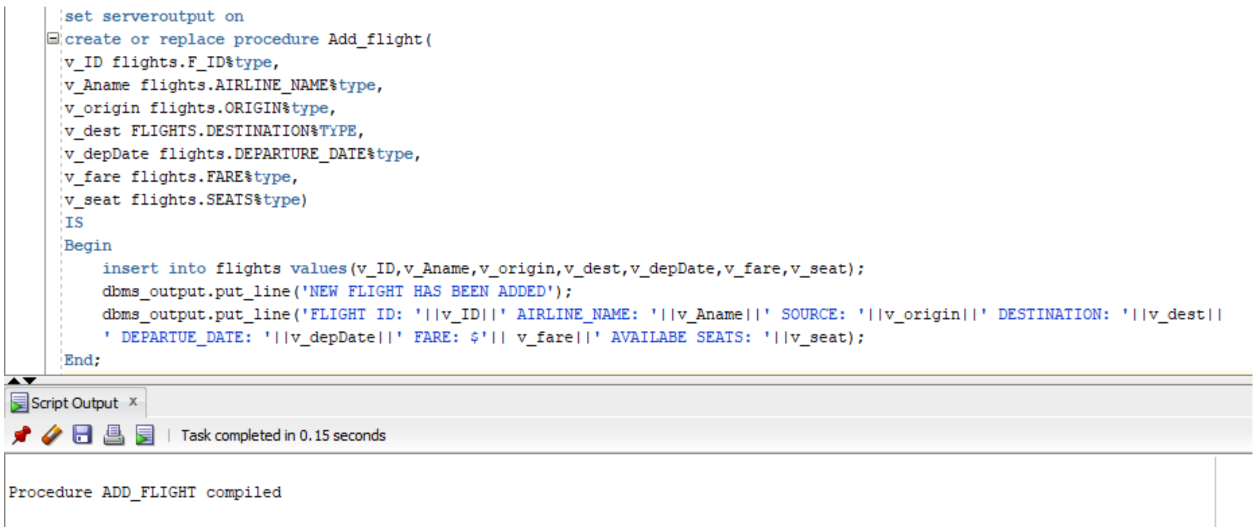
```
EXECUTE View_Customer(&v_num);
```

Script Output ×

Task completed in 4.108 seconds

```
Procedure VIEW_CUSTOMER compiled

THERE HAS NO DATA IN OUR SYSTEM
```

5. Add a new flight.



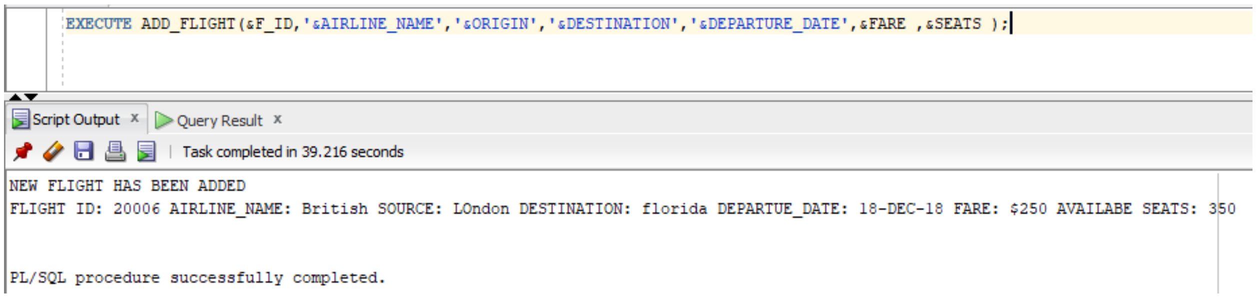
```
set serveroutput on
create or replace procedure Add_flight(
v_ID flights.F_ID%type,
v_Aname flights.AIRLINE_NAME%type,
v_origin flights.ORIGIN%type,
v_dest FLIGHTS.DESTINATION%TYPE,
v_depDate flights.DEPARTURE_DATE%type,
v_fare flights.FARE%type,
v_seat flights.SEATS%type)
IS
Begin
    insert into flights values(v_ID,v_Aname,v_origin,v_dest,v_depDate,v_fare,v_seat);
    dbms_output.put_line('NEW FLIGHT HAS BEEN ADDED');
    dbms_output.put_line('FLIGHT ID: '||v_ID||' AIRLINE_NAME: '||v_Aname||' SOURCE: '||v_origin||' DESTINATION: '||v_dest||
    ' DEPARTUE_DATE: '||v_depDate||' FARE: $'|| v_fare||' AVAILABE SEATS: '||v_seat);
End;
```

Script Output ×

Task completed in 0.15 seconds

```
Procedure ADD_FLIGHT compiled
```

**OUTPUT:**



```
EXECUTE ADD_FLIGHT(&F_ID,'&AIRLINE_NAME','&ORIGIN','&DESTINATION','&DEPARTURE_DATE',&FARE ,&SEATS );
```

Script Output × Query Result ×

Task completed in 39.216 seconds

```
NEW FLIGHT HAS BEEN ADDED
FLIGHT ID: 20006 AIRLINE_NAME: British SOURCE: LOndon DESTINATION: florida DEPARTUE_DATE: 18-DEC-18 FARE: $250 AVAILABE SEATS: 350


PL/SQL procedure successfully completed.
```