

**DATABASE MANAGEMENT SYSTEM**

**PROJECT REPORT: FLIGHT MANAGEMENT SYSTEM**

**2CO10**

**GROUP-4**

Harshita Pandey: 102003235

Meharamt Singh: 102003241

Anannya Singh: 102003253

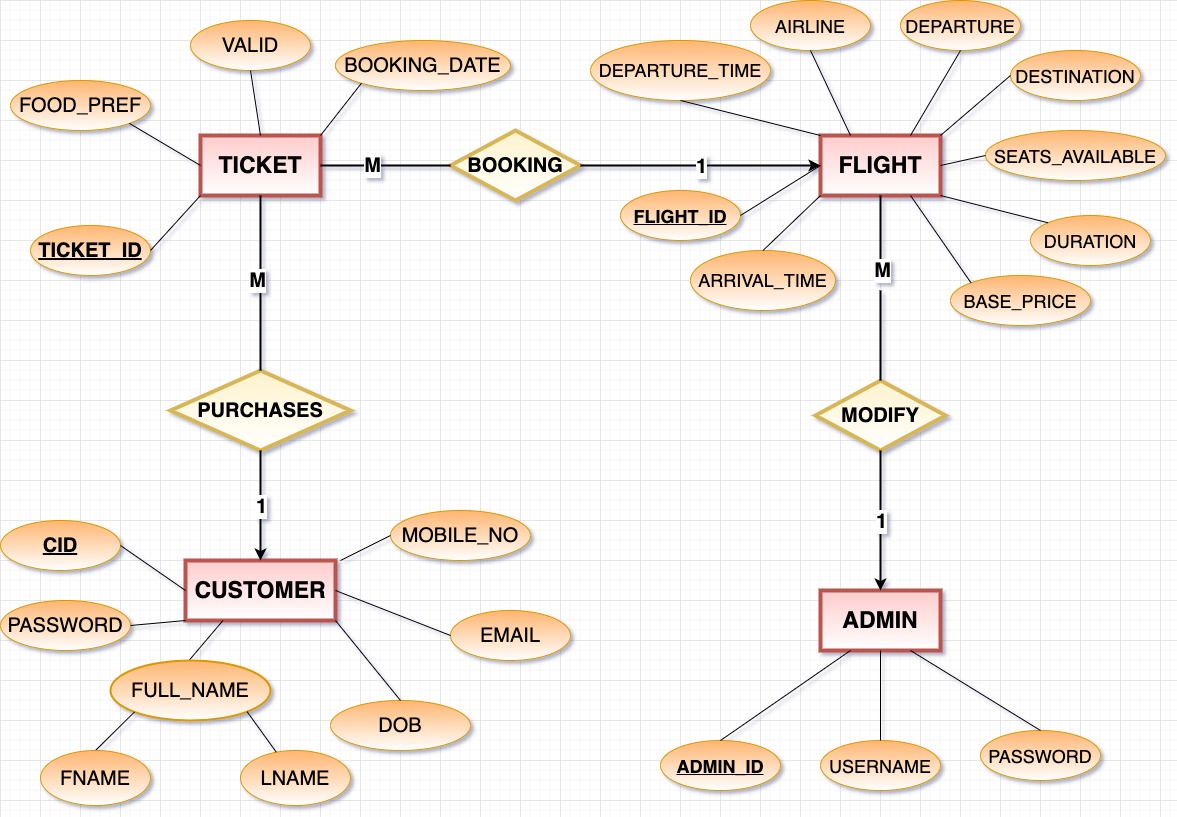
**INDEX**

|  |  |
| --- | --- |
| **Contents** | **Page No.** |
| Problem Statement | 3 |
| ER Diagram | 4 |
| ER to Table | 5 |
| Table Structures | 6 |
| Normalised Tables | 8 |
| First Normal Form | 8 |
| Second Normal Form | 9 |
| Third Normal Form | 9 |
| SQL/PLSQL Code | 12 |
| Output Screenshots | 25 |

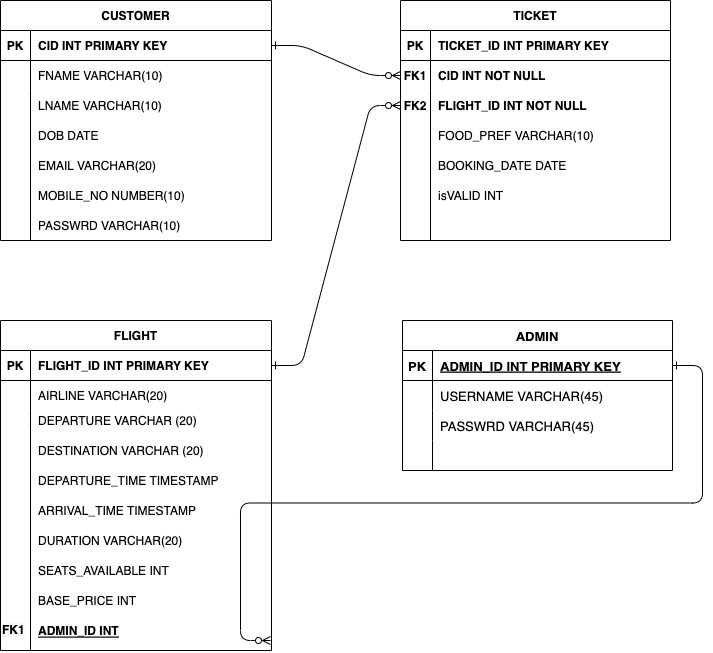
**PROBLEM STATEMENT**

Online Flight Ticket Reservation Application is to maintain customer details, ticket details and flight information. The customer details maintains customer\_id and password, which lets them access their flight tickets and make a new booking. It also maintains the personal information of the customer. The ticket details contain the ticket id, booking date and food preference of the customer. The flight details contain flight id, arrival and departure time, airline and destination and some relevant information. Then there are records of admin users maintained in the admin table, which lets the authorized users modify flight details. These entities are interlinked together for the successful working of the application.

**ER DIAGRAM**

****

**ER TO TABLE**



**TABLES**

1. **Admin**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| ADMIN\_ID | INT | PRIMARY KEY |
| USERNAME | VARCHAR(45) |  |
| PASSWRD | VARCHAR(45) |  |

1. **Flight**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| FLIGHT\_ID | INT | PRIMARY KEY |
| ADMIN\_ID | INT | CONSTRAINT FK\_ADMIN FOREIGN KEY(ADMIN\_ID) REFERENCES ADMIN(ADMIN\_ID) |
| AIRLINE | VARCHAR(20) |  |
| DEPARTURE | VARCHAR(20) |  |
| DESTINATION | VARCHAR(20) |  |
| DEPARTURE\_TIME | TIMESTAMP |  |
| ARRIVAL\_TIME | TIMESTAMP |  |
| DURATION | VARCHAR(20) |  |
| SEATS\_AVAILABLE | INT |  |
| BASE\_PRICE | INT |  |

1. **Ticket**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| TICKET\_ID | INT | PRIMARY KEY |
| CID | INT | CONTRAINT FK\_FLIGHT FOREIGN KEY(CID) REFERENCES CUSTOMER(CID) |
| FLIGHT\_ID | INT | CONSTRAINT FK\_FLIGHT FOREIGN KEY(FLIGHT\_ID) REFERENCES FLIGHT(FLIGHT\_ID) |
| FOOD\_PREF | VARCHAR(10) | CONSTRAINT CHK\_FOOD CHECK (FOOD\_PREF='VEG' OR FOOD\_PREF='NONVEG') |
| BOOKING\_DATE | TIMESTAMP | default systimestamp |
| isVALID | INT | CONSTRAINT CHK\_VALID CHECK (isVALID='0' OR isVALID='1') |

1. **Customer**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| CID | INT | PRIMARY KEY |
| FNAME | VARCHAR(10) | NOT NULL |
| LNAME | VARCHAR(10) |  |
| DOB | DATE | NOT NULL |
| EMAIL | VARCHAR(30) | UNIQUE |
| MOBILE\_NO | NUMBER(10) | CONSTRAINT CHK\_MOBILELEN CHECK (LENGTH(MOBILE)=10) |
| PASSWRD | VARCHAR(10) | NOT NULL |

**NORMALISATION**

If a table has data redundancy and is not properly normalized, then it will be difficult to handle and update the database without facing data loss. It will also eat up extra memory space, and Insertion, Update and Deletion Anomalies are very frequent if the database is not normalized.

Normalization is the process of minimizing redundancy from a relation or set of relations. Normal forms are used to eliminate or reduce redundancy in database tables.

1. **FIRST NORMAL FORM:**

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

A table is in 1 NF if:

1. There are only Single Valued Attributes.
2. Attribute Domain does not change.
3. There is a unique name for every Attribute/Column.
4. The order in which data is stored does not matter.

**The tables created in the database are already in First Normal form, as they satisfy the above criteria. They are two dimensional and flat and have only single valued attributes, with primary keys.**

1. **SECOND NORMAL FORM:**

Second Normal Form (2NF) is based on the concept of full functional dependency. Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes. A relation with a single-attribute primary key is automatically in at least 2NF. A relation that is not in 2NF may suffer from the update anomalies.

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

In other words, A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in Second Normal Form (2NF).

**Since all the tables are in first normal form (1NF) and all the non-key attributes are fully functionally dependent on the primary key, hence the tables are already in second normal (2NF) form.**

1. **THIRD NORMAL FORM:**

A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.

A relation is in 3NF if at least one of the following condition holds in every non-trivial function dependency X –> Y:

X is a super key.

Y is a prime attribute (each element of Y is part of some candidate key).

In other words, A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).

**We can see that there is a transitive dependency for Duration in table FLIGHT as:**

**Flight\_ID -> Destination,Departure**

**Flight\_ID -> Duration**

**Destination, Departure -> Duration**

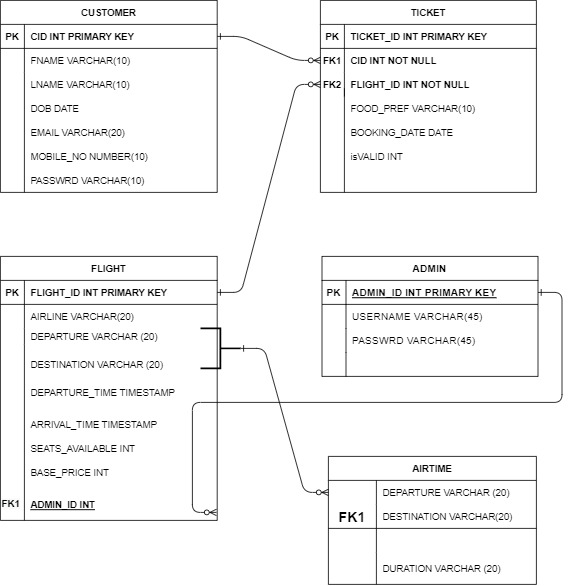
Hence **table flight** will normalise into 3NF in the following manner:

1. **FLIGHT**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| FLIGHT\_ID | INT | PRIMARY KEY |
| ADMIN\_ID | INT | CONSTRAINT FK\_ADMIN FOREIGN KEY(ADMIN\_ID) REFERENCES ADMIN(ADMIN\_ID) |
| AIRLINE | VARCHAR(20) | - |
| DEPARTURE | VARCHAR(20) | - |
| DESTINATION | VARCHAR(20) | - |
| DEPARTURE\_TIME | TIMESTAMP | - |
| ARRIVAL\_TIME | TIMESTAMP | - |
| SEATS\_AVAILABLE | INT | - |
| BASE\_PRICE | INT | - |

1. **AIRTIME**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraint** |
| DEPARTURE | VARCHAR(20) | PRIMARY KEY |
| DESTINATION | VARCHAR(20) |
| DURATION | VARCHAR(20) | - |



**PL-SQL and SQL CODES**

1. **Graphical user interface, text, application

   Description automatically generatedCREATING TABLES (after normalisation)**
2. **INSERTING DATA IN TABLES**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

1. **TRIGGERS**

**Total Triggers Made: 6**

--Auto increment trigger

-- Sequence to get the next ticket value

CREATE SEQUENCE increment\_ticket\_ID

START WITH 1

INCREMENT BY 1

CACHE 100;

--This sequence is used to increment ticket ID because auto\_increment doesn’t work.

/

--trigger associated with the sequence

--this will pick the next value from the sequence and insert it into the ticket table

CREATE OR REPLACE TRIGGER incrementTicketID

BEFORE INSERT ON TICKET

FOR EACH ROW

BEGIN

SELECT increment\_ticket\_ID.nextval

INTO :new.TICKET\_ID

FROM dual;

END;

/

--Trigger to decrement seats available in a flight after every ticket

CREATE OR REPLACE TRIGGER decrementSeats

BEFORE INSERT ON TICKET

FOR EACH ROW

BEGIN

UPDATE FLIGHT SET SEATS\_AVAILABLE=SEATS\_AVAILABLE-1 WHERE FLIGHT.FLIGHT\_ID=:new.FLIGHT\_ID;

END;

/

-- Trigger to increment seats available in a flight after every DELETED ticket

CREATE OR REPLACE TRIGGER incrementSeats

AFTER DELETE ON TICKET

FOR EACH ROW

BEGIN

UPDATE FLIGHT SET SEATS\_AVAILABLE=SEATS\_AVAILABLE+1 WHERE FLIGHT.FLIGHT\_ID=:old.FLIGHT\_ID;

END;

/

--Trigger to make all tickets invalid after departure date has passed

CREATE OR REPLACE TRIGGER checkInValid

BEFORE INSERT ON TICKET

FOR EACH ROW

BEGIN

UPDATE TICKET SET ISVALID=0 WHERE (SELECT FLIGHT.DEPARTURE\_TIME FROM FLIGHT WHERE FLIGHT.FLIGHT\_ID=TICKET.FLIGHT\_ID)<= CURRENT\_TIMESTAMP;

END;

/

--Trigger if a new ticket has been booked, it will show "thankyou for booking"

Create or Replace trigger new\_booking

after insert on TICKET

Begin

dbms\_output.put\_line('Thank you for booking with us');

End;

/

--Trigger if a new flight has been added, it will show "new flight has been added"

Create or Replace trigger new\_flight

after insert on FLIGHT

Begin

dbms\_output.put\_line('flight has been added');

End;

/

Graphical user interface, text, application, email

Description automatically generated

1. **PROCEDURES:**

**Total Procedures Made: 8**

**Graphical user interface, text, application, email

Description automatically generated**

1. **Graphical user interface, text, application, Word

   Description automatically generatedInsert Flight**
2. **Graphical user interface, application

   Description automatically generatedDelete Flight**
3. **Graphical user interface, text, application

   Description automatically generatedInsert Airtime**
4. **Graphical user interface, text, application

   Description automatically generatedDelete Airtime**
5. **Insert Customer**

**Graphical user interface, application, Word

Description automatically generated**

1. **Graphical user interface, application

   Description automatically generatedDelete Customer**
2. **Insert Ticket Details**

**Graphical user interface, text, application

Description automatically generated**

1. **Delete Ticket Details**

**Graphical user interface, application, Word

Description automatically generated**

1. **SEQUENCES:**

**Total Sequences Made: 1**

**Graphical user interface, application

Description automatically generated**

1. **EXCEPTIONS:**

**Total Exceptions Made: 2**

1. **Trigger that raises an exception when the arrival time is before departure time which is not possible.**

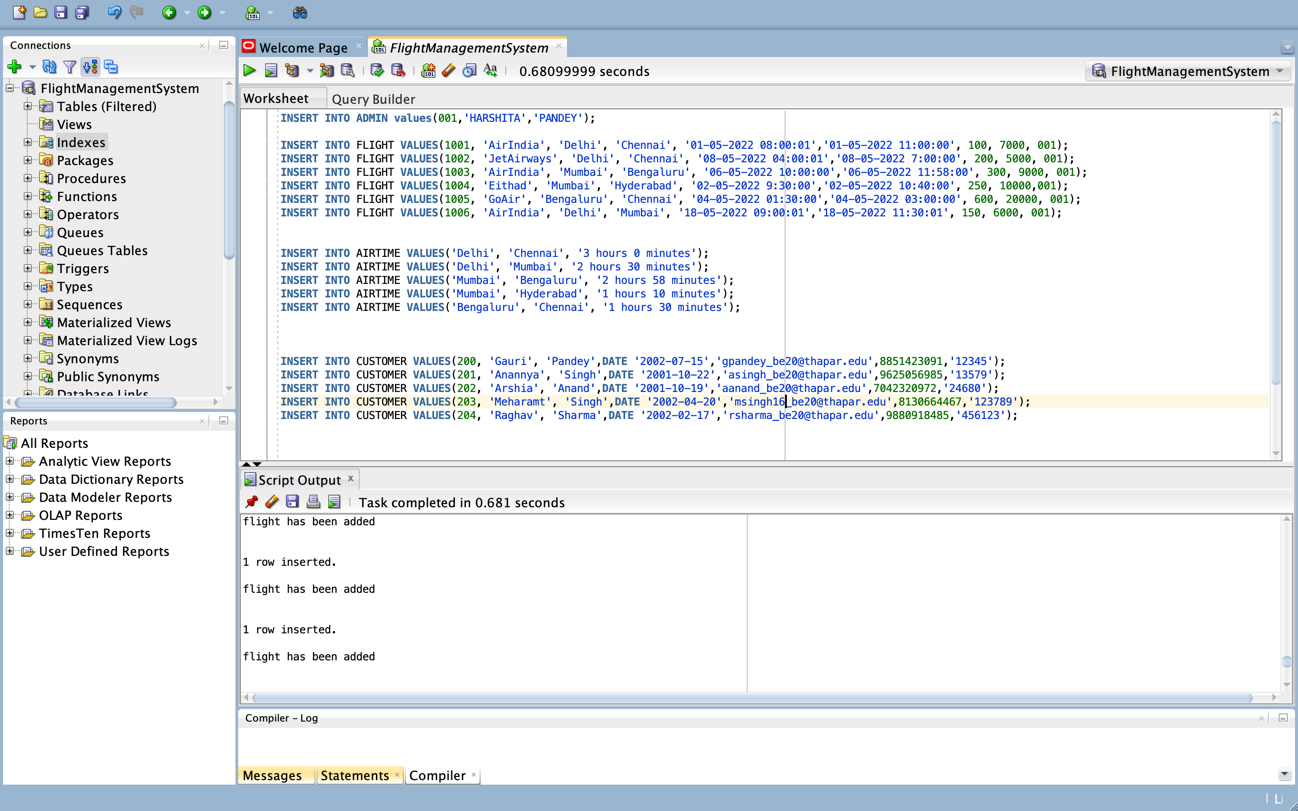
**Graphical user interface, text, application, email

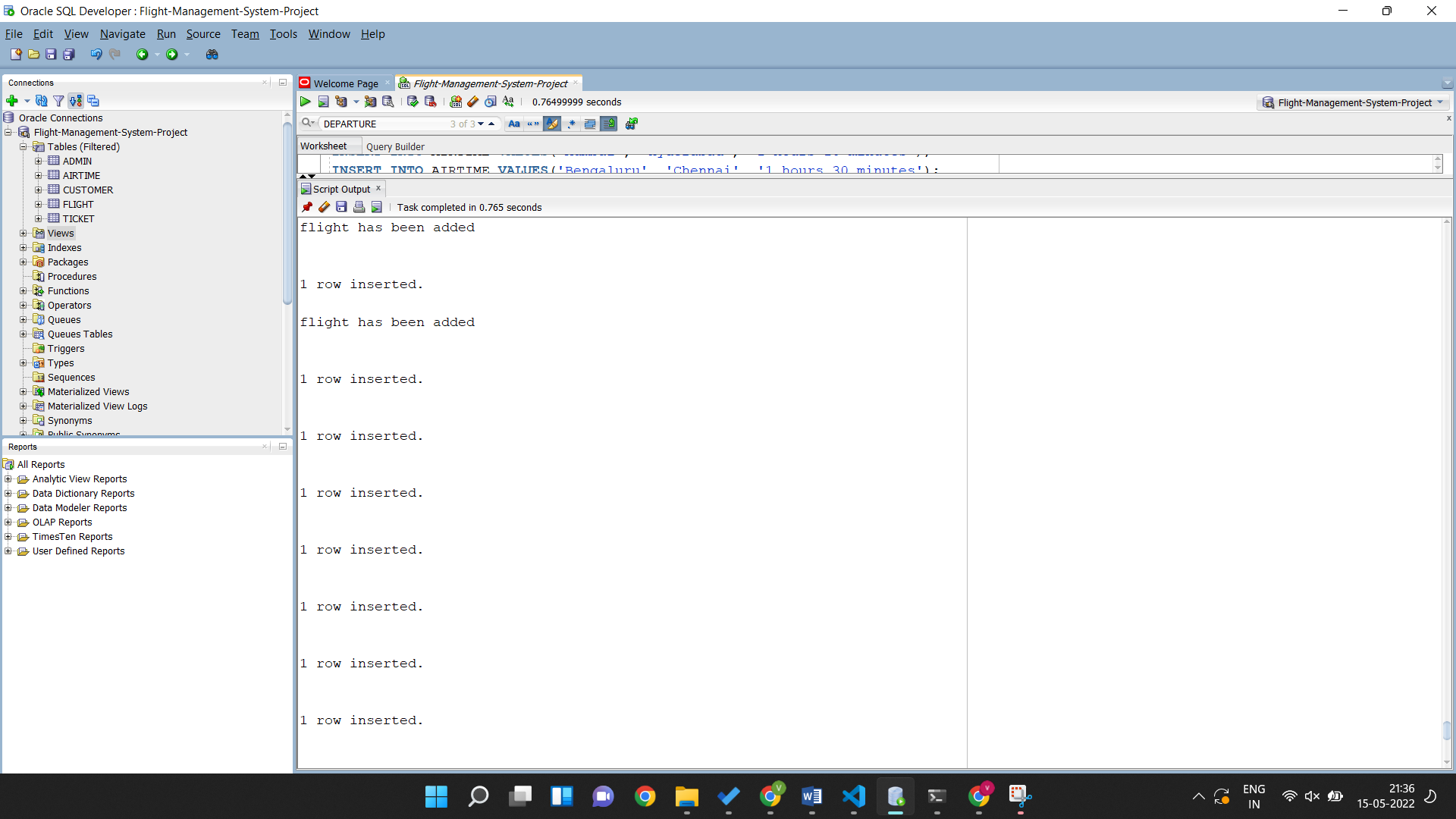
Description automatically generated**

1. **An exception that occurs when length of mobile number entered is not 10**Graphical user interface, application

   Description automatically generated

**OUTPUT-SCREENSHOTS**

****

****

**Graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated**

**Graphical user interface

Description automatically generatedTable

Description automatically generated**

**Raise Exception:**

**Exception since departure time was after arrival time**

**Graphical user interface, text, application

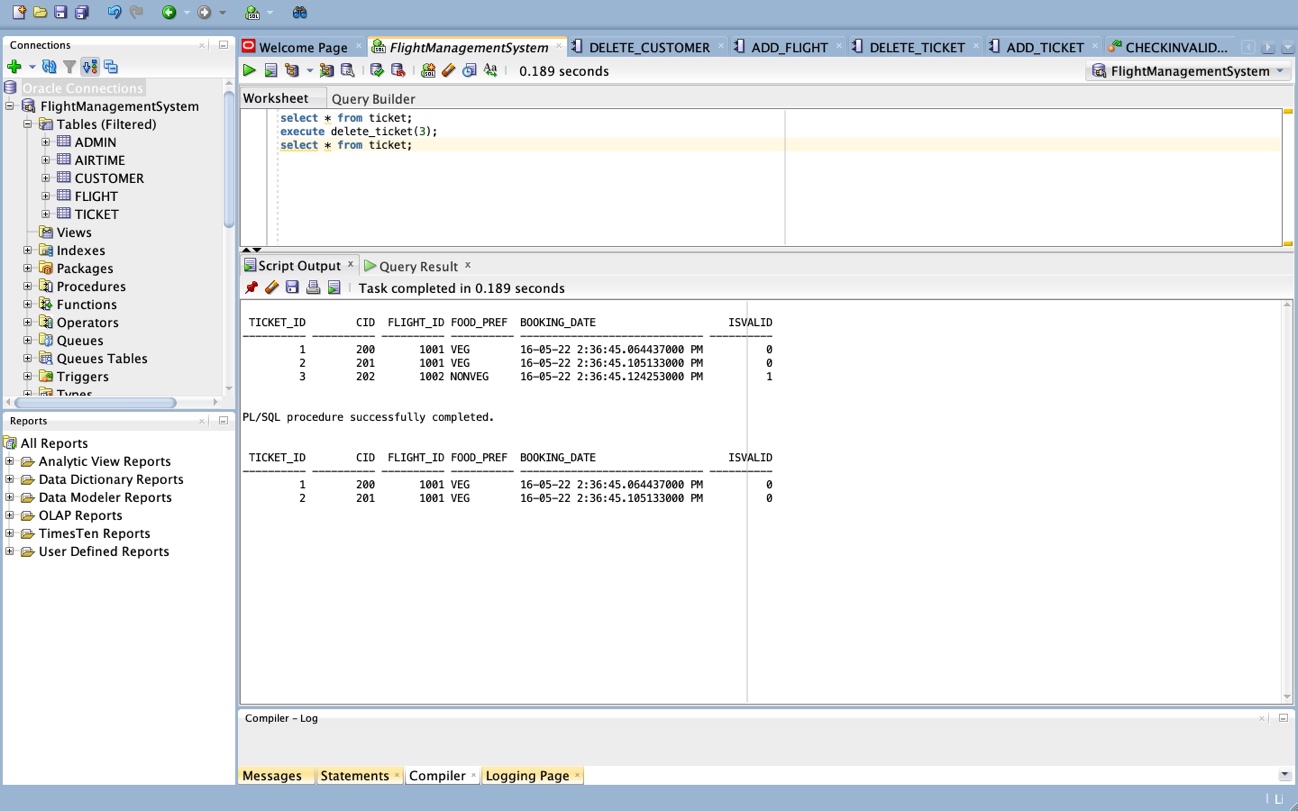
Description automatically generated**

**Invalid mobile number length**

**Graphical user interface, application

Description automatically generated**

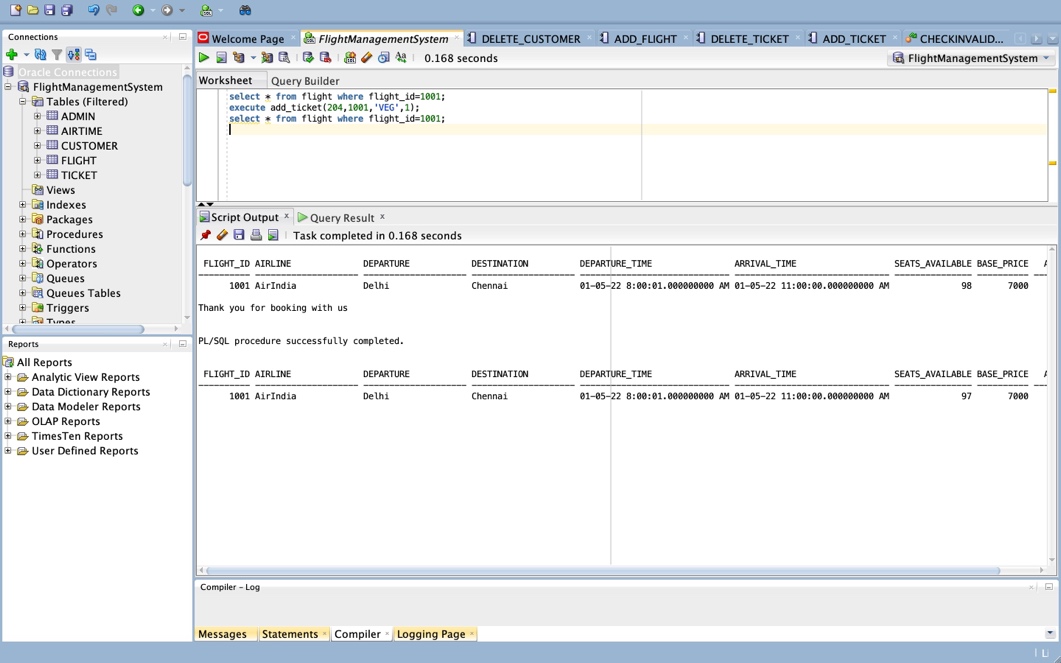
**PROCEDURES:**

**Delete ticket procedure:**

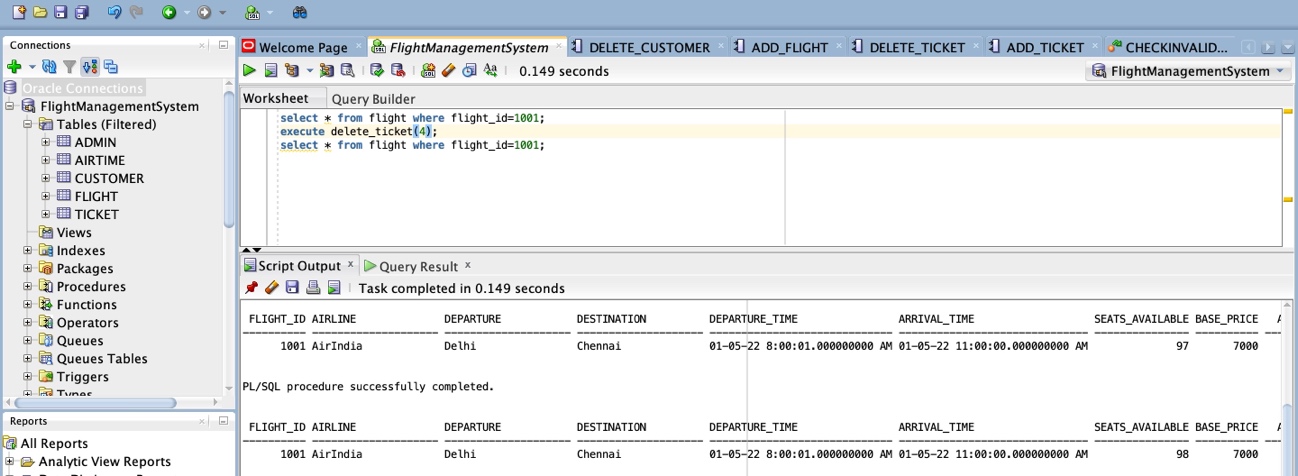
**Working of triggers:**

**Trigger 1: When you buy (add) a ticket, no of seats in a flight decrease.**

**Trigger 2: When you book a ticket, you get a message “Thankyou for booking with us”**

****

**Trigger 3: Seat number has been increased**

****