LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC6001NA Advanced Database Systems Development

Assessment Weightage & Type

40% Individual coursework

Year and Semester

2018-19 Autumn

Student Name: Rajat Shrestha

London Met ID: 17030954

College ID: np01cp4a170021

Assignment Due Date: 30th January 2020

Assignment Submission Date: 30th January 2020

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures:

Table of Tables:

1. Introduction

This is the first coursework of advanced database systems development year-long module and details in the development of a database system to manage daily operations in a travel company. This project includes the development of a Dashboard to view the overall picture of the data entered in the database, view data in the database, add data in the database and edit the data in the database. At first, this report discusses the overall process of the development of how the entities were managed and normalized into various tables to make the database more effective by reducing data redundancy. The report for database design includes the steps for Normalization, Final Third Normal Form of the database, Integrating the Normalization of two different viewpoints of daily transaction, the addition of New entities, assumptions, and finally, the combined ER-Diagram created according to the provided instruction. Then the tables are created, populated with data and then thoroughly checked. After the insertion of data simple forms are created to manipulate the data, complex forms for viewing data from multiple tables. The forms are created by the help of .Net Framework of C# Programming Language by providing an interface to do various operations on the connected Oracle Database.

The Final Application created is quite useful to view insights on the data inserted to the database and provides basic CRUD functions for essential entities such as Designation, Staffs, Customers, Assigned Tour-Guide, and Packages. Adding Packages is currently only possible via manual insertion by the administrator so it is not included. On top of the basic forms, the complex forms give the option to view various option such as Packages booked by the customer, Staff Count and Details, and Package-Activity details which provide essential operations to the operator. The dashboard provided also includes various tables and charts to give insight on the data stored. The instructions to use all of these forms in the application are described in the user manual and are extensively tested. All of the essential screenshots of the development phase is provided in chronological order.

2. Normalization

It is the process of correcting table structure to reduce redundancy and data anomalies, which minimizes storage space. It applies a series of rules called normal forms (Coronel & Morris, 2018). The database in the coursework is required to be normalized till 3NF.

According to the scenario the tour company have the following relation between the entities:

- Staffs have different roles assigned one for each
- Multiple Staffs (specifically tour guides) could be assigned for a tour
- Customer can only take one package from multiple available packages.
- The tour and package are taken as the same entity
- A registered customer might or might not have a tour assigned
- The role is taken as an attribute of staff

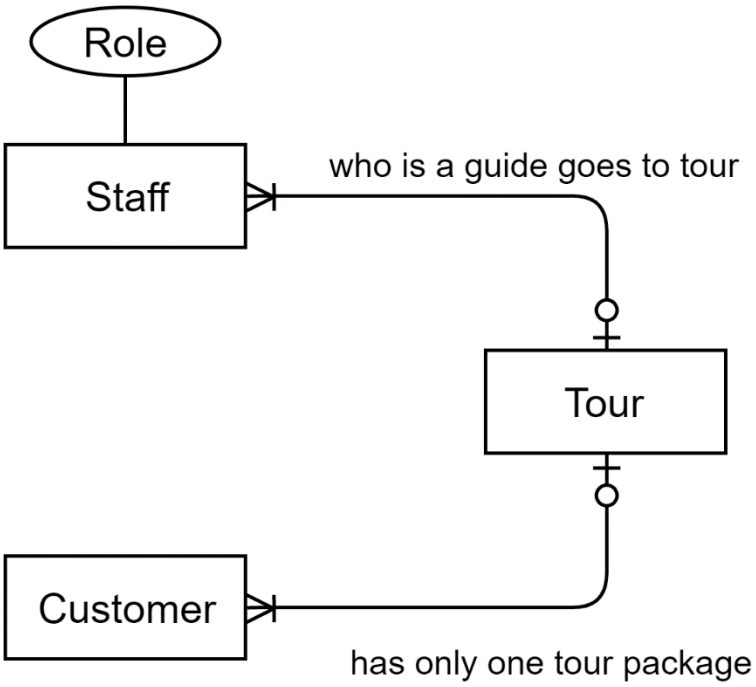

Figure 1: Basic Entities observed via the initial observation

2.1. Figure 1

The given figure gives us detail about the package record.

Package ID	Package Name	Destination	Total No of Days	Difficulty
LK25A	ABC	Annapurna Base Camp	7	Moderate
UI32A	Ghandruk	Ghandruk, Pokhara	4	Moderate
NB34G	Everest Short Trek	Lukla, Khumjung	4	Hard

Figure 2: Figure 1 as provided by the question

As all the information is enough for a package record, storing the given records as atomic value(i.e. storing each value in unique cells) we get:

Table 1: Storing Value from the given figure in Atomic cells

Package ID	Package Name	Destination	Days	Difficulty
LK25A	ABC	Annapurna Base Camp	7	Moderate
UI32A	Ghandruk	Ghandruk	4	Moderate
UI32A	Ghandruk	Pokhara	4	Moderate
NB34G	Everest Short Trek	Lukla	4	Hard
NB34G	Everest Short Trek	Khumjung	4	Hard

2.1.1. Un-normalized Form (UNF):

A database is said to be in UNF when it has not been normalized at all. The rules for creating a un-normalized form are:

- Entity and its attributes should be identified
- A Primary key need to be stated
- The repeating group should be acknowledged.

In relation, a distinctive describing a group of multiple entries for a single key attribute occurrence can be known as a repeating group. Example: Multiple items purchased by a customer in a bill (Coronel & Morris, 2018). From the above table we can observe that the destination column has repeating groups in it, representing the given data in UNF:

UNF: (**Package_ID**, Pakage_Name, {Destination}, Total_Days, Difficulty)

The destination is the repeating group because a package might have multiple destinations, for example, Pokhara and Ghandruk. Package_ID is taken as the primary key as it can uniquely identify all columns.

2.1.2. First Normal Form (1NF):

In First Normal Form, only atomic values are allowed at each cell and discourage repeating groups. For the database to be in 1NF it must be in UNF. The other rules for 1NF are:

- Primary Keys should be identified.
- Repeating groups from UNF must be separated.
- New table should have Composite Primary key including the Primary key of the original table.

As Package_Name, Total_Days, and Difficulty depend upon the Package_ID, it is assigned as a primary key. Package_Name also gives Destination

Package_ID ➔ Pakage_Name, Total_Days, Difficulty

Package_ID ➔ Destination

Removing the repeating group and creating the Destination Package and assigning Package_ID and Destination_ID as composite key and creating the first normal form.

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)

Destination_Package (**Package_ID*, Destination**)

2.1.3. Second Normal Form (2NF):

For the relation to be in Second Normal Form it must already be in First Normal Form and should not include any partial dependencies. Partial dependency is the kind of functional dependencies in which a non-key is dependent on part of a composite key (Coronel & Morris, 2018). The rules for 2NF are:

- All the functional dependencies between no key and composite key and parts of the composite key should be shown for the entities with composite Primary Key.
- Non-keys which are dependent on a part of composite key should be moved to a new table and identify its keys.

In the following situation, there are no Partial dependencies as all the non-key elements are directly dependent on their respective primary key.

Package_ID ➔ Pakage_Name, Total_Days, Difficulty

Package_ID and Destination gives the unique value for every package-destination pair:

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)

Destination_Package (**Package_ID***, **Destination**)

2.1.4. Third Normal Form (3NF):

For the database to be in Third Normal Form it must already be in Second Normal Form and should not include any transitive dependencies. Transitive dependency is a type of functional dependency in which a non-key is dependent on another non-key element (Coronel & Morris, 2018). The rules for 3NF are:

- Functional dependencies between no key and non-key should be separated into a new table in case of an entity with multiple Non-key.
- Primary Keys of the new table should be identified.

There are no transitive dependencies as every none key entries are dependent on the primary key assigned for each table.

Package (**Package_ID**, Pakage_Name, Total_Days, Difficulty)

Destination_Package (**Package_ID***, **Destination**)

Since there are no transitive dependencies this is the final normalized form

2.2. Figure 2

The second figure gives us detail about the tracking information.

Package Name: Ghandruk

Start date: 2018/Jan/05

End date: 2018/Jan/09

Tour Guide: Will Stark

Day	Travel Details	Activities	Status	Travel Mode	Difficulty Level
Day 1	Kathmandu to Pokhara	Driving from KTM to Pokhara Overnight stay in Hotel	Complete	Bus	Easy
Day 2	Pokhara to Ghandruk	Trek to Ghandruk. Explore the Ghandruk Village.	Complete	Bus/Walk	Hard
Day 3	Ghandruk to Pokhara	View the beautiful sunrise and Himalayas. Trek Down to Pokhara	Complete	Bus/Walk	Hard
Day 4	Pokhara to Kathmandu	Drive back to Pokhara	Remaining	Bus	Moderate

Figure 3: Figure 2 as provided by the question for tracking information

2.2.1. Assumptions For Figure 2:

- Activity, Travel Mode and Status depends on the Activity ID assigned for dividing activities
- A day in a package will determine travel details, difficulty and unique sets of activities consisting of activity, travel mode and the status of the day.
- Package ID details the tracking information of the tour.
- Status describes the activity progress

Storing the given records in unique cells we get the following table:

Table 2: Storing the given data in Atomic cells

Package ID	Package Name	Start Date	End Date	Guide	Day	Travel Details	Difficulty	Activity ID	Activities	Travel Mode	Status
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 1	Kathmandu to Pokhara	Easy	A1	Driving from KTM to Pokhara.	Bus	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 1	Kathmandu to Pokhara	Easy	A2	Overnight stay in the Hotel.	Walk	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 2	Pokhara to Ghandruk	Hard	A3	Trek to Ghandruk.	Bus	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 2	Pokhara to Ghandruk	Hard	A4	Explore the Ghandruk Village.	Walk	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 3	Ghandruk to Pokhara	Easy	A5	View the sunrise and Himalayas.	Walk	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 3	Ghandruk to Pokhara	Easy	A6	Trek Down to Pokhara	Bus	Complete
UI32A	Ghandruk	1st Jan 2019	7th Jan 2019	Will Stark	Day 4	Pokhara to Kathmandu	Moderate	A7	Drive back to Kathmandu	Bus	Remaining

2.2.2. Un-normalized Form (UNF):

From the above table we can observe that the destination column has repeating groups in it, representing the given data in UNF:

UNF: (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide, {Day, Travel_Details, Difficulty_Level, {Activity_ID, Activity, Travel_Mode, Status}})

2.2.3. First Normal Form (1NF):

Package ID is taken as the Primary key as it gives

Package_ID ➔ Package_Name, Start_Date, End_Date, Tour_Guide,

Package_ID, Day ➔ Travel_Details, Difficulty_Level

Package_ID, Day, Activity_ID ➔ Activity, Travel_Mode, Status

Removing the repeating group and creating the Itenary_Tour. Assigning Package_ID as Primary key and creating the first normal form.

Tour (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide)

Itenary_Tour (**Package_ID***, **Day**, Travel_Details, Difficulty_Level)

Day_Activities (**Package_ID***, **Day***, **Activity_ID**, Activity, Travel_Mode, Status)

2.2.4. Second Normal Form (2NF):

Here only Activity and Travel mode are in part dependent because it is not related to the Package ID and Package ID but directly dependent on the Activity ID.

Package_ID ➔ Package_Name, Start_Date, End_Date, Tour_Guide

Package_ID, Day ➔ Travel_Details, Difficulty_Level

Package_ID ➔ **X**

Day ➔ **X**

Package_ID, Day, Activity_ID ➔ Status

Activity_ID ➔ Travel_Mode, Activities

Removing the partial dependency, we get:

Tour (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide)

Itenary_Tour (**Package_ID***, **Day**, Travel_Details, Difficulty_Level, Status)

Day_Activities (**Package_ID***, **Day***, **Activity_ID**, Status)

Activities(**Activity_ID**, Activity, Travel_Mode)

2.2.5. Third Normal Form (3NF):

There are no transitive dependencies as every none key entries are dependent on the primary key assigned for each table.

Tour (**Package_ID**, Package_Name, Start_Date, End_Date, Tour_Guide)

Itenary_Tour (**Package_ID***, **Day**, Travel_Details, Difficulty_Level)

Day_Activities (**Package_ID***, **Day***, **Activity_ID***, Status)

Activities(**Activity_ID**, Activities, Travel_Mode)

Since there are no transitive dependencies this is the final normalized form

2.3. Integration

Combining the tables, we get the following entities. Also adding entities for staffs and customers.

Day_activities (**Package_Id*, Day*, Activity_Id***, Status)

Activity (**Activity_Id**, Activity_Name, Travel_Mode)

Itinerary (**Package_Id*, Day**, Difficulty_Level, Travel_Details)

Destination (**Package_Id***, **Destination**)

Packages (**Package_Id*,** Package_Name, Total_Days, Start_Date, End_Date, Difficulty)

Customer_Package (**Customer_Id*, Package_Id***)

Package_Guide (**Package_Id*, Staff_Id***)

Customer (**Customer_Id**, First_Name, Last_Name Address, Phone_No)

Staff (**Staff_Id**, First_Name, Last_Name, Address, Phone_No, Role_Id)

Role (**Role_Id**, Description)

3. ER-Diagram

An Entity-relationship diagram represents the relationship between the entities in the database. ERD is one of the most common data but effective models where objects are divided into entities and their characteristics into attributes and entities are connected via elaborate relationships. (Nishadha, 2017). SQL Developer Data Modeler is a free graphical tool that improves productivity and simplifies data modelling tasks where users can create, browse and edit, logical, relational, physical, multi-dimensional, and data type models supporting collaborative development through integrated source code control (Oracle, 2020). SQL Developer Data Modeler was used to Create the final ER-Diagram.

Now, we have all the tables from figure 1 and 2 we can observe that the package table from figure 1 and tour table from figure 2 gives the same information. Now for integration the created tables into one and adding new tables for clarity we need to lay down some assumptions.

3.1. Assumptions:

- Package from the figure(1) and tour from the figure(2) represent the same table.
- Difficulty in package table gives the difficulty in the overall package
- The difficulty in Itinerary gives the difficulty level in each day of the package.
- Staffs have Name (first name and last name), address and phone number.
- Staffs are identified by Staff ID.
- Guides are also staff and have guide ID which is referenced to the staff ID in a tour.
- Each staff have only one role assigned to them by the company
- The guide from the package is removed as a package might have multiple guides.
- While adding guides in the tour guide table, the user must only enter the staff who is a guide. (ie. The database will accept a receptionist as a tour guide but that is not ideal)
- Customers have Name (first name and last name), address and phone number.
- Staffs are identified by Staff ID
- Customers can only book one package only once at a time
- A Package can have no guide assigned; the user must avoid while entering data

3.2. Final ER-diagram:

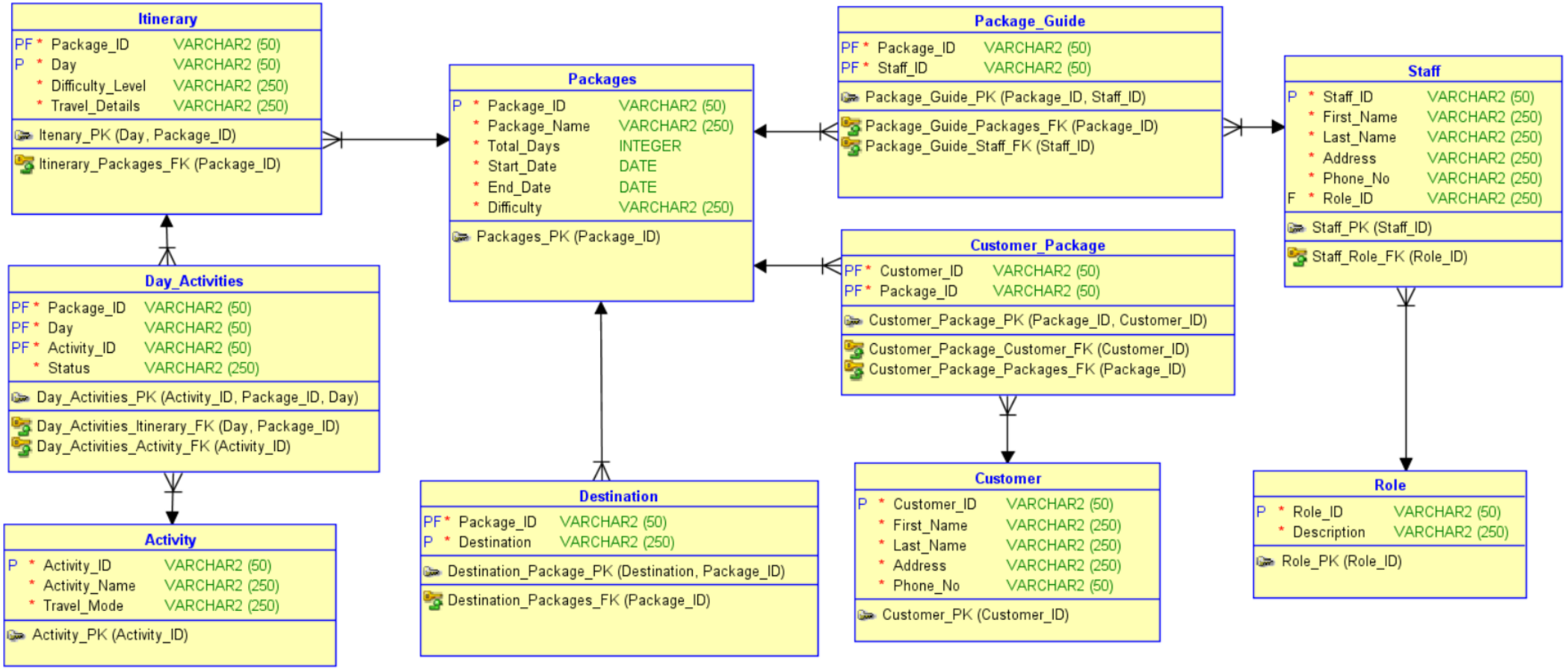

Figure 4: ER-Diagram created using the given entities

4. Data Dictionary

Table 3: Data Dictionary for the Activity table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Activity_ID	Y	Logical Type	VARCHAR	P		VARCHAR2	50	"A1"
Activity_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"Doing This"
Travel_Mode	Y	Logical Type	VARCHAR			VARCHAR2	250	"Bus"

Table 4: Data Dictionary for Day Activities table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Activity_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"A1"
Day	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"Day 1"
Package_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"GAND8"
Status	Y	Logical Type	VARCHAR			VARCHAR2	250	"Incomplete"

Table 5: Data Dictionary for Itinerary Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Day	Y	Logical Type	VARCHAR	P		VARCHAR2	50	"Day 1"
Difficulty_Level	Y	Logical Type	VARCHAR			VARCHAR2	250	"Hard"
Package_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"GAND8"
Travel_Details	Y	Logical Type	VARCHAR			VARCHAR2	250	"KTM - PKR"

Table 6: Data Dictionary for Destination Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Destination	Y	Logical Type	VARCHAR	P		VARCHAR2	250	"Kathmandu"
Package_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"GAND8"

Table 7: Data Dictionary for Package Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Difficulty	Y	Logical Type	VARCHAR			VARCHAR2	250	"Hard"
End_Date	Y	Logical Type	Date			DATE		"1 JAN 2020"
Package_ID	Y	Logical Type	VARCHAR	P		VARCHAR2	50	"GAND8"
Package_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"ABC Trek"
Start_Date	Y	Logical Type	Date			DATE		"1 JAN 2020"
Total_Days	Y	Logical Type	Integer			INTEGER		4

Table 8: Data Dictionary for Staff Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Address	Y	Logical Type	VARCHAR			VARCHAR2	250	"Jamal"
First_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"Ram"
Last_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"Karki"
Phone_No	Y	Logical Type	VARCHAR			VARCHAR2	250	213122323
Role_ID	Y	Logical Type	VARCHAR		F	VARCHAR2	250	"G"
Staff_ID	Y	Logical Type	VARCHAR	P		VARCHAR2	50	"S110"

Table 9: Data Dictionary for Customer Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Address	Y	Logical Type	VARCHAR			VARCHAR2	250	"Jamal"
Customer_ID	Y	Logical Type	VARCHAR	P		VARCHAR2	50	"C119"
First_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"Ram"
Last_Name	Y	Logical Type	VARCHAR			VARCHAR2	250	"Karki"
Phone_No	Y	Logical Type	VARCHAR			VARCHAR2	50	213122323

Table 10: Data Dictionary for Customer-Package Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Customer_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"C119"
Package_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	"GAND8"

Table 11: Data Dictionary for Package Guide Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Package_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	“GAND8”
Staff_ID	Y	Logical Type	VARCHAR	P	F	VARCHAR2	50	“S110”

Table 12: Data Dictionary for role Table

Column_Name	Mandatory	DataType Kind	Logical Type Name	PK	FK	Native Type	T Size	Example
Description	Y	Logical Type	VARCHAR			VARCHAR2	250	“Does This”
Role_ID	Y	Logical Type	VARCHAR	P		VARCHAR2	50	“DT”

5. Generation of Database

5.1. Create Statements

5.1.1. Generating DDL Script and Creating Tables:

The Datamodeler allows the generation of DDL Script according to the designed ER-Model. The following figure shows how the script was created and the script is also included. Then the script is pasted into the SQL Developer and ran to generate the tables.

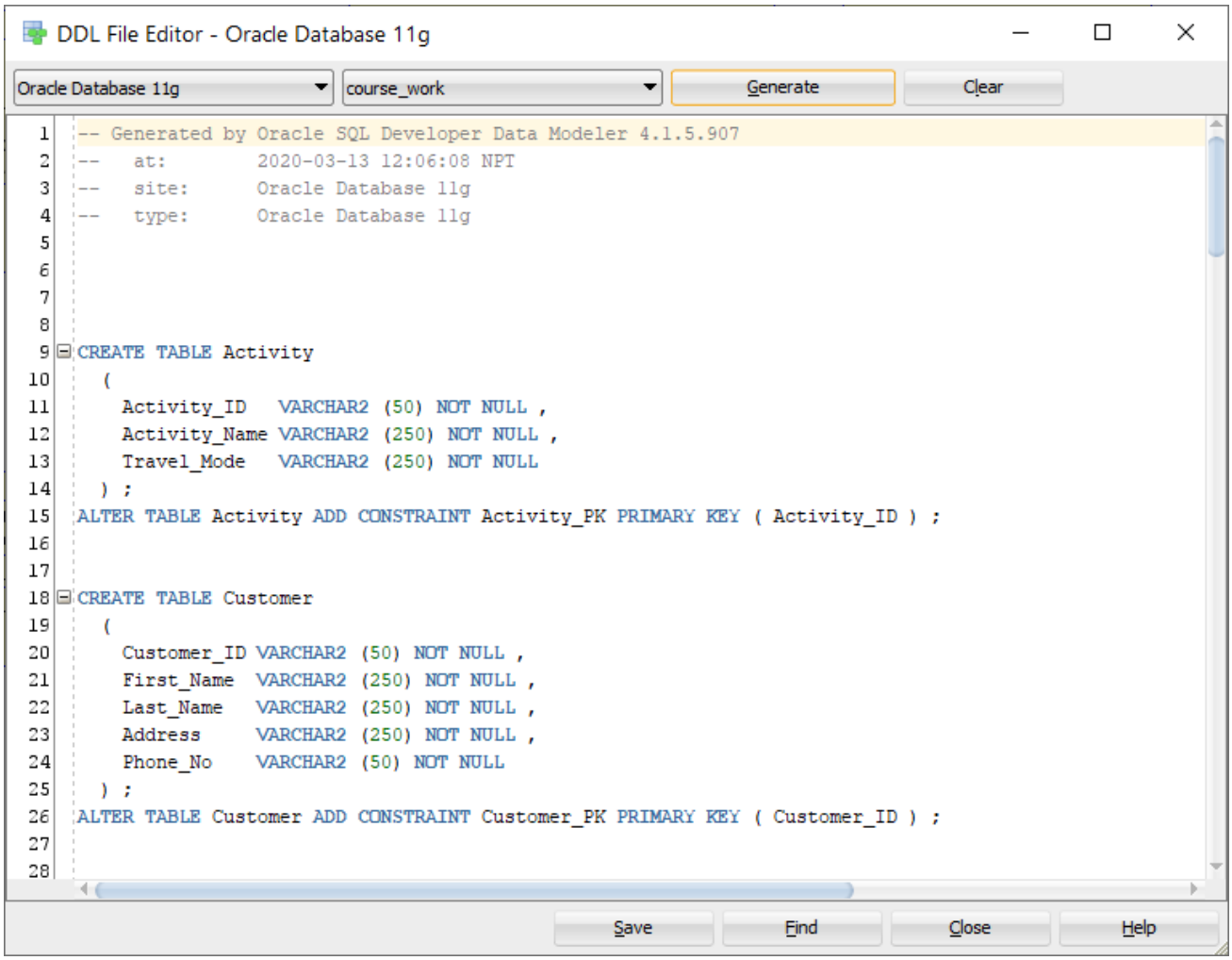

Figure 5: Process of generating the DDL Script via SQL developer Datamodeler

DDL Script:

```
CREATE TABLE Activity
  (
    Activity_ID   VARCHAR2 (50) NOT NULL ,
    Activity_Name VARCHAR2 (250) NOT NULL ,
    Travel_Mode   VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Activity ADD CONSTRAINT Activity_PK PRIMARY KEY ( Activity_ID ) ;


CREATE TABLE Customer
  (
    Customer_ID VARCHAR2 (50) NOT NULL ,
    First_Name  VARCHAR2 (250) NOT NULL ,
    Last_Name   VARCHAR2 (250) NOT NULL ,
    Address     VARCHAR2 (250) NOT NULL ,
    Phone_No    VARCHAR2 (50) NOT NULL
  ) ;
ALTER TABLE Customer ADD CONSTRAINT Customer_PK PRIMARY KEY ( Customer_ID ) ;


CREATE TABLE Customer_Package
  (
    Customer_ID VARCHAR2 (50) NOT NULL ,
    Package_ID  VARCHAR2 (50) NOT NULL
  ) ;
ALTER TABLE Customer_Package ADD CONSTRAINT Customer_Package_PK PRIMARY KEY ( Packag
e_ID, Customer_ID ) ;


CREATE TABLE Day_Activities
  (
    Package_ID  VARCHAR2 (50) NOT NULL ,
    DAY         VARCHAR2 (50) NOT NULL ,
    Activity_ID VARCHAR2 (50) NOT NULL ,
    Status      VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Day_Activities ADD CONSTRAINT Day_Activities_PK PRIMARY KEY ( Activity_I
D, Package_ID, DAY ) ;


CREATE TABLE Destination
  (
    Package_ID  VARCHAR2 (50) NOT NULL ,
    Destination VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Destination ADD CONSTRAINT Destination_Package_PK PRIMARY KEY ( Destinat
ion, Package_ID ) ;
```

```
CREATE TABLE Itinerary
  (
    Package_ID       VARCHAR2 (50) NOT NULL ,
    DAY              VARCHAR2 (50) NOT NULL ,
    Difficulty_Level VARCHAR2 (250) NOT NULL ,
    Travel_Details   VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Itinerary ADD CONSTRAINT Itenary_PK PRIMARY KEY ( DAY, Package_ID ) ;


CREATE TABLE Package_Guide
  (
    Package_ID VARCHAR2 (50) NOT NULL ,
    Staff_ID   VARCHAR2 (50) NOT NULL
  ) ;
ALTER TABLE Package_Guide ADD CONSTRAINT Package_Guide_PK PRIMARY KEY ( Package_ID,
Staff_ID ) ;


CREATE TABLE Packages
  (
    Package_ID   VARCHAR2 (50) NOT NULL ,
    Package_Name VARCHAR2 (250) NOT NULL ,
    Total_Days   INTEGER NOT NULL ,
    Start_Date   DATE NOT NULL ,
    End_Date     DATE NOT NULL ,
    Difficulty   VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Packages ADD CONSTRAINT Packages_PK PRIMARY KEY ( Package_ID ) ;


CREATE TABLE Role
  (
    Role_ID     VARCHAR2 (50) NOT NULL ,
    Description VARCHAR2 (250) NOT NULL
  ) ;
ALTER TABLE Role ADD CONSTRAINT Role_PK PRIMARY KEY ( Role_ID ) ;


CREATE TABLE Staff
  (
    Staff_ID   VARCHAR2 (50) NOT NULL ,
    First_Name VARCHAR2 (250) NOT NULL ,
    Last_Name  VARCHAR2 (250) NOT NULL ,
    Address    VARCHAR2 (250) NOT NULL ,
    Phone_No   VARCHAR2 (250) NOT NULL ,
    Role_ID    VARCHAR2 (250) NOT NULL
```

```
  ) ;
ALTER TABLE Staff ADD CONSTRAINT Staff_PK PRIMARY KEY ( Staff_ID ) ;


ALTER TABLE Customer_Package ADD CONSTRAINT Customer_Package_Customer_FK FOREIGN KEY
 ( Customer_ID ) REFERENCES Customer ( Customer_ID ) ;

ALTER TABLE Customer_Package ADD CONSTRAINT Customer_Package_Packages_FK FOREIGN KEY
 ( Package_ID ) REFERENCES Packages ( Package_ID ) ;

ALTER TABLE Day_Activities ADD CONSTRAINT Day_Activities_Activity_FK FOREIGN KEY ( A
ctivity_ID ) REFERENCES Activity ( Activity_ID ) ;

ALTER TABLE Day_Activities ADD CONSTRAINT Day_Activities_Itinerary_FK FOREIGN KEY (
DAY, Package_ID ) REFERENCES Itinerary ( DAY, Package_ID ) ;

ALTER TABLE Destination ADD CONSTRAINT Destination_Packages_FK FOREIGN KEY ( Package
_ID ) REFERENCES Packages ( Package_ID ) ;

ALTER TABLE Itinerary ADD CONSTRAINT Itinerary_Packages_FK FOREIGN KEY ( Package_ID
) REFERENCES Packages ( Package_ID ) ;

ALTER TABLE Package_Guide ADD CONSTRAINT Package_Guide_Packages_FK FOREIGN KEY ( Pac
kage_ID ) REFERENCES Packages ( Package_ID ) ;

ALTER TABLE Package_Guide ADD CONSTRAINT Package_Guide_Staff_FK FOREIGN KEY ( Staff_
ID ) REFERENCES Staff ( Staff_ID ) ;

ALTER TABLE Staff ADD CONSTRAINT Staff_Role_FK FOREIGN KEY ( Role_ID ) REFERENCES Ro
le ( Role_ID ) ;
```

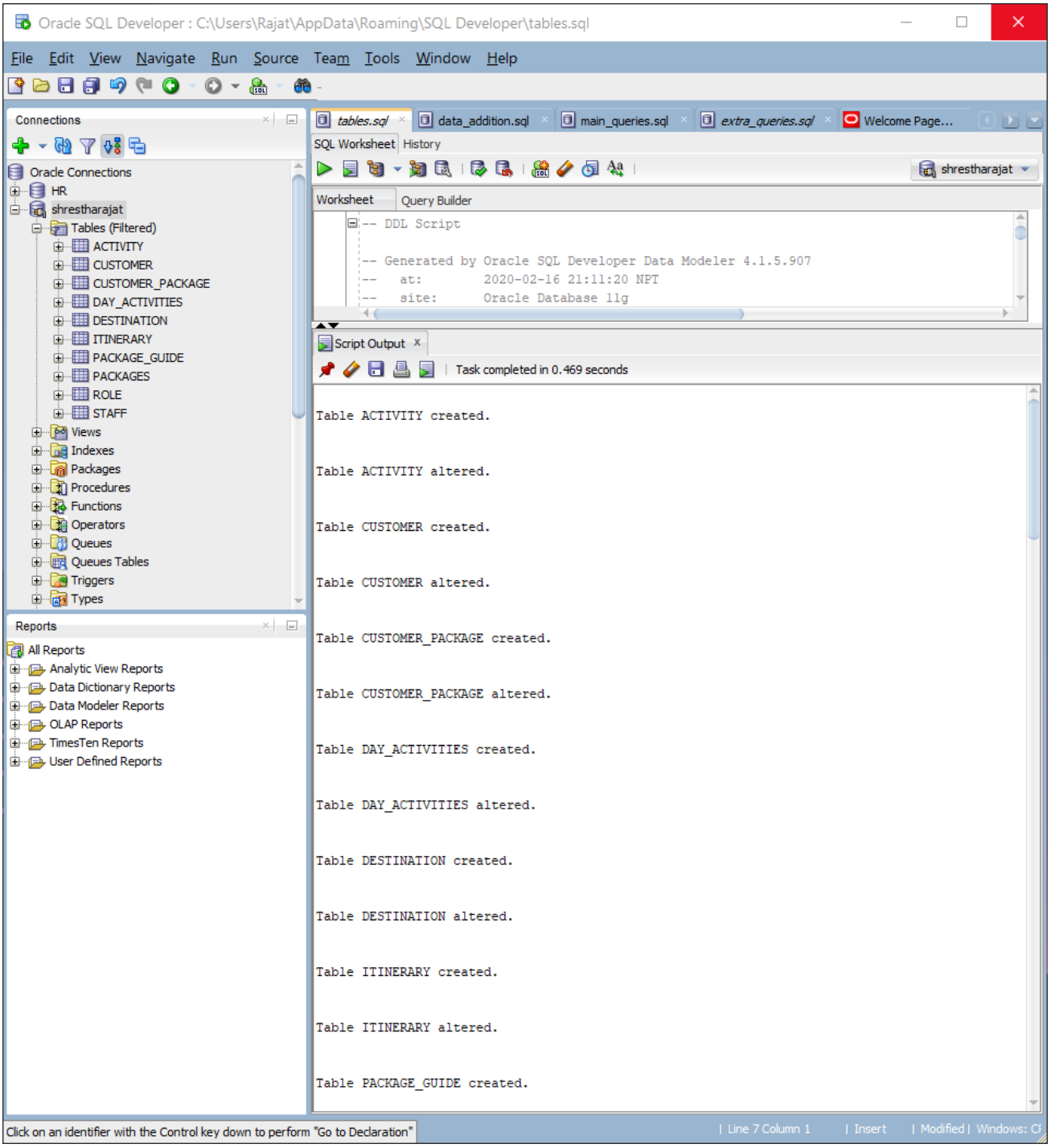

Figure 6: Running the DDL Script in SQL Developer

5.2. Insert Statements

After creating the tables, they need to be populated. The following data were inserted into the database for testing the database via sql developer

```
-- Populating the database
INSERT INTO packages (PACKAGE_ID, PACKAGE_NAME, TOTAL_DAYS, START_DATE, END_DATE, DI
FFICULTY)
    WITH names AS (
        SELECT 'GAND8', 'Ghandruk', 4, '01-JAN-2020',
            '04-JAN-2020', 'Moderate' FROM dual UNION ALL
        SELECT 'ABC21', 'Annapurna BC', 6, '12-FEB-2020',
            '17-FEB-2020', 'Hard' FROM dual UNION ALL
        SELECT 'PUNH1', 'Poon-Hill', 5, '01-JAN-2020',
            '05-JAN-2020', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'Everest BC', 8, '01-JAN-2020',
            '08-JAN-2020', 'Hard' FROM dual UNION ALL
        SELECT 'LUKL1', 'Lukla', 3, '01-MAR-2020',
            '03-MAR-2020', 'Moderate' FROM dual
    )
    SELECT * FROM names;


INSERT INTO destination (PACKAGE_ID, DESTINATION)
    WITH names AS (
        SELECT 'GAND8', 'Pokhara' FROM dual UNION ALL
        SELECT 'GAND8', 'Ghandruk' FROM dual UNION ALL
        SELECT 'ABC21', 'Pokhara' FROM dual UNION ALL
        SELECT 'ABC21', 'Ghandruk' FROM dual UNION ALL
        SELECT 'ABC21', 'ABC' FROM dual UNION ALL
        SELECT 'PUNH1', 'Pokhara' FROM dual UNION ALL
        SELECT 'PUNH1', 'Ghorepani' FROM dual UNION ALL
        SELECT 'PUNH1', 'PoonHill' FROM dual UNION ALL
        SELECT 'PUNH1', 'Ghandruk' FROM dual UNION ALL
        SELECT 'LUKL1', 'Lukla' FROM dual UNION ALL
        SELECT 'EBC77', 'Lukla' FROM dual UNION ALL
        SELECT 'EBC77', 'Namche-Bazzar' FROM dual UNION ALL
        SELECT 'EBC77', 'Dingboche' FROM dual UNION ALL
        SELECT 'EBC77', 'EBC' FROM dual
        )
    SELECT * FROM names;


INSERT INTO itinerary (PACKAGE_ID, DAY, TRAVEL_DETAILS, DIFFICULTY_LEVEL)
    WITH names AS (
        SELECT 'GAND8', 'day-1', 'Kathmandu-Pokhara', 'Easy' FROM dual UNION ALL
        SELECT 'GAND8', 'day-2', 'Pokhara-Ghandruk', 'Hard' FROM dual UNION ALL
        SELECT 'GAND8', 'day-3', 'Ghandruk-Pokhara', 'Hard' FROM dual UNION ALL
```

```
        SELECT 'GAND8', 'day-4', 'Pokhara-Kathmandu', 'Easy' FROM dual UNION ALL
        SELECT 'ABC21', 'day-1', 'Kathmandu-Pokhara', 'Easy' FROM dual UNION ALL
        SELECT 'ABC21', 'day-2', 'Pokhara-Ghandruk', 'Hard' FROM dual UNION ALL
        SELECT 'ABC21', 'day-3', 'Ghandruk-ABC', 'Hard' FROM dual UNION ALL
        SELECT 'ABC21', 'day-4', 'ABC-Ghandruk', 'Hard' FROM dual UNION ALL
        SELECT 'ABC21', 'day-5', 'Ghandruk-Pokhara', 'Hard' FROM dual UNION ALL
        SELECT 'ABC21', 'day-6', 'Pokhara-Kathmandu', 'Easy' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-1', 'Kathmandu-Pokhara', 'Easy' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-2', 'Pokhara-Ghorepani', 'Hard' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-3', 'Gorepani-Ghandruk', 'Hard' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-4', 'Ghandruk-Pokhara', 'Hard' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-5', 'Pokhara-Kathmandu', 'Easy' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-1', 'Kathmandu-Lukla', 'Easy' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-2', 'Lukla', 'Easy' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-3', 'Lukla-Kathmandu', 'Easy' FROM dual UNION ALL
        SELECT 'EBC77', 'day-1', 'Kathmandu-Lukla', 'Easy' FROM dual UNION ALL
        SELECT 'EBC77', 'day-2', 'Lukla-Namche Bazzar', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'day-3',
                  'Namche Bazzar-Dingboche', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'day-4', 'Dingboche-EBC', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'day-5', 'EBC-Dingboche', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'day-6',
                  'Dingboche-Namche Bazzar', 'Hard' FROM dual UNION ALL
        SELECT 'EBC77', 'day-7', 'Namche Bazzar-Lukla', 'Easy' FROM dual UNION ALL
        SELECT 'EBC77', 'day-8', 'Lukla-Kathmandu', 'Easy' FROM dual
    )
    SELECT * FROM names;


INSERT INTO activity (ACTIVITY_ID, ACTIVITY_NAME, TRAVEL_MODE)
    WITH names AS (
        SELECT 'A1', 'Driving from KTM to Pokhara', 'bus' FROM dual UNION ALL
        SELECT 'A2', 'Overnight stay in Hotel', 'stay' FROM dual UNION ALL
        SELECT 'A3', 'Trek to Ghandruk', 'walk' FROM dual UNION ALL
        SELECT 'A4', 'Explore the Ghandruk Village', 'walk' FROM dual UNION ALL
        SELECT 'A5', 'View the sunrise and Himalayas', 'walk' FROM dual UNION ALL
        SELECT 'A6', 'Trek Down to Pokhara', 'walk' FROM dual UNION ALL
        SELECT 'A7', 'Drive back to Kathmandu', 'walk' FROM dual UNION ALL
        SELECT 'A8', 'Trek to ABC from Ghandruk', 'walk' FROM dual UNION ALL
        SELECT 'A9', 'Explore ABC', 'walk' FROM dual UNION ALL
        SELECT 'A10', 'Trek back to Ghandruk from ABC', 'walk' FROM dual UNION ALL
        SELECT 'A11', 'Bus to halfway for ghorepani', 'walk' FROM dual UNION ALL
        SELECT 'A12', 'Trek to Ghorepani', 'walk' FROM dual UNION ALL
        SELECT 'A13', 'Hike to Poonhill', 'walk' FROM dual UNION ALL
        SELECT 'A14', 'Viewing Sunrise', 'walk' FROM dual UNION ALL
        SELECT 'A15', 'Flight to Lukla', 'plane' FROM dual UNION ALL
        SELECT 'A16', 'Explore Lukla', 'walk' FROM dual UNION ALL
        SELECT 'A17', 'Overnight stay in Lukla', 'stay' FROM dual UNION ALL
        SELECT 'A18', 'Flight from Lukla to kathmandu', 'plane' FROM dual UNION ALL
```

```
        SELECT 'A19', 'Hike from Lukla to Namche bazzar', 'walk' FROM dual UNION ALL
        SELECT 'A20', 'Trek from Namche to Dingboche', 'walk' FROM dual UNION ALL
        SELECT 'A21', 'Trek from Dingboche to EBC', 'walk' FROM dual UNION ALL
        SELECT 'A22', 'Explore EBC', 'walk' FROM dual UNION ALL
        SELECT 'A23', 'Trek from EBC to Dingboche', 'walk' FROM dual UNION ALL
        SELECT 'A24', 'Trek from Dingboche to Namche', 'walk' FROM dual UNION ALL
        SELECT 'A25', 'Trek from Namche bazzar to Lukla', 'walk' FROM dual
    )
    SELECT * FROM names;


INSERT INTO day_activities (PACKAGE_ID, DAY, ACTIVITY_ID, STATUS)
    WITH names AS (
        SELECT 'GAND8', 'day-1', 'A1', 'Incomplete' FROM dual UNION ALL
        SELECT 'GAND8', 'day-1', 'A2', 'Incomplete' FROM dual UNION ALL
        SELECT 'GAND8', 'day-2', 'A3', 'Incomplete' FROM dual UNION ALL
        SELECT 'GAND8', 'day-2', 'A4', 'Incomplete' FROM dual UNION ALL
        SELECT 'GAND8', 'day-3', 'A5', 'Incomplete' FROM dual UNION ALL
        SELECT 'GAND8', 'day-4', 'A6', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-1', 'A1', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-1', 'A2', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-2', 'A3', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-2', 'A4', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-3', 'A8', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-3', 'A9', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-4', 'A10', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-5', 'A5', 'Incomplete' FROM dual UNION ALL
        SELECT 'ABC21', 'day-6', 'A6', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-1', 'A1', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-1', 'A2', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-2', 'A11', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-2', 'A12', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-3', 'A13', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-3', 'A14', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-3', 'A3', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-4', 'A5', 'Incomplete' FROM dual UNION ALL
        SELECT 'PUNH1', 'day-5', 'A6', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-1' , 'A15', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-1' , 'A16', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-1' , 'A17', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-2' , 'A16', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-2' , 'A17', 'Incomplete' FROM dual UNION ALL
        SELECT 'LUKL1', 'day-3' , 'A18', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-1' , 'A15', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-1' , 'A16', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-1' , 'A17', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-2' , 'A19', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-3' , 'A20', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-4' , 'A21', 'Incomplete' FROM dual UNION ALL
```

```
        SELECT 'EBC77', 'day-4' , 'A22', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-5' , 'A23', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-6' , 'A24', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-7' , 'A25', 'Incomplete' FROM dual UNION ALL
        SELECT 'EBC77', 'day-8' , 'A18', 'Incomplete' FROM dual
    )
    SELECT * FROM names;


INSERT INTO role (ROLE_ID, DESCRIPTION)
    WITH names AS (
        SELECT 'TA', 'Tour Agent, helps customer' FROM dual UNION ALL
        SELECT 'G', 'Guide, goes to tours' FROM dual UNION ALL
        SELECT 'MA', 'Manager, the main manager of the office' FROM dual UNION ALL
        SELECT 'DV', 'Driver, driver for the company' FROM dual UNION ALL
        SELECT 'IN', 'Intern, agent intern in the company' FROM dual
    )
    SELECT * FROM names;


INSERT INTO staff (STAFF_ID, FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO, ROLE_ID)
    WITH names AS (
        SELECT 'S001', 'Rajat', 'Shrestha', 'Samakhushi, TownPlanning',
                  '9182934234', 'G' FROM dual UNION ALL
        SELECT 'S002', 'Ashley', 'Green', 'Baluwatar',
                  '9182343123', 'TA' FROM dual UNION ALL
        SELECT 'S003', 'Shia', 'Miton', 'patan',
                  '9182323423', 'TA' FROM dual UNION ALL
        SELECT 'S004', 'Ramendra', 'Sharma', 'Gongabu',
                  '9184322123', 'G' FROM dual UNION ALL
        SELECT 'S005', 'Silas', 'BK', 'Butwal',
                  '9182931112', 'G' FROM dual UNION ALL
        SELECT 'S006', 'Rabin', 'Tamang', 'Sankhamul',
                  '91323934234', 'G' FROM dual UNION ALL
        SELECT 'S007', 'Prabin', 'Maskey', 'Baluwatar',
                  '9182343123', 'G' FROM dual UNION ALL
        SELECT 'S008', 'Pranish', 'Chettri', 'baneshwor',
                  '9132323423', 'IN' FROM dual UNION ALL
        SELECT 'S009', 'Ramu', 'Sharma', 'Gongabu',
                  '9184322123', 'G' FROM dual UNION ALL
        SELECT 'S010', 'Ganesh', 'KC', 'Birendranagar',
                  '9182931112', 'MA' FROM dual
    )
    SELECT * FROM names;


INSERT INTO customer (CUSTOMER_ID, FIRST_NAME, LAST_NAME, ADDRESS, PHONE_NO)
    WITH names AS (
```

```
        SELECT 'C001', 'Ramesh', 'Nakarmi', 'Samakhushi, TownPlanning', '9182934234'
 FROM dual UNION ALL
    SELECT 'C002', 'John', 'LeBron', 'Baluwatar', '9182343423'  FROM dual UNION ALL
    SELECT 'C003', 'Saran', 'Manandar', 'patan', '9182323433'  FROM dual UNION ALL
    SELECT 'C004', 'Susma', 'Shakya', 'Gongabu', '9184422123' FROM dual UNION ALL
    SELECT 'C005', 'Siliya', 'Malla', 'Butwal', '9182931111'  FROM dual UNION AL
    SELECT 'C006', 'Shreyash', 'Subedi', 'Sankhamul', '91224234' FROM dual UNION ALL
    SELECT 'C007', 'Bhuwan', 'Khanal', 'Baluwatar', '9382343123' FROM dual UNION ALL
    SELECT 'C008', 'Prashant', 'Budha', 'baneshwor', '933223423' FROM dual UNION ALL
    SELECT 'C009', 'Gopal', 'KC', 'Gongabu', '9184323223' FROM dual UNION ALL
    SELECT 'C010', 'Naren', 'Bajracharya', 'Birendranagar', '9123931112'  FROM dual
    )
    SELECT * FROM names;

INSERT INTO customer_package (CUSTOMER_ID, PACKAGE_ID)
    WITH names AS (
        SELECT 'C001', 'GAND8' FROM dual UNION ALL
        SELECT 'C002', 'EBC77' FROM dual UNION ALL
        SELECT 'C003', 'EBC77' FROM dual UNION ALL
        SELECT 'C004', 'EBC77' FROM dual UNION ALL
        SELECT 'C005', 'PUNH1' FROM dual UNION ALL
        SELECT 'C006', 'ABC21' FROM dual UNION ALL
        SELECT 'C007', 'PUNH1' FROM dual UNION ALL
        SELECT 'C008', 'ABC21' FROM dual UNION ALL
        SELECT 'C009', 'LUKL1' FROM dual UNION ALL
        SELECT 'C010', 'GAND8' FROM dual
    )
    SELECT * FROM names;

INSERT INTO package_guide (PACKAGE_ID, STAFF_ID)
    WITH names AS (
        SELECT 'GAND8', 'S001' FROM dual UNION ALL
        SELECT 'ABC21', 'S004' FROM dual UNION ALL
        SELECT 'ABC21', 'S005' FROM dual UNION ALL
        SELECT 'PUNH1', 'S006' FROM dual UNION ALL
        SELECT 'EBC77', 'S007' FROM dual UNION ALL
        SELECT 'EBC77', 'S009' FROM dual
    )
    SELECT * FROM names;
```

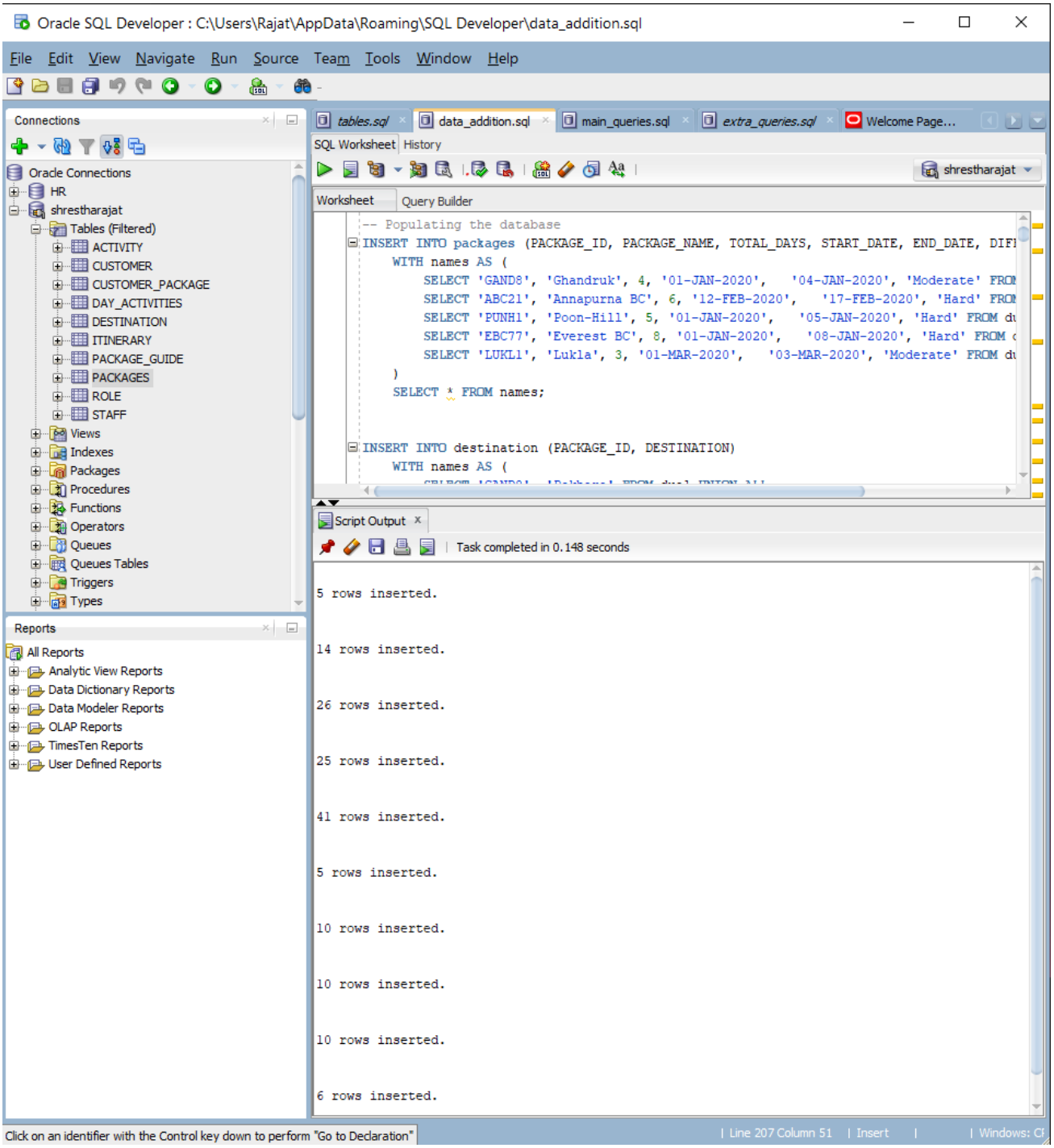

Figure 7: Running the Insert Statements to populate the database in SQL Developer

5.3. Select Statements

The data inserted are checked using the select statement for each table.

5.3.1. Day Activities

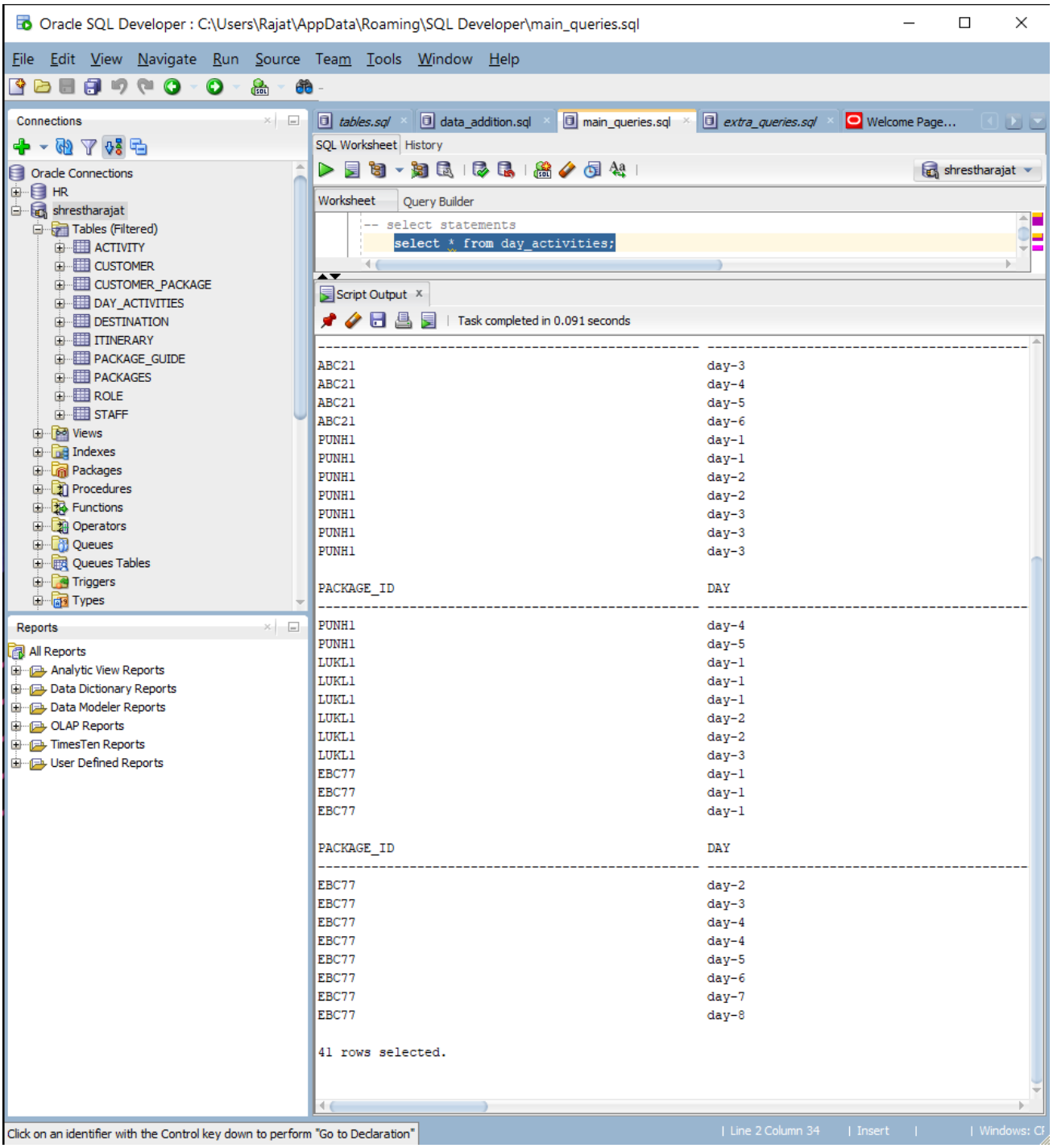

Figure 8: Running select Statement on Day Activities Table

5.3.2. Activity

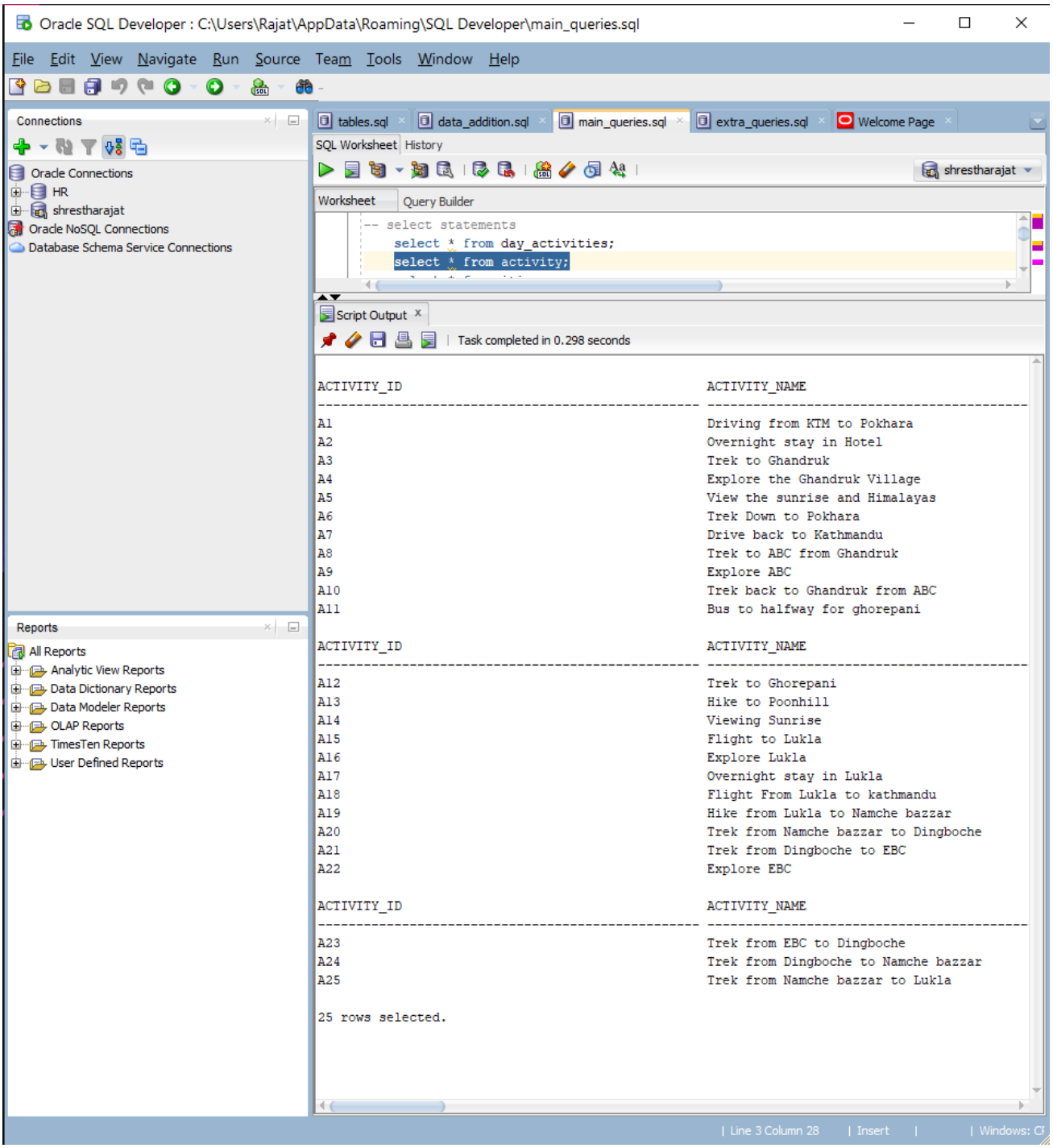

Figure 9: Running select Statement on Activity Table

5.3.3. Itinerary

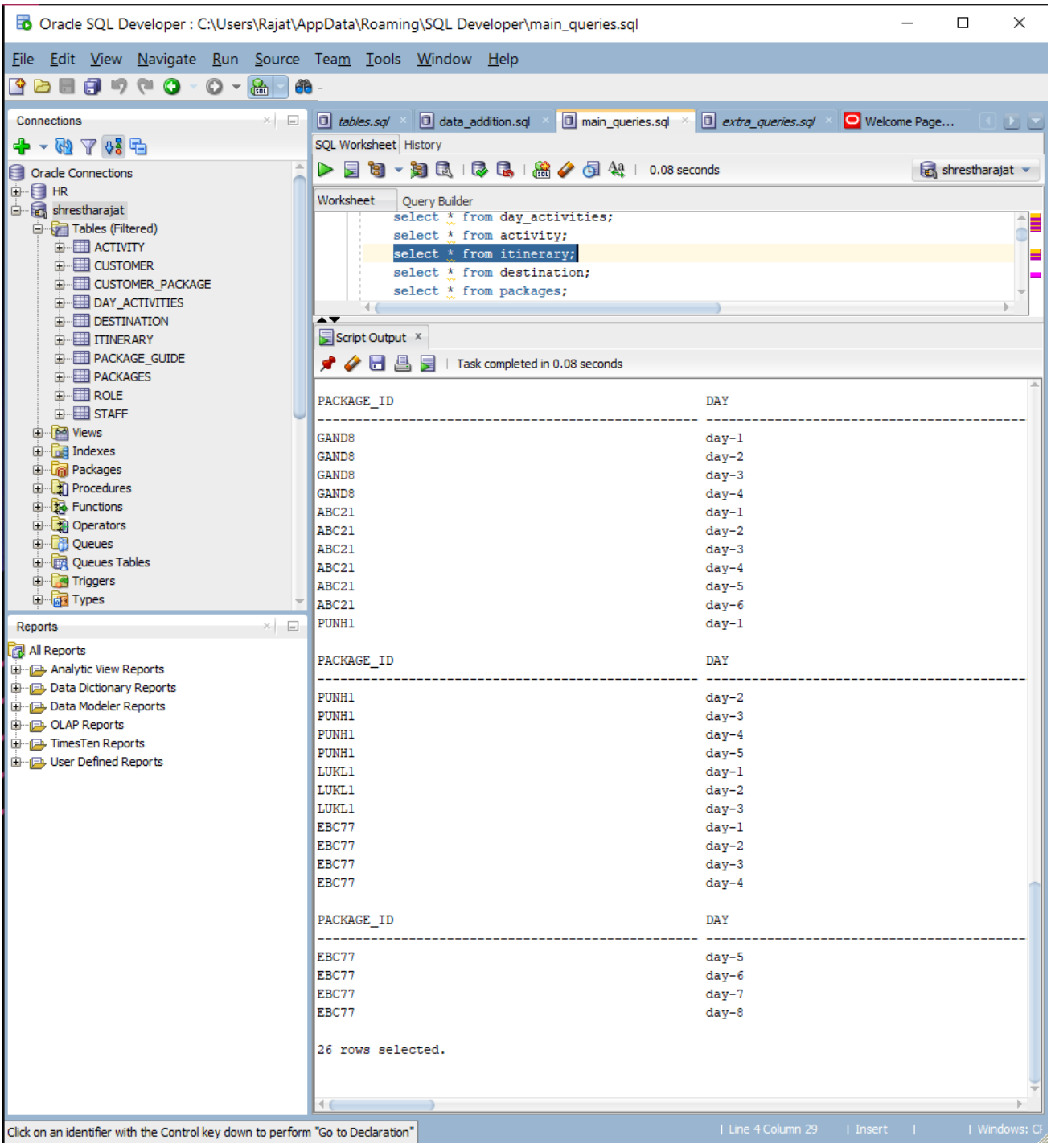

Figure 10: Running select Statement on Itinerary Table

5.3.4. Destination

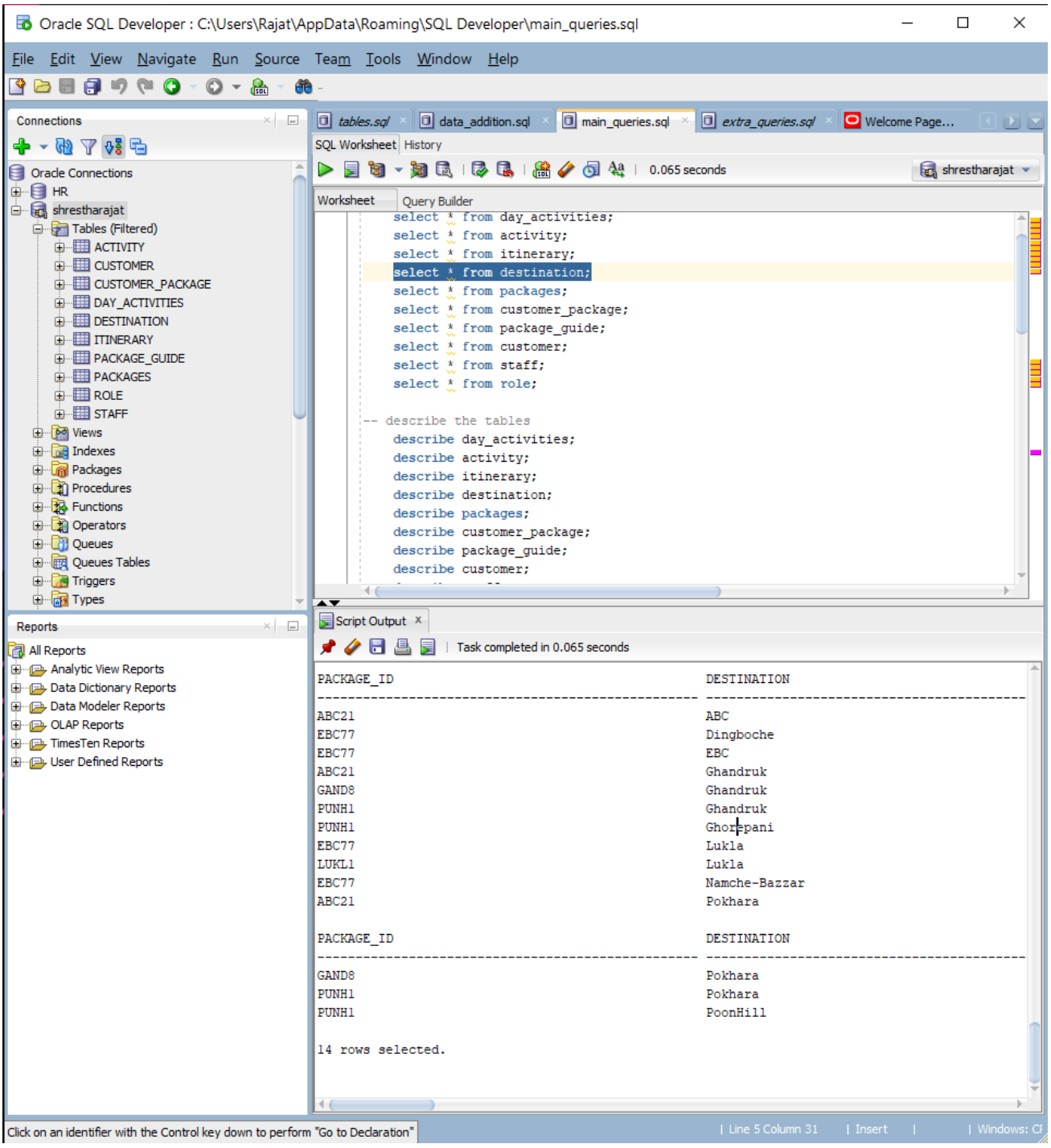

Figure 11: Running select Statement on Destination Table

5.3.5. Packages

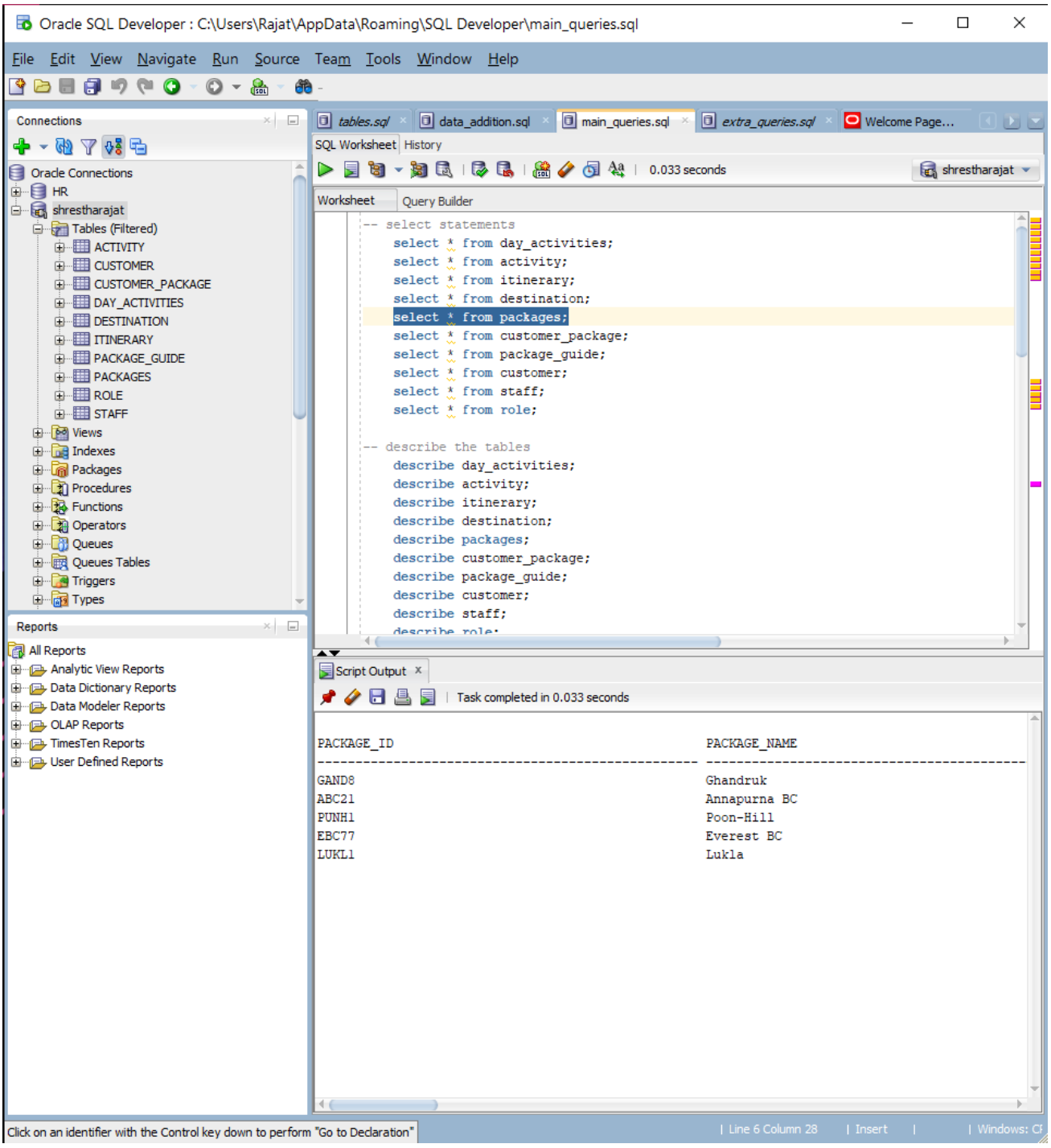

Figure 12: Running select Statement on Packages Table

5.3.6. Customer Packages

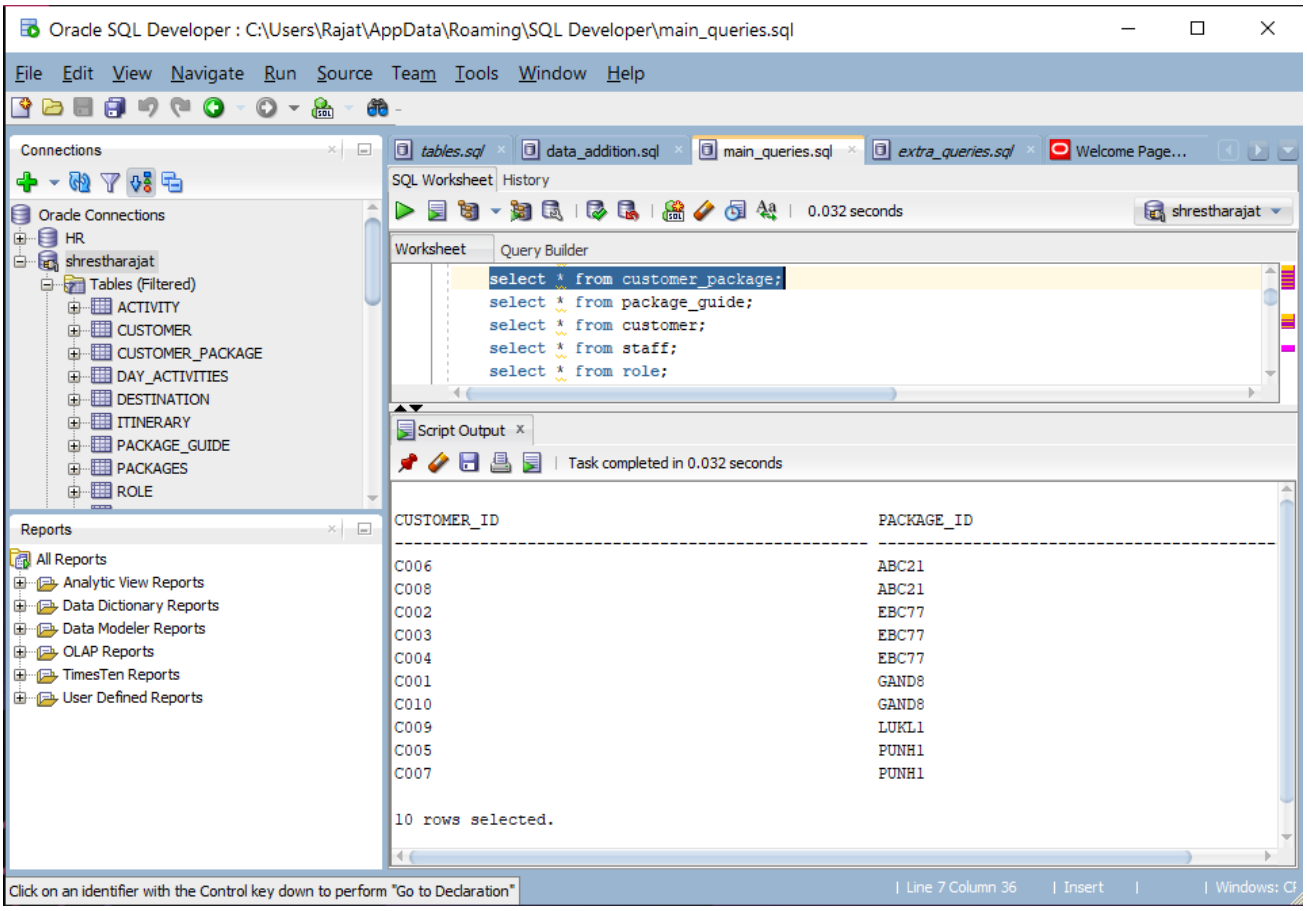

Figure 13: Running select Statement on Customer packages Table

5.3.7. Package Guide

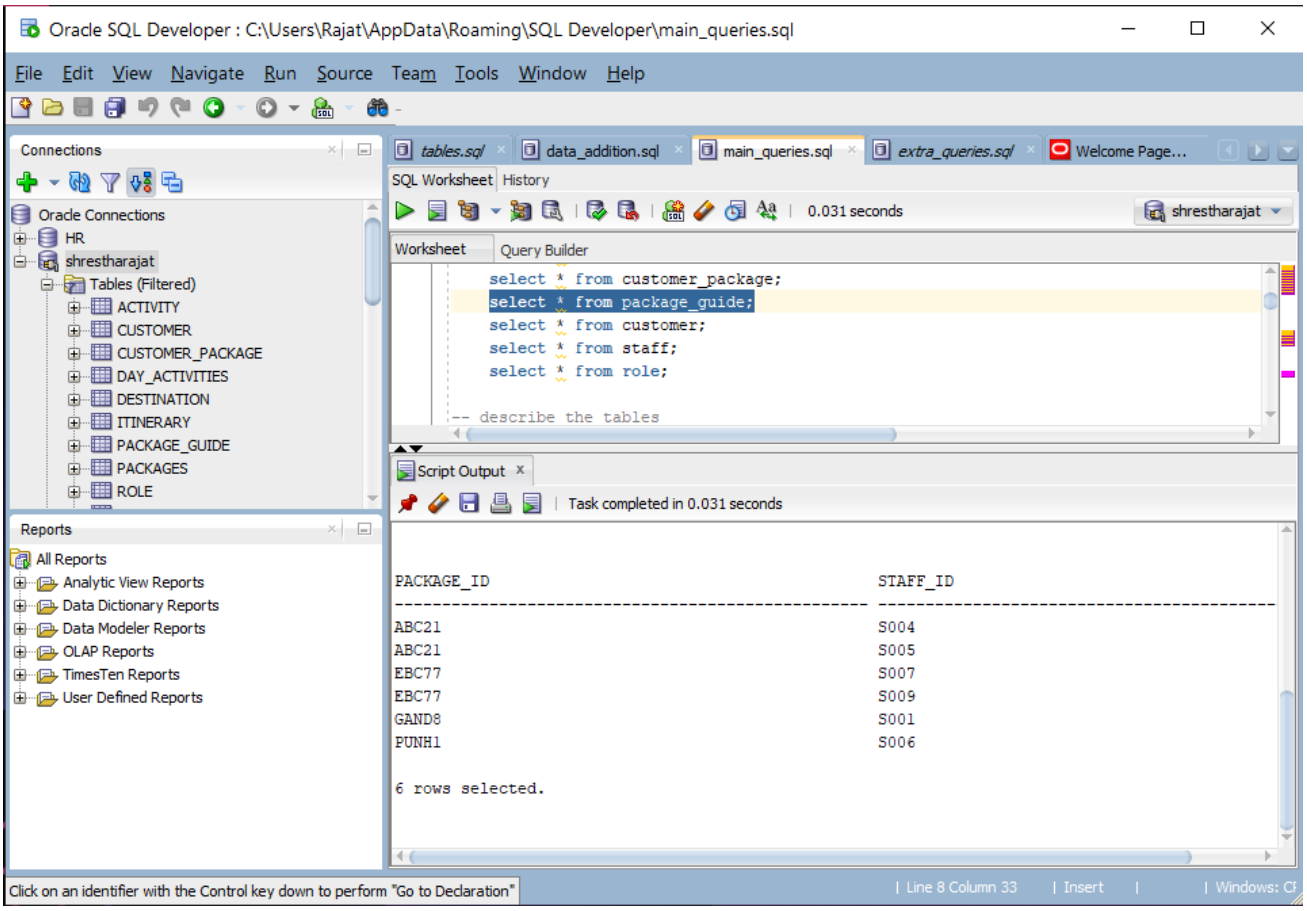

Figure 14: Running select Statement on Package Guide Table

5.3.8. Customer

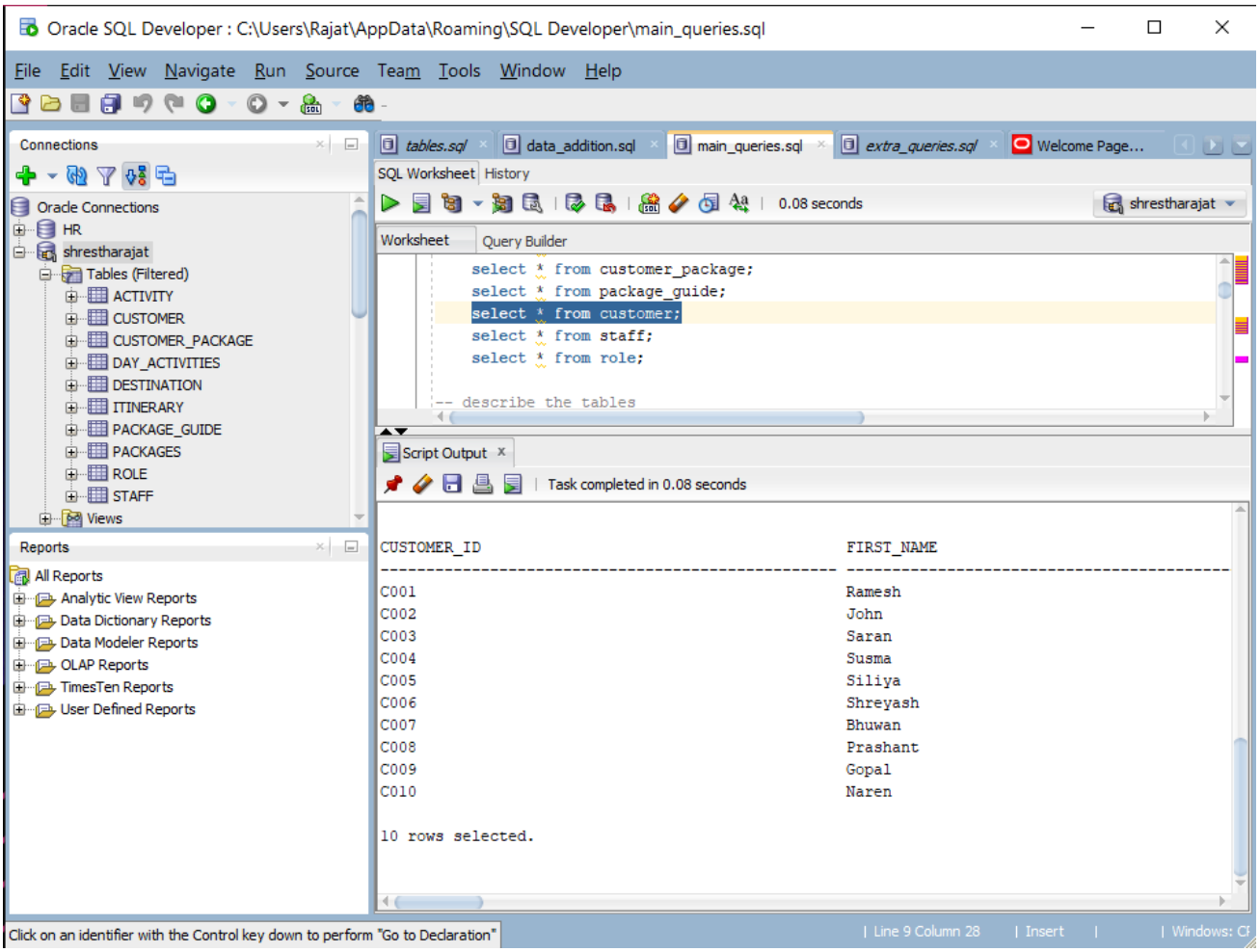

Figure 15: Running select Statement on Customer Table

5.3.9. Staff

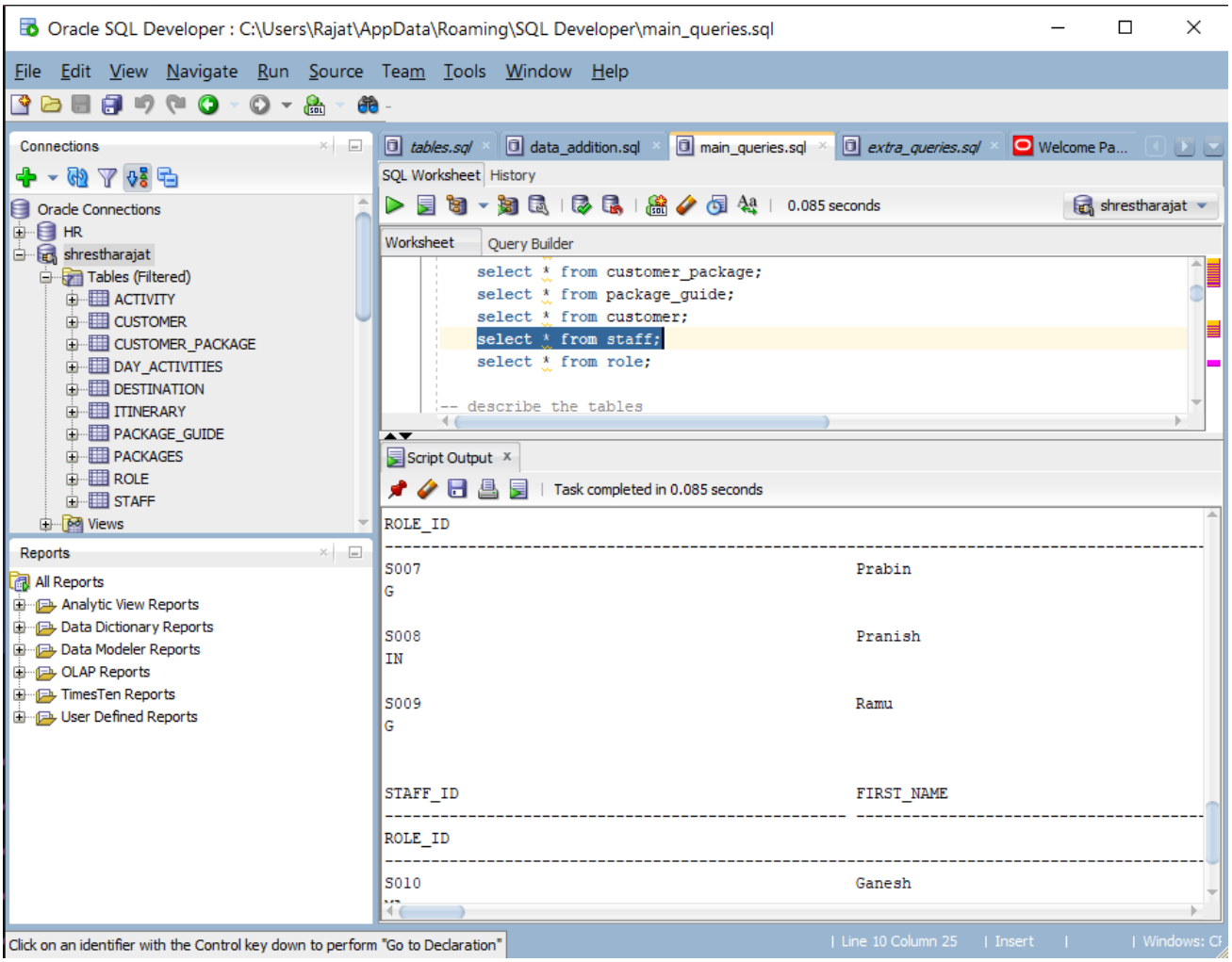

Figure 16: Running select Statement on Staff Table

5.3.10. Role

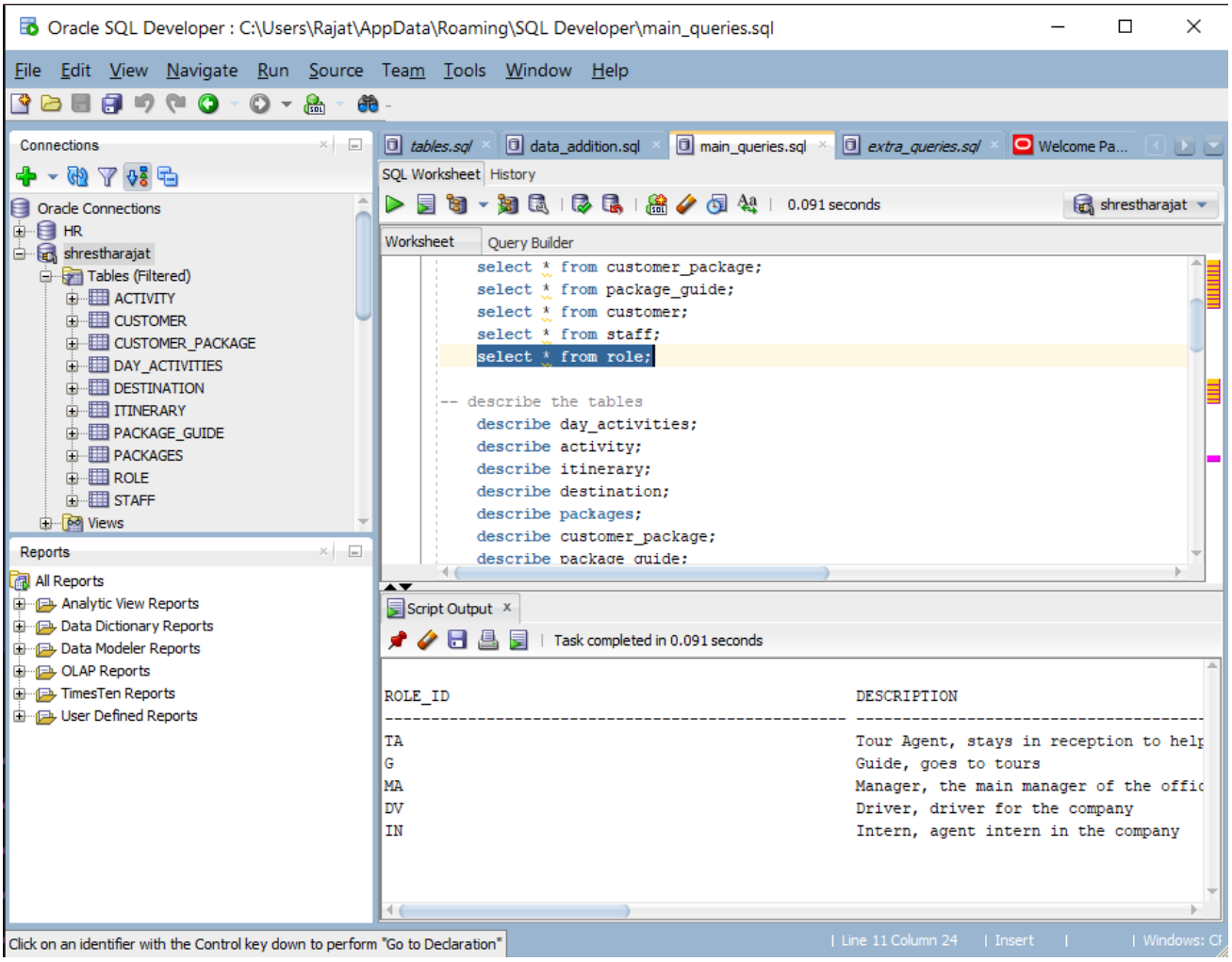

Figure 17: Running select Statement on Role Table

6. Implementation of Web-Based Database Application

6.1 Basic Webforms

These forms provide basic functionality for adding, updating or deleting any entries. The Add Delete and Edit Buttons on each Form Provide CRUD functionality for each basic form and the steps for each of the operation is detailed in the user manual.

6.1.1. Staff Details

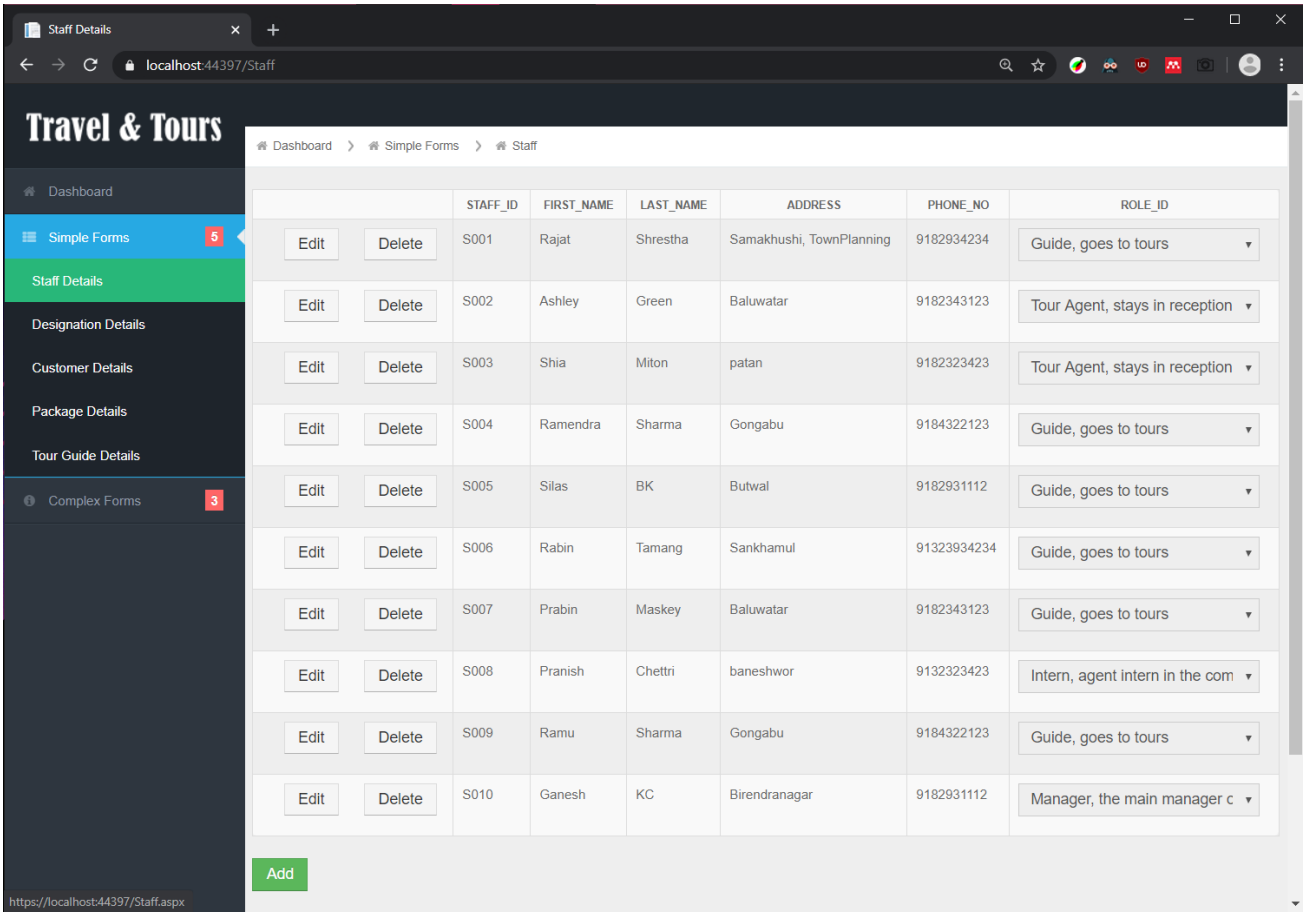

Figure 18: Simple Web-forms for Staff Details

6.1.2. Designation Details

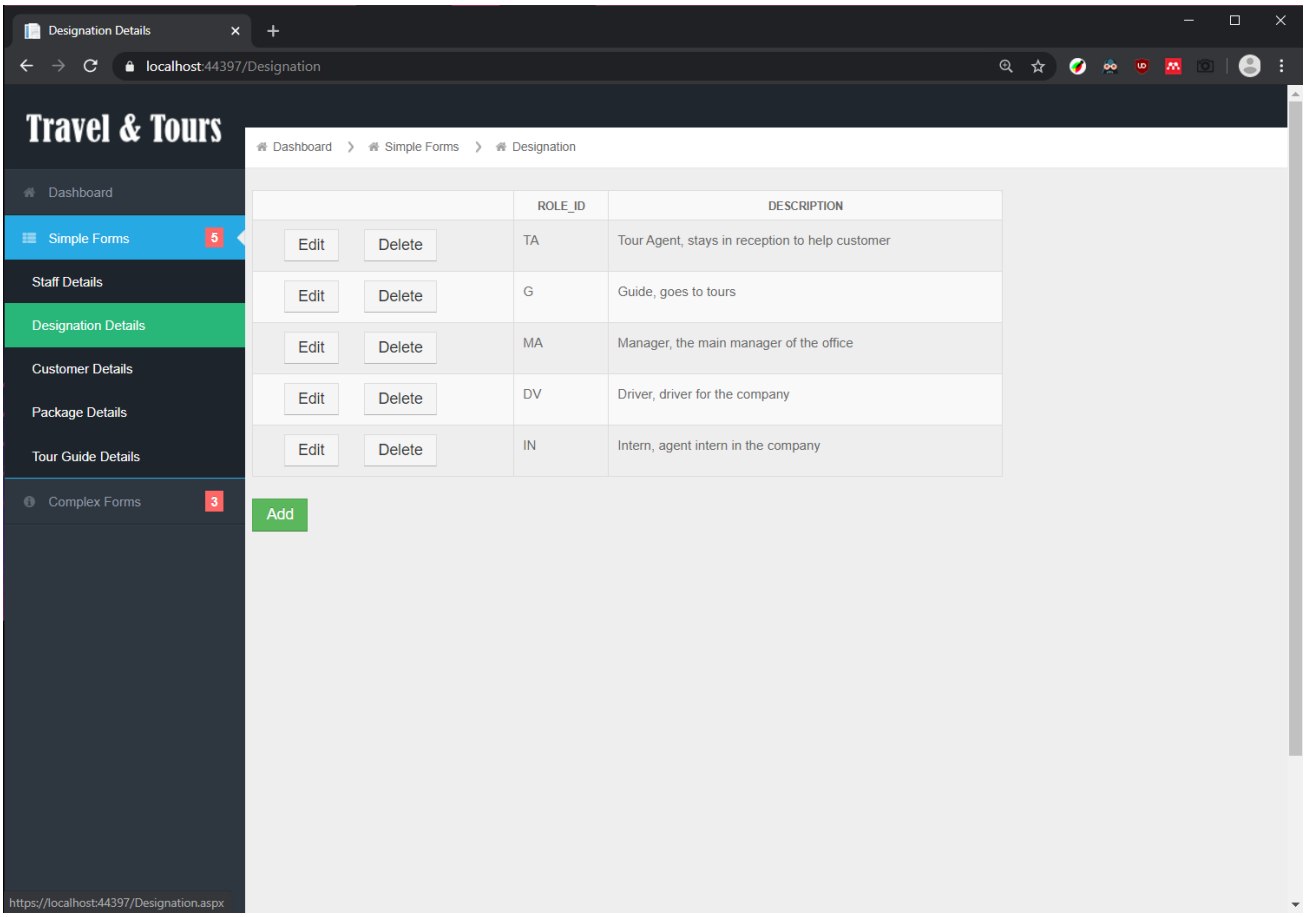

Figure 19: Simple Web-forms for Designation Details

6.1.3. Customer Details

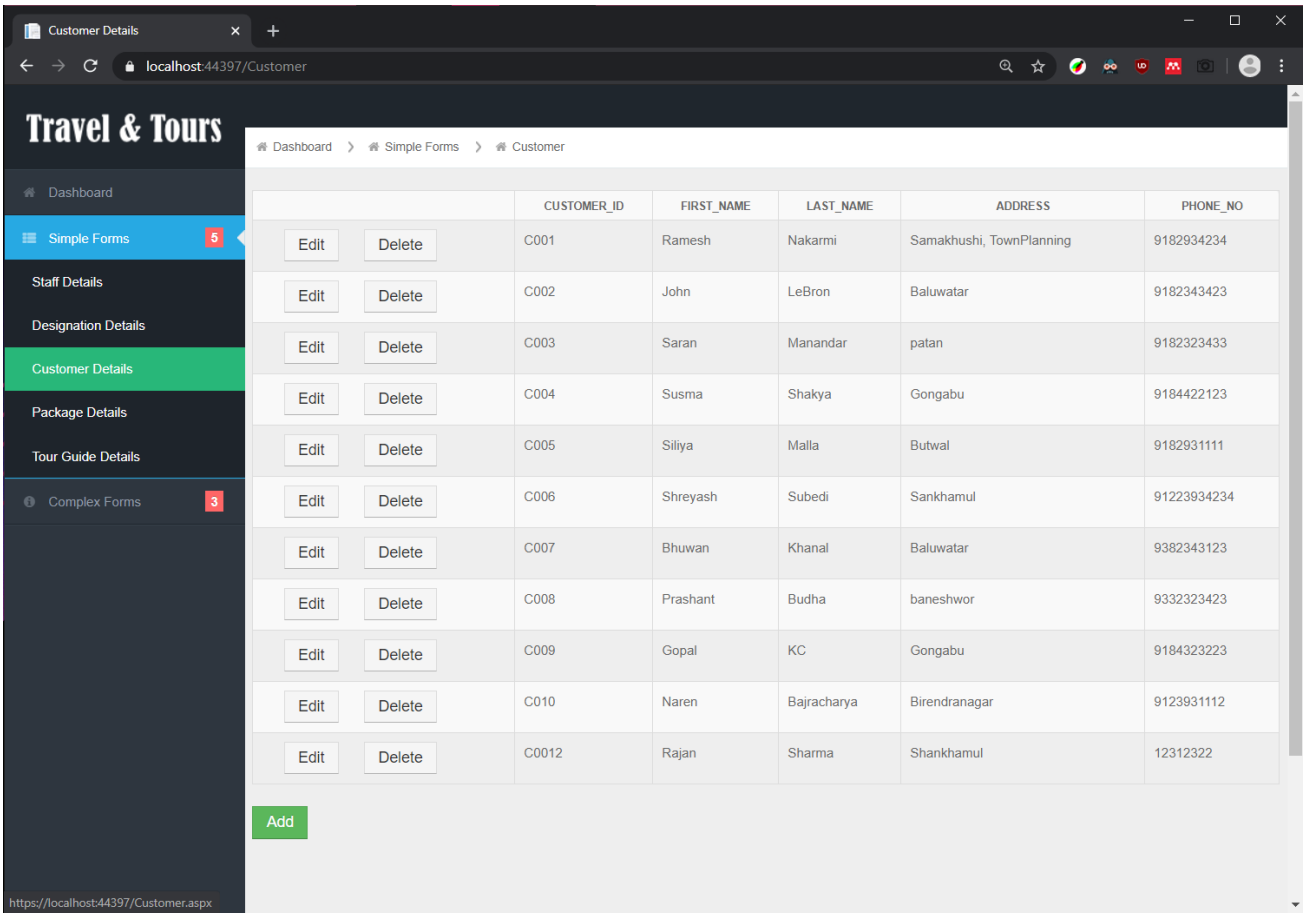

Figure 20: Simple Web-forms for Customer Details

6.1.4. Package Details

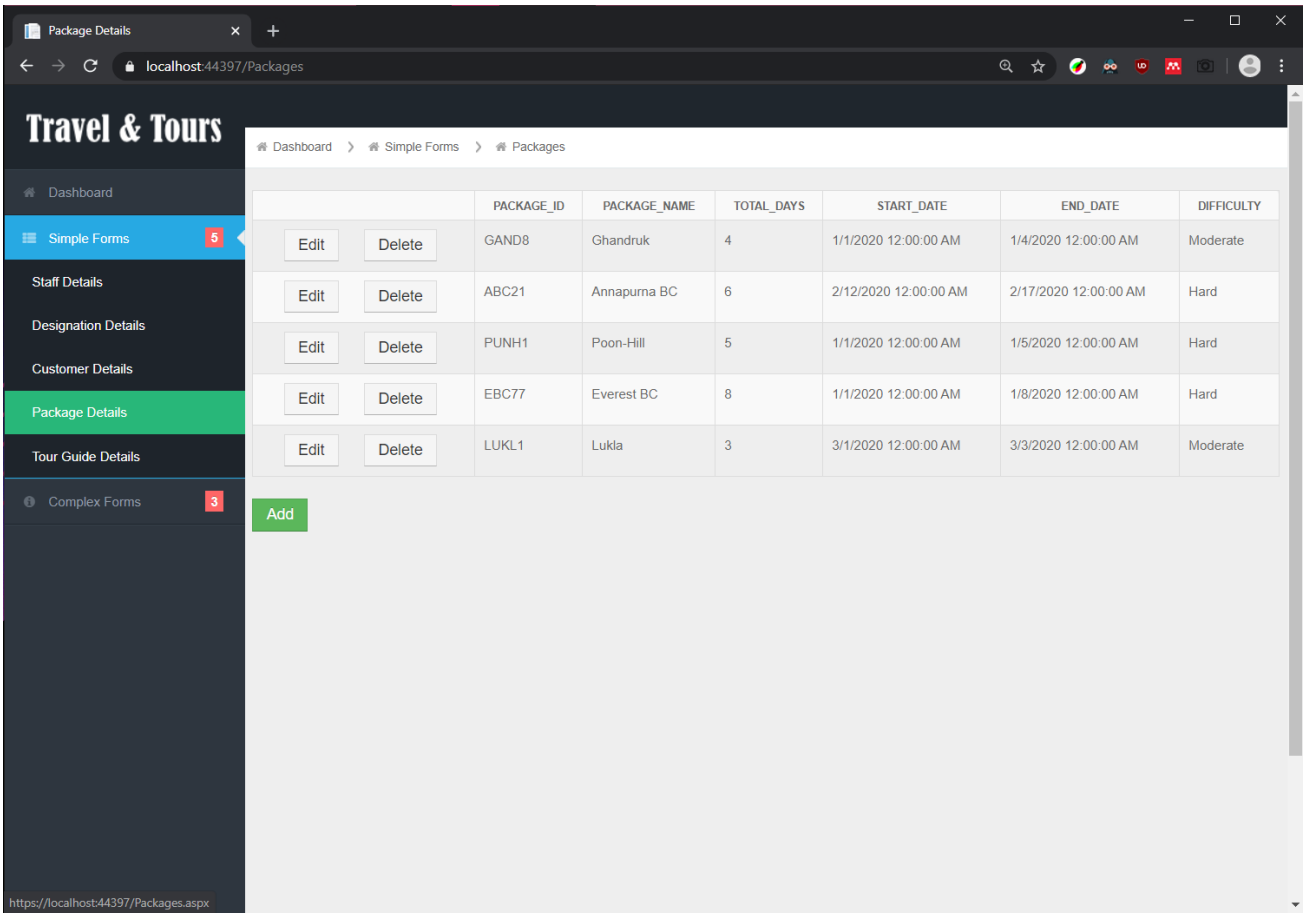

Figure 21: Simple Web-forms for Package Details

6.1.5. Tour Guide Details

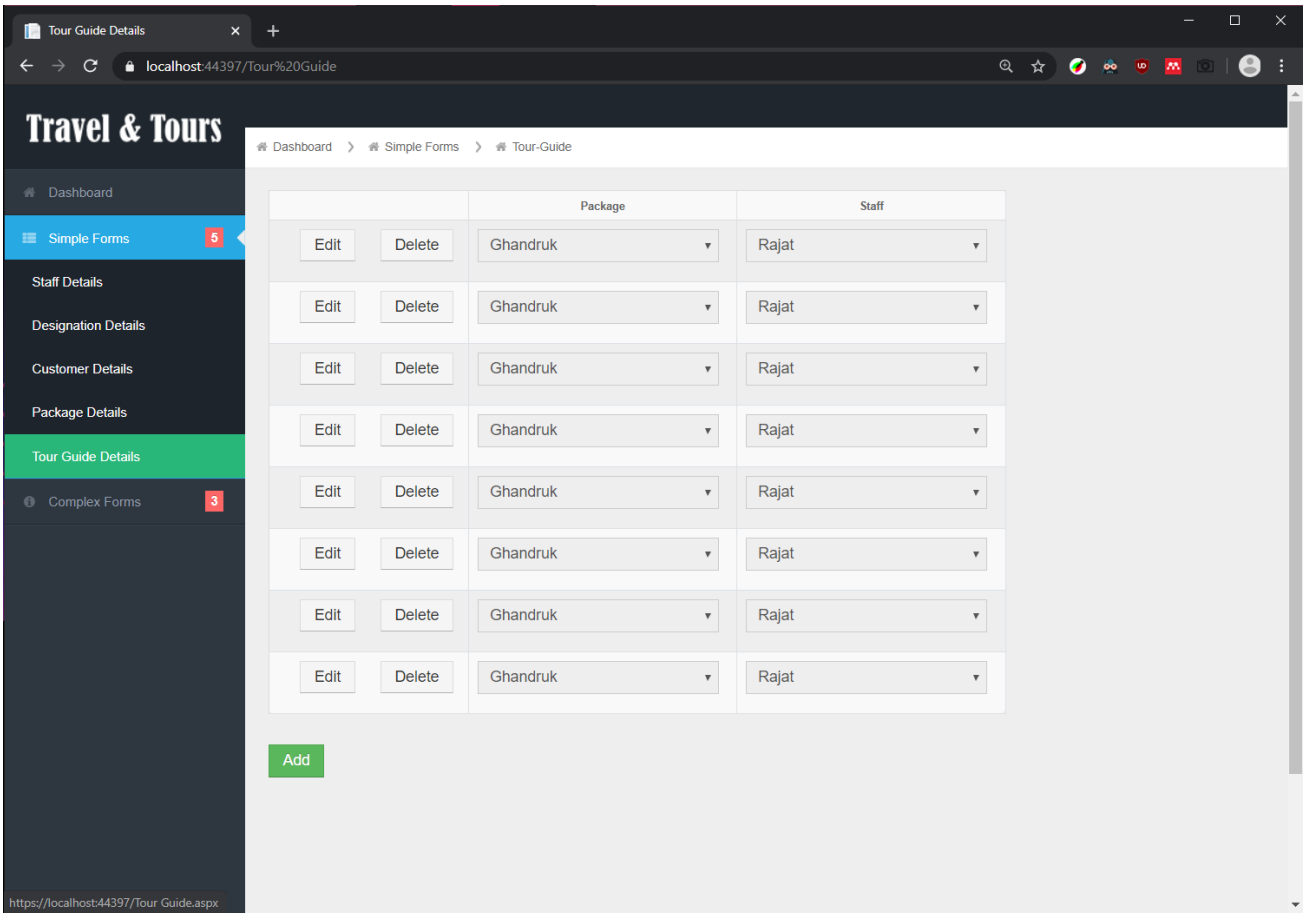

Figure 22: Simple Web-forms for Tour Guide Details

6.2. Complex Webforms

These web forms perform certain complex queries to extract information via joining various tables.

6.2.1. Customer-Package Schedule Form

The Customer-Package Schedule Form for any package shows the details of the package and the details of all customer who choose it.

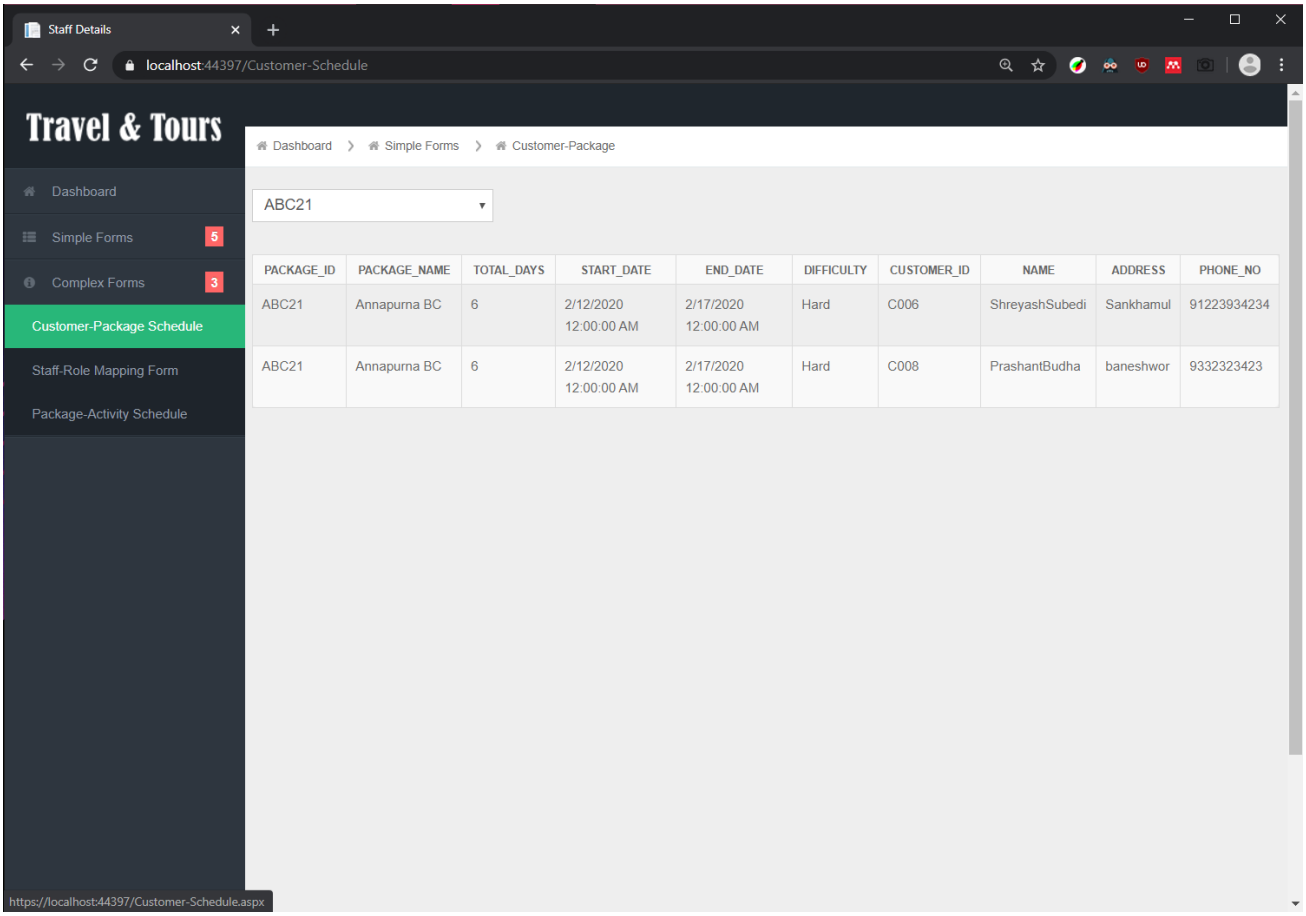

Figure 23: Complex Web-forms for Customer-Package Schedule Form

6.2.2. Staff-Role Mapping Form

Staff Role Mapping Form shows the count of staff of selected role in the company. This Form also shows additionally the details on staff members of the selected role types.

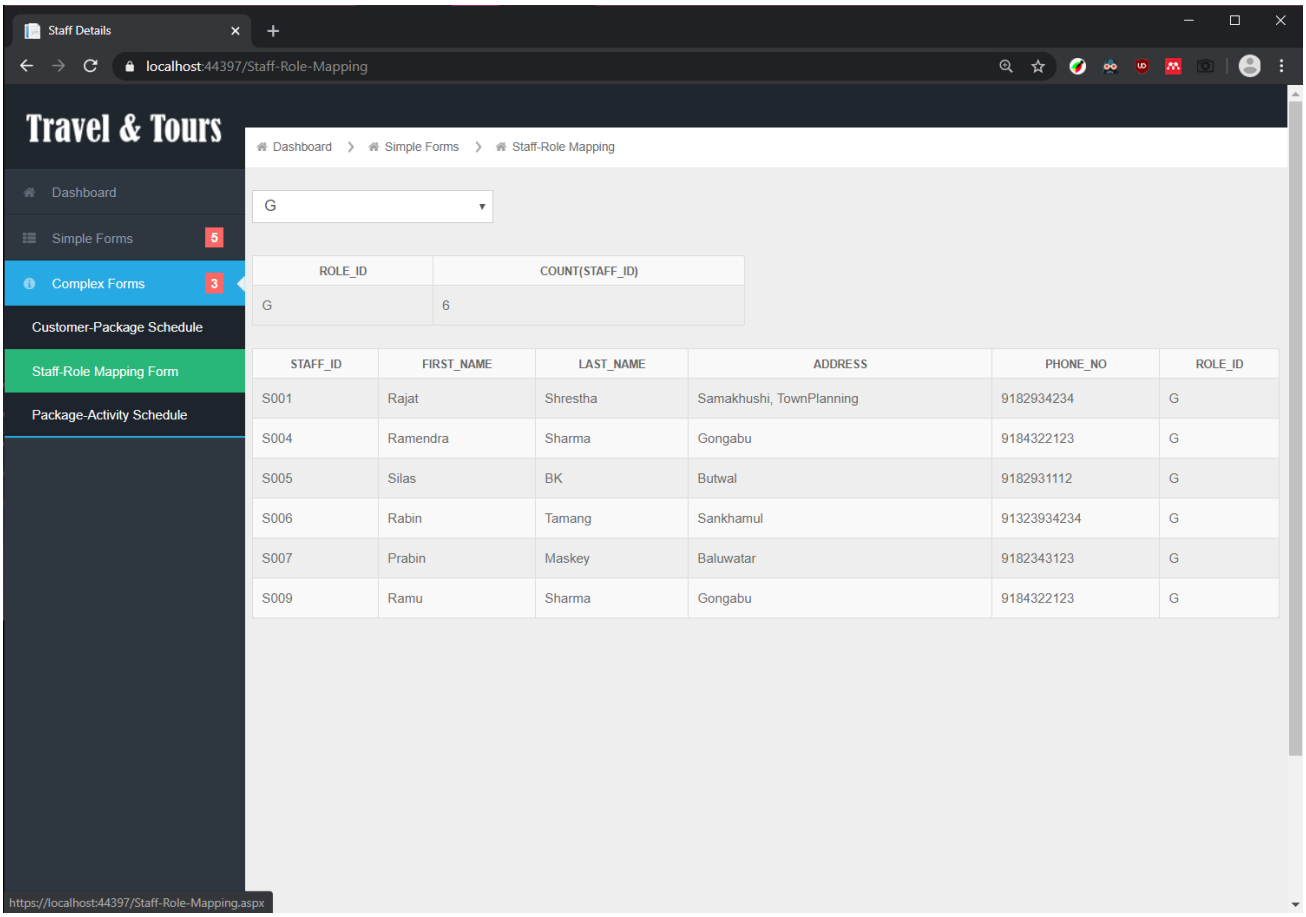

Figure 24: Complex Web-forms for Staff-Role Mapping Form

6.2.3. Package-Activity Schedule Form

Package-Activity Schedule Form shows the details of the activities, travel details, mode for any selected package

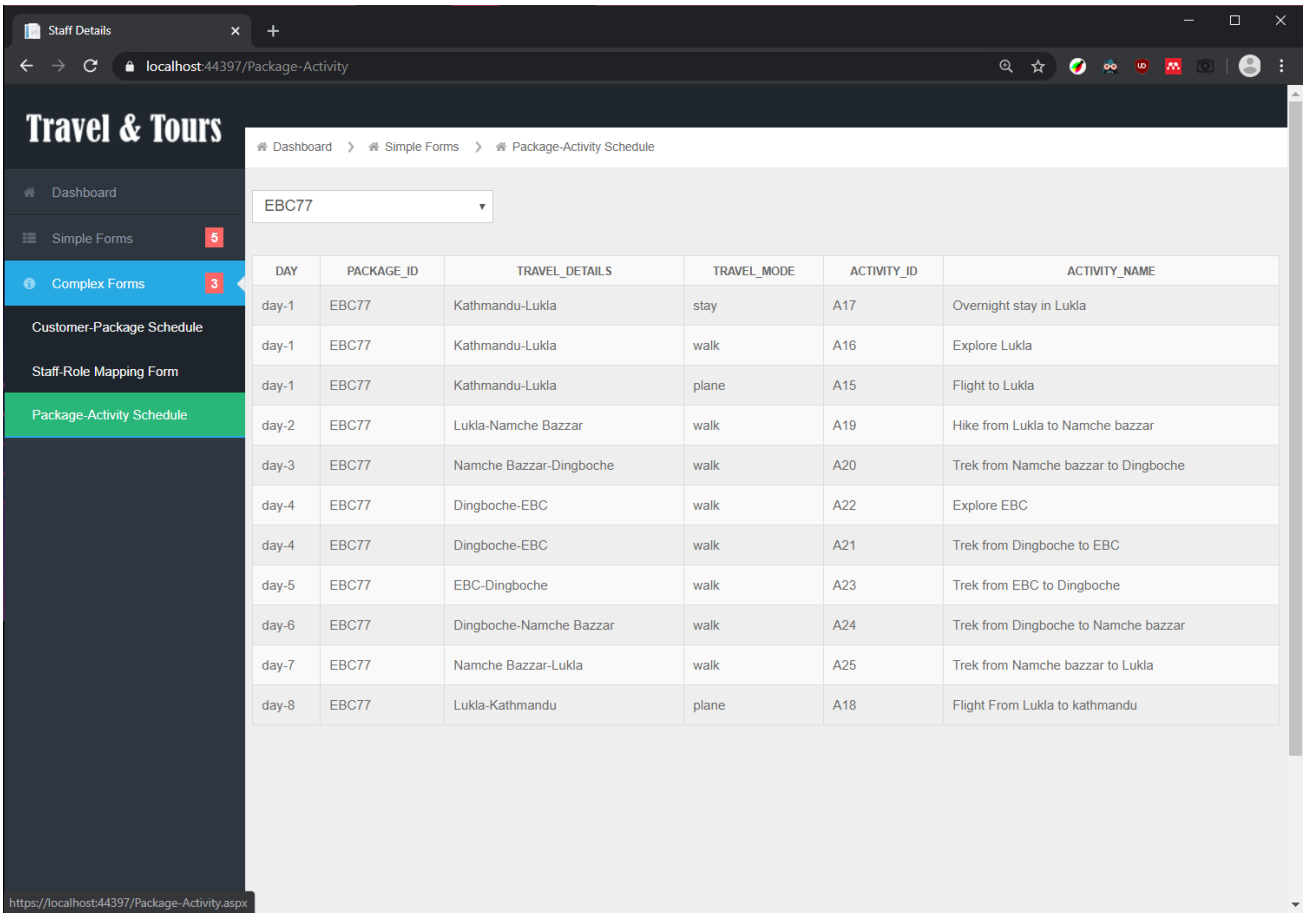

Figure 25: Complex Web-forms for Package Activity Schedule Form

6.3. Dashboard

The simple dashboard acts as the home screen of the application and shows various insights on the contents of the database. This includes links to the various simple and complex web forms. The charts show the insights on the staff count by designation and the customer count on each package. This page also contains the indicator for data inserted in each table.

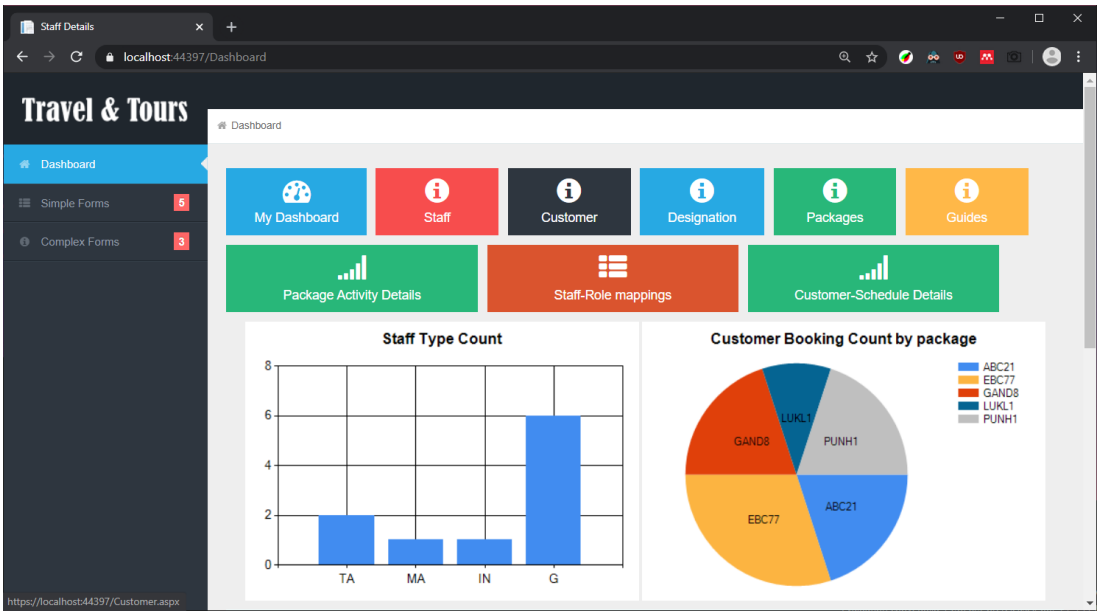

Figure 26: The Dashboard with Links to webforms and Charts

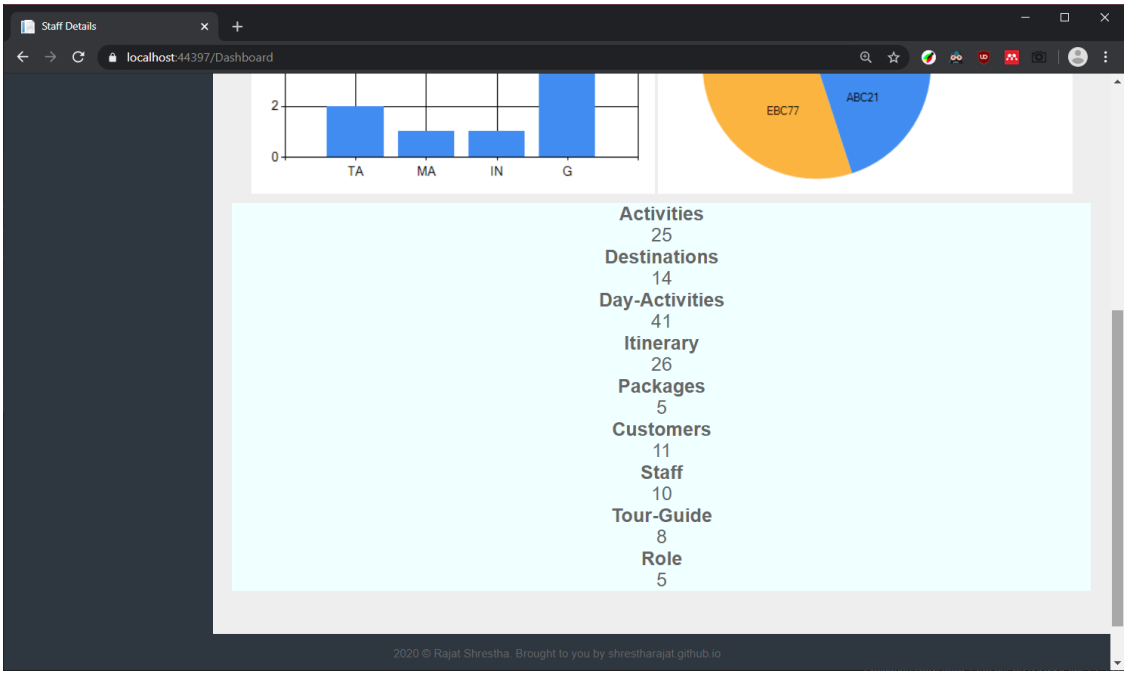

Figure 27: Dashboard with details on the Number of data entered in the database

7. Testing:

7.1. Simple Forms:

Since the Crud operations for the Simple forms are the same each form is selected to carry out unique tasks in the tests.

7.1.1. Adding Data

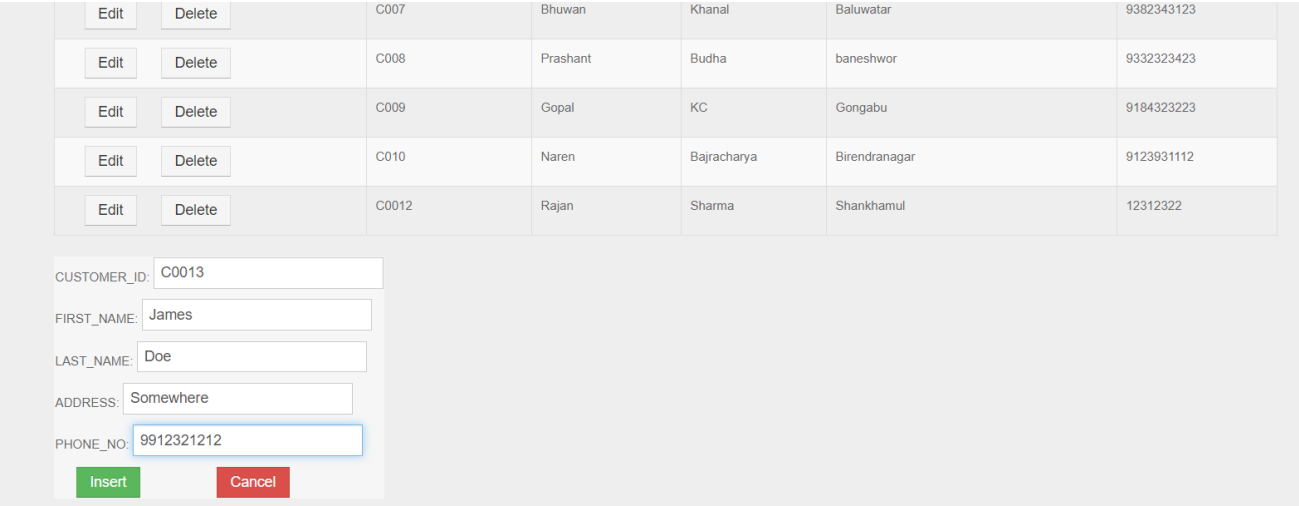

Figure 28: Entering Data in the Customer Form

Figure 29: New Customer inserted in the customer table

Test Summary	Adding data in Customer form
Action	Adding customer details
Expected Output	Details to be entered in table and shows up in form
Actual Output	Details are entered in table and are shown in the form
Result	Success

7.1.2. Editing Data

Figure 30: Staff data before editing

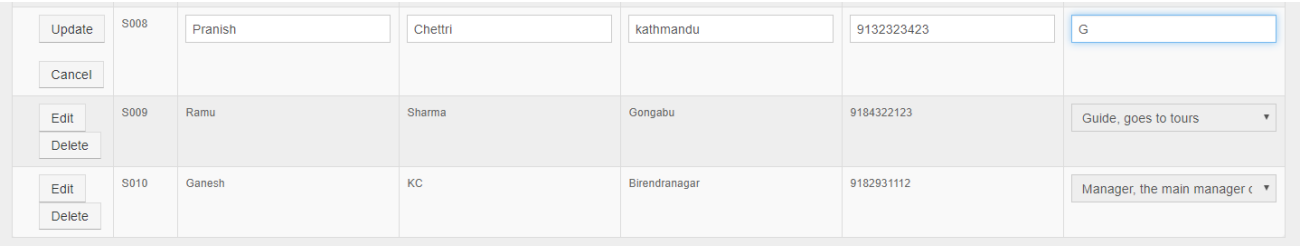

Figure 31: Editing Staff Data

Figure 32: Staff data after editing

Test Summary	Edit Staff Details
Action	Edit a staff detail in staff form
Expected Output	Details of a staff to be updated and Primary key should not be updated
Actual Output	Details of the staff is updated, and the PK was made inactive so that it could not be edited
Result	Success

7.1.3. Deleting Data

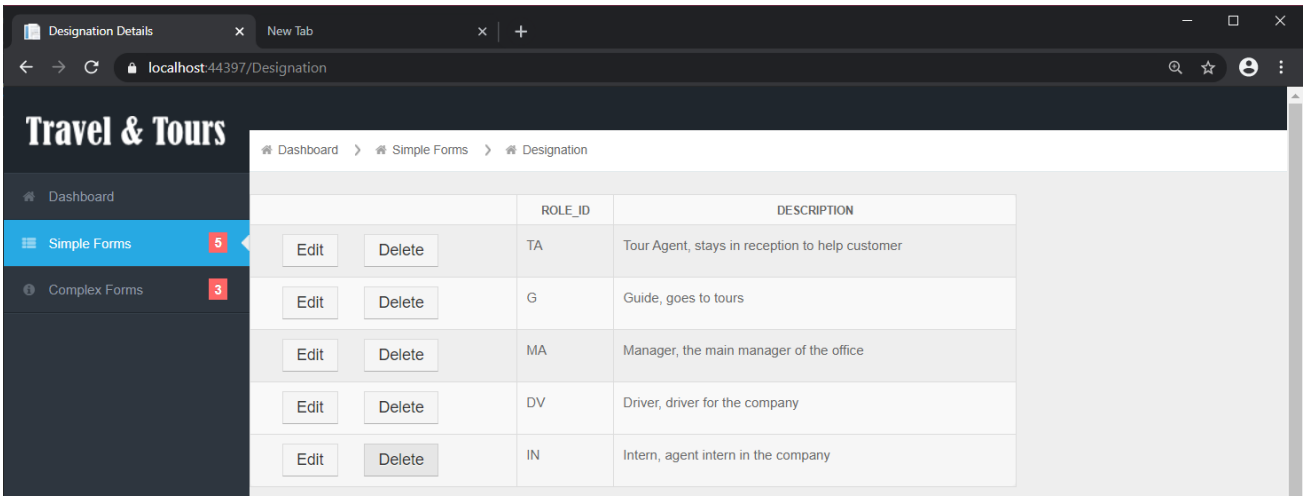

Figure 33: Designations before deleting

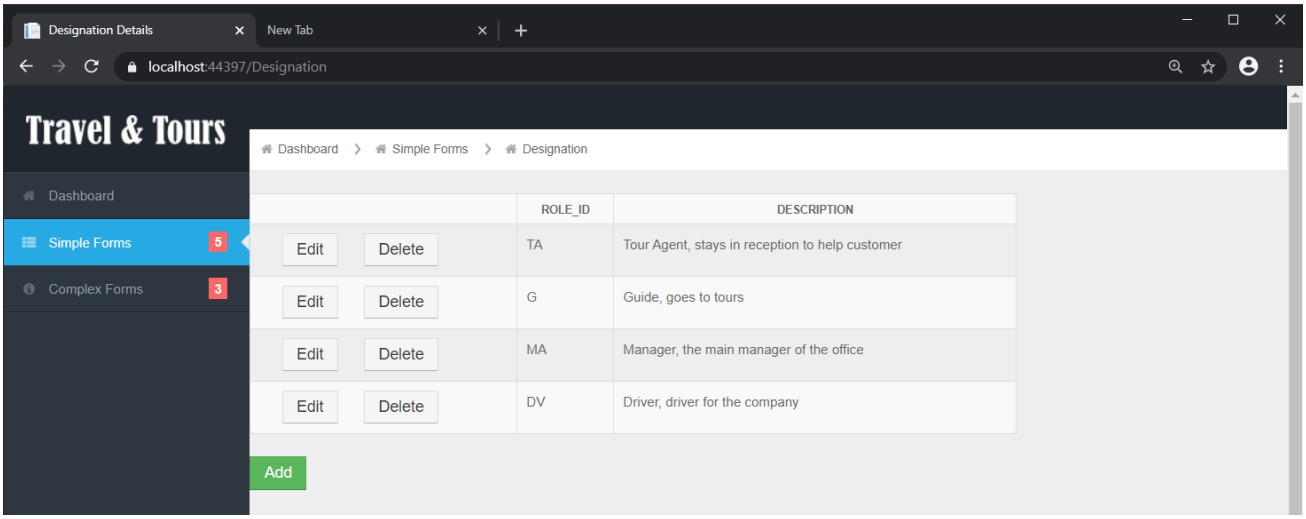

Figure 34: Intern designation removed after Deleting

Test Summary	Deleting an entry in Designation form
Action	Delete Intern from Designation as there are no interns
Expected Output	Intern to be removed
Actual Output	The Intern Entry was removed
Result	Success

7.1.4. Adding Duplicate Data

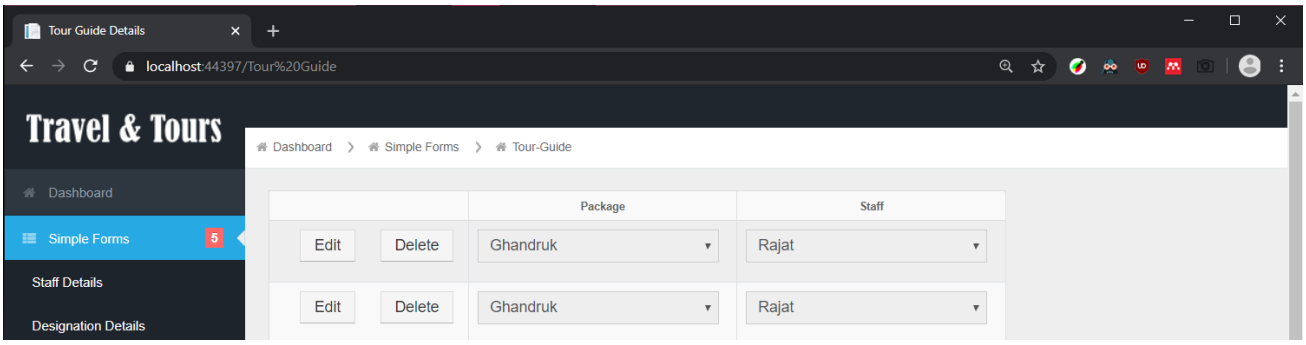

Figure 35: Editing the Tour-guide to be duplicate

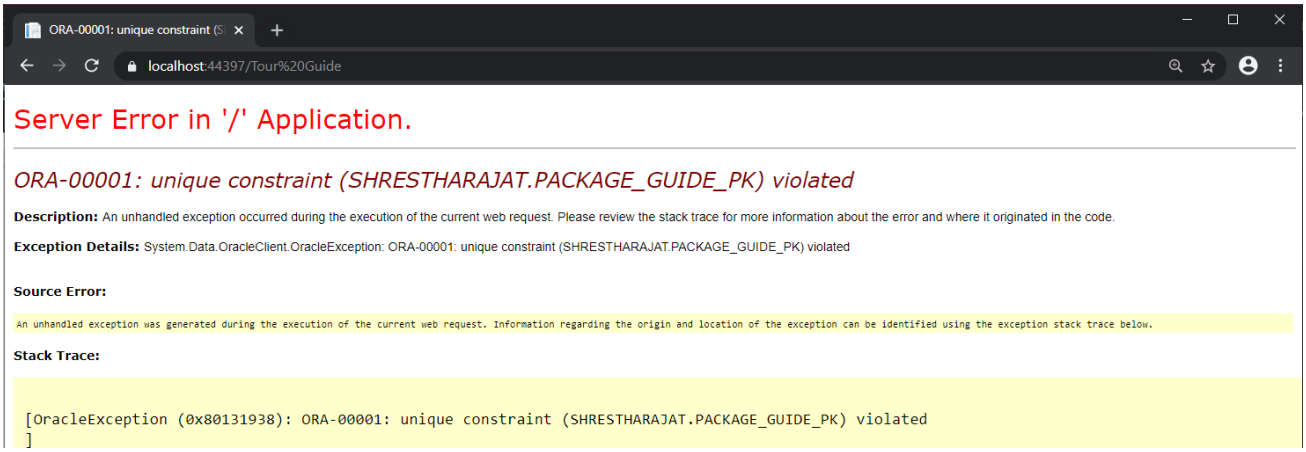

Figure 36: Exception occurred due to duplicate data

Test Summary	Try to add Duplicate data in Simple Form
Action	Add duplicate data in tour guide form
Expected Output	Error occurs and halts the operation
Actual Output	An Exception occurred not allowing data to be entered
Result	Success

7.1.5. Deleting referenced entry

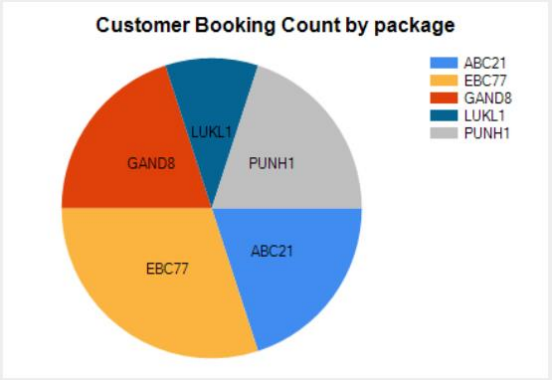

Figure 37: All of the Packages are currently referenced

	PACKAGE_ID	PACKAGE_NAME	TOTAL_DAYS	START_DATE	END_DATE	DIFFICULTY
Edit Delete	GAND8	Ghandruk	4	1/1/2020 12:00:00 AM	1/4/2020 12:00:00 AM	Moderate
Edit Delete	ABC21	Annapurna BC	6	2/12/2020 12:00:00 AM	2/17/2020 12:00:00 AM	Hard
Edit Delete	PUNH1	Poon-Hill	5	1/1/2020 12:00:00 AM	1/5/2020 12:00:00 AM	Hard
Edit Delete	EBC77	Everest BC	8	1/1/2020 12:00:00 AM	1/8/2020 12:00:00 AM	Hard
Edit Delete	LUKL1	Lukla	3	3/1/2020 12:00:00 AM	3/3/2020 12:00:00 AM	Moderate

Figure 38: Deleting one referenced data (package)

Server Error in '/' Application.

ORA-02292: integrity constraint (SHRESTHARAJAT.CUSTOMER_PACKAGE_PACKAGES_FK) violated - child record found

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.OracleClient.OracleException: ORA-02292: integrity constraint (SHRESTHARAJAT.CUSTOMER_PACKAGE_PACKAGES_FK) violated - child record found

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.

Stack Trace:

```
[OracleException (0x80131938): ORA-02292: integrity constraint (SHRESTHARAJAT.CUSTOMER_PACKAGE_PACKAGES_FK) violated - child record found
]
```

Figure 39: Exception occurred while deleting referenced item

Test Summary	Try to delete referenced data in Simple Form
Action	Delete any package from package form
Expected Output	Error occurs and halts the operation
Actual Output	An Exception occurred not allowing data to be deleted
Result	Success

7.2. Complex Forms:

7.2.1. Filtering the Package-Customer Form

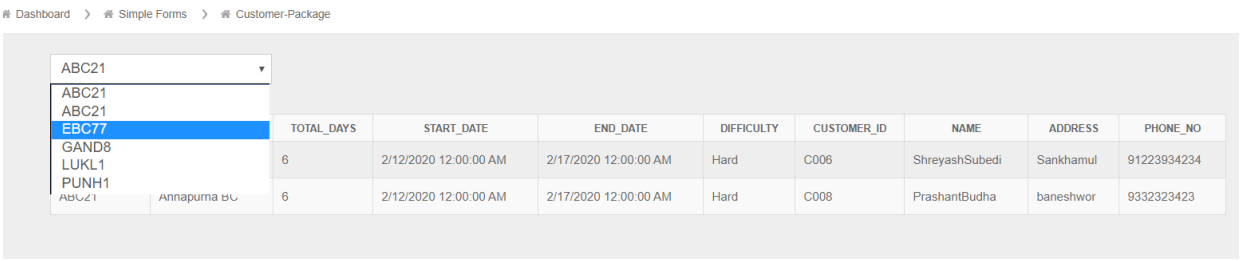

Figure 40: Selecting Package Id to show customer booking the selected package

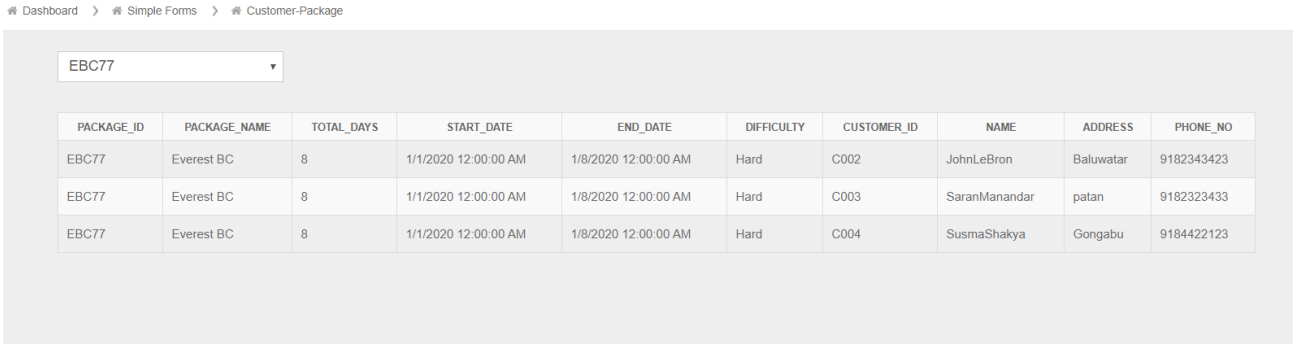

Figure 41: Table showing customers who is booking the selected Package

Test Summary	Check the Package-Customer Filtering
Action	Select any Package
Expected Output	The table will show customer data for selected package
Actual Output	The table showed customer data for selected package
Result	Success

7.2.2. Filtering Staff-Role mapping Form

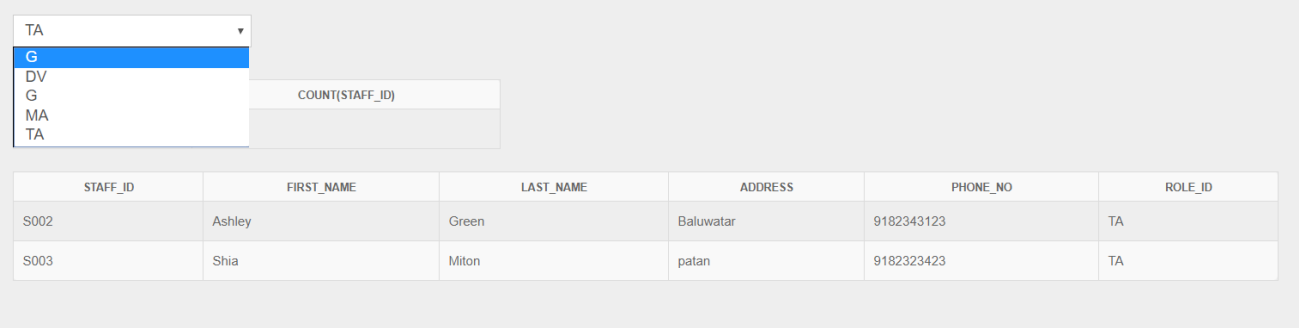

Figure 42: Selecting the Designation to count the no of staffs in it and staff details

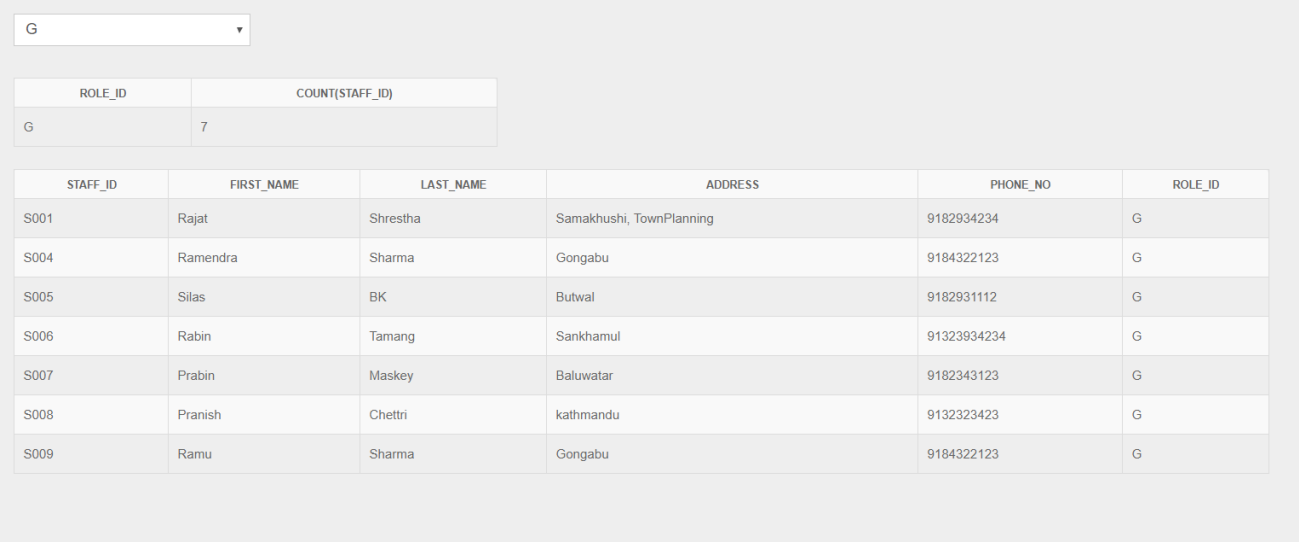

Figure 43: Staff details and count of the selected designation

Test Summary	Check the Staff Role mapping Filtering
Action	Select any Role to Filter
Expected Output	The table will show staff data for selected role
Actual Output	The table showed staff data for selected role
Result	Success

7.2.3. Filtering Package-Activity Schedule Form

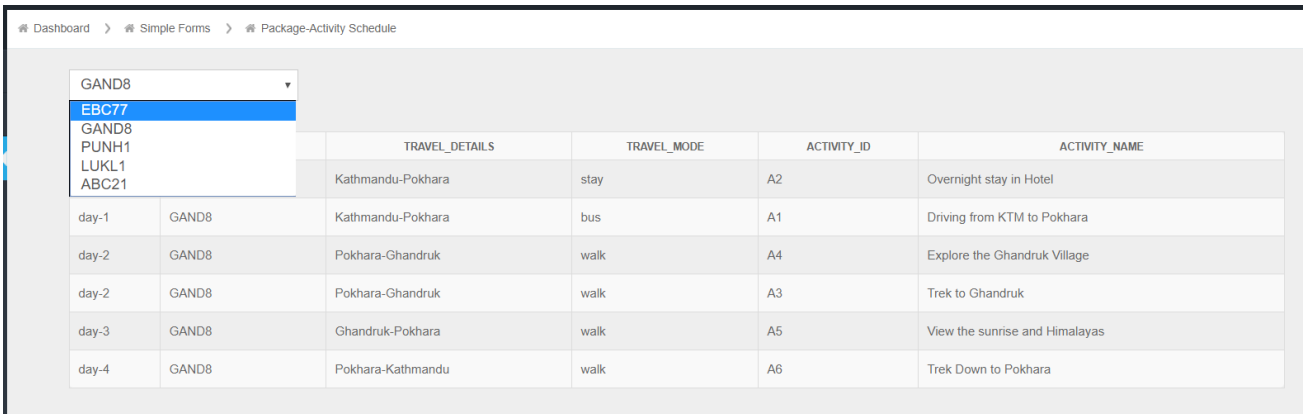

Figure 44: Selecting Package Id to view the enlisted activities

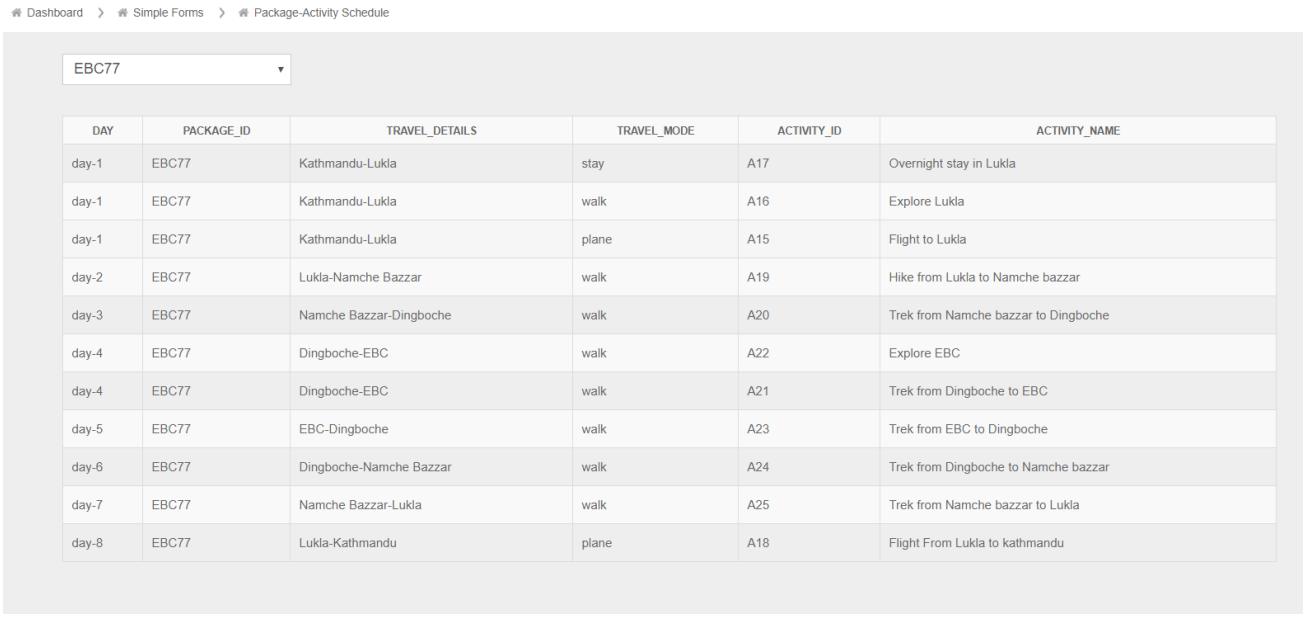

Figure 45: Enlisted Activities of the selected package

Test Summary	Check the package Activity filtering
Action	Select any Package to view Activities
Expected Output	The table will Activities of the selected package
Actual Output	The table showed Activities of the selected package
Result	Success

7.3. Dashboard

7.3.1. Testing All of the Links

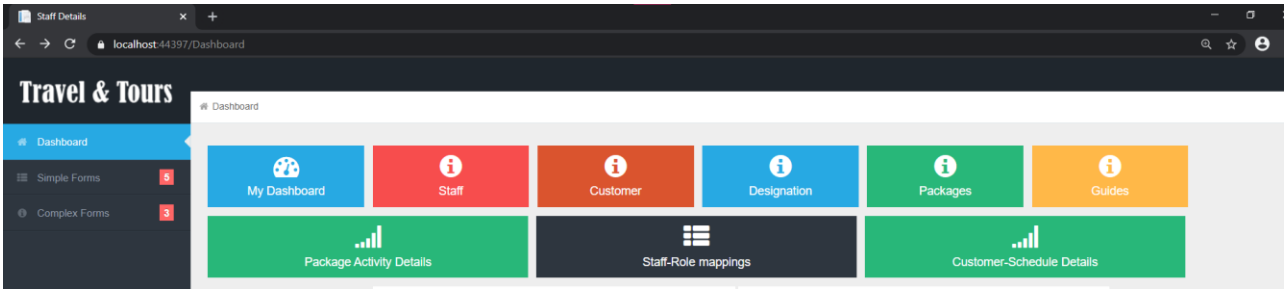

Figure 46: Clicking the staff role link

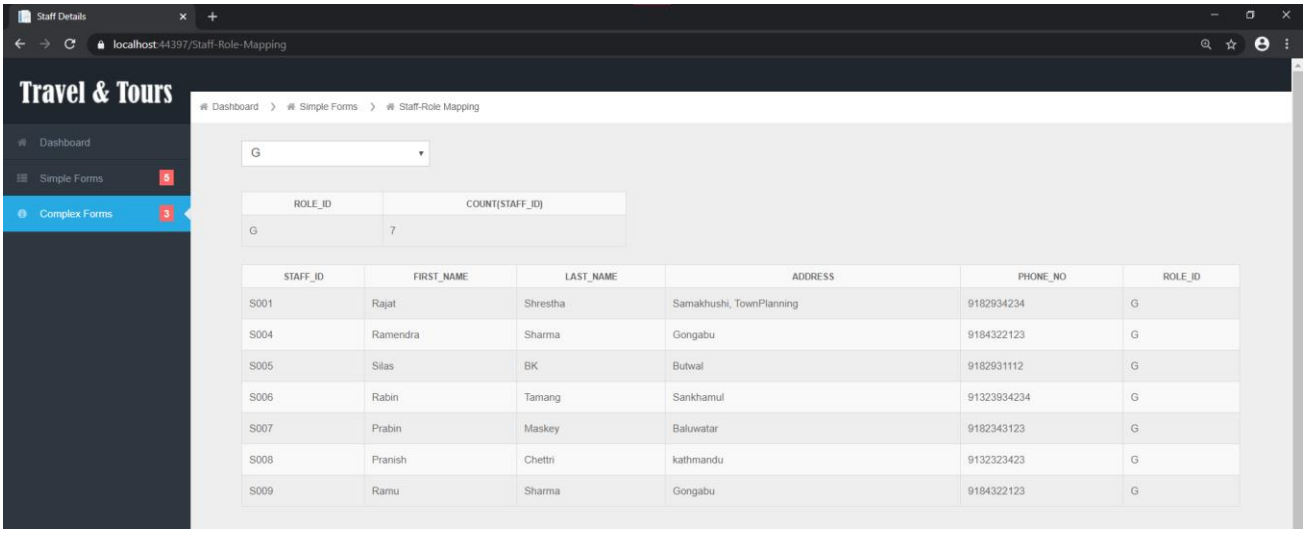

Figure 47: Getting redirected to the staff-role page

Test Summary	Check the Dashboard links
Action	Select Staff-Role mapping link in the dashboard
Expected Output	The page will be redirected to the staff-role form page
Actual Output	The page was redirected to the staff-role form page
Result	Success

7.3.2. Testing the charts in the Dashboard

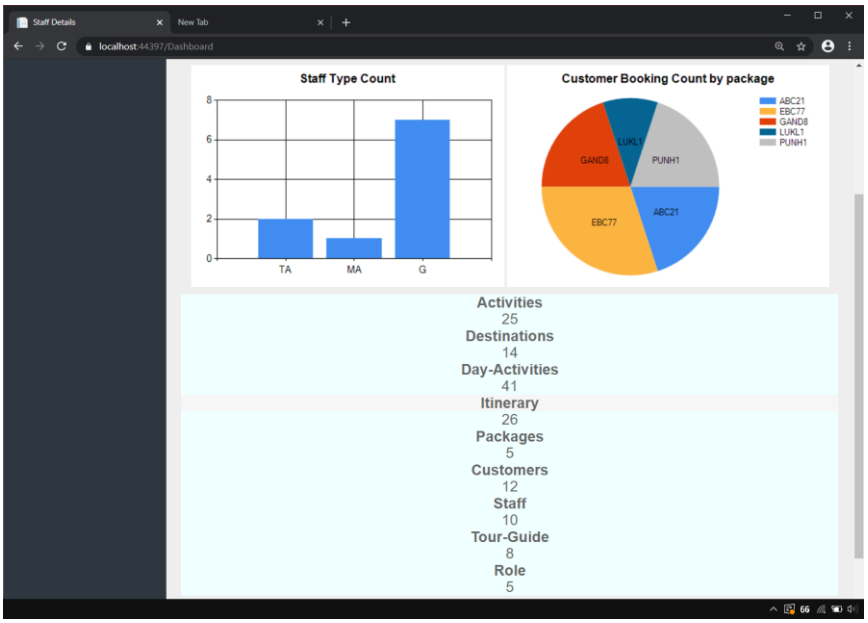

Figure 48: Charts and data before updating data in the database

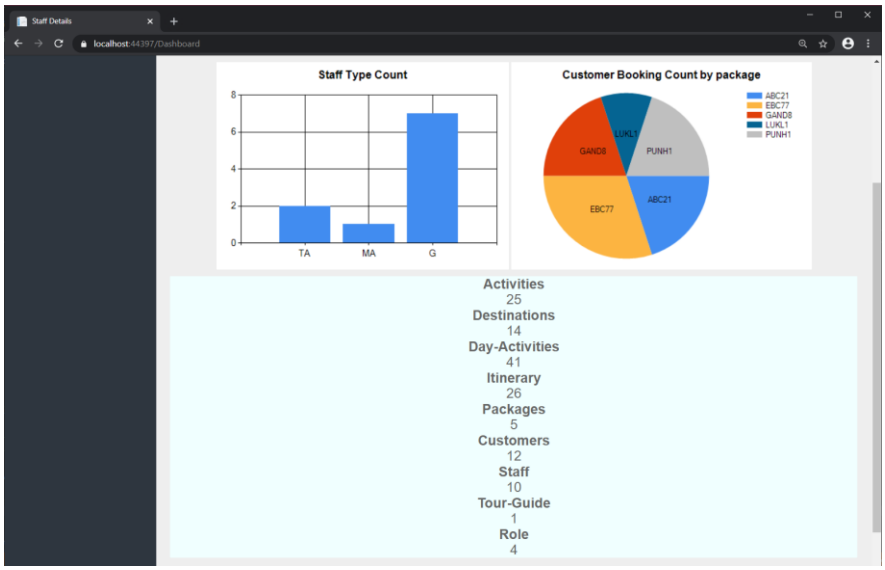

Figure 49: Charts and data after updating the database

Test Summary	Check the Dashboard charts
Action	Modify Data in the Database
Expected Output	The Graphs will readjust according to the data in the database.
Actual Output	The Graphs was readjusted according to the data in the database.
Result	Success

8. User Manual

9. Further Discussion

This coursework was quite fun to do and had a lot of new challenges which required various tricks and tips to continue this coursework has improved the database management and problem skills due the to the complex scenarios involving various entities and relations. Mainly this coursework focuses on using the Given tools for solving the problem:

1. Oracle database
2. Visual Studio
3. Oracle SQL Developer
4. Oracle SQL Developer Datamodeler
5. Visual Studio Code

The Following tools were essential in their own way for hosting a local database server, an Integrated Development Environment to work with ASP .NET framework, an SQL Query Developing platform, ER-Model generator and a simple text editor. After doing the required tasks using core technologies the project was further enhanced by various other technologies which improves various performance and development of the application such. The core and optional technologies used are:

1. HTML
2. CSS
3. Java Script
4. C#
5. Bootstrap
6. jQuery
7. ASP .NET
8. SQL
9. PL/SQL

tools and technologies required for the development of the given project greatly helped on understanding how different technologies work together to create a functional project and exposed various new tricks on how a developer can save time by using such technologies. This project also strongly demonstrated how coding has become easier due to various proprietary software designed to automate and make complex problems simple in the modern day.

References

Coronel, C. & Morris, S., 2018. *Database Systems: Design, Implementation, & Management.* s.l.:Cengage Learning.

Kreines, D., 2009. *Oracle Data Dictionary Pocket Reference.* 1st ed. Chicago: O'Reilly Media.

Nishadha, 2017. *Ultimate Guide to ER Diagrams (Entity Relationship Diagrams).* [Online]
Available at: https://creately.com/blog/diagrams/er-diagrams-tutorial/
[Accessed 28 January 2020].

Oracle, 2020. *Data Modeling with Oracle SQL Developer.* [Online]
Available at: https://www.oracle.com/in/database/technologies/appdev/datamodeler.html
[Accessed 28 January 2020].

Scottish Qualifications Authority, 2010. *Outcome 1:Fundamentals of Database Design.* [Online]
Available at: https://www.sqa.org.uk/e-learning/MDBS01CD/page_06.htm
[Accessed 12 1 2019].

Watt, A., 2012. *Database Design.* 2nd ed. Vancouver: BCcampus Open Textbook.

Appendix