# 

**Advanced Databases Assignment**

**Easy Travels Database**

**Module Title: Advanced Databases (B9IS100)**

**Module Leader: Dr. Shazia A Afzal**

**Group Members: Manik Mahashabde (10518579)**

**Saurabh Devade (10531140)**

**Sabitha Maram (10533048)**

# TABLE OF CONTENTS

[TABLE OF CONTENTS 1](#_Toc27512266)

[TABLE OF FIGURES 2](#_Toc27512267)

[1. BUSINESS CASE 4](#_Toc27512268)

[2. SCOPE OF THE PROJECT 4](#_Toc27512269)

[2.1 BUSINESS REQUIREMENTS 4](#_Toc27512270)

[3. BUSINESS RULES 5](#_Toc27512271)

[4. RELATIONAL SCHEMA 8](#_Toc27512272)

[4.1 XML SCHEMA 9](#_Toc27512273)

[5. Implementation in SQL SERVER 11](#_Toc27512274)

[5.1 Tables with Data Diagram 11](#_Toc27512275)

[5.2 Referential Integrity 11](#_Toc27512276)

[5.3 STORED PROCEDURES 13](#_Toc27512277)

[5.3.1. Employees should be able to login to the system, and their session should be maintained. 14](#_Toc27512278)

[5.3.2. Employees should be able to create holiday/tour packages. 15](#_Toc27512280)

[5.3.3. Employees should be able to update package details 17](#_Toc27512281)

[5.3.4. There should be functionality to search through the packages itinerary. 18](#_Toc27512282)

[5.3.5 On Deletion of Customer entry, all the enquiries associated with that customer will be deleted. 19](#_Toc27512283)

[5.3.6. Employees should be able to retrieve bookings created by customers with package details. 21](#_Toc27512284)

[5.3.7. Employees should be able to see the number of package based on package type 21](#_Toc27512286)

[5.4 TRIGGERS 22](#_Toc27512287)

[5.5 VIEWS 28](#_Toc27512288)

[8. Conclusion 29](#_Toc27512289)

[9. Innovation 29](#_Toc27512290)

[10. BIBLIOGRAPHY 31](#_Toc27512291)

[11.APPENDIX A 32](#_Toc27512292)

[CREATE TABLE QUERIES 32](#_Toc27512293)

[12. APPENDIX B 34](#_Toc27512294)

13. INDIVIDUAL CONTRIBUTION………………………………………………………………………………………….34

# TABLE OF FIGURES

[Figure 1 - ON DELETE CASCADE 5](#_Toc27512215)

[Figure 2 - Package Id foreign key 6](#_Toc27512216)

[Figure 3 - Employee role mapping 6](#_Toc27512217)

[Figure 4 - Role mapping table 7](#_Toc27512218)

[Figure 5 - Employee right function 7](#_Toc27512219)

[Figure 6 - Schema diagram 8](#_Toc27512220)

[Figure 7 - Schema.xsd file 9](#_Toc27512221)

[Figure 8 - XML Schema 10](#_Toc27512222)

[Figure 9 - Data Diagram 11](#_Toc27512223)

[Figure 10 - package table 12](#_Toc27512224)

[Figure 11 - Enquiry Table 12](#_Toc27512225)

[Figure 12 - Booking Table 13](#_Toc27512226)

[Figure 13 - Employee Role Mapping 13](#_Toc27512227)

[Figure 14 - generating a unique session token 14](#_Toc27512228)

[Figure 15 - Login Employee 14](#_Toc27512229)

[Figure 16 - SP1 Output 15](#_Toc27512230)

[Figure 17 - Create Package 16](#_Toc27512231)

[Figure 18 - Check Employee Session Function 16](#_Toc27512232)

[Figure 19 - Check employee has right function 17](#_Toc27512233)

[Figure 20 - SP2 Output 17](#_Toc27512234)

[Figure 21 - Update package Sp 18](#_Toc27512235)

[Figure 22 - Update package execution 18](#_Toc27512236)

[Figure 23 - SP3 Output 18](#_Toc27512237)

[Figure 24 - Search xml sp 19](#_Toc27512238)

[Figure 25 - search xml output 19](#_Toc27512239)

[Figure 26 – Enquiry Table 20](#_Toc27512240)

[Figure 27 - Delete customer sp 20](#_Toc27512241)

[Figure 28 - Customer booking retrieve sp 20](#_Toc27512242)

[Figure 29 - Customer booking retrieve output 21](#_Toc27512243)

[Figure 30 - Get Customer Count 22](#_Toc27512244)

[Figure 31 - Get Customer Count Execution 22](#_Toc27512245)

[Figure 32 - Customer audit table 23](#_Toc27512246)

[Figure 33 - package view 24](#_Toc27512247)

[Figure 34 - Package Caption 25](#_Toc27512248)

[Figure 35 - package trigger 26](#_Toc27512249)

[Figure 36 - get\_customer\_enquiry\_for\_employee 27](#_Toc27512250)

[Figure 37 - Get Customer PackageBooking CreatedBy Employee 28](#_Toc27512251)

[Figure 38 - Azure Clod Database 29](#_Toc27512252)

[Figure 39 - Connection from management studio 30](#_Toc27512253)

# BUSINESS CASE

Easy Travel is a system developed for a Tour packages management company to handle their travel/tour packages. It also provides features for managing and engaging their customers. The system is designed in such a way that every employee of the company will be able to use it and will perform operation they are allowed to. Every employee has a specific role which constraint them to performs specific task based on their roles. System is designed in such a way that employees can have many roles and there can be many employees of one role. The system can also be used for creating a customer's booking and also for entering the enquiries generated by the customer. For any enquiries customer contacts, the employee of the company based on their role and then enquiry gets entered into the system by employee. Customer can make booking on their own from a website of company based on available packages. Further reports can be generated using the system based on multiple factors like customers booking, enquiries, types of packages available etc.

# 2. SCOPE OF THE PROJECT

This project involves developing a system for tour management. In this report, the database aspect of this system is covered. Database management system used is MS SQL SERVER 2017, because fo its robust architecture and mainly for its feature which supports the XML datatypes as well.

According to the requirement, we have developed a database which includes several tables connected to each other.

We have entered a 5 records in every table for demo purpose including XML data. According to the requirements, every user will have different and multiple roles. But In this project, we have assigned every user a single role as of now but there is a functionality implemented for assigning them a multiple role.

For packages, we have inserted five records including 3 INTERNATIONAL and 2 DOMESTIC packages. Every package has a minimum of 3 days of itinerary. Image URL and date of every itinerary is kept null.

Also, Just two employee roles are being used now which are OPERATION & SALES, and other roles are created just to show the feature.

Other than the requirements and rules mentioned in this project, there are a lot more requirements and rules which is not discussed here.

## 2.1 BUSINESS REQUIREMENTS

1. Employees should be able to login to the system, and their session should be maintained.

2. Employees should be able to create holiday/tour packages.

3. Employees should be able to update package details

4. There should be functionality to search through the packages itinerary.

5. Employees should be able to retrieve bookings created by customers with package details.

6. Employees should be able to see the number of package based on package type

7. On Deletion of Customer entry, all the enquiries associated with that customer will be deleted.

# 3. BUSINESS RULES

**1. On deletion of a customer, all the enquiries associated with that customer must be deleted.**

It is required that if we delete the customer from the customer table all the enquires associated to that customer should be deleted from the enquiry table.

For this purpose, we have assigned the DELETE CASCADE RULE in enquiries table on customer\_id which is a foreign key in enquiry table and primary key in package table.

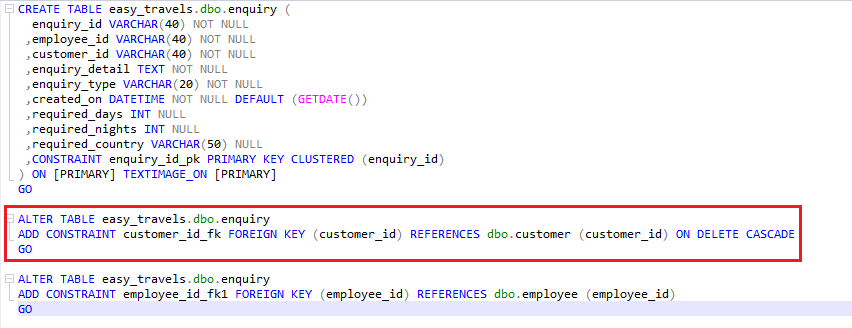


Figure 1 - ON DELETE CASCADE

**2. One booking must have only one valid package from the package table**

It is required that bookings done by customer must have a package, else booking cannot be done.

We have achieved this by setting not null in booking table on package\_id field and assigning foreign key for package\_id in booking table which is primary key in packages table

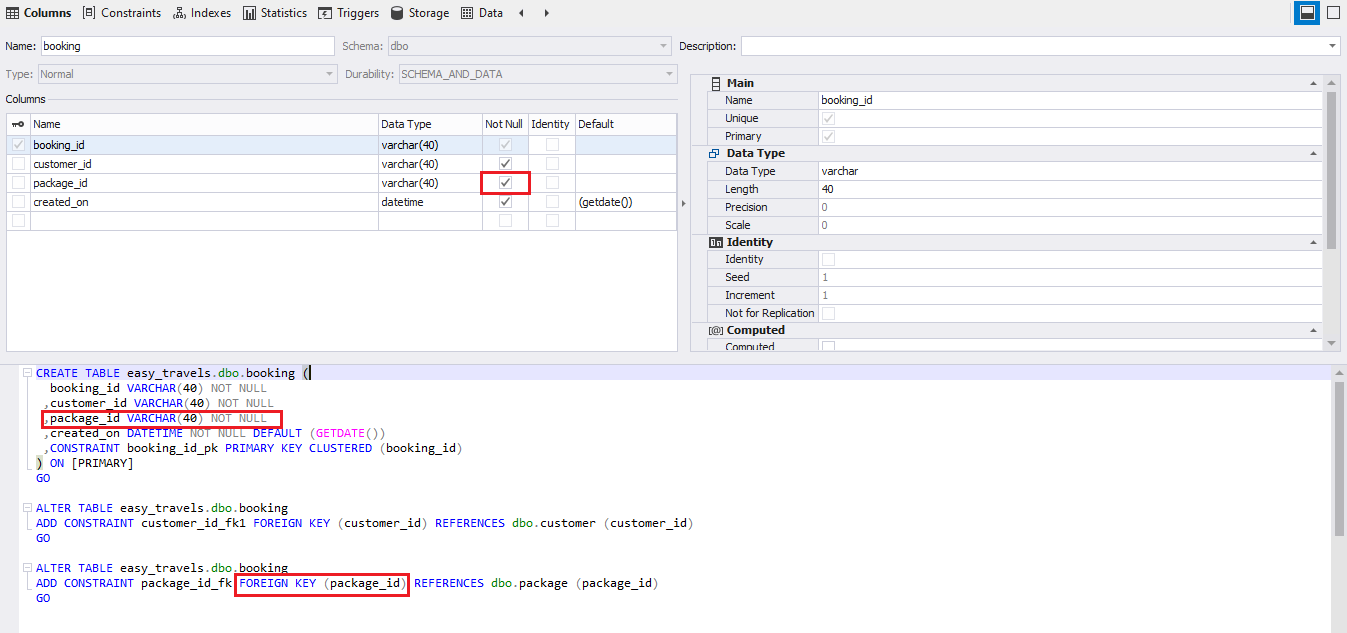


Figure 2 - Package Id foreign key

**3. Every employee should have at least one or many roles.**

It is required that every employee can have one or more than one roles. Ideally, every employee should have one role but, for a company which we are building a system is small scale company so one employee of that company can handle the many aspects of the organization.

Since it is many to many mapping, we have created a table named employee\_mapping which contains the mapping of employee and roles. role is a table which contains all the static data which is mapped against the employees in mapping table with employee\_id and role id as a foreign key constraints.

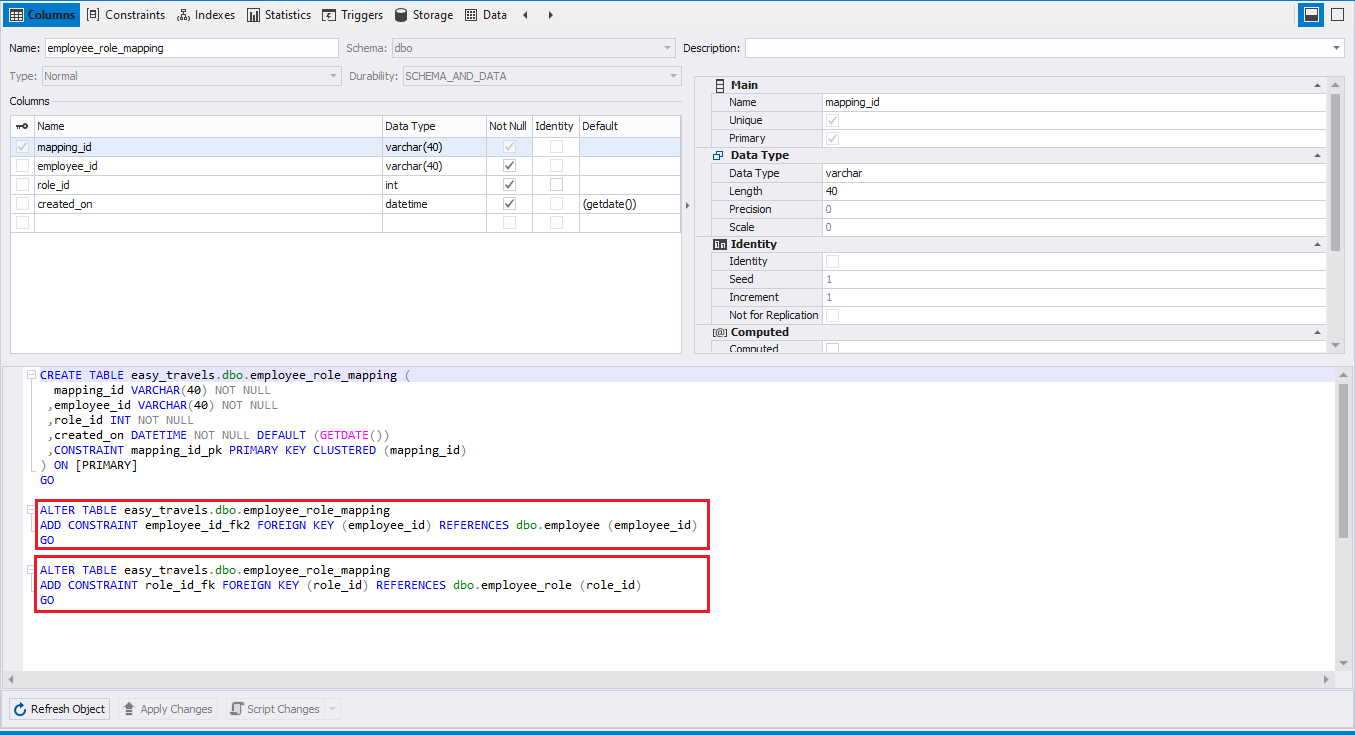


Figure 3 - Employee role mapping

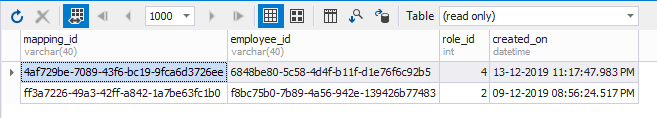
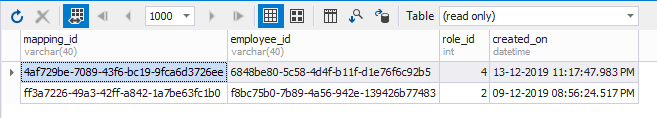


Figure 4 - Role mapping table

**4. For the creation of packages, only employees with operation role can create packages.**

One of the rules is that every package should be created by the employee who has an OPERATION role i.e 4. Employees with other roles cannot create or update package. And employee with operation role cannot perform any other operations.

To achieve this functionality we have mapped a role to every employee. And while creating a package we check whether the employee creating a package has the authority to create the package. We have written a function for this.



Below function will take two params @e\_id i.e employee\_id and @e\_id. It will return **bit 1** if the employee has a specified role and will return **bit 0** if he doesn’t have a right to perform this operation.

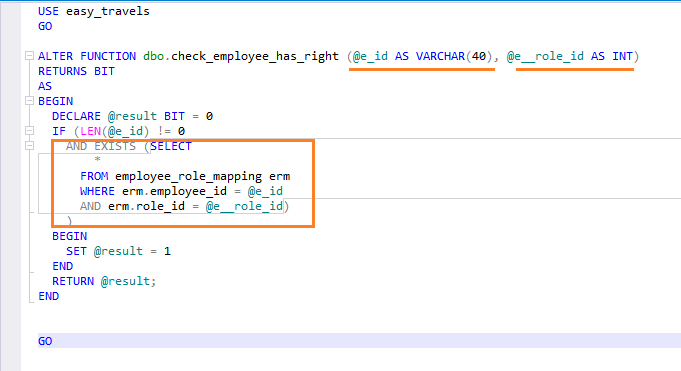


Figure 5 - Employee right function

# 4. RELATIONAL SCHEMA

Normalization is a process of designing a database in such a manner that data redundancy and data dependency is reduced. It is a process of dividing a larger table into smaller sub tables and then linking the tables via relationship.(Kumar and Azad, 2017)

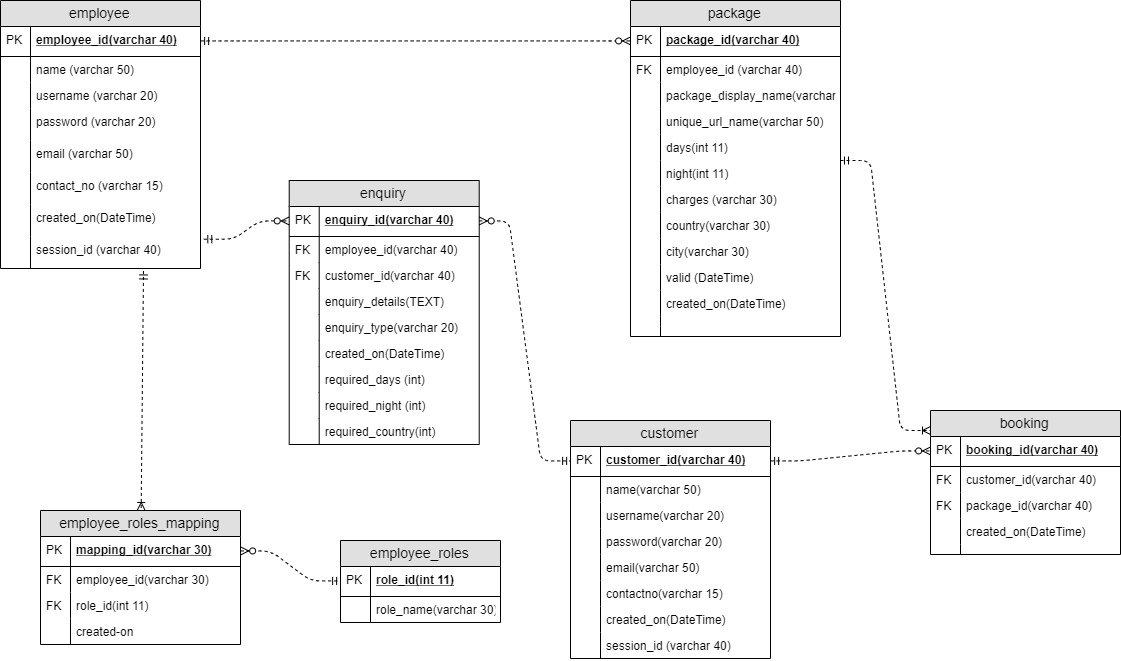


Figure 6 - Schema diagram

**First Normal form:** In the first normal form all the records should contain a single value and each record or row must be unique. Our schema is already in first normal form according to the data we have selected. (Ramez Elmasri and Navathe)

**Second Normal form:** In the second normal form the schema should be in first normal form and there should be no partial dependency. In our schema, all the values are already in 1NF and each table is having a unique primary key. There is no composite key, hence the schema is in 2NF. (Ramez Elmasri and Navathe)

**Third Normal form:** In the third normal form the schema should be in second normal form and there should be no transitive dependency. Transitive dependency is when a non-prime attribute depends upon another non-prime attribute in the table. In our schema, the data we have taken is in such a manner that all the non-prime attribute depends upon the primary key (Ramez Elmasri and Navathe)

of the table only and not on other non-prime attribute of the same table. We can say that our schema is in 3NF.

## 4.1 XML SCHEMA



Figure 7 - Schema.xsd file

Above diagram is created for defining our XML. XML schema is nothing but how and in what sequence your data will appear in XML format. It also shows what data type is used by each element. How many number of child elements it will have. It basically a well-defined structure of your XML. We can also validate our XML data with this schema. (“W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures”,(Checiu and Ionescu, 2010))

**4.2 Use of XML**

As the use of Internet is growing day by day, data is also generated in proportion to that. Relational databases have certain limitation on various factors like scalability, maintenance and performance. So to store this vast amount of data, use of just relational databases is inappropriate, to overcome that now hybrid databases are being used. In hybrid databases, one can use XML data type to store huge amount of data. Using XML datatype multiple rows and columns can be converted to a single cell. Also, data can be stored in a well-structured form using datatypes as well as their properties in a sequential manner.

We can perform all the CRUD operations in XML data also like we use in relational databases.

Overall XML is used for storing the well defined, well-structured data which is relation with other data.

In our system, we have used XML for storing our day-wise itinerary. In this day-wise itinerary, we have to store the records of the number of days which contains many different types of data. And one package contains many such days, so if we were not using XML for this field, we had to use many rows for this data. Hence using this data we can combine all the values in single filed hence performance is also increased.



Figure 8 - XML Schema

This is the sample of our XML data.

**5. Implementation in SQL SERVER**

## 5.1 Tables with Data Diagram

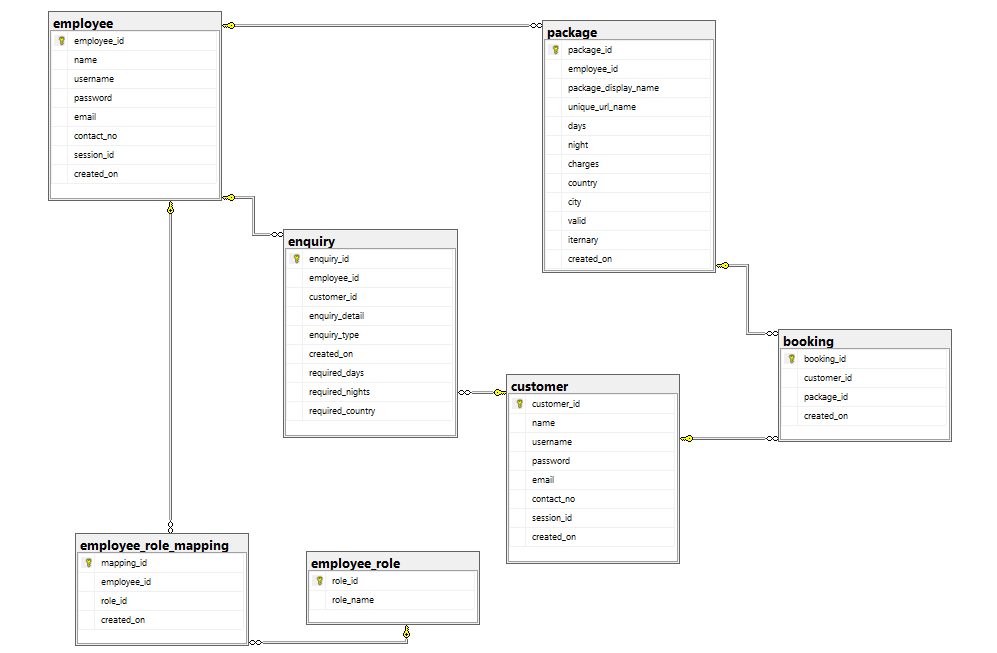


Figure 9 - Data Diagram

Above diagram shows the data diagram for our system.

Create a table and insert queries are placed in Appendix Section

## 5.2 Referential Integrity

Referential integrity is nothing but when we use the primary key of one table as the foreign key in another table, then that primary key should exist and should be valid. Referential integrity makes sure that all the data is integrated and accurate. Primary key cannot be modified or deleted if that is used as a foreign key in some other child table unless we set some specific rules. And using referential integrity data duplication is also avoided.

In this system, many referential integrities are being used discussed below.

**1.Package Table**

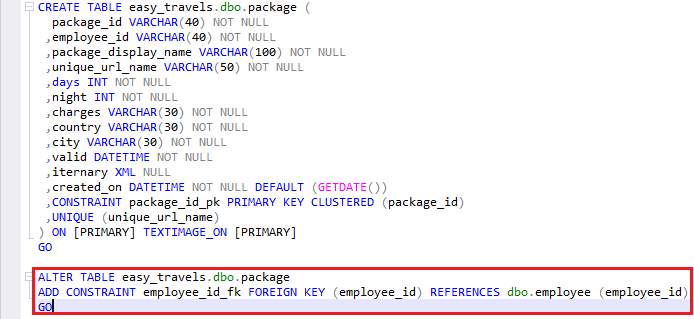


Figure 10 - package table

As mentioned earlier employees can create packages. So we have used **employee\_id\_fk** as a foreign key constraint in the package table. employee\_id is a primary key in the employee table.

If we try to delete the employee who has his entry in the package table, it will throw an error because we have used that employee’s primary key as a foreign key package table. On the other hand package’s entry cannot be created without employee id.

**2.Enquiry Table**

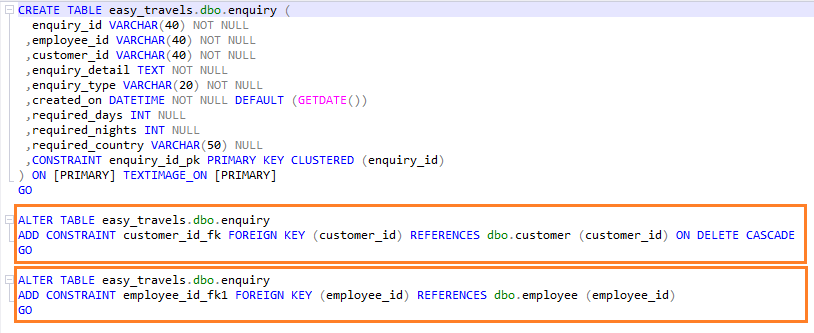
****

Figure 11 - Enquiry Table

In this table, two foreign keys are used **customer\_id\_fk** and **employee\_id\_fk1** because enquiries are created for customer so customer’s id is used as foreign key and enquiries are created by employee so his id is also used as a foreign key.

**3.Booking table**

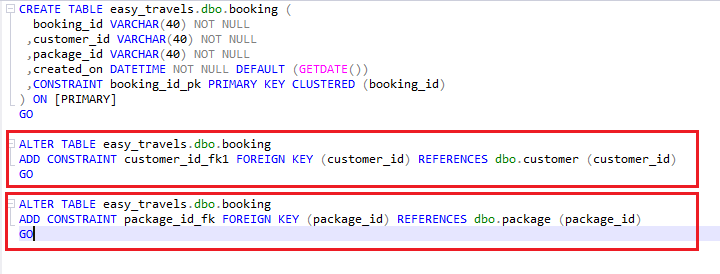
****

Figure 12 - Booking Table

In this table, two foreign keys are used one is **customer\_id\_fk1** because booking is done by the customer and one is **package\_id\_fk** because booking is done against the package.

**4.Employee Role Mapping**

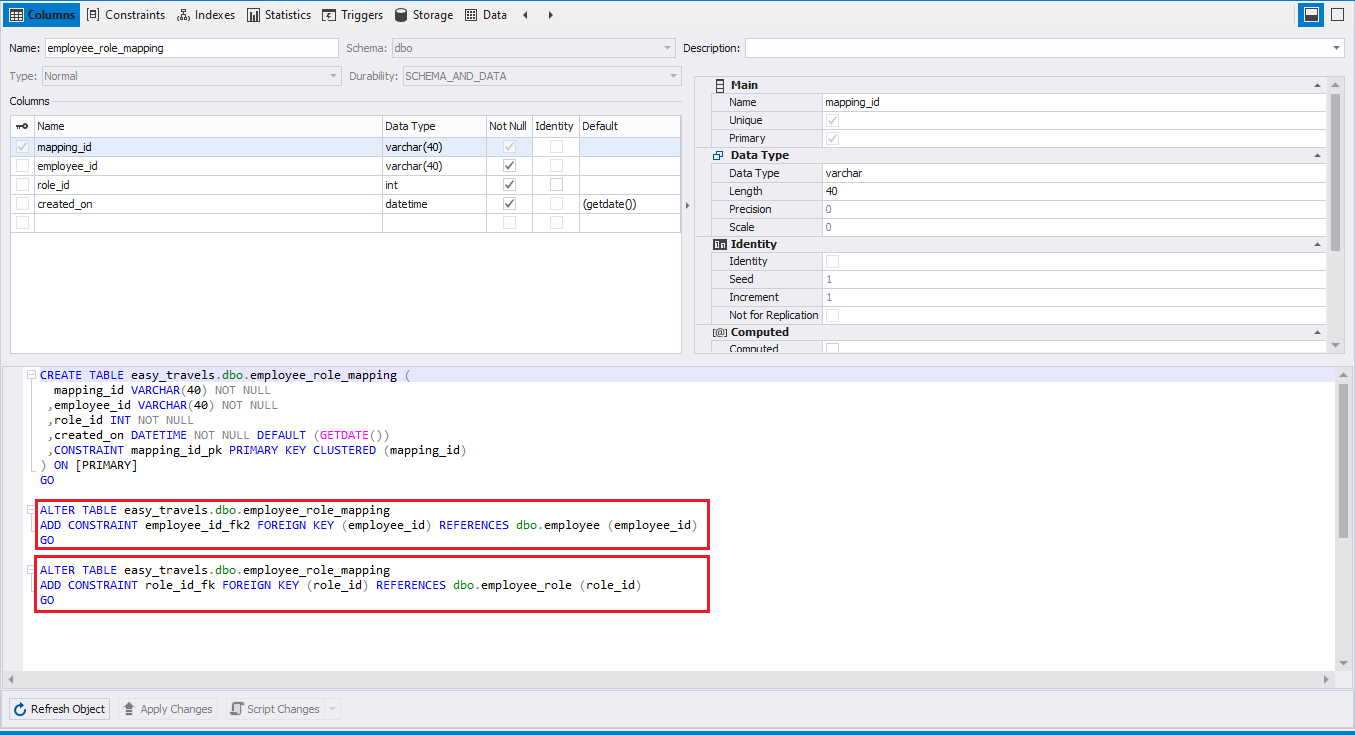


Figure 13 - Employee Role Mapping

As per the business requirement, every employee should have some role and every employee can have multiple roles, and a single role can be shared by multiple employees.

So, one foreign key used for **role\_id\_fk** which is primary key of roles table and other is **employee\_id\_fk2** because roles are mapped to employees.

## 5.3 STORED PROCEDURES

## 5.3.1. Employees should be able to login to the system, and their session should be maintained.

Features Used – Custom functions, Inbuilt functions

It is one of the most important requirement that while performing any operation, employee should be logged in and has a valid session token.

We have achieved this using a stored procedure and function as shown below.

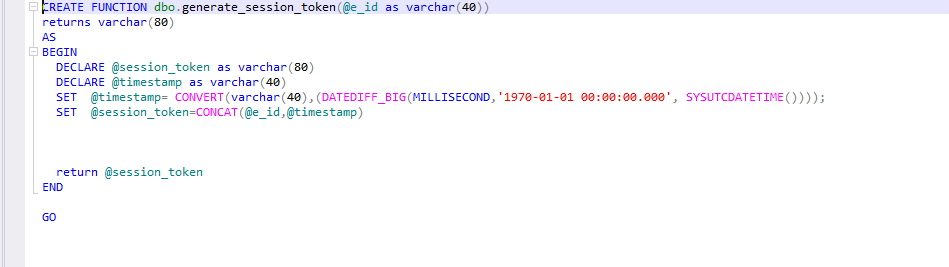


Figure 14 - generating a unique session token

The above-shown image is a function written for generating a unique session token. It accepts an employee id as a parameter. Then using DATEDIFF\_BIGG function it finds a current timestamp in milliseconds and in the next step it concatenates the time stamp with the employee id provided as a parameter and returns the same.

For eg – if we pass **‘5d2f-f25g4f5g-2s4dsd-sd41s5d4-ds’** as emp id then function will return

**‘5d2f-f25g4f5g-2s4dsd-sd41s5d4-ds11515151115’ string as a output**

****

Figure 15 - Login Employee

Above stored procedure is created for login functionality of an employee.

It takes @e\_username and @e\_password as an input parameter first it checks whether employee records exist with this username and password if it exists it stores 1 in a count variable. If count is 1 then it proceeds ahead or it will print **‘WRONG CREDENTIALS’** as an error message. After that, it fetches employee id from the employee table with that credentials

And passes it to the function **generate\_session\_token(@e\_empId)** for generating session token. Once the session token is generated, then the employee table is updated with the new session token. And further operations are performed based on this session token.

## OUTPUT –

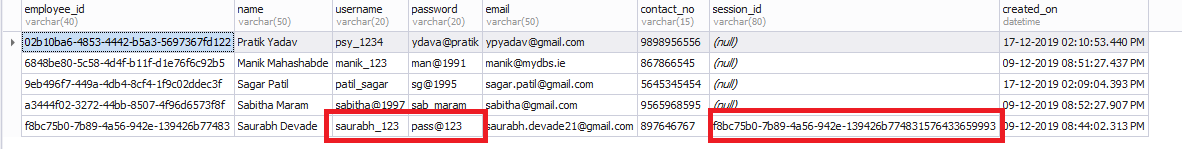


Figure 16 - SP1 Output

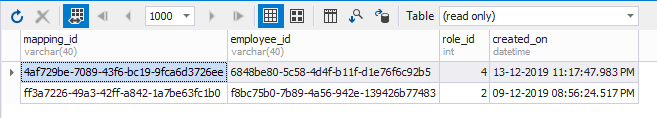
Session token is generated for the user whose credentials are passed.

## 5.3.2. Employees should be able to create holiday/tour packages.

Features Used – XML insert, Custom functions, Inbuilt functions

One of the requirement is that the employee should be able to create holiday packages for a specific tour. But as mentioned in the Business case it is required that employees with OPERATION role can create packages and other employees cannot do it.

Separate table is mentioned for maintaining employees and roles mapping.



And static data of employees are kept in another table.



Figure 17 - Create Package

This stored procedure is created for creating packages.

It accepts all the parameter which is required for creating a package and an extra **@session\_token** parameter to check employees authority.



Figure 18 - Check Employee Session Function

Above function is created to check whether the employee is logged in and has a valid session.

It accepts two parameters **@e\_id** and **@session\_token** and will return BIT as an output.

It will return 1 if the record is found with provided session\_token and employee id that means employee is logged in and has a valid session token and will return a 0 if no such record is found that means the employee is not logged in.

Then to check whether the employee has a right to perform this operation we have another function named as **check\_employee\_has\_right.**

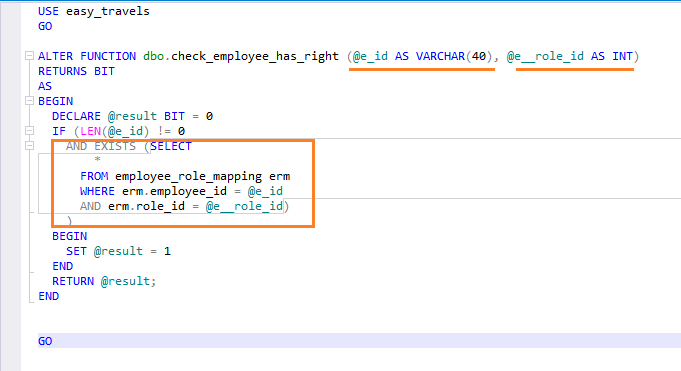


Figure 19 - Check employee has right function

While calling this function we have to pass two arguments i.e **@e\_id and @e\_role\_id**. Function will check whether a record exists for an employee for the provided role and will return BIT accordingly that. And if both the conditions satisfy, it will insert data into the package table having itinerary as an XML datatype.

**OUTPUT-**

Once the query is executed successfully data gets inserted into the table.

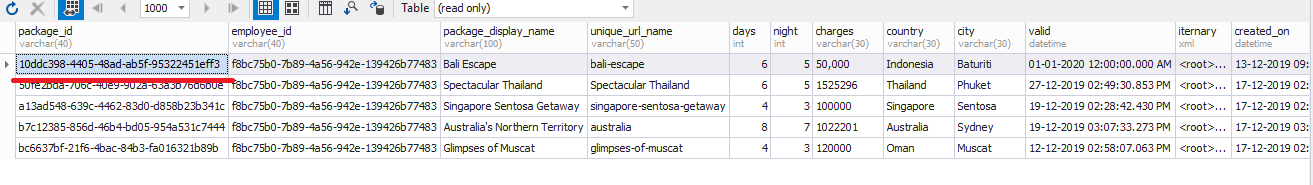


Figure 20 - SP2 Output

## 5.3.3. Employees should be able to update package details

Features Used – XML modify, Custom functions, Inbuilt functions

Employees should be able to update XML data available in package

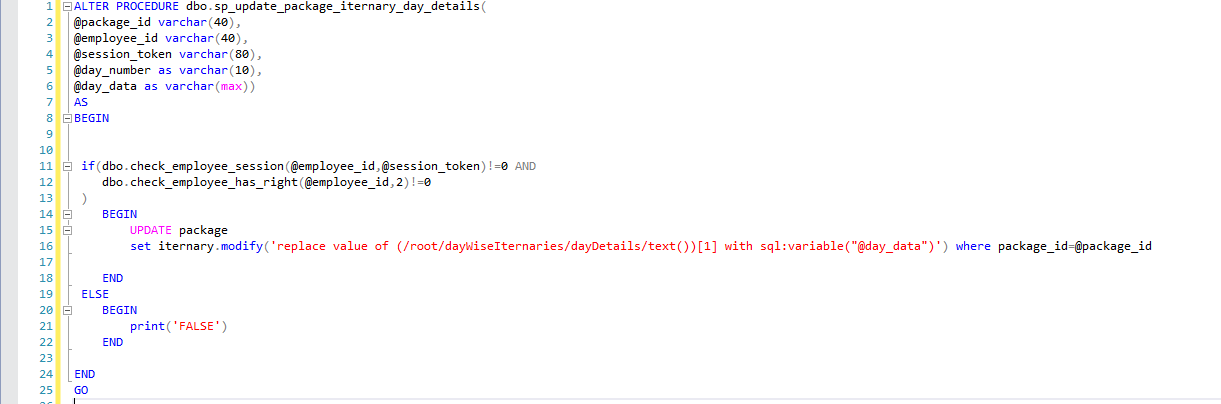


Figure 21 - Update package Sp

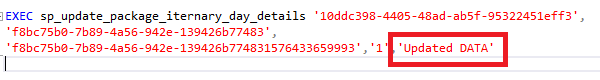


Figure 22 - Update package execution

When we execute the stored procedure and pass data to be updated. Its update the XML data of package which is passed as an input parameter.

**OUTPUT –**

****

Figure 23 - SP3 Output

## 

## 5.3.4. There should be functionality to search through the packages itinerary.

Features Used – XML search, LIKE statement, Custom functions, Inbuilt functions

If employee or customer wants to search for a package with specific information there should be some feature for that. Every package has separate Day-wise itinerary in XML format, and it is possible that employee or customer want to search through that XML data as well.

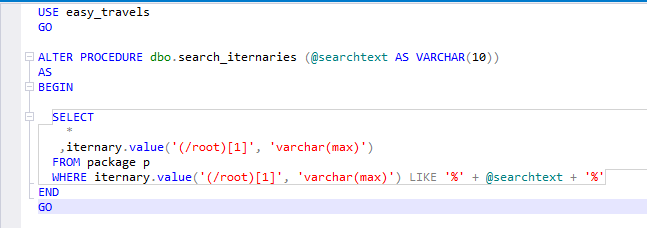


Figure 24 - Search xml sp

Above stored procedure search\_iternaries accepts one input parameter which is nothing but a search field. We have to search from an entire package itinerary that’s why we are searching from a root level using the **LIKE** keyword. ‘%’ sign at both ends indicates that it will find all matched records which contain the passed input.

**Output:**

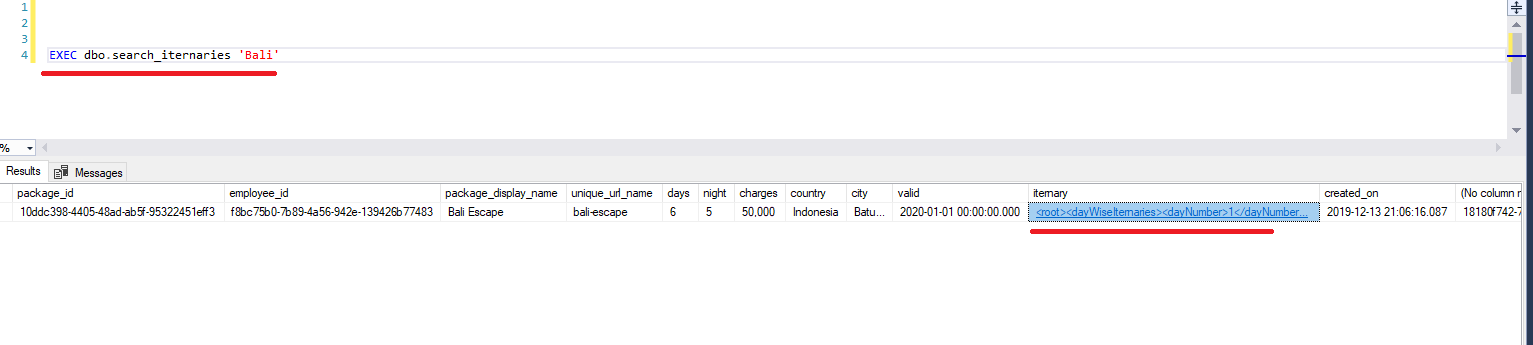
****

Figure 25 - search xml output

From the screenshot above it can be seen that when we execute the Stored procedure passing Bali as value it returns us all the packages which have Bali as a text in it.

## 5.3.5 On Deletion of Customer entry, all the enquiries associated with that customer will be deleted.

Features Used - DELETE CASCADE statement, Inbuilt functions

When an employee wants to delete a customer’s record. All the enquiries associated with that customer should automatically get deleted. It is needed to maintain the database very neatly and reduce the employee's works to remove the enquiries manually.

For that, we have mapped a customer\_id as a foreign key in enquiries table, and have set ON DELETE CASCADE rule which will automatically remove the customer's enquiry from the enquiries table.

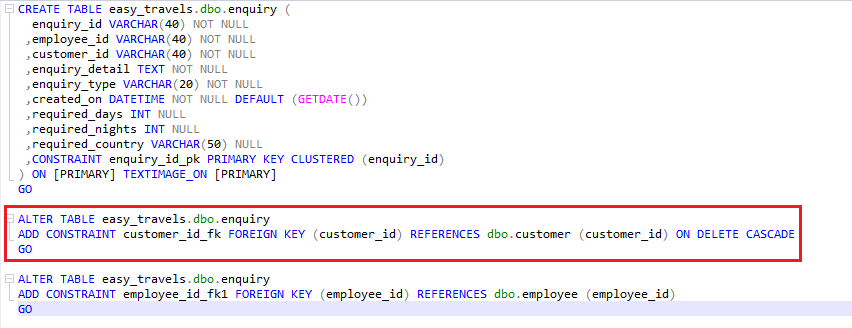


Figure 26 – Enquiry Table

Above image shows the foreign key **customer\_id\_fk** mapping and the **ON DELETE CASCADE**.

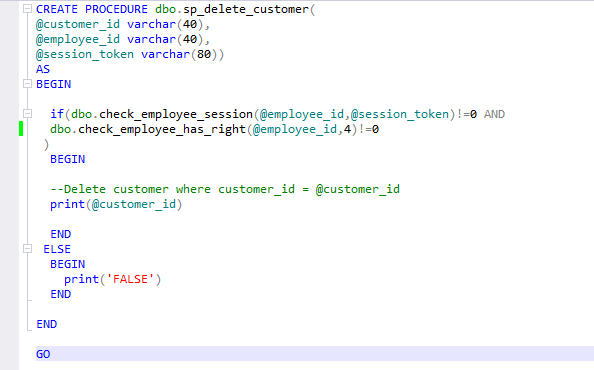


Figure 27 - Delete customer sp

As mentioned earlier every employee can perform specific operations only based on their role. And it is also discussed that employee should have valid session toke to perform operations. This two check is implemented in separate functionalities and called multiple time as shown in the above figure.

Once an employee is authenticated and authorized, the delete query is executed and as soon as this operation is performed customers enquiries associated with it is also deleted automatically.

## 5.3.6. Employees should be able to retrieve bookings created by customers with package details.

Features Used – Multiple Sql Joins, Custom functions, Inbuilt functions

Once the booking of an employee is done against the package id, its records will be saved in a booking table. To avoid the data duplication we are not storing all the package data into the booking table. We are just mapping the package id against the booking id and customer id.

After that, when customer or employee wants to see all the booking details which is nothing but a package and customers details we will have to join three tables.

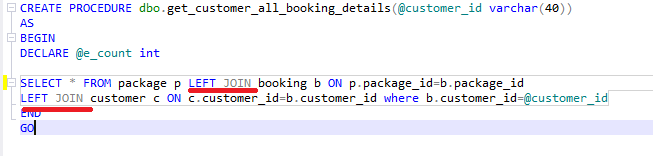


Figure 28 - Customer booking retrieve sp

We can see two joins here and customer\_id is passed as an argument. Joins are used for joining two tables and accessing their data.

Here we want to retrieve all the package details from package\_id which is stored in bookings table so we have used a join on **package\_id** of package table and **package\_id** from booking table. Same is done for customer details also we have joined customer table and booking table over the customer\_id for retrieving customers details as well.

## OUTPUT –

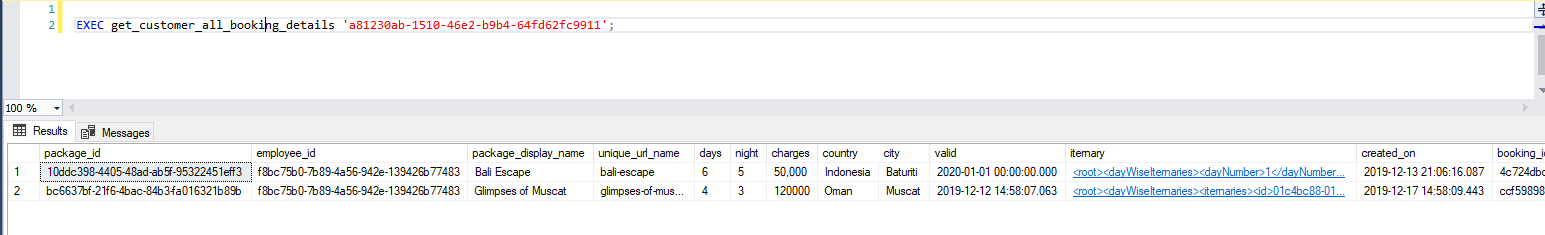


Figure 29 - Customer booking retrieve output

## 5.3.7. Employees should be able to see the number of package based on package type

Features Used – HAVING, GROUP BY, COUNT, Custom functions, Inbuilt functions

There are two types of enquiries basically. INTERNATIONAL and DOMESTIC. Once all the enquiries are generated if someone wants o see the number of enquires generated based on type there is featured implemented for this.

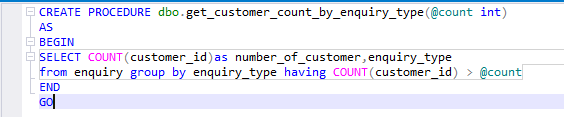


Figure 30 - Get Customer Count

Fro the code shown in the above screenshot we have used an inbuilt **COUNT()** function for counting a number of records. But instead of showing all the records we are grouping them bye a **group by** statement. We are grouping them using type column so it will group all the row which has the same type. And at last, we are passing a count as an argument so it will return the type of enquires which are above this passed count.

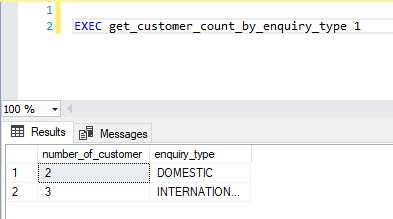


Figure 31 - Get Customer Count Execution

## 5.4 TRIGGERS

Triggers are the special stored procedure which is automatically executed whenever any DML, DDL action takes place. (Yoon *et al.*, 2001; Ding *et al.*, 2013)

We have created two triggers for our database.

**1. AFTER Trigger for customer record delete and insert from customer table.**

In this case, we have created a table named Customer.audit which will consist of inserted or deleted records from the customer table.

It is shown below:

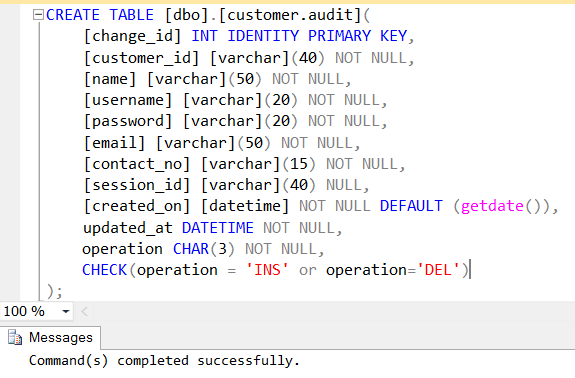
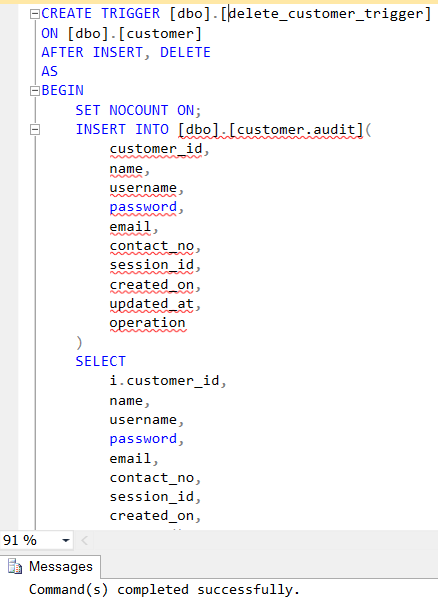


Figure 32 - Customer audit table

Then we have an AFTER trigger delete-customer-trigger which will be called after action is taken on the customer table. It will insert the delete/insert records in the customer-audit table.



2. INSTEAD OF trigger used for authentication before placing record in parent table.

To perform this we created a view package\_view which consist of records of package table along with the column approved or not approved.

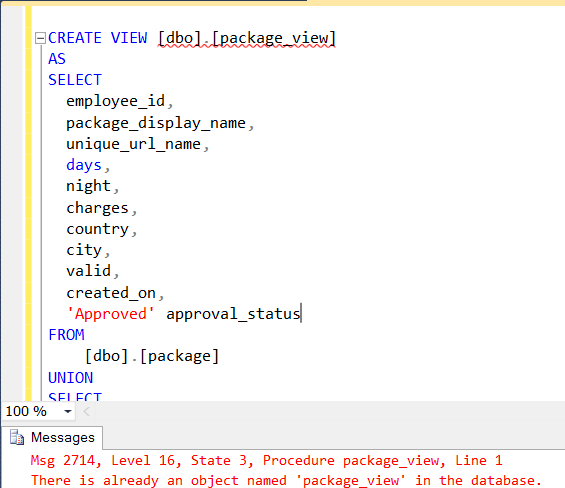


Figure 33 - package view

There is a secondary table package\_validity in which package records are inserted if the package name doesn’t exist in the package table. An INSTEAD OF trigger named package\_trigger is performing this job. As soon as records are inserted in the view trigger is fired which places the record in package\_validity table if it is not found in package table. The purpose of this trigger for authentication. If in case a new record is inserted in package table. It needs to be authenticated and validated. If everything is fine then it is inserted into the main table.

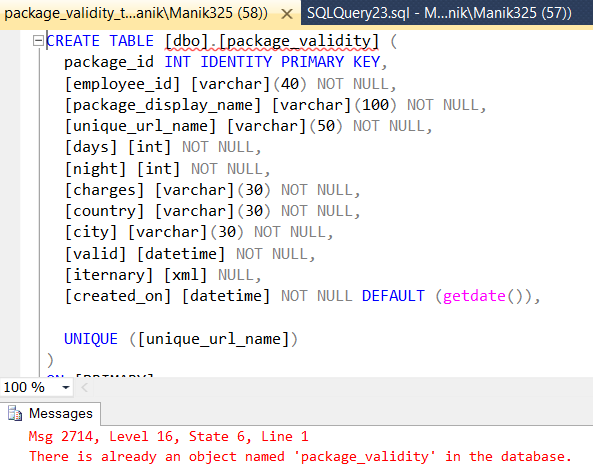


Figure 34 - Package Caption

INSTEAD OF trigger is shown below:

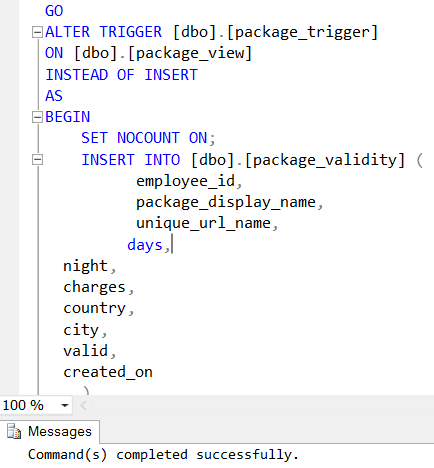


Figure 35 - package trigger

## 5.5 VIEWS

Views are used to create a virtual table. If we want to hide the data implementation from outside world views are generally used. If there is a requirement to join multiple tables and gives access to the user for specific fields from the table. Then as a general practice, a view is created and access of view is given to the customer so they would know about internal tables.

We have created two views for our database. (stevestein,2017)

1. In the first view get\_customer\_enquiry\_for\_employee, we are joining three tables customer, enquiry and employee.

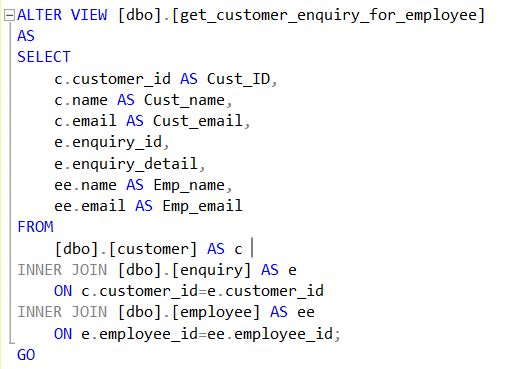
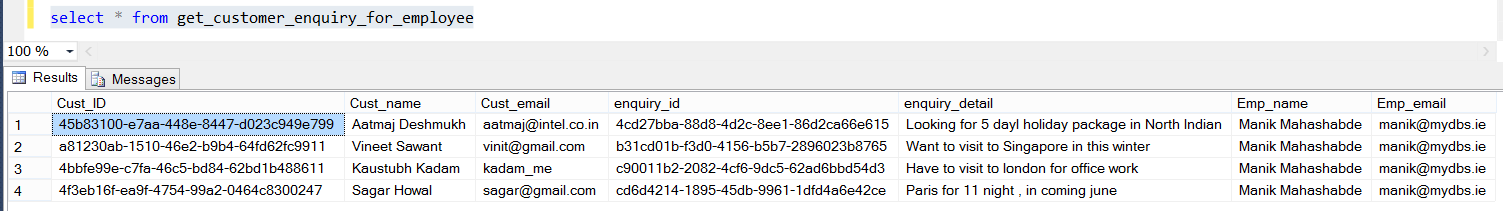


Figure 36 - get\_customer\_enquiry\_for\_employee

If we select from the view result can be seen below.



Purpose of this view is to give access to the customer about the enquiry raised by them and which employee is handling that enquiry request.

2. In this view [get\_customer\_packageBooking\_CreatedBy\_employee] we are creating a view by joining customer package, booking, employee as seen below.

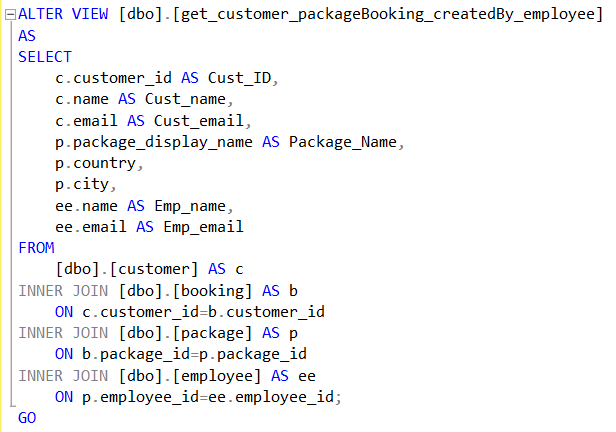
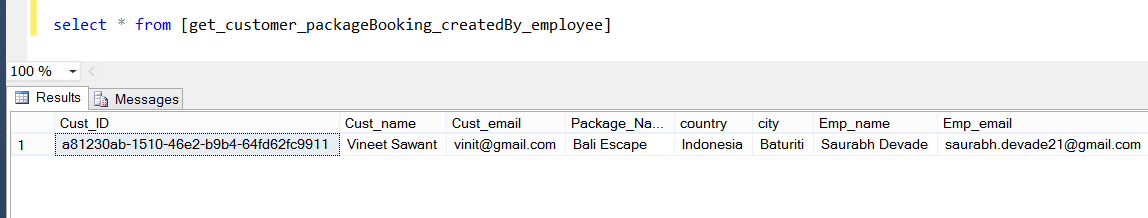


Figure 37 - Get Customer PackageBooking CreatedBy Employee

This purpose of this view to give customers the access so that they can view their booking and package details assigned to an employee of the organization as shown below.



# 

# 8. Conclusion

Finally, we conclude that the Use of SQL SERVER is preferable for so many reasons. One of the reason is, it gives support to the XML datatype which helps to keep data in a well-structured manner and also reduces the number of rows in a table. Also, the view feature provide by SQL server helps you in abstraction of data. i.e you can show only the required data to the users who shouldn’t be seeing any high sensitive data, you show them just virtual created tables.

Triggers help you to detect some events which allows you to generate audit reports. Using stored procedure is also essential because it enhances the performance of your application and also you can perform complex operation rather than performing just simple queries.

Even though making a schema diagram takes time, it is essential for your team to have one so they will easily get an idea of how your system communicates with each other. So overall Using of SQL SERVER with proper project planning helps you in creating a well-managed Database system.

# 9. Innovation

For innovation, we have deployed our easy\_travels database in management studio into the AWS cloud server. Then on our SQL Server Management Studio, we are creating a server named **easytraveldbcloud.** Service used for this is Azure SQL server. (“SQL Database – Cloud Database as a Service | Microsoft Azure”)

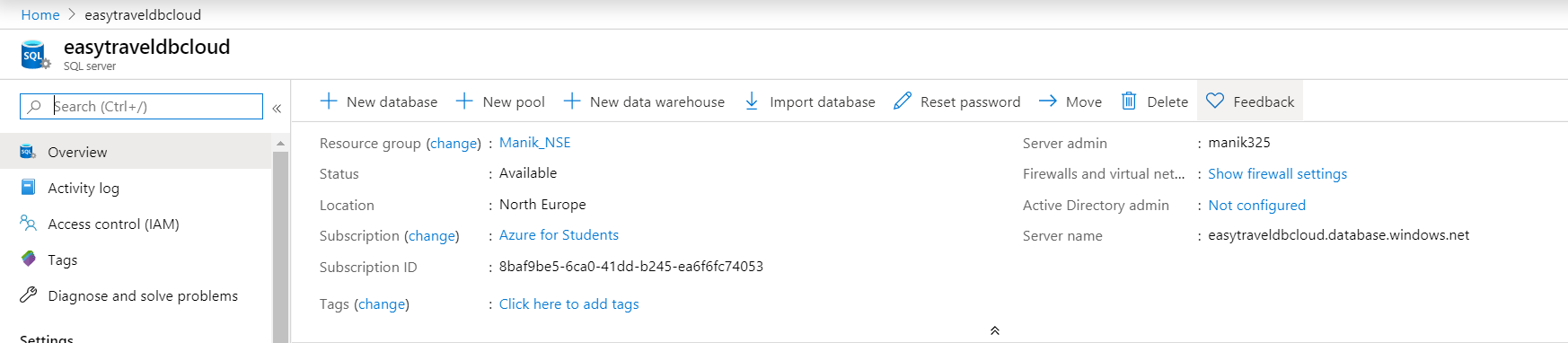
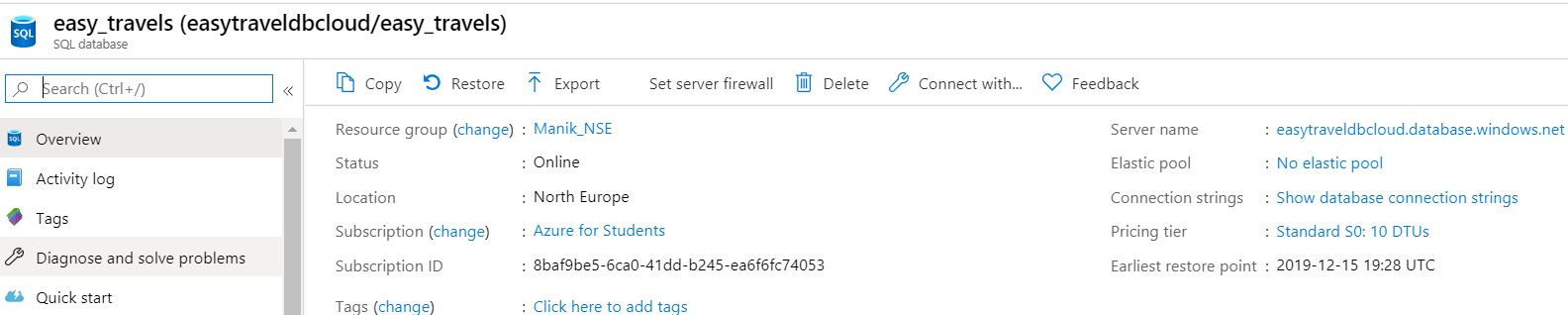


Figure 38 - Azure Clod Database

After creating and selecting hardware configuration of our server. Next step was to deploy the easy\_travels into the cloud server and adding the local system IP address into firewall settings of the Azure server.



Then we can access the azure server from our management studio using the below details:

Server Name: **easytraveldbcloud.database.windows.net**

Username: **manik325**

Password: **June-2019**

After this, it can be seen that we can connect to our azure cloud server instead of a regular one which is installed on the local machine.

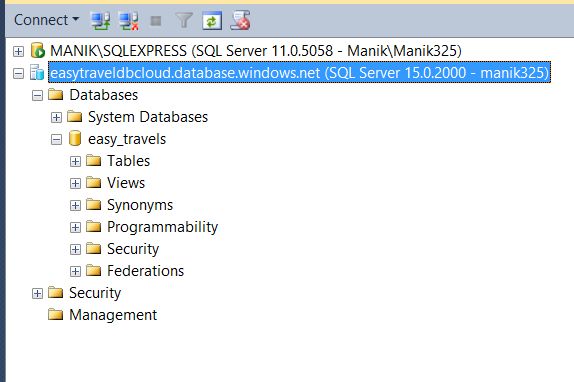


Figure 39 - Connection from management studio

# 10. BIBLIOGRAPHY

Checiu, L. and Ionescu, D. (2010) ‘A new algorithm for mapping XML schema to XML schema’, ICCC-CONTI 2010 - IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics, Proceedings. IEEE, pp. 625–630. doi: 10.1109/ICCCYB.2010.5491337.

Ding, L. et al. (2013) ‘Research on SQL Server trigger to implement referential integrity’, Proceedings of 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, ICIII 2013. IEEE, 3, pp. 286–288. doi: 10.1109/ICIII.2013.6703572.

Kumar, K. and Azad, S. K. (2017) ‘Database normalization design pattern’, 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics, UPCON 2017, 2018-Janua, pp. 318–322. doi: 10.1109/UPCON.2017.8251067.

Ramez Elmasri, and Shamkant B Navathe. Fundamentals of Database Systems. Boston, Pearson Education, 2007.

“SQL Database – Cloud Database as a Service | Microsoft Azure.” Microsoft.Com, Microsoft Azure, 2019, azure.microsoft.com/en-in/services/sql-database/.

stevestein. “Views - SQL Server.” Microsoft.Com, 14 Mar. 2017, docs.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver15. Accessed 17 Dec. 2019.

“W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures.” W3.Org, 2012, www.w3.org/TR/xmlschema11-1/. Accessed 17 Dec. 2019.

Yoon, H. S. et al. (2001) ‘Developing a triggering system for real-time databases in distributed environment’, Proceedings - 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC 2001, pp. 366–371. doi: 10.1109/ISORC.2001.922861.

‌

# 11.APPENDIX A

## CREATE TABLE QUERIES

1. Create Package Query

GO

CREATE TABLE package

(

package\_id varchar(40) not null,

employee\_id varchar(40) not null,

package\_display\_name varchar(100) not null,

unique\_url\_name varchar (50) not null UNIQUE,

days int not null,

night int not null,

charges varchar (30) not null,

country varchar (30) not null,

city varchar (30) not null,

valid DateTime not null,

iternary xml not null,

created\_on datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT package\_id\_pk PRIMARY KEY(package\_id),

CONSTRAINT employee\_id\_fk FOREIGN KEY(employee\_id)

REFERENCES employee(employee\_id)

)

2. Create Employee Table

GO

CREATE TABLE employee

(

employee\_id varchar(40) not null,

name varchar(50) not null,

username varchar(20) not null,

password varchar(20) not null,

email varchar(50) not null,

contact\_no varchar(15) not null,

session\_id varchar(40) null,

created\_on datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT employee\_id\_pk PRIMARY KEY(employee\_id)

)

3. Create Customer Table

CREATE TABLE customer

(

customer\_id varchar(40) not null,

name varchar(50) not null,

username varchar(20) not null,

password varchar(20) not null,

email varchar(50) not null,

contact\_no varchar(15) not null,

session\_id varchar(40) null,

created\_on datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT customer\_id\_pk PRIMARY KEY(customer\_id)

)

4. Create Enquiry Table

GO

CREATE TABLE enquiry(

enquiry\_id varchar(40) NOT NULL,

employee\_id varchar (40) NOT NULL,

customer\_id varchar (40) NOT NULL,

enquiry\_detail text NOT NULL,

enquiry\_type varchar (20) NOT NULL,

required\_days int NULL,

required\_nights int NULL,

required\_country varchar(50) NULL,

created\_on datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT enquiry\_id\_pk PRIMARY KEY (enquiry\_id),

CONSTRAINT employee\_id\_fk1 FOREIGN KEY(employee\_id)

REFERENCES employee(employee\_id),

CONSTRAINT customer\_id\_fk FOREIGN KEY(customer\_id)

REFERENCES customer(customer\_id)

)

5. Create Booking Table

GO

CREATE TABLE booking(

booking\_id varchar(40) NOT NULL,

customer\_id varchar (40) NOT NULL,

package\_id varchar (40) NOT NULL,

created\_on datetime NULL,

CONSTRAINT booking\_id\_pk PRIMARY KEY (booking\_id),

CONSTRAINT customer\_id\_fk1 FOREIGN KEY(customer\_id)

REFERENCES customer(customer\_id),

CONSTRAINT package\_id\_fk FOREIGN KEY(package\_id)

REFERENCES package(package\_id)

)

6. Create Employee Role Table

GO

CREATE TABLE employee\_role(

role\_id int NOT NULL,

role\_name varchar (30) NOT NULL,

CONSTRAINT role\_id\_pk PRIMARY KEY (role\_id)

)

7. Create Employee role mapping table

GO

CREATE TABLE employee\_role\_mapping(

mapping\_id varchar(40) NOT NULL,

employee\_id varchar (40) NOT NULL,

role\_id int NOT NULL,

created\_on datetime NULL,

CONSTRAINT mapping\_id\_pk PRIMARY KEY (mapping\_id),

CONSTRAINT employee\_id\_fk2 FOREIGN KEY(employee\_id)

REFERENCES employee(employee\_id),

CONSTRAINT role\_id\_fk FOREIGN KEY(role\_id)

REFERENCES employee\_role(role\_id),

)

# 12. APPENDIX B

All insert queries are attached as separate sql files.

# 13. Individual Contribution

My contribution in the project was to collaborate with Saurabh on selecting a database model for the project. We discussed various databases to work on for exam:- departmental store, hospital, travelling. Then finally both of us came up with easy-travels database. Then we came up with what kind of business requirements should be there. I specifically came up with the idea about 3 tables employee, enquiry and customer and learned about table creation in SQL management studio. Deciding on the type of attributes in each table should be there and what kind of relation should be there between them. I also learned about the relation between 3 tables. Then Saurabh and I read through CA and decided what kind of business requirements should be there and later came with 3 stored procedures for their implementation. I worked on **employee login stored procedure** and worked on **creating session token function**, **get all customer booking stored procedure** and **search for specific event stored procedure**. I worked on searching about xml field and suggested about kind of xml field we can keep in our project which was later continued by Saurabh. Then later I worked on creating **customer enquiry for employees view** and learned about joining tables and data abstraction for clients. After this I worked on trigger can came up with **INSTEAD OF trigger** where I created a view for storing package table data and creating a secondary table for package validity. I learned and implemented that whenever a new package is inserted in the view so before moving it to main package table I should be inserted in package\_validity table by the trigger for authentication purpose. Then me and Sourabh worked on schema diagram design where we studied about normalization is done to 3NF and various integrity constraints for our database. After that we designed the schema on draw.io and identified about weak and strong entity in the relation. Also identified about various types of relationship in the table. Once everything was completed and we had a running database in the management studio and all the functionalities which me and Sourabh added were working correctly. The final step was to introduce innovation in the project. We thought of introducing cloud computing in our project where we created a SQL server in AZURE and later on deployed our local database in the cloud from management studio. The reason for adding cloud is that anyone can access the database server from their local system. We first need to add the client IP into our azure firewall then using the credentials given by us in the report anyone can connect to the server and view our database. Overall, I learned about various functionalities of a database system as taught in the class and later on researched more about them on the internet and also by relating to my previous work experience and how I can efficiently use them for the given CA. I am very sure that these new skills which I learned are going to help me to become a good backend developer in my career.