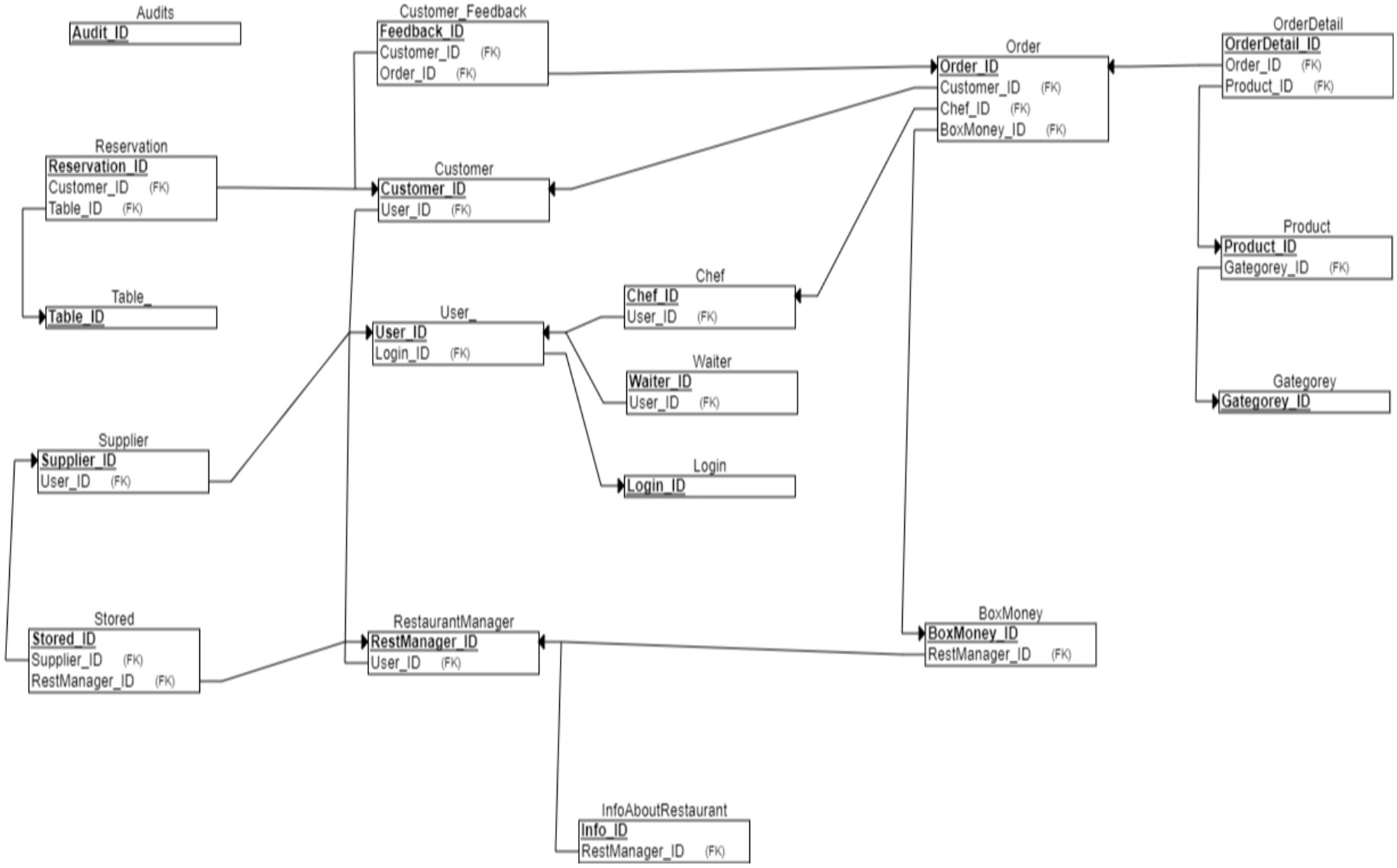
Audits
Audit_ID
Customer_Feedback
Feedback_ID
Customer_ID (FK)
Order_ID (FK)
Order
Order_ID
Customer_ID (FK)
Chef_ID (FK)
BoxMoney_ID (FK)
OrderDetail
OrderDetail_ID
Order_ID (FK)
Product_ID (FK)
Reservation
Reservation_ID
Customer_ID (FK)
Table_ID (FK)
Customer
Customer_ID
User_ID (FK)
Product
Product_ID
Gategorey_ID (FK)
Chef
Chef_ID
User_ID (FK)
Table_
Table_ID
User_
User_ID
Login_ID (FK)
Waiter
Waiter_ID
User_ID (FK)
Gategorey
Gategorey_ID
Supplier
Supplier_ID
User_ID (FK)
Login
Login_ID
Stored
Stored_ID
Supplier_ID (FK)
RestManager_ID (FK)
RestaurantManager
RestManager_ID
User_ID (FK)
BoxMoney
BoxMoney_ID
RestManager_ID (FK)
InfoAboutRestaurant
Info_ID
RestManager_ID (FK)

packages

CRUD Procedure for table User_

```
create or replace package User_Package as
        procedure GetAllUsers;
        procedure CeateUsers(UuserType in varchar , Uusername in varchar , Uphone in varchar , Udate in date , Uemail in varchar,
                                UPass in varchar ,ULoginID in int);
        procedure UpdateUsers(Uid in int, UuserType in varchar , Uusername in varchar , Uphone in varchar , Udate in date ,
                                Uemail in varchar,UPass in varchar ,ULoginID in int);
        procedure DeleteUsers(Uid in int);

end User_Package;
```

CRUD Procedure for table login

```
create or replace package P_login as
      procedure CreateLoginInfo_LoginVerification(email in varchar2,pass in Varchar2);
```

Verifies the entered information before registering

```
select count(*) into LoginVerifica from User_ where User_Email=email And User_Pass=pass;
if LoginVerifica = 1 then

                nextLoginID:=LoginID_sequence.nextVal;
                insert into login(Login_ID,Login_Email,Login_Pass) values (nextLoginID,email,pass);
                Update User_ set User_LoginID=nextLoginID where User_Email=email And User_Pass=pass;
                select User_Name,User_Type into Uname,Utype from User_ where User_Email=email And User_Pass=pass And User_LoginID=nextLoginID;

                DBMS_OUTPUT.put_line('User Type : ' ||Utype || ' ~~~~ ' ||'User Name : '||Uname);
                DBMS_OUTPUT.put_line('User Email : ' ||email || ' ~~~~ ' ||'User Pass : '||pass);
else
          DBMS_OUTPUT.put_line('Wrong login information');
end if;
```

```
    procedure DeleteLoginInfo(logid in int);
    procedure GetLoginInfo;
end P_login;
```

Create or replace package P_Customert as

procedure MenuFood;

Show Generally:
CategoreyName , ProductPrice , ProductPrice

procedure MenuFoodByCategoreyName(gatName Categorey.Categorey_Name%type);

Show By Categorey Name:
CategoreyName , ProductPrice , ProductPrice

procedure MenuFoodByProductprice(lowestPrice IN product.product_price%TYPE, higherPrice IN product.product_price%TYPE);

Show By Product price:
CategoreyName , ProductPrice , ProductPrice

procedure SelectFoodFromMenu(CustomertID in int, productID in int,productAmount in int,OrderType in varchar);

If there is a customer order that has not yet been fulfilled and a 20 minute has not passed,The product is added on the same order
Otherwise, a new order will be opened

```
---------------------------------------------------------
select COUNT(*) into nextOrderID_Test
 from Order_
 where
Order_CustomerID= CustomertID  and
CURRENT_TIMESTAMP between Order_Data- interval '10' minute and Order_Data+ interval '10' minute and
 CustomerCheck='Failure to fulfill the order';
 ---------------------------------------------------------
```

procedure ViewCart_MyOrder(CustomertID in int,OrdeID in int);

Full order information is displayed

```
DBMS_OUTPUT.put('welcome '|| C_UserType);
DBMS_OUTPUT.put_line('.'|| C_UserName);
----------------------------------------------------------------
DBMS_OUTPUT.put_line('Order Data: '|| C_OrderData);
DBMS_OUTPUT.put_line('Order Type: '|| C_OrderType);
DBMS_OUTPUT.put_line('Product Name: '|| C_proName);
DBMS_OUTPUT.put_line('Product Price: '|| C_proprice);
DBMS_OUTPUT.put_line('Product Amount: '|| C_OrderDetail_proAmount);
 ----------------------------------------------------------------
DBMS_OUTPUT.put_line('Total Price :'||C_totalPrice);
```

procedure CustomerCheck_ExecutionOrder(CustomertID in int,OrdeID in int);

The order is executed by the customer after making sure that the balance available in the customer's account is greater than the value of the order

```
if AvailableBalance>=TotalPrice then

update Order_ SET CustomerCheck='Execution of the order'
where Order_ID=OrdeID and Order_CustomerID=CusTID;

end if;
```

```
procedure inser_FeedBack(FB_Rest in varchar,FB_Food in varchar,FB_Chef in varchar,FB_Waiter in varchar,FB_Note  in varchar,CustomerID in int,OrderID in int);
```

Create FeedBack

```
procedure GetAllCustomer;
procedure CreateCustomer(CavailableBalance in number , CuserId in int,CvisaNumber in varchar);
procedure UpdateCustomer(Cid in int , CavailableBalance in number , CuserId in int , CvisaNumber in varchar);
procedure DeleteCustomer(Cid in int);
```

CRUD Procedure for table Customert

```
end P_Customert;
```

```
Create or replace package  P_Chef as

    procedure  ApprovalOrder(OrdeID in Order_.Order_ID%type,ChefID in Chef.Chef_ID%type);
```

Acceptance of the order by the chef and storing the ID of the chef that is based on the preparation of the order

```
update Order_ set ChefCheck='Approval' ,Order_ChefID=ChefID
wher Order_ID=OrdeID ;
```

```
    procedure  DisapprovalOrder(OrdeID in Order_.Order_ID%type,ChefID in Chef.Chef_ID%type);
```

The chef refused the order and stored the ID of the chef who refused the order

```
update Order_ set ChefCheck='Disapproval',Order_ChefID=ChefID
wher Order_ID=OrdeID ;
```

```
    procedure  UpdateState_OrderIsInProgress(OrdeID in Order_.Order_ID%type);
```

Change the status of the order to 'Order is in progress'

```
update Order_ set Waiter_Chef_StateOrder='Order is in progress'
wher Order_ID=OrdeID ;
```

```
    procedure  UpdateState_OrderComplete(OrdeID in Order_.Order_ID%type);
```

Change the status of the order to 'Order Complete'

```
update Order_ set Waiter_Chef_StateOrder='Order Complete'
wher Order_ID=OrdeID ;
```

```
procedure GetAllChef;

procedure CreateChef(CSalary in number , CDate in date ,CuserId in int);

procedure UpdateChef(Cid in int ,CSalary in number , CDate in date , CuserId in int);

Procedure DeleteChef(Cid in int);
```

CRUD Procedure for table Chef

```
end  P_Chef;
```

Create or replace package P_Waiter as

procedure ViewReadyOrders;

It displays all the orders that the customer has executed the order and the chef approved the order and the order status has been modified by the chef to 'Order Complete' and the waiter has not yet given the order to a customer

```
Cursor C_ViewReadyOrders is
 SELECT
Order_.Order_ID ,
User_.User_Name,
product.product_name,product.product_price,
OrderDetail.OrderDetail_productAmount
FROM Customer
INNER JOIN User_
ON Customer.Customer_UserID=User_.User_ID
INNER JOIN Order_
ON Customer.Customer_ID=Order_.Order_CustomerID
INNER JOIN OrderDetail
ON Order_.Order_ID=OrderDetail.OrderDetail_OrderID
INNER JOIN product
ON OrderDetail.OrderDetail_productID=product.product_ID
WHERE  Order_.CustomerCheck='Execution of the order'
And Order_.ChefCheck='Approval'
And Order_.Waiter_Chef_StateOrder='Order Complete'
And Order_.GiveOrderToCustomer='On'
Order by Order_.Order_ID;


DBMS_OUTPUT.put_line('Order ID: '|| C_orderID || ' ~~~~~ ' || 'User Name: '|| C_UserName);
DBMS_OUTPUT.put_line('Product Name: '|| C_ProName || ' | ' || 'Product price:  '|| C_Proprice || ' | ' || 'Product Amount:  '|| C_ProAmount);
DBMS_OUTPUT.put_line('_____________________________________________');
```

procedure ViewCustomerFeedBack ;

It displays feedback regarding the food and any comments and the name of the customer and the ID of the order

```
DBMS_OUTPUT.put_line('Order ID: '|| C_OrderID || ' ~~~~ ' ||' User Name: '||C_UserName);
DBMS_OUTPUT.put_line('FeedBack For Food: '|| C_FB_Food);
DBMS_OUTPUT.put_line('FeedBack For Note: '||C_FB_Note);
DBMS_OUTPUT.put_line('---------------------');
```

procedure UpdateGiveOrderToCustomer (OrderID in Order_.Order_ID%TYPE);

Modifies the status of 'GiveOrderToCustomer' from No to Yes provided that the customer has executed the order and the chef approved the order and the order status was 'Order Complete'

```
UPDATE Order_ set GiveOrderToCustomer='Yes' where Order_ID=OrderID;
```

procedure GetAllWaiter;

procedure CreateWaiter(WSalary in number , WDateOfHiring in date , WuserId in int);

procedure UpdateWaiter(Wid in int ,WSalary in number , WDateOfHiring in date , WuserId in int);

Procedure DeleteWaiter(Wid in int);

CRUD Procedure for table Waiter

end P_Waiter;

```
Create or replace package  P_Supplier as

        procedure  ViewStored;
```

```
Displays all the items in the Stored

Cursor C_ViewStored is
Select *
From Stored_;

DBMS_OUTPUT.put_line('Item ID: '|| C_ItemID || ' ~~~~~ ' || 'Item Name: '|| C_ItemName || ' ~~~~~ ' || 'Amount Of Ttem Available: '|| C_AmountOfItemAvailable);
DBMS_OUTPUT.put_line('Item Expiration Date: '|| C_ItemExpirationDate );
DBMS_OUTPUT.put_line('Notes/Component About The Item: '|| C_Notes_ComponentAboutTheItem);
DBMS_OUTPUT.put_line('Restaurant Manager ID : '|| C_StResMangId);
DBMS_OUTPUT.put_line('Supplier ID: '|| C_StSupplierID);
```

```
        procedure UpdateItemInStore(I_ItemID IN INT, I_ItemName IN VARCHAR,I_ItemExpirationDate IN DATE,
                I_AmountOfItemAvailable IN VARCHAR ,I_Notes_ComponentAboutTheItem IN VARCHAR,I_StResMangId in int,I_StSupplierID in int);
```

```
Updates a specific item in the Store

Update Stored_ set
Item_Name=I_ItemName ,
ItemExpirationDate=I_ItemExpirationDate ,
AmountOfItemAvailable=I_AmountOfItemAvailable ,
Notes_ComponentAboutTheItem=I_Notes_ComponentAboutTheItem,
Stored_ResMangId=I_StResMangId,
Stored_SupplierID=I_StSupplierID
Where Item_ID=I_ItemID;
```

```
        procedure GetAllSupplier;
        Procedure CreateSupplier(ScontractSigningDate in date ,SuserID in int);
        Procedure UpdateSupplier(Supid in int, ScontractSigningDate in date ,SuserID in int);
        Procedure DeleteSupplier(Supid in int);
```

CRUD Procedure for table Supplier

```
end  P_Supplier;
```

```
Create or replace package  P_RestaurantManager as

    procedure  ViewStored;
```

Displays all the items in the Stored

```
Cursor C_ViewStored is
Select *
From Stored_;

DBMS_OUTPUT.put_line('Item ID: '|| C_ItemID || ' ~~~~~ ' || 'Item Name: '|| C_ItemName || ' ~~~~~ ' || 'Amount Of Ttem Available: '|| C_AmountOfItemAvailable);
DBMS_OUTPUT.put_line('Item Expiration Date: '|| C_ItemExpirationDate );
DBMS_OUTPUT.put_line('Notes/Component About The Item: '|| C_Notes_ComponentAboutTheItem);
DBMS_OUTPUT.put_line('Restaurant Manager ID : '|| C_StResMangId);
DBMS_OUTPUT.put_line('Supplier ID: '|| C_StSupplierID);
```

```
    procedure WeekSalesShow_MyBoxMoney(ManID in RestaurantManager.ResMan_ID%type);
```

Displays the total sales per user in the position of Restaurant Manager during the past week

```
select SUM(Product.Product_price*OrderDetail.OrderDetail_productAmount)
into l_TotalAmount
from Order_
inner join OrderDetail
on OrderDetail.OrderDetail_OrderID=Order_.Order_ID
inner join Product
on Product.Product_ID=OrderDetail.OrderDetail_ProductID
where Order_.BoxMoneyID=l_BoxMoneyID AND
Order_.CustomerCheck='Execution of the order' AND
Order_.ChefCheck='Approval' AND
Order_.Waiter_Chef_StateOrder ='Order Complete' AND
Order_.GiveOrderToCustomer='Yes' AND

Order_.Order_Data between CURRENT_TIMESTAMP- interval '7' Day and CURRENT_TIMESTAMP

Order By Order_.Order_ID;


DBMS_OUTPUT.put_line('WeekSalesShow_MyBoxMoney');
DBMS_OUTPUT.put_line('ResMan ID: '|| ManID ||' ~~~~ '||'User Name : '||l_UserName);
DBMS_OUTPUT.put_line('Total Amount : '|| l_TotalAmount );
```

Procedure UpdateInfoAboutRestaurant(L_InfoID IN INT,L_DateOfEstablishment IN DATE,L_NumberOfTables IN INT,L_OpeningTime IN InfoAboutRestaurant.OpeningTime%TYPE,L_ClosingTime IN InfoAboutRestaurant.ClosingTime%TYPE,L_NumberOfEmployees IN INT,L_TotalSalary IN NUMBER,RMangId in int)

Update information about the restaurant

```
Update   InfoAboutRestaurant
set
DateOfEstablishment=L_DateOfEstablishment,
NumberOfTables=L_NumberOfTables,
OpeningTime=L_OpeningTime,
ClosingTime=L_ClosingTime,
NumberOfEmployees=L_NumberOfEmployees,
TotalSalary=L_TotalSalary,
ResMangId=RMangId
Where Info_ID=L_InfoID;
```

procedure GetAllResturantManager;
procedure CreateResturantManager(RSalary in number, RuserId in int);
procedure UpdateResturantManager(Rid in int,RSalary in number, RuserId in int);
procedure DeleteResturantManager(Rid in int);

end P_RestaurantManager;

CRUD Procedure for table RestaurantManager

CRUD Procedure for table Category

```
create or replace Package Category_Package as
        procedure GetAllCategories;
        procedure CreateCategory(CcategoryName in varchar);
        procedure UpdateCategory(Cid in int , CcategoryName in varchar);
        procedure DeleteCategory(Cid in int);
end Category_Package;
```

CRUD Procedure for table Product

```
create or replace Package Product_Package as
        procedure getAllProduct;
        procedure CreateProduct(Pname in varchar, Pprice in int , PcategoryId in int);
        procedure UpdateProduct(Pid in int , Pname in varchar, Pprice in int , PcategoryId in int);
        procedure DeleteProduct(Pid in int);
end Product_Package;
```

CRUD Procedure for table Order_

```
create or replace package Oder_Package as

    procedure GetAllOders;
    procedure CreateOder(Odata in TIMESTAMP WITH LOCAL TIME ZONE ,Otype in varchar , OcutstomerId in int ,OchefId in int ,BoxMID in int);
    procedure UpdateOrder(
          OoderId in int ,
          Odata in TIMESTAMP WITH LOCAL TIME ZONE ,
          Otype in varchar ,
          OcustomerCheck in varchar ,
          OchefCheck in varchar ,
          Ochef_Waiter_Stateorder in varchar ,
          OcutstomerId in int ,
          OchefId in int,
          OgiveOrderToCustomer in varchar,
          BoxMID in int
          );
   procedure DeleteOrder(OoderId in int);

end Oder_Package;
```

CRUD Procedure for table OrderDetail

```
create or replace package OrderDetail_Package
as
    procedure GetAllOrderDetail;
    procedure CreateOrderDetail(OorderId in int ,OproductId in int ,OproductAmount in int);
    procedure UpdateOrderDetail(OrdDetID in int,OorderId in int ,OproductId in int ,OproductAmount in int);
    procedure DeleteOrderDetail(OrdDetID in int);

end OrderDetail_Package;
```

```
create or replace package P_CustomerFeedBack as

procedure CreateCustomerFeedBack(feedBackRestaurant in varchar2,feedBackFood in varchar2,feedBackChef in
varchar2,feedBackWaiter in varchar2,feedBackNote in varchar2,custID in int,ordID in int);

procedure UdateCustomerFeedBack(feedBackID in int,feedBackRestaurant in varchar2,feedBackFood in
varchar2,feedBackChef in varchar2,feedBackWaiter in varchar2,feedBackNote in varchar2,custID in int,ordID in int);

procedure DeleteCustomerFeedBack(feedBackID in int);
procedure GetAllCustomerFeedBack;

end P_CustomerFeedBack;
```

CRUD Procedure for table Customer_FeedBack

CRUD Procedure for table BoxMoney

```
create or replace package BoxMoney_Package as

  procedure GetAllBoxMoney;
  procedure CreateBoxMoney(Bopentime in TIMESTAMP WITH LOCAL TIME ZONE, Bclosingtime TIMESTAMP WITH LOCAL TIME ZONE,Bresmanid in int);
  procedure UpdateBoxMoney(Boxid in int, Bopentime in TIMESTAMP WITH LOCAL TIME ZONE, Bclosingtime TIMESTAMP WITH LOCAL TIME ZONE,Bresmanid in int);
  procedure DeleteBoxMoney(Boxid in int);

end BoxMoney_Package;
```

CRUD Procedure for table Table_

```
create or replace package Table_Package as

    procedure GetAllTables;
    procedure CreateTables(Tcode in varchar , Tstatus in varchar);
    procedure UpdateTables(Tid in int ,Tcode in varchar , Tstatus in varchar);
    procedure DeleteTables(Tid in int);

end Table_Package;
```

```
Create or replace package  P_Reservation as
     procedure  InquiriesAboutReservations(tabID in table_.table_ID%type, bookDate in reservation.bookingDate%type);
```

The time available for the selected table appears on the selected date

```
// bookingSession =  booking hour

SELECT COUNT(*)
INTO QueryResult
FROM table_
WHERE table_ID=tabID AND table_Status='Available' AND table_ID NOT IN
 (SELECT tableID FROM reservation WHERE trunc(bookingDate)=trunc(bookDate) AND bookingSession=BSession)
FETCH NEXT 1 ROWS ONLY;


if QueryResult !=0 then
DBMS_OUTPUT.put_line('Table ID : '|| tabID ||' ~~~~ ' || 'Booking Date : ' || bookDate ||' ~~~~ ' || 'Booking Session : ' || BSession);
DBMS_OUTPUT.put_line('_____________________________________________________________________________');
end if;
```

```
procedure UpdateTableStatus_NotAvailable(tabID in table_.table_ID%type);
```

Update the table status for Not Available

```
Update  table_ Set table_Status='Not available' where table_ID=tabID;
```

```
procedure UpdateTableStatus_Available(tabID in table_.table_ID%type);
```

Update the table status for Available

```
Update  table_ Set table_Status='Available' where table_ID=tabID;
```

```
procedure CreateReservation(
tabID in table_.table_ID%type,
CustID in Customer.Customer_ID%type,
booSession in reservation.bookingSession%type,
bookDate in reservation.bookingDate%type
);
procedure GetAllReservation;
procedure DeleteReservation(bookID in int);

end  P_Reservation;
```

CRUD Procedure for table Reservation

CRUD Procedure for table Stored

```
create or replace package Stored_Package as
        procedure GetAllStored;
        procedure CreateStored(Sname in varchar , Sexpiredate in date , SamounrOfitemAvailable in varchar ,
                               Snotes in varchar, StResMangId in int, StSupplierID in int);
        procedure UpdateStore(Sexpiredate in date , SamounrOfitemAvailable in varchar ,
                               Snotes in varchar,StResMangId in int,StSupplierID in int);
        procedure DeleteStored(StoredId in int);

end Stored_Package;
```

CRUD Procedure for table InfoAboutRestaurant

```
create or replace package InfoResturant_Package as
        procedure GetAllInfoResturant;
        procedure CreateInfoResturant (Rdate in date , Rnumberoftable in int,
                                       Ropentime in TIMESTAMP WITH LOCAL TIME ZONE ,
                                       Rclosetime in TIMESTAMP WITH LOCAL TIME ZONE ,
                                        Rnumberofemployee in int,RtotalSalary in number,RMangId in int);

        procedure UpdateInfoResturant (Rid in int, Rdate in date , Rnumberoftable in int,
                                       Ropentime in TIMESTAMP WITH LOCAL TIME ZONE ,
                                       Rclosetime in  TIMESTAMP WITH LOCAL TIME ZONE ,
                                       Rnumberofemployee in int,RtotalSalary in number,RMangId in int);
        procedure DeleteInfoResturant(Rid in int);

end InfoResturant_Package;
```

Triggers

- **Trigger are created for all tables**

```
Create table Audits(
audit_ID int GENERATED by default on null as IDENTITY primary key,
table_name varchar2(30),
transaction_name varchar2(50),
by_user varchar2(25),
transaction_Date TIMESTAMP WITH LOCAL TIME ZONE
);
```

\\E.g

```
Create or replace Trigger User_Audit
after insert or update or delete
on User_ for each row
Declare
L_transaction audits.transaction_name%type;
begin

    L_transaction := case
    when inserting then 'insert'
    when updating then 'Update'
    when deleting then 'Delete'
    end;

    insert into audits(table_name,transaction_name,by_user,transaction_Date)
    values('User_',L_transaction,USER,CURRENT_TIMESTAMP);

end;
```