DBMS Project (UCS310)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Restaurant Management System

Submitted to:

Dr. Jhilik Bhattacharya	2CS1

Submitted By:

Sudansh Rana	102217005
Saksham Dhiman	102217026
Avneesh Jarangal	102217029
Karan Bagadi	102217031

INDEX

Problem Statement

The current manual restaurant management system leads to inefficiencies in ordering, inventory tracking, and customer service. A database management system is needed to streamline these processes and provide accurate and timely information for effective decision-making

Overview

The Chef's Track is an innovative restaurant management system that leverages PL/SQL for its backend. The Diagrams and Flowcharts shown are made with the help of Quick DBD and combination of different designing tools.

The system offers a wide range of features, including a user-friendly menu display, streamlined order management, personalized customer profiles, tipping options, and much more. By automating processes such as inventory tracking and order management, the system allows for more accurate and timely decision-making.

The creation of customer profiles enables the restaurant to better understand its customers, leading to more personalized service. With its polished user interface and comprehensive features, The Chef's Track is the perfect solution for any restaurant looking to improve the customer experience and operational efficiency.

Requirements Analysis

To develop a successful Restaurant Management System, we need to analyse the requirements of the system. The following are the primary requirements for the system:

1. Menu Management: The system must have the ability to display the menu, including the items, their prices, and descriptions.

2. Customer Management: The system should allow the staff to create customer profiles and store their information, including their name, contact details, etc.

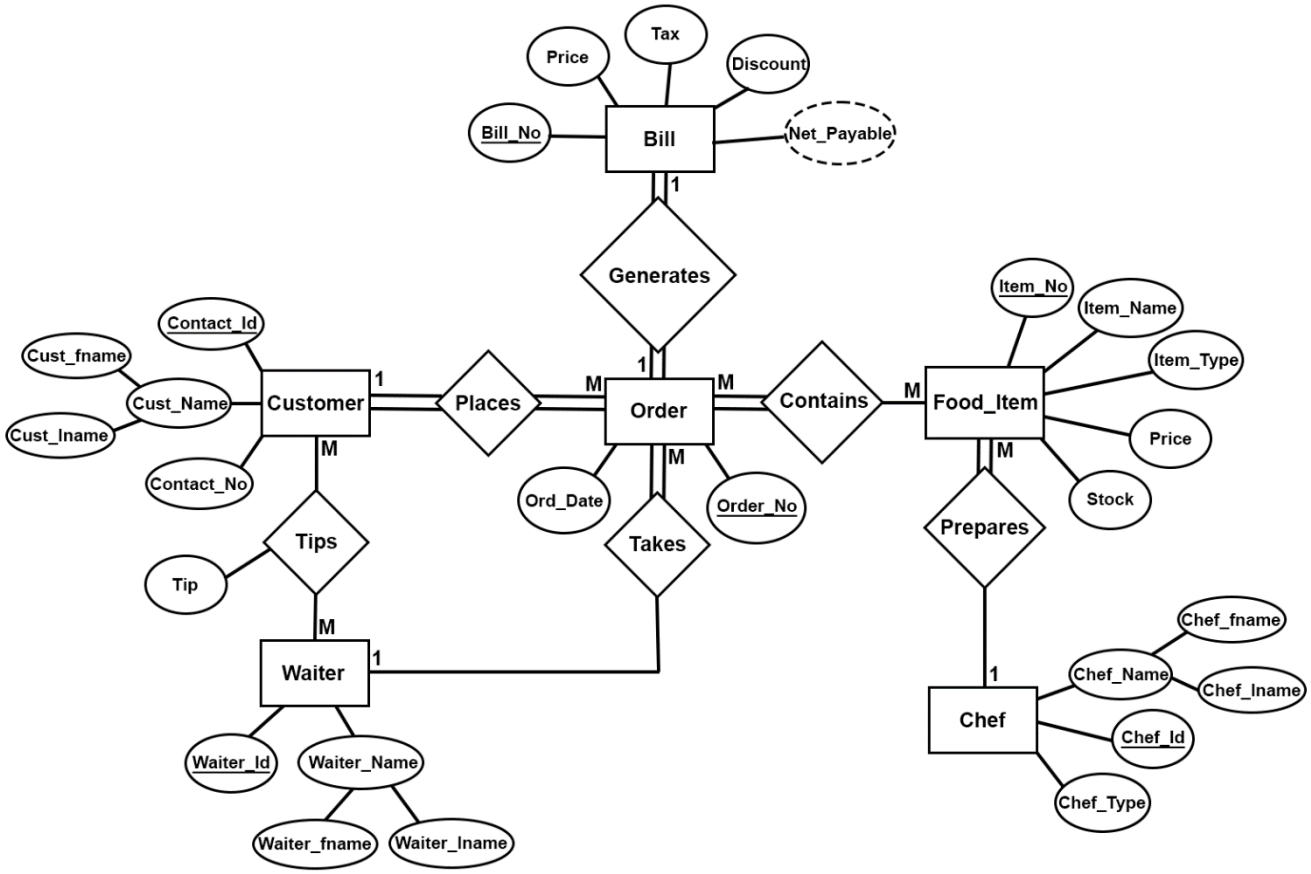
3. Order Management: The system should provide features to place orders and add more items to order.

4. Bill and Tips Management: The system should be able to generate bills and also allow customers to tip the waiter.

Software Requirements:

1. SQL
2. PL/SQL
3. Oracle Live SQL
4. Combination of different designing tools to create ER-Diagram

ER Diagram


Tax
Price
Discount
Bill_No
Bill
Net_Payable
1
Generates
Item_No
Item_Name
Item_Type
Price
Stock
Contact_Id
Cust_fname
Cust_Name
Cust_lname
Contact_No
Customer
1
Places
M
1
Order
M
Contains
M
Food_Item
M
M
Ord_Date
Order_No
Takes
Prepares
Tips
Tip
M
Waiter
1
Chef_fname
Chef_Name
Chef_lname
1
Chef
Chef_Id
Chef_Type
Waiter_Id
Waiter_Name
Waiter_fname
Waiter_lname

ER Diagram To Table

Relation 'Places'

Customer:- cust_id , cust_fname , cust_lname, contact_no
Orders:- ord_no , ord_date , cust_id (FK)

Relation 'Takes'

Orders:- ord_no , ord_date , waiter_id (FK)
Waiter:- waiter_id , waiter_name , waiter_lname

Relation 'Tips'

Customer:- cust_id , cust_fname , cust_lname , contact_no
Waiter:- waiter_id , waiter_fname , waiter_lname
Tips:- cust_id (FK) , waiter_id (FK) , tip

Relation 'Prepares'

Food:- item_no , item_name , item_type , item_price , item_stock ,chef_id (FK)
Chef:- chef_id , chef_fname , chef_lname , chef_type

Relation 'Generates'

Orders:- ord_no , ord_date
Bill:- bill_no , tot_price , tax , discount , net_payable , ord_no (FK)

Relation 'Contains'

Food:- item_no , item_name , item_type , item_price , item_stock
Contains:- item_no (FK) , ord_no (FK)
Orders:- ord_no , ord_date

Normalization

Customer Table (Already in 3NF):

cust_id	cust_fname	cust_lname	contact_no

Waiter Table (Already in 3NF)

waiter_id	waiter_fname	waiter_lname

Tips Table (Already in 3NF)

waiter_id (FK)	cust_id (FK)	tip

Orders Table (Already in 3NF)

ord_no	ord_date	cust_id (FK)	waiter_id (FK)

Chef Table (Already in 3NF)

chef_id	chef_fname	chef_lname	chef_type

Food Table (in 2NF)

As item_no → item_type and item_type → chef_id

item_no	item_name	item_type	item_price	item_stock	chef_id (FK)

Breaking it into further tables Food

Table:

item_no	item_name	item_type	item_price	item_stock

Prepares Table:

item_type	chef_id (FK)

Contains Table (Already in 3NF)

ord_no (FK)	item_no (FK)

Bill Table (Already in 3NF)

bill_no	tot_price	tax	discount	net_payable	ord_no (FK)

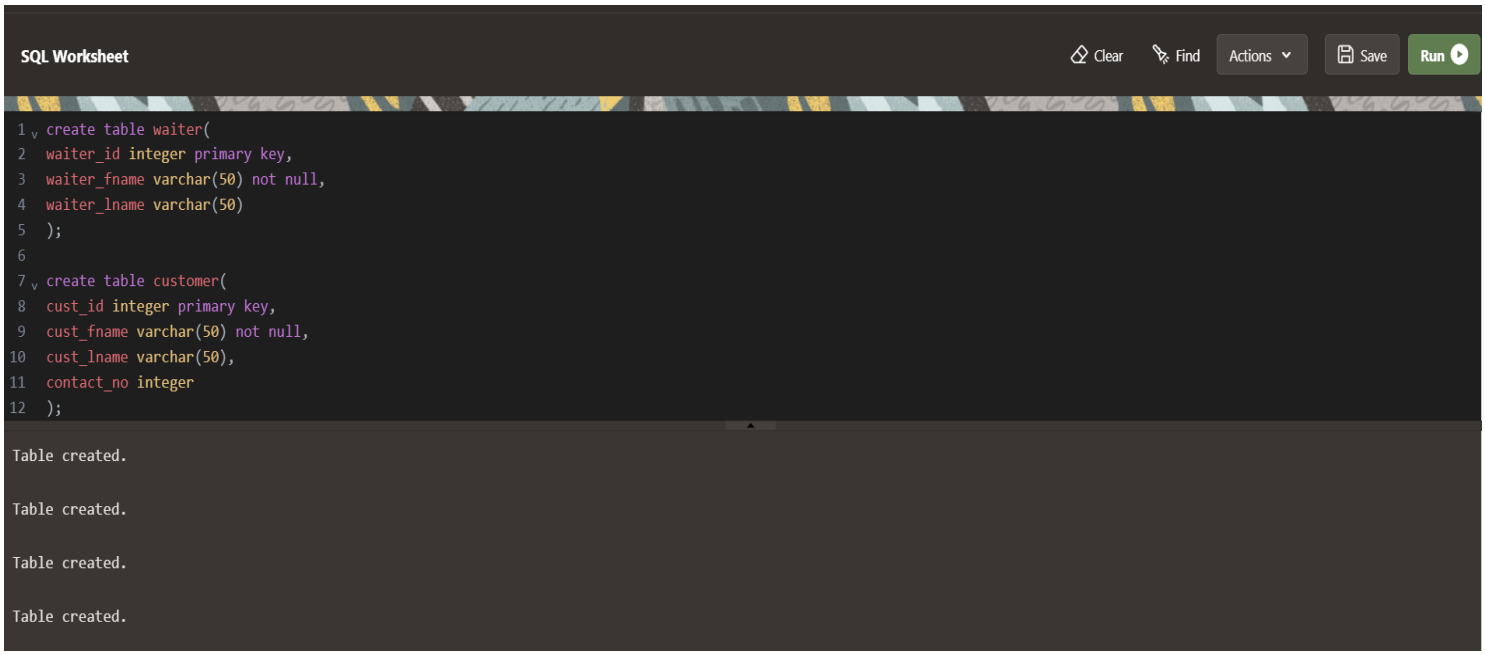
SQL Code

Creation of Tables

```
create table waiter(
waiter_id integer primary key,
waiter_fname varchar(50) not null,
waiter_lname varchar(50)
);

create table customer(
cust_id integer primary key,
cust_fname varchar(50) not null,
cust_lname varchar(50),
contact_no integer
);

create table tips(
waiter_id integer references waiter(waiter_id),
cust_id integer references customer(cust_id),
tips integer
);
```

```
create table orders(
ord_no integer primary key,
rd_date date not null,
cust_id integer references customer(cust_id),
waiter_id integer references waiter(waiter_id)
);

create table chef(
chef_id integer primary key,
chef_fname varchar(50) not null,
chef_lname varchar(50),
chef_type varchar(50) not null
);

create table food(
item_no integer primary key,
item_name varchar(50) not null,
item_type varchar(50) not null,
item_price integer not null,
item_stock integer
);

create table contains(
ord_no integer references orders(ord_no),
item_no integer references food(item_no)
);
```

```
create table prepares(
item_type varchar(50) primary key,
chef_id integer references chef(chef_id)
);

create table bill(
bill_no integer primary key,
total_price integer not null,
tax float default 5,
discount integer default 0,
net_payable float as (total_price+(total_price*tax/100)-(total_price*discount/100)),
ord_no integer references orders(ord_no)
);
```



Insertion Statements:

insert into waiter values(1, 'Raj', 'Patel');

insert into waiter values(2, 'Amit', 'Kumar');

insert into waiter values(3, 'Priya', 'Sharma');

select * from waiter

SQL Worksheet

```
insert into waiter values(1, 'Raj', 'Patel');
insert into waiter values(2, 'Amit', 'Kumar');
insert into waiter values(3, 'Priya', 'Sharma');
select * from waiter
```

WAITER_ID	WAITER_FNAME	WAITER_LNAME
1	Raj	Patel
2	Amit	Kumar
3	Priya	Sharma

insert into customer values (1, 'Aarav', 'Gupta', 9876543210);

insert into customer values (2, 'Neha', 'Patil', 8765432109);

insert into customer values (3, 'Vivek', 'Singh', 7654321098);

insert into customer values (4, 'Sneha', 'Sharma', 6543210987);

select * from customer

SQL Worksheet

```
insert into customer values (1, 'Aarav', 'Gupta', 9876543210);
insert into customer values (2, 'Neha', 'Patil', 8765432109);
insert into customer values (3, 'Vivek', 'Singh', 7654321098);
insert into customer values (4, 'Sneha', 'Sharma', 6543210987);
select * from customer
```

CUST_ID	CUST_FNAME	CUST_LNAME	CONTACT_NO
1	Aarav	Gupta	9876543210
2	Neha	Patil	8765432109
3	Vivek	Singh	7654321098
4	Sneha	Sharma	6543210987

insert into orders values (1, '2-APR-2024', 1, 1);

insert into orders values (2, '3-APR-2024', 2, 2);

insert into orders values (3, '4-APR-2024', 3, 2);

select * from orders

SQL Worksheet

Clear Find Actions Save Run

```
insert into orders values (1, '2-APR-2024', 1, 1);
insert into orders values (2, '3-APR-2024', 2, 2);
insert into orders values (3, '4-APR-2024', 3, 2);
select * from orders
```

ORD_NO	RD_DATE	CUST_ID	WAITER_ID
1	02-APR-24	1	1
2	03-APR-24	2	2
3	04-APR-24	3	2

insert into chef values (1, 'Vikram', 'Sharma', 'Indian Cuisine');

insert into chef values (2, 'Priya', 'Kaur', 'Tandoori Specialties');

insert into chef values (3, 'Rajesh', 'Patel', 'Vegetarian Delights');

select * from chef

SQL Worksheet

Clear Find Actions Save Run

```
insert into chef values (1, 'Vikram', 'Sharma', 'Indian Cuisine');
insert into chef values (2, 'Priya', 'Kaur', 'Tandoori Specialties');
insert into chef values (3, 'Rajesh', 'Patel', 'Vegetarian Delights');
select * from chef
```

CHEF_ID	CHEF_FNAME	CHEF_LNAME	CHEF_TYPE
1	Vikram	Sharma	Indian Cuisine
2	Priya	Kaur	Tandoori Specialties
3	Rajesh	Patel	Vegetarian Delights

insert into prepares values('Biryani', 1);

insert into prepares values('Main Course', 2);

insert into prepares values('Vegetarian', 3);

insert into prepares values('Bread', 3);

select * from prepares

SQL Worksheet

Clear Find Actions Save Run

```
insert into prepares values('Biryani', 1);
insert into prepares values('Main Course', 2);
insert into prepares values('Vegetarian', 3);
insert into prepares values('Bread', 3);
select * from prepares
```

ITEM_TYPE	CHEF_ID
Biryani	1
Main Course	2
Vegetarian	3
Bread	3

insert into food values (1, 'Chicken Biryani', 'Biryani', 250, 50);

insert into food values (2, 'Butter Chicken', 'Main Course', 300, 30);

insert into food values (3, 'Palak Paneer', 'Vegetarian', 200, 40);

insert into food values (4, 'Naan', 'Bread', 50, 100);

select * from food

SQL Worksheet

Clear Find Actions Save Run

```
insert into food values (1, 'Chicken Biryani', 'Biryani', 250, 50);
insert into food values (2, 'Butter Chicken', 'Main Course', 300, 30);
insert into food values (3, 'Palak Paneer', 'Vegetarian', 200, 40);
insert into food values (4, 'Naan', 'Bread', 50, 100);
select * from food
```

ITEM_NO	ITEM_NAME	ITEM_TYPE	ITEM_PRICE	ITEM_STOCK
1	Chicken Biryani	Biryani	250	50
2	Butter Chicken	Main Course	300	30
3	Palak Paneer	Vegetarian	200	40
4	Naan	Bread	50	100

insert into contains values(1,1);

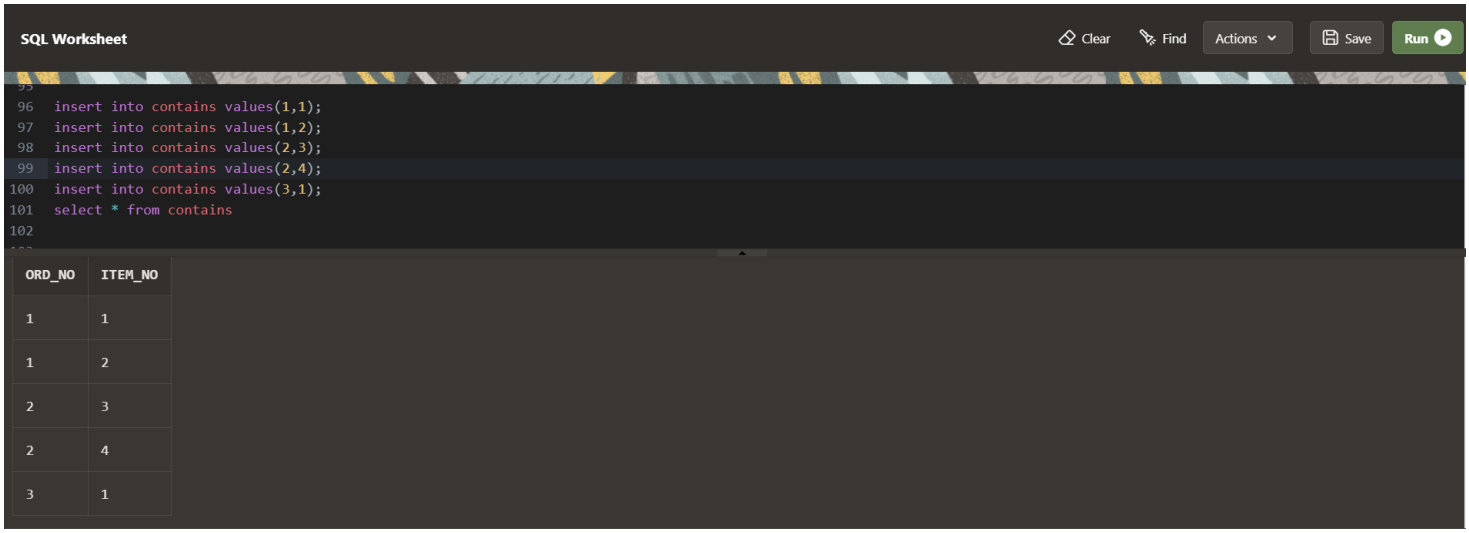
insert into contains values(1,2);

insert into contains values(2,3);

insert into contains values(2,4);

insert into contains values(3,1);

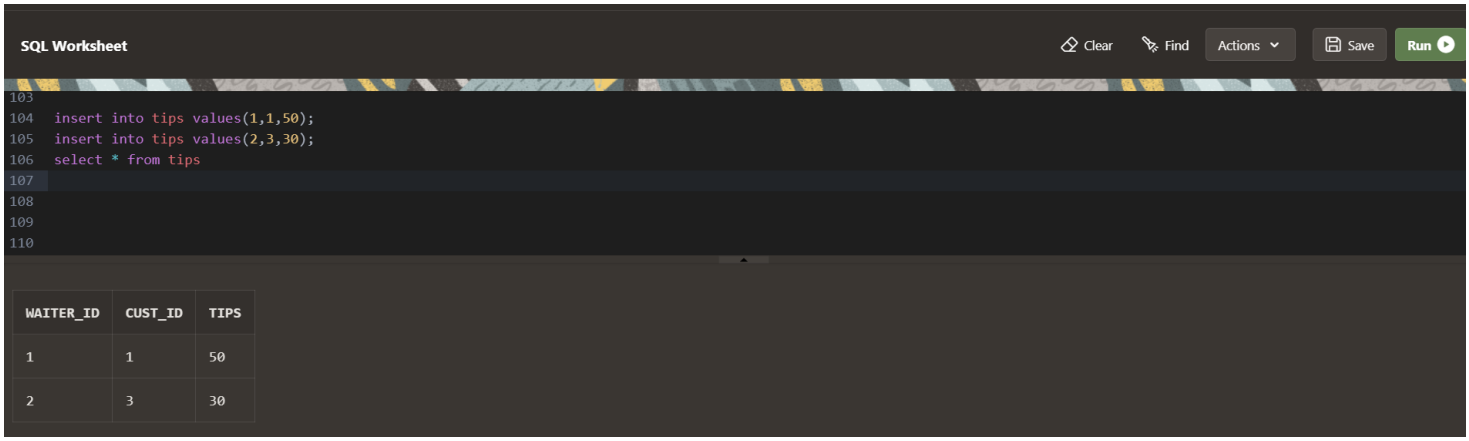
select * from contains



insert into tips values(1,1,50);

insert into tips values(2,3,30);

select * from tips



PL/SQL

1. Show_Menu Procedure:
   Displays the items and their Prices Using a Cursor.

```
declare
cursor c1 is select item_name,item_price from food;
rec1 c1%rowtype;
procedure show_menu is
begin
open c1;
loop
    fetch c1 into rec1;
    exit when c1%notfound;
    dbms_output.put_line('Item: '||rec1.item_name||' Pirce: $'||rec1.item_price);
end loop;
close c1;
end;

begin
show_menu;
end;
```

```
Statement processed.
Item: Chicken Biryani Pirce: $250
Item: Butter Chicken Pirce: $300
Item: Palak Paneer Pirce: $200
Item: Naan Pirce: $50
```

2. Get_Cust_Id Function:
   If a customer already exists then returns the existing ID else create a new customer and return the new ID

```
declare
id integer;
ifExists integer:=0;
function get_cust_id(fname in varchar,lname in varchar,contact in integer,wait_id in integer)return number as
begin
    select count(*) into ifExists from customer where cust_fname=fname and cust_lname=lname and contact_no=contact;
    if ifExists>0 then
        select cust_id into id from customer where cust_fname=fname and cust_lname=lname and contact_no=contact;
        return (id);
    else
        select count(*)+1 into id from customer;
        insert into customer values(id,fname,lname,contact);
        insert into tips(waiter_id,cust_id) values(wait_id,id);
        return (id);
    end if;
end;
begin
id:=get_cust_id('Neha','Patil',8765432109,2);
dbms_output.put_line('Customer id is '||id);
end;
```

```
Statement processed.
Customer id is 2
```

3. Placing Order
   a. In_Stock Trigger:
      Before inserting the food items in the order, it checks if the items are in stock if they are not in stock it will raise an error.

   b. After_Order Trigger:
      After inserting the food items in the order, it updates the stock of the items and decreases them accordingly.

   c. Place_Order Function:
      This function inserts the items in the form of an array of item_no in the order and returns the order_no to the customer.

```oracle_plsql


create or replace trigger in_stocks
before insert on contains for each row
declare
    stock integer;
begin
    select item_stock into stock from food where food.item_no= :new.item_no;
    if stock=0 then raise_application_error(-20000,'Out of Stock');
    else dbms_output.put_line('In stock');
    end if;
end;

create or replace trigger after_order
after insert on contains for each row
begin
    update food set item_stock=item_stock-1 where item_no = :new.item_no;
end;


```

```
Trigger created.
```

```oracle
declare
type num_array is varray(50) of integer;
items num_array;
order_no integer;
function place_order(id in integer,items in num_array,wait_id in number) return integer is
begin
    select count(*)+1 into order_no from orders;
    insert into orders values(order_no,sysdate,id,wait_id);
    for i in 1..items.count loop
        insert into contains values(order_no,items(i));
    end loop;
    return (order_no);
end;

begin
items:=num_array(1,2);
order_no:=place_order(4,items,3);
dbms_output.put_line('Order No: '||order_no);
end;
```

```
Statement processed.
In stock
In stock
Order No: 4
```

4. Generating Bill
   a. Display_Bill Trigger:
      It displays the bill along with the bill_no, ord_no, items, price, total price, discount, tax, and the net_payable amount to the customer.

   b. Generate_Bill Procedure:
      It takes in the values order_no and any discount value and inserts the given values into the bill table.

```oracle
create or replace trigger display_bill
after insert on bill for each row
begin
    dbms_output.put_line('Total Price: '||:new.total_price);
    dbms_output.put_line('Tax: '||:new.tax);
    dbms_output.put_line('Discount: '||:new.discount);
    dbms_output.put_line('Net Payable Amount: '||:new.net_payable);
end;
```

```
Trigger created.
```

```oracle
declare
cursor c2(n integer) is
select f.item_no,f.item_name,f.item_price,c.ord_no from food f,contains c where f.item_no=c.item_no and c.ord_no=n;
rec2 c2%rowtype;
total integer:=0;
b_no integer;
procedure generate_bill(order_no in integer,disc in float)is
begin
open c2(order_no);
select count(*)+1 into b_no from bill;
dbms_output.put_line('Bill No: '||b_no||'Order_no: '||order_no);
loop
fetch c2 into rec2;
exit when c2%notfound;
dbms_output.put_line('Item: '||rec2.item_name||' Price: $'||rec2.item_price);
total:=total+rec2.item_price;
end loop;
insert into bill (bill_no,total_price,discount,ord_no)values(b_no,total,disc,order_no);
end;
begin
generate_bill(4,0);
end;
```

```
Statement processed.
Bill No: 1Order_no: 4
Item: Chicken Biryani Price: $250
Item: Butter Chicken Price: $300
Item: Palak Paneer Price: $200
Total Price: 750
Tax: 5
Discount: 0
Net Payable Amount: 787.5
```

5. Tips
   a. Give_Tip Procedure:
      It takes in the value of the tip given by the customer to the denoted waiter.

   b. Display_Waiter_Tip Procedure:
      It displays the total tips collected by a specific waiter.

```
declare
procedure give_tip(id in integer,wait_id in integer,t in integer) is
begin
insert into tips values(wait_id,id,t);
dbms_output.put_line('Waiter '||wait_id||' Received $'||t||' tip' );
end;

begin
give_tip(4,3,10);
end;
```

```
Statement processed.
Waiter 3 Received $10 tip
```

```
declare
tot integer;
procedure display_waiter_tip(wait_id in integer) is
begin
select sum(tips) into tot from tips where waiter_id=wait_id;
dbms_output.put_line('Total tip for waiter id '||wait_id||' is $'||tot);
end;

begin
display_waiter_tip(3);
end;
```

```
Statement processed.
Total tip for waiter id 3 is $10
```

Conclusion

To conclude,
The development of a Restaurant Management System using SQL and PL/SQL offers numerous benefits to the hospitality industry. With its robust and scalable features, the system can handle a high volume of transactions and users, integrate with third-party systems, and comply with relevant regulations and standards.

The user-friendly interface makes it easy for restaurant staff to navigate and perform their tasks, reducing errors and providing better services to customers. By considering system requirements, external requirements, and hardware requirements, the Restaurant Management System can be optimized to perform optimally and securely. Overall, investing in a Restaurant Management System using SQL and PL/SQL is a wise decision for restaurants looking to streamline their operations, improve efficiency, and enhance the customer experience.

References

Websites –

www.youtube.com/parteekBhatia

https://app.creately.com/d/start/dashboard

Books –

1. Fundamentals of database systems (Ramez Elmsari, Shamkant B.Navathe)
2. Database System Concepts (Avi Silberschatz · Henry F.Korth · S. Sudarshan)