DBMS MINI PROJECT

RESTAURANT INVENTORY MANAGEMENT AND BILLING SYSTEM

Name	SRN
SINCAHANA K	PES1UG20EC271
YOGITHA H K	PES1UG20EC277
ANKIT LONI	PES1UG20EC286
POORVI RADDI	PES1UG20EC318

SEM – 6
SECTION – E 'ECE'

ABSTRACT

Restaurants always have the challenge in maintaining records , these records include the chef details the logistics details the supplier details and mainly the transaction details based on which important decisions can be made this project helps in digitizing and storing the data in a well modelled relational database

Powerful quires can be used to pull out important data like a particular supplier who supplies a particular ingredient and all the ingredients required for a particular dish and chefs with particular specialties

We have implemented a function that will calculate the total bill of a particular customer based on what all meals and the quantity he has ordered

A easy to use GUI has been implemented that uses the following python libraries

- **Tkinter**

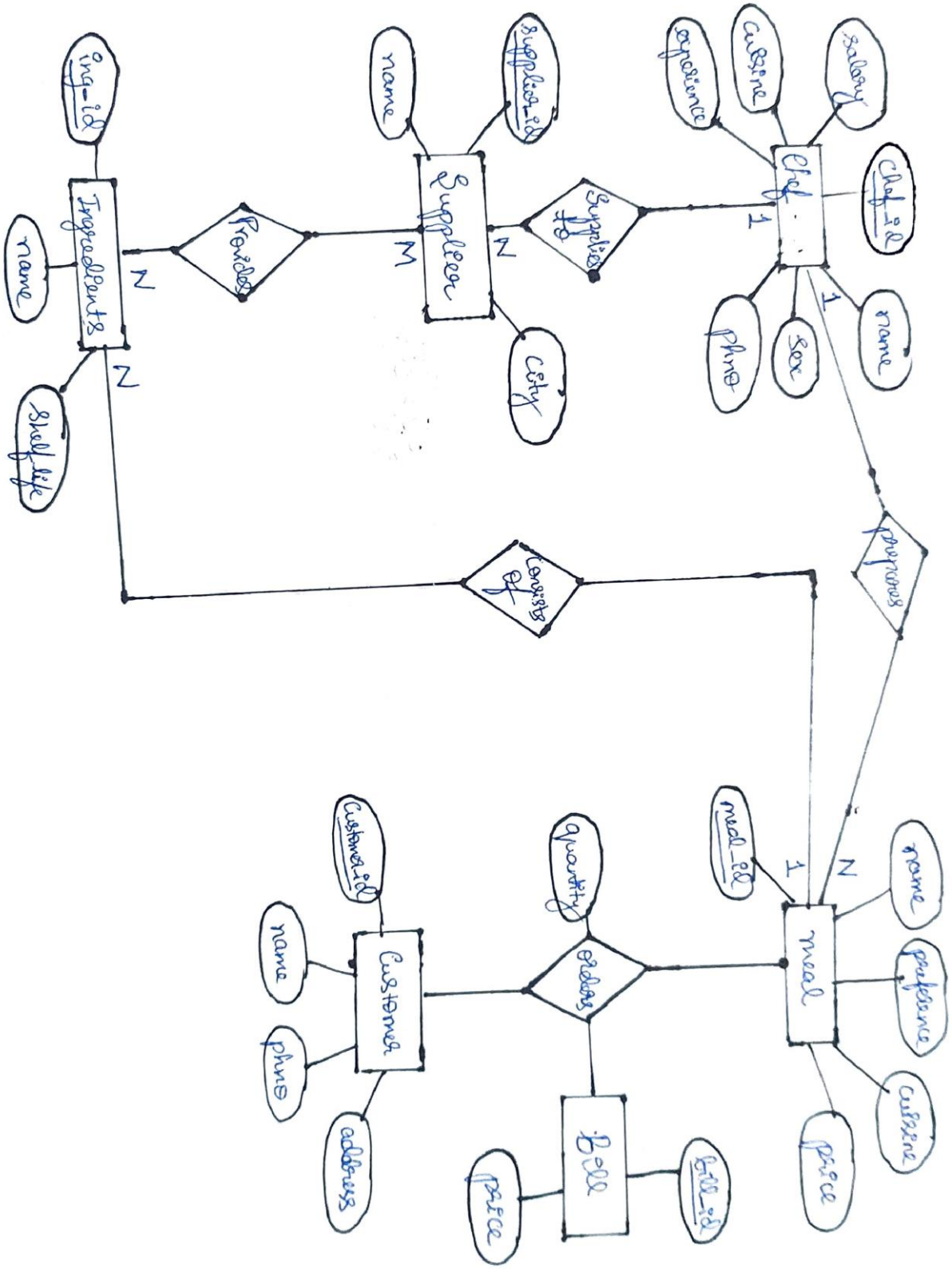
Used to create the GUI

- **My SQL connector**

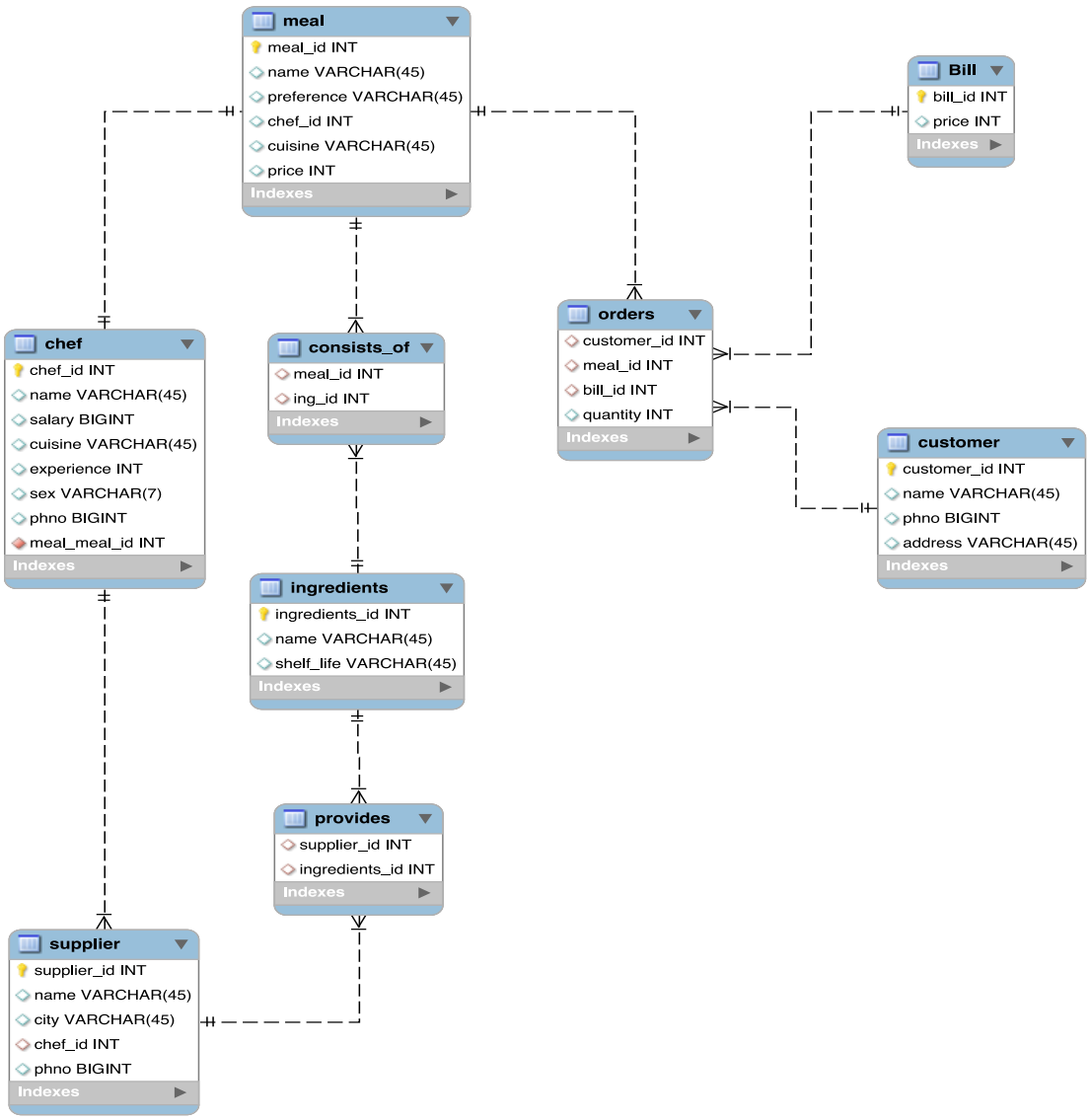
Used to connect to the Database

This GUI helps in taking the orders and generating and storing the transaction even the customer details can be added

ER DIAGRAM


Chef
Chef_id
salary
cuisine
experience
name
sex
phone
1
1
Supplies to
Supplier
Supplier_id
name
city
N
M
Provides
Ingredients
ing_id
name
shelf life
N
N
Consists of
prepares
meal
meal_id
name
preference
cuisine
price
1
N
orders
quantity
Customer
Customer_id
name
phone
address
Bill
bill_id
price

Relation Schema



DDL Commands to create tables

Creating DATABASE

CREATE DATABASE restaurant;

Creating TABELS

```
CREATE TABLE `Bill` (
  `bill_id` int NOT NULL,
  `price` int DEFAULT NULL,
  PRIMARY KEY (`bill_id`)
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**bill_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**price**	int(11)			Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `chef` (
  `chef_id` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `salary` bigint DEFAULT NULL,
  `cuisine` varchar(45) DEFAULT NULL,
  `experience` int DEFAULT NULL,
  `sex` varchar(7) DEFAULT NULL,
  `phno` bigint DEFAULT NULL,
  PRIMARY KEY (`chef_id`)
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**chef_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**name**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	3	**salary**	bigint(20)			Yes	*NULL*			Change	Drop	More
☐	4	**cuisine**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	5	**experience**	int(11)			Yes	*NULL*			Change	Drop	More
☐	6	**sex**	varchar(7)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	7	**phno**	bigint(20)			Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `consists of` (
  `meal_id` int DEFAULT NULL,
  `ing_id` int DEFAULT NULL,
  KEY `meal_idx` (`meal_id`),
  KEY `ing_idx` (`ing_id`),
  CONSTRAINT `ing` FOREIGN KEY (`ing_id`) REFERENCES `ingredients` (`ingredients_id`) ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT `meal` FOREIGN KEY (`meal_id`) REFERENCES `meal` (`meal_id`) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**meal_id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	2	**ing_id**	int(11)			Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `customer` (
  `customer_id` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `phno` bigint DEFAULT NULL,
  `address` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`customer_id`));
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**customer_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**name**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	3	**phno**	bigint(20)			Yes	*NULL*			Change	Drop	More
☐	4	**address**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `ingredients` (
  `ingredients_id` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `shelf_life` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ingredients_id`)
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**ingredients_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**name**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	3	**shelf_life**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `meal` (
  `meal_id` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `preference` varchar(45) DEFAULT NULL,
  `chef_id` int DEFAULT NULL,
  `cuisine` varchar(45) DEFAULT NULL,
  `price` int DEFAULT NULL,
  PRIMARY KEY (`meal_id`)
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**ingredients_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**name**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	3	**shelf_life**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `orders` (
  `customer_id` int DEFAULT NULL,
  `meal_id` int DEFAULT NULL,
  `bill_id` int DEFAULT NULL,
  `quantity` int DEFAULT '0',
  KEY `cust_idx` (`customer_id`),
  KEY `bill_idx` (`bill_id`),
  KEY `meal_idx` (`meal_id`),
  CONSTRAINT `bill` FOREIGN KEY (`bill_id`) REFERENCES `Bill` (`bill_id`),
  CONSTRAINT `cust` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`customer_id`)
ON UPDATE CASCADE,
  CONSTRAINT `meals` FOREIGN KEY (`meal_id`) REFERENCES `meal` (`meal_id`)
) ;
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**customer_id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	2	**meal_id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	3	**bill_id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	4	**quantity**	int(11)			Yes	0			Change	Drop	More

```
CREATE TABLE `provides` (
  `supplier id` int DEFAULT NULL,
  `ingredients id` int DEFAULT NULL,
  KEY `supp_idx` (`supplier id`),
  KEY `ing_idx` (`ingredients id`),
  CONSTRAINT `ings` FOREIGN KEY (`ingredients id`) REFERENCES `ingredients`
(`ingredients_id`),
  CONSTRAINT `supp` FOREIGN KEY (`supplier id`) REFERENCES `supplier` (`supplier_id`)
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**supplier id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	2	**ingredients id**	int(11)			Yes	*NULL*			Change	Drop	More

```
CREATE TABLE `supplier` (
  `supplier_id` int NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `city` varchar(45) DEFAULT NULL,
  `chef_id` int DEFAULT NULL,
  `phno` bigint DEFAULT NULL,
  PRIMARY KEY (`supplier_id`),
  KEY `chef_id_idx` (`chef_id`),
  CONSTRAINT `chef_supplier` FOREIGN KEY (`chef_id`) REFERENCES `chef` (`chef_id`) ON
DELETE SET NULL ON UPDATE CASCADE
);
```

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
☐	1	**supplier_id**	int(11)			No	*None*			Change	Drop	More
☐	2	**name**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	3	**city**	varchar(45)	utf8mb4_general_ci		Yes	*NULL*			Change	Drop	More
☐	4	**chef_id**	int(11)			Yes	*NULL*			Change	Drop	More
☐	5	**phno**	bigint(20)			Yes	*NULL*			Change	Drop	More

All Tables

Tables_in_project
bill
chef
consists of
customer
ingredients
meal
orders
provides
supplier

Populating values to the tables

INSERT INTO `chef` VALUES
(101,'Sunil',45000,'chinese',3,'male',7895672341),(102,'Rahul',30000,'south indian',2,'male',6783458761),(103,'Sonali',60000,'italian',5,'female',4672874671),(104,'Ratan',50000,'northindian',4,'male',7491089563),(105,'Reshma',45000,'lebanese',4,'female',9284820818),(106,'priya',45000,'chinese',4,'female',6284673829),(107,'prakash',80000,'italian',6,'male',7295728472),(108,'tom',60000,'american',3,'male',8629852983),(109,'divya',80000,'desserts',4,'female',7436372366);

INSERT INTO `consists_of` VALUES
(601,201),(601,201),(601,206),(601,207),(601,204),(601,210),(602,201),(602,219),(602,202),(602,204),(602,210),(602,207),(602,206),(602,208),(602,211),(603,217),(603,220),(603,212),(603,209),(604,203),(604,204),(604,210),(604,214),(604,216),(605,201),(605,211),(606,204),(606,210),(606,213),(606,216),(607,203),(607,215),(608,203),(608,215),(609,202),(609,204),(609,210),(609,216),(609,206),(610,204),(610,206),(610,210),(611,202),(611,207),(611,210),(611,216),(611,206),(612,208),(612,205),(612,218),(613,219),(613,206),(613,211),(613,204),(614,201),(614,217),(614,219),(614,209),(615,201),(615,208),(615,209),(615,218);

INSERT INTO `customer` VALUES (3001,'harsha',8296444590,'31/2 RT road Bangalore'),(3002,'chandan',2345287492,'21/34 MR road Bangalore'),(3003,'atharv',5573948930,'39/21 CT road Bangalore'),(3004,'puneeth',8757294759,'21/32 MM road Mysore'),(3005,'ishaan',8583957602,'90/11 KR road Tumkur'),(3006,'isha',7391257295,'21//33 LT road Davangere'),(3007,'ram',7382957482,'43/22 RR road

Bangalore'),(3008,'geetha',8147483028,'56/5 MK road
Bangalore'),(3009,'suresh',8472857692,'2/2 ML street
Dharwad'),(3010,'tarun',74138074102,'20/11 KR road
Bangalore'),(3011,'vishak',9843213478,'56/32 MP street
mumbai'),(3012,'roopa',8731492812,'32/45 TN street Chennai');

INSERT INTO `ingredients` VALUES (201,'flour','1 year'),(202,'noodles','1 year'),(203,'rice','2 years'),(204,'vegetables','1 week'),(205,'fruits','1 week'),(206,'sauces','10 months'),(207,'cheese','2 months'),(208,'milk','1 week'),(209,'sugar','1 year'),(210,'spices','2 years'),(211,'butter','1 month'),(212,'ice cream','1 month'),(213,'chicken','1 week'),(214,'mutton','1 week'),(215,'lentils','2 years'),(216,'greens','4 days'),(217,'chocholates','5 months'),(218,'eggs','2 weeks'),(219,'bread','3 days'),(220,'biscuits ','1 month');

INSERT INTO `meal` VALUES
(601,'pizza','veg',103,'italian',450),(602,'pasta','veg',107,'italian',350),(603,'chocolate milkshake','veg',109,'desserts',200),(604,'biryani','non veg',104,'north indian',400),(605,'roti','veg',104,'north indian',100),(606,'curry','non veg',104,'north indian',200),(607,'idly','veg',102,'south indian',100),(608,'dosa','veg',102,'south indian',150),(609,'noodles','veg',101,'chinese',300),(610,'manchurian','veg',106,'chinese',280),(611,'mac and cheese','veg',108,'american',370),(612,'panna cota','veg',107,'italian',260),(613,'sandwich ','veg',103,'american',250),(614,'brownies','egg',109,'desserts',180),(615,'cake','egg',109,'desserts',180);

INSERT INTO `provides` VALUES
(402,202),(401,201),(402,202),(401,203),(402,202),(403,204),(402,205),(411,206),(430,207),(404,208),(406,209),(409,210),(407,211),(404,212),(408,213),(407,214),(412,215),(411,216),(411,218),(416,219),(415,220),(420,202),(423,204),(424,207),(426,208),(427,209),(427,210),(430,211),(421,212),(411,213),(421,214),(412,215),(406,216),(409,217),(410,218),(411,219),(412,220),(413,212),(414,201),(415,220),(416,211),(417,213),(421,203),(422,211),(405,206),(402,209);

INSERT INTO `supplier` VALUES
(401,'yashas','bangalore',102,7258492854),(402,'manas','chennai',101,6928648382),(403,'vineeth','delhi',104,9887243873),(404,'pankaj','mumbai',106,9827842729),(405,'ankit','pune',107,4242488952),(406,'prajwal','hyderabad',103,4298635928),(407,'raghu','bangalore',104,9273649272),(408,'bhuvan','vizag',105,8237842703),(409,'bushan','mysore',108,837401740),(410,'barath','mysore',109,3413435351),(411,'bhanu','delhi',101,2455243453),(412,'bushan','bangalore',103,5325252354),(413,'prateek','delhi',102,3525523524),(414,'pavan','chennai',105,2352543364),(415,'ritviz','pune',104,3523523525),(416,'ganesh','bangalore',103,4234232354),(417,'jhon','mumbai',106,5252523543),(418,'bhasker','delhi',107,9274818246),(419,'vijay','delhi',108,9374836669),(420,'surya','pune',109,9779349718),(421,'karthik','bangalore',101,8712647916),(422,'kiran','mysore',104,7624791274),(423,'indira','pune',102,78247916

49),(424,'sonia','hyderabad',105,7461746917),(425,'mahesh','chennai',103,3796491799),(426,'manasa','chennai',107,982918792),(427,'vishnu','delhi',106,7691468198),(428,'chaitra','bangalore',109,7364719714),(429,'pranav','mumbai',108,98649128918),(430,'pruthvik','pune',108,83419864918);

After Populating the tables:

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	bill	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	chef	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	consists of	Browse Structure Search Insert Empty Drop	62	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	customer	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	ingredients	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	meal	Browse Structure Search Insert Empty Drop	15	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	orders	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	64.0 KiB	-
☐	provides	Browse Structure Search Insert Empty Drop	46	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	supplier	Browse Structure Search Insert Empty Drop	30	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	9 tables	Sum	194	InnoDB	utf8mb4_general_ci	272.0 KiB	0 B

The orders and the Bill table is Populated in the front END

SQL Queries Implementation for Real life Problem statements

1)**Display the count of chefs working for different cuisines**

select cuisine,count(chef.cuisine) from chef group by chef.cuisine;

←T→	cuisine	count(chef.cuisine)
☐ Edit Copy Delete	american	1
☐ Edit Copy Delete	chinese	2
☐ Edit Copy Delete	desserts	1
☐ Edit Copy Delete	italian	2
☐ Edit Copy Delete	lebanese	1
☐ Edit Copy Delete	northindian	1
☐ Edit Copy Delete	south indian	1

2) **Display the menu according to the cuisine**

select meal.cuisine, meal.name from meal order by meal.cuisine asc;

☐	Edit	Copy	Delete	american	mac and cheese
☐	Edit	Copy	Delete	chinese	manchurian
☐	Edit	Copy	Delete	chinese	noodles
☐	Edit	Copy	Delete	desserts	cake
☐	Edit	Copy	Delete	desserts	brownies
☐	Edit	Copy	Delete	desserts	chocolate milkshake
☐	Edit	Copy	Delete	italian	panna cota
☐	Edit	Copy	Delete	italian	pizza
☐	Edit	Copy	Delete	italian	pasta
☐	Edit	Copy	Delete	north indian	curry
☐	Edit	Copy	Delete	north indian	roti
☐	Edit	Copy	Delete	north indian	biryani
☐	Edit	Copy	Delete	south indian	idly
☐	Edit	Copy	Delete	south indian	dosa

3)Display the chef details whose experience is more that 3 years

select * from chef where chef.experience>3;

←T→				chef_id	name	salary	cuisine	experience	sex	phno
☐	Edit	Copy	Delete	103	Sonali	60000	italian	5	female	4672874671
☐	Edit	Copy	Delete	104	Ratan	50000	northindian	4	male	7491089563
☐	Edit	Copy	Delete	105	Reshma	45000	lebanese	4	female	9284820818
☐	Edit	Copy	Delete	106	priya	45000	chinese	4	female	6284673829
☐	Edit	Copy	Delete	107	prakash	80000	italian	6	male	7295728472
☐	Edit	Copy	Delete	109	divya	80000	desserts	4	female	7436372366

4) Display the ingredients whose shelf life is 1 year

select ingredients.name from ingredients where shelf_life= '1 year';

←T→			name
Edit	Copy	Delete	flour
Edit	Copy	Delete	noodles
Edit	Copy	Delete	sugar

5) **Display the meals ordered by harsha('customer')**

select customer.name,meal.name from meal,customer,orders where meal.meal_id=orders.meal_id and customer.customer_id=orders.customer_id and customer.name='yogitha';

name	name
Yogitha	brownies
Yogitha	cake

6) Display all the ingredients used in the preparation of noodles

select meal.name , ingredients.name from ingredients,meal,consists_of where consists_of.ing_id = ingredients.ing_id and consists_of.meal_id = meal.meal_id and meal.name='noodles';

name	name
noodles	noodles
noodles	vegetables
noodles	sauces
noodles	spices
noodles	greens

7) Display the supplier details for the spices in the inventory

select ingredients.name,supplier.name,supplier.supplier_id,supplier.city from supplier, ingredients , provides where supplier.supplier_id=provides.supplier_id and ingredients.ing_id = provides.ingredients_id and ingredients.name='spices';

name	name	supplier_id	city
spices	bushan	409	mysore
spices	vishnu	427	delhi

8) Display the total number of male and female chef in the restaurant

select sex,count(*) from chef group by sex;

←T→				sex	count(*)
☐	Edit	Copy	Delete	female	4
☐	Edit	Copy	Delete	male	5

9) Display the customer details who have ordered meals that costs more than 1000 rupees

select customer.name,customer.phno,customer.address,bill.price from meal,customer,orders,bill where meal.meal_id=orders.meal_id and customer.customer_id=orders.customer_id and bill.bill_id = orders.bill_id and bill.price>1000 ;

name	phno	address	price
harsha	8296444590	31/2 RT road Bangalore	1050

10) Display the customers name who have ordered veg from the menu

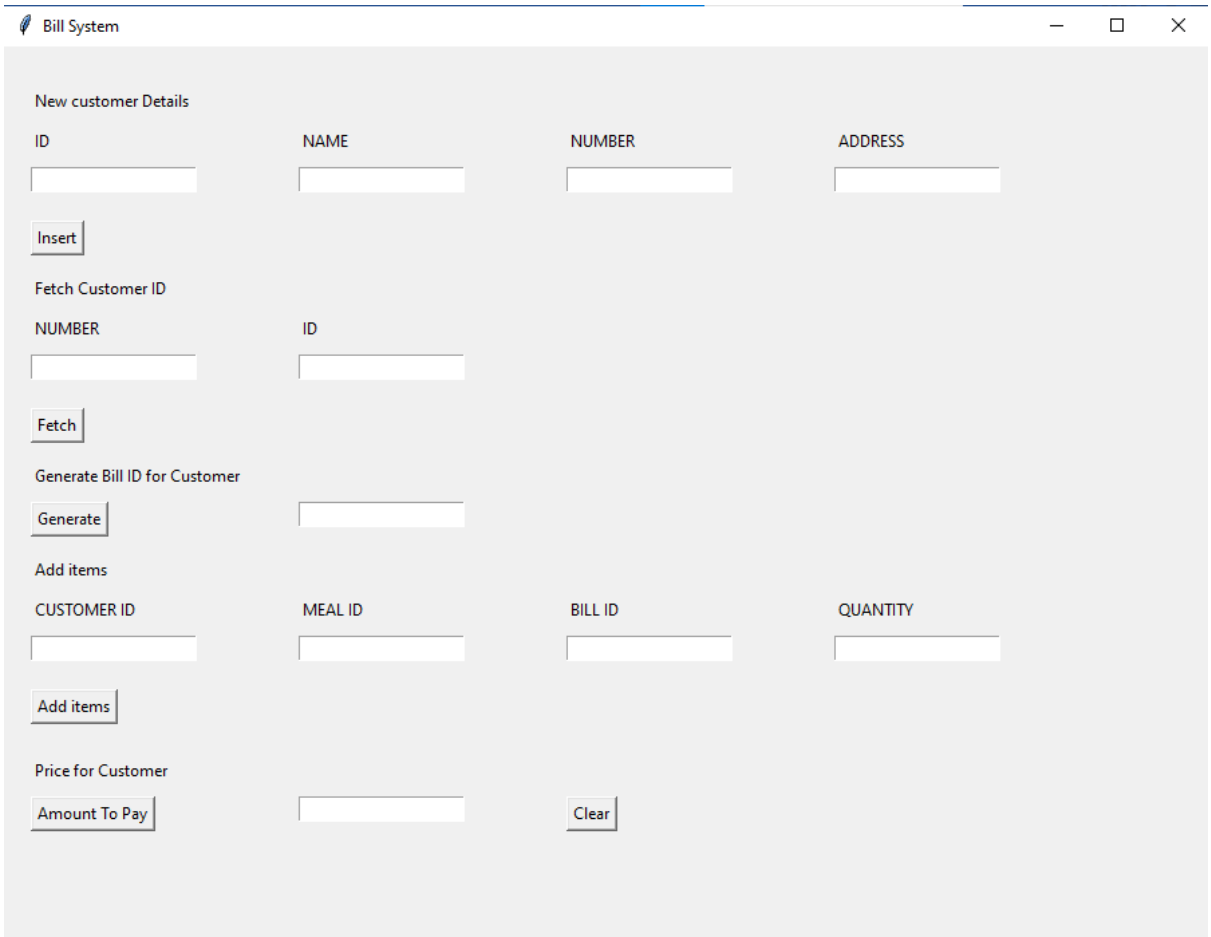
select customer.name,meal.name from meal,customer,orders where meal.meal_id=orders.meal_id and customer.customer_id=orders.customer_id and meal.preference = 'veg';

name	name
harsha	pasta

Function and procedure to calculate bill based on orders and updating the bills table

```
-- function to calculate the output of the bill price:
DELIMITER $$
CREATE FUNCTION compute_price(bill_no INTEGER)
RETURNS INTEGER
BEGIN
DECLARE meal_no INTEGER;
DECLARE cost INTEGER;
DECLARE meal_quantity INTEGER;
DECLARE meal_price INTEGER;
DECLARE finished INTEGER DEFAULT 0;
DECLARE c CURSOR for SELECT meal_id,quantity FROM orders WHERE bill_id = bill_no;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;
SET cost = 0;
OPEN c;

get_values: LOOP
FETCH FROM c INTO meal_no,meal_quantity;

IF finished = 1 THEN
LEAVE get_values;
END IF;

SELECT price into meal_price FROM meal WHERE meal_id = meal_no;
SET cost = cost + meal_price * meal_quantity;
END LOOP get_values;


CLOSE c;
RETURN cost;
END $$
DELIMITER ;

--procedure to update the bill section
DELIMITER $$
CREATE PROCEDURE up(IN bill_no integer)
BEGIN
UPDATE bill set price = compute_price(bill_no) where bill_id = bill_no;
end $$
DELIMITER ;
```

GUI Using Tkinter



This GUI helps the manager to generate bills by entering the customer ID and adding the details of the meal

First the records the phone number from the Customer to fetch his/her ID if it's a new customer then the GUI prompts the manager to add his/her details

Then a bill id is generated

After generating the bill id the manager has to enter the meal IDs ordered by the customer along with their ID and quantity

After adding all the meal IDs The manager can generate the bill amount to be paid

GUI code (Python Tkinter)

```
import mysql.connector as mysql
import tkinter.messagebox as MessageBox
from tkinter import *

def insert():
   id = e_id.get()
   name = e_name.get()
   num = e_num.get()
   add = e_add.get()
   print(id)
   if(id == "" or name == "" or num == "" or add == ""):
      MessageBox.showinfo("Insert status", "All fields are required")
   else:
      con = mysql.connect(host="localhost", user="root",
                     password="hars8296", database="restaurant")
      cursor = con.cursor()
      cursor.execute("insert into customer values('"+id +
                "','"+name+"','"+num+"','"+add+"')")
      cursor.execute("commit")
      e_id.delete(0, 'end')
      e_add.delete(0, 'end')
      e_num.delete(0, 'end')
      e_name.delete(0, 'end')
      MessageBox.showinfo("Insert status", "New customer added Successfully")
      con.close()

def fetch():
   if(e_numf.get() == ""):
      MessageBox.showerror("Fetch Status", "Enter Id")
   else:
      con = mysql.connect(host="localhost", user="root",
                     password="hars8296", database="restaurant")
      cursor = con.cursor()
      cursor.execute("select * from customer where phno='"+e_numf.get()+"'")
      rows = cursor.fetchall()
      if(len(rows) == 0):
         e_idf.delete(0, 'end')
         e_idf.insert(0, "New Customer")
      else:
         e_idf.delete(0, 'end')
         e_idf.insert(0, rows[0][0])
      con.close()

def generateid():
   con = mysql.connect(host="localhost", user="root",
                  password="hars8296", database="restaurant")
   cursor = con.cursor()
   cursor.execute("select MAX(bill_id) from bill")
   rows = cursor.fetchall()
   e_bid.delete(0, 'end')
   e_bid.insert(0, rows[0][0]+1)
   e_billid.delete(0, 'end')
   e_billid.insert(0, rows[0][0]+1)
   val = (e_bid.get())
```

```
    m = '0'
    cursor.execute("insert into bill values('"+val+"','"+m+"')")
    cursor.execute("commit")
    con.close()

def AddItems():
    r = e_custid.get()
    m = e_meal.get()
    s = e_billid.get()
    q = e_quant.get()
    con = mysql.connect(host="localhost", user="root",
                        password="hars8296", database="restaurant")
    cursor = con.cursor()
    cursor.execute("insert into orders values('" +
                  r+"','"+m+"','"+s+"','"+q+"')")
    cursor.execute("commit")
    con.close()
    e_custid.delete(0, 'end')
    e_meal.delete(0, 'end')
    e_quant.delete(0, 'end')

def generateprice():
    con = mysql.connect(host="localhost", user="root",
                        password="hars8296", database="restaurant")
    cursor = con.cursor()
    cursor.execute("select MAX(bill_id) from bill")
    rows2 = cursor.fetchall()
    val = str(rows2[0][0])
    print(val)
    print(type(val))
    cursor.execute("call up('"+val+"')")
    cursor.execute("commit")
    cursor.execute("select price from bill where bill_id='"+val+"'")
    rows3 = cursor.fetchall()
    e_aid.delete(0, 'end')
    e_aid.insert(0, rows3[0][0])
    e_billid.delete(0, 'end')

def clear():
    e_aid.delete(0, 'end')
    e_idf.delete(0, 'end')
    e_numf.delete(0, 'end')
    e_bid.delete(0, 'end')

root = Tk()
root.title('Bill System')
root.geometry("900x700")

head = Label(root, text='New customer Details')
head.place(x=20, y=30)
id = Label(root, text='ID')
id.place(x=20, y=60)
name = Label(root, text='NAME')
name.place(x=220, y=60)
num = Label(root, text='NUMBER')
num.place(x=420, y=60)
add = Label(root, text='ADDRESS')
```

```
add.place(x=620, y=60)

e_id = Entry()
e_id.place(x=20, y=90)
e_name = Entry()
e_name.place(x=220, y=90)
e_num = Entry()
e_num.place(x=420, y=90)
e_add = Entry()
e_add.place(x=620, y=90)
insert = Button(root, text="Insert", command=insert)
insert.place(x=20, y=130)

head1 = Label(root, text='Fetch Customer ID')
head1.place(x=20, y=170)
numf = Label(root, text='NUMBER')
numf.place(x=20, y=200)
idf = Label(root, text='ID')
idf.place(x=220, y=200)
e_numf = Entry()
e_numf.place(x=20, y=230)
e_idf = Entry()
e_idf.place(x=220, y=230)
fetch = Button(root, text="Fetch", command=fetch)
fetch.place(x=20, y=270)

head2 = Label(root, text='Generate Bill ID for Customer')
head2.place(x=20, y=310)
gen = Button(root, text="Generate", command=generateid)
gen.place(x=20, y=340)
e_bid = Entry()
e_bid.place(x=220, y=340)

head3 = Label(root, text='Add items ')
head3.place(x=20, y=380)
custid = Label(root, text='CUSTOMER ID')
custid.place(x=20, y=410)
meal = Label(root, text='MEAL ID')
meal.place(x=220, y=410)
billid = Label(root, text='BILL ID')
billid.place(x=420, y=410)
quant = Label(root, text="QUANTITY")
quant.place(x=620, y=410)
e_custid = Entry()
e_custid.place(x=20, y=440)
e_meal = Entry()
e_meal.place(x=220, y=440)
e_billid = Entry()
e_billid.place(x=420, y=440)
e_quant = Entry()
e_quant.place(x=620, y=440)
gen2 = Button(root, text="Add items", command=AddItems)
gen2.place(x=20, y=480)

head4 = Label(root, text='Price for Customer')
head4.place(x=20, y=530)
gen4 = Button(root, text="Amount To Pay", command=generateprice)
```

```
gen4.place(x=20, y=560)
e_aid = Entry()
e_aid.place(x=220, y=560)
gen5 = Button(root, text="Clear", command=clear)
gen5.place(x=420, y=560)

root.mainloop()
```

Individual Contribution :

NAME	CONTRIBUTION
Sinchana K	Tkinter and SQL queries
Yogitha HK	Tkinter and SQL queries
Ankit Loni	SQL queries and Relational Schema
Poorvi Raddi	E-R diagram and SQL queries

***