

Project Report

Dietician's Database

by

Krish Ahuja, Roll number: R002

Bhagyashree Birje, Roll number: R007

Tanishkaa Chaturvedi, Roll number: R011

Course: DBMS

AY: 2022-23

SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering
A.Y. 2022 - 23
Course: Database Management Systems

Project Report

Program	MBA Tech Artificial Intelligence	
Semester	III	
Name of the Project:	Dietician’s Database	
Details of Project Members		
Batch:	Roll No.	Name:
B1	R002	Krish Ahuja
B1	R007	Bhagyashree Birje
B1	R011	Tanishkaa Chaturvedi
Date of Submission: 19/10/2022		

Contribution of each project Members:

Roll No.	Name:	Contribution
R002	Krish Ahuja	ERD, storyline, Relational model
R007	Bhagyashree Birje	ERD, Debugging, normalization, queries
R011	Tanishkaa Chaturvedi	ERD, table creation, data entry

Table of Contents

Sr no.	Topic	Page no.
1	Storyline	3
2	Components of Database Design	4-7
3	Entity Relationship Diagram	8
4	Relational Model	9
5	Normalization	10
6	SQL Queries	11-15
7	Project Demonstration	16-19
8	Learning from the Project	20
9	Challenges you faced while doing the Project	21
10	Conclusion	22

I. Storyline

The database addressed in our project is that of a Dietician healthcare agency that is run in two branches, Bandra and Vile Parle. The dieticians are allocated to the different branches and their fees are dependent on the experience of the dietician. The dietician agency has to coordinate with customers, dieticians, vendors and hospitals in order to successfully run their agency.

The dieticians are appointed and paid a salary based on their experience and they decide the meal chart of their customers. Customers experience different issues which can be helped by providing a diet suited for it. The vendors are those stores that will help to provide the meals as per their meal chart. We can run specific searches to provide the customers with vendors that are near their location.

The food groups are divided on the basis of the nutrient it specializes in and each consists of combinations of carbohydrates, vitamins, minerals, proteins and fibers. These food groups are then used to decide breakfast, lunch and dinner. The meal chart then consists of combinations of breakfast, lunch and dinner and is decided on the basis of the condition faced by the customer. The permutations and combinations of these are stored in the databases as per the suggestions of the dietician.

Before prescribing a particular diet, the customer has to undertake a blood test and also book an appointment with the dietician who will study these blood test results. Specific searches can be performed by the dietician to detect any anomalies in the customer's blood which helps to note the nutrients that they lack and can be compensated by the suggested meal plan.

The customer pays for the dietician services in installments which is recorded in a different table with the details of payment recorded duly.

II. Components of Database Design

Entities and attributes:

1.Customer (customer_ID, c_firstname, c_lastname, c_address, c_phone, c_age, blood group, **dietician_ID**, meal_no, pay_no)

The 'Customer' entity displays the customer ID, customer's first as well as last name, customer's address, customer's phone number, customer's age, customer's blood group with foreign keys that are dietician ID, their mean number, and their payment number. In the 'Customer' entity, the primary key is 'Customer_ID' which can display all the attributes of the customer entity present.

Eg. show* from customer where customer_ID=101;

2.Dietician (dietician_ID, **issue_code**, **exp_code**, salary, location)

The 'Dietician' entity displays the dietician ID, their salary, their location with foreign keys that are issue code and exp code. In the 'Dietician' entity, the primary key is the 'dietician_ID' which can display all the attributes of the dietician entity present.

Eg. show* from dietician where dietician_id=3;

3.Issue(**issue_code**, issue_desc)

The 'Issue' entity displays the issue code and the issue description. In the 'Issue' entity, the primary key is 'issue_code' which can display all the attributes of the issue entity present. The 'Issue' entity is related to the 'Customer' entity.

4.Experience(**exp_code**, fees)

The 'Experience' entity displays the experiment code and the fees of the 'Dietician'. The primary key is 'exp_code' which can display all the attributes of the experience entity present. The 'Experiment' entity is related to the 'Dietician' entity.

5.Booking(dietappt_no, test_no, **customer_ID**)

The 'Booking' entity displays the diet appointment number, test number and has a foreign key- 'customer ID'. The primary key in the 'Booking' entity is customer_ID which is also the foreign key.

6.Dietbooking(dietappt_no, d_date, d_time)

The 'Dietbooking' entity displays the diet appointment number, diet booking date and diet booking time. The primary key is 'dietappt_no.' This is a weak entity which depends on the 'Booking' entity.

7. Testbooking(test_no, t_date, t_time, **hospital_no**)

The 'Testbooking' entity displays the test number, test date, test time and the respective hospital number for the test to be conducted at. The primary key is 'test_no'. This is a weak entity which depends on the 'Booking' entity.

8. Blood_Test_result(**test_no**, platelets, hemoglobin, Urine test, Calcium)

The 'Blood test result' entity displays the platelets, hemoglobin, urine test, calcium with a foreign key which is also the primary key for this entity. This is a weak entity which depends on the 'Testbooking' entity.

9. Meal chart(**issue_code**, meal_no, **b_no**, **l_no**, **d_no**)

The 'Meal chart' entity displays the meal number with the foreign keys: issue_code, breakfast_number, lunch_number, dinner_number. This entity depends on the 'Dietician' entity.

10. Breakfast(b_no, **group_no**)

The 'Breakfast' entity displays the breakfast number with the foreign key 'group_no'. This depends on the 'Meal chart' entity.

11. Lunch(l_no, **group_no**)

The 'Lunch' entity displays the lunch number with the foreign key 'group_no'. This depends on the 'Meal chart' entity.

12. Dinner(d_no, **group_no**)

The 'Dinner' entity displays the dinner number with the foreign key 'group_no'. This depends on the 'Meal chart' entity.

13. Food groups (group_no, group_name, **m_no**, **v_no**, **p_no**, **c_no**, **f_no**, group_desc, qty)

The 'Food groups' entity displays the group number, group name, group description, quantity with foreign keys 'minerals_no', 'vitamins_no', 'proteins_no', 'carbs_no', 'fibers_no'. The primary key is 'group_no'.

14. Minerals(m_no, m_name, m_desc)

The 'Minerals' entity displays the minerals number, minerals name and minerals description. The primary key is 'm_no'.

15. Vitamins (v_no, v_name, v_desc)

The 'Vitamins' entity displays the vitamins number, vitamins name and vitamins description. The primary key is 'v_no'.

16. Proteins(p_no, p_name, p_desc)

The 'Proteins' entity displays the protein number, protein name and protein description. The primary key is 'p_no'.

17. Carbs(c_no, c_name, c_desc)

The 'Carbs' entity displays the carb number, carb name and carb description. The primary key is 'c_no'.

18. Fibers(f_no, f_name, f_desc)

The 'Fibers' entity displays the fiber number, fiber name and fiber description. The primary key is 'f_no'.

19. Vendors(Vendor_no, vendor_name, v_contact, v_location , **group_no**)

The 'Vendors' entity displays the vendor number, vendor name, vendor contact, vendor location and the foreign key 'group_no'. The primary key is 'vendor_no'.

20. Hospitals(hospital_no, **issue_code**, h_location, h_phone)

The 'Hospital' entity displays the hospital number, hospital location, hospital phone with the foreign key 'issue_code'. The primary key is 'hospital_no'.

21. Payment(pay_no, pay_date, pay_amt)

The 'Payment' entity displays the payment number, payment date, payment amount. The primary key is 'pay_no'.

Relationships:

1. Dietician looks after Customer. (one-to-many)
2. Dietician assigns Meal Chart. (many-to-many)
3. Customer books booking.(one-to-one)
4. Testbooking records Blood test result.(one-to-one)
5. Blood test result is enabled by Hospital.(many-to-one)
6. Meal chart recommends Lunch .(many-to-one)
7. Meal chart recommends Dinner.(many-to-one)
8. Meal chart recommends Breakfast.(many-to-one)
9. Lunch consists food groups.(many-to-one)

- 10. Dinner consists food groups.(many-to-one)
- 11. Breakfast consists food groups.(many-to-one)
- 12. Food groups is sold by Vendors .(many-to-many)

III. Entity Relationship Diagram



IV. Relational Model

In the following relational model, primary keys are underlined while the foreign keys have been made bold:

1. Customer (customer_ID, c_firstname, c_lastname, c_address, c_phone, c_age, blood group, **dietician_ID**, **meal_no**, **pay_no**)
2. Dietician(dietician_ID, **issue_code**, **exp_code**, salary, location)
3. Issue(issue_code, issue_desc)
4. Experience(exp_code, fees)
5. Booking(**booking_id**, dietappt_no, test_no, **customer_ID**)
6. Dietbooking(dietappt_no, d_date, d_time)
7. Testbooking(test_no, t_date, t_time, **hospital_no**)
8. Blood Test result(**test_no**, platelets, hemoglobin, Urine test, Calcium)
9. Meal chart(**issue_code**, **meal_no**, **b_no**, **l_no**, **d_no**)
10. Breakfast(**b_no**, **group_no**)
11. Lunch(**l_no**, **group_no**)
12. Dinner(**d_no**, **group_no**)
13. Food groups (group_no, group_name, **m_no**, **v_no**, **p_no**, **c_no**, **f_no**, group_desc, qty)
14. Minerals(m_no, m_name, m_desc)
15. Vitamins (v_no, v_name, v_desc)
16. Proteins(p_no, p_name, p_desc)
17. Carbs(c_no, c_name, c_desc)
18. Fibers(f_no, f_name, f_desc)
19. Vendors(Vendor_no, vendor_name, v_contact, v_location , **group_no**)
20. Hospitals(hospital_no, **issue_code**, h_location, h_phone)
21. Payment(pay_no, pay_date, pay_amt, **customer_id**)

V. Normalization

Let us consider the table 'Customer' to be normalized:-

1NF:

For 1 Normal Form, the following requirements need to be achieved:

1. Each table has a primary key
2. The values in each column of a table are atomic
3. There are no repeating groups

Earlier the first table was customer with the following relational model:

Customer (customer_ID, c_name, c_address, c_phone, c_age, blood group, dietician_ID, meal_no, pay_no, issue_code)

In this the c_name was a multi-valued attribute, that is it had two values, first name and last name. So after applying 1NF, we divided the name attribute further into 2 attributes: first name and last name as follows:

Customer (customer_ID, c_firstname, c_lastname, c_address, c_phone, c_age, blood group, dietician_ID, meal_no, pay_no, issue_code)

2NF:

For 2 Normal Form, the following requirements need to be achieved:

1. The table should be in 1NF
2. The table should have no partial dependencies

Then further in this table, the foreign key attribute issue_code was determining dietician_ID which is a subset of the primary key:

Customer (customer_ID, c_firstname, c_lastname, c_address, c_phone, c_age, blood group, dietician_ID, meal_no, pay_no, issue_code)

This means issue_code displayed a partial dependency on dietician_ID. Thus in order to eliminate this, a new table for issues was formed:

Issue(issue_code, issue_desc)

The customer table finally had these attributes:

Customer (customer_ID, c_firstname, c_lastname, c_address, c_phone, c_age, blood group, dietician_ID, meal_no, pay_no)

3NF:

For 3 Normal Form, the following requirements need to be achieved:

3. The table should be in 2NF
4. The table should have no transitive dependencies

There does not exist a transitive dependency in the customer table, thus we can say the customer table is already in 3NF and BCNF form.

VI. SQL Queries

Q.1 Show the details of the customer ID 1111 except the customer's age.

Code: `select customer_id, c_firstname, c_lastname, c_address, c_phone, blood_group, dietician_id, meal_no, pay_no from customer where customer_id=1111;`

Output:

customer_id	c_firstname	c_lastname	c_address	c_phone	blood_group	dietician_id	meal_no	pay_no
1111	Tanishkaa	Chaturvedi	402, Chakala	8764859324	B+	1002	12	2

Q.2 Show the most experienced dietician available

Code:

`select * from dietician natural join experience where exp_code=(select max(exp_code) from experience);`

Output:

exp_code	dietician_id	salary	issue_code	location	fees
4	1001	70000	1020	Vile Parle	50000
4	1004	70000	1040	Vile Parle	50000
4	1007	60000	1010	Bandra	50000
4	1009	70000	1020	Vile Parle	50000

Q.3 Display customer's name where diet appointment number is 3.

Code: `select c_firstname, c_lastname from customer natural join dietbooking natural join booking where dietappt_no=3;`

Output:

c_firstname	c_lastname
Neha	Grandhi

Q.4 Show the customer's names whose hemoglobin is greater than 1.

Code: `select c_firstname, c_lastname from blood_test_result natural join customer natural join booking where hemoglobin>1;`

Output:

c_firstname	c_lastname
Tanishkaa	Chaturvedi
Krish	Ahuja

Q.5 Update the timing for test booking number 3 to 5:00

Code: update testbooking set t_time="5:00" where test_no=3;

Output:

test_no	t_date	t_time	hospital_no
1	2022-10-10	03:00:00	803
2	2022-10-10	04:00:00	804
3	2022-10-11	05:00:00	804
4	2022-10-12	12:00:00	800

Q.6 Display the dietician id and customer name and location for those dieticians and customers who are around each other.

Code:

select dietician.dietician_id, c_firstname, c_lastname, location from dietician, customer where location=c_address ;

Output:

dietician_id	c_firstname	c_lastname	location
1001	Krish	Ahuja	Vile Parle
1002	Krish	Ahuja	Vile Parle
1003	Bhagyashree	Birje	Bandra
1004	Krish	Ahuja	Vile Parle
1005	Bhagyashree	Birje	Bandra
1006	Krish	Ahuja	Vile Parle
1007	Bhagyashree	Birje	Bandra
1008	Bhagyashree	Birje	Bandra
1009	Krish	Ahuja	Vile Parle
1010	Bhagyashree	Birje	Bandra

Q.7 Sort the food groups on basis of quantity

Code:

`select * from food_groups order by qty desc;`

Output:

group_no	group_name	group_desc	qty	m_no	v_no	p_no	c_no	f_no
1903	Sattvic food	Foods for clean and toxin free body	6	94	63	51	70	31
1900	Vitamin packed non-veg	Healthy foods enriched with vitamins and minerals	3	92	64	54	72	30
1901	Vitamin packed veg	healthy veg food enrichd with vitamins and minerals	3	94	63	50	70	31
1904	Iron man	Iron-rich food for anemia and other similar conditions	3	94	63	50	70	31
1905	Carbohydrate rich	Food for higher energy and carbohydrates	3	94	63	50	70	31
1902	Calcium rich	Calcium rich food with necessary supplements for healthy bones	2	91	61	53	70	30

Q.8 Show those protein that have plant-based options

Code:`select * from proteins where p_name like "%plant%";`

Output:

p_no	p_name	p_desc
50	plant protein legume	lentils, beans, peas, soybean
51	plant protein nuts	almonds,cashews,walnuts,sunflower seeds, sesame, chia
52	plant protein grain	rice, quinoa, oats, millet
53	plant protein veggies	corn, broccoli, asparagus, sprouts

Q.9 Display dinner options for customer wishing to gain weight

Code:

`select m_desc, v_desc, p_desc,c_desc,f_desc,issue_code from meal_chart natural join dinner natural join food_groups natural join vitamins natural join minerals natural join proteins natural join carbs natural join fibers where issue_code=1020;`

Output:

m_desc	v_desc	p_desc	c_desc	f_desc	issue_code
Seafod, iodised salt	Grains,broccoli, fruits, fish,milk,cheese	fish, crabs, chicken	pumpkin, carrot, tomatoes, beans, broccoli, cucumber	oatmeal,chia,lentils,apples,blueberries	1020
Honey, almonds,seafod, chocolates, pineapple, leafy vegetables	Green leafy vegetables, turnip, beet green	lentils, beans, peas, soybean	apples, oranges, banana,pineapple,berries, milk	whole wheat, quinoa, brown rice, leafy greens, walnuts, seeds, pear, apples	1020

Q.10 What is the average salary for the dieticians

Code: `select avg(salary) from dietician;`

Output:

avg(salary)
39500.0000

Q.11 What is the number of meal options available for a kid

Code:

```
select issue_code,count(*) from meal_chart group by issue_code having issue_code=1050;
```

Output:

issue_code	count(*)
1050	2

Q.12 Create a view for the hospital displaying only test booking details and customer details

Code:

```
create view vwhospi as select test_no, customer_id,
t_date,t_time,hospital_no,h_location,h_phone,issue_code from booking natural join customer natural join
testbooking natural join hospital;
```

Output:

test_no	customer_id	t_date	t_time	hospital_no	h_location	h_phone	issue_code
1	1111	2022-10-10	03:00:00	803	Kanjurmarg	6478352672	1020
2	2222	2022-10-10	04:00:00	804	Malad	675432685	1010
3	4444	2022-10-11	05:00:00	804	Malad	675432685	1010
4	3333	2022-10-12	12:00:00	800	Andheri East	9876543672	1030

Q.13 Display the total amount paid by customer 1111

Code:

```
select sum(pay_amt), customer_id from customer natural join payment where customer_id=1111;
```

Output:

sum(pay_amt)	customer_id
1000	1111

Q.14 Show carbohydrate options with beans in it

Code:

```
select * from carbs where c_desc like "%beans%";
```

Output:

c_no	c_name	c_desc
71	starchy	whole grains, grain bread, beans, potatoes, cereals
72	fibrous	pumpkin, carrot, tomatoes, beans, broccoli, cucumber

Q.15 Create a vendor view and join the tables visible to both the vendors as well as the hospital
Code:

`select * from vwhospi natural join vwvendor1;`

Output:

issue_code	test_no	customer_id	t_date	t_time	hospital_no	h_location	h_phone	vendor_name	v_location	vendor_no	group_name	meal_no
1020	1	1111	2022-10-10	03:00:00	803	Kanjurmarg	6478352672	Delomite	Andheri West	2222	Calcium rich	12
1020	1	1111	2022-10-10	03:00:00	803	Kanjurmarg	6478352672	Delomite	Andheri West	2222	Calcium rich	19
1010	2	2222	2022-10-10	04:00:00	804	Malad	675432685	Nageshwar	Bandra	3333	Sattvic food	10
1010	2	2222	2022-10-10	04:00:00	804	Malad	675432685	Kinara	Bandra	5555	Sattvic food	10
1010	2	2222	2022-10-10	04:00:00	804	Malad	675432685	Ashapura	Vile Parle West	1111	Iron man	11
1010	2	2222	2022-10-10	04:00:00	804	Malad	675432685	Onega	Bhandup	4444	Iron man	11
1010	3	4444	2022-10-11	05:00:00	804	Malad	675432685	Nageshwar	Bandra	3333	Sattvic food	10
1010	3	4444	2022-10-11	05:00:00	804	Malad	675432685	Kinara	Bandra	5555	Sattvic food	10
1010	3	4444	2022-10-11	05:00:00	804	Malad	675432685	Ashapura	Vile Parle West	1111	Iron man	11
1010	3	4444	2022-10-11	05:00:00	804	Malad	675432685	Onega	Bhandup	4444	Iron man	11
1030	4	3333	2022-10-12	12:00:00	800	Andheri East	9876543672	Nageshwar	Bandra	3333	Sattvic food	13
1030	4	3333	2022-10-12	12:00:00	800	Andheri East	9876543672	Kinara	Bandra	5555	Sattvic food	13

VI. Project demonstration

Our code was executed on the terminal and mysqldump was used to export the .sql file provided with this report.

Thus the format of the .sql file might appear differently on different softwares for the same code:

MySQL code:

```
create table payment(payment_no int primary key,  
pay_date date,  
pay_amt int );
```

```
create table minerals(m_no int primary key,  
m_name varchar(100),  
m_desc varchar(300));
```

```
create table vitamins(v_no int primary key, v_name varchar(1000), v_desc varchar(3000));
```

```
create table proteins(p_no int primary key, p_name varchar(1000), p_desc varchar(3000));  
create table carbs(c_no int primary key, c_name varchar (1000), c_desc varchar(3000));
```

```
create table fibers(f_no int primary key, f_name varchar(1000), f_desc varchar(3000));
```

```
create table food_groups(group_no int primary key,  
group_name varchar(30),  
group_desc varchar(1000),  
qty int,  
m_no int,  
v_no int,  
p_no int,  
c_no int,  
f_no int,  
foreign key (m_no) references minerals(m_no),  
foreign key (v_no) references vitamins(v_no),  
foreign key (p_no) references proteins(p_no),  
foreign key (c_no) references carbs(c_no),  
foreign key (f_no) references fibers(f_no)  
);
```

```
create table vendors(vendor_no int primary key,  
vendor_name varchar (30),
```

```
v_contact int,  
v_location varchar(50),  
group_no int,  
foreign key(group_no) references food_groups(group_no)  
);
```

```
create table breakfast(  
b_no int primary key,  
group_no int,  
foreign key (group_no) references food_groups(group_no)  
);
```

```
create table lunch(  
l_no int primary key,  
group_no int,  
foreign key (group_no) references food_groups(group_no)  
);
```

```
create table dinner(  
d_no int primary key,  
group_no int,  
foreign key (group_no) references food_groups(group_no)  
);
```

```
create table issue  
(issue_code int primary key,  
issue_desc varchar(30));
```

```
create table meal_chart(  
issue_code int,  
meal_no int primary key,  
b_no int,  
l_no int,  
d_no int,  
foreign key(issue_code) references issue(issue_code),  
foreign key(b_no) references breakfast(b_no),  
foreign key(l_no) references lunch(l_no),  
foreign key(d_no) references dinner(d_no)  
);
```

```
create table experience (exp_code int primary key, fees int);
```

```

create table dietician
(dietician_id int,
exp_code int,
salary int,
issue_code int ,
location varchar(50),
primary key (dietician_id,issue_code),
foreign key (issue_code) references issue (issue_code),
foreign key(exp_code) references experience(exp_code)
);

```

```

create table dietbooking(dietappt_no int primary key, d_date date, d_time time);

```

```

create table hospital(
  hospital_no int primary key,
issue_code int,
h_location varchar(30),
h_phone varchar(10),
foreign key (issue_code) references issue (issue_code)
);

```

```

create table testbooking (test_no int, int primary key, t_date date, t_time time,hospital_no int,
foreign key(hospital_no) references hospital(hospital_no);

```

```

create table customer
(customer_id int ,
c_firstname varchar(30),
c_lastname varchar(30),
c_address varchar (50),
C_phone varchar(10),
c_age int,
blood_group varchar(10),
dietician_id int not null,
meal_no int not null,
pay_no int,
primary key (customer_id,dietician_id),
foreign key(dietician_id) references dietician(dietician_id),
foreign key(meal_no) references meal_chart(meal_no),
foreign key(pay_no) references payment(pay_no)
);

```

```

create table booking
(

```

```
dietappt_no int,  
test_no int,  
customer_id int primary key,  
foreign key(customer_id) references customer(customer_id),  
foreign key(dietappt_no) references dietbooking(dietappt_no),  
foreign key(test_no) references testbooking(test_no)  
);
```

```
create table blood_test_result(test_no int primary key,  
platelets int,  
hemoglobin int,  
urine_test int,  
calcium int,  
foreign key(test_no) references testbooking (test_no)  
);
```

VII. Learning from the Project

- How did this project help you?

This project helped us in providing an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more. We learnt different types of attributes and weak entities an entity has and the relations the entity could have. As per our research, we also got to know about different types of queries which also include the complicated queries to run efficiently. This also helps in understanding the input user (customer) gives and based on that our database works smoothly while keeping all the requirements by the customer in mind and giving the best possible result. In the case of multiple users, it also maintains data consistency. It also helped us to develop the analytical skills of planning any dataset given to us.

- What new aspects did you learn?

We learnt how to design and maintain business databases. One of the most important aspect we learnt is the use of different kinds of entities, attributes and their relations being displayed in the desired way as per the customer's need. The management between all the tables (entities) and all their relations including their tuples (attributes) helps us to make a more simple user friendly database. As all the tables have relationships either being (one-to-one, one-to-many, many-to-one or many-to-many), they have a complete balance which calculates all the possibilities and data given by the customer which would then help the database in giving the best possible result, as in this database to the customer. Using more tuples, more values, different relationships and depending on what type and also as complicated queries can be run shows what different aspects we learnt through this project.

VIII. Challenges Faced

During the implementation of the database, there were challenges during updation of the tables since the referential integrity needed to be maintained.

Eg:

After creating the table 'customer', we performed normalization and realized the foreign key attribute 'issue_code' needs to be removed from 'customer' table.

Inorder to do that, the command 'alter table customer drop issue_code' did not work. After researching on the same, we learnt that when dropping a foreign key attribute we first need to drop the constraint on it. So to know the constraint, we used the command 'show create table 'customer';'

This displayed the constraint name as 'customer_ibfk_1' for issue_code and so then we dropped the constraint first by using the command 'alter table customer drop customer_ibfk_1;' and then we ran the command 'alter table customer drop issue_code;'

During the insertion of data we faced certain challenges pertaining to either the data format or the type of "" used

Eg: For the testbooking and dietbooking tables, we had to accept dates in date format. So for the date format, there was an error which was resolved by using the command 'STR_TO_DATE("10-10-22", "%d-%m-%y")' to insert the data value according to our desired database format. Furthermore during insertion of data the type of quotations used by the code also made a difference to the success of the sql code. The usage of " instead of ' threw an error since the code surprisingly did not accept using ".

Extra care was required to ensure the table exists for the foreign keys before they are being inserted in the table being created.

Eg: The table 'dietician(dietician_ID, **issue_code**, **exp_code**, salary, location)' cannot be created before the tables that have 'issue_code' and 'exp_code' as primary keys have been created.

Overall debugging the code to ensure that all data inserted matches the value required was a crucial and time consuming task.

IX. Conclusion

The implementation of the diet agency database helped us to understand the components and their uses in a database more clearly and efficiently. It was an opportunity to reflect on the need for a database in real life as well. The following were our key takeaways from the project:

1. The better and more time is spent in planning of the database, the easier and faster it is to execute the database at the end.
2. There are several advantages for creating a database for any particular dataset like:
 - a. Saves time to run searches throughout the entire dataset
 - b. Helps to prevent anomalies during the insertion, deletion or updation of any database.
 - c. The data can be easily shared across to different people through the sql files.