FOOD DELIVERY SERVICE

Professor:
Dr.Mutlu Mete

Project Members:
Srivinusha Janga (50267673)
Lakshmi Chinnam (50263055)
Purushotam Pradeep Kumar Reddy Desireddy (50256934)

Whose Oracle Account is used for this project ? Srivinusha Janga

Group ID: 6

Step 1: Create an imaginary scenario. Your scenario should satisfy following conditions:

This project (Food Service) involves below entities:

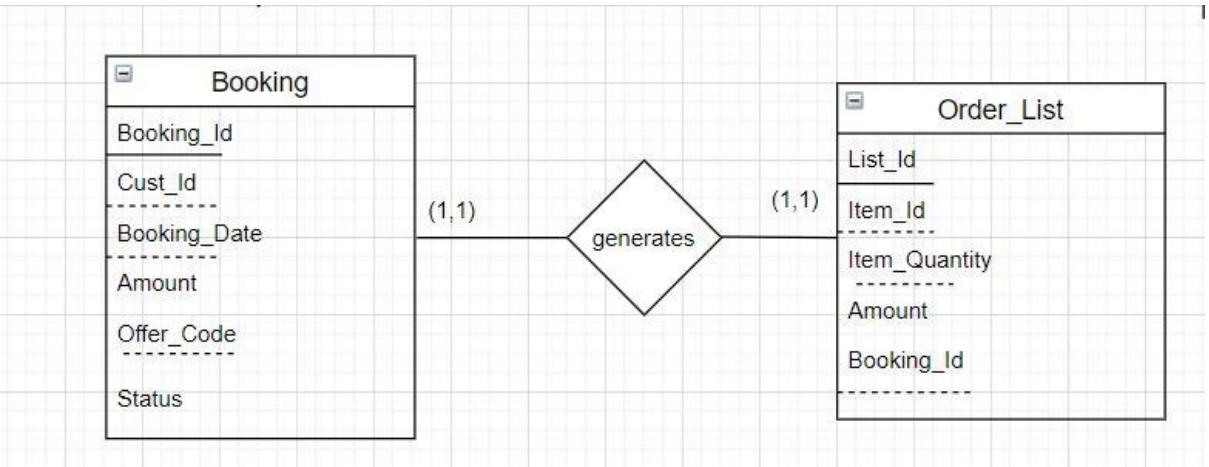
1) Delivery_Service
2) Restaurant
3) Feedback
4) Employee
5) Customer
6) Menu
7) Delivery_Vehicle
8) Order_List
9) Booking
10) Offer
11) Payment
12)Emp_Delivery
13)Nutrition_Fact

This project involves below relationships between entities.

1. A Scenario including at least two one-to-one binary relationships.

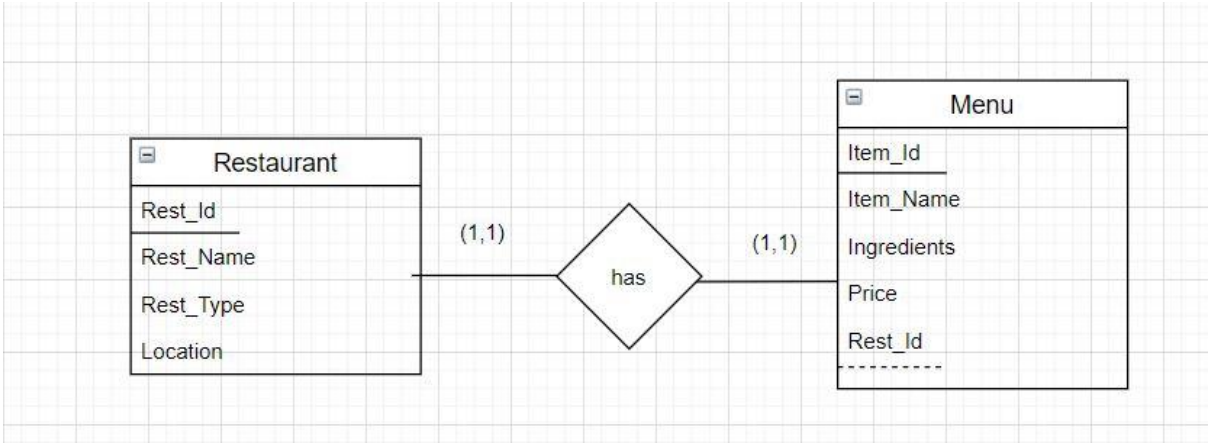
Order_List - Booking:

This project involves each Booking generating only one Order.For one Booking_ID only one Order_List_ID must be generated that can be written as at least one and at most one.



Restaurant-Menu:

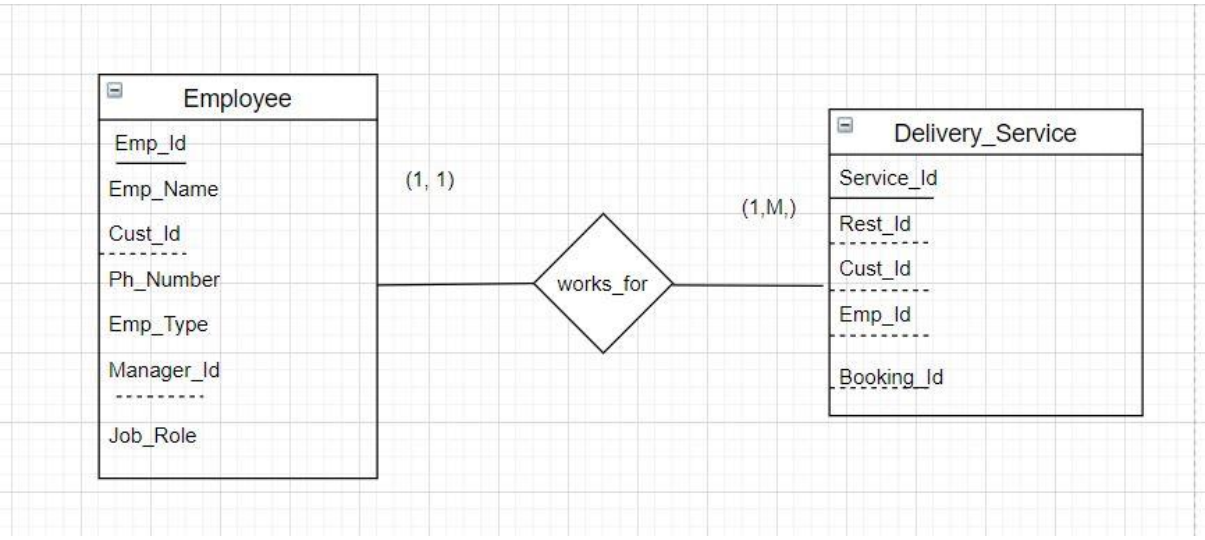
This project involves Restaurant having a one-to-one relationship with Menu. This Relationship shows Each Restaurant must contain one Menu.



2. A Scenario including at least two one-to-many binary relationships

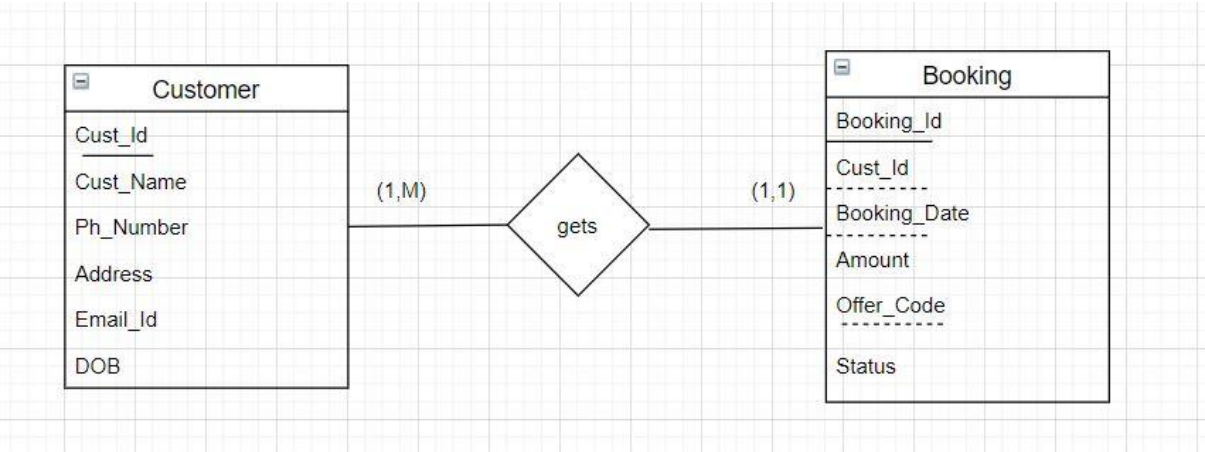
<u>Employee-DeliveryService :</u>

This project involves atleast one Employee at most many Employees working for Delivery_service having atleast one atmost one Service.



<u>Booking-Customer:</u>

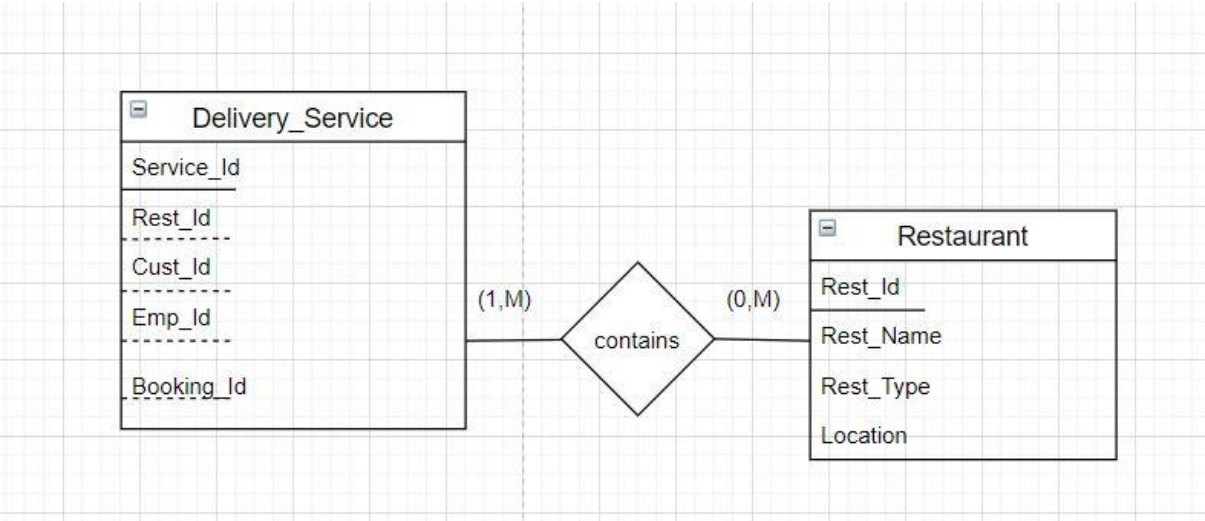
Customers get one or more Booking but Booking_Id is unique for every Customer i.e., Invoice generates atleast one at most one for atleast one atmost many Customers.



3. A Scenario including at least two many-to-many binary relationships

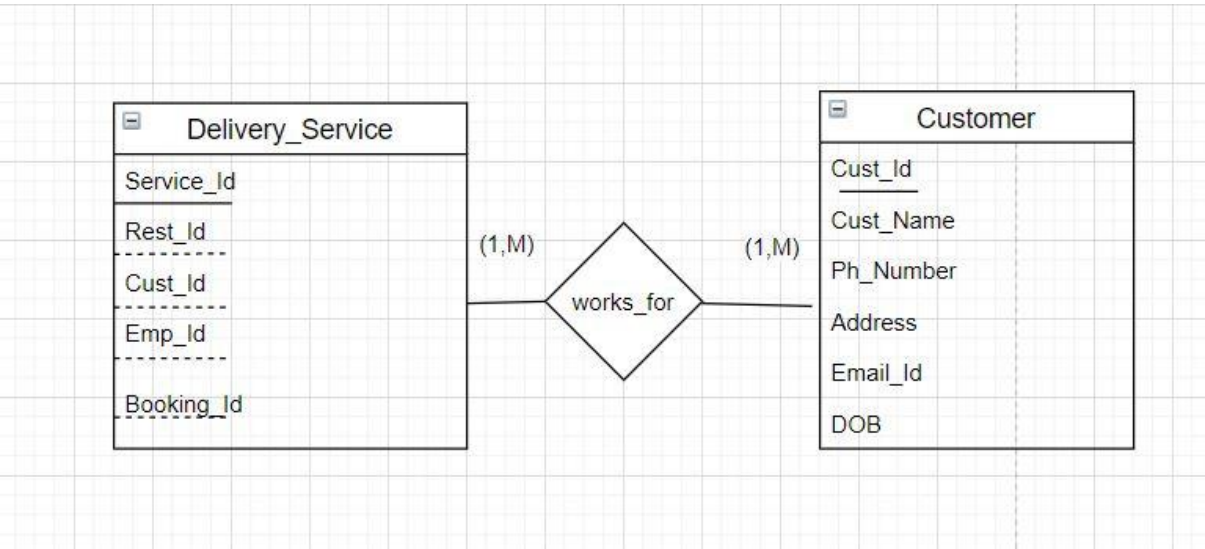
DeliveryService-Restaurant:

Delivery_Service contains atleast one atmost many Restaurant and each Restaurant may not group with Delivery Service Systems or may join with Many Delivery Service Systems.



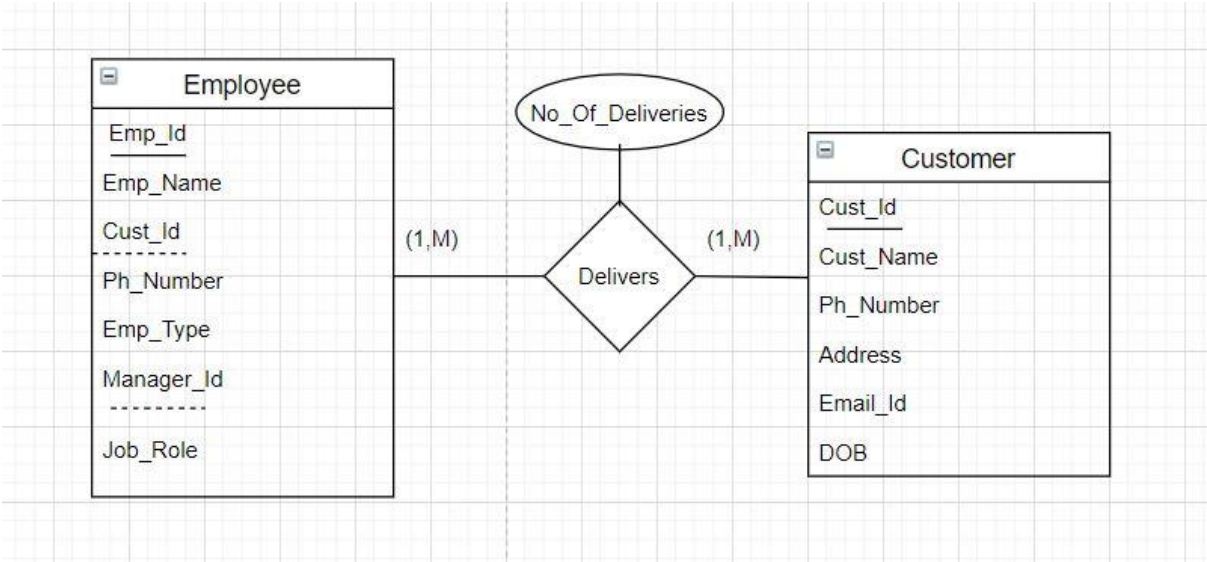
Delivery_Service - Customer :

Customer can use many Delivery Services so,Customer can order from minimum one or maximum many delivery services in the same way Delivery_Service is available for minimum one maximum many Customers.



4. A scenario including at least one intersection data over a many-to-many relationships

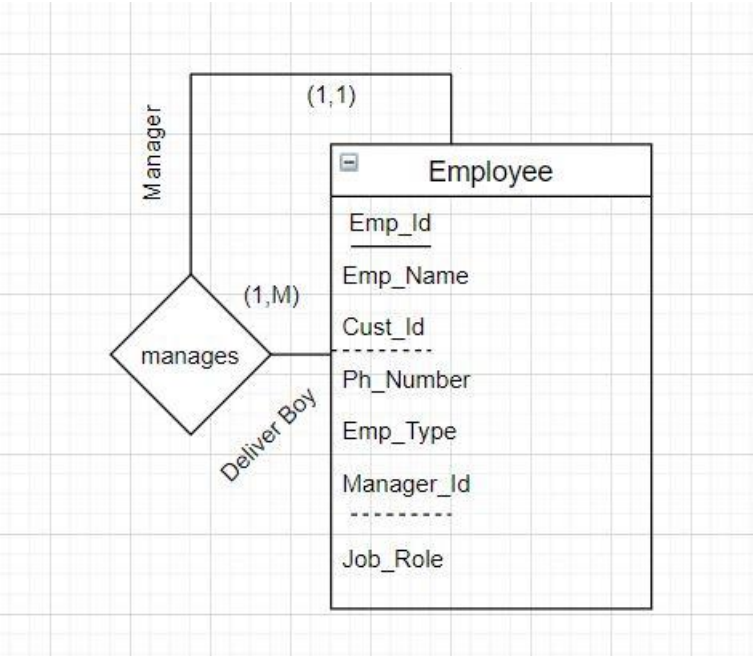
This project involves a many-to-many relation between Employee and Customer which further gives an intersection data . We have a Booking table which contains the BookingID from Booking table and Cust_ID from Customer table to give us the Deliveries count received by the customer and Deliveries with the help of BookingID and Deliveries_Number multivalued composite attribute of EMPLOYEE called Emp_Delivery with CUSTOMER .



5. A scenario including at least one one-to-many unary relationships

Employee:

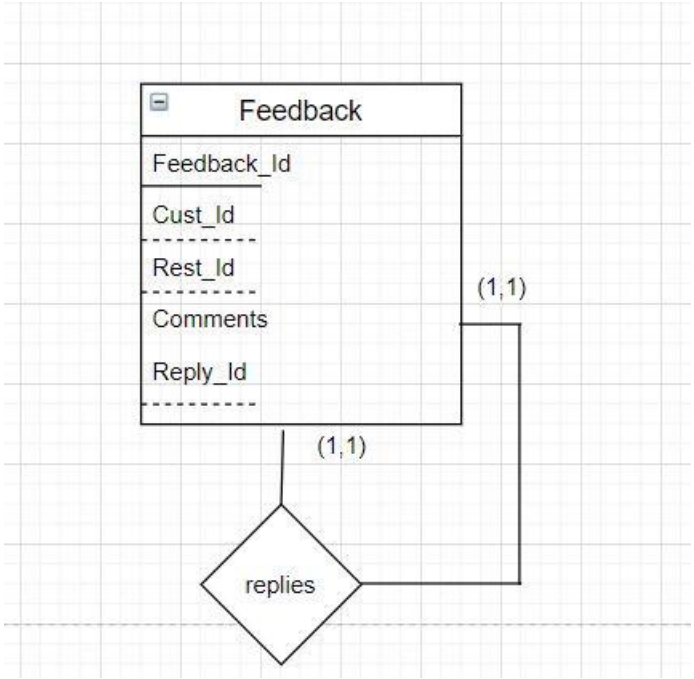
The Manages relationship type relates an employee to a Manager, where both employee and Manager entities are members of the same EMPLOYEE entity set. Hence, the EMPLOYEE entity type participates twice in Manages: once in the role of Manager and once in the role of food delivery person. Each relationship instance in Manages associates two different employee entities e1 and e2 one of which plays the role of Manager and the other the role of food delivery person.



6. A scenario including at least one one-to-one unary relationships

Feedback :

This project involves for Every Feedback comment there must be a reply from Restaurant Organization do reply for every comment in Feedback(minimum one reply maximum one reply).
Here, for one comment will allow only one reply to that comment which is recursive, Each reply can also be a comment.



Step 2: Explain the story behind the scenario, and all your assumptions, which are required to support relationships given above.

Food Delivery Service :

- Nowadays people are faced with a situation within which they are doing not have time to cook or prepare food.Access to a good sort of foods one among the foremost important advantages that these services provide is that the people may order from an oversized menu.
- These days a spread of local eateries and national restaurants participate in online food ordering by partnering with delivery companies like Uber Eats, Postmates, Grubhub and Waitr.
- These services allow customers to pursue a bigger selection of foods and restaurants and order food through a convenient online page or app.having it delivered to their exterior door,is helping these people get their work done typically seniors.

- Each Delivery_Service have sheer number of Restaurants offered.
- Adding delivery service to the restaurant will help you distinguish yourself from the competition,and it also attract Customers.
- It should have a good Food Delivery Staff(Employee) ,Delivery drivers have a their own vehicle(Emp_vehicle),Each Delivery person will be managed by the manager.
- Each employee will be trained on the best way to place the food in their cars so no spills occur and how to speak to the customer when they deliver the food.

<u>Story behind our mini world:</u>

This project involves

1) Food Delivery Service includes many offers

For e.g., the list of offers the project contains are;

- Senior Citizen Discount
- Volume Discount
- Trial Discount
- Seasonal
- Happy Hours
- Free delivery

Each OfferCode could also be available for under Booking_ID

3)This project provides below payment methods:

- Credit Card
- Debit Card
- PayPal
- Cash

4)Employee have two roles i.e Manager and Food Delivery Person.Manager who manages Many employees.

5)A Customer will have a chance to share feedback to the Restaurant and the Restaurant must give a reply to the comment.

Assumptions:
1) A Customer will have a chance to share feedback to the Restaurant and Restaurant must give a reply to the comment.

Step 3: Show your entities and their attributes. Each entity should have at least three attributes.

1) Delivery_Service
- *Service_ID
- Rest_ID
- Cust_ID
- Emp_ID
- Booking_ID

2) Restaurant
- *Rest_ID
- Rest_Name
- Rest_Type
- Location

3) Feedback
- *Feedback_ID
- Cust_ID
- Rest_ID
- Comments
- Reply_ID

4) Employee
- *Emp_ID
- Emp_Name
- Cust_ID
- Ph_Number
- Emp_Type

- Manager_ID
- Job_Role

5) Customer
- *Cust_Id
- Cust_Name
- Ph_Number
- Address
- Email_ID
- DOB

6) Menu
- *Item_ID
- Item_Name
- Ingredients
- Price
- Rest_ID

7) Delivery_Vehicle
- *Vehicle_ID
- Emp_ID
- Vehicle_Type

8) Order_List
- *List_ID
- Item_ID
- Item_Quantity
- Amount
- Booking_ID

9) Booking
- *Booking_ID
- Cust_ID
- Booking_Date
- Amount
- Offer_Code
- Status

10) Offer

- *Offer_Code
- Offer_Type
- Disc_Amount
- Validity

11) Payment

- *Pay_ID
- Pay_Type
- Amount
- Booking_ID

12)Emp_Delivery

- Emp_ID
- Service_ID
- Deliveries

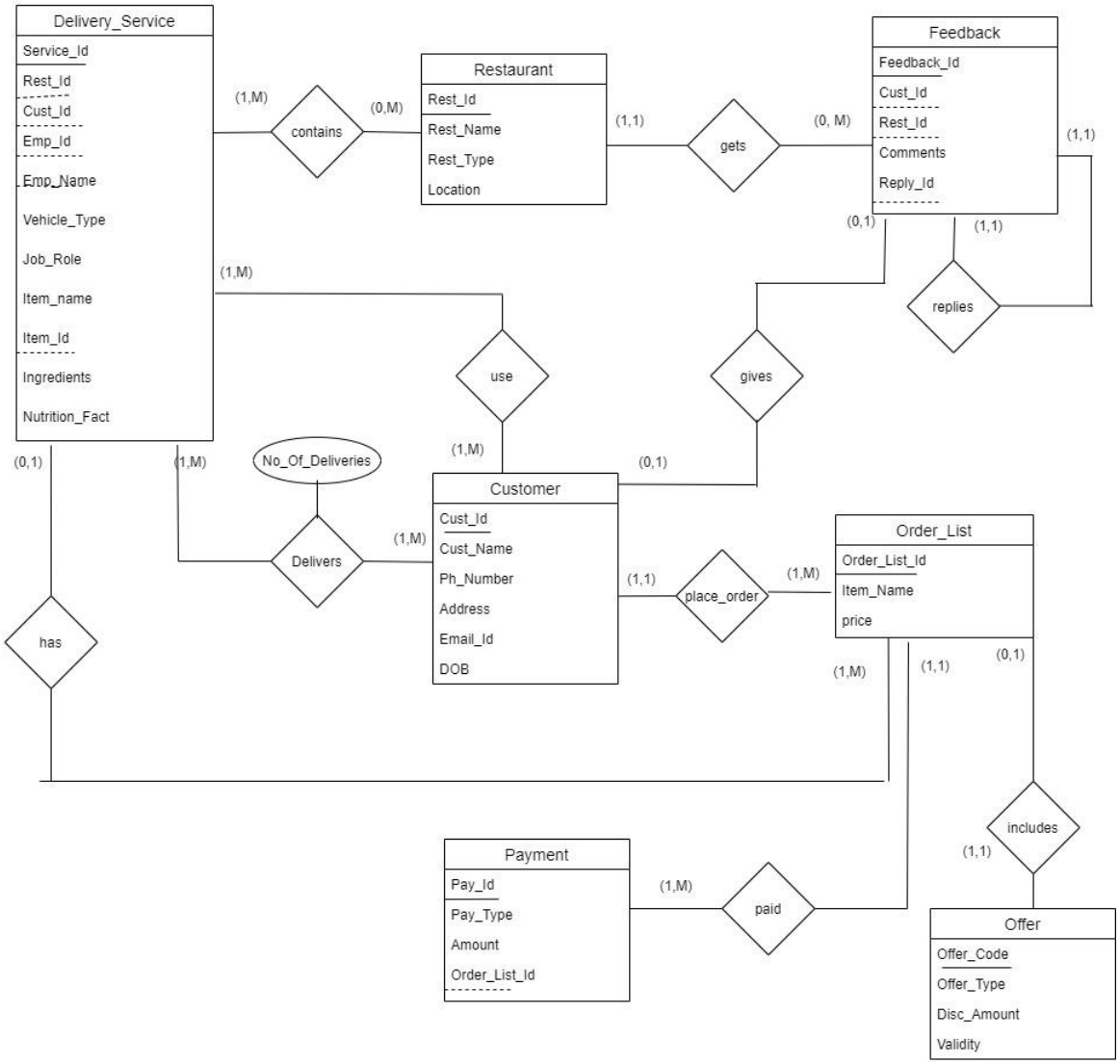
13)Nutrition_Fact

- *Nutrition_ID
- Item_ID
- Calories
- Vitamin

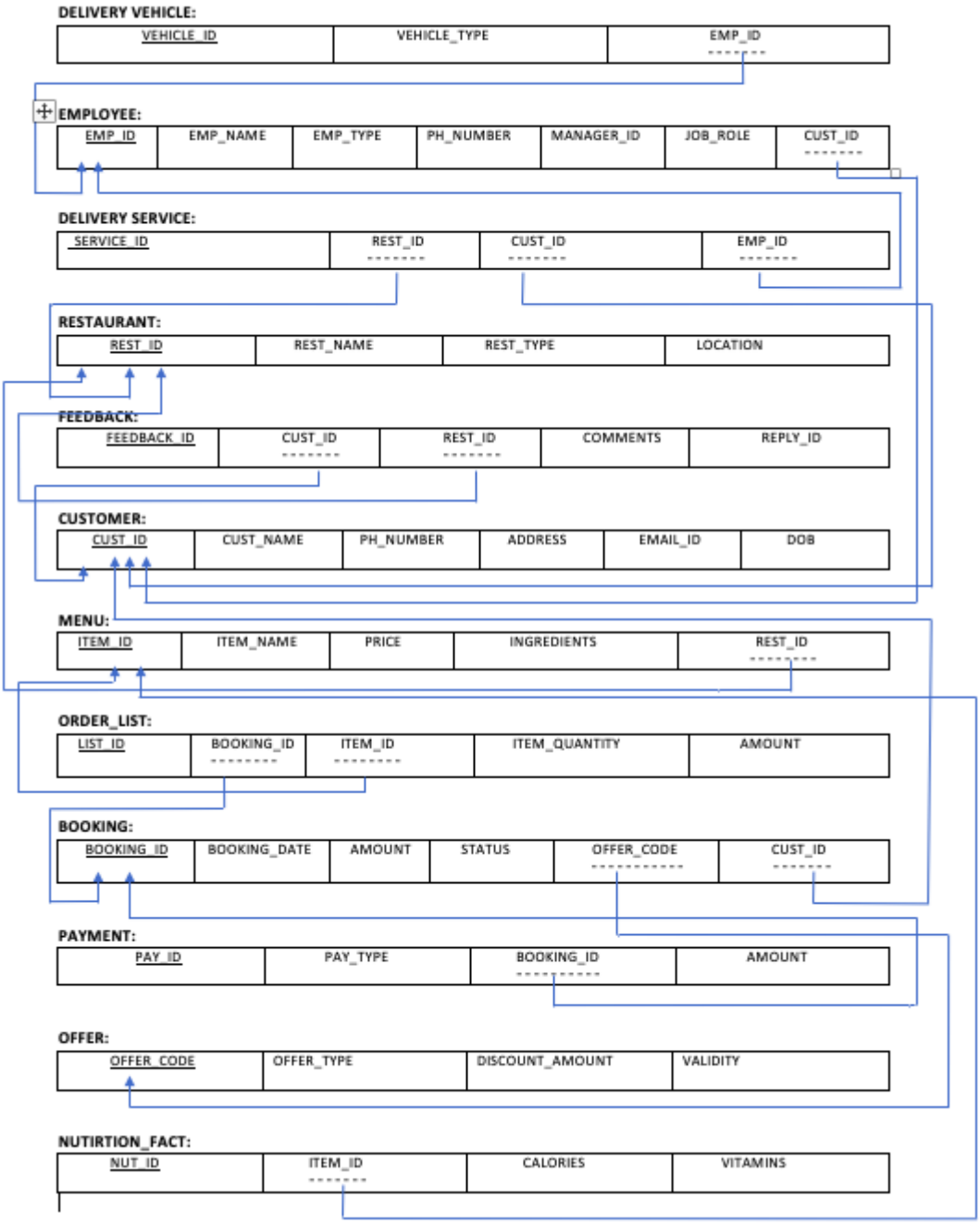
Step 4: Draw your initial ER diagram with min-max notation

Food Delivery Service

(Before Normalization)



Step 5: Show your referential integrity rules for your scenario(FK->PK)



6) Convert the ER diagram to tables. Show your tables with primary keys

Table 1:Delivery_Vehicle

[Worksheet]*

```sql
CREATE TABLE MYPROJECT.DELIVERY_SERVICE(
SERVICE_ID INT NOT NULL,
REST_ID INT NOT NULL,
CUST_ID INT,
EMP_ID INT,
BOOKING_ID INT,
CONSTRAINT SERVICE_ID_PK PRIMARY KEY(SERVICE_ID),
CONSTRAINT REST_ID_FK FOREIGN KEY (REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID),
CONSTRAINT BOOKING_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID)
);
SELECT * FROM MYPROJECT.DELIVERY_SERVICE;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.019 seconds

	service_id	rest_id	cust_id	booking_id	emp_id
1	10001	20001	30001	50001	40001
2	10002	20002	30002	50002	40002
3	10003	20003	30003	50003	40003
4	10004	20004	30004	50004	40004
5	10005	20005	30005	50005	40005
6	10006	20006	30006	50006	40006

Table 2: Restaurant

```sql
CREATE TABLE MYPROJECT.RESTAURANT(
REST_ID INT NOT NULL,
REST_NAME VARCHAR2(30) NOT NULL,
REST_TYPE VARCHAR2(30),
LOCATION VARCHAR2(30) NOT NULL,
CONSTRAINT REST_ID PRIMARY KEY(REST_ID)
);
SELECT * FROM MYPROJECT.RESTAURANT;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.013 seconds

	rest_id	rest_name	rest_type	location
1	20001	Yummy Kitchen	Ethnic	251 IUA Dallas
2	20002	Green Chilly	Osteria	012 HUH Plano
3	20003	Big Balls	Dine casual	311 FAM Irving
4	20004	Manis BBQ	Fast Food	872 FGC Commerce
5	20005	Coca Foods	Coffee hub	280 MNC Houston
6	20006	Burrito	Pizza	471 GreenVille
7	20007	Wild Dine	Dark	547 Garland

Table 3:Feedback

```sql
CREATE TABLE MYPROJECT.FEEDBACK(
FEEDBACK_ID INT NOT NULL,
COMMENTS VARCHAR2(50),
CUST_ID INT,
REST_ID INT,
CONSTRAINT FEEDBACK_ID PRIMARY KEY(FEEDBACK_ID),
CONSTRAINT FEEDBACK_CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON  DELETE CASCADE,
CONSTRAINT FEEDBACK_REST_ID_FK FOREIGN KEY(REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID) ON DELETE CASCADE
);

SELECT * FROM MYPROJECT.FEEDBACK;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.024 seconds

	feedback_id	comments	cust_id	rest_id
1	90001	Good Quality	30001	20001
2	90002	wonderful meal	30003	20002
3	90003	Loves Food	30006	20002
4	90004	Not Tastey	30002	20004
5	90005	Fresh Food	30003	20004
6	90006	it was delicious.	30004	20003
7	90007	Ambience is good	30005	20006

Table 4:Employee

```sql
CREATE TABLE MYPROJECT. EMPLOYEE(
EMP_ID INT NOT NULL,
EMP_NAME VARCHAR2(30) NOT NULL,
PH_NUMBER INT NOT NULL,
JOB_ROLE VARCHAR2(30),
EMP_TYPE VARCHAR2(30),
CUST_ID INT,
MGR_ID INT,
CONSTRAINT EMP_ID_PK PRIMARY KEY(EMP_ID),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON DELETE SET NULL,
CONSTRAINT MGR_FK FOREIGN KEY(MGR_ID) REFERENCES MYPROJECT.EMPLOYEE(EMP_ID)
);

SELECT * FROM MYPROJECT.EMPLOYEE;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.025 seconds

	emp_id	emp_name	ph_number	job_role	emp_type	cust_id	mgr_id
1	40002	Gary	9708955873	Manager	Part_Time	30005	40002
2	40004	Maina	3263035871	Manager	Full_Time	30007	(null)
3	40005	Dany	4863035709	manager	Full_Time	30001	40004
4	40006	Josh	3790350232	Delivery	Part_Time	30001	40004
5	40001	Samy	9863035870	Delivery	Full_Time	30004	40004

Table 5:Customer

```sql
CREATE TABLE MYPROJECT.CUSTOMER(
CUST_ID INT NOT NULL,
CUST_NAME VARCHAR2(30) NOT NULL,
PH_NUMBER INT NOT NULL,
ADDRESS VARCHAR2(50),
EMAIL_ID VARCHAR2(50),
DOB DATE,
CONSTRAINT CUST_ID_PK PRIMARY KEY(CUST_ID)
);

SELECT * FROM MYPROJECT.CUSTOMER;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.016 seconds

	cust_id	cust_name	ph_number	address	email_id	dob
1	30001	Binny	9708955873	121 FG Commerce	binny@gmail.com	10/27/95 12:00:00 ...
2	30002	Kalvin	5862222212	435 HFDUF Plano	Kalvin@yahoo.com	12/07/89 12:00:00 ...
3	30003	Macon	4743035872	1200 Mckinney	Macon@yahoo.com	01/20/60 12:00:00 ...
4	30004	Rafael	3263035871	901 Dallas	Rafael@gmail.com	02/14/92 12:00:00 ...
5	30005	Vanya	4863035709	234 VC Commerce	Vanya@gmail.com	07/04/01 12:00:00 ...
6	30006	Xavier	3790350232	805 DCM Garland	Xavier@yahoo.com	11/14/95 12:00:00 ...
7	30007	Jackie	7863038079	602 BCO Houston	Jackie@gmail.com	04/06/60 12:00:00

Table 6:Menu

```sql
CREATE TABLE MYPROJECT.MENU(
ITEM_ID INT NOT NULL,
ITEM_NAME VARCHAR2(100),
PRICE FLOAT,
REST_ID INT,
INGREDIENTS VARCHAR2(256),
CONSTRAINT ITEM_ID PRIMARY KEY(ITEM_ID),
CONSTRAINT REST_ID_FK FOREIGN KEY(REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID) ON DELETE SET NULL
);
SELECT * FROM MYPROJECT.MENU;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.016 seconds

	item_id	item_name	price	rest_id	ingredients
1	80001	Becon	3.5	20001	LETTUCE
2	80002	Chicken Roast	3.5	20001	TORTILLA BOWL WI...
3	80003	Chicken Wings	3.5	20001	GUACAMOLLE
4	80004	Mutton Balls	3.5	20001	julienned VEGETABL...
5	80005	Chicken Pizza	3.5	20002	GREEN GRAM
6	80006	Veg Pizza	3.5	20002	MEAT
7	80007	Mutton Pizza	4	20002	VEGGIES

Table 7:Delivery_Vehicle

```
CREATE TABLE MYPROJECT.DELIVERY_VEHICLE(
VEHICLE_ID INT NOT NULL,
VEHICLE_TYPE VARCHAR2(30),
EMP_ID INT,
CONSTRAINT VEHICLE_ID_PK PRIMARY KEY(VEHICLE_ID),
CONSTRAINT EMP_ID_FK FOREIGN KEY(EMP_ID) REFERENCES MYPROJECT.EMPLOYEE(EMP_ID) ON DELETE SET NULL
);

SELECT * FROM MYPROJECT.DELIVERY_VEHICLE;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.003 seconds

	feedback_id	comments	cust_id	rest_id
1	90001	Good Quality	30001	20001
2	90002	wonderful meal	30003	20002
3	90003	Loves Food	30006	20002
4	90004	Not Tastey	30002	20004
5	90005	Fresh Food	30003	20004
6	90006	it was delicious.	30004	20003
7	90007	Ambience is good	30005	20006

Table 8:Order_List

```
 CREATE TABLE ORDER_lIST(
LIST_ID INT,
ITEM_QUANTITY INT,
AMOUNT FLOAT,
ITEM_ID INT,
BOOKING_ID INT,
CONSTRAINT LIST_ID PRIMARY KEY(LIST_ID),
CONSTRAINT ORDER_ID_FK FOREIGN KEY(ITEM_ID) REFERENCES MYPROJECT.MENU(ITEM_ID),
CONSTRAINT ORDERS_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID) ON  DELETE CASCADE
);

SELECT * FROM MYPROJECT.ORDER_lIST;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.004 seconds

	list_id	item_quantity	amount	item_id	booking_id
1	200	4	30	80010	50004
2	201	3	40	80010	50003
3	202	2	50	80010	50002
4	203	2	50	80012	50001
5	204	2	50	80010	50005
6	205	2	50	80012	50006

Table 9:Booking

```
CREATE TABLE MYPROJECT.BOOKING(
BOOKING_ID INT,
AMOUNT FLOAT,
BOOKING_DATE DATE,
CUST_ID INT,
OFFER_CODE VARCHAR2(30),
STATUS VARCHAR2(50),
CONSTRAINT BOOKING_BOOKING_ID PRIMARY KEY(BOOKING_ID),
CONSTRAINT BOOKING_CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON DELETE SET NULL,
CONSTRAINT BOOKING_OFFER_CODE_FK FOREIGN KEY(OFFER_CODE) REFERENCES MYPROJECT.OFFER(OFFER_CODE) ON DELETE SET NULL
);
```

Script Output × Query Result ×

SQL | All Rows Fetched: 8 in 0.349 seconds

	BOOKING_ID	AMOUNT	CUST_ID	OFFER_CODE	STATUS	BOOKING_DATE
1	50008	10	30007	(null)	CANCELLED	07-APR-20
2	50001	100	30002	FOODDAY12893	BOOKING CONFIRMED	27-MAR-20
3	50002	300	30005	JSMSXZ009	BOOKING CONFIRMED	14-FEB-20
4	50003	250	30007	(null)	BOOKING FAILED	07-JAN-20
5	50004	150	30004	YUMMYIES0033	(null)	18-JUL-20
6	50005	110	30005	(null)	CANCELLED	19-AUG-20
7	50006	210	30006	JACCOKE1234	BOOKING FAILED	15-SEP-20
8	50007	180	30007	(null)	CANCELLED	27-OCT-20

Table10 :Offer

```
CREATE TABLE MYPROJECT. OFFER(
OFFER_CODE VARCHAR2(30) NOT NULL,
OFFER_TYPE VARCHAR2(30),
DISC_AMOUNT FLOAT,
VALIDITY DATE NOT NULL,
CONSTRAINT OFFER_CODE PRIMARY KEY(OFFER_CODE)
);

SELECT * FROM MYPROJECT.OFFER;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.003 seconds

	offer_code	offer_type	disc_amount	validity
1	JSMSXZ009	Seasonal	26.5	05/07/20 12:00:00 ...
2	YUMMYIES0033	Senior citizen disco...	50	06/01/20 12:00:00 ...
3	JACCOKE1234	Trail Offer	5	04/30/20 12:00:00 ...
4	FOODDAY12893	Happy_Hour	2.5	03/07/20 12:00:00 ...

Table 11:Payment

```
CREATE TABLE MYPROJECT. PAYMENT(
PAY_ID INT,
PAY_TYPE VARCHAR2(30),
AMOUNT FLOAT,
BOOKING_ID INT,
CONSTRAINT PAY_ID PRIMARY KEY(PAY_ID),
CONSTRAINT PAYMENT_BOOKING_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID) ON DELETE SET NULL
);


SELECT * FROM MYPROJECT.PAYMENT;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.069 seconds

	pay_id	pay_type	amount	booking_id
1	70001	CREDIT	45	50004
2	70002	DEBIT	40.5	50005
3	70003	PAYPAL	60.6	50001
4	70004	CASH	70.1	50002
5	70005	DEBIT	90.4	50006
6	70006	CREDIT	50	50007
7	70007	DEBIT	30.6	50003

Table 12:Emp_Delivery

```
CREATE TABLE MYPROJECT.Emp_Delivery(
EMP_ID INT,
SERVICE_ID INT,
DELIVERIES NUMBER,
CONSTRAINT DELIVERIES_PK PRIMARY KEY(EMP_ID,SERVICE_ID),
CONSTRAINT DELIVERIES_SERVICE_FK FOREIGN KEY(SERVICE_ID) REFERENCES DELIVERY_SERVICE(SERVICE_ID),
CONSTRAINT DELIVERIES_EMPLOYEE_FK FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE(EMP_ID));

SELECT * FROM MYPROJECT.DELIVERY_VEHICLE;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Data Loading

Download ▾ Execution time: 0.024 seconds

	vehicle_id	vehicle_type	emp_id
1	60001	Bike	40001
2	60002	Car	40003
3	60003	Car	40002
4	60004	Truck	40006
5	60005	Bike	40007
6	60006	Truck	40004
7	60007	Car	40005

Table 13:Nutrition_Fact

```
CREATE TABLE NUTRITION_FACT(
NF_ID INT,
ITEM_ID INT,
CALORIES INT,
VITAMIN CHAR,
CONSTRAINT NF_ID PRIMARY KEY(NF_ID),
CONSTRAINT NF_ID_FK FOREIGN KEY(ITEM_ID) REFERENCES MYPROJECT.MENU(ITEM_ID));

SELECT * FROM MYPROJECT.NUTRITION_FACT;
```

Query Result | Script Output | DBMS Output | Explain Plan | Autotrace | SQL History | Da

Download ▼ Execution time: 0.022 seconds

	nf_id	item_id	calories	vitamin
1	100	80001	100	A
2	102	80016	120	E
3	103	80015	10	D
4	104	80013	50	C
5	105	80011	150	A
6	106	80010	110	K
7	101	80013	70	B

7) Discuss database normalization rules on your tables. Do not explain what the rules are. Just check if each of the tables satisfies 1NF, 2NF, 3NF and BCNF. If not, normalize your tables.

Database Normalization rules on Tables:

This project violates Normalization rules in the table Delivery_service.Booking ,Vehicle information is repeated for each Employee.This violates 1st Normal Form .Primary key must uniquely identify attribute value .So, Booking and Vehicle Tables were formed.If we delete some critical information we may lose.

Emp_name is not fully dependent on Vehicle_Id,Booking_Id 2nd Normal Form violated. So ,an Identified new table Employee is formed.

Ingredients,Nutrition_Fact dependent on Item_Name non-prime attribute dependent on non-prime attribute violates third normal form,in order to remove the Transitive dependency Menu table was formed.Updating Item_Name forms inconsistent in data. New table with removed dependency was created.

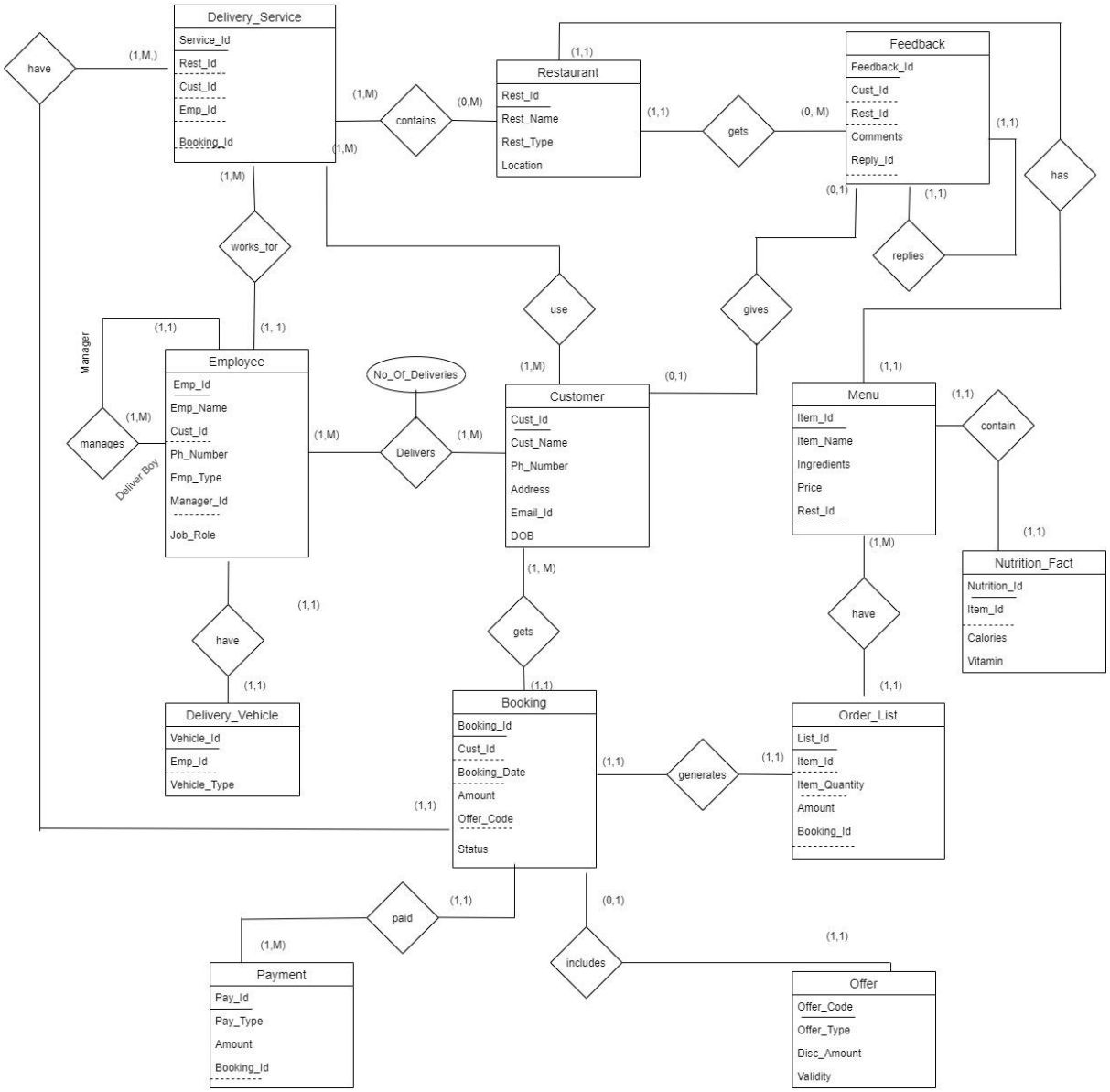
Category Table we see that not all attributes are fully dependent on the primary key.The only attribute that is fully dependent is Item_Name.

Amount is Fully dependent on Item_Quality ,Ingredients,Nutition_Fact is fully dependent on Item_Name and Item_Quality.These attributes were non-prime attributes So Menu Table is formed.Menu Table each non-key attribute i.e., Nutrition_Fact is not fully dependent on Primary Key(Item_Id). Here,the Menu table violates BCNF as every attribute should be a candidate key, so a new table (Nutition_Fact) is identified.

8) Show the latest table/ER design after normalization.

Food Delivery Service

(After Normalization)



9) State an imaginary cascade deletion rule on any table(s). (Not a SQL query required, only give an imaginary example, and describe the reason behind it.)

If Restaurant is removed due to some loss the corresponding feedback comments related to Restaurant will be deleted.i..e In This project If the Restaurant Primary Key is deleted then Corresponding Feedback Id will be deleted because Rest_Id is foreigh key in Feedback table.

10) Create tables using SQL commands. Think about the nature of your scenario to decide about NULL attributes.

```sql
/*createQueries*/
/*customer*/

CREATE TABLE MYPROJECT.CUSTOMER(
CUST_ID INT NOT NULL,
CUST_NAME VARCHAR2(30) NOT NULL,
PH_NUMBER INT NOT NULL,
ADDRESS VARCHAR2(50),
EMAIL_ID VARCHAR2(50),
DOB DATE,
CONSTRAINT CUST_ID_PK PRIMARY KEY(CUST_ID)
);

/*BOOKING_QUERIES*/

CREATE TABLE MYPROJECT.BOOKING(
BOOKING_ID INT,
AMOUNT FLOAT,
BOOKING_DATE DATE,
CUST_ID INT,
OFFER_CODE VARCHAR2(30),
STATUS VARCHAR2(50),
CONSTRAINT BOOKING_BOOKING_ID PRIMARY KEY(BOOKING_ID),
CONSTRAINT BOOKING_CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON DELETE SET NULL,
CONSTRAINT BOOKING_OFFER_CODE_FK FOREIGN KEY(OFFER_CODE) REFERENCES MYPROJECT.OFFER(OFFER_CODE) ON DELETE SET NULL
);

/*Emp_Delivery*/

CREATE TABLE MYPROJECT.Emp_Delivery(
EMP_ID INT,
SERVICE_ID INT,
DELIVERIES NUMBER,
CONSTRAINT DELIVERIES_PK PRIMARY KEY(EMP_ID,SERVICE_ID),
CONSTRAINT DELIVERIES_SERVICE_FK FOREIGN KEY(SERVICE_ID) REFERENCES DELIVERY_SERVICE(SERVICE_ID),
CONSTRAINT DELIVERIES_EMPLOYEE_FK FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE(EMP_ID));

/*DELIVERY_VEHICLE*/
CREATE TABLE MYPROJECT.DELIVERY_VEHICLE(
VEHICLE_ID INT NOT NULL,
VEHICLE_TYPE VARCHAR2(30),
EMP_ID INT,
CONSTRAINT VEHICLE_ID_PK PRIMARY KEY(VEHICLE_ID),
CONSTRAINT EMP_ID_FK FOREIGN KEY(EMP_ID) REFERENCES MYPROJECT.EMPLOYEE(EMP_ID) ON DELETE SET NULL
);
/*FEEDBACK*/
CREATE TABLE MYPROJECT.FEEDBACK(
FEEDBACK_ID INT NOT NULL,
COMMENTS VARCHAR2(50),
CUST_ID INT,
REST_ID INT,
REPLY_ID INT,
CONSTRAINT FEEDBACK_ID PRIMARY KEY(FEEDBACK_ID),
CONSTRAINT FEEDBACK_CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON  DELETE CASCADE,
CONSTRAINT FEEDBACK_REST_ID_FK FOREIGN KEY(REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID) ON DELETE CASCADE,
CONSTRAINT FEEDBACK_REP_ID_FK FOREIGN KEY(REPLY_ID) REFERENCES MYPROJECT.FEEDBACK(FEEDBACK_ID) ON DELETE CASCADE
);

/*ORDER_LIST*/
CREATE TABLE ORDER_LIST(
LIST_ID INT,
ITEM_QUANTITY INT,
AMOUNT FLOAT,
ITEM_ID INT,
BOOKING_ID INT,
CONSTRAINT LIST_ID PRIMARY KEY(LIST_ID),
CONSTRAINT ORDER_ID_FK FOREIGN KEY(ITEM_ID) REFERENCES MYPROJECT.MENU(ITEM_ID),
CONSTRAINT ORDERS_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID) ON  DELETE CASCADE
);
```

```sql
/*NUTRITION_FACT*/
CREATE TABLE NUTRITION_FACT(
NF_ID INT,
ITEM_ID INT,
CALORIES INT,
VITAMIN CHAR,
CONSTRAINT NF_ID PRIMARY KEY(NF_ID),
CONSTRAINT NF_ID_FK FOREIGN KEY(ITEM_ID) REFERENCES MYPROJECT.MENU(ITEM_ID));

/*MENU*/

CREATE TABLE MYPROJECT.MENU(
ITEM_ID INT NOT NULL,
ITEM_NAME VARCHAR2(100),
PRICE FLOAT,
REST_ID INT,
INGREDIENTS VARCHAR2(256),
CONSTRAINT ITEM_ID PRIMARY KEY(ITEM_ID),
CONSTRAINT REST_ID_FK FOREIGN KEY(REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID) ON DELETE SET NULL
);

CREATE TABLE MYPROJECT. PAYMENT(
PAY_ID INT,
PAY_TYPE VARCHAR2(30),
AMOUNT FLOAT,
BOOKING_ID INT,
CONSTRAINT PAY_ID PRIMARY KEY(PAY_ID),
CONSTRAINT PAYMENT_BOOKING_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID) ON DELETE SET NULL
);
CREATE TABLE MYPROJECT.DELIVERY_SERVICE(
SERVICE_ID INT NOT NULL,
REST_ID INT NOT NULL,
CUST_ID INT,
EMP_ID INT,
BOOKING_ID INT,
CONSTRAINT SERVICE_ID_PK PRIMARY KEY(SERVICE_ID),
CONSTRAINT REST_ID_FK FOREIGN KEY (REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID),
CONSTRAINT BOOKING_ID_FK FOREIGN KEY(BOOKING_ID) REFERENCES MYPROJECT.BOOKING(BOOKING_ID)
```

```sql
CREATE TABLE MYPROJECT.RESTAURANT(
REST_ID INT NOT NULL,
REST_NAME VARCHAR2(30) NOT NULL,
REST_TYPE VARCHAR2(30),
LOCATION VARCHAR2(30) NOT NULL,
CONSTRAINT REST_ID PRIMARY KEY(REST_ID)
);

CREATE TABLE MYPROJECT. OFFER(
OFFER_CODE VARCHAR2(30) NOT NULL,
OFFER_TYPE VARCHAR2(30),
DISC_AMOUNT FLOAT,
VALIDITY DATE NOT NULL,
CONSTRAINT OFFER_CODE PRIMARY KEY(OFFER_CODE)
);
CREATE TABLE MYPROJECT. EMPLOYEE(
EMP_ID INT NOT NULL,
EMP_NAME VARCHAR2(30) NOT NULL,
PH_NUMBER INT NOT NULL,
JOB_ROLE VARCHAR2(30),
EMP_TYPE VARCHAR2(30),
CUST_ID INT,
MGR_ID INT,
CONSTRAINT EMP_ID_PK PRIMARY KEY(EMP_ID),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON DELETE SET NULL,
CONSTRAINT MGR_FK FOREIGN KEY(MGR_ID) REFERENCES MYPROJECT.EMPLOYEE(EMP_ID)
);
```

Here few Tables that accepts null:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	EMP_ID	NUMBER(38,0)	No	(null)	1	(null)
2	EMP_NAME	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	PH_NUMBER	NUMBER(38,0)	No	(null)	3	(null)
4	JOB_ROLE	VARCHAR2(30 BYTE)	Yes	(null)	4	(null)
5	EMP_TYPE	VARCHAR2(30 BYTE)	Yes	(null)	5	(null)
6	CUST_ID	NUMBER(38,0)	Yes	(null)	6	(null)
7	MGR_ID	NUMBER(38,0)	Yes	(null)	7	(null)

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CUST_ID	NUMBER(38,0)	No	(null)	1	(null)
2	CUST_NAME	VARCHAR2(30 BYTE)	No	(null)	2	(null)
3	PH_NUMBER	NUMBER(38,0)	No	(null)	3	(null)
4	ADDRESS	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	EMAIL_ID	VARCHAR2(50 BYTE)	Yes	(null)	5	(null)
6	DOB	DATE	Yes	(null)	6	(null)

11) Insert at least five (5) records into each table.

```
CREATE TABLE MYPROJECT. EMPLOYEE(
EMP_ID INT NOT NULL,
EMP_NAME VARCHAR2(30) NOT NULL,
PH_NUMBER INT NOT NULL,
JOB_ROLE VARCHAR2(30),
EMP_TYPE VARCHAR2(30),
CUST_ID INT,
MGR_ID INT,
CONSTRAINT EMP_ID_PK PRIMARY KEY(EMP_ID),
CONSTRAINT CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON DELETE SET NULL,
CONSTRAINT MGR_FK FOREIGN KEY(MGR_ID) REFERENCES MYPROJECT.EMPLOYEE(EMP_ID)
);
```

```sql
/*EMPLOYEEInsertQuery*/

INSERT INTO  EMPLOYEE VALUES(40001,'Samy',9863035870,'Delivery','Full_Time',30004, 40004);
INSERT INTO  EMPLOYEE VALUES(40002,'Gary',9708955873,'Manager','Part_Time',30005, 40002);
INSERT INTO  EMPLOYEE VALUES(40003,'Tena',4743035872,'Delivery','Part_Time',30006, 40004);
INSERT INTO  EMPLOYEE VALUES(40004,'Maina',3263035871,'Manager','Full_Time',30007, NULL);
INSERT INTO  EMPLOYEE VALUES(40005,'Dany',4863035709,'manager','Full_Time', 30001,40004);
INSERT INTO  EMPLOYEE VALUES(40006,'Josh',3790350232,'Delivery','Part_Time',30001, 40004);
INSERT INTO  EMPLOYEE VALUES(40007,'Anil',7863038079,'Delivery','Part_Time',30001, 40001);

/*CUSTOMERInsertQuery*/

INSERT INTO  CUSTOMER VALUES(30001, 'Binny', 9708955873, '121 FG Commerce', 'binny@gmail.com', '27 oct 1995');
INSERT INTO  CUSTOMER VALUES(30002, 'Kalvin',5862222212, '435 HFDUF Plano ', 'Kalvin@yahoo.com', '07 dec 1889');
INSERT INTO  CUSTOMER VALUES(30003, 'Macon', 4743035872, '1200 Mckinney', 'Macon@yahoo.com','20 jan 1960');
INSERT INTO  CUSTOMER VALUES(30004, 'Rafael',3263035871, '901 Dallas', 'Rafael@gmail.com', '14 feb 1992');
INSERT INTO  CUSTOMER VALUES(30005, 'Vanya', 4863035709, '234 VC Commerce', 'Vanya@gmail.com', '04 july 2001');
INSERT INTO  CUSTOMER VALUES(30006, 'Xavier', 3790350232, '805 DCM Garland', 'Xavier@yahoo.com', '14 nov 1995');
INSERT INTO  CUSTOMER VALUES(30007, 'Jackie', 7863038079, '602 BCO Houston', 'Jackie@gmail.com','06 apr 1960');

/*DELIVERY_SERVICEInsertQuery*/

INSERT INTO  DELIVERY_SERVICE VALUES(10001, 'FOODIES GROUP', 20001, 30001, 50001,40001);
INSERT INTO DELIVERY_SERVICE VALUES(10002, 'FOODIES GROUP', 20002, 30002,  50002,40002);
INSERT INTO DELIVERY_SERVICE VALUES(10003, 'FOODIES GROUP', 20003, 30003,  50003,40003);
INSERT INTO DELIVERY_SERVICE VALUES(10004, 'FOODIES GROUP', 20004, 30004,  50004,40004);
INSERT INTO DELIVERY_SERVICE VALUES(10005, 'FOODIES GROUP', 20005, 30005,  50005,40005);
INSERT INTO DELIVERY_SERVICE VALUES(10006, 'FOODIES GROUP', 20006, 30006,  50006,40006);
INSERT INTO DELIVERY_SERVICE VALUES(10007, 'FOODIES GROUP', 20007, 30007,  50007,40007);

/*RESTAURANTInsertQuery*/

INSERT INTO  RESTAURANT VALUES(20001, 'Yummy Kitchen', 'Ethnic', '251 IUA Dallas',null);
INSERT INTO  RESTAURANT VALUES(20002, 'Green Chilly', 'Osteria', '012 HUH Plano',null);
INSERT INTO  RESTAURANT VALUES(20003, 'Big Balls', 'Dine casual', '311 FAM Irving',null);
INSERT INTO  RESTAURANT VALUES(20004, 'Manis BBQ', 'Fast Food', '872 FGC Commerce',null);
INSERT INTO  RESTAURANT VALUES(20005, 'Coca Foods', 'Coffee hub', '280 MNC Houston',null);
INSERT INTO  RESTAURANT VALUES(20006, 'Burrito', 'Pizza', '471 GreenVille',null);
INSERT INTO  RESTAURANT VALUES(20007, 'Wild Dine', 'Dark', '547 Garland',null);
```

```sql
/*BOOKINGInsertQuery*/
INSERT INTO  BOOKING VALUES(50001,4.5,30002,   'FOODDAY12893','BOOKING CONFIRMED','27 MARCH 2020');
INSERT INTO  BOOKING VALUES(50002,5.5,30005,   'JSMSXZ009','BOOKING CONFIRMED','14 FEB 2020');
INSERT INTO  BOOKING VALUES(50003,6, 30007,     '',' BOOKING FAILED','07 JAN 2020');
INSERT INTO  BOOKING VALUES(50004,7, 30004,    'YUMMYIES0033','','18 JULY 2020');
INSERT INTO  BOOKING VALUES(50005,7, 30005,     '','CANCELLED','19 AUG 2020');
INSERT INTO  BOOKING VALUES(50006,3.5,30006,    'JACCOKE1234','BOOKING FAILED','15 SEP 2020');
INSERT INTO  BOOKING VALUES(50007,10,30007,    '','CANCELLED','27 OCT 2020');
INSERT INTO  BOOKING VALUES(50008,10,30007,    '','CANCELLED','07 APRIL 2020');

/*MENUInsertQuery*/

INSERT INTO  MENU VALUES(80001, 'Becon',  3.5, 20001);
INSERT INTO  MENU VALUES(80002, 'Chicken Roast',  3.5, 20001);
INSERT INTO  MENU VALUES(80003, 'Chicken Wings',  3.5, 20001);
INSERT INTO  MENU VALUES(80004, 'Mutton Balls',  3.5, 20001);
INSERT INTO  MENU VALUES(80005, 'Chicken Pizza',  3.5, 20002);
INSERT INTO  MENU VALUES(80006, 'Veg Pizza',  3.5, 20002);
INSERT INTO  MENU VALUES(80007, 'Mutton Pizza',  4, 20002);
INSERT INTO  MENU VALUES(80008, 'Mushroom Pizza',  3.5, 20003);
INSERT INTO  MENU VALUES(80009, 'Mushroom Dry',  3.5, 20003);
INSERT INTO  MENU VALUES(80010, 'Chicken Pepper Fry',  4, 20003);
INSERT INTO  MENU VALUES(80011, 'BBQ Chicken',  5, 20004);
INSERT INTO  MENU VALUES(80012, 'Biriyani ',  8, 20004);
INSERT INTO  MENU VALUES(80013, 'Dosa',  4, 20004);
INSERT INTO  MENU VALUES(80014, 'Meat Sauce',  4, 20004);
INSERT INTO  MENU VALUES(80015, 'Veg Palav',  4, 20004);
INSERT INTO  MENU VALUES(80016, 'Jeera Rice',  4, 20004);
```

```sql
/*OFFERInsertQuery*/

INSERT INTO OFFER VALUES('FOODDAY12893','Happy_Hour', 2.5, '07 mar 2020');
INSERT INTO OFFER VALUES('JSMSXZ009','Seasonal', 26.5, '07 may 2020');
INSERT INTO OFFER VALUES('YUMMYIES0033', 'Senior citizen discount', 50, '01 jun 2020');
INSERT INTO OFFER VALUES('JACCOKE1234', 'Trail Offer', 5, '30 apr 2020');
INSERT INTO OFFER VALUES('HAPPY10000', 'FREE DELIVERY', 5, '30 JUN 2020');

/*DELIVERY_VEHICLEInsertQuery*/

INSERT INTO  DELIVERY_VEHICLE VALUES(60001, 'Bike' ,40001);
INSERT INTO  DELIVERY_VEHICLE VALUES(60002, 'Car'  ,40003);
INSERT INTO  DELIVERY_VEHICLE VALUES(60003, 'Car'  ,40002);
INSERT INTO  DELIVERY_VEHICLE VALUES(60004, 'Truck' ,40006);
INSERT INTO  DELIVERY_VEHICLE VALUES(60005, 'Bike'  ,40007);
INSERT INTO  DELIVERY_VEHICLE VALUES(60006, 'Truck' ,40004);
INSERT INTO  DELIVERY_VEHICLE VALUES(60007, 'Car'   ,40005);

/*FEEDBACKInsertQuery*/

INSERT INTO  FEEDBACK VALUES(90001,  'Good Quality',        30001, 20001,90001);
INSERT INTO  FEEDBACK VALUES(90002,  'wonderful meal',          30003, 20002,90007);
INSERT INTO  FEEDBACK VALUES(90003,  'Loves Food',  30006, 20002,90006);
INSERT INTO  FEEDBACK VALUES(90004,  'Not Tastey',      30002, 20004,90005);
INSERT INTO  FEEDBACK VALUES(90005,  'Fresh Food',              30003, 20004,90003);
INSERT INTO  FEEDBACK VALUES(90006,  'it was delicious.',           30004, 20003,90002);
INSERT INTO  FEEDBACK VALUES(90007,  'Ambience is good',    30005, 20006,90001);

/*ORDER_lISTInsertQuery*/

INSERT INTO  ORDER_lIST VALUES(200,4,30,80010 ,50004);
INSERT INTO  ORDER_lIST VALUES(201,3,40,80010 ,50003);
INSERT INTO  ORDER_lIST VALUES(202,2,50,80010 ,50002);
INSERT INTO  ORDER_lIST VALUES(203,2,50,80012 ,50001);
INSERT INTO  ORDER_lIST VALUES(204,2,50,80010 ,50005);
INSERT INTO  ORDER_lIST VALUES(205,2,50,80012 ,50006);
```

```sql
/*BOOKINGInsertQuery*/

INSERT INTO  BOOKING VALUES(50001,4.5,30002,   'FOODDAY12893','BOOKING CONFIRMED','');
INSERT INTO  BOOKING VALUES(50002,5.5,30005,   'JSMSXZ009','BOOKING CONFIRMED');
INSERT INTO  BOOKING VALUES(50003,6, 30007,     '',' BOOKING FAILED');
INSERT INTO  BOOKING VALUES(50004,7, 30004,    'YUMMYIES0033','');
INSERT INTO  BOOKING VALUES(50005,7, 30005,     '','CANCELLED');
INSERT INTO  BOOKING VALUES(50006,3.5,30006,    'JACCOKE1234','BOOKING FAILED');
INSERT INTO  BOOKING VALUES(50007,10,30007,    '','CANCELLED');
INSERT INTO  BOOKING VALUES(50008,10,30007,    '','CANCELLED');

/*NUTRITION_FACTInsertQuery*/

INSERT INTO  NUTRITION_FACT VALUES(100,80001 ,100,'A');
INSERT INTO  NUTRITION_FACT VALUES(101,80013 ,70,'B');
INSERT INTO  NUTRITION_FACT VALUES(102,80016 ,120,'E');
INSERT INTO  NUTRITION_FACT VALUES(103,80015 ,10,'D');
INSERT INTO  NUTRITION_FACT VALUES(104,80013 ,50,'C');
INSERT INTO  NUTRITION_FACT VALUES(105,80011 ,150,'A');
INSERT INTO  NUTRITION_FACT VALUES(106,80010 ,110,'K');

/*EMP_DELIVERYInsertQuery*/

INSERT INTO  EMP_DELIVERY VALUES(40001,10001  ,4);
INSERT INTO  EMP_DELIVERY VALUES(40005,10003  ,4);
INSERT INTO  EMP_DELIVERY VALUES(40004,10004 ,4);
INSERT INTO  EMP_DELIVERY VALUES(40006,10005  ,4);
INSERT INTO  EMP_DELIVERY VALUES(40002,10006  ,4);
INSERT INTO  EMP_DELIVERY VALUES(40003,10002  ,4);

/*PAYMENT*/

INSERT INTO  PAYMENT VALUES(70001, 'CREDIT',  45, 50004);
INSERT INTO  PAYMENT VALUES(70002, 'DEBIT',  40.5, 50005);
INSERT INTO  PAYMENT VALUES(70003,'PAYPAL',  60.6, 50001);
INSERT INTO  PAYMENT VALUES(70004,'CASH',  70.1, 50002);
INSERT INTO  PAYMENT VALUES(70005, 'DEBIT', 90.4 , 50006);
INSERT INTO  PAYMENT VALUES(70006, 'CREDIT', 50, 50007);
INSERT INTO  PAYMENT VALUES(70007, 'DEBIT',  30.6, 50003);
```

12) List only primary keys values for three different tables. (Select command on three different tables. No join operation is required.)

```
SELECT OFFER_CODE FROM OFFER;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 6 in 0

	OFFER_CODE
1	FOODDAY12893
2	HAPPY10000
3	HAPPYHOUR
4	JACCOKE1234
5	JSMSXZ009
6	YTMMXTES0033

```
SELECT NF_ID FROM NUTRITION_FACT;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 7 in 0.344 sec

	NF_ID
1	100
2	101
3	102
4	103
5	104
6	105
7	106

```
SELECT PAY_ID FROM PAYMENT;
```

Script Output × Query Result ×

SQL | All Rows Fetched: 7 in (

	PAY_ID
1	70001
2	70002
3	70003
4	70004
5	70005
6	70006
7	70007

```
SELECT P.PAY_ID,
O.OFFER_CODE,
N.NF_ID FROM PAYMENT P
,OFFER O,
NUTRITION_FACT N;
```

Script Output × | Query Result ×

SQL | Fetched 100 rows in 0.642 sec

	PAY_ID	OFFER_CODE	NF_ID
37	70006	FOODDAY12893	101
38	70006	FOODDAY12893	102
39	70006	FOODDAY12893	103
40	70006	FOODDAY12893	104
41	70006	FOODDAY12893	105
42	70006	FOODDAY12893	106
43	70007	FOODDAY12893	100
44	70007	FOODDAY12893	101
45	70007	FOODDAY12893	102
46	70007	FOODDAY12893	103
47	70007	FOODDAY12893	104
48	70007	FOODDAY12893	105
49	70007	FOODDAY12893	106
50	70001	HAPPY10000	100
51	70001	HAPPY10000	101
52	70001	HAPPY10000	102
53	70001	HAPPY10000	103
54	70001	HAPPY10000	104
55	70001	HAPPY10000	105
56	70001	HAPPY10000	106
57	70002	HAPPY10000	100
58	70002	HAPPY10000	101
59	70002	HAPPY10000	102
60	70002	HAPPY10000	103
61	70002	HAPPY10000	104
62	70002	HAPPY10000	105

13) Demonstrate two (2) SELECT commands with WHERE statement.

Listing all the Customers ID's and Names who uses offer code 'JACCOKE1234'

```
SELECT C.CUST_ID,C.CUST_NAME FROM CUSTOMER C,OFFER O,BOOKING B
WHERE (O.OFFER_CODE='JACCOKE1234' AND B.CUST_ID=C.CUST_ID);
```

ript Output × Query Result ×

SQL | All Rows Fetched: 7 in 0.294 seconds

	CUST_ID	CUST_NAME
1	30002	Kalvin
2	30005	Vanya
3	30007	Jackie
4	30004	Rafael
5	30005	Vanya
6	30006	Xavier
7	30007	Jackie

```
SELECT C.CUST_NAME,F.COMMENTS  FROM CUSTOMER C,FEEDBACK F
WHERE F.CUST_ID=C.CUST_ID;
```

cript Output × Query Result ×

SQL | All Rows Fetched: 7 in 0.274 seconds

	CUST_NAME	COMMENTS
1	Binny	Good Quality
2	Macon	wonderful meal
3	Xavier	Loves Food
4	Kalvin	Not Tastey
5	Macon	Fresh Food
6	Rafael	it was delicious.
7	Vanya	Ambience is good

Listing all the Customer name along with their Comments to the

14) Demonstrate two (2) SELECT commands with GROUP BY statement.

- Payment Type count used by Customers.
- Show number of Payment_Types used by Customers.

```
SELECT * FROM PAYMENT;

SELECT PAY_TYPE,COUNT(*) AS NO_OF_TYPES FROM PAYMENT GROUP BY PAY_TYPE;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 4 in 0.274 seconds

	PAY_TYPE	NO_OF_TYPES
1	DEBIT	3
2	PAYPAL	1
3	CASH	1
4	CREDIT	2

- Number of orders placed by each customer.

```
SELECT C.CUST_NAME , COUNT(*) AS NO_OF_ORDERS FROM CUSTOMER C,BOOKING B
WHERE B.CUST_ID=C.CUST_ID GROUP BY C.CUST_NAME;
```

ript Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.28 seconds

	CUST_NAME	NO_OF_ORDERS
1	Kalvin	1
2	Vanya	2
3	Jackie	3
4	Xavier	1
5	Rafael	1

15) Demonstrate two (2) SELECT commands with HAVING statement.

- To know which customer has more than or equal to two orders.
- Showing Item_Id along with Item_Name with more than 16 orders to know the item which have placed more than 16 times.

```
SELECT C.CUST_NAME , COUNT(*) AS NO_OF_ORDERS FROM CUSTOMER C,BOOKING B
WHERE B.CUST_ID=C.CUST_ID GROUP BY C.CUST_NAME HAVING COUNT(*)>=2;
```

ript Output × | Query Result ×

SQL | All Rows Fetched: 2 in 0.281 seconds

	CUST_NAME	NO_OF_ORDERS
1	Vanya	2
2	Jackie	3

- Item which have the highest number of orders.
- Listing Customers who have placed orders more than or equal to 2

```
SELECT O.ITEM_ID,M.ITEM_NAME, COUNT(*) COUNT_OF_ORDERS FROM MENU M,ORDER_1IST O,BOOKING B
WHERE(O.ITEM_ID=M.ITEM_ID )
GROUP BY O.ITEM_ID ,M.ITEM_NAME HAVING COUNT(*)>16;
```

ript Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.303 seconds

	ITEM_ID	ITEM_NAME	COUNT_OF_ORDERS
1	80010	Chicken Pepper Fry	32

16) Using two related tables (meaning logically connected with primary-key and foreign-key pairs), run an inner join statement to show matching rows. For instance, assume that Table A and Table B have 4 and 5 attributes respectively. Also, assume that Table A's primary key is seen as foreign key in Table B. Use join operations to show matching rows whose primary key and foreign key is the same.

Showing all Restaurant Names who has comments which includes Food in it.

```
SELECT R.REST_NAME,F.COMMENTS FROM RESTAURANT R
INNER JOIN
FEEDBACK F ON
F.REST_ID=R.REST_ID
WHERE COMMENTS LIKE '%Food';
```

cript Output × Query Result ×

SQL | All Rows Fetched: 2 in 0.318 seconds

	REST_NAME	COMMENTS
1	Green Chilly	Loves Food
2	Manis BBQ	Fresh Food

17) Demonstrate a left join statement.

List all the Customers who received orders from Employees.

```
/*LEFTOUTERJOIN*/

SELECT C.CUST_NAME,E.EMP_NAME,EMP_TYPE FROM (CUSTOMER C LEFT OUTER JOIN EMPLOYEE E ON C.CUST_ID=E.CUST_ID);
```

ript Output × Query Result ×

SQL | All Rows Fetched: 9 in 0.289 seconds

	CUST_NAME	EMP_NAME	EMP_TYPE
1	Vanya	Gary	Part_Time
2	Jackie	Maina	Full_Time
3	Binny	Dany	Full_Time
4	Binny	Josh	Part_Time
5	Rafael	Samy	Full_Time
6	Xavier	Tena	Part_Time
7	Binny	Anil	Part_Time
8	Macon	(null)	(null)
9	Kalvin	(null)	(null)

18) Demonstrate ORDER BY statement to order inner join operation according to foreign key. (Either ascending or descending is acceptable)

Listing all Customers name, Date of birth who have commented

```
SELECT C.CUST_NAME,C.DOB AS BIRTH_DATE,F.COMMENTS FROM CUSTOMER C
INNER JOIN FEEDBACK F ON  F.CUST_ID=C.CUST_ID ORDER BY DOB ASC;
```

cript Output × | Query Result ×

SQL | All Rows Fetched: 7 in 0.383 seconds

	CUST_NAME	BIRTH_DATE	COMMENTS
1	Kalvin	07-DEC-89	Not Tastey
2	Macon	20-JAN-60	wonderful meal
3	Macon	20-JAN-60	Fresh Food
4	Rafael	14-FEB-92	it was delicious.
5	Binny	27-OCT-95	Good Quality
6	Xavier	14-NOV-95	Loves Food
7	Vanya	04-JUL-01	Ambience is good

19) Create a "cascade delete" SQL statement over two tables.

```
/*FEEDBACK*/
CREATE TABLE MYPROJECT.FEEDBACK(
FEEDBACK_ID INT NOT NULL,
COMMENTS VARCHAR2(50),
CUST_ID INT,
REST_ID INT,
REPLY_ID INT,
CONSTRAINT FEEDBACK_ID PRIMARY KEY(FEEDBACK_ID),
CONSTRAINT FEEDBACK_CUST_ID_FK FOREIGN KEY(CUST_ID) REFERENCES MYPROJECT.CUSTOMER(CUST_ID) ON  DELETE CASCADE,
CONSTRAINT FEEDBACK_REST_ID_FK FOREIGN KEY(REST_ID) REFERENCES MYPROJECT.RESTAURANT(REST_ID) ON DELETE CASCADE,
CONSTRAINT FEEDBACK_REP_ID_FK FOREIGN KEY(REPLY_ID) REFERENCES MYPROJECT.FEEDBACK(FEEDBACK_ID) ON DELETE CASCADE
);
```

If the Rest_Id is deleted then the corresponding feedback will be deleted.Similarly, if the Cust_Id is deleted then the corresponding feedback will be deleted.

20) Demonstrate UNION statement.

Get the name list of Employee and Customer

```
SELECT EMP_NAME AS NAME FROM EMPLOYEE
UNION
SELECT CUST_NAME AS NAME FROM CUSTOMER;
```

ript Output × Query Result ×

SQL | All Rows Fetched: 14 in 0.289 seconds

	NAME
1	Anil
2	Binny
3	Dany
4	Gary
5	Jackie
6	Josh
7	Kalvin
8	Macon
9	Maina
10	Rafael
11	Samy
12	Tena
13	Vanya
14	Xavier

21) Demonstrate a SQL statement in which a DATE data type is subject of where statement (such as, select ... from ... where birthday > DATE)

List all the Customers who Booked after 7th January 2020

```
SELECT B.BOOKING_ID,C.CUST_NAME,B.BOOKING_DATE FROM BOOKING B,
CUSTOMER C WHERE (B.CUST_ID=C.CUST_ID AND B.BOOKING_DATE > '07 JAN 2020');
```

Script Output | Query Result

SQL | All Rows Fetched: 7 in 0.347 seconds

	BOOKING_ID	CUST_NAME	BOOKING_DATE
1	50008	Jackie	07-APR-20
2	50001	Kalvin	27-MAR-20
3	50002	Vanya	14-FEB-20
4	50004	Rafael	18-JUL-20
5	50005	Vanya	19-AUG-20
6	50006	Xavier	15-SEP-20
7	50007	Jackie	27-OCT-20

22) Demonstrate CREATE VIEW statement.

```
grant create view to myproject;

CREATE VIEW VIEW_OFFER AS
SELECT O.OFFER_CODE AS COUPON_ID ,B.BOOKING_ID AS BOOKING_NUMBER
FROM MYPROJECT.OFFER O,MYPROJECT.BOOKING B
WHERE O.OFFER_CODE=B.OFFER_CODE ;
```

Script Output | Query Result

Task completed in 1.236 seconds

```
View VIEW_OFFER created.
```

```sql
CREATE VIEW VIEW_OFFER AS
SELECT O.OFFER_CODE AS COUPON_ID ,B.BOOKING_ID AS BOOKING_NUMBER
FROM MYPROJECT.OFFER O,MYPROJECT.BOOKING B
WHERE O.OFFER_CODE=B.OFFER_CODE ;

INSERT INTO MYPROJECT.BOOKING VALUES(50015,600,30001,'HAPPYHOUR','CONFIRMED','28 jul 2020');

SELECT * FROM VIEW_OFFER;
```

Script Output | Query Result

All Rows Fetched: 7 in 0.308 seconds

	COUPON_ID	BOOKING_NUMBER
1	HAPPYHOUR	50015
2	FOODDAY12893	50001
3	JSMSXZ009	50002
4	YUMMYIES0033	50004
5	JACCOKE1234	50006
6	HAPPY10000	50010
7	HAPPYHOUR	50011

23) Delete three rows from a table.

```sql
DELETE  FROM PAYMENT WHERE PAY_TYPE='DEBIT';
```

Query Result | Script Output

Task completed in 0.929 seconds

```
3 rows deleted.
```

24) Delete all rows from a table, then delete the empty table from database.

```
DELETE  FROM PAYMENT WHERE PAY_TYPE='DEBIT';
```

Query Result | Script Output

Task completed in 0.929 seconds

```
3 rows deleted.
```

```
DELETE FROM PAYMENT;
```

Query Result | Script Output

Task completed in 1.243 second

```
4 rows deleted.
```

```
DROP TABLE PAYMENT;
```

Query Result | Script Output

Task completed in 1.493 s

```
Table PAYMENT dropped.
```