*Thapar Institute of Engineering & Technology, Patiala*
*Department of Computer Science and Engineering*

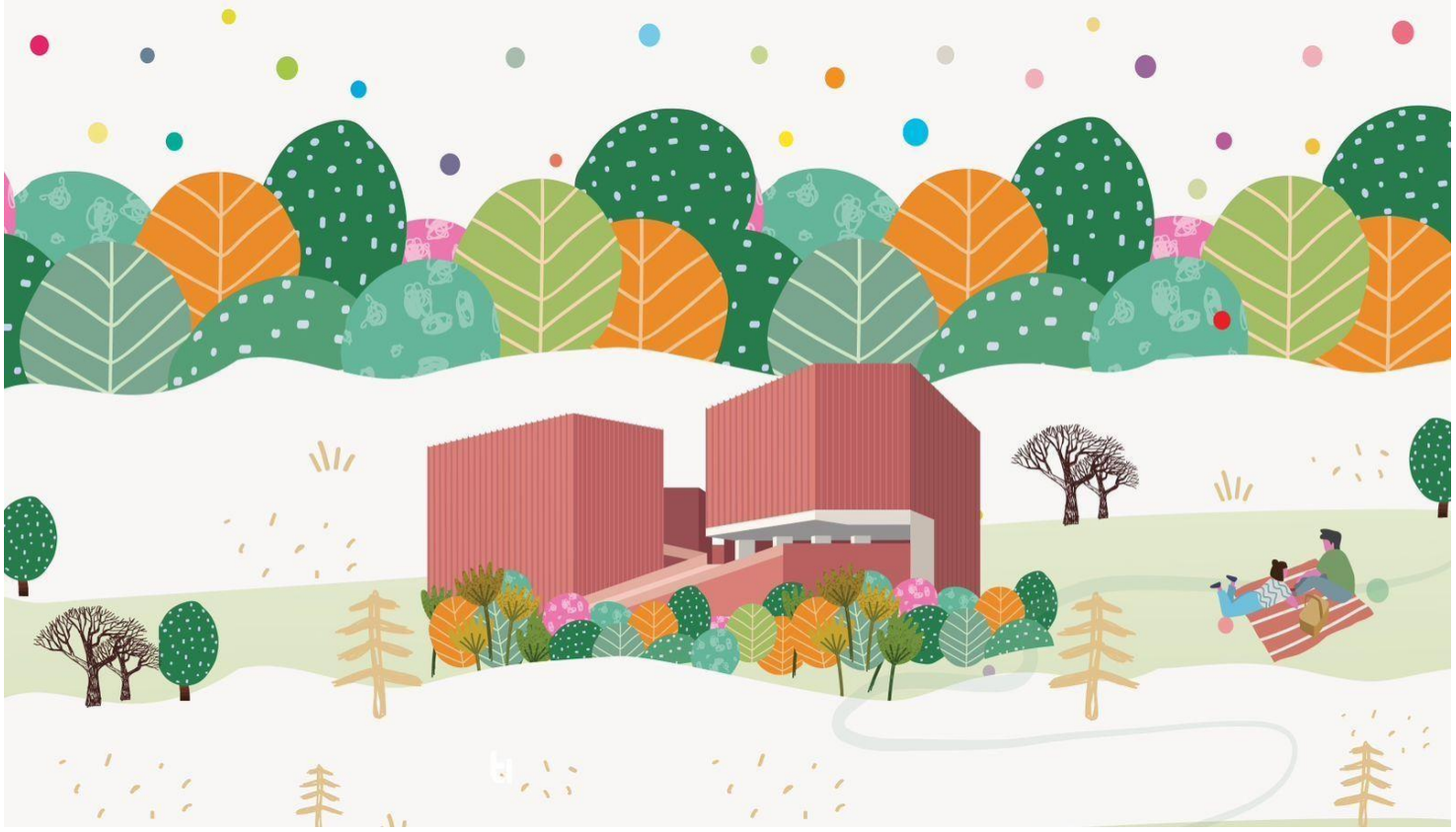
**UCS310:DATABASE MANAGEMENT SYSTEM**

**A PROJECT REPORT on**

# Thapar Treats

## A Smart Food Delivery System for Campus Convenience

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



***By: Smarth Kaushal [102497023] • Diya Kalra [102317174] • Eliza Arora [102317299]***
***B.E.CSE: 2Q2A***

***Submitted to : Ms. Ananya Kaim***
***Dated. May 7th, 2025***

## ABSTRACT

***"At Thapar, every meal is more than just food-it's a promise delivered. With a click, flavors travel across campus, connecting hearts, fueling minds, and turning hunger into happiness, one order at a time."***

In the age of digital transformation, convenience and efficiency have become vital to the quality of daily life—especially within academic campuses. *Thapar Treats* is a student-developed PL/SQL-driven design to streamline the food delivery experience at Thapar Institute of Engineering & Technology (TIET), replacing manual processes with automated workflows.

The system employs **stored procedures** for core functionalities like real-time order tracking (track_order_status), dynamic item availability checks (place_order_if_available), and customer-vendor rating analytics (CheckCustomerDiscountEligibility, CheckSpecificRestaurantRisk). Data integrity is ensured through atomic transactions, such as updating COD permissions based on customer ratings (update_cod_based_on_rating) and validating payment statuses during order placement.

**Role-based access control** and **encrypted credentials** secure interactions across student, vendor, and admin roles. Procedures like GetBestRestaurant aggregate seller ratings for performance insights, while GetDeliveryPartnerInfoByOrderID fetches delivery details using joins between Catalogue and DeliveryPartner tables. Exception handling in PL/SQL blocks mitigates errors like invalid order IDs or out-of-stock items, ensuring robust fault tolerance1.
By centralizing food logistics within an Oracle Database, Thapar Treats demonstrates how PL/SQL's procedural logic enhances operational efficiency, data accuracy, and user experience in academic ecosystems.

The project aims to overcome the inefficiencies associated with traditional mess systems, crowded eateries, and manual order-taking by offering real-time order processing, dynamic menu management, and role-based access control. Built using structured database concepts, *Thapar Treats* ensures data integrity, user authentication, and privacy through encrypted login credentials and secure transaction management. With both delivery and pick-up options, this system not only enhances user experience but also empowers local food vendors by providing them with digital exposure and order management tools. By leveraging PL/SQL's procedural rigor for CRUD operations and stakeholder-specific workflows, the project exemplifies how academic environments can optimize logistical challenges through focused database design.



**Keywords**

Online Food Delivery · Database Management System (DBMS) · Campus Automation · Secure Transactions · Menu Management · Order Tracking · Student Project · Relational Database · User Authentication

# ACKNOWLEDGEMENT

The successful completion of our DBMS project, “Thapar Treats: A PL/SQL-Driven Campus Food Ordering System,” has been a rewarding journey marked by learning, collaboration, and growth. We are deeply grateful to the Lord Almighty for providing us with the strength, clarity, and perseverance to see this project through every stage.

Our motivation stemmed from a desire to address practical challenges faced by students, faculty, and vendors in campus food management. Through extensive discussions, we identified the need for an efficient, technology-driven solution to streamline food ordering, delivery, and vendor management at Thapar Institute. This vision inspired us to design and implement a robust relational database and a suite of PL/SQL procedures that automate and secure the entire process.

Throughout the development, we encountered technical hurdles-from designing normalized tables for customers, sellers, delivery partners, and orders, to writing and debugging complex PL/SQL procedures for real-time order tracking, inventory checks, risk assessment, and user eligibility for discounts. Each challenge became an opportunity to deepen our understanding of database concepts, transactional integrity, and procedural logic. The support and guidance from our mentors and faculty members were invaluable, providing direction and constructive feedback that shaped the quality and scope of our work. We also appreciate the encouragement and insights from our peers, which helped us refine our approach and overcome obstacles as a team.

We owe special thanks to our families for their unwavering support, patience, and motivation throughout this demanding process. Their belief in us fueled our determination during times of uncertainty and long hours of coding and documentation.

This project has not only enhanced our technical skills in PL/SQL and database management but has also taught us important lessons in teamwork, problem-solving, and effective communication. We are sincerely thankful to everyone who contributed-directly or indirectly-to the realization of Thapar Treats. Their support has enabled us to deliver a solution that aspires to make a meaningful impact on campus life, and we hope it serves as a foundation for future innovation in academic environments.

---

Diya Kalra [102317174]
Eliza Arora [102317299]
Smarth Kaushal [102497023]

**TABLE OF CONTENTS**

*"Food is our common ground, a universal experience."*
**— James Beard**

## Problem Statement

With the ever-increasing academic and professional pressure on students and faculty members at institutions like TIET, the challenge of accessing timely, hygienic, and affordable meals has become more pronounced than ever. Traditional food systems-such as on-campus mess services and local eateries-often fall short in meeting expectations for variety, efficiency, and personalization. The absence of digital infrastructure in campus food services leads to persistent issues: long queues, frequent miscommunication in orders, limited transparency, and an overall unsatisfactory user experience. In urban environments, where both men and women are actively engaged in demanding professional and academic pursuits, home-cooked meals are becoming less common, further driving the need for quick, reliable food solutions.

This project directly addresses these gaps by proposing a digitally accessible, secure, and user-friendly food delivery system tailored for the campus ecosystem. Thapar Treats enables users to seamlessly browse diverse menus, place orders, make secure payments, and track deliveries in real-time-all from a single, unified platform. By integrating robust backend support using structured PL/SQL procedures and relational database design, the system ensures data integrity, user authentication, and privacy at every step. Features such as role-based access, automated order management, and personalized recommendations not only streamline operations but also empower local food vendors with digital exposure and efficient order handling.

The goal is to minimize reliance on outdated, manual food management practices and offer a modernized solution that aligns with the lifestyle of today's tech-savvy campus community. Thapar Treats stands as a transformative approach to campus food services, fostering seamless communication between customers and vendors, optimizing operational efficiency, and ultimately enhancing the quality of daily life for all stakeholders within academic institutions.

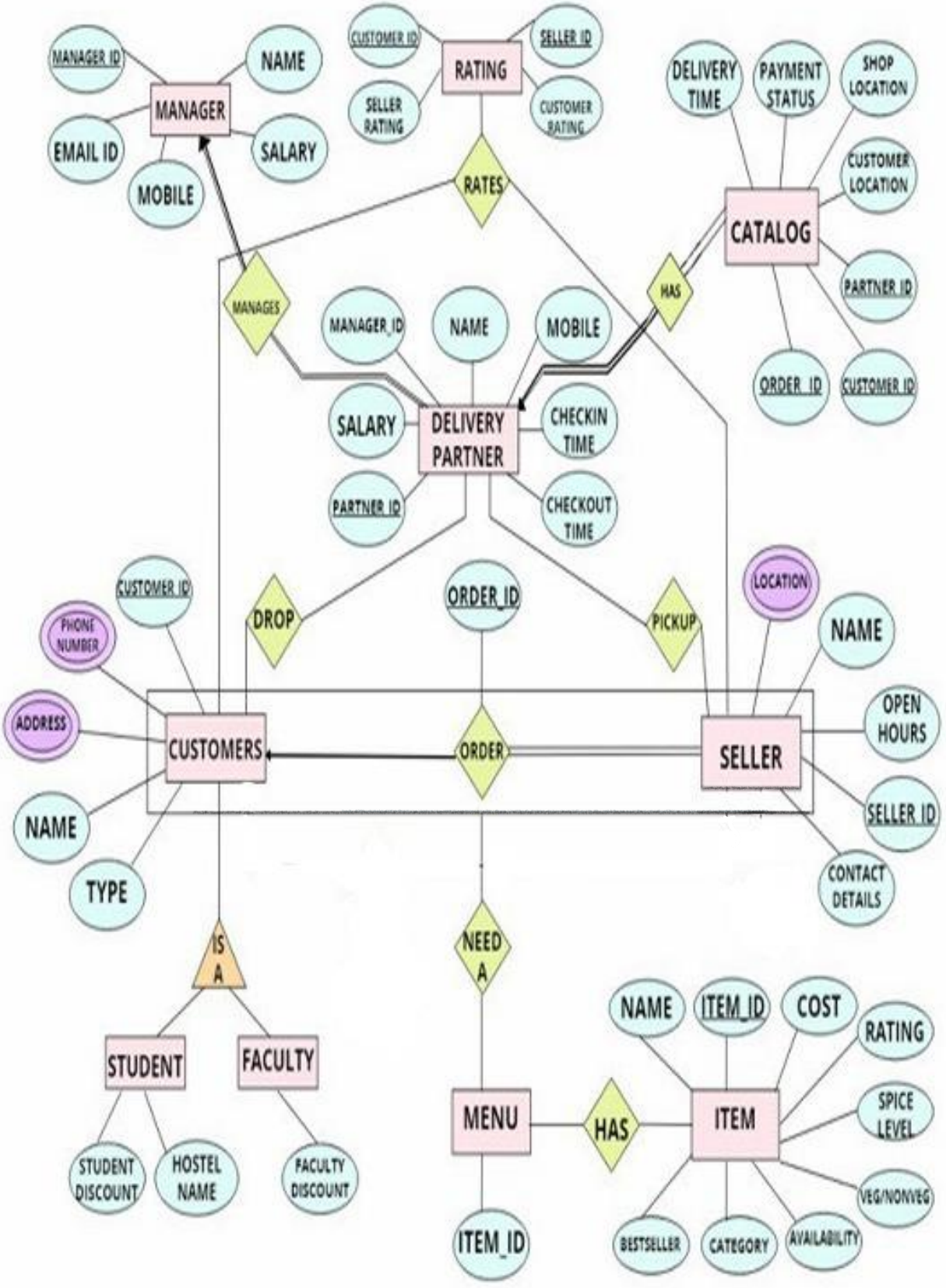
The purpose of this project is to provide an online food delivery system.

Online Food ordering system is a process in which one can order various foods and beverages from some local restaurant and hotels through the use of internet, just by sitting at home or any place. And the order is delivered to the told location. Nowadays everyone is having busy schedule whether it is urban area or rural. But talking specifically about the urban areas and deeply about the big cities, people out there are so busy in their life that they don't get enough of time to have their meals properly. As these days women are no less than men, in any field. So in big cities even wives are working women, therefore mostly the small families manage to have their food ordered from somewhere, as they lack time. Not only this is the case, if we talk about the children in the modern era they like only fast food or something from the outside. But they ignore eating homemade meals.

So food ordering system these days has one of the fastest growing market, though being a new idea. In this project we have developed something like the same to earn from and serve the nation in a much better way possible. Nowadays, people are more regular to dine-in at restaurant for their meals. The online food ordering system provides convenience for the customers that are nothing special but the general busy people of the society. It overcomes the demerits of the manual hotel or mess system and the old fashioned queuing system. This system enhances the readymade of foods than people.

Therefore, this system enhances the speed of getting food in person's plate and quality and manner of taking the order from the customer. It provides a better communication platform. The user's details are stored using the electronic media. The online food ordering system provides the menu online and the customers can easily place the order by just clicking the mouse or by touching a button on their smart phones. Also with the food ordering system online, people can easily track their orders, and admin can maintain customer's database and advance the food delivery system.

This food ordering system allows the user to select the desired food items from a list of available menu items provided by the local hotel or restaurant. The user can place orders for the food items of their like from the list. The payment can be made online or pay-on-delivery system. The user's details are maintained confidential because it maintains a separate account for each user. An id and password is provided for each user. And several encryption techniques have also been used on the server side to protect the card details. Therefore it provides a more secured and safe ordering system.


MANAGER ID
NAME
MANAGER
EMAIL ID
SALARY
MOBILE
CUSTOMER ID
SELLER ID
RATING
SELLER RATING
CUSTOMER RATING
RATES
DELIVERY TIME
PAYMENT STATUS
SHOP LOCATION
CUSTOMER LOCATION
CATALOG
PARTNER ID
ORDER_ID
CUSTOMER ID
HAS
MANAGES
MANAGER_ID
NAME
MOBILE
SALARY
DELIVERY PARTNER
CHECKIN TIME
PARTNER ID
CHECKOUT TIME
CUSTOMER ID
PHONE NUMBER
ADDRESS
CUSTOMERS
NAME
TYPE
DROP
ORDER_ID
PICKUP
LOCATION
NAME
OPEN HOURS
SELLER
SELLER ID
CONTACT DETAILS
ORDER
IS A
STUDENT
FACULTY
STUDENT DISCOUNT
HOSTEL NAME
FACULTY DISCOUNT
NEED A
MENU
ITEM_ID
HAS
NAME
ITEM_ID
COST
RATING
ITEM
SPICE LEVEL
VEG/NONVEG
BESTSELLER
CATEGORY
AVAILABILITY

## ER-MODEL TO RELATIONAL MODEL

*Customer*

- customer_id (PK)
- type
- firstname
- middlename
- lastname

*CustPhone*

- customer_id (PK,FK)
- phone_no (PK)

*CustAddress*

- customer_id (PK,FK)
- address (PK)

*Manager*

- manager_id (PK)
- name
- salary
- mobile
- emailid

*DeliveryPartner*

- partner_id (PK)
- manager_id (FK)
- name
- salary
- mobile
- working_hours

*Seller*

- seller_id (PK)
- name
- opening_time
- closing_time
- mobile
- location

*Orders*

- order_id (PK)
- customer_id (FK)
- seller_id (FK)
- item_id (FK)

*Feedback*

- customer_id (FK)
- seller_id (FK)
- reviews

*Item*

- item_id (PK)
- name
- cost
- availability
- veg_nonveg
- bestseller
- category
- spice_level
- rating

*Catalogue*

- order_id (FK)
- customer_id (FK)
- partner_id (FK)
- customer_location
- shop_location
- delivery_time
- payment_status

*Rating*

- customer_id (FK)
- seller_id (FK)
- customer_rating
- seller_rating

1. *Customer:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. type, firstname, middlename, and lastname are all fully dependent on the primary key customer_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (customer_id) uniquely identifies all attributes.

2. *CustPhone:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. phone_no is fully dependent on the foreign key customer_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (customer_id, phone_no) uniquely identifies all attributes.

3. *CustAddress:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. Address is fully dependent on the foreign key customer_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (customer_id, address) uniquely identifies all attributes.

4. *Manager:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the primary key manager_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (manager_id) uniquely identifies all attributes.

5. *DeliveryPartner:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the primary key partner_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (partner_id) uniquely identifies all attributes.

6. *Seller:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the primary key seller_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (seller_id) uniquely identifies all attributes.

7. *Orders:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the primary key order_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (order_id) uniquely identifies all attributes.

8. *Feedback:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** No. reviews is not dependent on the primary key (seller_id, customer_id).
- **3NF:** Not applicable due to violation of 2NF.
- **BCNF:** Not applicable due to violation of 2NF.

9. *Item:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the primary key item_id.
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The primary key (item_id) uniquely identifies all attributes.

10. *Catalogue:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** Yes. All attributes are fully dependent on the composite primary key (order_id, customer_id, partner_id).
- **3NF:** Yes. There are no transitive dependencies.
- **BCNF:** Yes. The composite primary key uniquely identifies all attributes.

11. *Rating:*

- **1NF:** Yes. No repeating groups or multivalued attributes.
- **2NF:** No. customer_rating and seller_rating are not dependent on the primary key (seller_id, customer_id).
- **3NF:** Not applicable due to violation of 2NF.
- **BCNF:** Not applicable due to violation of 2NF.

*Notes:*

- Feedback and Rating tables violate 2NF because some attributes are not fully dependent on the primary key. You might consider separating reviews into another table or redesigning the primary key.

# SQL & PL/SQL Queries & Snapshots with output

## I. RELATION SHEMAS: TABLE CREATION STATEMENTS

```
create table Customer(
    customer_id number primary key,
    type varchar2(10),
    firstname varchar2(10),
    middlename varchar2(10),
    lastname varchar2(10)
)

create table CustPhone(
    customer_id number,
    phone_no number,
    foreign key (customer_id) references Customer (customer_id),
    primary key (customer_id,phone_no)
)

create table CustAddress(
    customer_id number,
    address varchar2(30),
    foreign key (customer_id) references Customer (customer_id),
    primary key (customer_id,address)
)

create table Manager(
    manager_id number primary key,
    name varchar2(20),
    salary number,
    mobile number,
    emailid varchar(30)
)

create table DeliveryPartner(
    partner_id varchar2(15) primary key,
    manager_id number,
    name varchar2(20),
    salary number,
    mobile number,
    checkin_time timestamp,
    checkout_time timestamp,
    foreign key (manager_id) references Manager (manager_id)
)
```

```
create table Seller(
    seller_id number primary key,
    name varchar2(20),
    opening_time timestamp,
    closing_time timestamp,
    mobile number,
    location varchar2(15),
    check (closing_time>opening_time)
)

create table Orders(
    order_id number primary key,
    customer_id number,
    seller_id number,
     item_id varchar2(15),
    foreign key (customer_id) references Customer (customer_id),
    foreign key (seller_id) references Seller (seller_id),
     foreign key (item_id) references Item (item_id)
)

create table Feedback(
    customer_id number,
    seller_id number,
    reviews varchar2(100),
    foreign key (seller_id) references Seller (seller_id),
    foreign key (customer_id) references Customer (customer_id),
    primary key (seller_id,customer_id)
)

create table Item(
    item_id varchar2(15) primary key,
    name varchar2(20),
    cost number,
    availability number(1),
    veg_nonveg number(1),
    bestseller number(1),
    category varchar2(20),
    spice_level varchar2(10),
    rating number,
    constraint check_rating check (rating>=0 and rating<=5)
)
```

```
create table Catalogue(
    order_id number,
    customer_id number,
    partner_id varchar2(15),
    customer_location varchar2(20),
    shop_location varchar2(20),
    delivery_time timestamp,
    payment_status number(1),
    primary key (order_id, customer_id, partner_id),
    foreign key (order_id) references Orders (order_id),
    foreign key (partner_id) references DeliveryPartner (partner_id),
    foreign key (customer_id) references Customer (customer_id)
)

create table Rating(
    customer_id number,
    seller_id number,
    customer_rating number default (2.5),
    seller_rating number default (2.5),
    foreign key (seller_id) references Seller (seller_id),
    foreign key (customer_id) references Customer (customer_id),
    primary key (seller_id,customer_id),
    constraint check_cust_rating check(customer_rating>=0 and customer_rating<=5),
    constraint check_sell_rating check(seller_rating>=0 and seller_rating<=5)
)
```

## II. INSERT QUERIES ON TABLES

```
insert into Customer values (1,'student','rehan','kumar','bansal'),
(3,'student','avni','kay','gupta'),
(2,'faculty','ankush','aman','pathania'),
(4,'faculty','prashant','singh','rana'),
(5,'student','prarthana','jenny','samal'),
(6,'faculty','harpreet','singh','bawa'),
(7,'student','atansh','ajay','dhiman'),
(8,'faculty','tarunpreet','kaur','bhatia'),
(9,'student','aashi','zoya','yadav'),
(10,'faculty','shubhra','claire','dwivedi');
select * from Customer;
```

| CUSTOMER_ID | TYPE | FIRSTNAME | MIDDLENAME | LASTNAME |
|---|---|---|---|---|
| 1 | student | rehan | kumar | bansal |
| 2 | faculty | ankush | aman | pathania |
| 3 | student | avni | kay | gupta |
| 4 | faculty | prashant | singh | rana |
| 5 | student | prarthana | jenny | samal |
| 6 | faculty | harpreet | singh | bawa |
| 7 | student | atansh | ajay | dhiman |

```
insert into CustPhone values (1,4985672156),
 (1,6598745123),
 (2,4875963216),
 (2,9897562355),
 (4,5698985612),
 (5,7864554353),
 (3,3264994659),
 (4,9652313498),
 (7,7985632659),
 (6,8745315878);
select * from CustPhone;
```

| CUSTOMER_ID | PHONE_NO |
|---|---|
| 1 | 4985672156 |
| 1 | 6598745123 |
| 2 | 4875963216 |
| 2 | 9897562355 |
| 3 | 3264994659 |
| 4 | 5698985612 |
| 4 | 9652313498 |

```
insert into CustAddress values (1,'hostel-c'),
 (1,'thapar'),
 (2,'model town'),
 (2,'patiala'),
 (4,'adarsh nagar'),
 (5,'hostel-n'),
 (3,'hostel-q'),
 (4,'patiala'),
 (7,'hostel-o'),
 (6,'puda colony patiala');
select * from CustAddress;
```

| CUSTOMER_ID | ADDRESS |
|---|---|
| 1 | hostel-c |
| 1 | thapar |
| 2 | model town |
| 2 | patiala |
| 3 | hostel-q |
| 4 | adarsh nagar |
| 4 | patiala |

```
insert into Manager values(100,'raman',50000,1288912998,'rmn@gmail.com');
insert into Manager values(200,'manan',35600,6708567900,'mn@gmail.com');
insert into Manager values(300,'daman',67800,7859376748,'dmn@gmail.com');
insert into Manager values(400,'chaman',98230,3248967898,'chmn@gmail.com');
insert into Manager values(500,'aman',40000,9824382970,'amn@gmail.com');
insert into Manager values(600,'kim jon',30000,2324384570,'kj@gmail.com');
insert into Manager values(700,'julius',70000,4324384450,'jl@gmail.com');
insert into Manager values(800,'dinesh',82300,7552438200,'dn@gmail.com');
select * from Manager;
```

| MANAGER_ID | NAME | SALARY | MOBILE | EMAILID |
|---|---|---|---|---|
| 100 | raman | 50000 | 1288912998 | rmn@gmail.com |
| 200 | manan | 35600 | 6708567900 | mn@gmail.com |
| 300 | daman | 67800 | 7859376748 | dmn@gmail.com |
| 400 | chaman | 98230 | 3248967898 | chmn@gmail.com |
| 500 | aman | 40000 | 9824382970 | amn@gmail.com |
| 600 | kim jon | 30000 | 2324384570 | kj@gmail.com |
| 700 | julius | 70000 | 4324384450 | jl@gmail.com |

```
insert into Seller (seller_id,name,mobile,location) values
(10,'fashion point',349843897,'cos');
insert into Seller (seller_id,name,mobile,location) values
(15,'honey cafe',89735983,'cos');
insert into Seller (seller_id,name,mobile,location) values
(20,'sip n bites',09378529307,'cos');
insert into Seller (seller_id,name,mobile,location) values
(25,'pizza nation',43678423,'cos');
insert into Seller (seller_id,name,mobile,location) values
(30,'the dessert club',234632498,'cos');
insert into Seller (seller_id,name,mobile,location) values
(35,'jp foods',230947329,'G block');
insert into Seller (seller_id,name,mobile,location) values
(40,'patiala shahi',9865897597,'G block');
insert into Seller (seller_id,name,mobile,location) values
(45,'kulcha zone',152345612,'H block');
select * from Seller;
update  Seller set name = 'Wrapchik' where seller_id = 15;
```

Download Execution time: 0.067 seconds

| | SELLER_ID | NAME | OPENING_TIME | CLOSING_TIME | MOBILE | LOCATION |
|---|---|---|---|---|---|---|
| 1 | 10 | fashion point | 2025-05-01T09:30:( | 2025-05-01T20:00:( | 349843897 | cos |
| 2 | 15 | Wrapchik | 2025-05-01T10:00:( | 2025-05-01T21:30:( | 89735983 | cos |
| 3 | 20 | sip n bites | 2025-05-01T10:30:( | 2025-05-01T21:30:( | 9378529307 | cos |
| 4 | 25 | pizza nation | 2025-05-01T11:00:( | 2025-05-01T23:59:( | 43678423 | cos |
| 5 | 30 | the dessert club | 2025-05-01T13:00:( | 2025-05-01T22:00:( | 234632498 | cos |
| 6 | 35 | jp foods | 2025-05-01T09:00:( | 2025-05-01T20:30:( | 230947329 | G block |
| 7 | 40 | patiala shahi | 2025-05-01T11:00:( | 2025-05-01T23:00:( | 9865897597 | G block |
| 8 | 45 | kulcha zone | 2025-05-01T13:30:( | 2025-05-01T16:00:( | 152345612 | H block |

```
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('fp1',100,'ajay',4000,9238643289);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('fp2',100,'ramu',7400,3847783444);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('hc1',200,'kundan',7700,432478893);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('hc2',200,'sukhbir',3300,8997849454);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('snb1',300,'laadi',3000,4523403285);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('snb2',300,'munna',4010,6584893324);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('pn1',400,'arsh',5500,544521555);
```

```
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('pn2',400,'lokesh',5000,7853889437);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('dc1',500,'rahul',8000,763298479843);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('dc2',500,'mike',4500,5638793298);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('jp1',600,'john',5000,7538984987);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('jp2',600,'keval',4000,989563235);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('ps1',700,'tejas',6700,664584635);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('ps2',700,'shobhit',5000,33242398954);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('kz1',800,'mayank',3200,673898645);
insert into DeliveryPartner (partner_id,manager_id,name,salary,mobile)
values('kz2',800,'bhuvi',4500,93847903);
select * from DeliveryPartner;
```

| PARTNER_ID | MANAGER_ID | NAME | SALARY | MOBILE | WORKING_HOURS |
|---|---|---|---|---|---|
| fp1 | 100 | ajay | 4000 | 9238643289 | - |
| fp2 | 100 | ramu | 7400 | 3847783444 | - |
| hc1 | 200 | kundan | 7700 | 432478893 | - |
| hc2 | 200 | sukhbir | 3300 | 8997849454 | - |
| snb1 | 300 | laadi | 3000 | 4523403285 | - |
| snb2 | 300 | munna | 4010 | 6584893324 | - |
| pn1 | 400 | arsh | 5500 | 544521555 | - |
| pn2 | 400 | lokesh | 5000 | 7853889437 | - |

```
insert into Item values ('hc1','spring roll',70,1,1,0,'snacks','medium',3);
insert into Item values ('hc2','noodles',45,0,0,0,'snacks','low',2.5);
insert into Item values ('hc3','manchurian',50,1,1,1,'snacks','high',4);
insert into Item values ('snb1','spicy paneer',70,0,1,0,'wrap','medium',2);
insert into Item values ('snb2','grilled chicken',70,1,0,1,'wrap','high',4.3);
insert into Item values ('snb3','big sumo',60,1,1,1,'burger','low',5);
insert into Item values ('snb4','fried chicken',80,1,0,0,'burger','medium',2.3);
insert into Item values ('dc1','oreo king',200,1,1,0,'bubble waffle','low',3.7);
insert into Item values ('dc2','ferrero king',200,1,1,1,'bubble waffle','low',5);
insert into Item values ('dc3','brownie queen',120,1,1,0,'square waffle','low',1.5);
insert into Item values ('kz1','garlic',140,1,0,1,'chaap','high',4.6);
insert into Item values ('kz2','paneer',60,0,1,0,'kulcha','medium',3);
select * from Item;
```

| ITEM_ID | NAME | COST | AVAILABILITY | VEG_NONVEG | BESTSELLER | CATEGORY | SPICE_LEVEL | RATING |
|---|---|---|---|---|---|---|---|---|
| snb1 | spicy paneer | 70 | 0 | 0 | 0 | wrap | medium | 2 |
| dc1 | oreo king | 200 | 1 | 1 | 0 | bubble waffle | low | 3.7 |
| hc1 | spring roll | 70 | 1 | 1 | 0 | snacks | medium | 3 |
| hc3 | manchurian | 50 | 1 | 1 | 1 | snacks | high | 4 |
| snb2 | grilled chicken | 70 | 1 | 0 | 1 | wrap | high | 4.3 |
| snb3 | big sumo | 60 | 1 | 1 | 1 | burger | low | 5 |
| snb4 | fried chicken | 80 | 1 | 0 | 0 | burger | medium | 2.3 |

```
update Seller set opening_time=TO_DATE('09:30:00',
'HH24:MI:SS'),closing_time=TO_DATE('20:00:00', 'HH24:MI:SS') where seller_id=10;
update Seller set opening_time=TO_DATE('13:30:00',
'HH24:MI:SS'),closing_time=TO_DATE('16:00:00', 'HH24:MI:SS') where seller_id=45;
update Seller set opening_time=TO_DATE('10:00:00',
'HH24:MI:SS'),closing_time=TO_DATE('21:30:00', 'HH24:MI:SS') where seller_id=15;
update Seller set opening_time=TO_DATE('10:30:00',
'HH24:MI:SS'),closing_time=TO_DATE('21:30:00', 'HH24:MI:SS') where seller_id=20;
update Seller set opening_time=TO_DATE('11:00:00',
'HH24:MI:SS'),closing_time=TO_DATE('23:59:00', 'HH24:MI:SS') where seller_id=25;
update Seller set opening_time=TO_DATE('13:00:00',
'HH24:MI:SS'),closing_time=TO_DATE('22:00:00', 'HH24:MI:SS') where seller_id=30;
update Seller set opening_time=TO_DATE('09:00:00',
'HH24:MI:SS'),closing_time=TO_DATE('20:30:00', 'HH24:MI:SS') where seller_id=35;
update Seller set opening_time=TO_DATE('11:00:00',
'HH24:MI:SS'),closing_time=TO_DATE('23:00:00', 'HH24:MI:SS') where seller_id=40;
```

```
insert into Feedback values (1,10,'quality was good. a bit spicy. lower the sugar content next time');
insert into Feedback values (1,20,'quality was poor. highly spicy. food was uncooked');
insert into Feedback values (2,10,'quality was not good. food was smelly. lower the sugar content next time');
insert into Feedback values (3,15,'ice cream flavour was very unique. good presentation');
insert into Feedback values (3,20,'overall nice experience. satisfied. tasty food');
insert into Feedback values (6,30,'delivery was late. delivery partner was rude');
insert into Feedback values (7,40,'insects found in food');
insert into Feedback values (4,35,'got wrong order');
insert into Feedback values (6,40,'quality was good. a bit spicy.');
select * from Feedback;
```

| CUSTOMER_ID | SELLER_ID | REVIEWS |
|---|---|---|
| 1 | 10 | quality was good. a bit spicy. lower the sugar content next time |
| 1 | 20 | quality was poor. highly spicy. food was uncooked |
| 2 | 10 | quality was not good. food was smelly. lower the sugar content next time |
| 3 | 15 | ice cream flavour was very unique. good presentation |
| 3 | 20 | overall nice experience. satisfied. tasty food |
| 6 | 30 | delivery was late. delivery partner was rude |
| 7 | 40 | insects found in food |
| 4 | 35 | got wrong order |

```
insert into Rating values (1,10,4.2,4.5);
insert into Rating values (2,10,3.2,4);
insert into Rating values (2,35,5,2);
insert into Rating values (5,40,1.2,4.8);
insert into Rating values (5,45,2.2,3.3);
insert into Rating values (9,20,1.7,4.5);
insert into Rating values (8,20,5,5);
insert into Rating values (8,35,3.9,2);
insert into Rating values (9,40,4.9,3.9);
insert into Rating values (10,15,4,5);
select * from Rating;
```

| CUSTOMER_ID | SELLER_ID | CUSTOMER_RATING | SELLER_RATING |
|---|---|---|---|
| 1 | 10 | 4.2 | 4.5 |
| 2 | 10 | 3.2 | 4 |
| 2 | 35 | 5 | 2 |
| 5 | 40 | 1.2 | 4.8 |
| 5 | 45 | 2.2 | 3.3 |
| 9 | 20 | 1.7 | 4.5 |
| 8 | 20 | 5 | 5 |

```
insert into Orders values (000,5,15,'hc1');
insert into Orders values (001,4,15,'hc2');
insert into Orders values (010,3,20,'snb2');
insert into Orders values (011,5,20,'snb2');
insert into Orders values (100,7,30,'dc1');
insert into Orders values (101,7,30,'dc3');
insert into Orders values (110,2,45,'kz2');
insert into Orders values (111,3,20,'snb4');
insert into Orders values (1000,1,15,'hc3');
insert into Orders values (1001,6,45,'kz2');
insert into Orders values (1010,6,45,'kz1');
select * from Orders;
```

| ORDER_ID | CUSTOMER_ID | SELLER_ID | ITEM_ID |
|---|---|---|---|
| 0 | 5 | 15 | hc1 |
| 1 | 4 | 15 | hc2 |
| 10 | 3 | 20 | snb2 |
| 11 | 5 | 20 | snb2 |
| 100 | 8 | 30 | dc1 |
| 101 | 8 | 30 | dc3 |
| 110 | 2 | 45 | kz2 |

```
insert into Catalogue values (000,5,'hc1','hostel-
n','cos',to_date('00:15:00','HH24:MI:SS'),1);
insert into Catalogue values (001,4,'hc2','adarsh
nagar','cos',to_date('00:10:00','HH24:MI:SS'),0);
insert into Catalogue values (010,3,'snb1','hostel-
q','cos',to_date('00:20:00','HH24:MI:SS'),0);
insert into Catalogue values (011,5,'snb1','hostel-
n','cos',to_date('00:25:00','HH24:MI:SS'),1);
insert into Catalogue values (100,7,'dc1','hostel-
o','cos',to_date('00:15:00','HH24:MI:SS'),0);
insert into Catalogue values (101,7,'dc2','hostel-
o','cos',to_date('00:10:00','HH24:MI:SS'),1);
insert into Catalogue values (110,2,'kz2','patiala','H
block',to_date('00:30:00','HH24:MI:SS'),1);
insert into Catalogue values (111,3,'snb2','hostel-
q','cos',to_date('00:05:00','HH24:MI:SS'),0);
insert into Catalogue values (1000,1,'hc1','hostel-c
thapar','cos',to_date('00:10:00','HH24:MI:SS'),1);
insert into Catalogue values (1001,6,'kz1','puda colony patiala','H
block',to_date('00:20:00','HH24:MI:SS'),1);
insert into Catalogue values (1010,6,'kz1','puda colony patiala','H
block',to_date('00:35:00','HH24:MI:SS'),1);
select * from Catalogue;
```

| | ORDER_ID | CUSTOMER_ID | PARTNER_ID | CUSTOMER_LOCATI | SHOP_LOCATION | DELIVERY_TIME | PAYMENT_STATUS |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 5 | hc1 | hostel-n | cos | 2025-05-01T00:15:( | 1 |
| 2 | 1 | 4 | hc2 | adarsh nagar | cos | 2025-05-01T00:10:( | 0 |
| 3 | 10 | 3 | snb1 | hostel-q | cos | 2025-05-01T00:20:( | 0 |
| 4 | 11 | 5 | snb1 | hostel-n | cos | 2025-05-01T00:25:( | 1 |
| 5 | 100 | 7 | dc1 | hostel-o | cos | 2025-05-01T00:15:( | 0 |

## III. PROCEDURES CREATED

### 1. track_order_status

```
-- track order status

CREATE OR REPLACE PROCEDURE track_order_status (
    p_order_id IN NUMBER
) AS
    v_customer_id  NUMBER;
    v_partner_id    VARCHAR2(15);
    v_customer_loc  VARCHAR2(20);
    v_shop_loc      VARCHAR2(20);
    v_delivery_time TIMESTAMP;
    v_payment_stat  NUMBER(1);
BEGIN

    SELECT customer_id, partner_id, customer_location, shop_location, delivery_time, payment_status
    INTO v_customer_id, v_partner_id, v_customer_loc, v_shop_loc, v_delivery_time, v_payment_stat
    FROM Catalogue
    WHERE order_id = p_order_id;


    DBMS_OUTPUT.PUT_LINE('Order ID: ' || p_order_id);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer_id);
    DBMS_OUTPUT.PUT_LINE('Partner ID: ' || v_partner_id);
    DBMS_OUTPUT.PUT_LINE('Customer Location: ' || v_customer_loc);
    DBMS_OUTPUT.PUT_LINE('Shop Location: ' || v_shop_loc);
    DBMS_OUTPUT.PUT_LINE('Delivery Time: ' || TO_CHAR(v_delivery_time, 'YYYY-MM-DD
HH24:MI:SS'));
    DBMS_OUTPUT.PUT_LINE('Payment Status: ' || v_payment_stat);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No matching order found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/

EXEC track_order_status(0);
```

```
SQL> EXEC track_order_status(0)


Order ID: 0
Customer ID: 5
Partner ID: hc1
Customer Location: hostel-n
Shop Location: cos
Delivery Time: 2025-05-01 00:15:00
Payment Status: 1


PL/SQL procedure successfully completed.
```

## 2. get_item_details

```
-- Item details
CREATE OR REPLACE PROCEDURE get_item_details (
    p_item_id IN VARCHAR2
) AS
    v_name         VARCHAR2(20);
    v_cost         NUMBER;
    v_avail        NUMBER(1);
    v_veg_nonveg   NUMBER(1);
    v_bestseller   NUMBER(1);
    v_category     VARCHAR2(20);
    v_spice_level  VARCHAR2(10);
    v_rating       NUMBER;
BEGIN

    SELECT name, cost, availability, veg_nonveg, bestseller,
           category, spice_level, rating
    INTO v_name, v_cost, v_avail, v_veg_nonveg, v_bestseller,
         v_category, v_spice_level, v_rating
    FROM Item
    WHERE item_id = p_item_id;

    DBMS_OUTPUT.PUT_LINE('Item ID: ' || p_item_id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_name);
    DBMS_OUTPUT.PUT_LINE('Cost: ' || v_cost);
    DBMS_OUTPUT.PUT_LINE('Availability: ' || CASE WHEN v_avail = 1 THEN 'true' ELSE 'false'
END);
    DBMS_OUTPUT.PUT_LINE('Veg/Non-Veg: ' || CASE WHEN v_veg_nonveg = 1 THEN 'Veg'
ELSE 'Non-Veg' END);
    DBMS_OUTPUT.PUT_LINE('Bestseller: ' || CASE WHEN v_bestseller = 1 THEN 'true' ELSE 'false'
END);
    DBMS_OUTPUT.PUT_LINE('Category: ' || v_category);
    DBMS_OUTPUT.PUT_LINE('Spice Level: ' || v_spice_level);
    DBMS_OUTPUT.PUT_LINE('Rating: ' || v_rating);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No item found for ID: ' || p_item_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
EXEC get_item_details('hc1');
```

```
Procedure created.
Item ID: hc1
Name: spring roll
Cost: 70
Availability: true
Veg/Non-Veg: Veg
Bestseller: false
Category: snacks
Spice Level: medium
Rating: 3


PL/SQL procedure successfully completed.
```

### 3. place_order_if_available

```
-- place order
CREATE OR REPLACE PROCEDURE place_order_if_available (
    p_customer_id IN NUMBER,
    p_item_id IN VARCHAR2
) AS
    v_availability NUMBER;
    v_new_order_id NUMBER;
    v_seller_id NUMBER;
BEGIN
    -- Check availability
    SELECT availability INTO v_availability
    FROM Item
    WHERE item_id = p_item_id;

    IF v_availability = 1 THEN
        -- Get seller_id from Orders (if fixed relationship) or choose from Item/any other logic
        SELECT seller_id INTO v_seller_id
        FROM Orders
        WHERE item_id = p_item_id
        AND ROWNUM = 1;

        -- Generate new order_id (should use a sequence ideally)
        SELECT NVL(MAX(order_id), 0) + 1 INTO v_new_order_id FROM Orders;

        -- Insert order
        INSERT INTO Orders(order_id, customer_id, seller_id, item_id)
        VALUES (v_new_order_id, p_customer_id, v_seller_id, p_item_id);

        DBMS_OUTPUT.PUT_LINE('Order placed successfully. Order ID: ' || v_new_order_id);
        DBMS_OUTPUT.PUT_LINE('Order Status: Order placed successfully. Order ID: ' ||
v_new_order_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Item with ID ' || p_item_id || ' is not available.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Item ID not found or seller not linked.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/
BEGIN
    place_order_if_available(5, 'hc1');
END;
/
Select * from orders;
```

```
Order placed successfully. Order ID: 1012
Order Status: Order placed successfully. Order ID: 1012
```

```
-- COD
select * from Customer;
ALTER TABLE Customer ADD cod_allowed NUMBER(1) DEFAULT 1;
```

## 4. update_cod_based_on_rating

```
CREATE OR REPLACE PROCEDURE update_cod_based_on_rating (
    p_customer_id IN NUMBER
) AS
    v_rating NUMBER;
BEGIN
    -- Calculate average rating for the customer
    SELECT AVG(customer_rating)
    INTO v_rating
    FROM Rating
    WHERE customer_id = p_customer_id;

    IF v_rating < 2 THEN
        UPDATE Customer
        SET cod_allowed = 0
        WHERE customer_id = p_customer_id;
        DBMS_OUTPUT.PUT_LINE('Customer ID ' || p_customer_id || ' has low rating (' || v_rating || ').
COD disabled.');
    ELSE
        UPDATE Customer
        SET cod_allowed = 1
        WHERE customer_id = p_customer_id;
        DBMS_OUTPUT.PUT_LINE('Customer ID ' || p_customer_id || ' has sufficient rating (' || v_rating
|| '). COD enabled.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No rating found for customer ID ' || p_customer_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
BEGIN
    update_cod_based_on_rating(5);
END;
/
```

```
Customer ID 5 has low rating (1.7). COD disabled.
```

## 5. track_order_status

```
-- CheckSpecificRestaurantRisk

CREATE OR REPLACE PROCEDURE CheckSpecificRestaurantRisk(
    p_seller_id IN NUMBER
) IS
    v_avg_rating NUMBER := 0;
    v_negative_feedback_count NUMBER := 0;
BEGIN
    -- Get average rating (use NVL to avoid null)
    SELECT NVL(AVG(seller_rating), 0)
    INTO v_avg_rating
    FROM Rating
    WHERE seller_id = p_seller_id;

    -- Count negative feedbacks
    SELECT COUNT(*)
    INTO v_negative_feedback_count
    FROM Feedback
    WHERE seller_id = p_seller_id
      AND (
            LOWER(reviews) LIKE '%poor%' OR
            LOWER(reviews) LIKE '%smelly%' OR
            LOWER(reviews) LIKE '%wrong%' OR
            LOWER(reviews) LIKE '%insects%'
          );

    -- Debug: print the values
    DBMS_OUTPUT.PUT_LINE('Average Rating: ' || v_avg_rating);
    DBMS_OUTPUT.PUT_LINE('Negative Feedback Count: ' || v_negative_feedback_count);

    -- Risk evaluation
    IF v_avg_rating < 2.5 OR v_negative_feedback_count > 2 THEN
        DBMS_OUTPUT.PUT_LINE('Restaurant ' || p_seller_id || ' is At Risk of Removal');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Restaurant ' || p_seller_id || ' is Not At Risk');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for the specified restaurant ID: ' || p_seller_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/

EXEC CheckSpecificRestaurantRisk(35);
```

```
Average Rating: 2
Negative Feedback Count: 1
Restaurant 35 is At Risk of Removal
```

## 6. GetDeliveryPartnerInfoByOrderID

```
-- Delivery Partner Info
CREATE OR REPLACE PROCEDURE GetDeliveryPartnerInfoByOrderID(
    p_order_id IN NUMBER  -- Input parameter for order_id
) IS
    v_partner_id VARCHAR2(15);
    v_partner_name VARCHAR2(20);
    v_partner_mobile NUMBER;
    v_partner_salary NUMBER;
    v_partner_manager_id NUMBER;
BEGIN
    -- Get the delivery partner information for the given order_id
    SELECT
        dp.partner_id,
        dp.name AS partner_name,
        dp.mobile AS partner_mobile,
        dp.salary AS partner_salary,
        dp.manager_id
    INTO
        v_partner_id, v_partner_name, v_partner_mobile, v_partner_salary, v_partner_manager_id
    FROM
        Catalogue c
    JOIN
        DeliveryPartner dp ON c.partner_id = dp.partner_id
    WHERE
        c.order_id = p_order_id;

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Delivery Partner ID: ' || v_partner_id);
    DBMS_OUTPUT.PUT_LINE('Partner Name: ' || v_partner_name);
    DBMS_OUTPUT.PUT_LINE('Partner Mobile: ' || v_partner_mobile);
    DBMS_OUTPUT.PUT_LINE('Partner Salary: ' || v_partner_salary);
    DBMS_OUTPUT.PUT_LINE('Manager ID: ' || v_partner_manager_id);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No delivery partner found for the specified order ID: ' ||
p_order_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred while fetching delivery partner info.');
END GetDeliveryPartnerInfoByOrderID;
/
Exec GetDeliveryPartnerInfoByOrderID(0);
```

```
Delivery Partner ID: hc1
Partner Name: kundan
Partner Mobile: 432478893
Partner Salary: 7700
Manager ID: 200


PL/SQL procedure successfully completed.
```

## 7. CheckCustomerDiscountEligibility

```
-- Best customer based on rating for discount on next order
-- A customer is eligible if:

-- Their average customer rating (as given by sellers) is ≥ 3.0, and

-- They have placed at least 1 order, or

-- They have given consistently high ratings to sellers (i.e., are active & positive)
CREATE OR REPLACE PROCEDURE CheckCustomerDiscountEligibility(
    p_customer_id IN NUMBER
) IS
    v_avg_rating NUMBER := 0;
    v_order_count NUMBER := 0;
BEGIN
    -- Get the average customer rating (rated by sellers)
    SELECT NVL(AVG(customer_rating), 0)
    INTO v_avg_rating
    FROM Rating
    WHERE customer_id = p_customer_id;

    -- Get number of orders placed
    SELECT COUNT(*)
    INTO v_order_count
    FROM Orders
    WHERE customer_id = p_customer_id;

    -- Show details
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || p_customer_id);
    DBMS_OUTPUT.PUT_LINE('Average Rating by Sellers: ' || ROUND(v_avg_rating, 2));
    DBMS_OUTPUT.PUT_LINE('Total Orders: ' || v_order_count);

    -- Check eligibility
    IF v_avg_rating >= 3.0 AND v_order_count >= 1 THEN
        DBMS_OUTPUT.PUT_LINE(' Customer is eligible for a discount on the next order!');
    ELSE
        DBMS_OUTPUT.PUT_LINE(' Customer is not eligible for a discount.');
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for Customer ID: ' || p_customer_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
END;
/
EXEC CheckCustomerDiscountEligibility(2);
```

```
Customer ID: 2
Average Rating by Sellers: 4.1
Total Orders: 1
 Customer is eligible for a discount on the next order!
```

### 8. GetBestRestaurant

```
-- Best Restaurant
CREATE OR REPLACE PROCEDURE GetBestRestaurant IS
    v_seller_id Seller.seller_id%TYPE;
    v_name Seller.name%TYPE;
    v_avg_rating NUMBER;
BEGIN
    SELECT s.seller_id, s.name, AVG(r.seller_rating)
    INTO v_seller_id, v_name, v_avg_rating
    FROM Seller s
    JOIN Rating r ON s.seller_id = r.seller_id
    GROUP BY s.seller_id, s.name
    ORDER BY AVG(r.seller_rating) DESC
    FETCH FIRST 1 ROWS ONLY;

    DBMS_OUTPUT.PUT_LINE(' Best Restaurant: ' || v_name || ' (ID: ' || v_seller_id || ')');
    DBMS_OUTPUT.PUT_LINE('Average Rating: ' || ROUND(v_avg_rating, 2));
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No restaurant rating data found.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

exec GETBESTRESTAURANT;
```

```
Best Restaurant: Wrapchik (ID: 15)
Average Rating: 5
```

# Conclusion

***This project focuses on the core element of a college food delivery app - the database design. This robust relational database serves as the foundation for a mobile application that streamlines food ordering for students on campus.***

***Key features:***

- Stores information about users, restaurants, menus, orders, and addresses (optionally reviews).
- Enables efficient management of food orders placed by students.
- Establishes relationships between tables to ensure data consistency.
- Adheres to data normalization principles for optimized data manipulation.

***Benefits:***

- Provides a foundation for a mobile app that simplifies food ordering.
- Offers flexibility for future functionalities like reviews and promotions.
- Improves data organization and accessibility for managing food delivery services.

***Unity in Action:***

- **Communication and Transparency:** "Maintaining open communication channels throughout the project was crucial. We utilized a project management tool and held regular meetings to ensure everyone was up to date on tasks and progress."
- **Conflict Resolution:** "Inevitably, minor disagreements arose during the design phase. However, our commitment to open dialogue and a focus on the project's goals allowed us to reach mutually agreeable solutions."

***Celebrating Achievements:***

- **Shared Accomplishment:** "The final database design reflects the combined efforts of the entire team. We are proud of the secure, scalable, and **user-friendly foundation we've created for the college food delivery app."**
- **Future Collaboration:** "This project has strengthened our teamwork skills and fostered a sense of unity. We look forward to collaborating on future endeavors, leveraging the valuable lessons learned from this experience."

***Future Enhancements:***

**The database design allows for future functionalities:**

- **Promotional Offers:** Additional tables can be added to manage promotional offers and discounts offered by restaurants.
- **Delivery Driver Management:** A table for delivery drivers can be integrated for a comprehensive food delivery system.

## References


- Normalization Process in DBMS - GeeksforGeeks
- How to Convert ER Diagrams to Tables in DBMS? - GeeksforGeeks
- ER (Entity Relationship) Diagram Model in DBMS Examples - javatpoint
- Oracle Live SQL
- PL/SQL Introduction - GeeksforGeeks