Group number: 12 **Members:** Omkar Kadam, Mrunmayi Parker, Nahush Patil

DATABASE DESIGN FINAL PROJECT REPORT

A.] PROJECT DESCRIPTION

GlobalEats is a company that connects restaurants worldwide with customers through an online platform. GlobalEats would like to build a relational database to efficiently manage their operations. The database should handle the following key modules: Users, Employees, Restaurants, Menu Items, Orders, Payments, and Reviews.

A User could be a Customer or a Restaurant Owner or both. All users have common attributes like User ID, Name (First, Middle, Last), Address, Gender, Date of Birth, and Phone number. A user can have multiple phone numbers. Customers can place orders, and Restaurant Owners can manage restaurant profiles and menu items.

Each restaurant has a Restaurant ID, Restaurant Name, Cuisine Type, Location, and Operational Hours. The Restaurant Owner manages the restaurant. The database should store details about the operational hours, including opening and closing times for each day of the week. Restaurants can run Promotions. Each Promotion has a Promotion Code and Description. Promotion Codes are only unique within a particular restaurant.

Menu Items are associated with restaurants. Each Menu Item has an Item ID, Name, Description, Price, and Category (e.g., Appetizer, Main Course, Dessert, Beverage). Menu Items can have multiple categories. A customer can save items in a "Favorites" list.

Customers can place Orders. Each order records details such as Order ID, Date of Order, Total Amount, Payment Method (Credit Card, Debit Card, Digital Wallet, etc.), and Delivery Status. Orders are linked to both the Customer and the Restaurant. Each order can have multiple items from the menu item of the restaurant.

Customers can leave Reviews on menu items. Reviews include Review ID, Rating (1-5), Review Text, and Date of Review. Each review is associated with a specific menu item.

GlobalEats also employs Employees, who can be one of four roles: Platform Managers, Support Agents, Delivery Coordinators, and Delivery Drivers. Employees must be at least 18 years old. Each Employee has an Employee ID in the format “EXXX,” where X is a number from 0 to 9 (e.g., “E000,” “E999”). Employees also have attributes such as Start Date and Department.

- Platform Managers oversee the overall system and manage high-level operations.

- Support Agents handle inquiries from users. Each inquiry has its own ID and inquiry time. Support Agents must be trained by a Trainer, who can be either a Platform Manager or a Delivery Coordinator. Trainers have unique Trainer Certificates, and the certificate issuing date is recorded. One Trainer can train multiple Support Agents.
- Delivery Coordinators manage the logistics of delivery drivers, ensuring efficient order deliveries. They are responsible for assigning drivers to orders.
- Delivery Drivers, who are responsible for delivering orders. Delivery drivers have attributes such as Driver ID, Name, Contact Information, and Vehicle Type. A driver can deliver multiple orders, and an order can have only one assigned driver. Delivery details, including pickup and delivery times, must be stored.

B.] PROJECT QUESTIONS

1.] Would a superclass/subclass relationship be beneficial in the GlobalEats database design? Why or why not?

Yes, a superclass/subclass relationship would be beneficial for the GlobalEasts database design. Creating a superclass/subclass relationship is beneficial because it avoids repetition on attributes across entities. We can create a superclass entity and include all the common attributes and then create subclasses. If required, attributes specific to the instance of the superclass can be added. The database involves the entities like EMPLOYEE and USER where the use of subclass/superclass is helpful. There are multiple types of employees - platform manager, delivery coordinators, support agents and delivery drivers. The design requirements suggest that support agents must be trained by a Platform manager or delivery coordinator. Hence, trainer is a subclass for Platform manager and delivery coordinator with a union relationship. Similarly, the database design demands two types of Users - Customer and Restaurant_Owner. There are subclasses for the User entity.

2.] Can you think of 5 additional rules (other than those described above) that would likely be used in this environment? How would your design change to accommodate these rules?

1. Users can have multiple Phone numbers.
2. A restaurant can have multiple cuisines and cuisines can overlap
3. A restaurant can be present in multiple locations
4. A restaurant can have multiple shifts. Every restaurant can have different operational hours.
5. A single order is permitted to have more than one delivery.

For rule number 1, 2, 3, 4 we create a multivalued attribute for the ones that can take multiple values and later map it to relational schema by creating a separate relation out of it. For rule 5, we create a separate primary key ‘Delivery_ID’ as opposed to using the composite of {Order ID, Delivery Coordinator ID, Delivery Driver ID}

3.] Justify the use of a Relational DBMS like Oracle for this project (Successfully design a relational database system, and show all implementation in the final report at Phase IV).

The GlobalEats database design has various different entities and relationships. To capture the meaning of these entities with their attributes, we need to store them in a formatted manner using tables and columns. A relational database is a good choice for such a use case where all the data is stored in a structured way. We can use tools like Oracle DB to create the schema for the relations and define key constraints if required. The relations in RDBMS can be normalized to help in avoiding any form of redundancy that may arise in the future. It supports views that can help in creating custom relations by joining the existing ones without storing them in physical memory. Hence, using RDBMS would be justified in this case.

C.] EER DIAGRAM

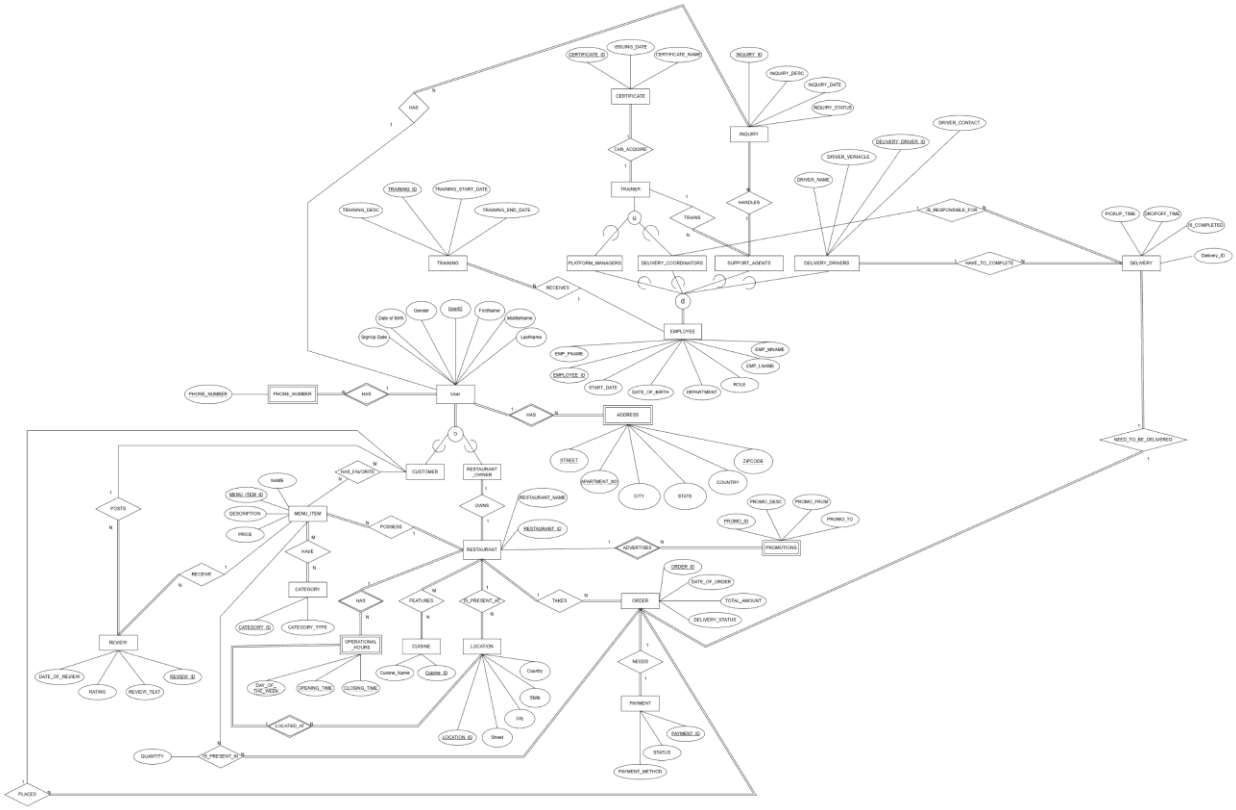

Figure 1: EER Diagram

Assumptions:

-A User can have multiple Addresses.
-Each Restaurant can only have one Owner.
-A Restaurant can be present in more than one Location.
-A single Restaurant can feature multiple Cuisines.
-Each Restaurant can have multiple shifts of Operational Hours for every Location.
-Menu Items don’t overlap in between Restaurants.
-A Customer can leave multiple reviews for the same Menu Item.
-An Order can have multiple Deliveries.

D.] RELATIONAL SCHEMA AFTER NORMALIZATION

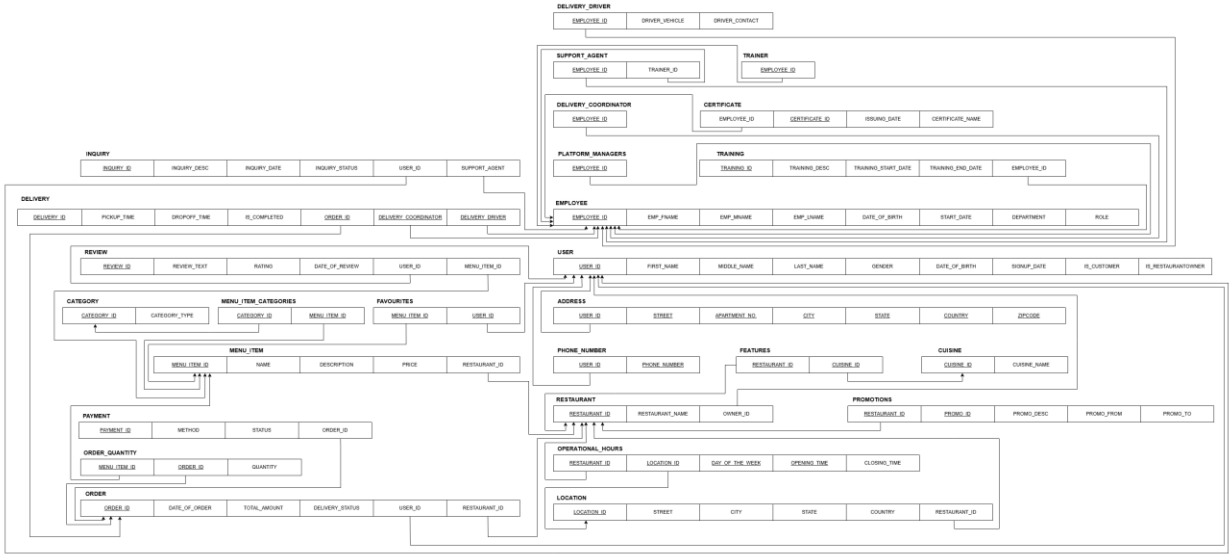

Figure 2: Relational Schema (after Normalization)

- The schemas have been normalized to 3NF.
- Primary keys are indicated by underlining the attributes that form them.
- Referential integrity is visually represented using arrows to denote foreign key relationships between tables.

E]. DEPENDENCY DIAGRAM

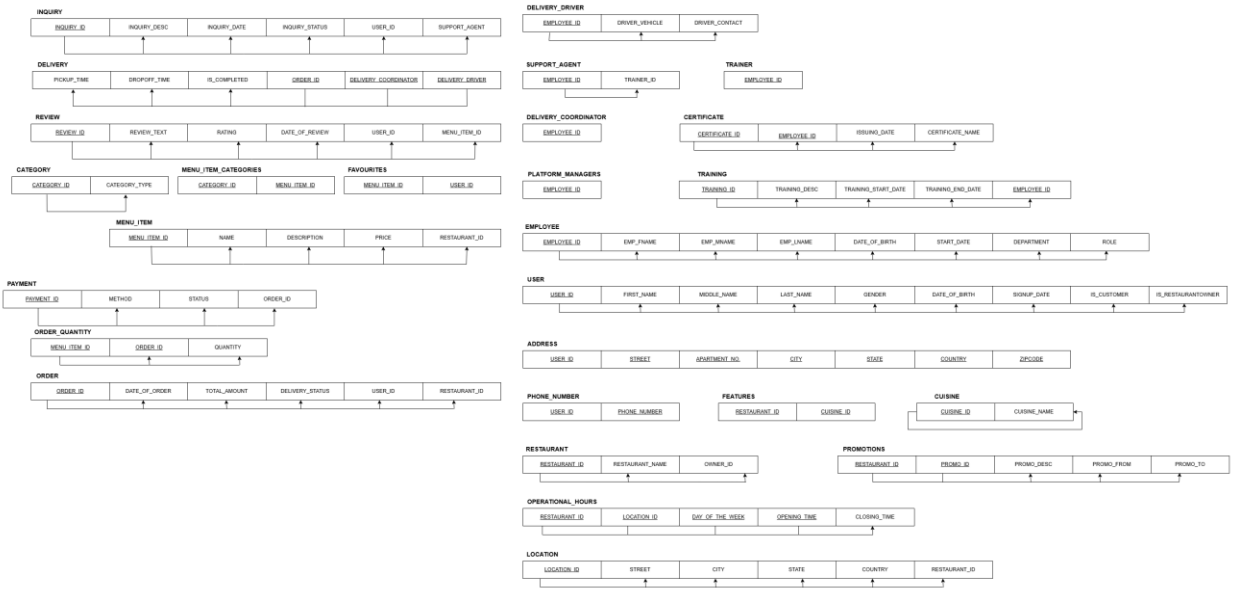

Figure 3: Dependency Diagram

F]. SQL STATEMENTS

- **SCHEMA CREATION:**

USERS TABLE

```
CREATE TABLE USERS (
   UserID              NUMBER PRIMARY KEY,
   First_Name          VARCHAR2(50) NOT NULL,
   Middle_Name         VARCHAR2(50),
   Last_Name           VARCHAR2(50) NOT NULL,
   Gender              VARCHAR2(10),
   DateofBirth         DATE,
   SignupDate          DATE NOT NULL,
   Is_Customer         NUMBER(1) DEFAULT 0 NOT NULL,
   Is_Restaurantowner  NUMBER(1) DEFAULT 0 NOT NULL,
   CONSTRAINT check_boolean_values CHECK (Is_Customer IN (0,1) AND Is_Restaurantowner IN (0,1))
);
```

ADDRESS TABLE

```
CREATE TABLE ADDRESS (
   UserID        NUMBER,
   Street        Varchar2(100)   NOT NULL,
   Apartment     Varchar2(50)    NOT NULL,
   City          Varchar2(100)   NOT NULL,
   State         Varchar2(100)   NOT NULL,
   Country       Varchar2(100)   NOT NULL,
   Zipcode       Varchar2(50)    NOT NULL,
   PRIMARY KEY (UserID, Street, Apartment, City, State, Country, Zipcode),
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE
);
```

PHONE_NUMBER TABLE

```
CREATE TABLE PHONE_NUMBER (
   UserID              NUMBER,
   Phone_Number        VARCHAR2(20) NOT NULL,
   PRIMARY KEY (UserID, Phone_Number),
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE
);
```

RESTAURANT TABLE

```
CREATE TABLE RESTAURANT (
   RestaurantID          NUMBER PRIMARY KEY,
   RestaurantName        VARCHAR2(50) NOT NULL,
   OwnerID               NUMBER,
   FOREIGN KEY (OwnerID) REFERENCES USERS(UserID) ON DELETE CASCADE,
   CONSTRAINT unique_restaurant_name UNIQUE (RestaurantName)
);
```

CUISINE TABLE

```
CREATE TABLE CUISINE(
   CuisineID             NUMBER PRIMARY KEY,
   CuisineName           VARCHAR2(50) NOT NULL
);
```

FEATURES TABLE

```
CREATE TABLE FEATURES (
   RestaurantID          NUMBER,
   CuisineID             NUMBER,
   PRIMARY KEY (RestaurantID, CuisineID),
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE,
   FOREIGN KEY (CuisineID) REFERENCES CUISINE(CuisineID) ON DELETE CASCADE
);
```

PROMOTIONS TABLE

```
CREATE TABLE PROMOTIONS (
   RestaurantID          NUMBER,
   PromoID               NUMBER NOT NULL,
   PromoDesc             VARCHAR2(200) NOT NULL,
   PromoFrom             DATE NOT NULL,
   PromoEnd              DATE NOT NULL,
   PRIMARY KEY (RestaurantID, PromoID),
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE,
   CONSTRAINT promo_dur CHECK (PromoFrom < PromoEnd)
);
```

LOCATIONS TABLE

```
CREATE TABLE LOCATIONS (
   LocationID          NUMBER PRIMARY KEY,
   City                VARCHAR2(100)   NOT NULL,
   State               VARCHAR2(100)   NOT NULL,
   Country             VARCHAR2(100)   NOT NULL,
   Zipcode             VARCHAR2(50)    NOT NULL,
   RestaurantID        NUMBER,
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE
);
```

OPERATIONAL_HOURS TABLE

```
CREATE TABLE OPERATIONAL_HOURS (
   RestaurantID        NUMBER,
   LocationID          NUMBER,
   Dayoftheweek        NUMBER CHECK (Dayoftheweek BETWEEN 1 AND 7) NOT NULL,
   Openingtime         DATE,
   Closingtime         DATE,
   PRIMARY KEY (RestaurantID, LocationID, Dayoftheweek, Openingtime),
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE,
   FOREIGN KEY (LocationID) REFERENCES LOCATIONS(LocationID) ON DELETE CASCADE
);
```

ORDER TABLE

```
CREATE TABLE ORDERS (
   OrderID             NUMBER PRIMARY KEY,
   DateofOrder         DATE NOT NULL,
   TotalAmount         NUMBER NOT NULL,
   DeliveryStatus      VARCHAR2(20) NOT NULL
                       CHECK (DeliveryStatus IN ('Pending', 'Delivered', 'In Progress','Canceled')),
   UserID              NUMBER,
   RestaurantID        NUMBER,
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE,
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE
);
```

MENU_ITEM TABLE

```
CREATE TABLE MENU_ITEM (
   MenuItemID          NUMBER PRIMARY KEY,
   Name                VARCHAR(50) NOT NULL,
   Description         VARCHAR(200) NOT NULL,
   Price               NUMBER NOT NULL,
   RestaurantID        NUMBER,
   FOREIGN KEY (RestaurantID) REFERENCES RESTAURANT(RestaurantID) ON DELETE CASCADE
);
```

ORDER_QUANTITY TABLE

```
CREATE TABLE ORDER_QUANTITY(
   MenuItemID          NUMBER,
   OrderID             NUMBER,
   Quantity            NUMBER NOT NULL,
   PRIMARY KEY (MenuItemID, OrderID),
   FOREIGN KEY (MenuItemID) REFERENCES MENU_ITEM(MenuItemID) ON DELETE CASCADE,
   FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID) ON DELETE CASCADE
);
```

REVIEW TABLE

```
CREATE TABLE REVIEW (
   ReviewID            NUMBER PRIMARY KEY,
   ReviewText          VARCHAR(500) NOT NULL,
   Rating              NUMBER  NOT NULL,
   DateofReview        DATE,
   UserID              NUMBER,
   MenuItemID          NUMBER,
   FOREIGN KEY (MenuItemID) REFERENCES MENU_ITEM(MenuItemID) ON DELETE CASCADE,
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE
);
```

PAYMENTS TABLE

```
CREATE TABLE PAYMENTS (
   PaymentID           NUMBER PRIMARY KEY,
   PMethod             VARCHAR(50) NOT NULL,
   Status              VARCHAR2(20)  NOT NULL,
   OrderID             NUMBER,
   FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID) ON DELETE CASCADE
  CONSTRAINT payment_status CHECK (Status IN ('PENDING', 'COMPLETED', 'FAILED', 'CANCELED'))
);
```

CATEGORY TABLE

```
CREATE TABLE CATEGORIES(
   CategoryID          NUMBER PRIMARY KEY,
   CategoryName        VARCHAR2(50) NOT NULL
);
```

MENU_ITEM_CATEGORIES TABLE

```
CREATE TABLE MENU_ITEM_CATEGORIES (
   MenuItemID          NUMBER,
   CategoryID          NUMBER,
   PRIMARY KEY (MenuItemID, CategoryID),
   FOREIGN KEY (MenuItemID) REFERENCES MENU_ITEM(MenuItemID) ON DELETE CASCADE,
   FOREIGN KEY (CategoryID) REFERENCES CATEGORIES(CategoryID) ON DELETE CASCADE
);
```

FAVORITE TABLE

```
CREATE TABLE FAVORITE (
   MenuItemID          NUMBER,
   UserID              NUMBER,
   PRIMARY KEY (MenuItemID, UserID),
   FOREIGN KEY (MenuItemID) REFERENCES MENU_ITEM(MenuItemID) ON DELETE CASCADE,
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE
);
```

DELIVERY TABLE

```
CREATE TABLE DELIVERY (
   DeliveryID          NUMBER PRIMARY KEY,
   PickupTime          TIMESTAMP DEFAULT NULL,
   DropoffTime         TIMESTAMP DEFAULT NULL,
   Is_Completed        NUMBER(1) DEFAULT 0 NOT NULL,
   OrderID             NUMBER,
   DelcoorID           VARCHAR(4),
   DeldrivID           VARCHAR(4),
   FOREIGN KEY (OrderID) REFERENCES ORDERS(OrderID) ON DELETE CASCADE,
   FOREIGN KEY (DelcoorID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE,
   FOREIGN KEY (DeldrivID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

Group number: 12 **Members:** Omkar Kadam, Mrunmayi Parker, Nahush Patil

EMPLOYEE TABLE

```
CREATE TABLE Employees (
   EmployeeID            VARCHAR(4) PRIMARY KEY,
   Emp_FName             VARCHAR2(50) NOT NULL,
   Emp_MName             VARCHAR2(50) NOT NULL,
   Emp_LName             VARCHAR2(50) NOT NULL,
   Date_of_Birth         DATE NOT NULL,
   Start_Date            DATE NOT NULL,
   Department            VARCHAR2(50) NOT NULL,
   EmpRole               VARCHAR2(50) NOT NULL,
   CONSTRAINT EmployeeID_Format CHECK (REGEXP_LIKE(EmployeeID, '^E[0-9]{3}$')),
   CONSTRAINT Role_Constraint CHECK (EmpRole IN ('PLATFORM MANAGER', 'DELIVERY
                                    COORDINATOR', 'SUPPORT AGENT', 'DELIVERY DRIVER')),
   CONSTRAINT Dept_Constraint CHECK (Department IN ('MANAGEMENT', 'DELIVERY
                                    COORDINATION', 'SUPPORT', 'DELIVERY'))
);
```

TRAINING TABLE

```
CREATE TABLE TRAINING (
   TrainingID            NUMBER PRIMARY KEY,
   TrainingDesc          VARCHAR(200) DEFAULT 'NONE' NOT NULL,
   TrainingFromDate      DATE NOT NULL,
   TrainingToDate        DATE NOT NULL,
   EmployeeID            VARCHAR(4),
   FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

CERTIFICATE TABLE

```
CREATE TABLE CERTIFICATE(
   CertificateID         VARCHAR(30) PRIMARY KEY,
   Issuing_Date          DATE NOT NULL,
   CertificateName       VARCHAR(100),
   EmployeeID            VARCHAR(4),
   FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

INQUIRY TABLE

```
CREATE TABLE INQUIRY(
   InquiryID       NUMBER PRIMARY KEY,
   InquiryDesc     VARCHAR(255) NOT NULL,
   InquiryDate     DATE NOT NULL,
   InquiryStatus   VARCHAR(20),
   UserID          NUMBER,
   EmployeeID      VARCHAR(4),
   FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE,
   FOREIGN KEY (UserID) REFERENCES USERS(UserID) ON DELETE CASCADE,
   CONSTRAINT inquiry_status CHECK (Status IN ('PENDING', 'RESOLVED'))
);
```

DELIVERY_DRIVER TABLE

```
CREATE TABLE DELIVERY_DRIVER(
   EmployeeID              VARCHAR(4),
   DriverVehicle           VARCHAR(30),
   DriverContact           VARCHAR(20),
   PRIMARY KEY(EmployeeID),
   FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

PLATFORM_MANAGER TABLE

```
CREATE TABLE PLATFORM_MANAGER(
   PlatManID      VARCHAR(4),
   PRIMARY KEY(PlatManID),
   FOREIGN KEY (PlatManID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

DELIVERY_COORDINATOR TABLE

```
CREATE TABLE DELIVERY_COORDINATOR (
   DelCoorID      VARCHAR(4),
   PRIMARY KEY(DelCoorID),
   FOREIGN KEY (DelCoorID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

TRAINER TABLE

```
CREATE TABLE TRAINER (
   TrainerID       VARCHAR(4),
   PRIMARY KEY(TrainerID),
   FOREIGN KEY (TrainerID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

SUPPORT_AGENT

```
CREATE TABLE SUPPORT_AGENT (
   SupportAgentID          VARCHAR(4),
   TrainerID               VARCHAR(4),
   PRIMARY KEY(EmployeeID),
   FOREIGN KEY (SupportAgentID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
   FOREIGN KEY (TrainerID) REFERENCES EMPLOYEES(EmployeeID) ON DELETE CASCADE
);
```

- **VIEW STATEMENTS-**

I] *TopCustomers*: View of customers who placed the most orders in the past month.

```
CREATE VIEW  TOP_CUSTOMERS
AS
SELECT       U.UserID, U.First_Name, U.Last_Name,
             COUNT(O.OrderID) AS TOTALORDERS_PASTMONTH
FROM         ORDERS O JOIN USERS U ON O.UserID = U.UserID
WHERE        O.DateofOrder >= ADD_MONTHS(SYSDATE,-1)
GROUP BY     U.UserID, U.First_Name, U.Last_Name
ORDER BY     TOTALORDERS_PASTMONTH DESC;
```

	USERID	FIRST_NAME	LAST_NAME	TOTALORDERS_PASTMONTH
1	6	Frank	Castle	5
2	9	Ivy	Adams	3
3	4	Diana	Prince	3
4	11	Karen	Walker	2
5	13	Mona	Lisa	2
6	5	Elena	Gilbert	1
7	3	Charlie	Brown	1

II] *PopularRestaurants*: View of the most ordered-from restaurants in the past year.

```
CREATE VIEW  POPULAR_RESTAURANTS
AS
SELECT       R.RestaurantID, R.RestaurantName, COUNT(O.OrderID) AS TotalOrders
FROM         ORDERS O JOIN RESTAURANT R ON O.RestaurantID = R.RestaurantID
WHERE        O.DateofOrder >= ADD_MONTHS(SYSDATE,-12)
GROUP BY     R.RestaurantID, R.RestaurantName
ORDER BY     TotalOrders DESC;
```

	RESTAURANTID	RESTAURANTNAME	TOTALORDERS
1	6	Henrys Diner	3
2	3	Charlies Grill	3
3	7	Jacks Joint	3
4	4	Elena Eatery	2
5	1	Alice Cafe	2
6	5	Graces Gourmet	2
7	2	Bobs Bistro	2

III] *HighlyRatedItems*: View of menu items that have an average rating of at least 4.5.

```
CREATE VIEW  HIGHLY_RATED_ITEMS
AS
SELECT              N.MenuItemID, N.Name, R.RestaurantID, R.RestaurantName
FROM                MENU_ITEM N JOIN RESTAURANT R
                    ON N.RestaurantID=R.RestaurantID
WHERE               N.MenuItemID IN (
                                SELECT      R.MenuItemID
                                FROM        REVIEW R JOIN MENU_ITEM M
                                            ON R.MenuItemID=M.MenuItem
                                GROUP BY    R.MenuItemID
                                HAVING      AVG(R.Rating)>=4.5);
```

	MENUITEMID	NAME	RESTAURANTID	RESTAURANTNAME
1	1	Espresso	1	Alice Cafe
2	22	Tiramisu	1	Alice Cafe
3	4	Grilled Chicken	2	Bobs Bistro
4	5	Cheeseburger	3	Charlies Grill
5	11	Vegan Burger	6	Henrys Diner

IV] *FrequentDrivers*: View of delivery drivers who have delivered the most orders in the past month.

```
CREATE VIEW  FrequentDrivers
AS
SELECT          E.EmployeeID AS DriverID, E.Emp_FName AS DriverFirstName,
                E.Emp_LName AS DriverLastName, COUNT(D.OrderID) AS DeliveredOrders
FROM            (Delivery D JOIN ORDERS O ON D.OrderID=O.OrderID)
                JOIN Employees E ON D.DeldrivID = E.EmployeeID
WHERE           O.DeliveryStatus = 'Delivered' AND D.Is_Completed = 1
                AND TRUNC(D.PickupTime)>=ADD_MONTHS(SYSDATE,-1)
                AND TRUNC(D.DropoffTime)<=SYSDATE
GROUP BY        D.DeldrivID, E.EmployeeID, E.Emp_FName, E.Emp_LName
ORDER BY        DeliveredOrders DESC
FETCH           FIRST 3 ROWS ONLY;
```

	DRIVERID	DRIVERFIRSTNAME	DRIVERLASTNAME	DELIVEREDORDERS
1	E003	John	Smith	6
2	E005	Michael	Brown	3
3	E004	Emily	Johnson	3

V] *PotentialOwners*: View of customers who have added at least 10 menu items to their Favorites list but have not yet registered as Restaurant Owners.

```
CREATE VIEW  POTENTIAL_OWNERS
AS
SELECT       U.UserID, U.First_Name, U.Last_Name,
             COUNT(F.MenuItemID) AS NoofFavorites
FROM         FAVORITE F JOIN USERS U ON F.UserID=U.UserID
WHERE        Is_Restaurantowner = 0
GROUP BY     U.UserID, U.First_Name, U.Last_Name
HAVING       COUNT(F.MenuItemID)>=10;
```

	USERID	FIRST_NAME	LAST_NAME	NOOFFAVORITES
1	9	Ivy	Adams	10

- **SQL QUERIES -**

1]. List details of restaurant owners who have signed up within the past three months.

```
SELECT      UserID, First_Name, Last_Name, SignupDate
FROM        USERS
WHERE       Is_Restaurantowner = 1
AND         SignupDate >= ADD_MONTHS(SYSDATE, -3);
```

	USERID	FIRST_NAME	LAST_NAME	SIGNUPDATE
1	1	Alice	Johnson	2024-10-02
2	3	Charlie	Brown	2024-11-16
3	7	Grace	Hopper	2024-10-17
4	10	Jack	Ryan	2024-11-01
5	12	Leo	Tolstoy	2024-11-21
6	14	Nathan	Drake	2024-11-26
7	15	Olivia	Pope	2024-09-17
8	27	Lily	Smith	2024-09-10
9	28	Olivia	Johnson	2024-09-10

2]. Find the names of customers who placed orders with only two restaurants in the past month.

```
SELECT      U.First_Name, U.Last_Name, U.UserID
FROM        USERS U JOIN ORDERS O ON U.UserID=O.UserID
WHERE       O.DateofOrder>=SYSDATE-30
GROUP BY    U.First_Name, U.Last_Name, U.UserID
HAVING      COUNT(DISTINCT O.RestaurantID)=2;
```

	FIRST_NAME	LAST_NAME	USERID
1	Mona	Lisa	13
2	Karen	Walker	11
3	Ivy	Adams	9
4	Diana	Prince	4

3]. Calculate the average number of orders placed by the top five customers in the platform.

```
SELECT      AVG(OrderCount) AS AverageOrders
FROM   (
            SELECT      UserID, COUNT(*) AS OrderCount
            FROM   Orders
            GROUP BY    UserID
            ORDER BY    OrderCount DESC
            ) TopCustomers
WHERE       ROWNUM <= 5;
```

	AVERAGEORDERS
1	3.8

4]. List the name of each restaurant and its most popular menu item.

```
WITH RESTNAMES AS (
        SELECT      M.RestaurantID,  M.MenuItemID,
                    SUM(Q.Quantity) AS TotalQuantity
        FROM        MENU_ITEM M JOIN ORDER_QUANTITY Q
                    ON M.MenuItemID = Q.MenuItemID
        GROUP BY    M.RestaurantID, M.MenuItemID
),
RANKED_ITEMS AS (
        SELECT      RestaurantID, MenuItemID, TotalQuantity,
                    RANK() OVER
                    (PARTITION BY RestaurantID ORDER BY TotalQuantity DESC)
                    AS rank
        FROM        RESTNAMES
)
SELECT      R.RestaurantName, M.Name AS MostPopularItem
FROM        (RESTAURANT R JOIN RANKED_ITEMS RI
            ON R.RestaurantID = RI.RestaurantID) JOIN
            MENU_ITEM M ON RI.MenuItemID = M.MenuItemID
WHERE       RI.rank = 1;
```

	RESTAURANTNAME	MOSTPOPULARITEM
1	Alice Cafe	Cappuccino
2	Bobs Bistro	Caesar Salad
3	Charlies Grill	Cheeseburger
4	Elena Eatery	Margherita Pizza
5	Graces Gourmet	Tempura
6	Henrys Diner	Vegan Burger
7	Henrys Diner	Fries
8	Jacks Joint	Bulgogi

5] Identify menu items that haven't been ordered in the last six months.

```
WITH MI AS (
        SELECT      Q.MenuItemID
        FROM        ORDER_QUANTITY Q
        JOIN    ORDERS O ON Q.OrderID = O.OrderID
        WHERE       O.DateOfOrder > ADD_MONTHS(SYSDATE, -6)
),
MU AS (
        SELECT      M.ITEMID
        FROM    MENU_ITEM M
)
SELECT      ITEMID
FROM        MU
MINUS
SELECT      MENU_ITEM_ID
FROM        MI;
```

	MENUITEMID
1	4
2	8
3	9
4	15
5	16
6	17
7	19
8	20
9	21
10	22
11	24
12	25

6] Find customers who have reviewed all the items from a specific restaurant.

```
WITH REST_MENU_ITEMS AS(
        SELECT      COUNT(M.MenuItemID) as COUNTITEMS
        FROM        Menu_Item M
        WHERE       M.RestaurantID=1
),
USER_REVW_ITEMS AS(
        SELECT      U.First_Name, U.Last_Name, R.UserID,
                    COUNT(DISTINCT R.MenuItemID) as REVIEWCOUNT
        FROM        ((REVIEW R JOIN USERS U ON R.UserID=U.UserID)
                    JOIN MENU_ITEM I ON R.MenuItemID=I.MenuItemID)
                    JOIN RESTAURANT T ON I.RestaurantID=T.RestaurantID
        WHERE       T.RestaurantID=1
        GROUP BY    R.UserID, U.First_Name, U.Last_Name
)
SELECT      URT.First_Name,URT.Last_Name
FROM        USER_REVW_ITEMS URT, REST_MENU_ITEMS RMI
WHERE       RMI.COUNTITEMS - URT.REVIEWCOUNT = 0;
```

	FIRST_NAME	LAST_NAME
1	Diana	Prince

7]. Identify the restaurant with the most promotions' amount in the past year

```
SELECT      R.RestaurantID as SrNo, R.RestaurantName as TOP_RESTAURANTS,
            COUNT(PROMOID) AS TOTALNOOFPROMOTIONS
FROM        RESTAURANT R JOIN PROMOTIONS P ON
            R.RestaurantID=P.RestaurantID
WHERE       PromoFrom>=ADD_MONTHS(SYSDATE,-12)
GROUP BY    R.RestaurantID, R.RestaurantName
ORDER BY    COUNT(PROMOID) DESC
FETCH       FIRST 3 ROWS ONLY;
```

	SRNO	TOP_RESTAURANTS	TOTALNOOFPROMOTIONS
1	3	Charlies Grill	4
2	1	Alice Cafe	4
3	5	Graces Gourmet	4

8]. Find the year with the highest total order payment.

```
SELECT      EXTRACT(YEAR FROM DateofOrder) AS OrderYear,
            SUM(TotalAmount) AS TotalOrderAmount
FROM        Orders
WHERE       DeliveryStatus='Delivered'
GROUP BY    EXTRACT(YEAR FROM DateofOrder)
ORDER BY    TotalOrderAmount DESC
FETCH       FIRST 1 ROWS ONLY;
```

	ORDER...	TOTALORDERAMOUNT
1	2024	433

9]. List the names of customers who ordered the most popular menu items.

```
WITH RESTNAMES AS (
        SELECT      M.RestaurantID, M.MenuItemID, SUM(Q.Quantity) AS TotalQuantity
        FROM        MENU_ITEM M JOIN ORDER_QUANTITY Q
                    ON M.MenuItemID = Q.MenuItemID
        GROUP BY    M.RestaurantID, M.MenuItemID
),
RANKED_ITEMS AS (
        SELECT      RestaurantID, MenuItemID, TotalQuantity,
                    RANK() OVER
                    (PARTITION BY RestaurantID ORDER BY TotalQuantity DESC)
                    AS rank
        FROM        RESTNAMES
)
SELECT      First_Name, Last_Name, RI.RestaurantID
FROM        ((ORDER_QUANTITY Q JOIN RANKED_ITEMS RI
            ON Q.MenuItemID=RI.MenuItemID)
            JOIN ORDERS O ON Q.OrderID=O.OrderID)
            JOIN USERS U ON O.UserID=U.UserID
WHERE       RANK=1;
```

	FIRST_NAME	LAST_NAME	RESTAURANTID
1	Diana	Prince	1
2	Diana	Prince	2
3	Frank	Castle	3
4	Frank	Castle	3
5	Frank	Castle	4
6	Frank	Castle	4
7	Frank	Castle	5
8	Mona	Lisa	5
9	Ivy	Adams	6
10	Ivy	Adams	6
11	Karen	Walker	6
12	Ivy	Adams	6
13	Karen	Walker	6
14	Mona	Lisa	7

10]. **Find delivery drivers who have delivered at least 10 orders in the past month.**

```
SELECT      D.DeldrivID AS DriverID, E.Emp_FName AS FirstName,
            E.Emp_LName AS LastName, COUNT(D.OrderID) AS TotalDeliveries
FROM        DELIVERY D JOIN ORDERS O ON D.OrderID = O.OrderID
            JOIN EMPLOYEES E ON D.DeldrivID = E.EmployeeID
WHERE       D.Is_Completed = 1 AND
            O.DateofOrder >= ADD_MONTHS(SYSDATE, -1)
GROUP BY    D.DeldrivID, E.Emp_FName, E.Emp_LName
HAVING      COUNT(D.OrderID) >= 10
ORDER BY    TotalDeliveries DESC;
```

	DRIVERID	FIRSTNAME	LASTNAME	TOTALDELIVERIES
1	E003	John	Smith	10

11]. List customers who have been active for more than two years.

```
SELECT      UserID, First_Name, Middle_Name, Last_Name, SignupDate
FROM        USERS
WHERE       Is_Customer = 1 AND (Signupdate >= ADD_MONTHS(SYSDATE, -24)
            OR UserID in (
                    SELECT      O.UserID
                    FROM        ORDERS O
                    WHERE       DateofOrder>=ADD_MONTHS(SYSDATE,-24)));
```

	USERID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	SIGNUPDATE
1	18	Rita	N.	Skeeter	2023-07-21
2	20	Tina	(null)	Fey	2024-02-01
3	23	Willow	R.	Smith	2023-03-25
4	4	Diana	(null)	Prince	2024-11-11
5	5	Elena	C.	Gilbert	2023-06-01
6	6	Frank	(null)	Castle	2024-01-10
7	9	Ivy	F.	Adams	2024-02-05
8	11	Karen	H.	Walker	2023-12-10
9	13	Mona	(null)	Lisa	2024-03-01

12]. Find the number of orders delivered by the top three delivery drivers.

```
SELECT      E.EmployeeID AS DriverID, E.Emp_FName AS DriverFirstName,
            E.Emp_LName AS DriverLastName,
            COUNT(D.OrderID) AS DeliveredOrders
FROM        (Delivery D JOIN ORDERS O ON D.OrderID=O.OrderID)
            JOIN Employees E ON D.DeldrivID = E.EmployeeID
WHERE       O.DeliveryStatus = 'Delivered' AND D.Is_Completed = 1
GROUP BY    D.DeldrivID, E.EmployeeID, E.Emp_FName, E.Emp_LName
ORDER BY    DeliveredOrders DESC
FETCH       FIRST 3 ROWS ONLY;
```

	DRIVERID	DRIVERFIRSTNAME	DRIVERLASTNAME	DELIVEREDORDERS
1	E003	John	Smith	6
2	E005	Michael	Brown	3
3	E004	Emily	Johnson	3

13] List the restaurant owner who manages the most restaurants.

```
SELECT      U.UserID AS OwnerID, U.First_Name AS OwnerFirstName,
            U.Last_Name AS OwnerLastName,
            COUNT(R.RestaurantID) AS NumberOfRestaurants
FROM        USERS U JOIN RESTAURANT R ON U.UserID = R.OwnerID
GROUP BY    U.UserID, U.First_Name, U.Last_Name
ORDER BY    NumberOfRestaurants DESC
FETCH       FIRST 1 ROWS ONLY;
```

	OWNERID	OWNERFIRSTNAME	OWNERLASTNAME	NUMBEROFRESTAURANTS
1	28	Olivia	Johnson	3

14] Identify restaurants that have run promotions in every quarter of the past year.

```
SELECT      R.RestaurantID, R.RestaurantName
FROM        RESTAURANT R JOIN PROMOTIONS P
            ON R.RestaurantID = P.RestaurantID
WHERE       EXTRACT(YEAR FROM P.PromoFrom) = 2023
GROUP BY    R.RestaurantID, R.RestaurantName
HAVING      COUNT(DISTINCT TO_NUMBER(TO_CHAR(P.PromoFrom, 'Q'))) = 4;
```

	RESTAURANTID	RESTAURANTNAME
1	5	Graces Gourmet
2	1	Alice Cafe

15] List all employees who are also restaurant owners, and display their employee details along with the details of the restaurant they own.

```
SELECT      E.Emp_Fname, E.Emp_Lname, E.Start_Date, E.Department, E.EmpRole,
            R.RestaurantName
FROM        (EMPLOYEES E JOIN USERS U ON (E.Emp_Fname=U.First_Name
            AND E.Emp_Lname=U.Last_Name AND E.Date_of_Birth=U.DateofBirth))
            JOIN RESTAURANT R ON U.UserID=R.OwnerID
WHERE       Is_RestaurantOwner=1;
```

	EMP_FNAME	EMP_LNAME	START_DATE	DEPARTMENT	EMPROLE	RESTAURANTNAME
1	Alice	Johnson	2023-05-15	MANAGEMENT	PLATFORM MANAGER	Alice Cafe

16]. List the names and contact information of all employees who were hired before a specific date but have not received any new training since that date.

```
SELECT      E.EmployeeID,E.Emp_FName,E.Emp_MName, E.Emp_LName, E.EmpRole,
            E.Department, E.Start_Date
FROM        Employees E LEFT JOIN Training T ON E.EmployeeID = T.EmployeeID
WHERE       E.Start_Date > DATE '2022-01-01'AND
            (T.TrainingToDate IS NULL OR t.TrainingToDate < DATE '2022-01-01');
```

	EMPLOYEEID	EMP_FNAME	EMP_MNAME	EMP_LNAME	EMPROLE	DEPARTMENT	START_DATE
1	E005	Michael	C.	Brown	DELIVERY DRIVER	DELIVERY	2023-06-15