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
);
```

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
);
```

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);
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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)));
```

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;
```

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;
```

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;
```

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;
```

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');
```

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;
```

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;
```

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);
```

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;
```

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;
```