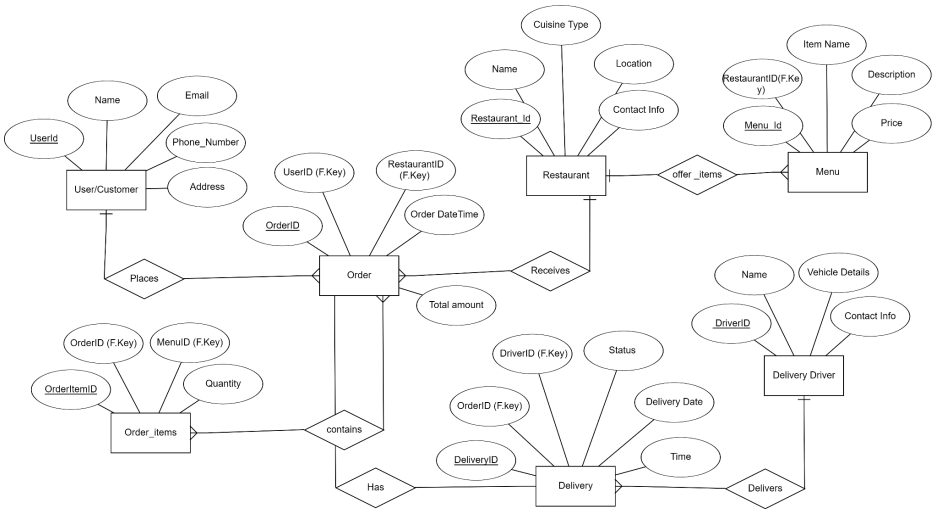
Description about the case study

E-R Model

Logical DB Design



Worksheet Query Builder

```
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Phone VARCHAR(20) UNIQUE NOT NULL,
    Address VARCHAR(255) NOT NULL
);
```

Worksheet Query Builder

```
INSERT INTO Users (UserID, Name, Email, Phone, Address)
VALUES
    (1, 'John Doe', 'john@example.com', '123-456-7890', '123 Main St');
INSERT INTO Users (UserID, Name, Email, Phone, Address)
VALUES
    (2, 'Jane Smith', 'jane@example.com', '987-654-3210', '456 Oak Ave');
INSERT INTO Users (UserID, Name, Email, Phone, Address)
VALUES
    (3, 'Michael Johnson', 'michael@example.com', '555-123-4567', '789 Elm St');
INSERT INTO Users (UserID, Name, Email, Phone, Address)
VALUES
    (4, 'Emily Davis', 'emily@example.com', '111-222-3333', '456 Pine Rd');
INSERT INTO Users (UserID, Name, Email, Phone, Address)
VALUES
    (5, 'Sarah Wilson', 'sarah@example.com', '333-555-7777', '1010 Broadway');
```

Worksheet | Query Builder

```
select * from Users;
```

Query Result

SQL | All Rows Fetched: 5 in 0.083 seconds

	USERID	NAME	EMAIL	PHONE	ADDRESS
1	1	John Doe	john@example.com	123-456-7890	123 Main St
2	2	Jane Smith	jane@example.com	987-654-3210	456 Oak Ave
3	3	Michael Johnson	michael@example.com	555-123-4567	789 Elm St
4	4	Emily Davis	emily@example.com	111-222-3333	456 Pine Rd
5	5	Sarah Wilson	sarah@example.com	333-555-7777	1010 Broadway

Worksheet | Query Builder

```
CREATE TABLE Restaurant (
    RestaurantID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    CuisineType VARCHAR(255),
    Location VARCHAR(255) NOT NULL,
    ContactInfo VARCHAR(255) NOT NULL
);
```

Worksheet | Query Builder

```
INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
VALUES
    (101, 'Pizza Palace', 'Italian', '789 Elm St', 'info@pizzapalace.com');
INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
VALUES
    (102, 'Burger Barn', 'American', '456 Maple Ave', 'info@burgerbarn.com');
INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
VALUES
    (103, 'Sushi House', 'Japanese', '123 Sakura Rd', 'info@sushihouse.com');
INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
VALUES
    (104, 'Taco Town', 'Mexican', '777 Jalapeno St', 'info@tacotown.com');
INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
VALUES
    (105, 'Curry Corner', 'Indian', '555 Spice Ave', 'info@currycorner.com');
```

Worksheet | Query Builder

```
select * from Restaurant ;
```

Query Result

SQL | All Rows Fetched: 5 in 0.008 seconds

	RESTAURANTID	NAME	CUISINETYPE	LOCATION	CONTACTINFO
1	101	Pizza Palace	Italian	789 Elm St	info@pizzapalace.com
2	102	Burger Barn	American	456 Maple Ave	info@burgerbarn.com
3	103	Sushi House	Japanese	123 Sakura Rd	info@sushihouse.com
4	104	Taco Town	Mexican	777 Jalapeno St	info@tacotown.com
5	105	Curry Corner	Indian	555 Spice Ave	info@currycorner.com

Worksheet | Query Builder

```
CREATE TABLE Menu (
    MenuID INT PRIMARY KEY,
    RestaurantID INT,
    ItemName VARCHAR(255) NOT NULL,
    Description varchar(30),
    Price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)
);
```

Worksheet | Query Builder

```
INSERT INTO Menu (MenuID, RestaurantID, ItemName, Description, Price)
VALUES
    (1001, 101, 'Margherita Pizza', 'Classic tomato and mozzarella', 12.99
INSERT INTO Menu (MenuID, RestaurantID, ItemName, Description, Price)
VALUES
    (1002, 101, 'Pepperoni Pizza', 'Tomato, mozzarella', 14.99);
INSERT INTO Menu (MenuID, RestaurantID, ItemName, Description, Price)
VALUES
    (2001, 102, 'Bacon Cheeseburger', 'Beef patty with bacon', 8.99);
INSERT INTO Menu (MenuID, RestaurantID, ItemName, Description, Price)
VALUES
    (2002, 102, 'Veggie Burger', 'Vegetarian patty', 7.99);
INSERT INTO Menu (MenuID, RestaurantID, ItemName, Description, Price)
VALUES
    (3001, 103, 'Sashimi Platter', 'AssortmenT of fresh sashimi', 22.99);
```

Worksheet | Query Builder

```
select * from Menu;
```

Query Result

SQL | All Rows Fetched: 5 in 0.224 seconds

	MENUID	RESTAURANTID	ITEMNAME	DESCRIPTION	PRICE
1	1001	101	Margherita Pizza	Classic tomato and mozzarella	12.99
2	1002	101	Pepperoni Pizza	Tomato, mozzarella	14.99
3	2001	102	Bacon Cheeseburger	Beef patty with bacon	8.99
4	2002	102	Veggie Burger	Vegetarian patty	7.99
5	3001	103	Sashimi Platter	AssortmenT of fresh sashimi	22.99

Worksheet | Query Builder

```sql
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    UserID INT,
    RestaurantID INT,
    OrderDateTime varchar(35)  NOT NULL,
    TotalAmount DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)
);
```

Worksheet | Query Builder

```sql
INSERT INTO Orders (OrderID, UserID, RestaurantID, OrderDateTime, TotalAmount)
VALUES
    (10001, 1, 101, '2023-10-04 12:00:00', 27.98);
INSERT INTO Orders (OrderID, UserID, RestaurantID, OrderDateTime, TotalAmount)
VALUES (10002, 2, 102, '2023-10-04 12:30:00', 16.98);
INSERT INTO Orders (OrderID, UserID, RestaurantID, OrderDateTime, TotalAmount)
VALUES
    (10003, 3, 103, '2023-10-04 13:00:00', 39.98);
INSERT INTO Orders (OrderID, UserID, RestaurantID, OrderDateTime, TotalAmount)
VALUES
    (10004, 4, 104, '2023-10-04 13:30:00', 13.98);
INSERT INTO Orders (OrderID, UserID, RestaurantID, OrderDateTime, TotalAmount)
VALUES
    (10005, 5, 105, '2023-10-04 14:00:00', 25.98);
```

Worksheet | Query Builder

```sql
select * from Orders;
```

Query Result

SQL | All Rows Fetched: 5 in 0.727 seconds

	ORDERID	USERID	RESTAURANTID	ORDERDATETIME	TOTALAMOUNT
1	10001	1	101	2023-10-04 12:00:00	27.98
2	10002	2	102	2023-10-04 12:30:00	16.98
3	10003	3	103	2023-10-04 13:00:00	39.98
4	10004	4	104	2023-10-04 13:30:00	13.98
5	10005	5	105	2023-10-04 14:00:00	25.98

Worksheet | Query Builder

```sql
CREATE TABLE OrderItems (
    OrderItemID INT PRIMARY KEY,
    OrderID INT,
    MenuID INT,
    Quantity INT NOT NULL
);
```

Worksheet | Query Builder

```sql
INSERT INTO OrderItems (OrderItemID, OrderID, MenuID, Quantity)
VALUES
    (5001, 10001, 1001, 1);
INSERT INTO OrderItems (OrderItemID, OrderID, MenuID, Quantity)
VALUES
    (5002, 10001, 1002, 1);
INSERT INTO OrderItems (OrderItemID, OrderID, MenuID, Quantity)
VALUES
    (5003, 10002, 2001, 2);
INSERT INTO OrderItems (OrderItemID, OrderID, MenuID, Quantity)
VALUES
    (5004, 10003, 3001, 1);
INSERT INTO OrderItems (OrderItemID, OrderID, MenuID, Quantity)
VALUES
    (5005, 10003, 3002, 1);
```

Worksheet | Query Builder

```sql
select * from OrderItems;
```

Query Result

SQL | All Rows Fetched: 5 in 0.237 seconds

	ORDERITEMID	ORDERID	MENUID	QUANTITY
1	5001	10001	1001	1
2	5002	10001	1002	1
3	5003	10002	2001	2
4	5004	10003	3001	1
5	5005	10003	3002	1

Worksheet | Query Builder

```sql
CREATE TABLE DeliveryDriver (
    DriverID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    VehicleDetails VARCHAR(255),
    ContactInfo VARCHAR(255) NOT NULL
);
```

Worksheet | Query Builder

```sql
INSERT INTO DeliveryDriver (DriverID, Name, VehicleDetails, ContactInfo)
VALUES
    (501, 'Sam Johnson', 'Car - ABC123', 'sam@example.com');
INSERT INTO DeliveryDriver (DriverID, Name, VehicleDetails, ContactInfo)
VALUES
    (502, 'Emily Davis', 'Bike - XYZ789', 'emily@example.com');
INSERT INTO DeliveryDriver (DriverID, Name, VehicleDetails, ContactInfo)
VALUES
    (503, 'Michael Brown', 'Car - DEF456', 'michael@example.com');
INSERT INTO DeliveryDriver (DriverID, Name, VehicleDetails, ContactInfo)
VALUES
    (504, 'Lisa Smith', 'Motorcycle - MN0789', 'lisa@example.com');
INSERT INTO DeliveryDriver (DriverID, Name, VehicleDetails, ContactInfo)
VALUES
    (505, 'Daniel Wilson', 'Car - GHI123', 'daniel@example.com');
```

Worksheet | Query Builder

```sql
select * from DeliveryDriver;
```

Query Result

SQL | All Rows Fetched: 5 in 9.527 seconds

	DRIVERID	NAME	VEHICLEDETAILS	CONTACTINFO
1	501	Sam Johnson	Car - ABC123	sam@example.com
2	502	Emily Davis	Bike - XYZ789	emily@example.com
3	503	Michael Brown	Car - DEF456	michael@example.com
4	504	Lisa Smith	Motorcycle - MN0789	lisa@example.com
5	505	Daniel Wilson	Car - GHI123	daniel@example.com

Worksheet | Query Builder

```sql
CREATE TABLE Delivery (
    DeliveryID INT PRIMARY KEY,
    OrderID INT UNIQUE,
    DriverID INT,
    Status varchar(30) NOT NULL,
    DeliveryDateTime VARCHAR(35),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (DriverID) REFERENCES DeliveryDriver(DriverID)
);
```

Worksheet | Query Builder

```sql
INSERT INTO Delivery (DeliveryID, OrderID, DriverID, Status, DeliveryDateTime)
VALUES
    (1001, 10001, 501, 'Delivered', '2023-10-04 12:45:00');
INSERT INTO Delivery (DeliveryID, OrderID, DriverID, Status, DeliveryDateTime)
VALUES
    (1002, 10002, 502, 'In Transit', NULL);
INSERT INTO Delivery (DeliveryID, OrderID, DriverID, Status, DeliveryDateTime)
VALUES
    (1003, 10003, 503, 'Delivered', '2023-10-04 13:15:00');
INSERT INTO Delivery (DeliveryID, OrderID, DriverID, Status, DeliveryDateTime)
VALUES
    (1004, 10004, 504, 'Delivered', '2023-10-04 13:30:00');
INSERT INTO Delivery (DeliveryID, OrderID, DriverID, Status, DeliveryDateTime)
VALUES
    (1005, 10005, 505, 'In Transit', NULL);
```

Worksheet | Query Builder

```sql
select * from Delivery;
```

Query Result

SQL | All Rows Fetched: 5 in 1.666 seconds

	DELIVERYID	ORDERID	DRIVERID	STATUS	DELIVERYDATETIME
1	1001	10001	501	Delivered	2023-10-04 12:45:00
2	1002	10002	502	In Transit	(null)
3	1003	10003	503	Delivered	2023-10-04 13:15:00
4	1004	10004	504	Delivered	2023-10-04 13:30:00
5	1005	10005	505	In Transit	(null)

Execute SQL Queries:

- 1.Simple queries
- 2.Nested Queries
- 3.Correlated Nested Queries
- 4. Set Comparison Operators
- 5.Queries using groupby and having
- 6.Joins

1.Inner Join to Retrieve Order Details with User Information:

Worksheet | Query Builder

```
SELECT Orders.OrderID, Users.Name AS CustomerName, Restaurant.Name AS
RestaurantName, Orders.OrderDateTime, Orders.TotalAmount
FROM Orders
INNER JOIN Users ON Orders.UserID = Users.UserID
INNER JOIN Restaurant ON Orders.RestaurantID = Restaurant.RestaurantID;
```

Script Output

Task completed in 0.09 seconds

```
   ORDERID CUSTOMERNAME
---------- ----------------------------------------
     10001 John Doe
     10002 Jane Smith
     10003 Michael Johnson
     10004 Emily Davis
     10005 Sarah Wilson
```

Compiler - Log

Messages | Statements | Compiler

Line 2 Column 1 | Insert

```sql
SELECT Orders.OrderID, Users.Name AS CustomerName, Restaurant.Name AS RestaurantName, Orders.OrderDateTime, Orders.Total
FROM Orders
INNER JOIN Users ON Orders.UserID = Users.UserID
INNER JOIN Restaurant ON Orders.RestaurantID = Restaurant.RestaurantID;
```

OrderID	CustomerName	RestaurantName	OrderDateTime	TotalAmount
10001	John Doe	Pizza Palace	2023-10-04 12:00:00	27.98
10002	Jane Smith	Burger Barn	2023-10-04 12:30:00	16.98
10003	Michael Johnson	Sushi House	2023-10-04 13:00:00	39.98
10004	Emily Davis	Taco Town	2023-10-04 13:30:00	13.98
10005	Sarah Wilson	Curry Corner	2023-10-04 14:00:00	25.98

2. Left Join to Retrieve Orders with or without Delivery Information:

SQLite | SQLite.1 | SQLite.3

```sql
SELECT Orders.OrderID, Users.Name AS CustomerName, Delivery.Status, Delivery.DeliveryDateTime
FROM Orders
LEFT JOIN Users ON Orders.UserID = Users.UserID
LEFT JOIN Delivery ON Orders.OrderID = Delivery.OrderID;
```

OrderID	CustomerName	Status	DeliveryDateTime
10001	John Doe	Delivered	2023-10-04 12:45:00
10002	Jane Smith	In Transit	NULL
10003	Michael Johnson	Delivered	2023-10-04 13:15:00
10004	Emily Davis	Delivered	2023-10-04 13:30:00
10005	Sarah Wilson	In Transit	NULL

3.Right Join to Retrieve Delivery Information with Order Details:

SQLite | SQLite.1 | SQLite.3

```sql
SELECT Delivery.DeliveryID, Orders.OrderID, Users.Name AS CustomerName, Delivery.Status, Delivery.DeliveryDateTime
FROM Delivery
RIGHT JOIN Orders ON Delivery.OrderID = Orders.OrderID
RIGHT JOIN Users ON Orders.UserID = Users.UserID;
```

DeliveryID	OrderID	CustomerName	Status	DeliveryDateTime
1001	10001	John Doe	Delivered	2023-10-04 12:45:00
1002	10002	Jane Smith	In Transit	NULL
1003	10003	Michael Johnson	Delivered	2023-10-04 13:15:00
1004	10004	Emily Davis	Delivered	2023-10-04 13:30:00
1005	10005	Sarah Wilson	In Transit	NULL

4. Full Outer Join to Retrieve All Orders and Deliveries:

SQLite | SQLite.1 | SQLite.3

```sql
SELECT Orders.OrderID, Users.Name AS CustomerName, Delivery.Status, Delivery.DeliveryDateTime
FROM Orders
FULL OUTER JOIN Delivery ON Orders.OrderID = Delivery.OrderID
FULL OUTER JOIN Users ON Orders.UserID = Users.UserID;
```

OrderID	CustomerName	Status	DeliveryDateTime
10001	John Doe	Delivered	2023-10-04 12:45:00
10002	Jane Smith	In Transit	NULL
10003	Michael Johnson	Delivered	2023-10-04 13:15:00
10004	Emily Davis	Delivered	2023-10-04 13:30:00
10005	Sarah Wilson	In Transit	NULL

5. Inner join to retrieve Delivery Drivers with Delivery Information:

SQLite | SQLite.1 | SQLite.3

```sql
SELECT Delivery.DeliveryID, DeliveryDriver.Name AS DriverName, Delivery.Status, Delivery.DeliveryDateTime
FROM Delivery
INNER JOIN DeliveryDriver ON Delivery.DriverID = DeliveryDriver.DriverID;
```

DeliveryID	DriverName	Status	DeliveryDateTime
1001	Sam Johnson	Delivered	2023-10-04 12:45:00
1002	Emily Davis	In Transit	NULL
1003	Michael Brown	Delivered	2023-10-04 13:15:00
1004	Lisa Smith	Delivered	2023-10-04 13:30:00
1005	Daniel Wilson	In Transit	NULL

6. Full outer join to retrieve Orders with User Information and Restaurant Details:

SQLite | SQLite.1 | SQLite.3

```sql
SELECT Orders.OrderID, Users.Name AS CustomerName, Restaurant.Name AS RestaurantName, Orders.OrderDateTime, Orders.TotalA
FROM Orders
FULL OUTER JOIN Users ON Orders.UserID = Users.UserID
FULL OUTER JOIN Restaurant ON Orders.RestaurantID = Restaurant.RestaurantID;
```

OrderID	CustomerName	RestaurantName	OrderDateTime	TotalAmount
10001	John Doe	Pizza Palace	2023-10-04 12:00:00	27.98
10002	Jane Smith	Burger Barn	2023-10-04 12:30:00	16.98
10003	Michael Johnson	Sushi House	2023-10-04 13:00:00	39.98
10004	Emily Davis	Taco Town	2023-10-04 13:30:00	13.98
10005	Sarah Wilson	Curry Corner	2023-10-04 14:00:00	25.98

7. Left join to retrieve Menu Items with Restaurant Names:

```sql
SELECT Menu.ItemName, Restaurant.Name AS RestaurantName, Menu.Price
FROM Menu
LEFT JOIN Restaurant ON Menu.RestaurantID = Restaurant.RestaurantID;
```

ItemName	RestaurantName	Price
Margherita Pizza	Pizza Palace	12.99
Pepperoni Pizza	Pizza Palace	14.99
Bacon Cheeseburger	Burger Barn	8.99
Veggie Burger	Burger Barn	7.99
Sashimi Platter	Sushi House	22.99

- 7. PL/SQL

PL/SQL Block to Calculate Total Orders for a User:

Script Output | Query Result

SQL | All Rows Fetched: 5 in 0.023 seconds

Reports

	ORDERID	USERID	RESTAURANTID	ORDERDATETIME	TOTALAMOUNT
1	10001	1	101	2023-10-04 12:00:00	27.98
2	10002	2	102	2023-10-04 12:30:00	16.98
3	10003	3	103	2023-10-04 13:00:00	39.98
4	10004	4	104	2023-10-04 13:30:00	13.98
5	10005	5	105	2023-10-04 14:00:00	25.98

Worksheet | Query Builder

```sql
DECLARE

    total_orders NUMBER;
BEGIN
    SELECT COUNT(*) INTO total_orders
    FROM Orders;
    dbms_output.put_line('Total orders are' || ': ' || total_orders);
END;
```

Script Output

Task completed in 0.07 seconds

```
Total orders are: 5


PL/SQL procedure successfully completed.
```

PL/SQL Block to Update Order Status:

sairam

Worksheet | Query Builder

```
DECLARE
    order_id NUMBER := 10002;
BEGIN
    UPDATE Orders
    SET OrderDateTime = '2023-10-04 13:00:00'
    WHERE OrderID = order_id;

    DBMS_OUTPUT.PUT_LINE('Order ' || order_id || ' status updated.');
END;
```

Query Result | Script Output

Task completed in 0.053 seconds

```
Order 10002 status updated.


PL/SQL procedure successfully completed.
```

Worksheet | Query Builder

```
DECLARE
    new_restaurant_id NUMBER := 419;
    new_restaurant_name VARCHAR2(255) := 'Sairam parlour';
    new_cuisine_type VARCHAR2(255) := 'Thai_Cuisine';
    new_location VARCHAR2(255) := '419 gitam road';
    new_contact_info VARCHAR2(255) := 'info@Sairam_parlour.com';
BEGIN
    INSERT INTO Restaurant (RestaurantID, Name, CuisineType, Location, ContactInfo)
    VALUES (new_restaurant_id, new_restaurant_name, new_cuisine_type, new_location, new_contact_info);

    DBMS_OUTPUT.PUT_LINE('New restaurant ' || new_restaurant_name || ' added with ID ' || new_restaurant_id);
END;
```

```
PL/SQL procedure successfully completed.

New restaurant Sairam parlour added with ID 419


PL/SQL procedure successfully completed.
```

Worksheet | Query Builder

```
DECLARE
    v_delivery_status VARCHAR(30);
BEGIN
    -- Assuming you have a delivery status stored in the variable v_delivery_status
    v_delivery_status := 'Delivered';

    IF v_delivery_status = 'Delivered' THEN
        DBMS_OUTPUT.PUT_LINE('The delivery has been completed.');
    ELSIF v_delivery_status = 'In Transit' THEN
        DBMS_OUTPUT.PUT_LINE('The delivery is currently in transit.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The status of the delivery is unknown.');
    END IF;
END;
```

Script Output

Task completed in 0.14 seconds

```
The delivery has been completed.


PL/SQL procedure successfully completed.
```

Worksheet | Query Builder

```
DECLARE
    v_restaurant_id INT := 101; -- Change this to the desired restaurant ID
    v_total_amount DECIMAL(10, 2);
BEGIN
    SELECT SUM(TotalAmount)
    INTO v_total_amount
    FROM Orders
    WHERE RestaurantID = v_restaurant_id;

    DBMS_OUTPUT.PUT_LINE('Total amount for restaurant ' || v_restaurant_id || ': $' || v_total_amount);
END;
```

```
Script Output
Task completed in 0.097 seconds
Total amount for restaurant 101: $27.98


PL/SQL procedure successfully completed.
```

```
Worksheet   Query Builder
DECLARE
   v_order_id INT;
BEGIN
   FOR orders_cursor IN (SELECT OrderID FROM Orders) LOOP
      v_order_id := orders_cursor.OrderID;
      DBMS_OUTPUT.PUT_LINE('Order ID: ' || v_order_id);
   END LOOP;
END;
```

```
Script Output
Task completed in 0.075 seconds
Order ID: 10001
Order ID: 10002
Order ID: 10003
Order ID: 10004
Order ID: 10005


PL/SQL procedure successfully completed.
```

PL/SQL block to create a new table and insert data

- 8.Triggers

Trigger to Prevent Delivery Updates After Delivery:

Worksheet Query Builder

```
CREATE OR REPLACE TRIGGER PreventDeliveryUpdates
BEFORE UPDATE ON Delivery
FOR EACH ROW
BEGIN
    IF :OLD.Status = 'Delivered' THEN
        RAISE_APPLICATION_ERROR(-20003, 'Cannot update delivered orders.');
    END IF;
END;
```

This trigger prevents updates to the `Delivery` table for orders that have already been marked as 'Delivered'.

Worksheet Query Builder

```
CREATE OR REPLACE TRIGGER PreventDeliveryUpdates
BEFORE UPDATE ON Delivery
FOR EACH ROW
BEGIN
    IF :OLD.Status = 'Delivered' THEN
        RAISE_APPLICATION_ERROR(-20003, 'Cannot update delivered orders.');
    END IF;
END;
```

This trigger sets the `OrderDateTime` column to the current date and time when a new order is inserted into the `Orders` table.

Worksheet Query Builder

```
CREATE OR REPLACE TRIGGER SetOrderDateTime
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
    :NEW.OrderDateTime := TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS');
END;
/
```

Worksheet Query Builder

```
CREATE OR REPLACE TRIGGER PreventZeroTotalAmount
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
    IF :NEW.TotalAmount <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'TotalAmount must be greater than zero.');
    END IF;
END;
```

https://chat.openai.com/share/7f4cb627-5e73-4ead-a660-833983aa96b4

3)Correlated Nested Queries:

1)Customers Who Placed Orders and Total amount Spent:

```
SELECT u.Name AS CustomerName, u.Email, SUM(o.TotalAmount) AS TotalAmountSpent
FROM Users u
JOIN Orders o ON u.UserID = o.UserID
GROUP BY u.Name, u.Email;
```

CUSTOMERNAME	EMAIL	TOTALAMOUNTSPENT
John Doe	john@example.com	27.98
Jane Smith	jane@example.com	16.98
Emily Davis	emily@example.com	13.98
Sarah Wilson	sarah@example.com	25.98
Michael Johnson	michael@example.com	29.98

2)Find Restaurants with Their Total Sales:

```sql
SELECT r.Name AS RestaurantName, SUM(o.TotalAmount) AS TotalSales
FROM Restaurant r
JOIN Orders o ON r.RestaurantID = o.RestaurantID
GROUP BY r.Name;
```

RESTAURANTNAME	TOTALSALES
Burger Barn	16.98
Pizza Palace	27.98
Sushi House	39.98
Curry Corner	25.98

3) Find Customers Who Ordered a Specific Menu Item :

```sql
SELECT DISTINCT u.Name AS CustomerName, u.Email
FROM Users u
JOIN Orders o ON u.UserID = o.UserID
JOIN OrderItems oi ON o.OrderID = oi.OrderID
WHERE oi.MenuID = 1002;
```

CUSTOMERNAME	EMAIL
John Doe	john@example.com

Download CSV

4) Find Orders with Their Associated Delivery Status :

```sql
SELECT o.OrderID, o.OrderDateTime, d.Status AS DeliveryStatus
FROM Orders o
LEFT JOIN Delivery d ON o.OrderID = d.OrderID;
```

10001	2023-10-04 12:00:00	Delivered
10002	2023-10-04 12:30:00	In Transit
10003	2023-10-04 13:00:00	Delivered
10004	2023-10-04 13:30:00	Delivered
10005	2023-10-04 14:00:00	In Transit

5) Find Customers and Their Most Recent Order :

```sql
SELECT u.Name AS CustomerName, o.OrderID, o.OrderDateTime, o.TotalAmount
FROM Users u
JOIN Orders o ON u.UserID = o.UserID
WHERE o.OrderDateTime = (
    SELECT MAX(OrderDateTime)
    FROM Orders
    WHERE UserID = u.UserID
);
```

CUSTOMERNAME	ORDERID	ORDERDATETIME	TOTALAMOUNT
John Doe	10001	2023-10-04 12:00:00	27.98
Jane Smith	10002	2023-10-04 12:30:00	16.98
Michael Johnson	10003	2023-10-04 13:00:00	39.98
Emily Davis	10004	2023-10-04 13:30:00	13.98

6) Find Customers with Multiple Orders :

```sql
SELECT u.Name AS CustomerName, COUNT(o.OrderID) AS OrderCount
FROM Users u
JOIN Orders o ON u.UserID = o.UserID
GROUP BY u.Name
HAVING COUNT(o.OrderID) > 1;
```

no data found

4. Set Comparison Operators

1) List All Menu Items Available at Any Restaurant :

```sql
SELECT ItemName
FROM Menu
UNION
SELECT ItemName
FROM Menu;
```

ITEMNAME
Bacon Cheeseburger
Margherita Pizza
Pepperoni Pizza
Sashimi Platter
Veggie Burger

2) Find Common Menu Items Between Two Restaurants :

```sql
SELECT ItemName
FROM Menu
WHERE RestaurantID = 101
INTERSECT
SELECT ItemName
FROM Menu
WHERE RestaurantID = 102;
```

no data found

3) Find Menu Items Exclusive to One Restaurant :

```sql
SELECT ItemName
FROM Menu
WHERE RestaurantID = 101

MINUS

SELECT ItemName
FROM Menu
WHERE RestaurantID = 102;
```

ITEMNAME
Margherita Pizza
Pepperoni Pizza

Download CSV

4) cartesian product

```sql
SELECT *
FROM Users
CROSS JOIN Restaurant;
```

USERID	NAME	EMAIL	PHONE	ADDRESS	RESTAURANTID	NAME	CUISINETYPE	LOCATION	CONTACTINFO
1	John Doe	john@example.com	123-456-7890	123 Main St	101	Pizza Palace	Italian	789 Elm St	info@pizzapalace.com
1	John Doe	john@example.com	123-456-7890	123 Main St	102	Burger Barn	American	456 Maple Ave	info@burgerbarn.com
1	John Doe	john@example.com	123-456-7890	123 Main St	103	Sushi House	Japanese	123 Sakura Rd	info@sushihouse.com