```sql
-- Report 1: Generate a list of customers who have made a purchase, including their
names and contact information.
-- This query fetches the names, email addresses, and phone numbers of customers
who have placed at least one order.
SELECT DISTINCT c.CustomerName, c.CustomerEmail, c.CustomerPhoneNumber
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID;

-- Report 2: Retrieve a count of orders placed for each customer.
-- This query counts how many orders each customer has made by grouping their
customer ID.
SELECT c.CustomerName, COUNT(o.OrderID) AS TotalOrders
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.CustomerName;

-- Report 3: List the top-selling products based on the total quantity sold.
-- This query determines the total number of units sold for each product type
(BeanType) and orders the result descending by quantity.
SELECT b.BeanType, SUM(o.Quantity) AS TotalQuantitySold
FROM Orders o
JOIN BeanBatches b ON o.BeanBatchID = b.BeanBatchID
GROUP BY b.BeanType
ORDER BY TotalQuantitySold DESC;

-- Report 4: Display the total revenue generated by each product category.
-- This query calculates total revenue by summing up order amounts grouped by the
supplier category.
SELECT s.SupplierCategory, SUM(o.OrderAmount) AS TotalRevenue
FROM Orders o
JOIN BeanBatches b ON o.BeanBatchID = b.BeanBatchID
JOIN Suppliers s ON b.SupplierID = s.SupplierID
GROUP BY s.SupplierCategory;

-- Report 5: Find the customers who have made purchases within the last 30 days.
-- This query uses a subquery to find customers who made purchases within the 30-
day period from the current date.
SELECT DISTINCT c.CustomerID, c.CustomerName, c.CustomerEmail,
c.CustomerPhoneNumber
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderDate >= (SELECT CURDATE() - INTERVAL 30 DAY);

-- Report 6: Calculate the average price of products in each category.
-- This query groups data by supplier category and calculates the average purchase
price for each category.
SELECT s.SupplierCategory, AVG(b.PurchasePrice) AS AveragePrice
FROM BeanBatches b
JOIN Suppliers s ON b.SupplierID = s.SupplierID
GROUP BY s.SupplierCategory;

-- Report 7: Determine the total revenue generated by each customer.
-- This query sums up all order amounts placed by each customer and groups them by
customer ID.
SELECT c.CustomerName, SUM(o.OrderAmount) AS TotalRevenue
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.CustomerName;
```

```sql
-- Report 8: Retrieve the order details (product, quantity, price) for a specific
customer.
-- This query fetches order details by joining `Orders` with `BeanBatches` for a
given specific customer.
SELECT o.OrderID, b.BeanType, o.Quantity, o.OrderAmount
FROM Orders o
JOIN BeanBatches b ON o.BeanBatchID = b.BeanBatchID
WHERE o.CustomerID = 1; -- Replace '1' with the specific customer's ID

-- Report 9: Identify customers who have not made any purchases.
-- This query uses a LEFT JOIN to find customers without corresponding orders in
the `Orders` table.
SELECT c.CustomerID, c.CustomerName, c.CustomerEmail
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderID IS NULL;

-- Report 10: Retrieve the orders that include products with a price higher than
the average price.
-- This query uses a subquery to compute the average purchase price and filters for
orders containing products with prices above this value.
SELECT o.OrderID, b.BeanType, o.OrderAmount, b.PurchasePrice
FROM Orders o
JOIN BeanBatches b ON o.BeanBatchID = b.BeanBatchID
WHERE b.PurchasePrice > (SELECT AVG(PurchasePrice) FROM BeanBatches);
```