Trường Đại Học Quốc Tế - ĐHQG TP.HCM

LAB REPORT

Course: Principles Of Database Management 8

Full Name: Trần Minh Phúc . . . .

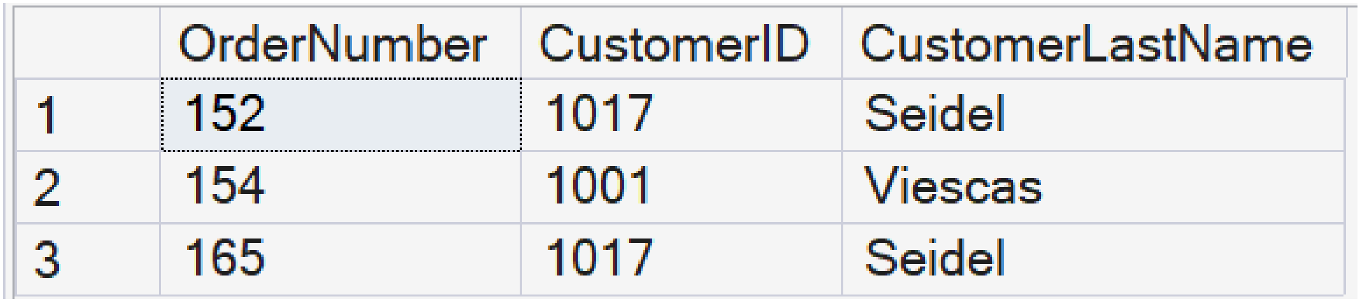
Student's ID: ITCSIU24070 . . . .

1. **Query**: Show me all the orders shipped on October 3, 2017, and each order's related customer last name.

   SQL:

```
SELECT
  OrderNumber,
  CustomerID,
  (SELECT CustLastName
    FROM Customers
    WHERE CustomerID = o.CustomerID
  ) AS CustomerLastName
FROM Orders AS o
WHERE ShipDate = '2017-10-3';
GO
```

   Result set:



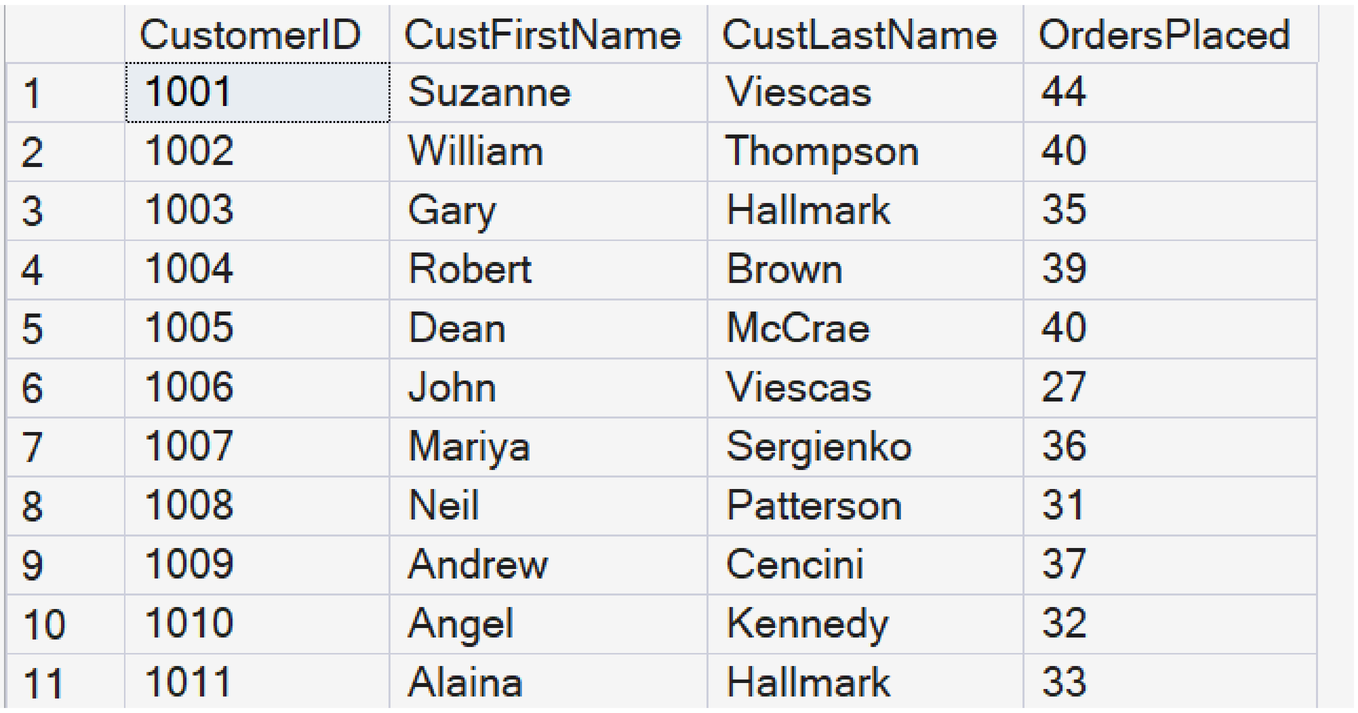
	OrderNumber	CustomerID	CustomerLastName
1	152	1017	Seidel
2	154	1001	Viescas
3	165	1017	Seidel

2. **Query**: List all the customer names and a count of the orders they placed.

   SQL:

```
SELECT
  CustomerID,
  CustFirstName,
  CustLastName,
  (SELECT
    COUNT(OrderNumber)
    FROM Orders
    WHERE c.CustomerID = CustomerID
    GROUP BY CustomerID
  ) AS OrdersPlaced
FROM Customers AS c;
GO
```

Result set:



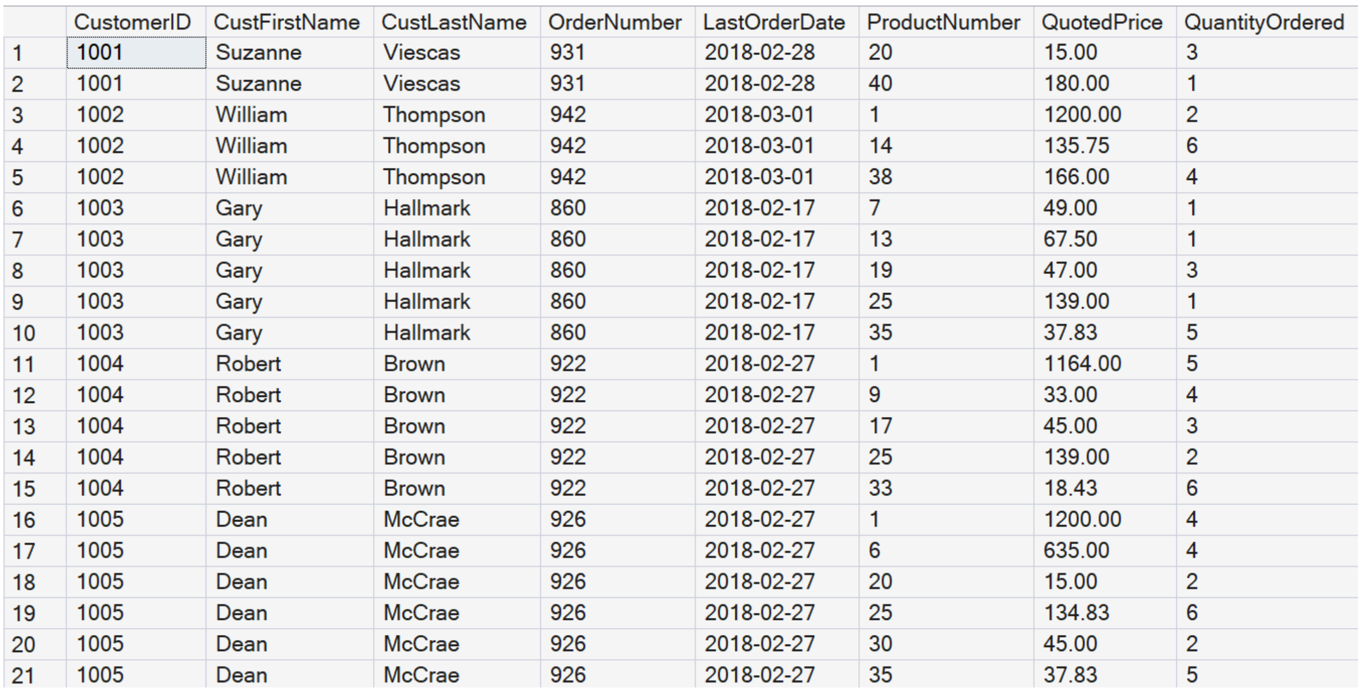
	CustomerID	CustFirstName	CustLastName	OrdersPlaced
1	1001	Suzanne	Viescas	44
2	1002	William	Thompson	40
3	1003	Gary	Hallmark	35
4	1004	Robert	Brown	39
5	1005	Dean	McCrae	40
6	1006	John	Viescas	27
7	1007	Mariya	Sergienko	36
8	1008	Neil	Patterson	31
9	1009	Andrew	Cencini	37
10	1010	Angel	Kennedy	32
11	1011	Alaina	Hallmark	33

3. **Query**: List customers and all the details from their last order.

   SQL:

```sql
SELECT
  c.CustomerID,
  c.CustFirstName,
  c.CustLastName,
  lo.OrderNumber,
  lo.OrderDate AS LastOrderDate,
  d.ProductNumber,
  d.QuotedPrice,
  d.QuantityOrdered
FROM Customers AS c
LEFT JOIN (
  SELECT o.CustomerID, o.OrderNumber, o.OrderDate
  FROM Orders o
  WHERE o.OrderDate = (
    SELECT MAX(OrderDate)
    FROM Orders o2
    WHERE o2.CustomerID = o.CustomerID
  )
) AS lo
ON lo.CustomerID = c.CustomerID
LEFT JOIN Order_Details AS d
ON d.OrderNumber = lo.OrderNumber;
GO
```

Result set:



	CustomerID	CustFirstName	CustLastName	OrderNumber	LastOrderDate	ProductNumber	QuotedPrice	QuantityOrdered
1	1001	Suzanne	Viescas	931	2018-02-28	20	15.00	3
2	1001	Suzanne	Viescas	931	2018-02-28	40	180.00	1
3	1002	William	Thompson	942	2018-03-01	1	1200.00	2
4	1002	William	Thompson	942	2018-03-01	14	135.75	6
5	1002	William	Thompson	942	2018-03-01	38	166.00	4
6	1003	Gary	Hallmark	860	2018-02-17	7	49.00	1
7	1003	Gary	Hallmark	860	2018-02-17	13	67.50	1
8	1003	Gary	Hallmark	860	2018-02-17	19	47.00	3
9	1003	Gary	Hallmark	860	2018-02-17	25	139.00	1
10	1003	Gary	Hallmark	860	2018-02-17	35	37.83	5
11	1004	Robert	Brown	922	2018-02-27	1	1164.00	5
12	1004	Robert	Brown	922	2018-02-27	9	33.00	4
13	1004	Robert	Brown	922	2018-02-27	17	45.00	3
14	1004	Robert	Brown	922	2018-02-27	25	139.00	2
15	1004	Robert	Brown	922	2018-02-27	33	18.43	6
16	1005	Dean	McCrae	926	2018-02-27	1	1200.00	4
17	1005	Dean	McCrae	926	2018-02-27	6	635.00	4
18	1005	Dean	McCrae	926	2018-02-27	20	15.00	2
19	1005	Dean	McCrae	926	2018-02-27	25	134.83	6
20	1005	Dean	McCrae	926	2018-02-27	30	45.00	2
21	1005	Dean	McCrae	926	2018-02-27	35	37.83	5

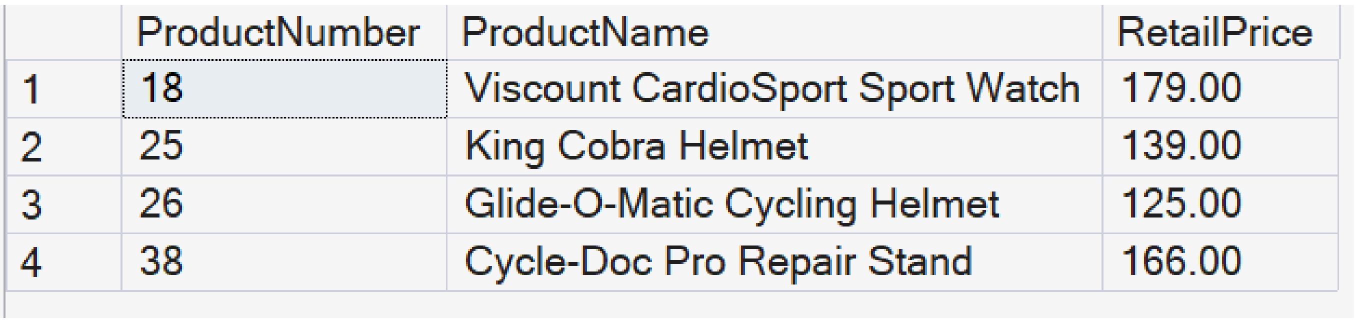
4. **Query**: Find all accessories that are priced greater than any clothing item. (hint use ALL)

   SQL:

```
SELECT
  ProductNumber,
  ProductName,
  RetailPrice
FROM Products
WHERE CategoryID IN (
  SELECT CategoryID
  FROM Categories
  WHERE CategoryDescription = 'Accessories'
)
AND RetailPrice >  ALL(
  SELECT
  RetailPrice
```

```sql
  FROM Products
  WHERE CategoryID IN (
    SELECT CategoryID
    FROM Categories
    WHERE CategoryDescription = 'Clothing'
  )
);
GO
```

Result set:



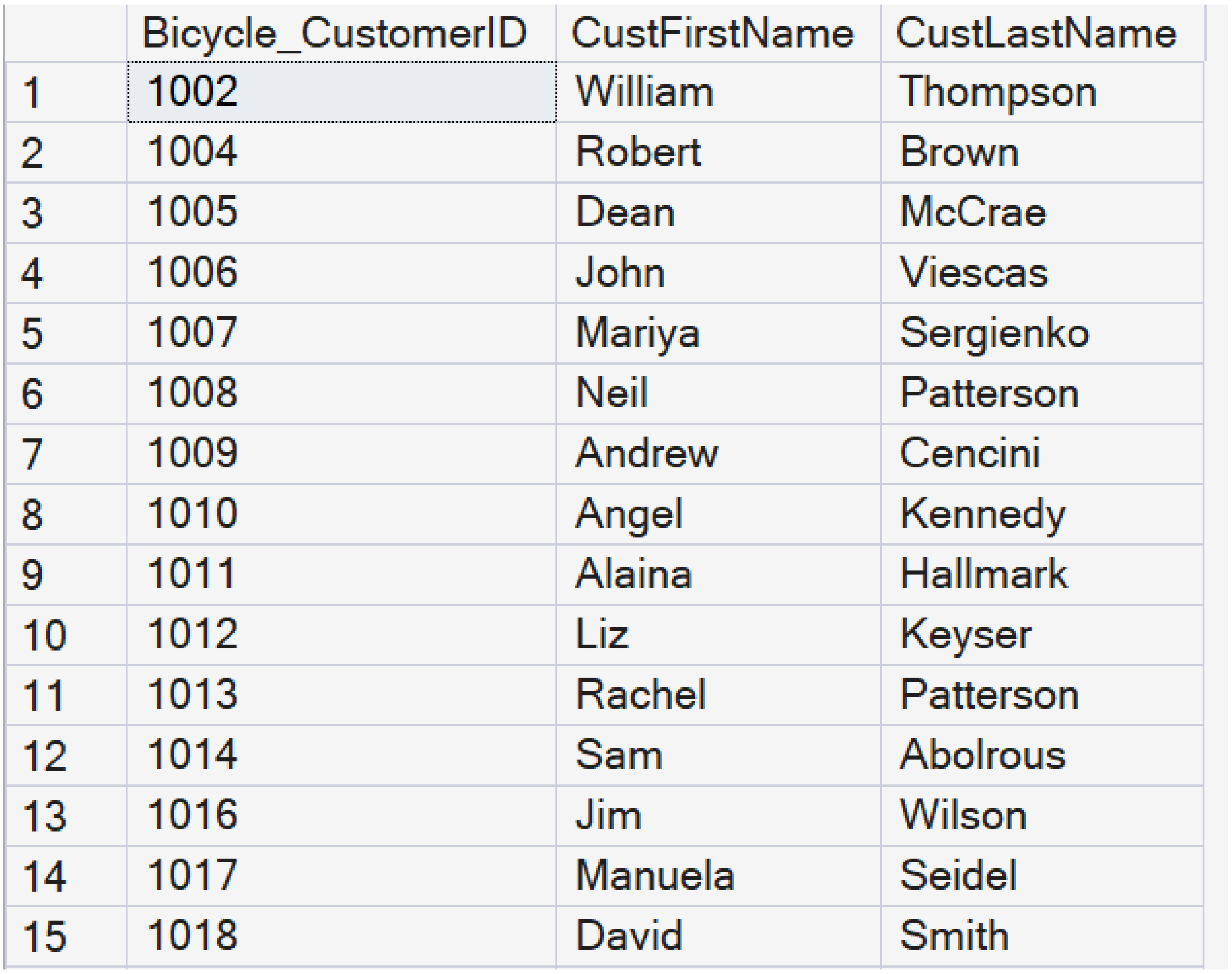
	ProductNumber	ProductName	RetailPrice
1	18	Viscount CardioSport Sport Watch	179.00
2	25	King Cobra Helmet	139.00
3	26	Glide-O-Matic Cycling Helmet	125.00
4	38	Cycle-Doc Pro Repair Stand	166.00

5. **Query**: Find all the customers who ordered a bicycle. (Use EXISTS)

   SQL:

```sql
SELECT
  c.CustomerID AS Bicycle_CustomerID,
  c.CustFirstName,
  c.CustLastName
FROM Customers AS c
WHERE EXISTS (
  SELECT 1
  FROM Orders AS o
  WHERE c.CustomerID = o.CustomerID
  AND EXISTS (
    SELECT 1
    FROM Order_Details AS od
    WHERE o.OrderNumber = od.OrderNumber
    AND EXISTS (
      SELECT 1
      FROM Products AS p
      WHERE od.ProductNumber = p.ProductNumber
      AND EXISTS (
        SELECT 1
        FROM Categories AS cat
        WHERE p.CategoryID = cat.CategoryID
        AND CategoryDescription = 'Bikes'
      )
    )
  )
);
GO
```

Result set:



	Bicycle_CustomerID	CustFirstName	CustLastName
1	1002	William	Thompson
2	1004	Robert	Brown
3	1005	Dean	McCrae
4	1006	John	Viescas
5	1007	Mariya	Sergienko
6	1008	Neil	Patterson
7	1009	Andrew	Cencini
8	1010	Angel	Kennedy
9	1011	Alaina	Hallmark
10	1012	Liz	Keyser
11	1013	Rachel	Patterson
12	1014	Sam	Abolrous
13	1016	Jim	Wilson
14	1017	Manuela	Seidel
15	1018	David	Smith

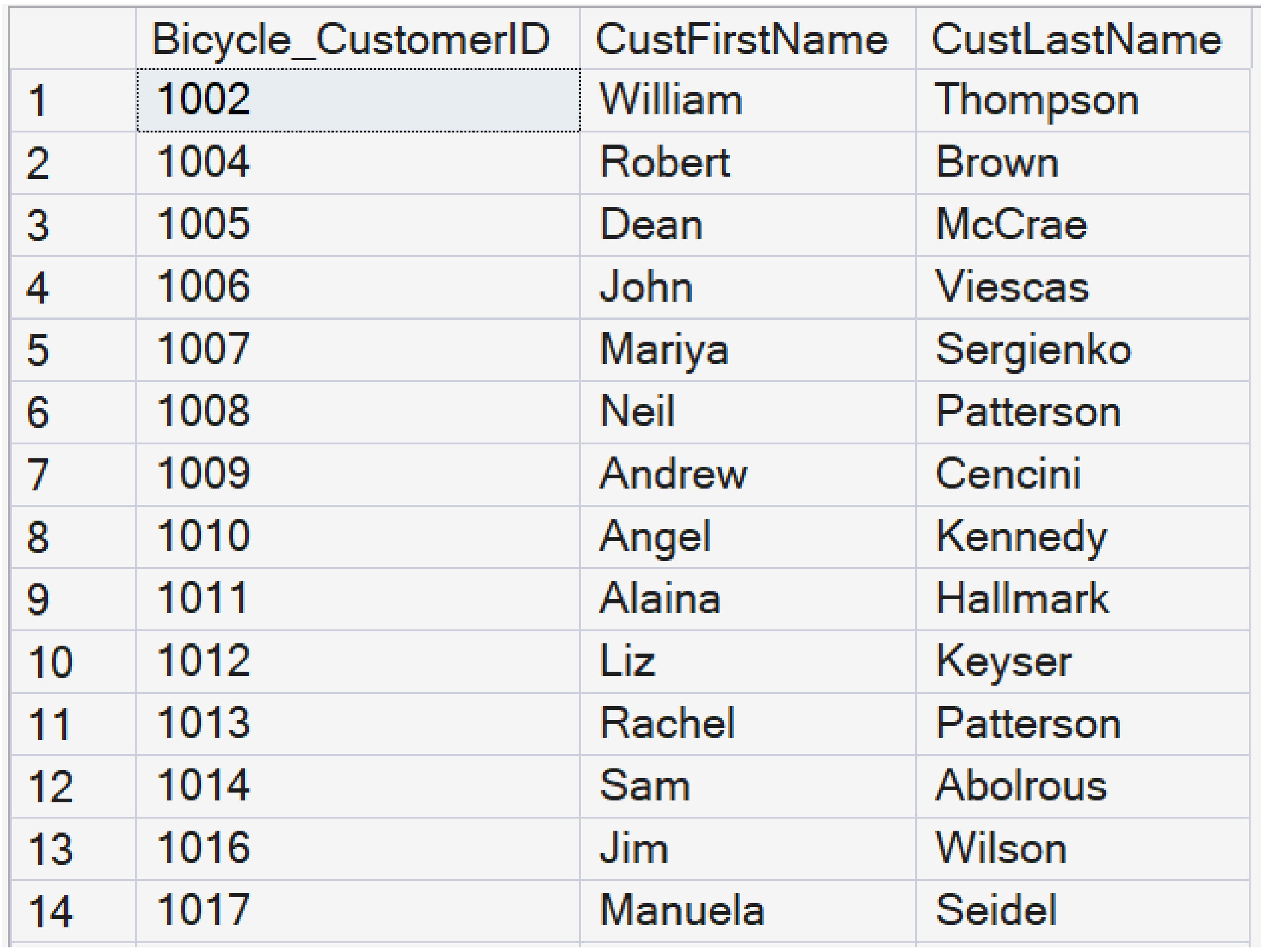
6. **Query**: List customers who ordered bikes.(use IN)

   SQL:

```
SELECT
```

```sql
  CustomerID AS Bicycle_CustomerID,
  CustFirstName,
  CustLastName
FROM Customers
WHERE CustomerID IN (
  SELECT CustomerID
  FROM Orders
  WHERE OrderNumber IN (
    SELECT OrderNumber
    FROM Order_Details
    WHERE ProductNumber IN (
      SELECT ProductNumber
      FROM Products
      WHERE CategoryID IN (
        SELECT CategoryID
        FROM Categories AS cat
        WHERE CategoryDescription = 'Bikes'
      )
    )
  )
);
GO
```

Result set:



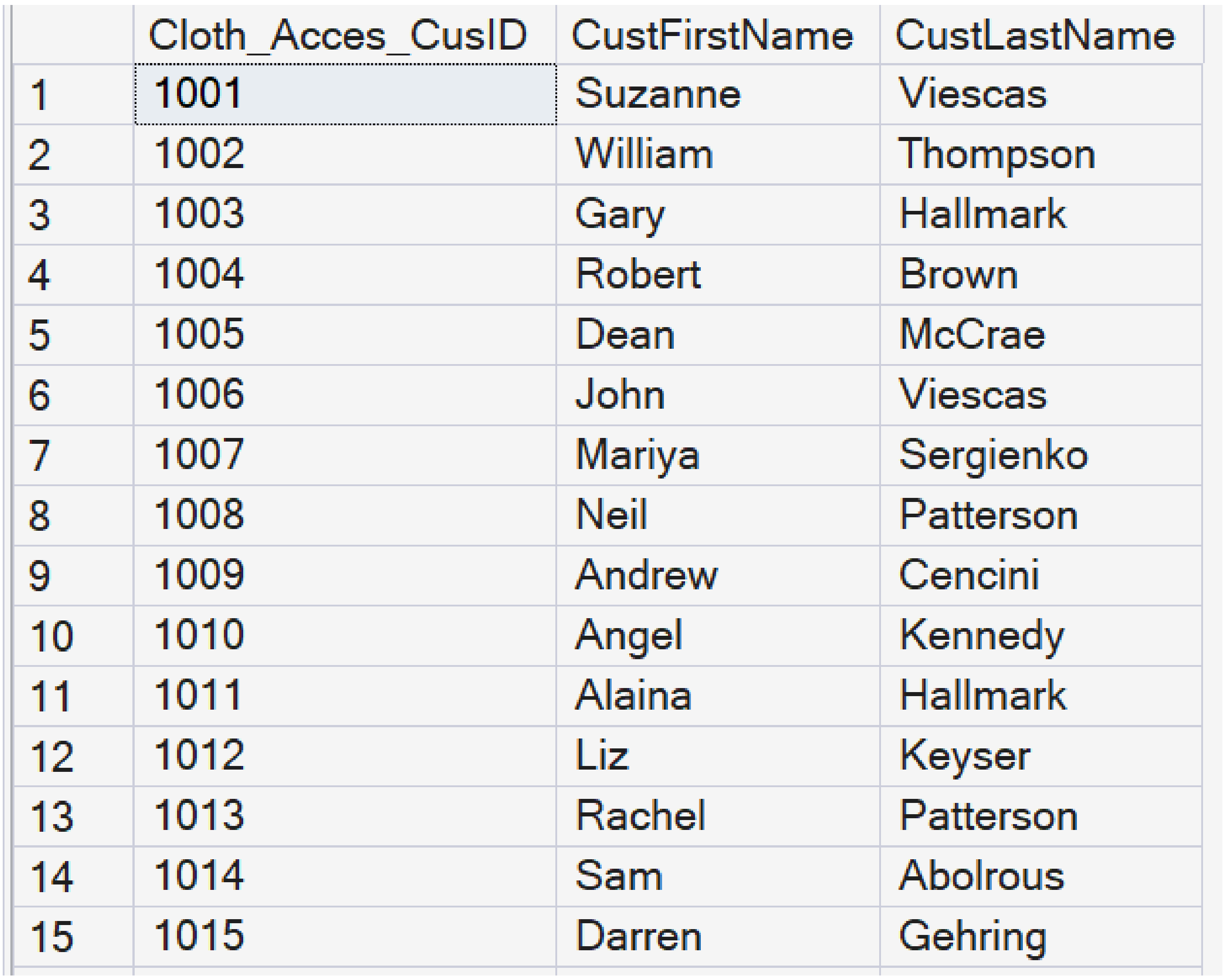
	Bicycle_CustomerID	CustFirstName	CustLastName
1	1002	William	Thompson
2	1004	Robert	Brown
3	1005	Dean	McCrae
4	1006	John	Viescas
5	1007	Mariya	Sergienko
6	1008	Neil	Patterson
7	1009	Andrew	Cencini
8	1010	Angel	Kennedy
9	1011	Alaina	Hallmark
10	1012	Liz	Keyser
11	1013	Rachel	Patterson
12	1014	Sam	Abolrous
13	1016	Jim	Wilson
14	1017	Manuela	Seidel

7. **Query**: Display customers who ordered clothing or accessories.

   SQL:

```
SELECT
  CustomerID AS Cloth_Acces_CusID,
  CustFirstName,
  CustLastName
FROM Customers
WHERE CustomerID = SOME (
  SELECT CustomerID
  FROM Orders
  WHERE OrderNumber = SOME (
    SELECT OrderNumber
    FROM Order_Details
    WHERE ProductNumber = SOME (
      SELECT ProductNumber
      FROM Products
      WHERE CategoryID = SOME (
        SELECT CategoryID
        FROM Categories
        WHERE CategoryDescription = 'Clothing'
        OR CategoryDescription = 'Accessories'
      )
    )
  )
);
GO
```

Result set:



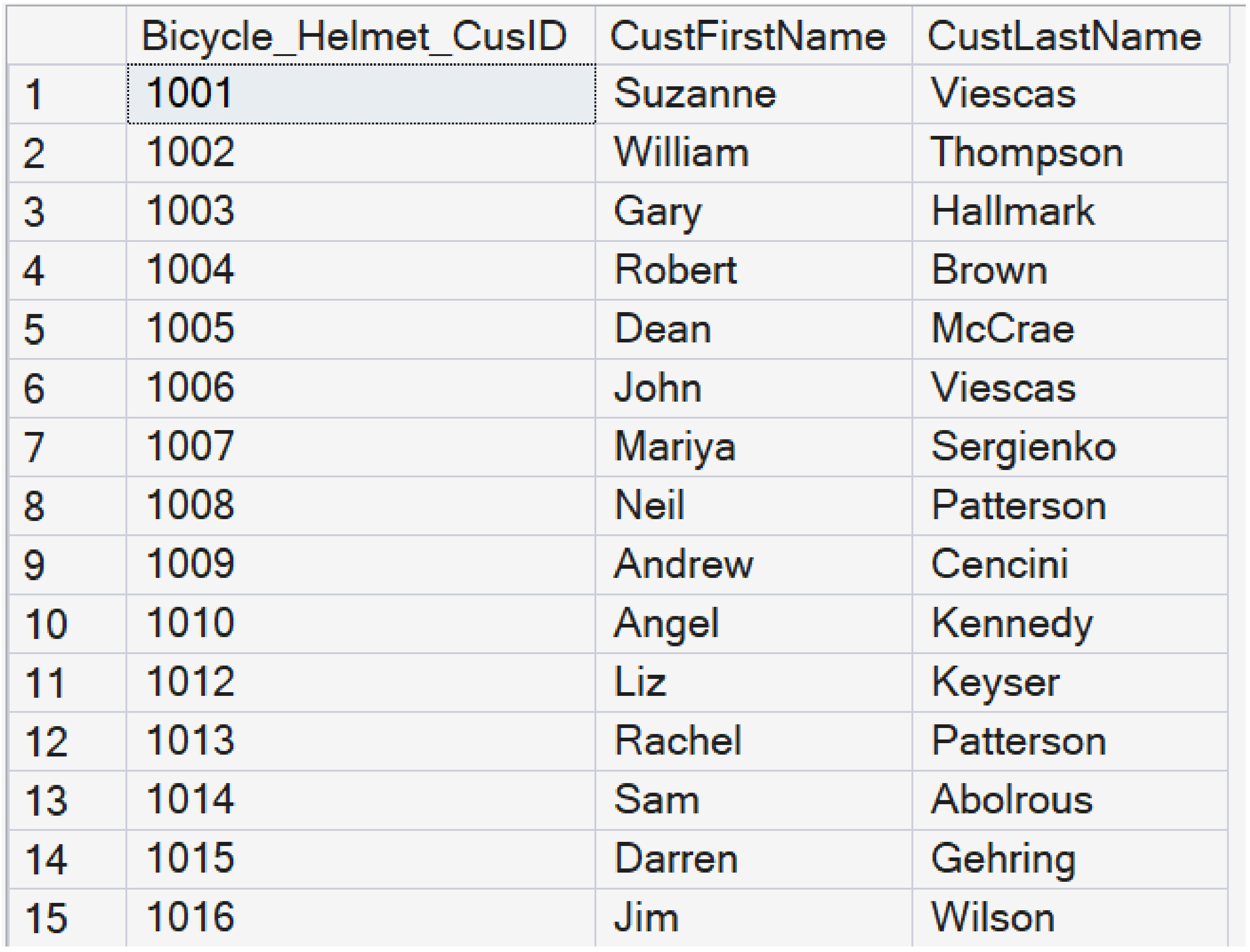
	Cloth_Acces_CusID	CustFirstName	CustLastName
1	1001	Suzanne	Viescas
2	1002	William	Thompson
3	1003	Gary	Hallmark
4	1004	Robert	Brown
5	1005	Dean	McCrae
6	1006	John	Viescas
7	1007	Mariya	Sergienko
8	1008	Neil	Patterson
9	1009	Andrew	Cencini
10	1010	Angel	Kennedy
11	1011	Alaina	Hallmark
12	1012	Liz	Keyser
13	1013	Rachel	Patterson
14	1014	Sam	Abolrous
15	1015	Darren	Gehring

8. **Query**: Find all the customers who ordered a bicycle helmet.

SQL:

```
SELECT
  CustomerID AS Bicycle_Helmet_CusID,
  CustFirstName,
  CustLastName
FROM Customers
WHERE CustomerID IN (
  SELECT CustomerID
  FROM Orders
  WHERE OrderNumber IN (
    SELECT OrderNumber
    FROM Order_Details
    WHERE ProductNumber IN (
      SELECT ProductNumber
      FROM Products
      WHERE ProductName LIKE '%Helmet%'
    )
  )
);
GO
```

Result set:



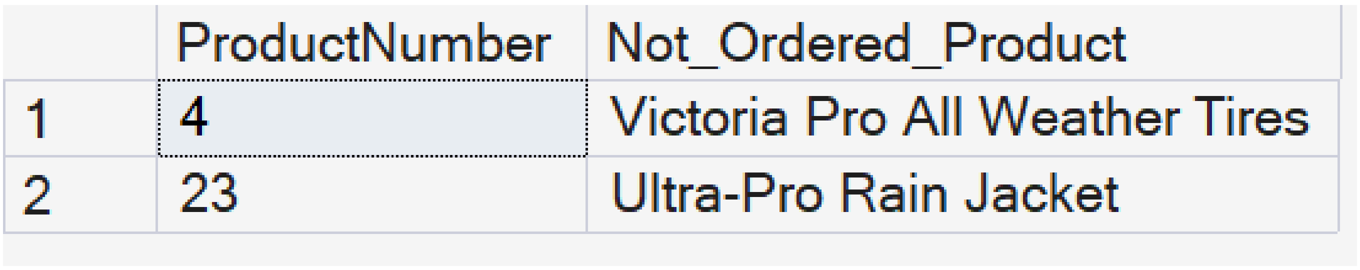
	Bicycle_Helmet_CusID	CustFirstName	CustLastName
1	1001	Suzanne	Viescas
2	1002	William	Thompson
3	1003	Gary	Hallmark
4	1004	Robert	Brown
5	1005	Dean	McCrae
6	1006	John	Viescas
7	1007	Mariya	Sergienko
8	1008	Neil	Patterson
9	1009	Andrew	Cencini
10	1010	Angel	Kennedy
11	1012	Liz	Keyser
12	1013	Rachel	Patterson
13	1014	Sam	Abolrous
14	1015	Darren	Gehring
15	1016	Jim	Wilson

9. **Query**: What products have never been ordered?

   SQL:

```
SELECT
  ProductNumber,
  ProductName AS Not_Ordered_Product
FROM Products
WHERE ProductNumber NOT IN (
  SELECT ProductNumber
  FROM Order_Details
);
GO
```

   Result set:



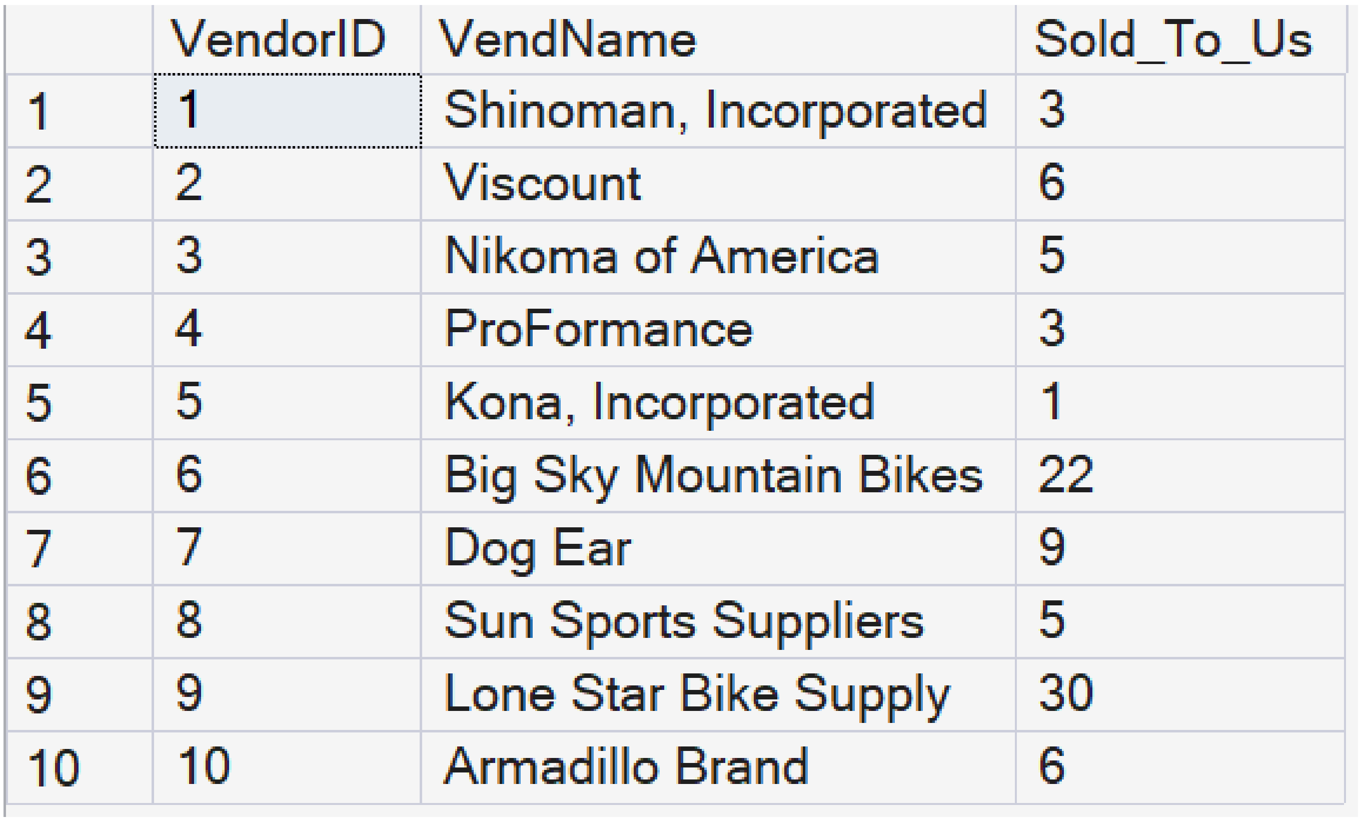
	ProductNumber	Not_Ordered_Product
1	4	Victoria Pro All Weather Tires
2	23	Ultra-Pro Rain Jacket

10. **Query**: List vendors and a count of the products they sell to us.

    SQL:

```sql
SELECT
  VendorID,
  (SELECT VendName
    FROM Vendors
    WHERE Vendors.VendorID = Product_Vendors.VendorID
  ) AS VendName,
  COUNT(ProductNumber) AS Sold_To_Us
FROM Product_Vendors
GROUP BY VendorID;
GO
```

Result set:



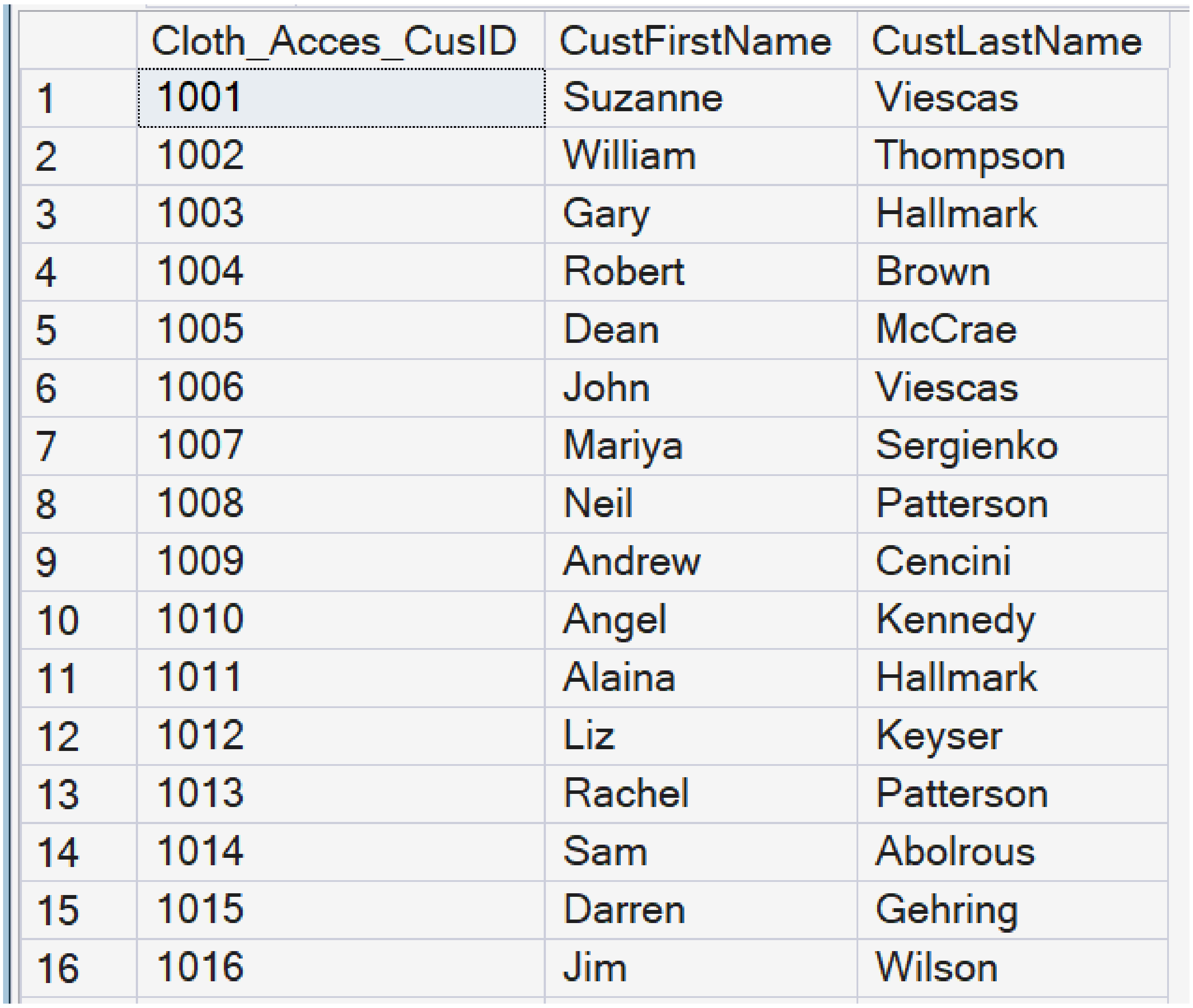
	VendorID	VendName	Sold_To_Us
1	1	Shinoman, Incorporated	3
2	2	Viscount	6
3	3	Nikoma of America	5
4	4	ProFormance	3
5	5	Kona, Incorporated	1
6	6	Big Sky Mountain Bikes	22
7	7	Dog Ear	9
8	8	Sun Sports Suppliers	5
9	9	Lone Star Bike Supply	30
10	10	Armadillo Brand	6

11. **Query**: Display customers who ordered clothing or accessories

    SQL:

```sql
SELECT
  CustomerID AS Cloth_Acces_CusID,
  CustFirstName,
  CustLastName
FROM Customers
WHERE CustomerID = ANY (
  SELECT CustomerID
  FROM Orders
  WHERE OrderNumber = ANY (
    SELECT OrderNumber
    FROM Order_Details
    WHERE ProductNumber = ANY (
      SELECT ProductNumber
      FROM Products
      WHERE CategoryID = ANY (
        SELECT CategoryID
        FROM Categories
        WHERE CategoryDescription = 'Clothing'
        OR CategoryDescription = 'Accessories'
      )
    )
  )
);
GO
```

    Result set:



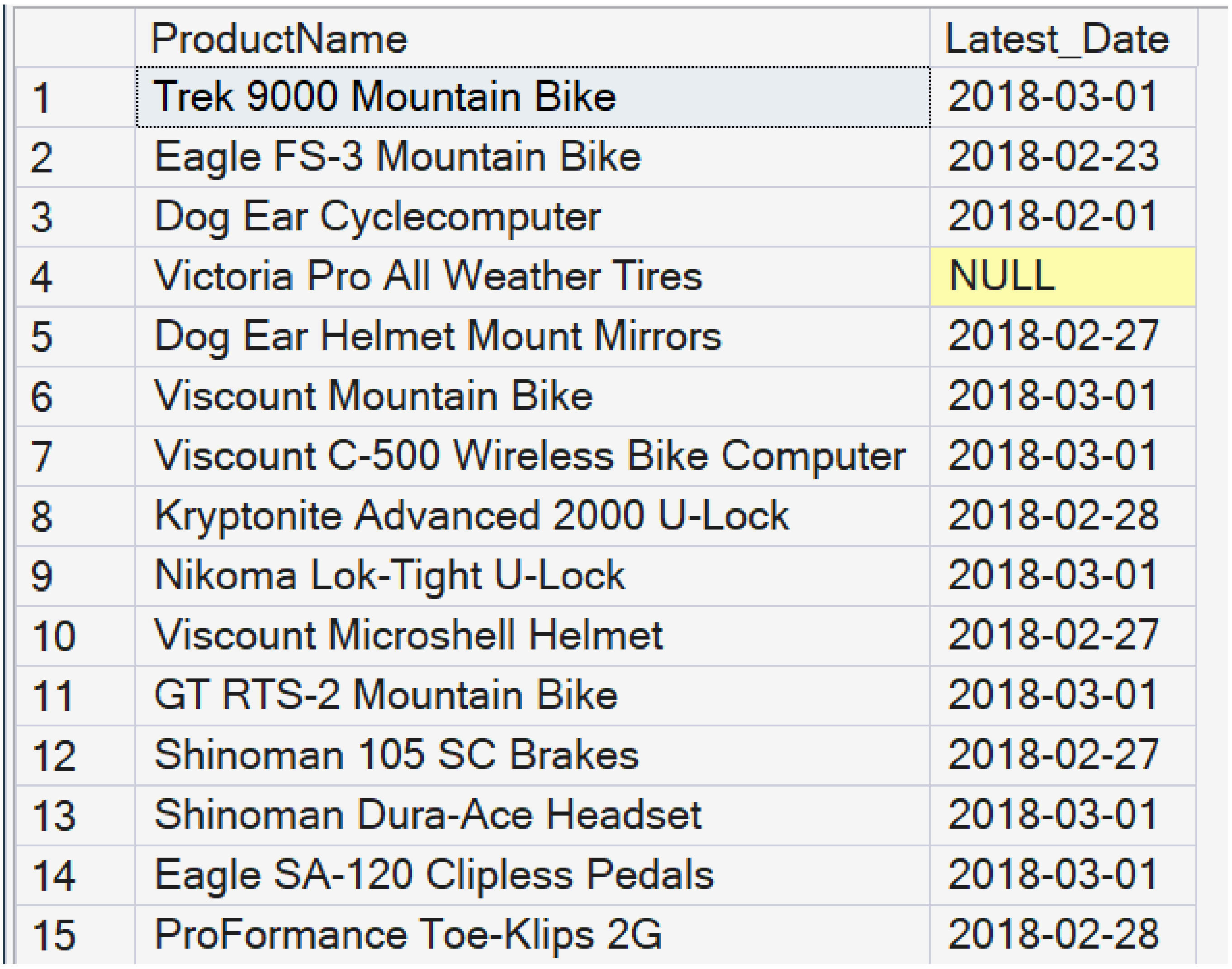
	Cloth_Acces_CusID	CustFirstName	CustLastName
1	1001	Suzanne	Viescas
2	1002	William	Thompson
3	1003	Gary	Hallmark
4	1004	Robert	Brown
5	1005	Dean	McCrae
6	1006	John	Viescas
7	1007	Mariya	Sergienko
8	1008	Neil	Patterson
9	1009	Andrew	Cencini
10	1010	Angel	Kennedy
11	1011	Alaina	Hallmark
12	1012	Liz	Keyser
13	1013	Rachel	Patterson
14	1014	Sam	Abolrous
15	1015	Darren	Gehring
16	1016	Jim	Wilson

12. **Query**: Display products and the latest date each product was ordered. (Hint: Use the MAX aggregate function.) (40 rows).

SQL:

```sql
SELECT TOP 40
  ProductName,
  (SELECT MAX(OrderDate)
    FROM Orders AS o
    WHERE o.OrderNumber IN (
      SELECT OrderNumber
      FROM Order_Details AS od
      WHERE od.ProductNumber = p.ProductNumber
    )
  ) AS Latest_Date
FROM Products AS p;
GO
```

Result set:



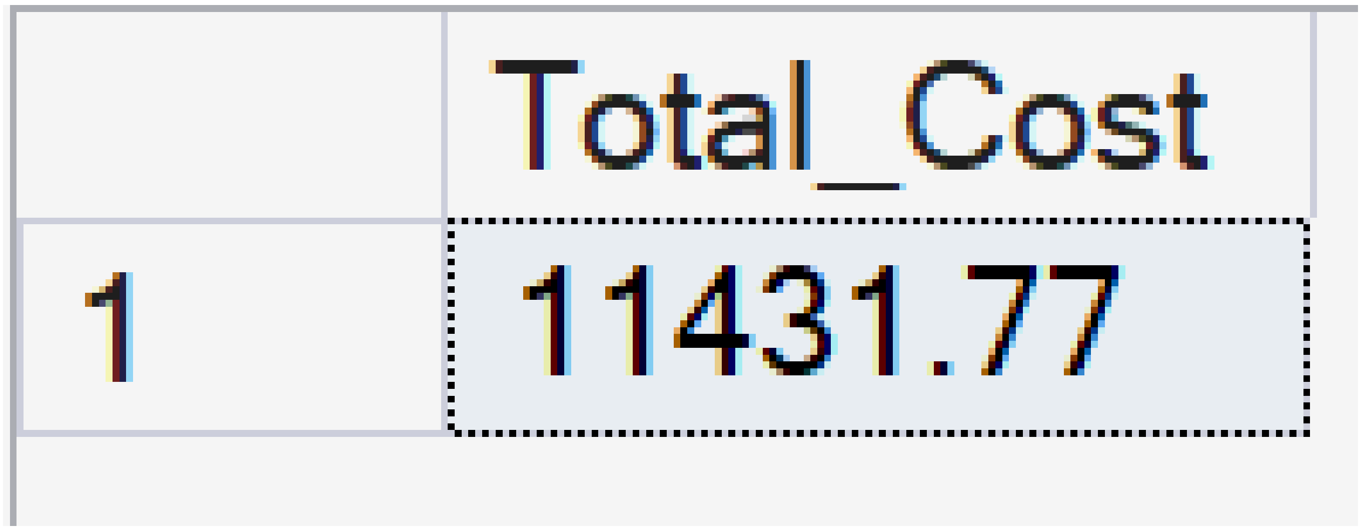
	ProductName	Latest_Date
1	Trek 9000 Mountain Bike	2018-03-01
2	Eagle FS-3 Mountain Bike	2018-02-23
3	Dog Ear Cyclecomputer	2018-02-01
4	Victoria Pro All Weather Tires	NULL
5	Dog Ear Helmet Mount Mirrors	2018-02-27
6	Viscount Mountain Bike	2018-03-01
7	Viscount C-500 Wireless Bike Computer	2018-03-01
8	Kryptonite Advanced 2000 U-Lock	2018-02-28
9	Nikoma Lok-Tight U-Lock	2018-03-01
10	Viscount Microshell Helmet	2018-02-27
11	GT RTS-2 Mountain Bike	2018-03-01
12	Shinoman 105 SC Brakes	2018-02-27
13	Shinoman Dura-Ace Headset	2018-03-01
14	Eagle SA-120 Clipless Pedals	2018-03-01
15	ProFormance Toe-Klips 2G	2018-02-28

13. **Query**: Calculate a total of all unique wholesale costs for the products we sell. (use SUM)

SQL:

```
SELECT SUM(DISTINCT WholesalePrice) AS Total_Cost
FROM Product_Vendors;
GO
```

Result set:



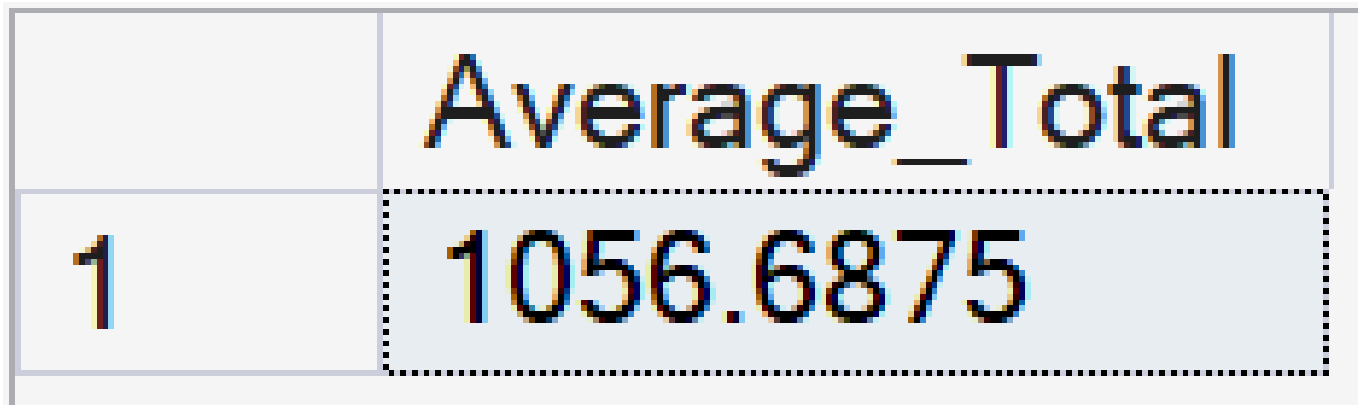
	Total_Cost
1	11431.77

14. **Query**: What is the average item total for order 64?

    SQL:

```sql
SELECT AVG(QuotedPrice * QuantityOrdered) AS Average_Total
FROM Order_Details
WHERE OrderNumber = 64;
GO
```

    Result set:



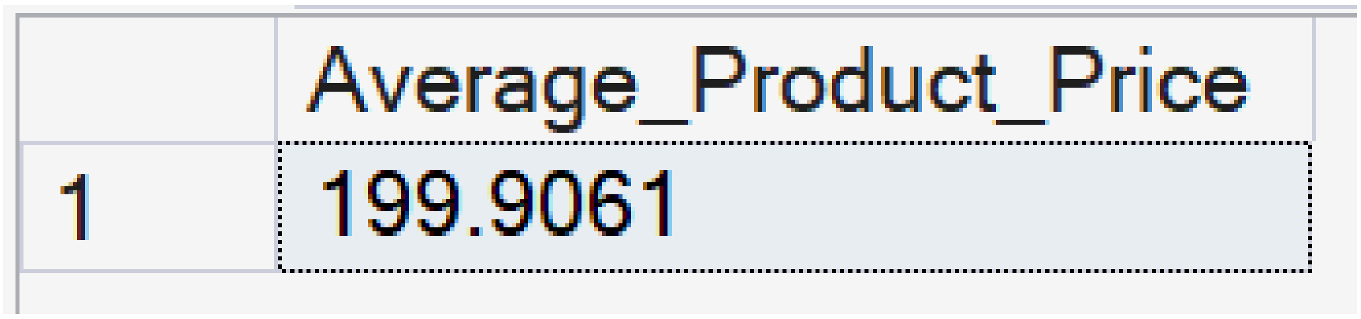
	Average_Total
1	1056.6875

15. **Query**: Calculate an average of all unique product prices.

    SQL:

```sql
SELECT AVG(DISTINCT RetailPrice) AS Average_Product_Price
FROM Products;
GO
```

    Result set:



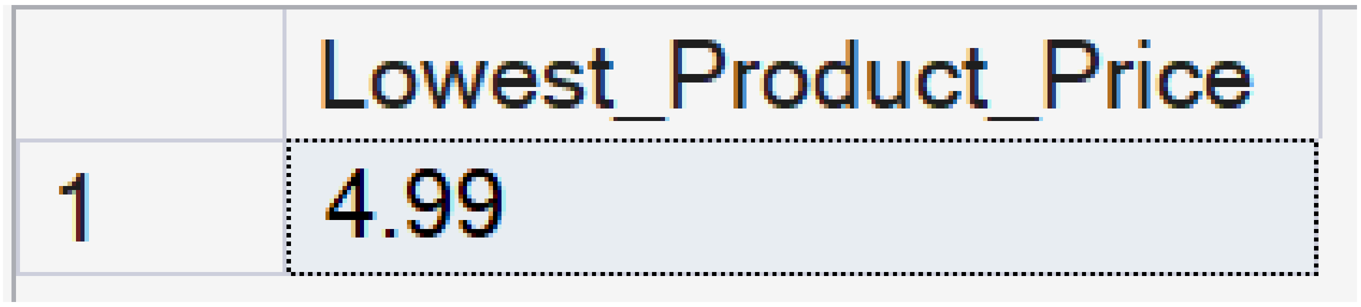
	Average_Product_Price
1	199.9061

16. **Query**: What is the lowest price we charge for a product?

    SQL:

```
SELECT MIN(RetailPrice) AS Lowest_Product_Price
FROM Products;
GO
```

Result set:



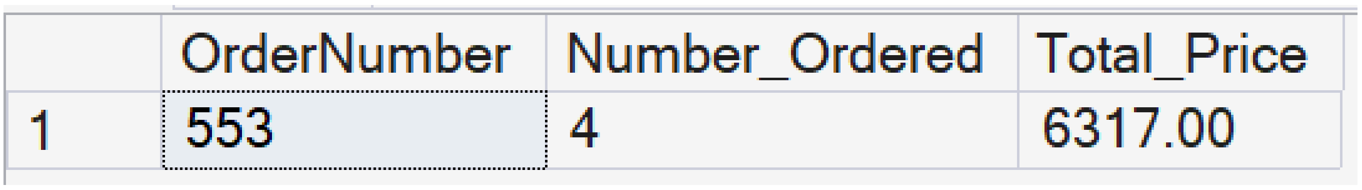
	Lowest_Product_Price
1	4.99

17. **Query**: How many different products were ordered on order number 553, and what was the total cost of that order? (use SUM and COUNT)

    SQL:

```
SELECT
  OrderNumber,
  COUNT(Ordernumber) AS Number_Ordered,
  SUM(QuotedPrice * QuantityOrdered) AS Total_Price
FROM Order_Details
GROUP BY OrderNumber
HAVING OrderNumber = 553;
GO
```

Result set:



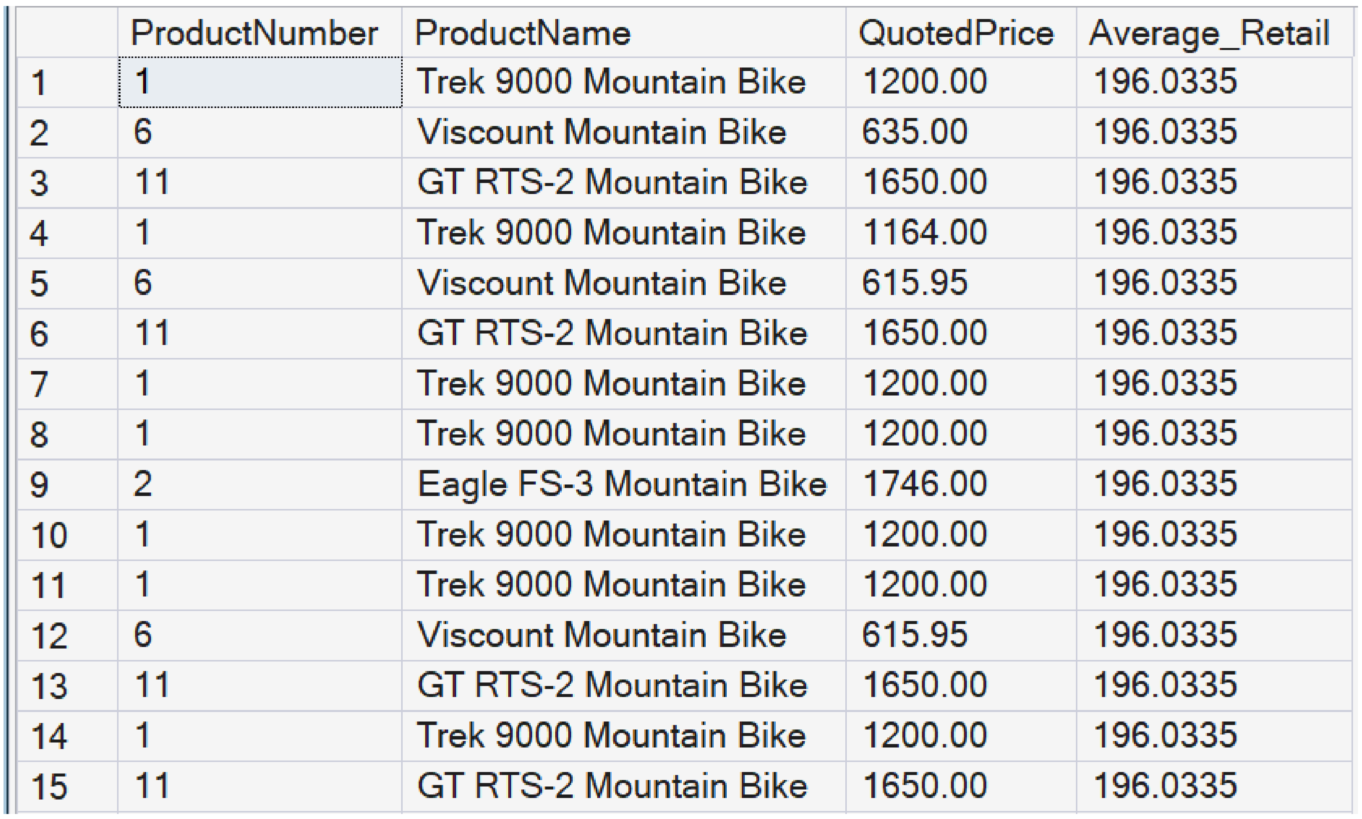
	OrderNumber	Number_Ordered	Total_Price
1	553	4	6317.00

18. **Query**: List the product names and numbers that have a quoted price greater than or equal to the overall average retail price in the products table.

    SQL:

```sql
SELECT
  p.ProductNumber,
  p.ProductName,
  od.QuotedPrice,
  (SELECT AVG(RetailPrice)
  FROM Products) AS Average_Retail
FROM Products AS p INNER JOIN Order_Details AS od
ON p.ProductNumber = od.ProductNumber
WHERE QuotedPrice >= (
  SELECT AVG(RetailPrice)
  FROM Products
);
GO
```

Result set:



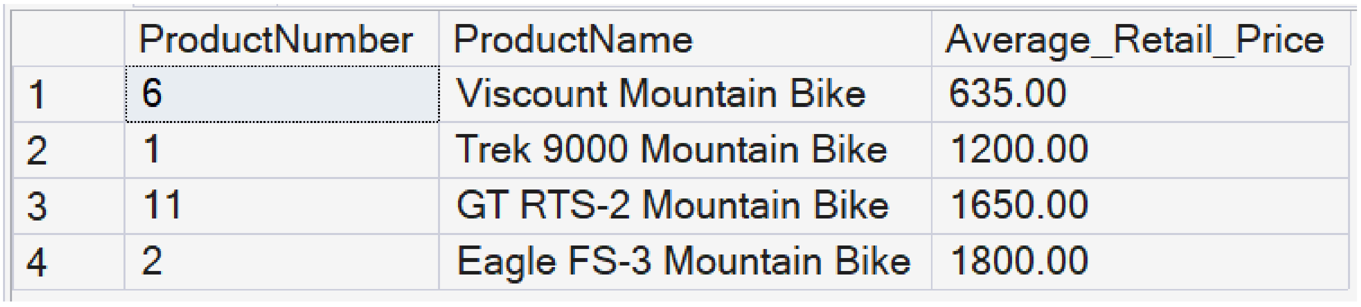
	ProductNumber	ProductName	QuotedPrice	Average_Retail
1	1	Trek 9000 Mountain Bike	1200.00	196.0335
2	6	Viscount Mountain Bike	635.00	196.0335
3	11	GT RTS-2 Mountain Bike	1650.00	196.0335
4	1	Trek 9000 Mountain Bike	1164.00	196.0335
5	6	Viscount Mountain Bike	615.95	196.0335
6	11	GT RTS-2 Mountain Bike	1650.00	196.0335
7	1	Trek 9000 Mountain Bike	1200.00	196.0335
8	1	Trek 9000 Mountain Bike	1200.00	196.0335
9	2	Eagle FS-3 Mountain Bike	1746.00	196.0335
10	1	Trek 9000 Mountain Bike	1200.00	196.0335
11	1	Trek 9000 Mountain Bike	1200.00	196.0335
12	6	Viscount Mountain Bike	615.95	196.0335
13	11	GT RTS-2 Mountain Bike	1650.00	196.0335
14	1	Trek 9000 Mountain Bike	1200.00	196.0335
15	11	GT RTS-2 Mountain Bike	1650.00	196.0335

19. **Query**: What is the average retail price of a mountain bike?

    SQL:

```sql
SELECT
  ProductNumber,
  ProductName,
  AVG(RetailPrice) AS Average_Retail_Price
FROM Products
WHERE CategoryID IN (
  SELECT CategoryID
  FROM Categories
  WHERE CategoryDescription = 'Bikes'
)
GROUP BY ProductNumber, ProductName
ORDER BY Average_Retail_Price;
GO
```

    Result set:



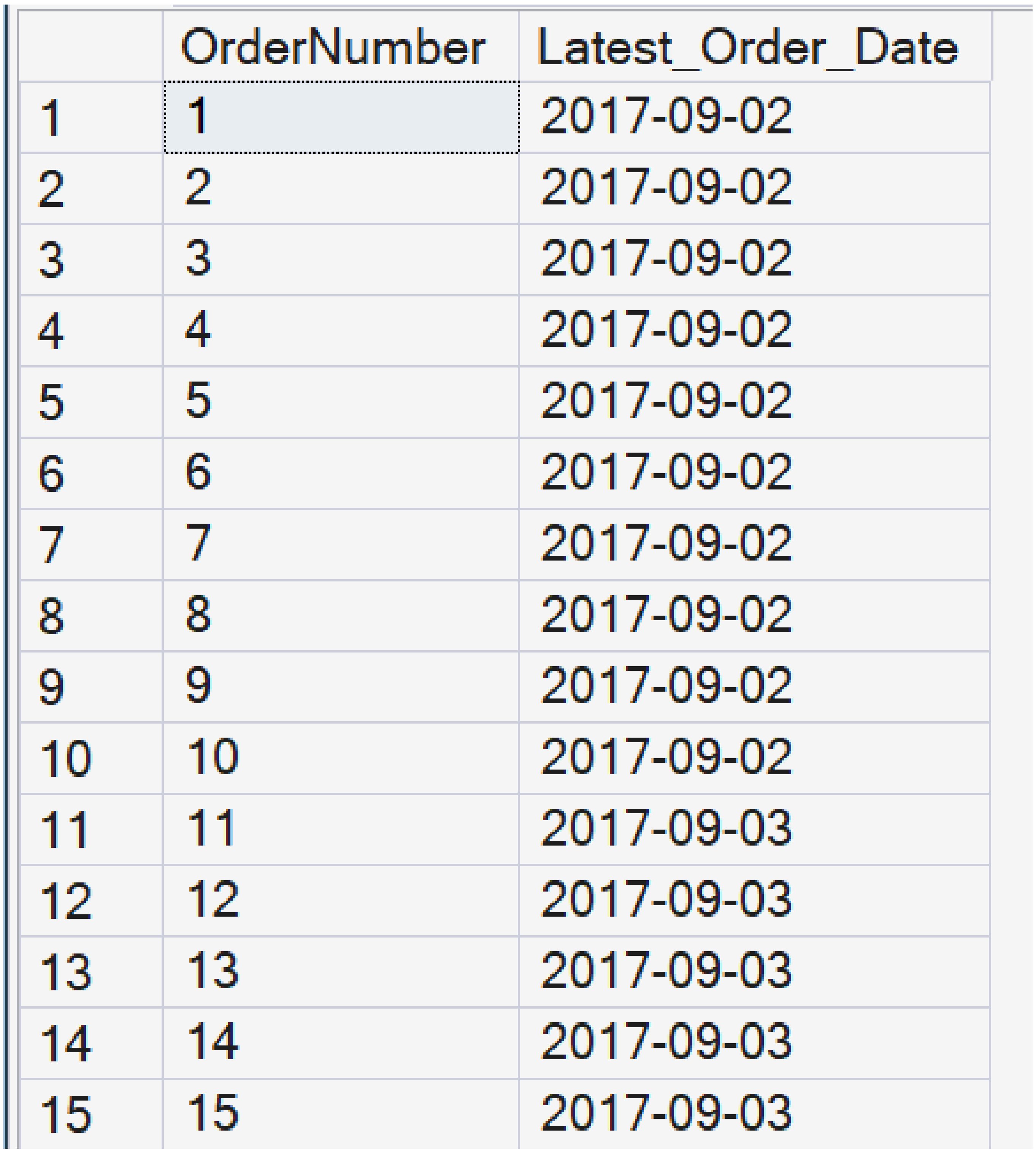
	ProductNumber	ProductName	Average_Retail_Price
1	6	Viscount Mountain Bike	635.00
2	1	Trek 9000 Mountain Bike	1200.00
3	11	GT RTS-2 Mountain Bike	1650.00
4	2	Eagle FS-3 Mountain Bike	1800.00

20. **Query**: What was the date of our most recent order?

    SQL:

```
SELECT
  OrderNumber,
  MAX(OrderDate) AS Latest_Order_Date
FROM Orders
GROUP BY OrderNumber;
GO
```

Result set:



	OrderNumber	Latest_Order_Date
1	1	2017-09-02
2	2	2017-09-02
3	3	2017-09-02
4	4	2017-09-02
5	5	2017-09-02
6	6	2017-09-02
7	7	2017-09-02
8	8	2017-09-02
9	9	2017-09-02
10	10	2017-09-02
11	11	2017-09-03
12	12	2017-09-03
13	13	2017-09-03
14	14	2017-09-03
15	15	2017-09-03

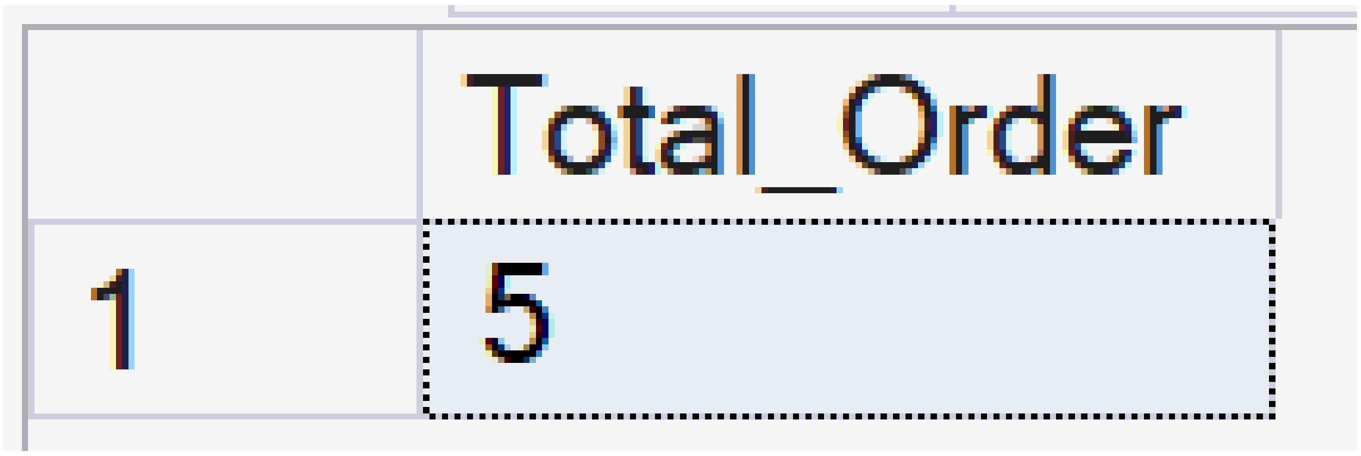
21. **Query**: What was the total amount for order number 8?

    SQL:

```sql
SELECT COUNT(OrderNumber) AS Total_Order
FROM Order_Details
WHERE OrderNumber = 8;
```

```
GO
```

Result set:



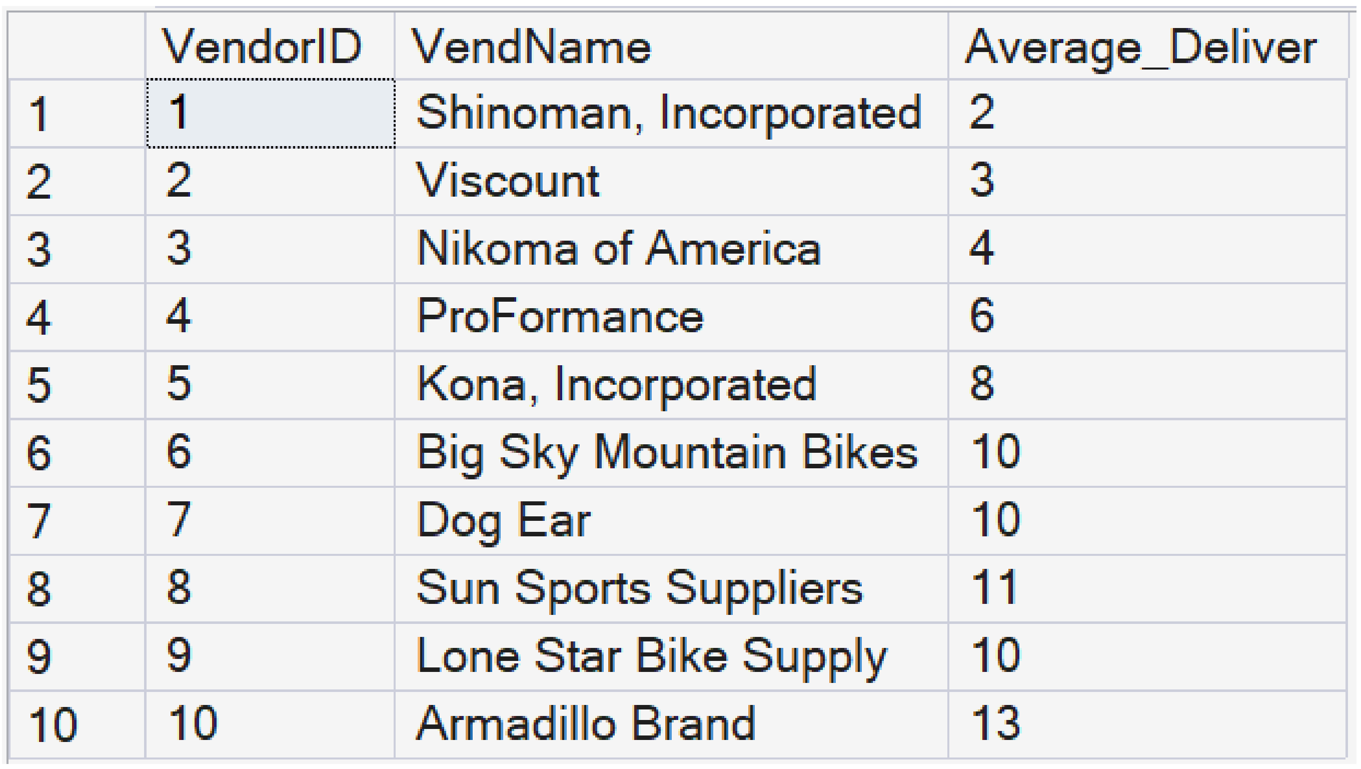
	Total_Order
1	5

22. **Query**: Show me each vendor and the average by vendor of the number of days to deliver products. (Hint: Use the AVG aggregate function and group on vendor.)

    SQL:

```sql
SELECT
  v.VendorID,
  v.VendName,
  AVG(pv.DaysToDeliver) AS Average_Deliver
FROM Vendors AS v INNER JOIN Product_Vendors AS pv
ON v.VendorID = pv.VendorID
GROUP BY v.VendorID, v.VendName;
GO
```

Result set:



	VendorID	VendName	Average_Deliver
1	1	Shinoman, Incorporated	2
2	2	Viscount	3
3	3	Nikoma of America	4
4	4	ProFormance	6
5	5	Kona, Incorporated	8
6	6	Big Sky Mountain Bikes	10
7	7	Dog Ear	10
8	8	Sun Sports Suppliers	11
9	9	Lone Star Bike Supply	10
10	10	Armadillo Brand	13

23. **Query**: My clothing supplier just announced a price increase of 4 percent. Update the price of the clothing products and add 4 percent. (use UPDATE)

    SQL:

```sql
UPDATE Product_Vendors
SET WholesalePrice = 1.04*WholesalePrice
WHERE ProductNumber IN (
  SELECT ProductNumber
  FROM Products
  WHERE CategoryID IN (
    SELECT CategoryID
    FROM Categories
    WHERE CategoryDescription = 'Clothing'
  )
);
GO
```

24. **Query**: Modify products by increasing the retail price by 4 percent for products that are clothing. (use UPDATE)

    SQL:

```sql
UPDATE Products
```

```sql
SET RetailPrice = RetailPrice + 0.04*RetailPrice
WHERE CategoryID IN (
  SELECT CategoryID
  FROM Categories
  WHERE CategoryDescription = 'Clothing'
);
GO
```

25. **Query**: Change the orders table by setting the order total to the sum of quantity ordered times quoted price for all related order detail rows. (use UPDATE with subquery)

    SQL:

```sql
UPDATE Orders
SET OrderTotal = sub.OrderTotal
FROM Orders AS o INNER JOIN (
  SELECT OrderNumber, SUM(QuotedPrice*QuantityOrdered) AS OrderTotal
  FROM Order_Details AS od
  GROUP BY OrderNumber
) AS sub ON o.OrderNumber = sub.OrderNumber;
GO
```

26. **Query**: Reduce the quoted price by 2 percent for orders shipped more than 30 days after the order date. (use UPDATE with subquery)

    SQL:

```sql
UPDATE Order_Details
SET QuotedPrice = 0.98*QuotedPrice
WHERE OrderNumber IN (
  SELECT OrderNumber
  FROM Orders
  WHERE DATEDIFF(DAY, OrderDate, ShipDate) > 30
);
GO
```

27. **Query**: * Make sure the retail price for all bikes is at least a 45 percent markup over the wholesale price of the vendor with the lowest cost. (update and subsequent) - 1 row

    SQL:

```
UPDATE Products
SET RetailPrice = 1.45 * WholesalePrice
FROM Products AS p INNER JOIN Product_Vendors AS pv
ON p.ProductNumber = pv.ProductNumber
WHERE WholesalePrice = (
  SELECT MIN(WholesalePrice)
  FROM Product_Vendors)
AND RetailPrice < (
SELECT MIN(WholesalePrice) * 1.45
FROM Product_Vendors
)
AND CategoryID IN (
  SELECT CategoryID
  FROM Categories
  WHERE CategoryDescription = 'Bikes'
);
GO
```

28. **Query**: Apply a 5 percent discount to all orders for customers who purchased more than $50,000 in the month of October 2017. (hint You need a subquery within a subquery to fetch the order numbers for all orders where the customer ID of the order is in the set of customers who ordered more than $50,000 in the month of October.) (639 rows changed)

    SQL:

```
UPDATE Orders
SET OrderTotal = 0.95 * OrderTotal
WHERE OrderNumber IN (
  SELECT OrderNumber
  FROM Orders
  WHERE CustomerID IN (
    SELECT CustomerID
    FROM Orders
    WHERE MONTH(OrderDate) = 10 AND YEAR(OrderDate) = 2017
    GROUP BY CustomerID
    HAVING SUM(OrderTotal) > 50000
  )
);
GO
```

29. **Query**: Set the retail price of accessories (category = 1) to the wholesale price of the highestpriced vendor plus 35 percent. (11 rows changed).

    SQL:

```sql
UPDATE Products
SET RetailPrice = (
  SELECT MAX(WholesalePrice) * 1.35
  FROM Product_Vendors
)
WHERE CategoryID = 1;
GO
```

30. **Query**: Copy to the Employees table the relevant columns in the Customers table for customer David Smith. (INSERT INTO)

    SQL:

```sql
INSERT INTO Employees(EmpFirstName, EmpLastName, EmpStreetAddress, Em
pCity, EmpState, EmpZipCode, EmpAreaCode, EmpPhoneNumber)
SELECT CustFirstName, CustLastName, CustStreetAddress, CustCity, Cust
State, CustZipCode, CustAreaCode, CustPhoneNumber
FROM Customers
WHERE CustFirstName = 'David' AND CustLastName = 'Smith';
GO
```

31. **Query**: Add a new product named 'Hot Dog Spinner' with a retail price of $895 in the Bikes category. (INSERT INTO)

    SQL:

```sql
INSERT INTO Products(ProductName, RetailPrice, CategoryID)
VALUES('Hot␣Dog␣Spinner', 895, (SELECT CategoryID FROM Categories
WHERE CategoryDescription = 'Bikes'));
GO
```

This is the end of the report