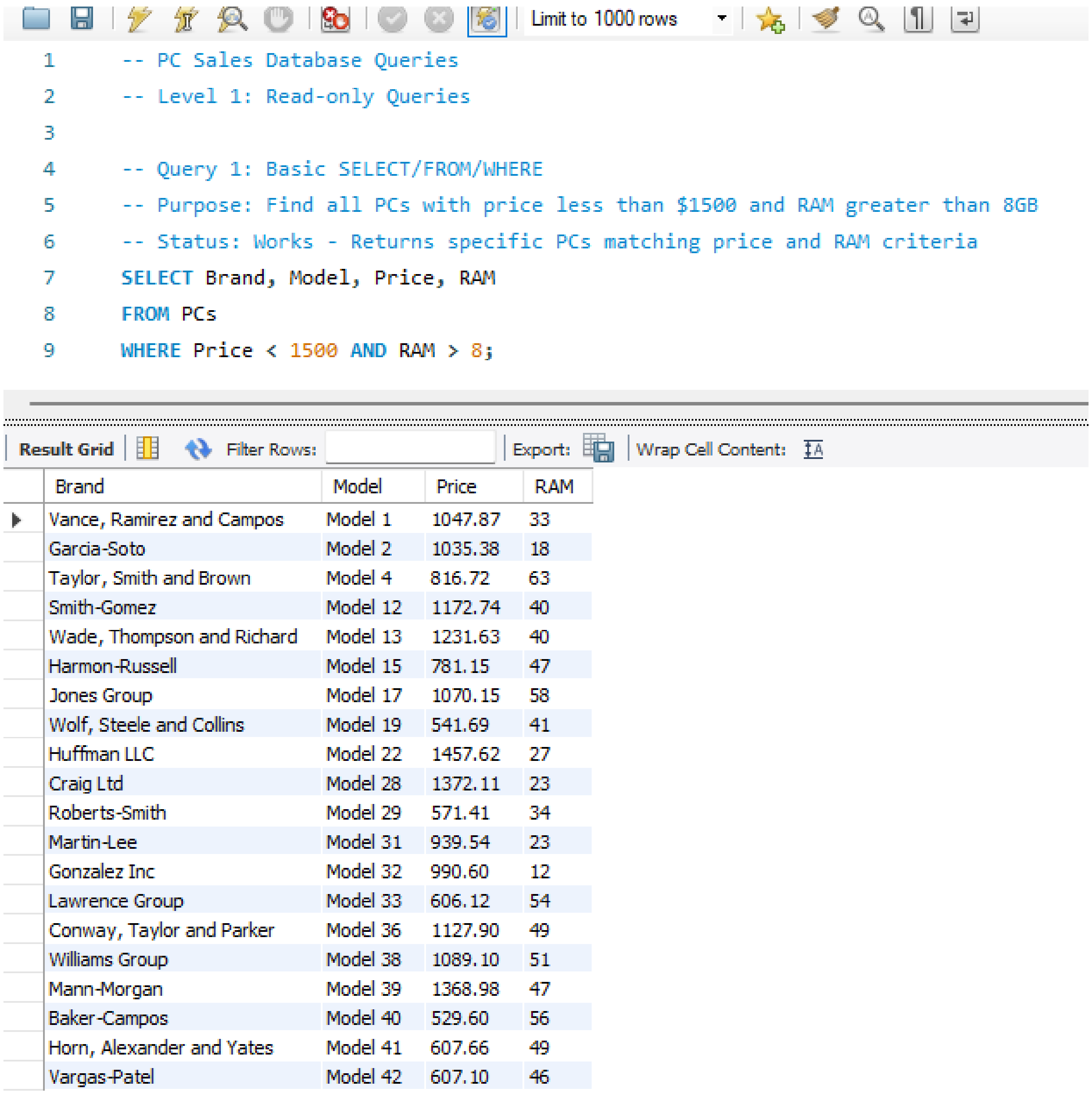
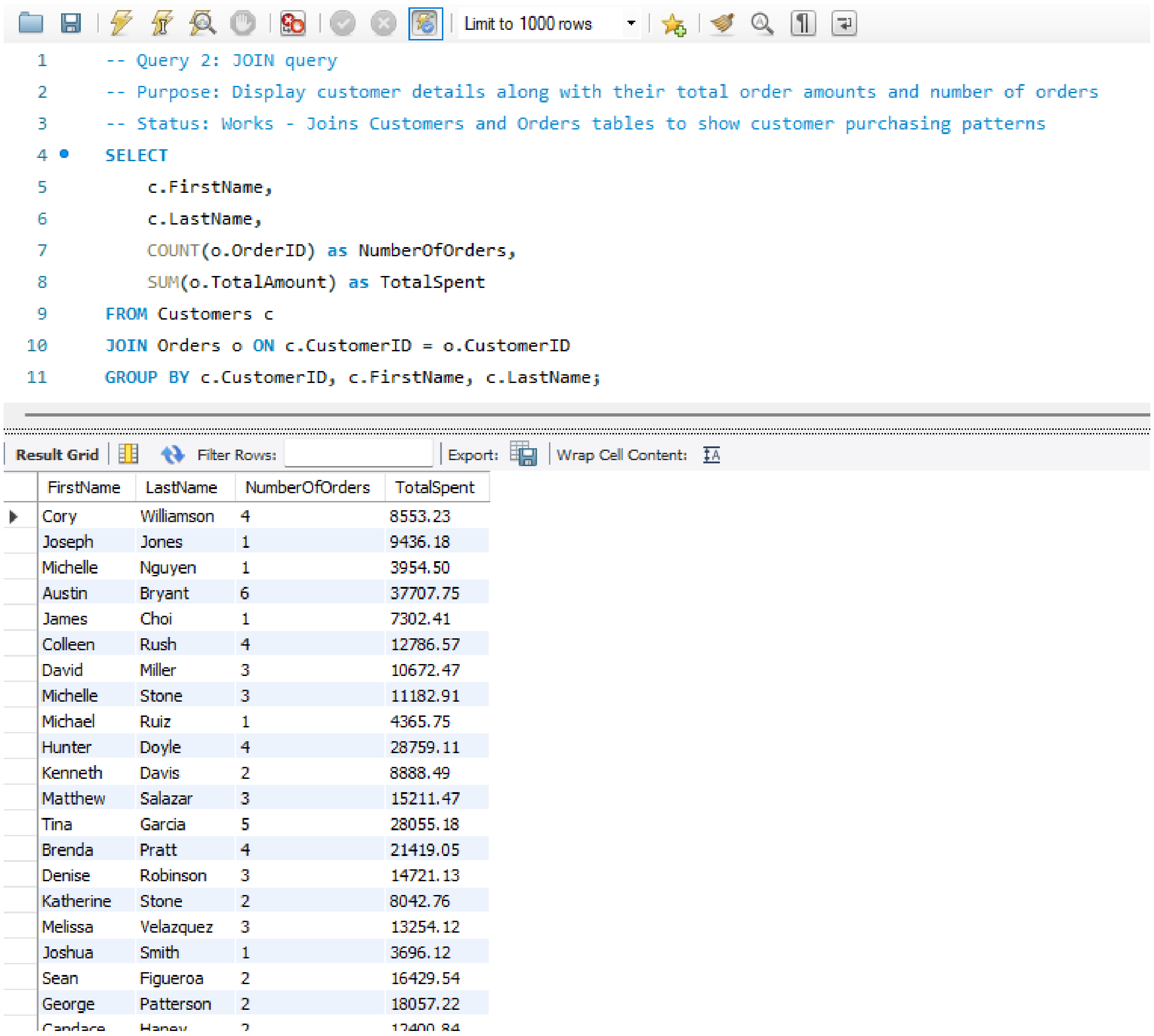


```sql
-- PC Sales Database Queries
-- Level 1: Read-only Queries

-- Query 1: Basic SELECT/FROM/WHERE
-- Purpose: Find all PCs with price less than $1500 and RAM greater than 8GB
-- Status: Works - Returns specific PCs matching price and RAM criteria
SELECT Brand, Model, Price, RAM
FROM PCs
WHERE Price < 1500 AND RAM > 8;
```

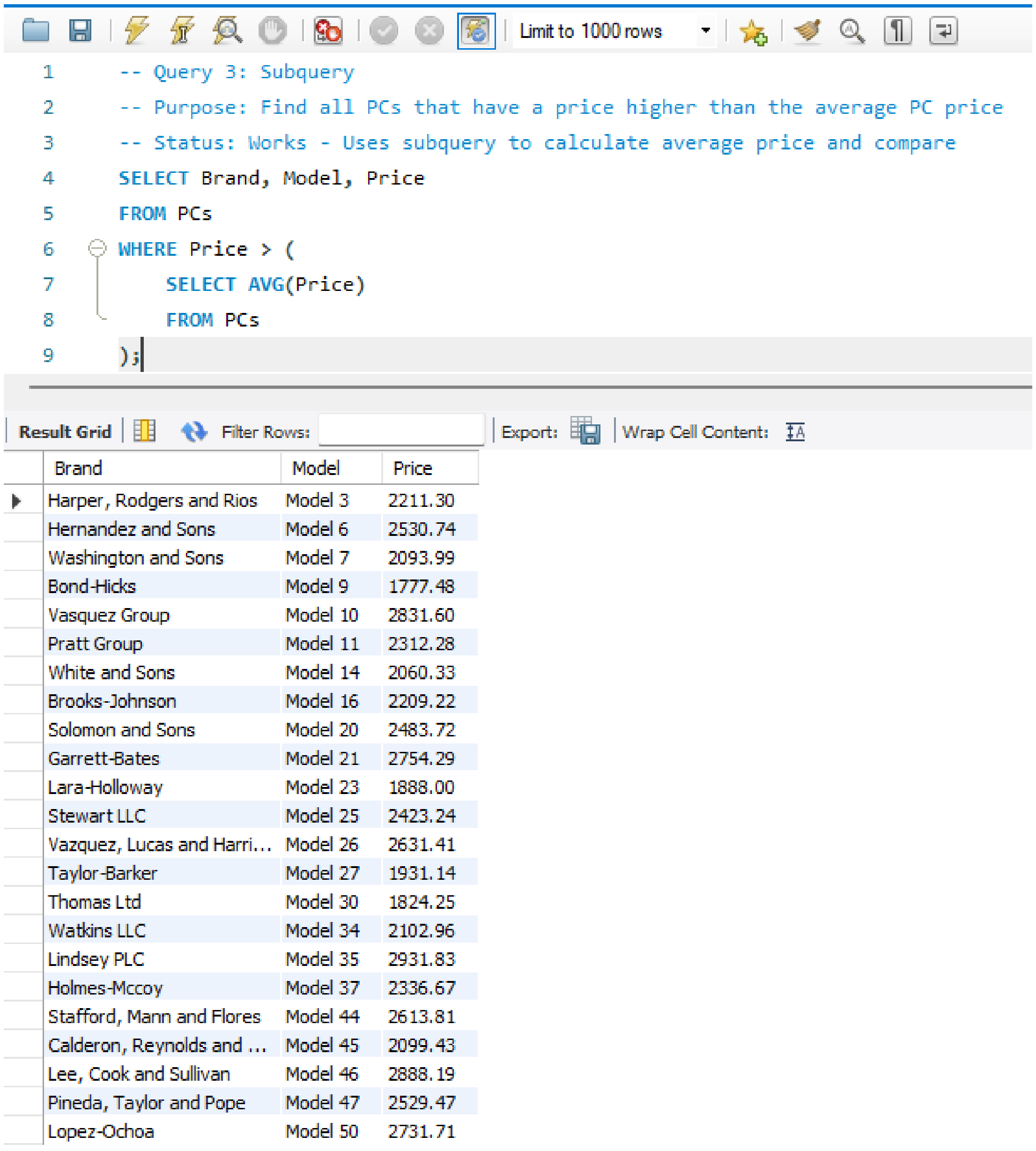
Brand	Model	Price	RAM
Vance, Ramirez and Campos	Model 1	1047.87	33
Garcia-Soto	Model 2	1035.38	18
Taylor, Smith and Brown	Model 4	816.72	63
Smith-Gomez	Model 12	1172.74	40
Wade, Thompson and Richard	Model 13	1231.63	40
Harmon-Russell	Model 15	781.15	47
Jones Group	Model 17	1070.15	58
Wolf, Steele and Collins	Model 19	541.69	41
Huffman LLC	Model 22	1457.62	27
Craig Ltd	Model 28	1372.11	23
Roberts-Smith	Model 29	571.41	34
Martin-Lee	Model 31	939.54	23
Gonzalez Inc	Model 32	990.60	12
Lawrence Group	Model 33	606.12	54
Conway, Taylor and Parker	Model 36	1127.90	49
Williams Group	Model 38	1089.10	51
Mann-Morgan	Model 39	1368.98	47
Baker-Campos	Model 40	529.60	56
Horn, Alexander and Yates	Model 41	607.66	49
Vargas-Patel	Model 42	607.10	46



Limit to 1000 rows

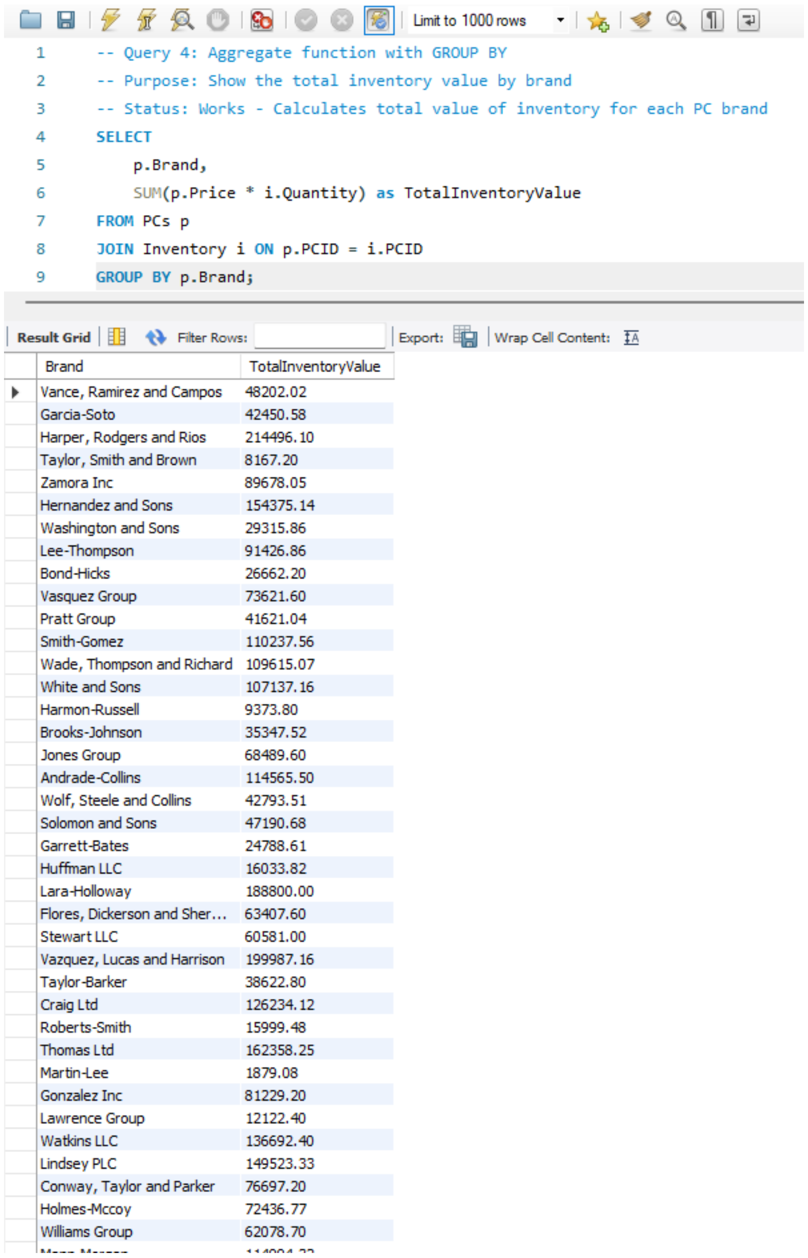
```sql
-- Query 2: JOIN query
-- Purpose: Display customer details along with their total order amounts and number of orders
-- Status: Works - Joins Customers and Orders tables to show customer purchasing patterns
SELECT
    c.FirstName,
    c.LastName,
    COUNT(o.OrderID) as NumberOfOrders,
    SUM(o.TotalAmount) as TotalSpent
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName;
```

FirstName	LastName	NumberOfOrders	TotalSpent
Cory	Williamson	4	8553.23
Joseph	Jones	1	9436.18
Michelle	Nguyen	1	3954.50
Austin	Bryant	6	37707.75
James	Choi	1	7302.41
Colleen	Rush	4	12786.57
David	Miller	3	10672.47
Michelle	Stone	3	11182.91
Michael	Ruiz	1	4365.75
Hunter	Doyle	4	28759.11
Kenneth	Davis	2	8888.49
Matthew	Salazar	3	15211.47
Tina	Garcia	5	28055.18
Brenda	Pratt	4	21419.05
Denise	Robinson	3	14721.13
Katherine	Stone	2	8042.76
Melissa	Velazquez	3	13254.12
Joshua	Smith	1	3696.12
Sean	Figueroa	2	16429.54
George	Patterson	2	18057.22
Candace	Haney	2	12400.84



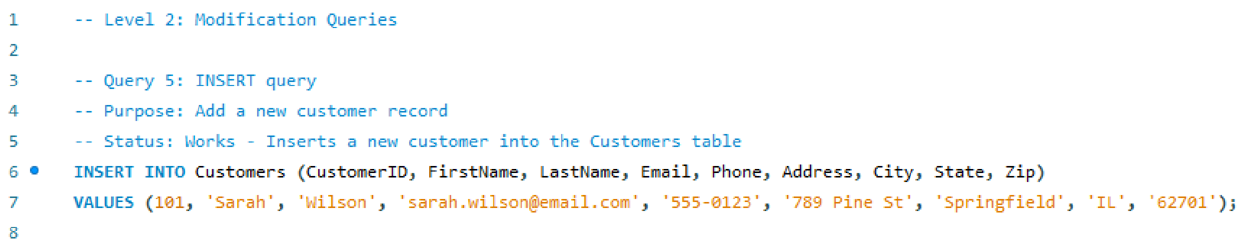
```sql
-- Query 3: Subquery
-- Purpose: Find all PCs that have a price higher than the average PC price
-- Status: Works - Uses subquery to calculate average price and compare
SELECT Brand, Model, Price
FROM PCs
WHERE Price > (
    SELECT AVG(Price)
    FROM PCs
);
```

Brand	Model	Price
Harper, Rodgers and Rios	Model 3	2211.30
Hernandez and Sons	Model 6	2530.74
Washington and Sons	Model 7	2093.99
Bond-Hicks	Model 9	1777.48
Vasquez Group	Model 10	2831.60
Pratt Group	Model 11	2312.28
White and Sons	Model 14	2060.33
Brooks-Johnson	Model 16	2209.22
Solomon and Sons	Model 20	2483.72
Garrett-Bates	Model 21	2754.29
Lara-Holloway	Model 23	1888.00
Stewart LLC	Model 25	2423.24
Vazquez, Lucas and Harri...	Model 26	2631.41
Taylor-Barker	Model 27	1931.14
Thomas Ltd	Model 30	1824.25
Watkins LLC	Model 34	2102.96
Lindsey PLC	Model 35	2931.83
Holmes-Mccoy	Model 37	2336.67
Stafford, Mann and Flores	Model 44	2613.81
Calderon, Reynolds and ...	Model 45	2099.43
Lee, Cook and Sullivan	Model 46	2888.19
Pineda, Taylor and Pope	Model 47	2529.47
Lopez-Ochoa	Model 50	2731.71

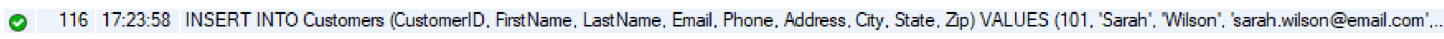


```sql
-- Query 4: Aggregate function with GROUP BY
-- Purpose: Show the total inventory value by brand
-- Status: Works - Calculates total value of inventory for each PC brand
SELECT
    p.Brand,
    SUM(p.Price * i.Quantity) as TotalInventoryValue
FROM PCs p
JOIN Inventory i ON p.PCID = i.PCID
GROUP BY p.Brand;
```

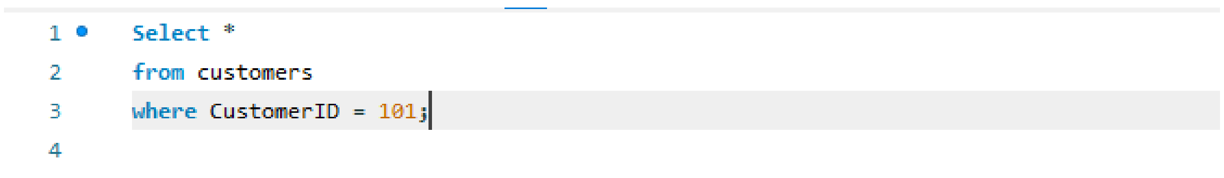
Brand	TotalInventoryValue
Vance, Ramirez and Campos	48202.02
Garcia-Soto	42450.58
Harper, Rodgers and Rios	214496.10
Taylor, Smith and Brown	8167.20
Zamora Inc	89678.05
Hernandez and Sons	154375.14
Washington and Sons	29315.86
Lee-Thompson	91426.86
Bond-Hicks	26662.20
Vasquez Group	73621.60
Pratt Group	41621.04
Smith-Gomez	110237.56
Wade, Thompson and Richard	109615.07
White and Sons	107137.16
Harmon-Russell	9373.80
Brooks-Johnson	35347.52
Jones Group	68489.60
Andrade-Collins	114565.50
Wolf, Steele and Collins	42793.51
Solomon and Sons	47190.68
Garrett-Bates	24788.61
Huffman LLC	16033.82
Lara-Holloway	188800.00
Flores, Dickerson and Sher...	63407.60
Stewart LLC	60581.00
Vazquez, Lucas and Harrison	199987.16
Taylor-Barker	38622.80
Craig Ltd	126234.12
Roberts-Smith	15999.48
Thomas Ltd	162358.25
Martin-Lee	1879.08
Gonzalez Inc	81229.20
Lawrence Group	12122.40
Watkins LLC	136692.40
Lindsey PLC	149523.33
Conway, Taylor and Parker	76697.20
Holmes-Mccoy	72436.77
Williams Group	62078.70



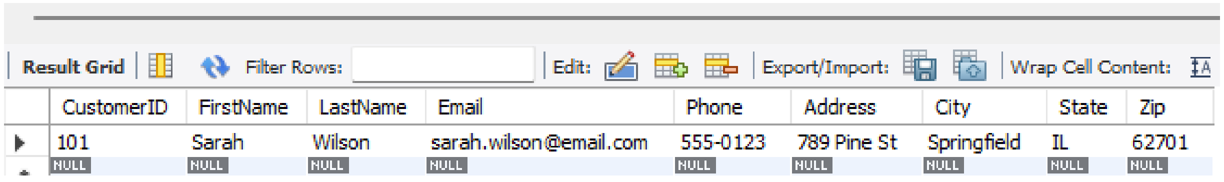
```sql
-- Level 2: Modification Queries

-- Query 5: INSERT query
-- Purpose: Add a new customer record
-- Status: Works - Inserts a new customer into the Customers table
INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address, City, State, Zip)
VALUES (101, 'Sarah', 'Wilson', 'sarah.wilson@email.com', '555-0123', '789 Pine St', 'Springfield', 'IL', '62701');

```



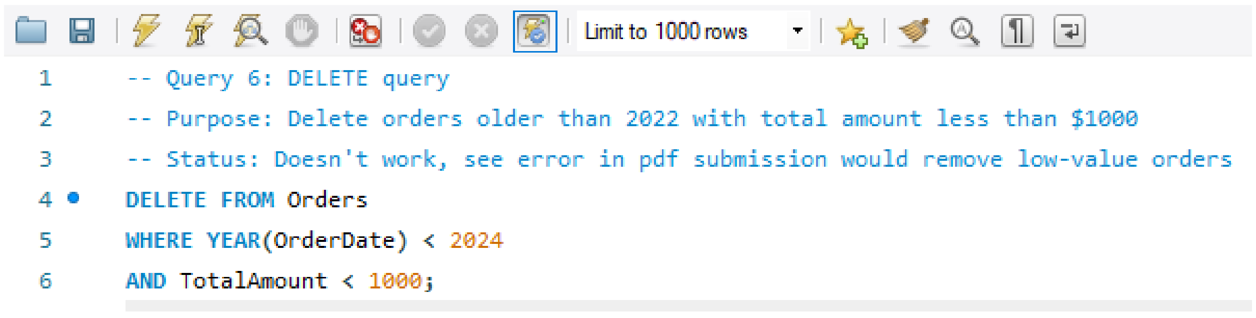
```
✓  116  17:23:58  INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address, City, State, Zip) VALUES (101, 'Sarah', 'Wilson', 'sarah.wilson@email.com',...
```



```sql
Select *
from customers
where CustomerID = 101;
```

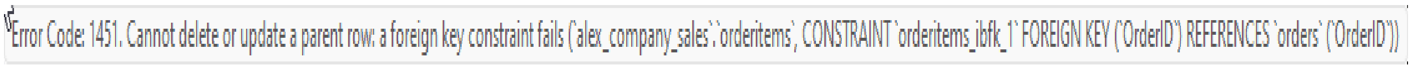


CustomerID	FirstName	LastName	Email	Phone	Address	City	State	Zip
101	Sarah	Wilson	sarah.wilson@email.com	555-0123	789 Pine St	Springfield	IL	62701
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

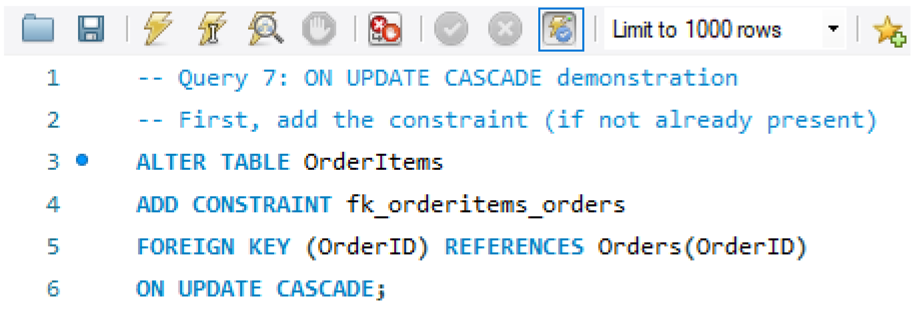


Limit to 1000 rows

```sql
-- Query 6: DELETE query
-- Purpose: Delete orders older than 2022 with total amount less than $1000
-- Status: Doesn't work, see error in pdf submission would remove low-value orders
DELETE FROM Orders
WHERE YEAR(OrderDate) < 2024
AND TotalAmount < 1000;
```

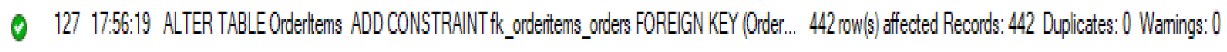


```
Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails (`alex_company_sales`.`orderitems`, CONSTRAINT `orderitems_ibfk_1` FOREIGN KEY (`OrderID`) REFERENCES `orders` (`OrderID`))
```

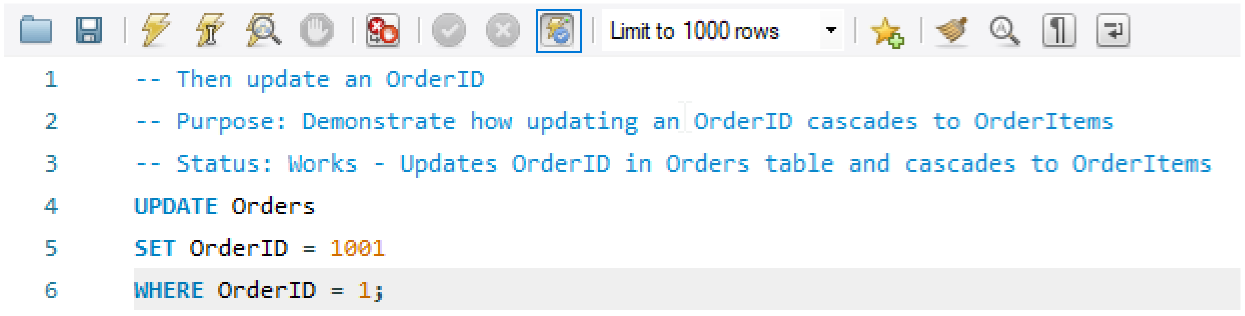


Limit to 1000 rows

```sql
-- Query 7: ON UPDATE CASCADE demonstration
-- First, add the constraint (if not already present)
ALTER TABLE OrderItems
ADD CONSTRAINT fk_orderitems_orders
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
ON UPDATE CASCADE;
```

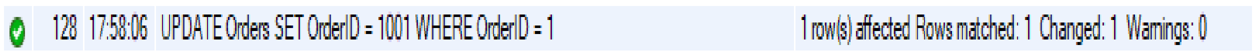


```
127  17:56:19  ALTER TABLE OrderItems  ADD CONSTRAINT fk_orderitems_orders FOREIGN KEY (Order...  442 row(s) affected Records: 442  Duplicates: 0  Warnings: 0
```

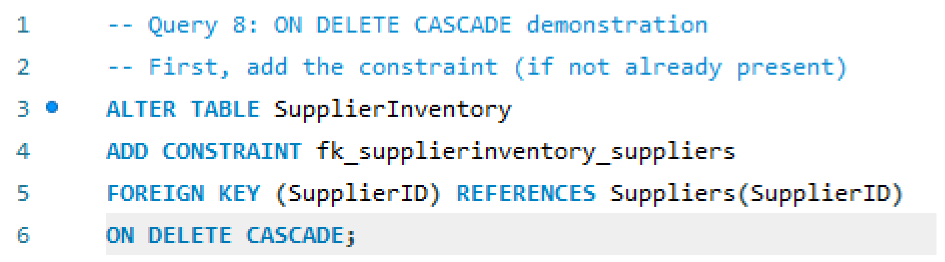


Limit to 1000 rows

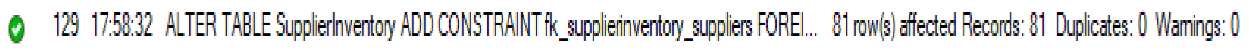
```sql
-- Then update an OrderID
-- Purpose: Demonstrate how updating an OrderID cascades to OrderItems
-- Status: Works - Updates OrderID in Orders table and cascades to OrderItems
UPDATE Orders
SET OrderID = 1001
WHERE OrderID = 1;
```



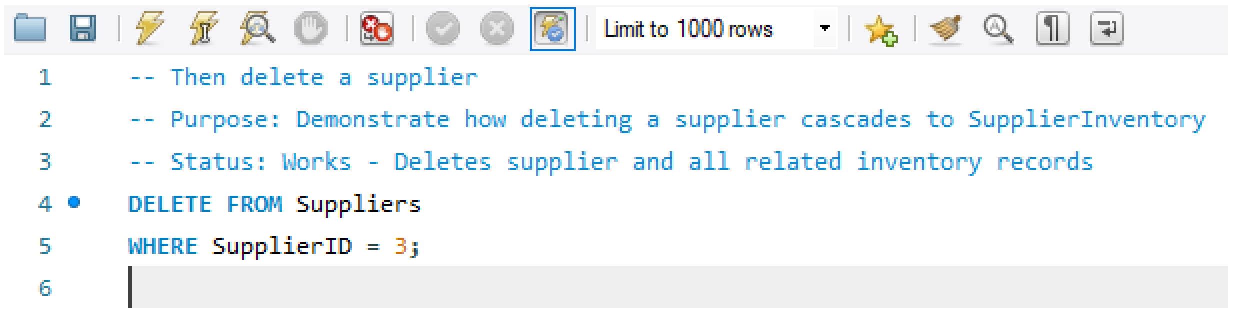
```
128  17:58:06  UPDATE Orders SET OrderID = 1001 WHERE OrderID = 1      1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0
```



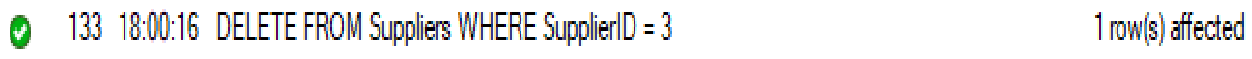
```sql
-- Query 8: ON DELETE CASCADE demonstration
-- First, add the constraint (if not already present)
ALTER TABLE SupplierInventory
ADD CONSTRAINT fk_supplierinventory_suppliers
FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)
ON DELETE CASCADE;
```



```
129  17:58:32  ALTER TABLE SupplierInventory ADD CONSTRAINT fk_supplierinventory_suppliers FOREI...  81 row(s) affected Records: 81  Duplicates: 0  Warnings: 0
```



```sql
-- Then delete a supplier
-- Purpose: Demonstrate how deleting a supplier cascades to SupplierInventory
-- Status: Works - Deletes supplier and all related inventory records
DELETE FROM Suppliers
WHERE SupplierID = 3;
```



```
133  18:00:16  DELETE FROM Suppliers WHERE SupplierID = 3                                  1 row(s) affected
```