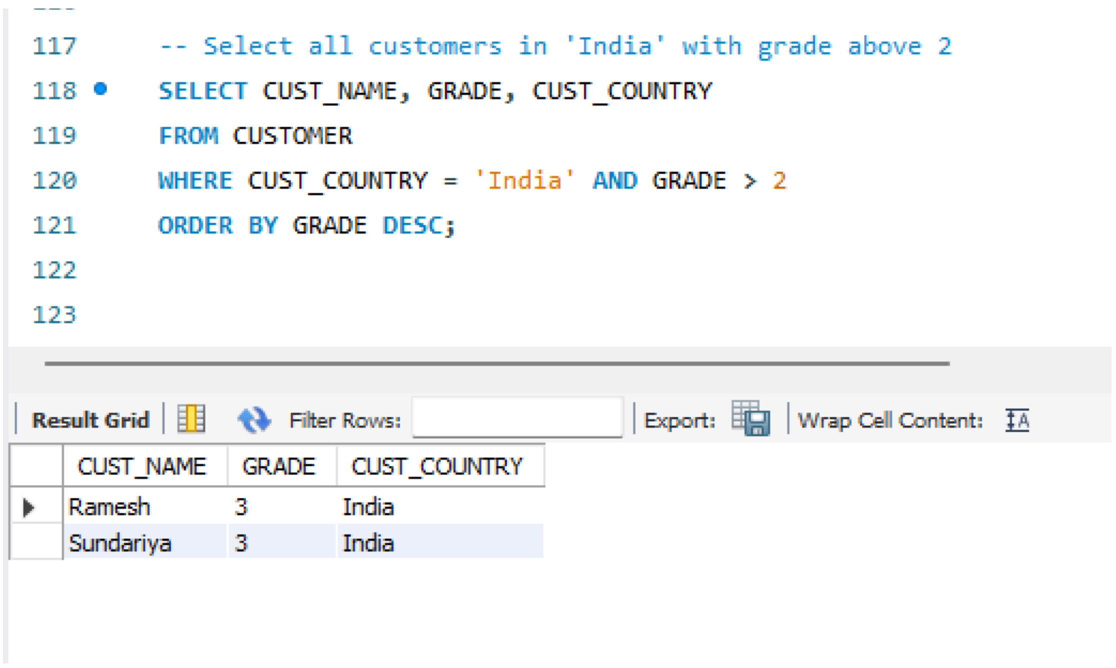
SQL for Data Analysis

Queries and Outputs :

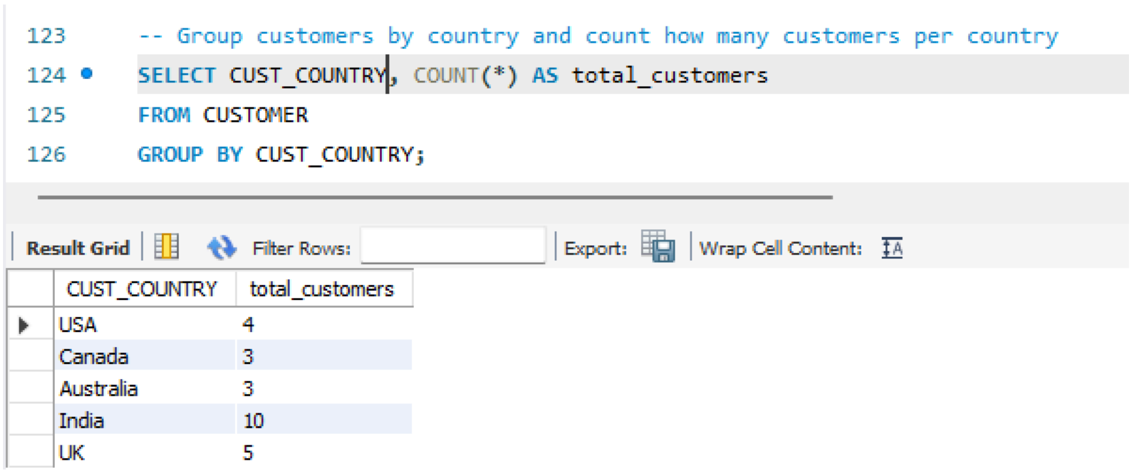
1.



```sql
-- Select all customers in 'India' with grade above 2
SELECT CUST_NAME, GRADE, CUST_COUNTRY
FROM CUSTOMER
WHERE CUST_COUNTRY = 'India' AND GRADE > 2
ORDER BY GRADE DESC;
```

CUST_NAME	GRADE	CUST_COUNTRY
Ramesh	3	India
Sundariya	3	India

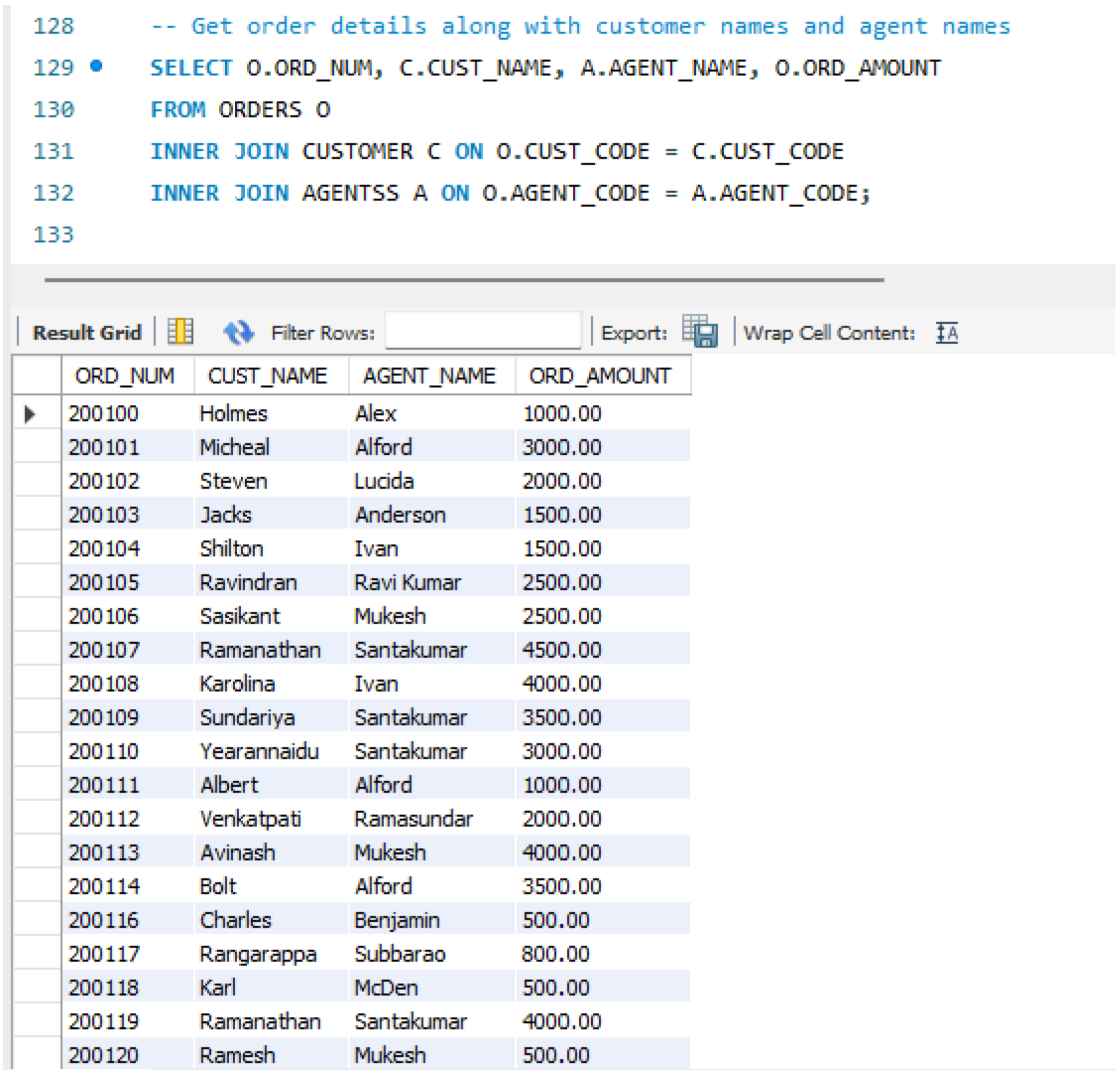
2.



```sql
-- Group customers by country and count how many customers per country
SELECT CUST_COUNTRY, COUNT(*) AS total_customers
FROM CUSTOMER
GROUP BY CUST_COUNTRY;
```

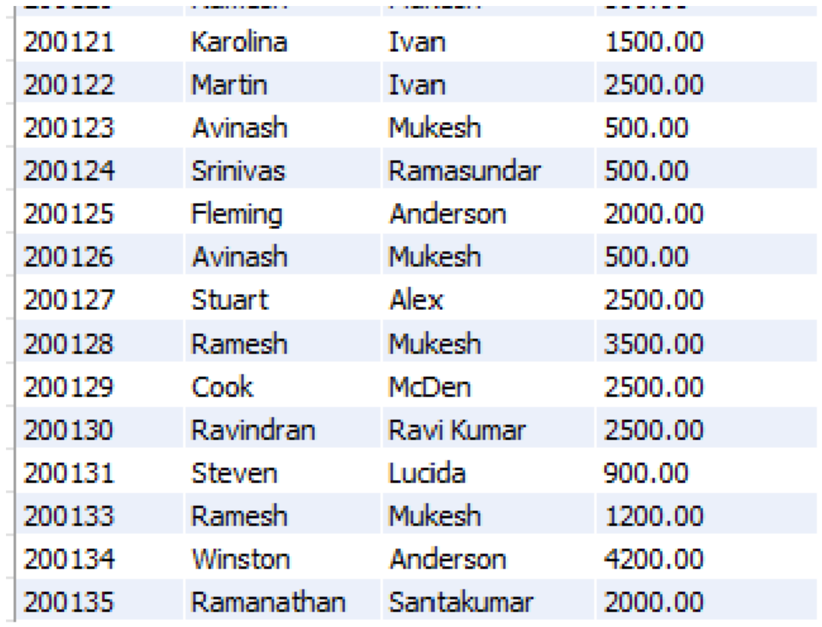
CUST_COUNTRY	total_customers
USA	4
Canada	3
Australia	3
India	10
UK	5

3.Inner Join



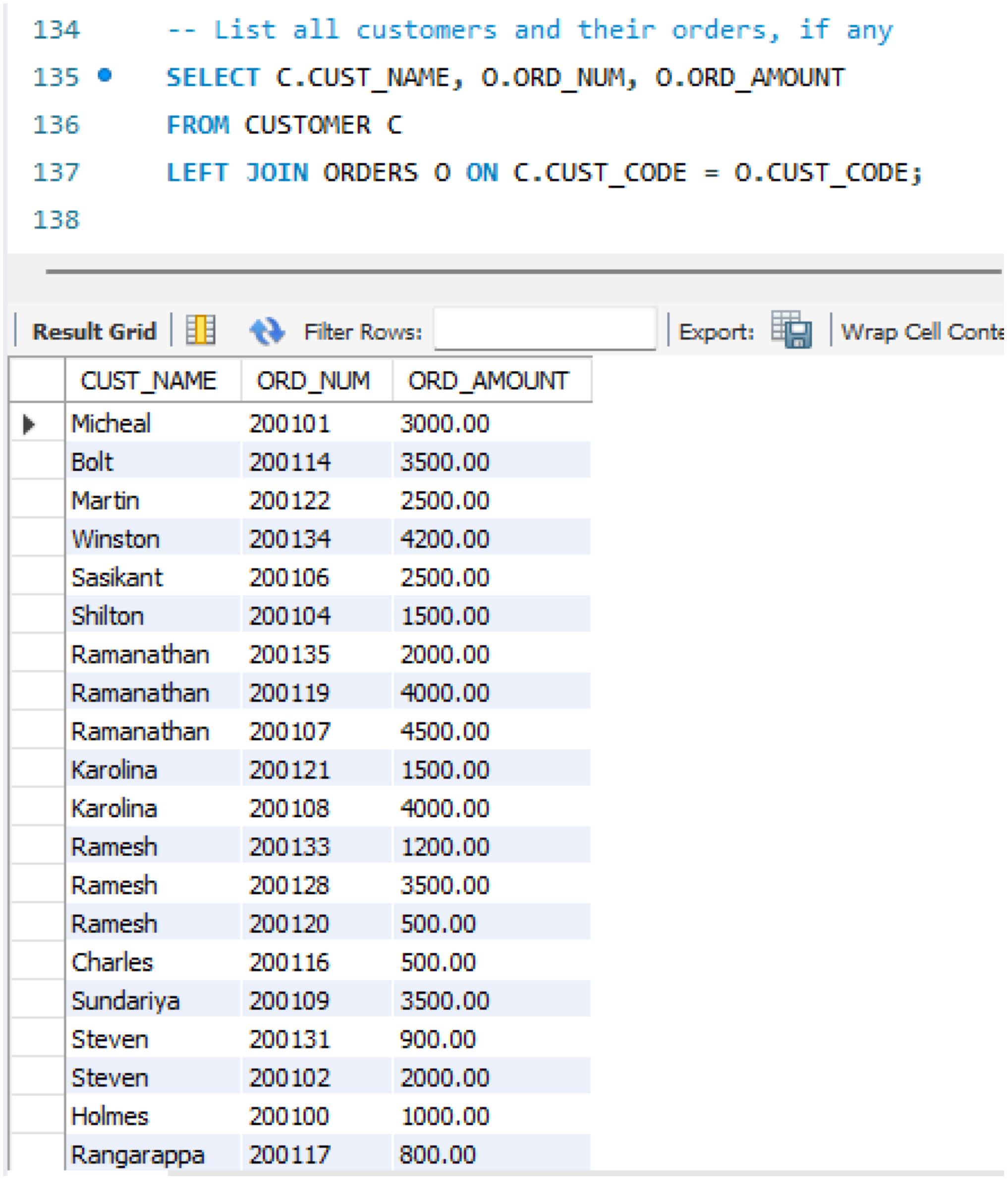
```
-- Get order details along with customer names and agent names
SELECT O.ORD_NUM, C.CUST_NAME, A.AGENT_NAME, O.ORD_AMOUNT
FROM ORDERS O
INNER JOIN CUSTOMER C ON O.CUST_CODE = C.CUST_CODE
INNER JOIN AGENTSS A ON O.AGENT_CODE = A.AGENT_CODE;
```

ORD_NUM	CUST_NAME	AGENT_NAME	ORD_AMOUNT
200100	Holmes	Alex	1000.00
200101	Micheal	Alford	3000.00
200102	Steven	Lucida	2000.00
200103	Jacks	Anderson	1500.00
200104	Shilton	Ivan	1500.00
200105	Ravindran	Ravi Kumar	2500.00
200106	Sasikant	Mukesh	2500.00
200107	Ramanathan	Santakumar	4500.00
200108	Karolina	Ivan	4000.00
200109	Sundariya	Santakumar	3500.00
200110	Yearannaidu	Santakumar	3000.00
200111	Albert	Alford	1000.00
200112	Venkatpati	Ramasundar	2000.00
200113	Avinash	Mukesh	4000.00
200114	Bolt	Alford	3500.00
200116	Charles	Benjamin	500.00
200117	Rangarappa	Subbarao	800.00
200118	Karl	McDen	500.00
200119	Ramanathan	Santakumar	4000.00
200120	Ramesh	Mukesh	500.00



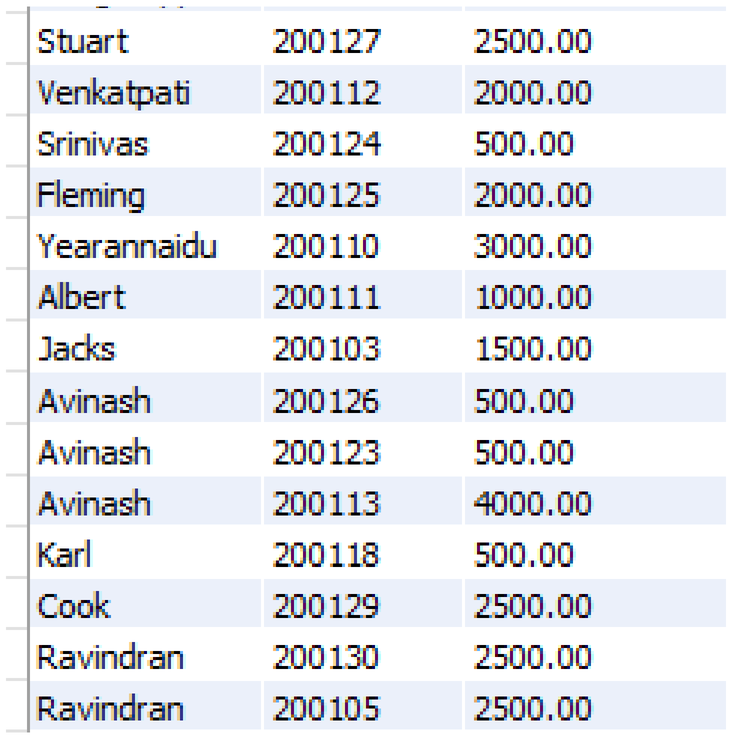
200121	Karolina	Ivan	1500.00
200122	Martin	Ivan	2500.00
200123	Avinash	Mukesh	500.00
200124	Srinivas	Ramasundar	500.00
200125	Fleming	Anderson	2000.00
200126	Avinash	Mukesh	500.00
200127	Stuart	Alex	2500.00
200128	Ramesh	Mukesh	3500.00
200129	Cook	McDen	2500.00
200130	Ravindran	Ravi Kumar	2500.00
200131	Steven	Lucida	900.00
200133	Ramesh	Mukesh	1200.00
200134	Winston	Anderson	4200.00
200135	Ramanathan	Santakumar	2000.00

4. Left Join



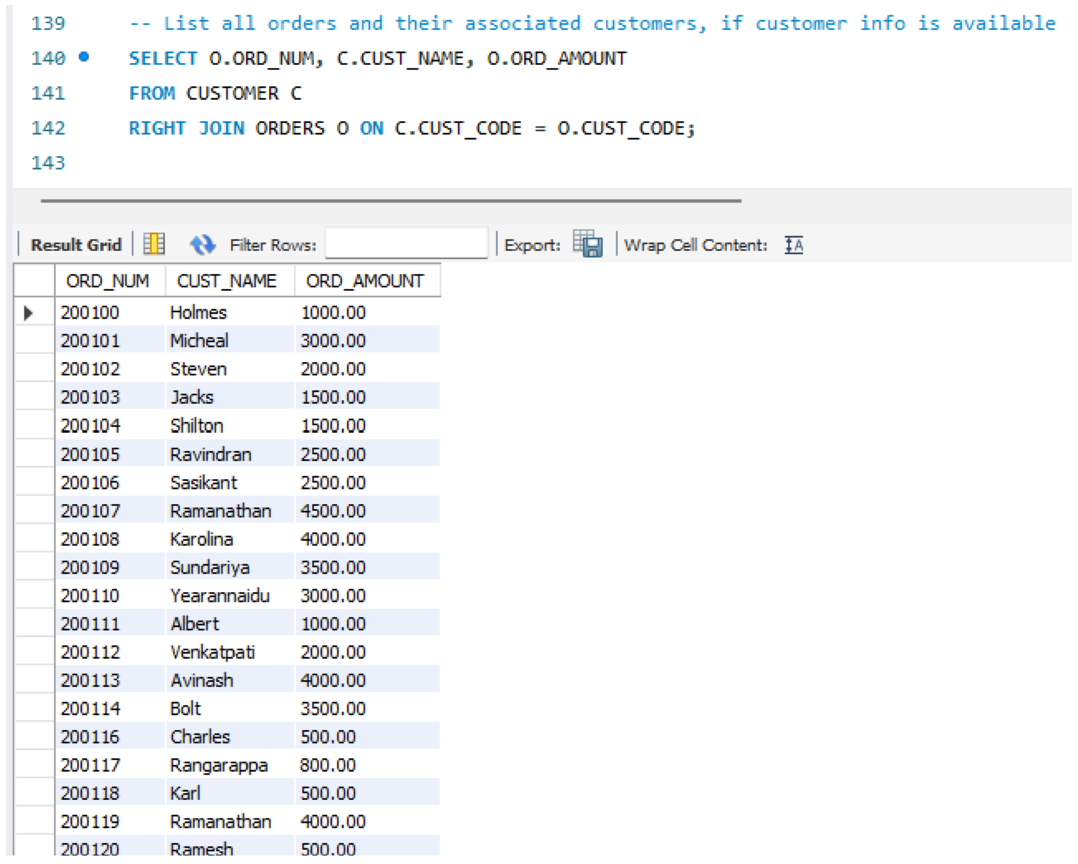
```
-- List all customers and their orders, if any
SELECT C.CUST_NAME, O.ORD_NUM, O.ORD_AMOUNT
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUST_CODE = O.CUST_CODE;
```

CUST_NAME	ORD_NUM	ORD_AMOUNT
Micheal	200101	3000.00
Bolt	200114	3500.00
Martin	200122	2500.00
Winston	200134	4200.00
Sasikant	200106	2500.00
Shilton	200104	1500.00
Ramanathan	200135	2000.00
Ramanathan	200119	4000.00
Ramanathan	200107	4500.00
Karolina	200121	1500.00
Karolina	200108	4000.00
Ramesh	200133	1200.00
Ramesh	200128	3500.00
Ramesh	200120	500.00
Charles	200116	500.00
Sundariya	200109	3500.00
Steven	200131	900.00
Steven	200102	2000.00
Holmes	200100	1000.00
Rangarappa	200117	800.00



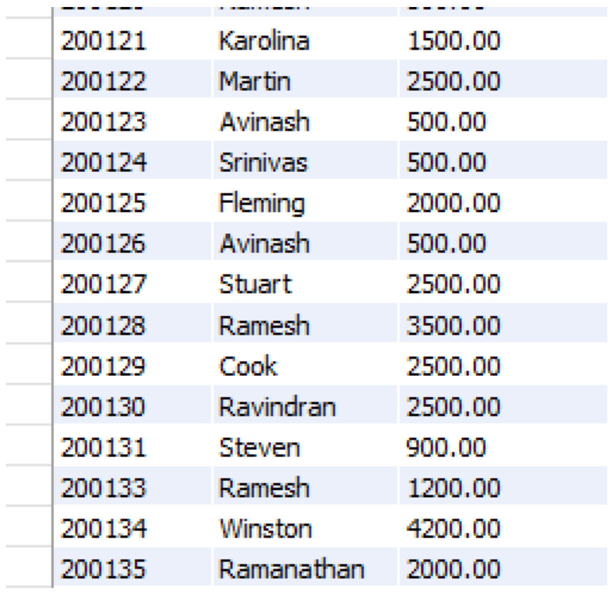
Stuart	200127	2500.00
Venkatpati	200112	2000.00
Srinivas	200124	500.00
Fleming	200125	2000.00
Yearannaidu	200110	3000.00
Albert	200111	1000.00
Jacks	200103	1500.00
Avinash	200126	500.00
Avinash	200123	500.00
Avinash	200113	4000.00
Karl	200118	500.00
Cook	200129	2500.00
Ravindran	200130	2500.00
Ravindran	200105	2500.00

5. Right Join



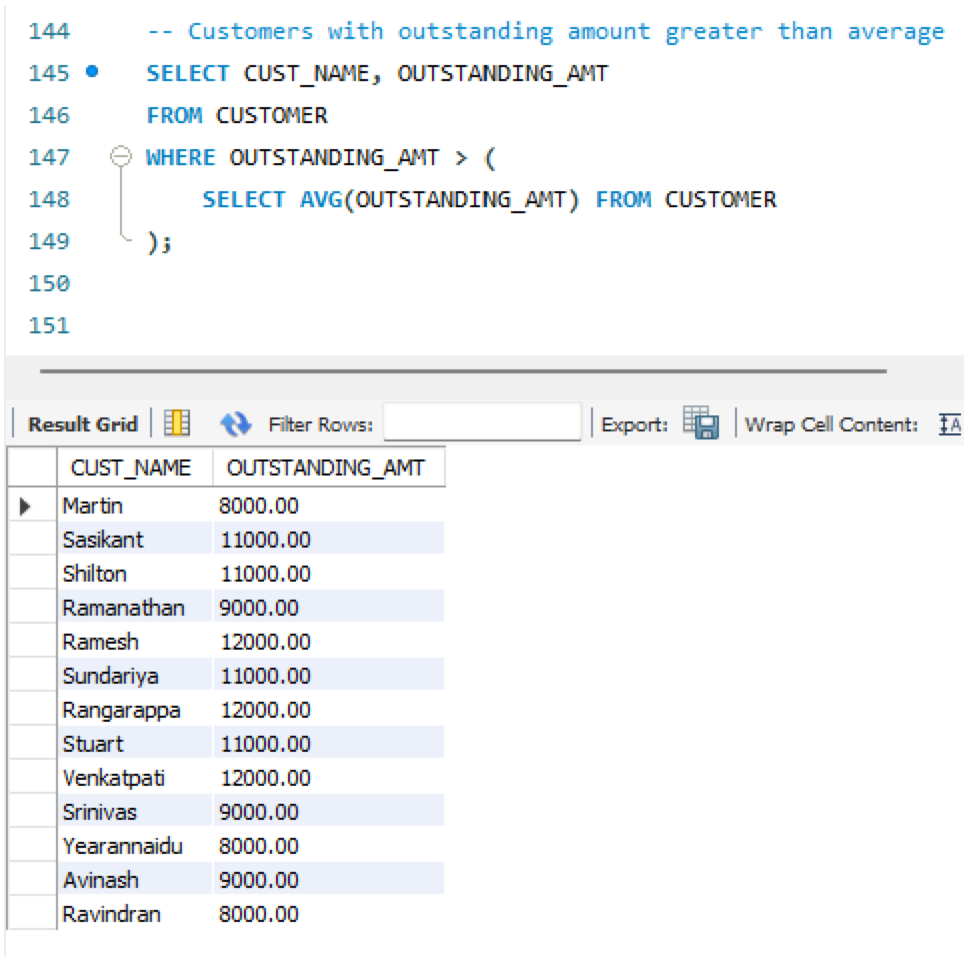
```
-- List all orders and their associated customers, if customer info is available
SELECT O.ORD_NUM, C.CUST_NAME, O.ORD_AMOUNT
FROM CUSTOMER C
RIGHT JOIN ORDERS O ON C.CUST_CODE = O.CUST_CODE;
```

ORD_NUM	CUST_NAME	ORD_AMOUNT
200100	Holmes	1000.00
200101	Micheal	3000.00
200102	Steven	2000.00
200103	Jacks	1500.00
200104	Shilton	1500.00
200105	Ravindran	2500.00
200106	Sasikant	2500.00
200107	Ramanathan	4500.00
200108	Karolina	4000.00
200109	Sundariya	3500.00
200110	Yearannaidu	3000.00
200111	Albert	1000.00
200112	Venkatpati	2000.00
200113	Avinash	4000.00
200114	Bolt	3500.00
200116	Charles	500.00
200117	Rangarappa	800.00
200118	Karl	500.00
200119	Ramanathan	4000.00
200120	Ramesh	500.00



200121	Karolina	1500.00
200122	Martin	2500.00
200123	Avinash	500.00
200124	Srinivas	500.00
200125	Fleming	2000.00
200126	Avinash	500.00
200127	Stuart	2500.00
200128	Ramesh	3500.00
200129	Cook	2500.00
200130	Ravindran	2500.00
200131	Steven	900.00
200133	Ramesh	1200.00
200134	Winston	4200.00
200135	Ramanathan	2000.00

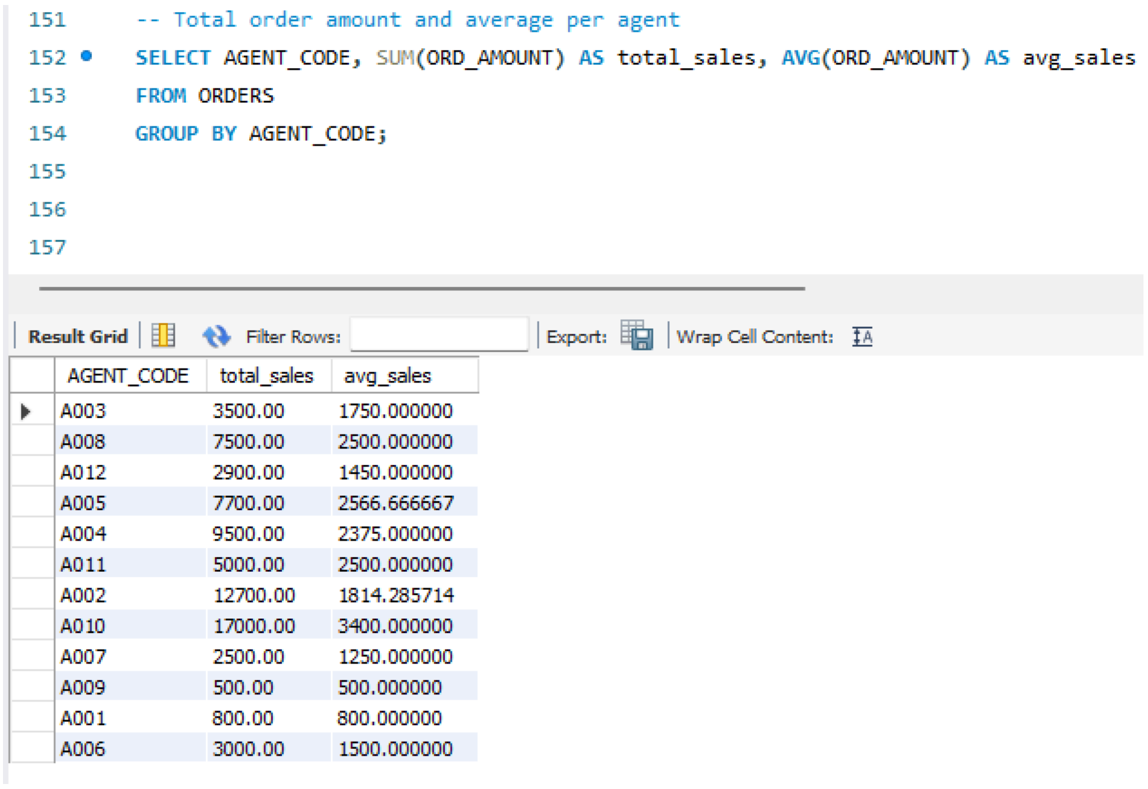
6. Subqueries



```sql
-- Customers with outstanding amount greater than average
SELECT CUST_NAME, OUTSTANDING_AMT
FROM CUSTOMER
WHERE OUTSTANDING_AMT > (
    SELECT AVG(OUTSTANDING_AMT) FROM CUSTOMER
);
```

CUST_NAME	OUTSTANDING_AMT
Martin	8000.00
Sasikant	11000.00
Shilton	11000.00
Ramanathan	9000.00
Ramesh	12000.00
Sundariya	11000.00
Rangarappa	12000.00
Stuart	11000.00
Venkatpati	12000.00
Srinivas	9000.00
Yearannaidu	8000.00
Avinash	9000.00
Ravindran	8000.00

7. Aggregate Function

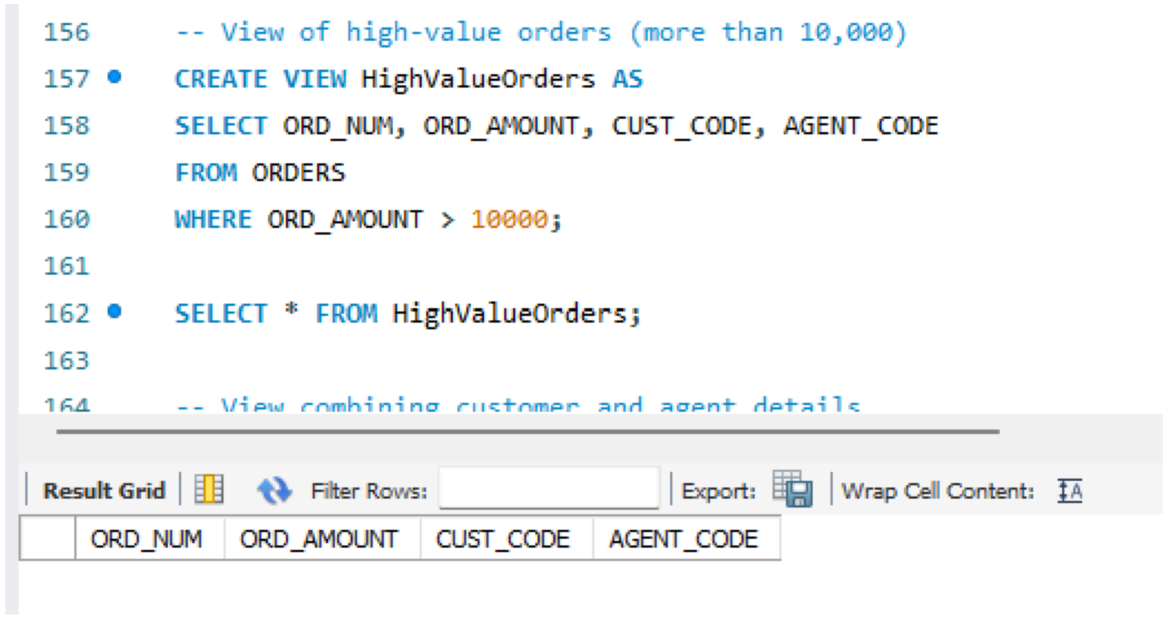


```
-- Total order amount and average per agent
SELECT AGENT_CODE, SUM(ORD_AMOUNT) AS total_sales, AVG(ORD_AMOUNT) AS avg_sales
FROM ORDERS
GROUP BY AGENT_CODE;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AGENT_CODE	total_sales	avg_sales
A003	3500.00	1750.000000
A008	7500.00	2500.000000
A012	2900.00	1450.000000
A005	7700.00	2566.666667
A004	9500.00	2375.000000
A011	5000.00	2500.000000
A002	12700.00	1814.285714
A010	17000.00	3400.000000
A007	2500.00	1250.000000
A009	500.00	500.000000
A001	800.00	800.000000
A006	3000.00	1500.000000

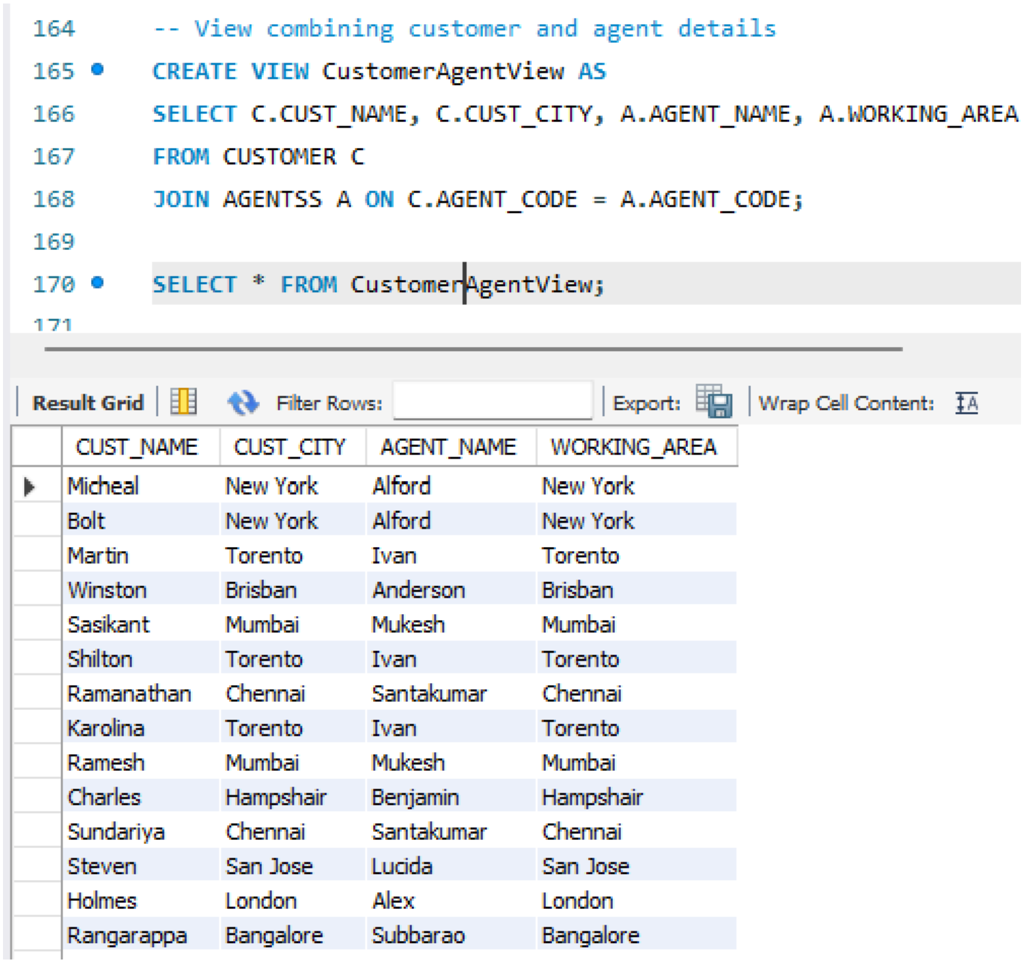
8. Views



```
-- View of high-value orders (more than 10,000)
CREATE VIEW HighValueOrders AS
SELECT ORD_NUM, ORD_AMOUNT, CUST_CODE, AGENT_CODE
FROM ORDERS
WHERE ORD_AMOUNT > 10000;

SELECT * FROM HighValueOrders;

-- View combining customer and agent details
```

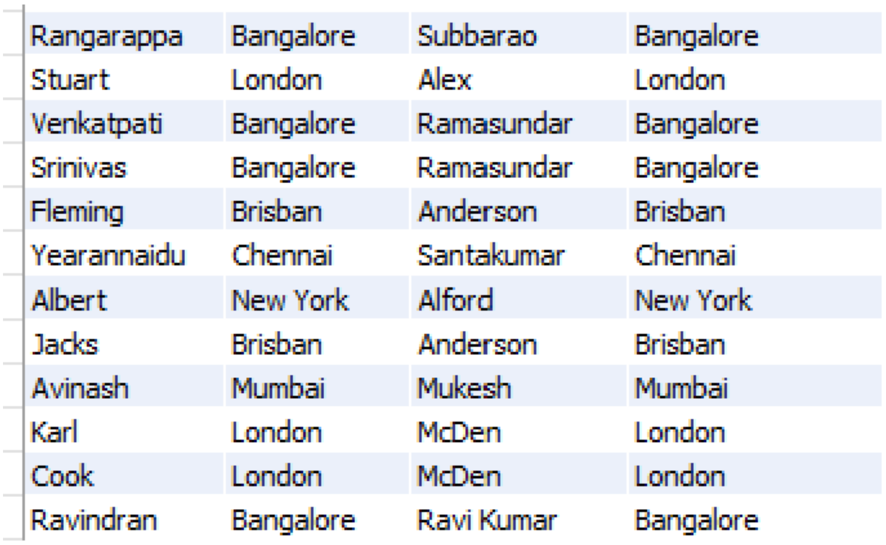
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ORD_NUM	ORD_AMOUNT	CUST_CODE	AGENT_CODE



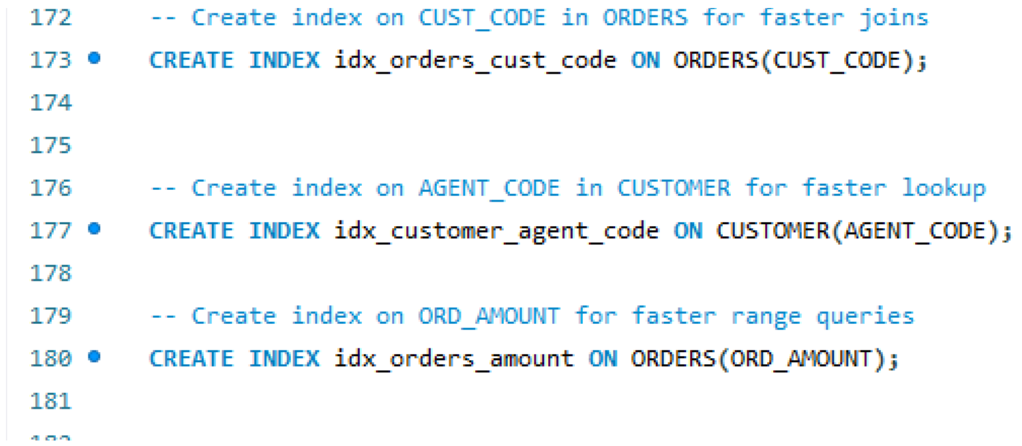
```sql
-- View combining customer and agent details
CREATE VIEW CustomerAgentView AS
SELECT C.CUST_NAME, C.CUST_CITY, A.AGENT_NAME, A.WORKING_AREA
FROM CUSTOMER C
JOIN AGENTSS A ON C.AGENT_CODE = A.AGENT_CODE;

SELECT * FROM CustomerAgentView;
```

CUST_NAME	CUST_CITY	AGENT_NAME	WORKING_AREA
Micheal	New York	Alford	New York
Bolt	New York	Alford	New York
Martin	Torento	Ivan	Torento
Winston	Brisban	Anderson	Brisban
Sasikant	Mumbai	Mukesh	Mumbai
Shilton	Torento	Ivan	Torento
Ramanathan	Chennai	Santakumar	Chennai
Karolina	Torento	Ivan	Torento
Ramesh	Mumbai	Mukesh	Mumbai
Charles	Hampshair	Benjamin	Hampshair
Sundariya	Chennai	Santakumar	Chennai
Steven	San Jose	Lucida	San Jose
Holmes	London	Alex	London
Rangarappa	Bangalore	Subbarao	Bangalore



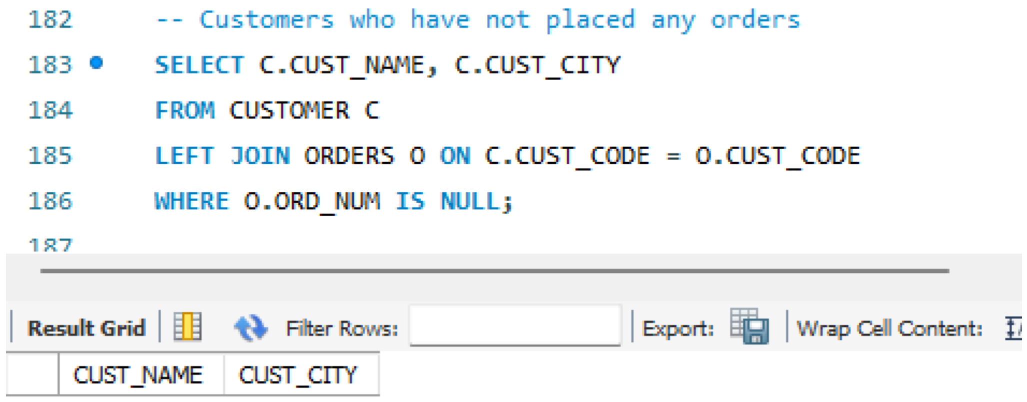
Rangarappa	Bangalore	Subbarao	Bangalore
Stuart	London	Alex	London
Venkatpati	Bangalore	Ramasundar	Bangalore
Srinivas	Bangalore	Ramasundar	Bangalore
Fleming	Brisban	Anderson	Brisban
Yearannaidu	Chennai	Santakumar	Chennai
Albert	New York	Alford	New York
Jacks	Brisban	Anderson	Brisban
Avinash	Mumbai	Mukesh	Mumbai
Karl	London	McDen	London
Cook	London	McDen	London
Ravindran	Bangalore	Ravi Kumar	Bangalore

9.Optimize Queries with Indexes



```sql
-- Create index on CUST_CODE in ORDERS for faster joins
CREATE INDEX idx_orders_cust_code ON ORDERS(CUST_CODE);


-- Create index on AGENT_CODE in CUSTOMER for faster lookup
CREATE INDEX idx_customer_agent_code ON CUSTOMER(AGENT_CODE);

-- Create index on ORD_AMOUNT for faster range queries
CREATE INDEX idx_orders_amount ON ORDERS(ORD_AMOUNT);

```

10. Optimized Left Join to find Customers without any order



```sql
-- Customers who have not placed any orders
SELECT C.CUST_NAME, C.CUST_CITY
FROM CUSTOMER C
LEFT JOIN ORDERS O ON C.CUST_CODE = O.CUST_CODE
WHERE O.ORD_NUM IS NULL;
```

	CUST_NAME	CUST_CITY