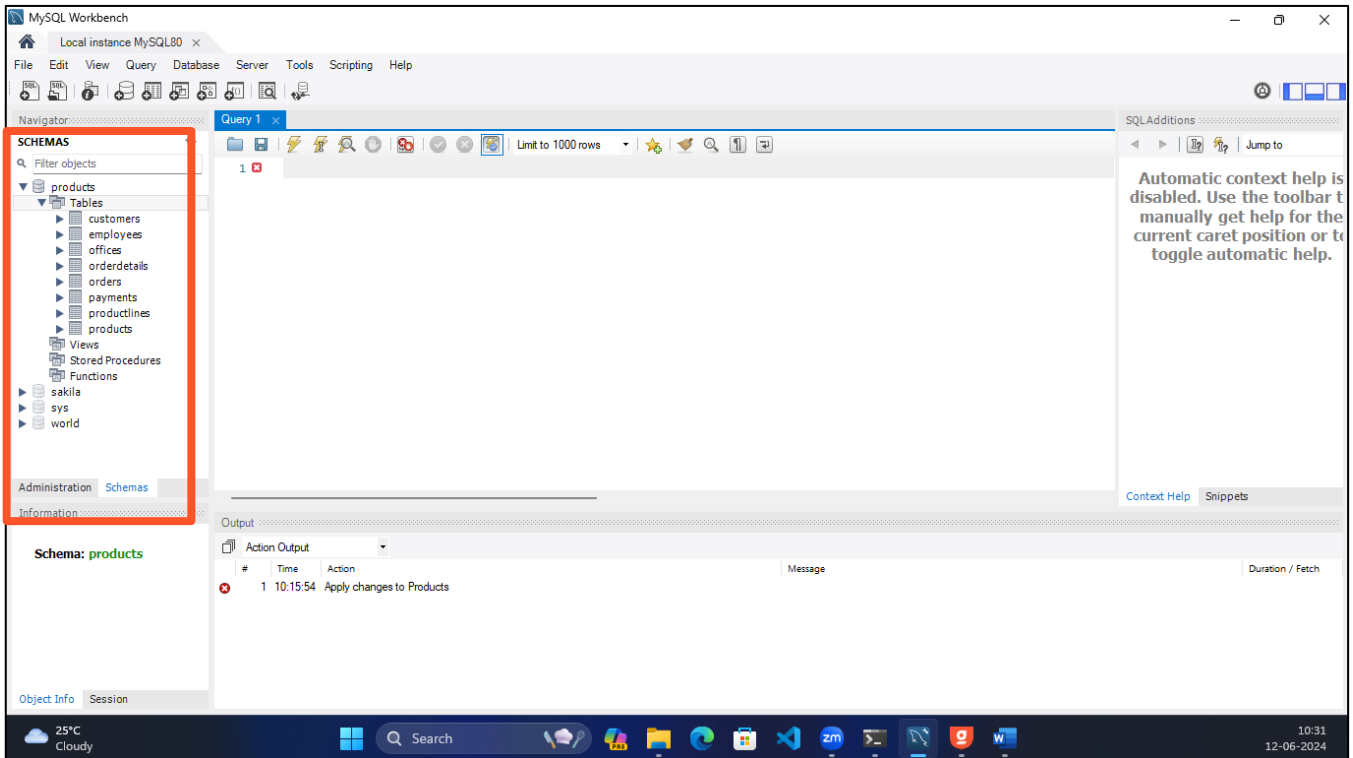


## MySQL Assignment 2

### Using WINDOWS

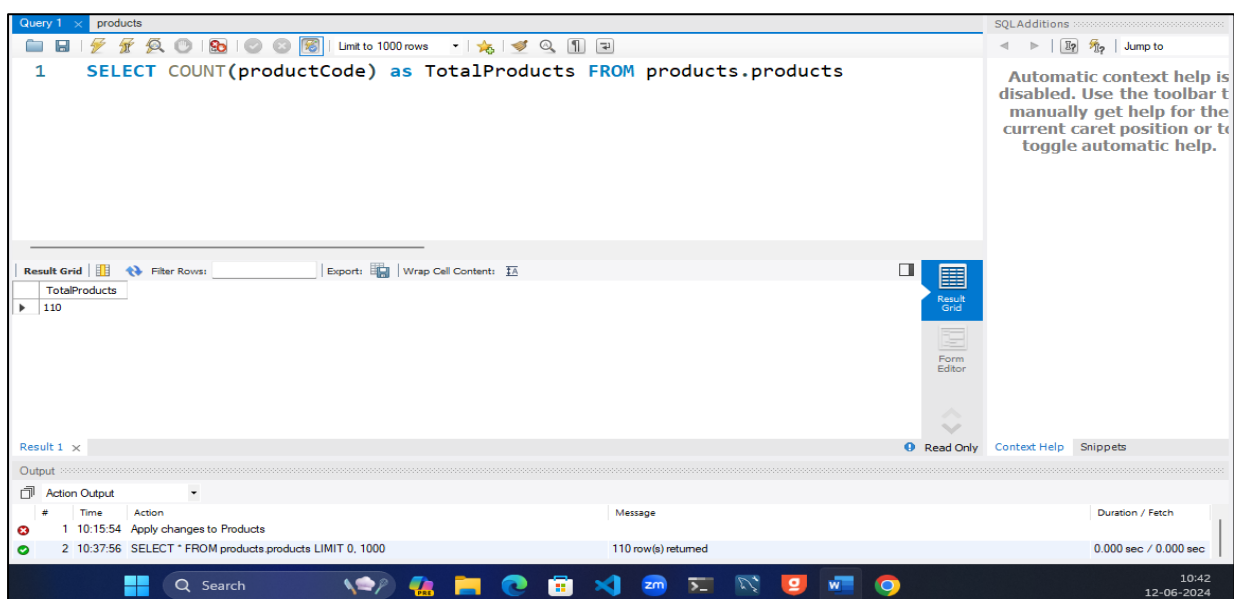
### TABLES CREATED SHOWN IN WORKBENCH



### Execution of all the queries

Q1. Write a query to calculate the total number of products in the database.

```
SELECT COUNT(productCode) as TotalProducts FROM products.products
```



Q2. Write a query to find the average buy price of all products

```
SELECT AVG(buyPrice) as AveragePrice FROM products.products
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the query: `1 SELECT AVG(buyPrice) as AveragePrice FROM products.products`. The results pane on the right shows a single row with the value `54.395182` under the column `AveragePrice`. The bottom pane shows the execution log with two entries: `4 10:44:20 SELECT AVG(buyPrice) as TotalProducts FROM products.products LIMIT 0, 1000` and `5 10:44:37 SELECT AVG(buyPrice) as AveragePrice FROM products.products LIMIT 0, 1000`, both returning 1 row(s).

| Result Grid  |
|--------------|
| AveragePrice |
| 54.395182    |

| # | Time     | Action   | Message           | Duration / Fetch      |
|---|----------|--|-------------------|-----------------------|
| 4 | 10:44:20 | SELECT AVG(buyPrice) as TotalProducts FROM products.products LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 5 | 10:44:37 | SELECT AVG(buyPrice) as AveragePrice FROM products.products LIMIT 0, 1000  | 1 row(s) returned | 0.000 sec / 0.000 sec |

Q3. Write a query to determine the maximum quantity in stock across all products.

```
SELECT MAX(quantityInStock) as MaxQuantity FROM products.products
```

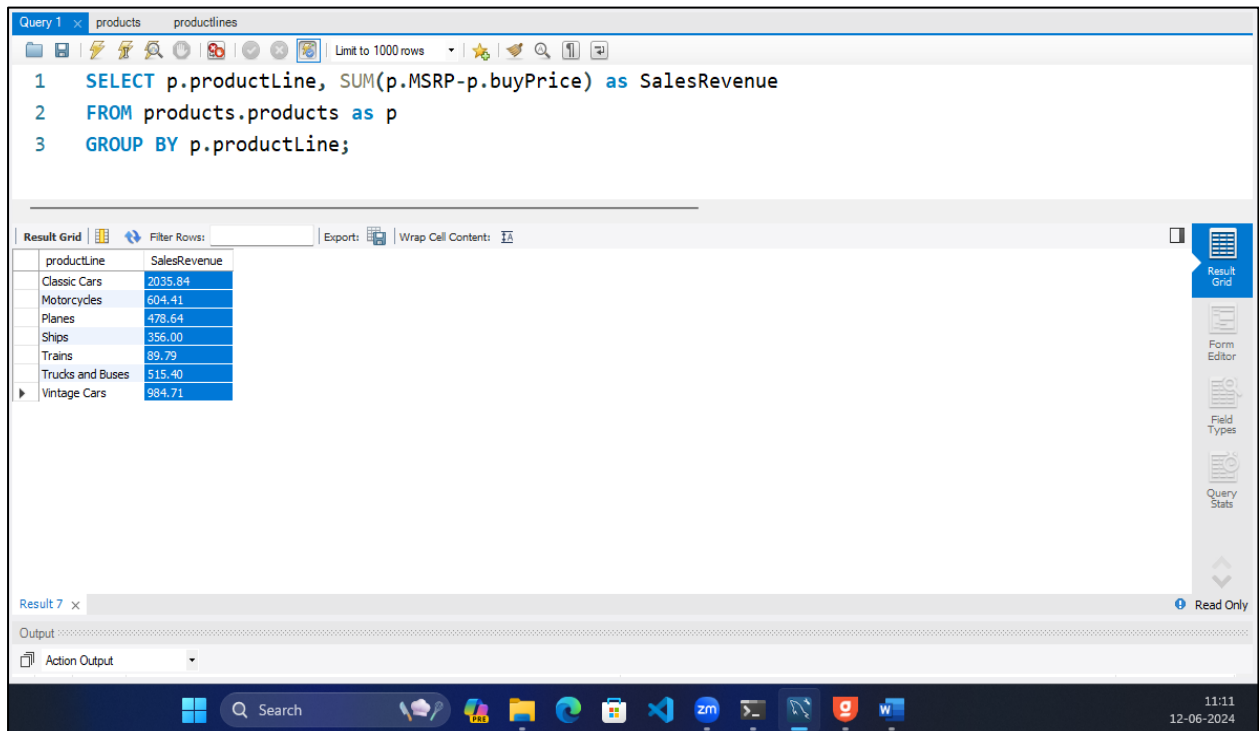
The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the query: `1 SELECT MAX(quantityInStock) as MaxQuantity FROM products.products`. The results pane on the right shows a single row with the value `9997` under the column `MaxQuantity`. The bottom pane shows the execution log with two entries: `6 10:50:52 SELECT MAX(quantityInStock) as MaxQuantity FROM products.products LIMIT 0, 1000` and `7 10:52:02 SELECT * FROM products.productlines LIMIT 0, 1000`, both returning 1 row(s) and 7 row(s) respectively.

| Result Grid |
|-------------|
| MaxQuantity |
| 9997        |

| # | Time     | Action  | Message           | Duration / Fetch      |
|---|----------|---|-------------------|-----------------------|
| 6 | 10:50:52 | SELECT MAX(quantityInStock) as MaxQuantity FROM products.products LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 7 | 10:52:02 | SELECT * FROM products.productlines LIMIT 0, 1000                               | 7 row(s) returned | 0.000 sec / 0.000 sec |

Q4. Write a query to calculate the total sales revenue for each product line.

```
SELECT p.productLine, SUM(p.MSRP-p.buyPrice) as SalesRevenue
FROM products.products as p
GROUP BY p.productLine;
```



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

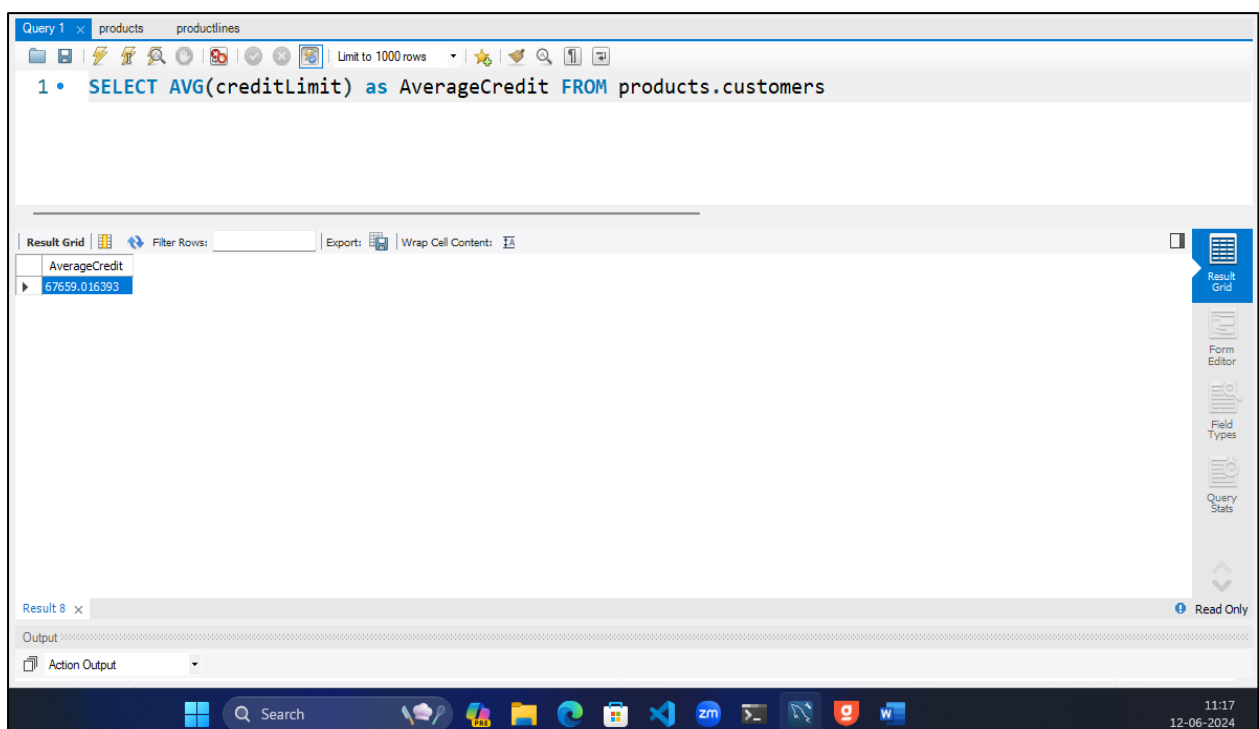
```
1 SELECT p.productLine, SUM(p.MSRP-p.buyPrice) as SalesRevenue
2 FROM products.products as p
3 GROUP BY p.productLine;
```

The results grid displays the following data:

| productLine      | SalesRevenue |
|------------------|--------------|
| Classic Cars     | 2035.84      |
| Motorcycles      | 604.41       |
| Planes           | 478.64       |
| Ships            | 356.00       |
| Trains           | 89.79        |
| Trucks and Buses | 515.40       |
| Vintage Cars     | 984.71       |

Q5. Write a query to determine the average credit limit for all customers.

```
SELECT AVG(creditLimit) as AverageCredit FROM products.customers
```



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results grid. The query editor contains the following SQL code:

```
1 • SELECT AVG(creditLimit) as AverageCredit FROM products.customers
```

The results grid displays the following data:

| AverageCredit |
|---------------|
| 67659.016393  |

Q6. Write a query to find the highest payment amount made by a customer.

```
SELECT MAX(amount) as HighestPayment FROM products.payments
```

The screenshot shows a database query editor with a query window and a results grid. The query window contains the following SQL query:

```
1 • SELECT MAX(amount) as HighestPayment FROM products.payments
```

The results grid shows the following data:

| HighestPayment |
|----------------|
| 120166.58      |

The interface includes a toolbar with various icons for query execution, a sidebar with options like 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats', and a bottom status bar showing the date and time.

Q7. Write a query to calculate the total quantity ordered for each product.

```
SELECT o.productCode, p.productName, SUM(o.quantityOrdered) as TotalQuantity  
FROM products.orderdetails o  
INNER JOIN products.products p WHERE p.productCode = o.productCode  
GROUP BY productCode  
ORDER BY productCode ASC;
```

The screenshot shows a database query editor with a query window and a results grid. The query window contains the following SQL query:

```
1 • SELECT o.productCode, p.productName, SUM(o.quantityOrdered) as TotalQuantity  
2 FROM products.orderdetails o  
3 INNER JOIN products.products p WHERE p.productCode = o.productCode  
4 GROUP BY productCode  
5 ORDER BY productCode ASC;
```

The results grid shows the following data:

| productCode | productName                           | TotalQuantity |
|-------------|---------------------------------------|---------------|
| S10_1678    | 1969 Harley Davidson Ultimate Chopper | 1057          |
| S10_1949    | 1952 Alpine Renault 1300              | 961           |
| S10_2016    | 1996 Moto Guzzi 1100i                 | 999           |
| S10_4698    | 2003 Harley-Davidson Eagle Drag Bike  | 985           |
| S10_4757    | 1972 Alfa Romeo GTA                   | 1030          |
| S10_4962    | 1962 Lancia Delta 16V                 | 932           |
| S12_1099    | 1968 Ford Mustang                     | 933           |
| S12_1108    | 2001 Ferrari Enzo                     | 1019          |
| S12_1666    | 1958 Setra Bus                        | 972           |
| S12_2823    | 2002 Suzuki XREO                      | 1028          |
| S12_3148    | 1969 Corvair Monza                    | 963           |
| S12_3380    | 1968 Dodge Charger                    | 925           |
| S12_3891    | 1969 Ford Falcon                      | 965           |
| S12_3990    | 1970 Plymouth Hemi Cuda               | 900           |

The interface includes a toolbar with various icons for query execution, a sidebar with options like 'Result Grid', 'Form Editor', 'Field Types', and 'Query Stats', and a bottom status bar showing the date and time.

Q8. Write a query to determine the number of employees in each office.

```
SELECT COUNT(employeeNumber) FROM products.employees
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the query: `1 • SELECT COUNT(employeeNumber) FROM products.employees`. Below the query editor, the 'Result Grid' tab is active, displaying a single row with the value 23. The 'Output' pane at the bottom shows the execution log with two entries: a successful query execution at 11:36:45 and a subsequent error at 11:36:49. The Windows taskbar at the bottom shows the system clock as 11:38 on 12-06-2024.

| #  | Time     | Action   | Message            | Duration / Fetch      |
|----|----------|--|--------------------|-----------------------|
| 46 | 11:36:45 | SELECT COUNT(employeeNumber) FROM products.employees LIMIT 0, 1000 | 1 row(s) returned  | 0.015 sec / 0.000 sec |
| 47 | 11:36:49 | SELECT * FROM products.employees LIMIT 0, 1000                     | 23 row(s) returned | 0.000 sec / 0.000 sec |

Q9. Write a query to calculate the average price for each order.

```
SELECT orderNumber,AVG((quantityOrdered*priceEach)) as averagePrice  
FROM products.orderdetails  
GROUP BY orderNumber;
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the query: `1 SELECT orderNumber,AVG((quantityOrdered*priceEach)) as averagePrice  
2 FROM products.orderdetails  
3 GROUP BY orderNumber;`. Below the query editor, the 'Result Grid' tab is active, displaying a list of order numbers and their corresponding average prices. The 'Output' pane at the bottom shows the execution log with two entries: a successful query execution at 11:42:02 and a subsequent error at 11:42:10. The Windows taskbar at the bottom shows the system clock as 11:43 on 12-06-2024.

| orderNumber | averagePrice |
|-------------|--------------|
| 10100       | 2555.957500  |
| 10101       | 2637.252500  |
| 10102       | 2747.390000  |
| 10103       | 3138.684375  |
| 10104       | 3092.784615  |
| 10105       | 3597.280667  |
| 10106       | 2897.322778  |
| 10107       | 2786.577500  |
| 10108       | 3187.576250  |
| 10109       | 4305.523333  |
| 10110       | 3026.605625  |
| 10111       | 2756.308333  |
| 10112       | 3837.470000  |

Q10. Write a query to determine the total sales revenue for each country.

```
SELECT c.country, SUM((od.priceEach * od.quantityOrdered)) as Sales-  
Revenue  
FROM products.orderdetails od  
LEFT JOIN products.orders o  
ON od.orderNumber = o.orderNumber  
RIGHT JOIN products.customers c  
ON c.customerNumber = o.customerNumber  
GROUP BY c.country;
```

The screenshot shows a SQL Server Enterprise Manager window with a query executed. The query is:   
1 SELECT c.country, SUM((od.priceEach \* od.quantityOrdered)) as SalesRevenue  
2 FROM products.orderdetails od  
3 LEFT JOIN products.orders o  
4 ON od.orderNumber = o.orderNumber  
5 RIGHT JOIN products.customers c  
6 ON c.customerNumber = o.customerNumber  
7 GROUP BY c.country;  
The result grid shows the following data:

| country   | SalesRevenue |
|-----------|--------------|
| France    | 1007374.02   |
| USA       | 3273280.05   |
| Australia | 562582.59    |
| Norway    | 270846.30    |
| Poland    | 10033        |
| Germany   | 196470.99    |
| Spain     | 1099389.09   |
| Sweden    | 187638.35    |
| Denmark   | 218994.92    |

The output pane shows two successful actions, both returning 27 rows.

Q11. Write a query to calculate the average quantity in stock for each product line.

```
SELECT productLine, AVG(quantityInStock) as AverageQuantity  
FROM products.products  
GROUP BY productLine;
```

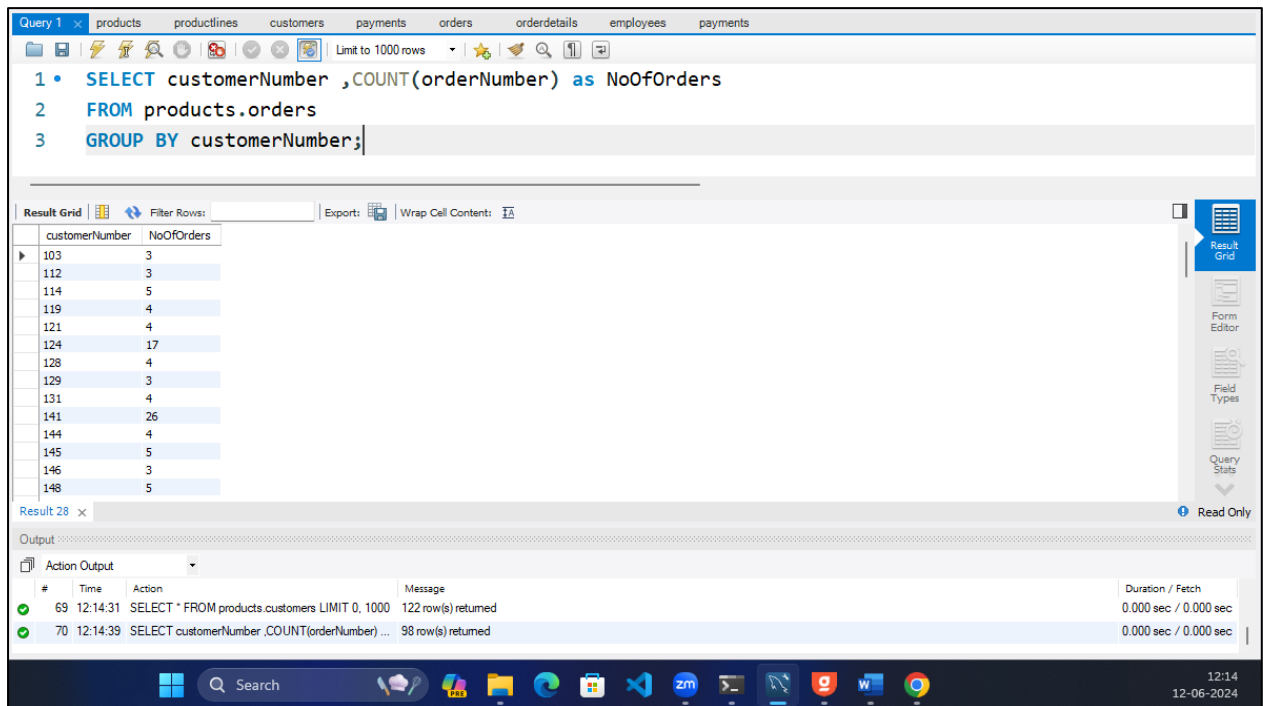
The screenshot shows a SQL Server Enterprise Manager window with a query executed. The query is:   
1 • SELECT productLine, AVG(quantityInStock) as AverageQuantity  
2 FROM products.products  
3 GROUP BY productLine;  
The result grid shows the following data:

| productLine      | AverageQuantity |
|------------------|-----------------|
| Classic Cars     | 5767.9737       |
| Motorcycles      | 5338.5385       |
| Planes           | 5190.5833       |
| Ships            | 2981.4444       |
| Trains           | 5565.3333       |
| Trucks and Buses | 3259.1818       |
| Vintage Cars     | 5203.3333       |

The output pane shows two successful actions, both returning 7 rows.

Q12. Write a query to determine the total number of orders placed by each customer.

```
SELECT customerNumber ,COUNT(orderNumber) as NoOfOrders
FROM products.orders
GROUP BY customerNumber;
```



Query 1: products, productlines, customers, payments, orders, orderdetails, employees, payments

1 • SELECT customerNumber ,COUNT(orderNumber) as NoOfOrders  
2 FROM products.orders  
3 GROUP BY customerNumber;

Result Grid

| customerNumber | NoOfOrders |
|----------------|------------|
| 103            | 3          |
| 112            | 3          |
| 114            | 5          |
| 119            | 4          |
| 121            | 4          |
| 124            | 17         |
| 128            | 4          |
| 129            | 3          |
| 131            | 4          |
| 141            | 26         |
| 144            | 4          |
| 145            | 5          |
| 146            | 3          |
| 148            | 5          |

Result 28 x

Output

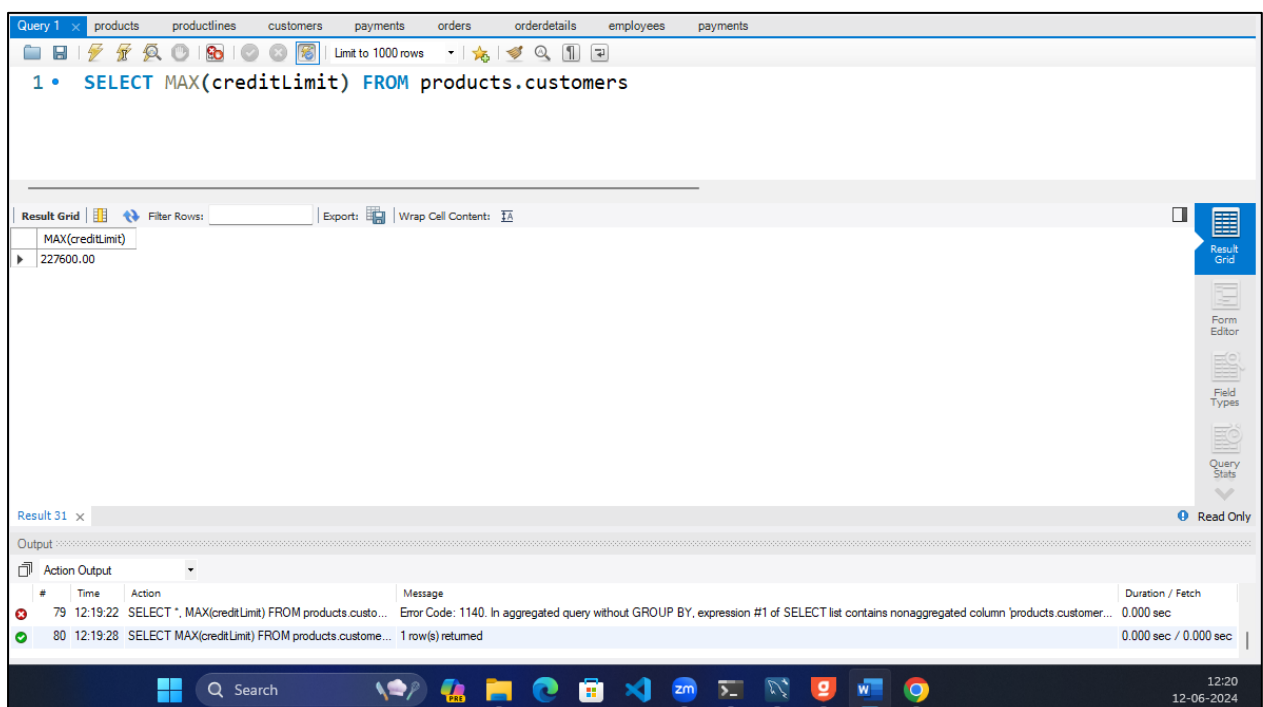
Action Output

| #  | Time     | Action   | Message             | Duration / Fetch      |
|----|----------|--|---------------------|-----------------------|
| 69 | 12:14:31 | SELECT * FROM products.customers LIMIT 0, 1000 | 122 row(s) returned | 0.000 sec / 0.000 sec |
| 70 | 12:14:39 | SELECT customerNumber ,COUNT(orderNumber) ...  | 98 row(s) returned  | 0.000 sec / 0.000 sec |

12:14 12-06-2024

Q13. Write a query to find the maximum credit limit among all customers.

```
SELECT MAX(creditLimit) FROM products.customers
```



Query 1: products, productlines, customers, payments, orders, orderdetails, employees, payments

1 • SELECT MAX(creditLimit) FROM products.customers

Result Grid

| MAX(creditLimit) |
|------------------|
| 227600.00        |

Result 31 x

Output

Action Output

| #  | Time     | Action  | Message  | Duration / Fetch      |
|----|----------|---|--|-----------------------|
| 79 | 12:19:22 | SELECT *, MAX(creditLimit) FROM products.custo... | Error Code: 1140. In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'products.customer... | 0.000 sec             |
| 80 | 12:19:28 | SELECT MAX(creditLimit) FROM products.custome...  | 1 row(s) returned  | 0.000 sec / 0.000 sec |

12:20 12-06-2024

Q14. Write a query to count the number of offices in each country.

```
SELECT country, COUNT(officeCode) FROM products.offices
GROUP BY country;
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the following SQL query:

```
1 • SELECT country, COUNT(officeCode) FROM products.offices
2 GROUP BY country;
```

The Results pane below the query editor displays the output of the query in a grid format:

| country   | COUNT(officeCode) |
|-----------|-------------------|
| USA       | 3                 |
| France    | 1                 |
| Japan     | 1                 |
| Australia | 1                 |
| UK        | 1                 |

The bottom pane shows the Action Output, indicating that the query was executed successfully and returned 5 rows.

Q15. Write a query to calculate the average payment amount for each customer.

```
SELECT customerNumber, AVG(amount) as AverageAmount
FROM products.payments
GROUP BY customerNumber;
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor at the top contains the following SQL query:

```
1 • SELECT customerNumber, AVG(amount) as AverageAmount
2 FROM products.payments
3 GROUP BY customerNumber;
```

The Results pane below the query editor displays the output of the query in a grid format:

| customerNumber | AverageAmount |
|----------------|---------------|
| 103            | 7438.120000   |
| 112            | 26726.993333  |
| 114            | 45146.267500  |
| 119            | 38983.226667  |
| 121            | 26056.197500  |
| 124            | 64909.804444  |
| 128            | 18984.440000  |
| 129            | 22236.853333  |
| 131            | 35879.980000  |
| 141            | 55056.844615  |

The bottom pane shows the Action Output, indicating that the query was executed successfully and returned 98 rows.



Q16. Write a query to determine the number of products in each product line.

```
SELECT productLine ,COUNT(productName) as NumberOfProducts
FROM products.products
GROUP BY productLine;
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL query:

```
1 • SELECT productLine ,COUNT(productName) as NumberOfProducts
2 FROM products.products
3 GROUP BY productLine;
```

The Results pane shows the output of the query in a grid format:

| productLine      | NumberOfProducts |
|------------------|------------------|
| Classic Cars     | 38               |
| Motorcycles      | 13               |
| Planes           | 12               |
| Ships            | 9                |
| Trains           | 3                |
| Trucks and Buses | 11               |
| Vintage Cars     | 24               |

The Output pane shows the execution details of the query:

| #  | Time     | Action   | Message            | Duration / Fetch      |
|----|----------|--|--------------------|-----------------------|
| 85 | 12:25:50 | SELECT customerNumber, AVG(amount) as Avera... | 98 row(s) returned | 0.000 sec / 0.000 sec |
| 86 | 12:29:40 | SELECT productLine ,COUNT(productName) as N... | 7 row(s) returned  | 0.000 sec / 0.000 sec |

Q17. Write a query to count the number of customers in each state.

```
SELECT state, COUNT(customerNumber) as NumberOfCustomers
FROM products.customers
GROUP BY state;
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following SQL query:

```
1 • SELECT state, COUNT(customerNumber) as NumberOfCustomers
2 FROM products.customers
3 GROUP BY state;
```

The Results pane shows the output of the query in a grid format:

| state    | NumberOfCustomers |
|----------|-------------------|
| AL       | 73                |
| NV       | 1                 |
| Victoria | 2                 |
| CA       | 11                |
| NY       | 6                 |
| PA       | 3                 |
| CT       | 4                 |
| MA       | 9                 |
| Osaka    | 1                 |
| BC       | 2                 |

The Output pane shows the execution details of the query:

| #  | Time     | Action   | Message            | Duration / Fetch      |
|----|----------|--|--------------------|-----------------------|
| 86 | 12:29:40 | SELECT productLine ,COUNT(productName) as N... | 7 row(s) returned  | 0.000 sec / 0.000 sec |
| 87 | 12:32:17 | SELECT state, COUNT(customerNumber) as Num...  | 19 row(s) returned | 0.000 sec / 0.000 sec |

Q18. Write a query to find the minimum payment amount among all customers.

```
SELECT MIN(amount) FROM products.payments
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following query:

```
1 • SELECT MIN(amount) FROM products.payments
```

The Results pane shows the result grid with one row:

| MIN(amount) |
|-------------|
| 615.45      |

The Output pane shows the execution log with the following message:

| #   | Time     | Action   | Message   | Duration / Fetch      |
|-----|----------|--|---|-----------------------|
| 101 | 12:43:26 | SELECT p.* FROM products.payments p INNER J... | Error Code: 1055, Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'products.payments.custom... | 0.000 sec             |
| 102 | 12:43:51 | SELECT MIN(amount) FROM products.payments L... | 1 row(s) returned   | 0.000 sec / 0.000 sec |

Q19. Write a query to calculate the average sales revenue per order.

```
SELECT orderNumber, AVG((priceEach*quantityOrdered))  
FROM products.orderdetails  
GROUP BY orderNumber
```

The screenshot shows the SQL Server Enterprise Manager interface. The query editor displays the following query:

```
1 SELECT orderNumber, AVG((priceEach*quantityOrdered))  
2 FROM products.orderdetails  
3 GROUP BY orderNumber
```

The Results pane shows the result grid with 10 rows:

| orderNumber | AVG((priceEach*quantityOrdered)) |
|-------------|----------------------------------|
| 10100       | 2555.957500                      |
| 10101       | 2637.252500                      |
| 10102       | 2747.390000                      |
| 10103       | 3138.684375                      |
| 10104       | 3092.784615                      |
| 10105       | 3597.280667                      |
| 10106       | 2897.322778                      |
| 10107       | 2786.577500                      |
| 10108       | 3187.576250                      |
| 10109       | 4305.523333                      |

The Output pane shows the execution log with the following message:

| #   | Time     | Action   | Message             | Duration / Fetch      |
|-----|----------|--|---------------------|-----------------------|
| 102 | 12:43:51 | SELECT MIN(amount) FROM products.payments L...   | 1 row(s) returned   | 0.000 sec / 0.000 sec |
| 103 | 12:53:39 | SELECT orderNumber, AVG((priceEach*quantityOr... | 326 row(s) returned | 0.000 sec / 0.000 sec |

Q20. Write a query to determine the total quantity ordered for each product line.

```
SELECT p.productLine, SUM(o.quantityOrdered) FROM products.orderde-  
tails o  
INNER JOIN products.products p  
ON o.productCode = p.productCode  
GROUP BY p.productLine;
```

The screenshot shows a database query editor with a query window and a results grid. The query is as follows:

```
1 • SELECT p.productLine, SUM(o.quantityOrdered) FROM products.orderde-  
tails o  
2 INNER JOIN products.products p  
3 ON o.productCode = p.productCode  
4 GROUP BY p.productLine;
```

The results grid displays the following data:

| productLine      | SUM(o.quantityOrdered) |
|------------------|------------------------|
| Classic Cars     | 35582                  |
| Motorcycles      | 12778                  |
| Planes           | 11872                  |
| Ships            | 8532                   |
| Trains           | 2818                   |
| Trucks and Buses | 11001                  |
| Vintage Cars     | 22933                  |

The bottom of the screenshot shows the 'Output' window with the following log entries:

| #   | Time     | Action  | Message             | Duration / Fetch      |
|-----|----------|---|---------------------|-----------------------|
| 104 | 12:57:15 | SELECT SUM(quantityOrdered) FROM products.or...   | 109 row(s) returned | 0.016 sec / 0.000 sec |
| 105 | 12:59:01 | SELECT p.productLine, SUM(o.quantityOrdered) F... | 7 row(s) returned   | 0.015 sec / 0.000 sec |