**Task 3.3 Using aggregate functions**

**Product table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ProductID** | **ProductName** | **SupplierID** | **CategoryID** | **Unit** | **Price** |
| 1 | Chais | 1 | 1 | 10 boxes x 20 bags | 18 |
| 2 | Chang | 1 | 1 | 24 - 12 oz bottles | 19 |
| 3 | Aniseed Syrup | 1 | 2 | 12 - 550 ml bottles | 10 |
| 4 | Chef Anton's Cajun Seasoning | 2 | 2 | 48 - 6 oz jars | 22 |
| 5 | Chef Anton's Gumbo Mix | 2 | 2 | 36 boxes | 21.35 |

1. Find the lowest price in the price column.
2. Find the highest price in the price column.
3. Find the total number of rows in the product table.(use count());
4. Find the number of products where the ProductName is not null:
5. Find the number of products where Price is higher than 20:
6. How many different prices are there in the Products table(use didtinct keyword)
7. eturn the number of records for each category in the Products table:
8. To give the column a descriptive name, use the AS keyword (alias name for column
9. Example: SELECT MIN(Price) AS smallestprice  
   FROM Products;
10. Return the smallest price for each category in the Products table(use group by)

**Order table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderDetailID** | **OrderID** | **ProductID** | **Quantity** |
| 1 | 10248 | 11 | 12 |
| 2 | 10248 | 42 | 10 |
| 3 | 10248 | 72 | 5 |
| 4 | 10249 | 14 | 9 |
| 5 | 10249 | 51 | 40 |

1. Return the sum of all Quantity fields in the OrderDetails table
2. Return the sum of the Quantity field for the product with ProductID 11
3. Use an expression inside the SUM() function
4. Join OrderDetails with Products, and use SUM() to find the total amount
5. Find the average price of all products
6. Return the average price of products in category 1
7. Return all products with a higher price than the average price