# SQL Assignment

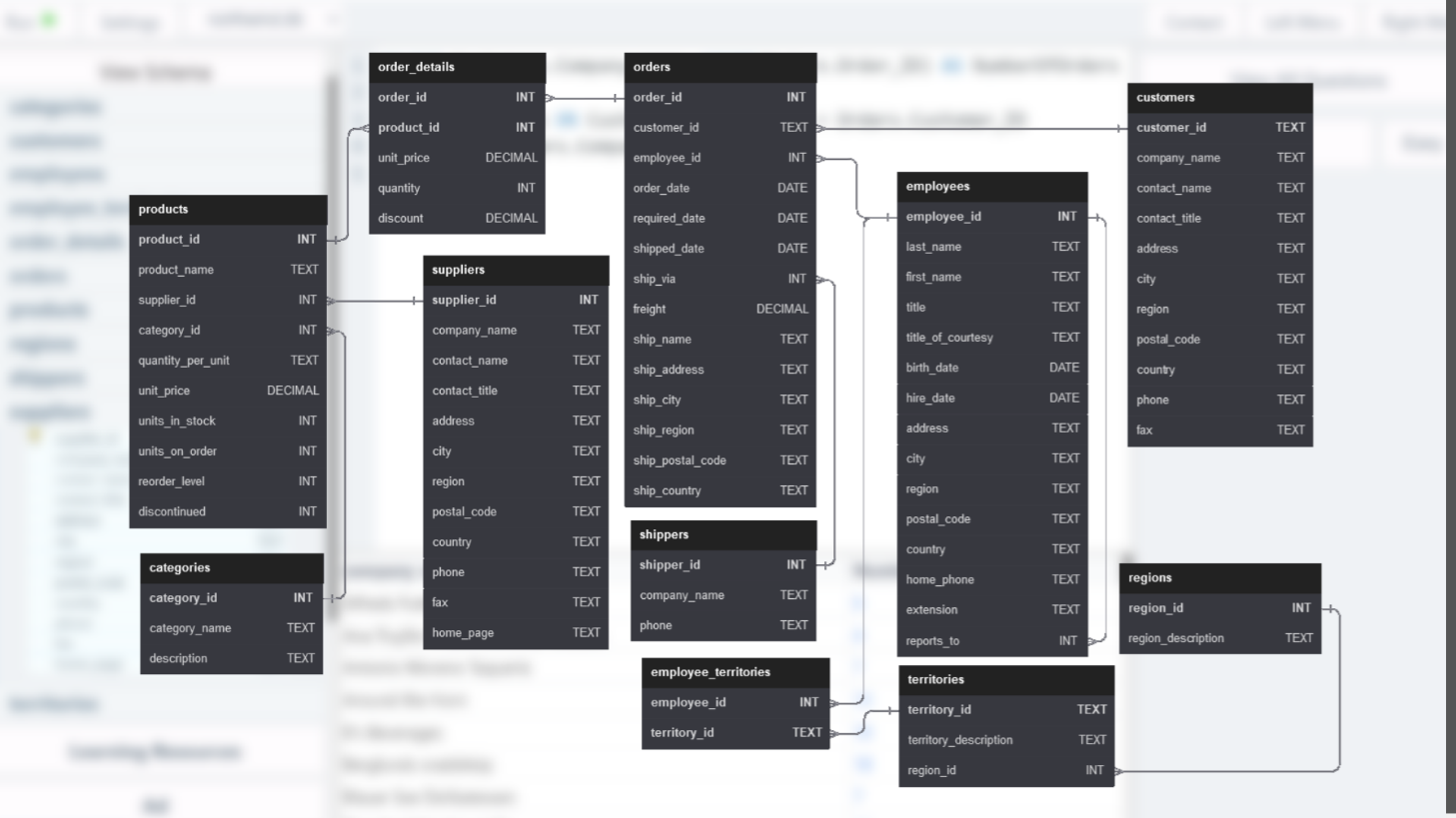
## Link to Practice

[**https://www.sql-practice.com/**](https://www.sql-practice.com/)

## DB To Select

**Northwind.db**

## Schema



## Questions

1. Get the total number of orders placed by each customer

🡪 SELECT customer\_id, count(order\_id) AS no\_of\_orders FROM orders GROUP BY customer\_id;

1. Find all suppliers who provide products in the ‘Seafood’ category

🡪 SELECT s.supplier\_id FROM products p

JOIN categories c ON p.category\_id =c.category\_id

JOIN suppliers s ON p.supplier\_id = s.supplier\_id

WHERE category\_name = 'Seafood';

1. Get the total quantity of each product sold

🡪 SELECT product\_id , SUM(quantity) FROM order\_details GROUP By product\_id;

1. Find the total sales (Quantity\*Unit\_Price) for each category of products

🡪 SELECT category\_id, SUM(quantity\_per\_unit \* unit\_price) AS total\_sales

FROM products GROUP BY category\_id;

1. List the employees and the number of orders each employee has taken

🡪 SELECT employee\_id , COUNT(order\_id) AS no\_of\_orders

FROM orders GROUP BY employee\_id;

1. Get the customers who have placed more than 10 orders

🡪 SELECT customer\_id , COUNT(order\_id) AS total\_order FROM orders

GROUP BY customer\_id HAVING total\_order>10 ;

1. Get the top 5 most sold products

🡪 SELECT product\_id, sum(quantity ) FROM order\_details GROUP BY product\_id ORDER BY quantity DESC LIMIT 5 ;

1. Find the products that have never been ordered

🡪 SELECT p.product\_id

FROM products p

LEFT JOIN order\_details o ON p.product\_id=o.product\_id

WHERE order\_id is null;

1. Find the customers who have not placed any orders

🡪 SELECT c.customer\_id , order\_id FROM customers c

LEFT JOIN orders o ON c.customer\_id = o.customer\_id

WHERE order\_id Is null;

1. List all ‘Orders’ with ‘Customer’ details and ‘Employee’ who processed it

🡪 SELECT order\_id , c.customer\_id, company\_name, contact\_name, c.address, e.employee\_id

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

JOIN employees e ON o.employee\_id = e.employee\_id

WHERE e.employee\_id IS NOT null;

1. Calculate the average product price by category

🡪 SELECT category\_id, AVG(unit\_price) as average FROM products GROUP BY category\_id;

1. Find the total revenue generated by each employee

🡪 SELECT employee\_id, SUM(quantity\* unit\_price) AS revenue FROM order\_details od

JOIN orders o on od.order\_id = o.order\_id

GROUP BY employee\_id;

1. List all orders shipped to ‘Germany’

🡪 SELECT order\_id FROM orders WHERE ship\_country='Germany';

1. Find the most expensive product in each category

🡪 SELECT category\_id , MAX(unit\_price) FROM products GROUP BY category\_id;

1. Find the total revenue for the year 2016

🡪 SELECT YEAR(order\_date) AS year\_order, SUM(quantity\*unit\_price) AS revenue

FROM order\_details od

JOIN orders o ON od.order\_id = o.order\_id

WHERE year\_order = 2016;

1. List all products that are discontinued

🡪 SELECT product\_id FROM products WHERE discontinued IS NOT '0';

1. List all the distinct countries to which orders have been shipped

🡪 SELECT DISTINCT(ship\_country) FROM orders;

1. Find all employees who report to ‘Andrew Fuller’

🡪 SELECT employee\_id FROM employees WHERE reports\_to = 2;

1. Find the customers who have spent more than $5000 in total

🡪 SELECT customer\_id, SUM(unit\_price\* quantity ) AS sales

FROM order\_details od

JOIN orders o ON od.order\_id = o.order\_id

GROUP BY customer\_id HAVING sales> 5000;

1. List the top 5 employees who have processed the most orders

🡪 SELECT employee\_id , COUNT(order\_id) AS od

FROM orders

GROUP BY employee\_id

ORDER BY od DESC LIMIT 5 ;

1. Get the list of customers who have ordered ‘Chai’ product

🡪 SELECT customer\_id FROM order\_details od

JOIN orders o ON o.order\_id = od.order\_id

JOIN products p ON od.product\_id = p.product\_id

WHERE product\_name = 'Chai';

1. Get the employees who have processed orders for ‘Chai’ product

🡪 SELECT employee\_id

FROM order\_details od

JOIN orders o ON o.order\_id = od.order\_id

JOIN products p ON od.product\_id = p.product\_id

WHERE product\_name = 'Chai';

1. Find the most common shipping country

🡪 SELECT ship\_country , COUNT(order\_id) AS total\_shipping FROM orders

GROUP BY ship\_country ORDER BY total\_shipping DESC LIMIT 1;

1. Find the order with the highest total cost

🡪 SELECT order\_id , SUM((unit\_price\* quantity) - (unit\_price\* quantity\*discount) )AS total\_cost

FROM order\_details

JOIN products p on p.product\_id=od.product\_id

ORDER BY total\_cost desc LIMIT 1;

1. Find the employees who have processed more than 100 orders

🡪 SELECT employee\_id , COUNT(order\_id) AS total\_orders FROM orders GROUP BY employee\_id HAVING total\_orders>100 ;

1. Find the customer who has ordered the most ‘Chai’ product

🡪 SELECT customer\_id , COUNT(o.order\_id) AS no\_orders

FROM order\_details od

JOIN orders o ON o.order\_id = od.order\_id

JOIN products p ON od.product\_id = p.product\_id

WHERE product\_name = 'Chai'

GROUP BY customer\_id ORDER BY no\_orders DESC LIMIT 1;

1. Find the average quantity of products ordered in each order

🡪 SELECT order\_id, SUM(quantity) AS average FROM order\_details GROUP BY order\_id;

1. Find the top 3 most popular categories of products ordered

🡪 SELECT category\_id, COUNT(order\_id) AS total\_orders FROM products p

JOIN order\_details od ON p.product\_id=od.product\_id

GROUP BY category\_id ORDER BY total\_orders desc limit 3;

1. Find the month in the year 2016 with the highest total sales

🡪 SELECT MONTH(order\_date) AS month\_order, SUM(quantity\*unit\_price) AS sales

FROM order\_details od

JOIN orders o ON od.order\_id = o.order\_id WHERE YEAR(order\_date) IS 2016

ORDER BY sales DESC LIMIT 1;

1. Find the employee who processed the most orders in August 2016

🡪 SELECT employee\_id , COUNT(order\_id) AS total\_orders FROM orders

WHERE order\_date BETWEEN '2016-08-01' AND '2016-08-31'

GROUP BY employee\_id order by total\_orders DESC LIMIT 1;

1. Find the top 3 customers who have ordered the most products

🡪 SELECT customer\_id , COUNT(order\_id) AS total\_orders

FROM orders GROUP BY customer\_id ORDER BY total\_orders DESC LIMIT 3;

1. Find the employees who have not processed any orders

🡪 SELECT e.employee\_id, o.order\_id FROM employees e

LEFT join orders o ON e.employee\_id = o.employee\_id

WHERE o.order\_id IS NULL;

1. Find the suppliers who supply the top 5 most sold products

🡪 SELECT supplier\_id FROM order\_details od

JOIN products p ON p.product\_id = od.product\_id

GROUP BY od.product\_id ORDER BY quantity DESC LIMIT 5;

1. Find the customers who have ordered products from all categories

🡪 SELECT customer\_id FROM order\_details od

JOIN orders o ON o.order\_id = od.order\_id

JOIN products p ON p.product\_id = od.product\_id

GROUP BY o.order\_id HAVING COUNT (DISTINCT(category\_id))>7;

1. Find the total sales for each year

🡪 SELECT YEAR(order\_date ) as order\_year , SUM(quantity\*unit\_price) as sales

FROM order\_details od JOIN orders o ON od.order\_id = o.order\_id

GROUP BY YEAR(order\_date);

36. classify customers based on their total order amounts such that total order amounts > 5000 should be classified as ‘High Value’, if > 1000 then as ‘Medium Value’ and otherwise as ‘Low Value’

Here is the sample output

A screenshot of a computer

Description automatically generated with low confidence

🡪SELECT company\_name, SUM(unit\_price\*quantity) as sales,

( CASE WHEN SUM(unit\_price\*quantity) > 5000 THEN 'High Value'

WHEN SUM(unit\_price\*quantity) > 1000 THEN 'Medium Value' ELSE 'Low Value' END ) AS CustomerClass

FROM orders o

JOIN order\_details od ON od.order\_id=o.order\_id

JOIN customers c ON c.customer\_id= o.customer\_id

GROUP BY o.customer\_id;

37. classify products based on their sales volume such that TotalQuantity > 1000 Then SalesCategory as ‘High Sales’. If TotalQuantity>500 Then ‘Medium Sales’ and else ‘Lower Sales’

A screenshot of a computer

Description automatically generated with low confidence

🡪SELECT product\_name, SUM(quantity) AS TotalOrderedAmount,

( CASE WHEN SUM(quantity) > 1000 THEN 'High Sales'

WHEN SUM(quantity) > 500 THEN 'Medium Sales'

ELSE 'Low Sales' END )

AS SalesCategory

FROM order\_details od

JOIN products p ON p.product\_id= od.product\_id

GROUP By product\_name;

38. classify employees based on the number of orders they have processed such that NumberOfOrders > 100 Then PerformanceCategory as ‘High Performing’. If NumberOfOrders>50 Then ‘Medium Performing’ and else ‘Lower Performing’

Here is the sample output

A screenshot of a number of orders

Description automatically generated with medium confidence

🡪SELECT first\_name, last\_name, COUNT(order\_id) AS NumberOfOrders ,

( CASE WHEN COUNT(order\_id) > 100 THEN 'High Performing'

WHEN COUNT(order\_id)> 50 THEN 'Medium Performing'

ELSE 'Low Performing' END )

AS PerformanceCategory

FROM orders o

JOIN employees e ON e.employee\_id=o.employee\_id

GROUP BY e.employee\_id

ORDER BY first\_name;