SQL queries assignment

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

SELECT

Customer\_ID,

COUNT(\*) AS TotalOrders

FROM

Orders

GROUP BY

Customer\_ID

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

SELECT DISTINCT

s.Supplier\_ID,

s.Supplier\_Name

FROM

Suppliers s

JOIN Products p ON s.Supplier\_ID = p.Supplier\_ID

JOIN Categories c ON p.Category\_ID = c.Category\_ID

WHERE

c.Category\_Name = 'Seafood'

3. Get the total quantity of each product sold

SELECT

p.Product\_ID,

p.Product\_Name,

SUM(od.Quantity) AS TotalQuantity

FROM

Products p

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

p.Product\_ID,

p.Product\_Name

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

SELECT

c.Category\_ID,

c.Category\_Name,

SUM(od.Quantity \* od.Unit\_Price) AS TotalSales

FROM

Categories c

JOIN Products p ON c.CategoryID = p.CategoryID

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

c.Category\_ID,

c.Category\_Name

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

SELECT

e.Employee\_ID,

e.First\_Name,

e.Last\_Name,

COUNT(o.Order\_ID) AS NumberOfOrders

FROM

Employees e

LEFT JOIN Orders o ON e.Employee\_ID = o.Employee\_ID

GROUP BY

e.Employee\_ID,

e.First\_Name,

e.Last\_Name

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

SELECT

c.Customer\_ID,

c.Company\_Name,

COUNT(o.Order\_ID) AS TotalOrders

FROM

Customers c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

GROUP BY

c.Customer\_ID,

c.Company\_Name

HAVING

COUNT(o.Order\_ID) > 10

7. Get the top 5 most sold products

SELECT

p.Product\_ID,

p.Product\_Name,

SUM(od.Quantity) AS TotalQuantity

FROM

Products p

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

p.Product\_ID,

p.Product\_Name

ORDER BY

TotalQuantity DESC

LIMIT 5

8. Find the products that have never been ordered

SELECT

p.ProductID,

p.ProductName

FROM

Products p

LEFT JOIN OrderDetails od ON p.ProductID = od.ProductID

WHERE

od.ProductID IS NULL

9. Find the customers who have not placed any orders

SELECT

c.CustomerID,

c.CompanyName

FROM

Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

WHERE

o.CustomerID IS NULL

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

SELECT

o.Order\_ID,

c.Customer\_ID,

c.Company\_Name,

e.Employee\_ID,

CONCAT(e.First\_Name, ' ', e.Last\_Name) AS Employee\_Name

FROM

Orders o

JOIN Customers c ON o.Customer\_ID = c.Customer\_ID

JOIN Employees e ON o.Employee\_ID = e.Employee\_ID

11. Calculate the average product price by category

select categories.category\_id, categories.category\_name,

avg(products.unit\_price)

from products

INNER JOIN categories ON products.category\_id= categories.category\_id

GROUP BY

categories.category\_id,

categories.category\_name;

12. Find the total revenue generated by each employee

SELECT

employees.employee\_id,

CONCAT(employees.first\_name, ' ', employees.last\_name) AS employee\_name,

SUM(order\_details.unit\_price \* order\_details.quantity \* (1 - order\_details.discount)) AS TotalRevenue

FROM

Employees

INNER JOIN orders ON employees.employee\_id = orders.employee\_id

INNER JOIN order\_details ON orders.order\_id = order\_details.order\_id

GROUP BY

employees.employee\_id,

employee\_name;

13. .List all orders shipped to ‘Germany’

select order\_id, ship\_country

from orders

where ship\_country = 'Germany';

14. Find the most expensive product in each category

Select products.product\_name, products.product\_id,

categories.category\_name, categories.category\_id,

max(products.unit\_price)

from products

inner join categories on products.category\_id = categories.category\_id

group by categories.category\_name

15. Find the total revenue for the year 2016

select

sum(order\_details.unit\_price \* order\_details.quantity \* (1- order\_details.discount)) as totalrevenue

from

orders

inner join order\_details on orders.order\_id = order\_details.order\_id

where

year (orders.order\_date) == 2016;

16. List all products that are discontinued

Select product\_name, discontinued

from products

where discontinued is '0'

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

select distinct ship\_country

from orders

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

Select last\_name, first\_name, employee\_id

from employees

where reports\_to = (

SELECT

employee\_id

FROM

employees

WHERE

First\_Name = 'Andrew' AND Last\_Name = 'Fuller'

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

SELECT

C.Customer\_ID,

C.Contact\_Name,

SUM(OD.Unit\_Price \* OD.Quantity) AS TotalSpent

FROM

Customers AS C

INNER JOIN Orders AS O ON C.Customer\_ID = O.Customer\_ID

INNER JOIN Order\_Details AS OD ON O.Order\_ID = OD.Order\_ID

GROUP BY

C.Customer\_ID,

C.Contact\_Name

HAVING

SUM(OD.Unit\_Price \* OD.Quantity) > 5000

ORDER BY

TotalSpent DESC;

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

SELECT

Employees.Employee\_ID,

CONCAT(Employees.First\_Name, ' ', Employees.Last\_Name) AS EmployeeName,

COUNT(Orders.Order\_ID) AS OrderCount

FROM

Employees

INNER JOIN Orders ON Employees.Employee\_ID = Orders.Employee\_ID

GROUP BY

Employees.Employee\_ID,

EmployeeName

ORDER BY

OrderCount DESC

LIMIT 5;

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

SELECT

Customers.Customer\_ID,

Customers.Company\_Name

FROM

Customers

INNER JOIN Orders ON Customers.Customer\_ID = Orders.Customer\_ID

INNER JOIN Order\_Details ON Orders.Order\_ID = Order\_Details.Order\_ID

INNER JOIN Products ON Order\_Details.Product\_ID = Products.Product\_ID

WHERE

Products.Product\_Name = 'Chai';

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

SELECT DISTINCT

Employees.Employee\_ID,

CONCAT(Employees.First\_Name, ' ', Employees.Last\_Name) AS EmployeeName

FROM

Employees

INNER JOIN Orders ON Employees.Employee\_ID = Orders.Employee\_ID

INNER JOIN Order\_Details ON Orders.Order\_ID = Order\_Details.Order\_ID

INNER JOIN Products ON Order\_Details.Product\_ID = Products.Product\_ID

WHERE

Products.Product\_Name = 'Chai';

23. Find the most common shipping country

SELECT

Ship\_Country,

COUNT(\*) AS Order\_Count

FROM

Orders

GROUP BY

Ship\_Country

ORDER BY

Order\_Count DESC

LIMIT 1;

24. Find the order with the highest total cost

SELECT

Orders.Order\_ID,

SUM(Order\_Details.Unit\_Price \* Order\_Details.Quantity \* (1 - Order\_Details.Discount)) AS TotalCost

FROM

Orders

INNER JOIN Order\_Details ON Orders.Order\_ID = Order\_Details.Order\_ID

GROUP BY

Orders.Order\_ID

ORDER BY

TotalCost DESC

LIMIT 1;

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

SELECT

Employees.Employee\_ID,

CONCAT(Employees.First\_Name, ' ', Employees.Last\_Name) AS Employee\_Name,

COUNT(Orders.Order\_ID) AS Order\_Count

FROM

Employees

INNER JOIN Orders ON Employees.Employee\_ID = Orders.Employee\_ID

GROUP BY

Employees.Employee\_ID,

Employee\_Name

HAVING

Order\_Count > 100;

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

SELECT products.product\_name,customers.customer\_id,customers.company\_name,

sum(order\_details.quantity) as TQ

from

customers

inner join order\_details on order\_details.order\_id = orders.order\_id

inner join orders on orders.customer\_id = customers.customer\_id

inner join products on products.product\_id = order\_details.product\_id

where products.product\_name = 'Chai'

group by

customers.customer\_id,

customers.company\_name

order by TQ desc

limit 1;

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

SELECT

o.Order\_ID,

AVG(od.Quantity) AS AverageQuantity

FROM

Orders o

JOIN Order\_Details od ON o.Order\_ID = od.Order\_ID

GROUP BY

o.Order\_ID

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

SELECT

c.Category\_Name,

COUNT(\*) AS Order\_Count

FROM

Products p

JOIN Categories c ON p.Category\_ID = c.Category\_ID

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

c.Category\_Name

ORDER BY

Order\_Count DESC

limit 3

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

SELECT

order\_date,

TotalSales

FROM

(

SELECT

orders.order\_date,

SUM(order\_details.unit\_price \* order\_details.Quantity \* (1 - order\_details.Discount)) AS TotalSales

FROM

Orders

INNER JOIN order\_details ON orders.order\_ID = Order\_Details.Order\_ID

WHERE

YEAR(Order\_Date) = 2016

GROUP BY

orders.order\_date

) AS MonthlySales

ORDER BY

TotalSales DESC

LIMIT 1;

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

SELECT

e.Employee\_ID,

e.First\_Name,

e.Last\_Name,

COUNT(\*) AS OrderCount

FROM

Employees e

JOIN Orders o ON e.Employee\_ID = o.Employee\_ID

WHERE

o.Order\_Date >= '2016-08-01'

AND o.Order\_Date < '2016-09-01'

GROUP BY

e.Employee\_ID,

e.First\_Name,

e.Last\_Name

ORDER BY

OrderCount DESC

limit 1;

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

SELECT

c.Category\_Name,

COUNT(\*) AS OrderCount

FROM

Products p

JOIN Categories c ON p.Category\_ID = c.Category\_ID

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

c.Category\_Name

ORDER BY

OrderCount DESC

limit 3;

32. Find the employees who have not processed any orders

SELECT

e.Employee\_ID,

e.First\_Name,

e.Last\_Name

FROM

Employees e

WHERE

e.Employee\_ID NOT IN (SELECT DISTINCT Employee\_ID FROM Orders)

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

SELECT

s.supplier\_ID,

COUNT(\*) AS ProductCount

FROM

Suppliers s

JOIN Products p ON s.Supplier\_ID = p.Supplier\_ID

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

s.Supplier\_ID

ORDER BY

ProductCount DESC

limit 5 ;

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

SELECT

c.Customer\_ID,

c.Company\_Name

FROM

Customers c

WHERE

NOT EXISTS (

SELECT

DISTINCT Category\_ID

FROM

Categories

WHERE

Category\_ID NOT IN (

SELECT DISTINCT

p.Category\_ID

FROM

Products p

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

JOIN Orders o ON od.Order\_ID = o.Order\_ID

WHERE

o.Customer\_ID = c.Customer\_ID

)

);

35.  Find the total sales for each year

SELECT

YEAR(Order\_Date) AS SalesYear,

SUM(products.Unit\_Price \* od.Quantity) AS TotalSales

FROM

Orders o

JOIN Order\_Details od ON o.Order\_ID = od.Order\_ID

GROUP BY

YEAR(Order\_Date)

ORDER BY

SalesYear;

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’

SELECT

c.Customer\_ID,

c.Company\_Name,

SUM(p.Unit\_Price \* od.Quantity) AS TotalOrderAmount,

CASE

WHEN SUM(p.Unit\_Price \* od.Quantity) > 5000 THEN 'High Value'

WHEN SUM(p.Unit\_Price \* od.Quantity) > 1000 THEN 'Medium Value'

ELSE 'Low Value'

END AS Classification

FROM

Customers c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

JOIN Order\_Details od ON o.Order\_ID = od.Order\_ID

JOIN Products p ON od.Product\_ID = p.Product\_ID

GROUP BY

c.Customer\_ID,

c.Company\_Name

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’

SELECT

p.Product\_Name,

SUM(od.Quantity) AS TotalQuantity,

CASE

WHEN SUM(od.Quantity) > 1000 THEN 'High Sales'

WHEN SUM(od.Quantity) > 500 THEN 'Medium Sales'

ELSE 'Lower Sales'

END AS SalesCategory

FROM

Products p

JOIN Order\_Details od ON p.Product\_ID = od.Product\_ID

GROUP BY

p.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’

SELECT

e.First\_Name,

e.Last\_Name,

COUNT(\*) AS NumberOfOrders,

CASE

WHEN COUNT(\*) > 100 THEN 'High Performing'

WHEN COUNT(\*) > 50 THEN 'Medium Performing'

ELSE 'Lower Performing'

END AS PerformanceCategory

FROM

Employees e

JOIN Orders o ON e.Employee\_ID = o.Employee\_ID

GROUP BY

e.First\_Name,

e.Last\_Name