SQL Practices:

Using schema “classicmodels”

1. Write a SQL query that returns the total number of customers from city = ‘NYC’;
   ```
   select count(*) from customers where city = 'NYC';
   ```

2. Write a SQL query that returns the number of NOT NULL PriceEach values in the orderDetails table where the productcode = ‘S24_3969’
   ```
   select count(priceeach) from orderdetails where productcode = 'S24_3969';
   ```

3. Write a SQL query that returns the number of distinct NOT NULL PriceEach values in the orderDetails table where the productcode = ‘S24_3969’
   ```
   select count(distinct priceeach) from orderdetails where productcode = 'S24_3969';
   ```

4. Write a SQL query that returns the productcode and sum of the quantityordered of all orderDetails tuples of which the productcode = ‘S24_2840’
   ```
   select productcode, sum(quantityordered) as total_quantity from orderdetails where
   productcode = 'S24_2840';
   ```

5. Write a SQL query that returns the total quantityordered in the orderDetails table
   ```
   select sum(quantityordered) from orderdetails;
   ```

6. Write a SQL query that returns the weighted average priceeach of produccode = ‘S24_2840’ in the orderDetails table
   ```
   select avg(priceeach) as weighted_avg_price from orderdetails where productcode =
   'S24_2840';
   ```

7. Write a SQL query that returns the productcode and the unweighted average priceeach of produccode = ‘S24_2840’ in the orderDetails table
   ```
   select avg(distinct priceeach) as unweighted_avg_price from orderdetails where productcode =
   'S24_2840';
   ```

8. Write a SQL query that returns the productcode and the variance of the priceeach of produccode = ‘S24_2840’ in the orderDetails table
   ```
   select variance(priceeach) as variance_price from orderdetails where productcode = 'S24_2840';
   ```

9. Write a SQL query that returns the productcode, minimum priceeach, maximum priceeach of produccode = ‘S24_2840’ in the orderDetails table
   ```
   select min(priceeach) as min_price, max(priceeach) as max_price from orderdetails where
   productcode = 'S24_2840';
   ```

10. Write a SQL query that returns all productcode with at least 25 outstanding orders in the orderDetails table

```
select productcode from orderdetails
group by productcode
having count(*) >= 25;
```

11. Write a SQL query that returns all productcode and total quantityordered with the total quantityordered exceeds 1000 in the orderDetails table

```
select productcode from orderdetails
group by productcode
having sum(quantityordered) > 1000;
```

12. Write a SQL query that returns a list of outstanding orders, ordered ascending by date and descending by customernumber

```
select ordernumber from orders order by orderdate asc, customernumber desc;
```

13. Write a SQL query that returns productcode, ordernumber, priceeach with productcode = 'S24_2840' order by the 3rd column in descending order

```
select productcode, ordernumber, priceeach from orderdetails
where productcode = 'S24_2840'
order by 3 desc;
```

14. Write a SQL query that returns, for each customer with outstanding orders, customernumber, customername, together with the ordernumber, date, the productcode, productname, and the quantityordered

```
select c.customernumber, c.customername, o.ordernumber, o.orderdate, p.productcode,
p.productname, od.quantityordered
from customers c, orders o, orderdetails od, products p
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber and
od.productcode = p.productcode;
```

15. Write a SQL query that returns all pairs of customers who are located in the same city

```
select c1.customername, c2.customername, c1.city
from customers c1, customers c2
where c1.city = c2.city and c1.customernumber < c2.customernumber;
```

16. Write a SQL query that returns customernumber, customername who have ordered at least one type of productline = 'planes'

```
select c.customernumber, c.customername
from customers c, orders o, orderdetails od, products p
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber
and od.productcode = p.productcode and p.productline = 'planes';
```

17. Write a SQL query that returns customername, productname, shippeddate, quantityordered for all shipped orders

```
select c.customername, p.productname, o.shippeddate, od.quantityordered
from customers c, orders o, orderdetails od, products p
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber and
od.productcode = p.productcode and o.status = 'shipped';
```

18. Write a SQL query that returns the productcode, productname, and total quantityordered for each product that is in an order

```
select od.productcode, p.productname, sum(quantityordered) as totalquantity
from orderdetails od, products p
where od.productcode = p.productcode
group by od.productcode;
```

19. Write a SQL query that returns customernumber, customername, and if applicable, include the ordernumbers of their orders. Namely the information of all customer even if they do not have any order

```
select c.customernumber, c.customername, o.ordernumber
from customers as c left outer join orders as o
on c.customernumber = o.customernumber;
```

20. Write a SQL query that returns all productcode, together with the productname and total quantityordered, even if there are currently no outstanding orders for a product

```
select p.productcode, p.productname, sum(od.quantityordered)
from orderdetails as od right outer join products as p
on od.productcode = p.productcode
group by p.productcode;
```

21. Write a nested SQL query that returns the name of the customer with whom the order number ‘10202’ is placed

```
select customername
from customers
where customernumber = (select customernumber from orders where ordernumber = '10202');
```

22. Write a nested SQL query that returns the names of the customers who ordered the product with productcode ‘S24_2840’

```
select customername
from customers
```

```
where customernumber in (select o.customernumber from orders o, orderdetails od
where o.ordernumber = od.ordernumber and od.productcode = 'S24_2840');
```

23. Write a nested SQL query that returns the names of the customers who ordered the product with productcode ‘S24_2840’ and the product with productcode ‘S50_1341’

```
select customername
from customers
where customernumber in (select o.customernumber from orders o, orderdetails od
where o.ordernumber = od.ordernumber and od.productcode = 'S24_2840') and
customernumber in (select o.customernumber from orders o, orderdetails od
where o.ordernumber = od.ordernumber and od.productcode = 'S50_1341');
```

24. Write a correlated SQL query that returns the productname of all products with at least 5 orders

```
select p.productname
from products p
where (select count(*) from orderdetails od where od.productcode = p.productcode group by
od.productcode) >=5;
```

25. Write a correlated SQL query that returns the customernumber, customername of the customers who ordered a product at a priceeach lower than the average priceeach of that product, together with the productcode, productname, priceeach, and quantityordered

```
select c.customernumber, c.customername, p.productcode, p.productname, od.priceeach,
od.quantityordered
from customers c, products p, orders o, orderdetails od
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber and
od.productcode = p.productcode
and od.priceeach < (select avg(priceeach) from orderdetails where productcode =
p.productcode);
```

26. Write a correlated SQL query that returns the customernumber, customername with the top 2 most orders

```
select customernumber, customername
from customers
where customernumber in
(select m1.customernumber from (select customernumber, count(*) as num from orders group
by customernumber) as m1
where 2 > (select count(*) from (select customernumber, count(*) as num from orders group by
customernumber) as m2
where m1.num < m2.num));
```

(Below is not a preferred approach!)

```
select c1.customernumber, c1.customername
from customers c1, orders o
where c1.customernumber = o.customernumber
group by o.customernumber
order by count(*) desc
limit 2;
```

27. Write a SQL query that returns the customernumber, customername who ordered the productcode ‘S18_3136’ with the highest priceeach

```
select c.customernumber, c.customername
from customers c
where c.customernumber = (select o.customernumber from orders o, orderdetails od where
o.ordernumber = od.ordernumber
and productcode = 'S18_3136' and od.priceeach >= all (select priceeach from orderdetails where
productcode = 'S18_3136'));
```

28. Write a SQL query that returns the customernumber, customername who ordered the productcode ‘S18_3136’ and did not pay the lowest priceeach

```
select c.customernumber, c.customername
from customers c
where c.customernumber in (select o.customernumber from orders o, orderdetails od where
o.ordernumber = od.ordernumber
and productcode = 'S18_3136' and od.priceeach > any (select priceeach from orderdetails where
productcode = 'S18_3136'));
```

29. Write a SQL query using ‘EXISTS’ that returns the customernumber, customername who ordered the productcode ‘S18_3136’

```
select c.customernumber, c.customername
from customers c
where exists (select * from orders o, orderdetails od where o.ordernumber = od.ordernumber
and o.customernumber = c.customernumber and productcode = 'S18_3136');
```

30. Write a SQL query that returns the customernumber, customername who are either located in ‘Boston’ or have ordered the productcode ‘S18_3136’

```
select customernumber, customername
from customers
where city = 'boston'
union
select c.customernumber, c.customername
```

```
from customers c, orders o, orderdetails od
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber and
od.productcode = 'S18-3136';
```

31. Write a SQL query that returns the customernumber, customername who are located in ‘Boston’ and have ordered the productcode ‘S18_3136’. (one may use ‘intersect’. However, ‘intersect’ is not supported in MySQL. Thank about an alternative)

    An alternative to “INTERSECT”

```
select c.customernumber, c.customername
from customers c, orders o, orderdetails od
where c.customernumber = o.customernumber and o.ordernumber = od.ordernumber and
c.city = ‘boston’ and od.productcode = 'S18-3136';
```

32. Write a SQL query that returns the customernumber, customername who have not ordered any product. (one may use ‘except’. However, ‘except’ is not supported in MySQL. Think about an alternative).

    An alternative to “EXCEPT”

```
select customernumber, customername
from customers
where customernumber not in (select customernumber from orders);
```