--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--Name:Gozde Arslan

--ID: 150320190

--Date: 20/02/2021

--Purpose: Assignment 1 DBS311

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- Q1. Display the employee number, full employee name, jobtitle,and hire date of all employees hired in Septemberwith the most recently hired employees displayed first.

SELECT

employee\_id AS "Employee Number",

rpad(first\_name || ', ' || last\_name, 25, ' ') AS "Full Name",

job\_title AS "JOB TITLE",

to\_char(last\_day(hire\_date), '[Month ddTH "of" yyyy]') AS "Hire Date"

FROM

employees

WHERE

EXTRACT(MONTH

FROM

hire\_date) IN

(

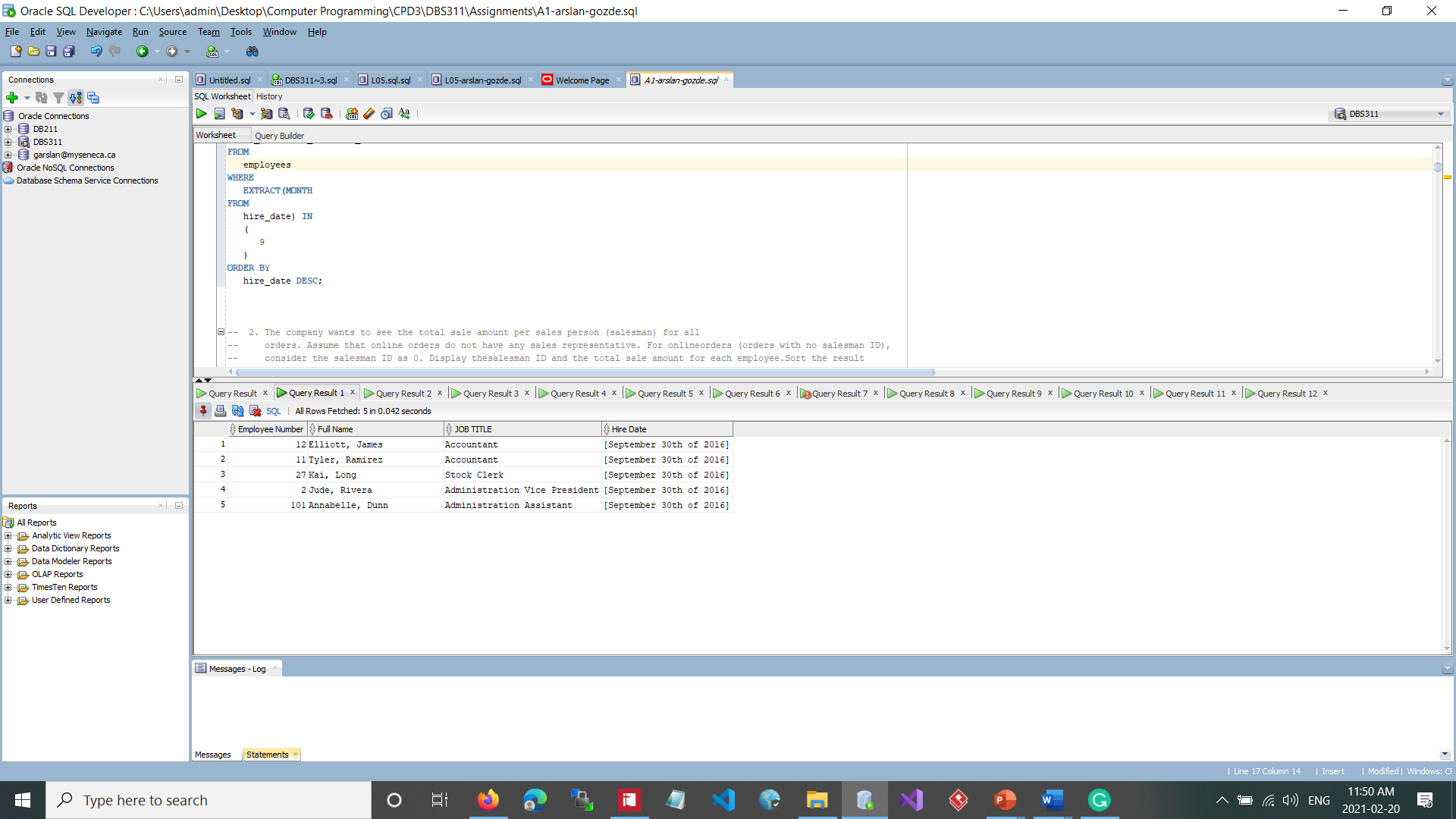
9

)

ORDER BY

hire\_date DESC;

**-- Solution Q1--**



-- 2. The company wants to see the total sale amount per sales person (salesman) for all

-- orders. Assume that online orders do not have any sales representative. For onlineorders (orders with no salesman ID),

-- consider the salesman ID as 0. Display thesalesman ID and the total sale amount for each employee.Sort the result

-- according to employee number.

SELECT

NVL( o.salesman\_id,0) AS "Employee Number",

SUM(i.unit\_price\*i.quantity) AS "Total Sale"

FROM

orders o,

order\_items i

WHERE

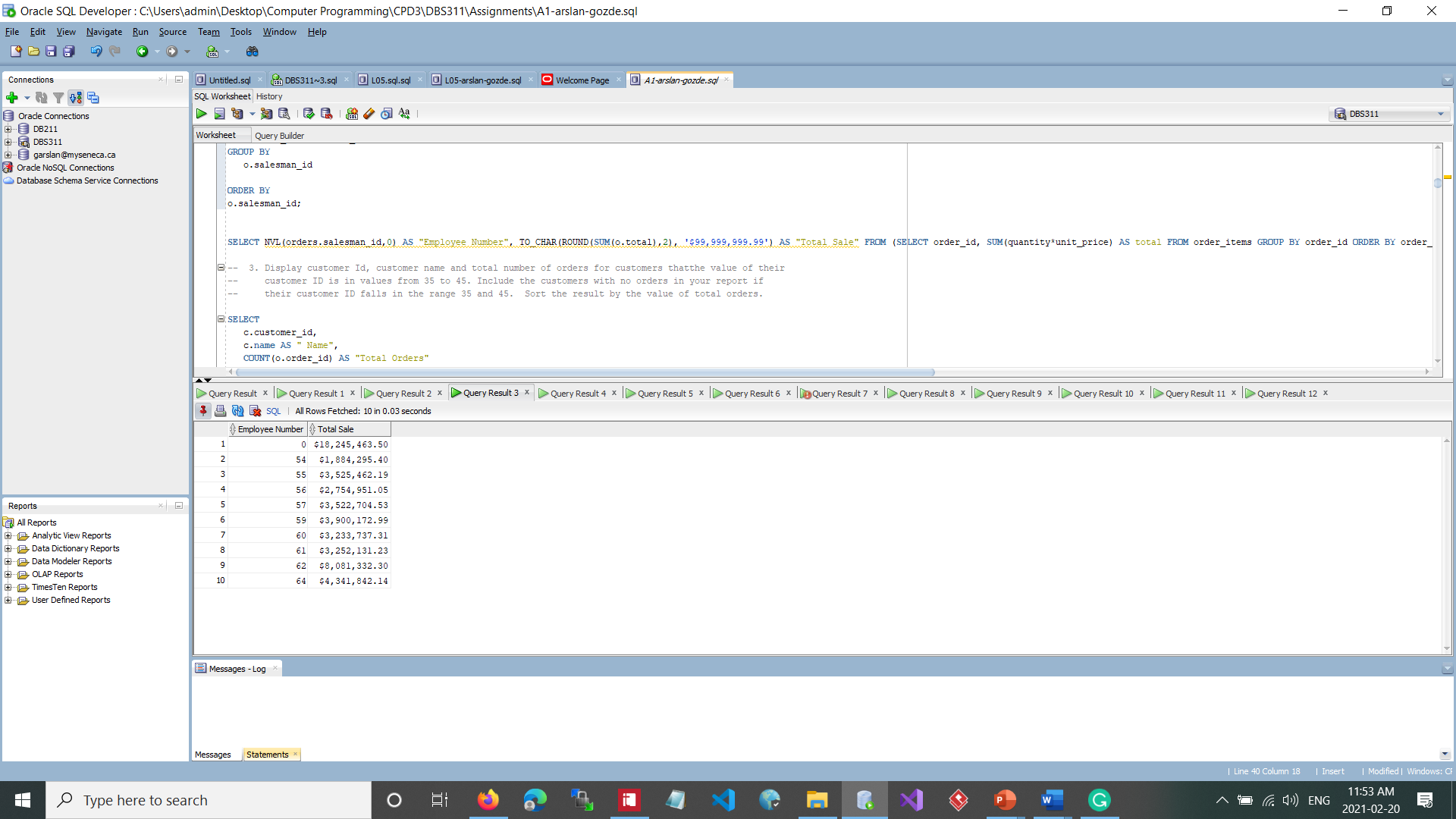
o.order\_id = i.order\_id

GROUP BY

o.salesman\_id

ORDER BY NVL(o.salesman\_id,0) ASC;

**-- Solution Q2--**



-- 3. Display customer Id, customer name and total number of orders for customers thatthe value of their

-- customer ID is in values from 35 to 45. Include the customers with no orders in your report if

-- their customer ID falls in the range 35 and 45. Sort the result by the value of total orders.

SELECT

c.customer\_id,

c.name AS " Name",

COUNT(o.order\_id) AS "Total Orders"

FROM

customers c

LEFT JOIN

orders o

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

WHERE

c.customer\_id BETWEEN 35 AND 45

GROUP BY

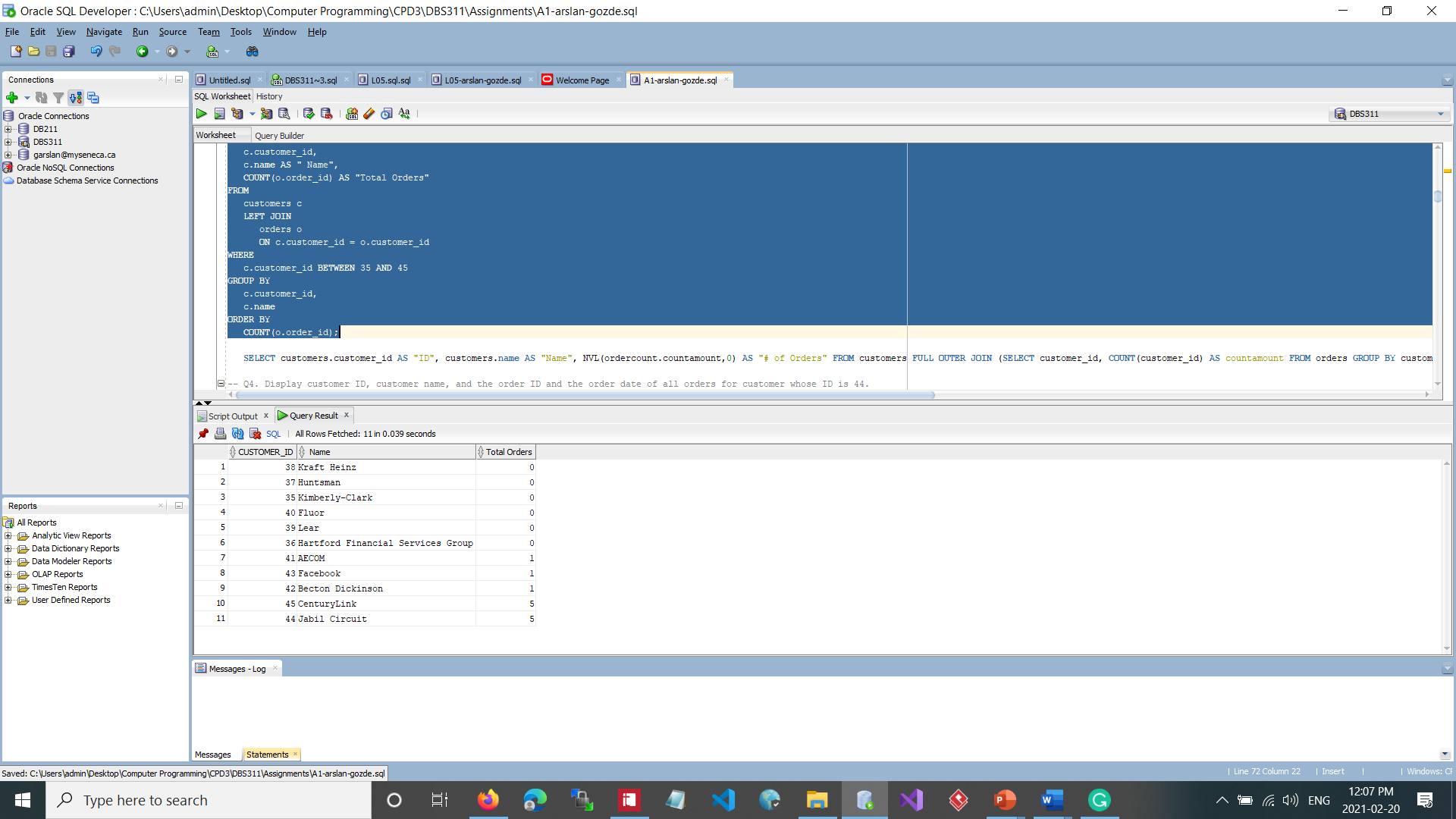
c.customer\_id,

c.name

ORDER BY

COUNT(o.order\_id);

**-- Solution Q3--**



-- 4. Display customer ID, customer name, and the order ID and the order date of all orders for customer whose ID is 44.

-- a. Show also the total quantity and the total amount of each customer?s order.

-- b. Sort the result from the highest to lowest total order amount.

SELECT

c.customer\_id,

name,

o.order\_id,

order\_date,

SUM(quantity) AS "Total\_items",

SUM(quantity \* unit\_price) AS "Total\_amount"

FROM

customers c

JOIN

orders o

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

JOIN

order\_items oi

ON o.order\_id = oi.order\_id

WHERE

c.customer\_id = 44

GROUP BY

c.customer\_id,

name,

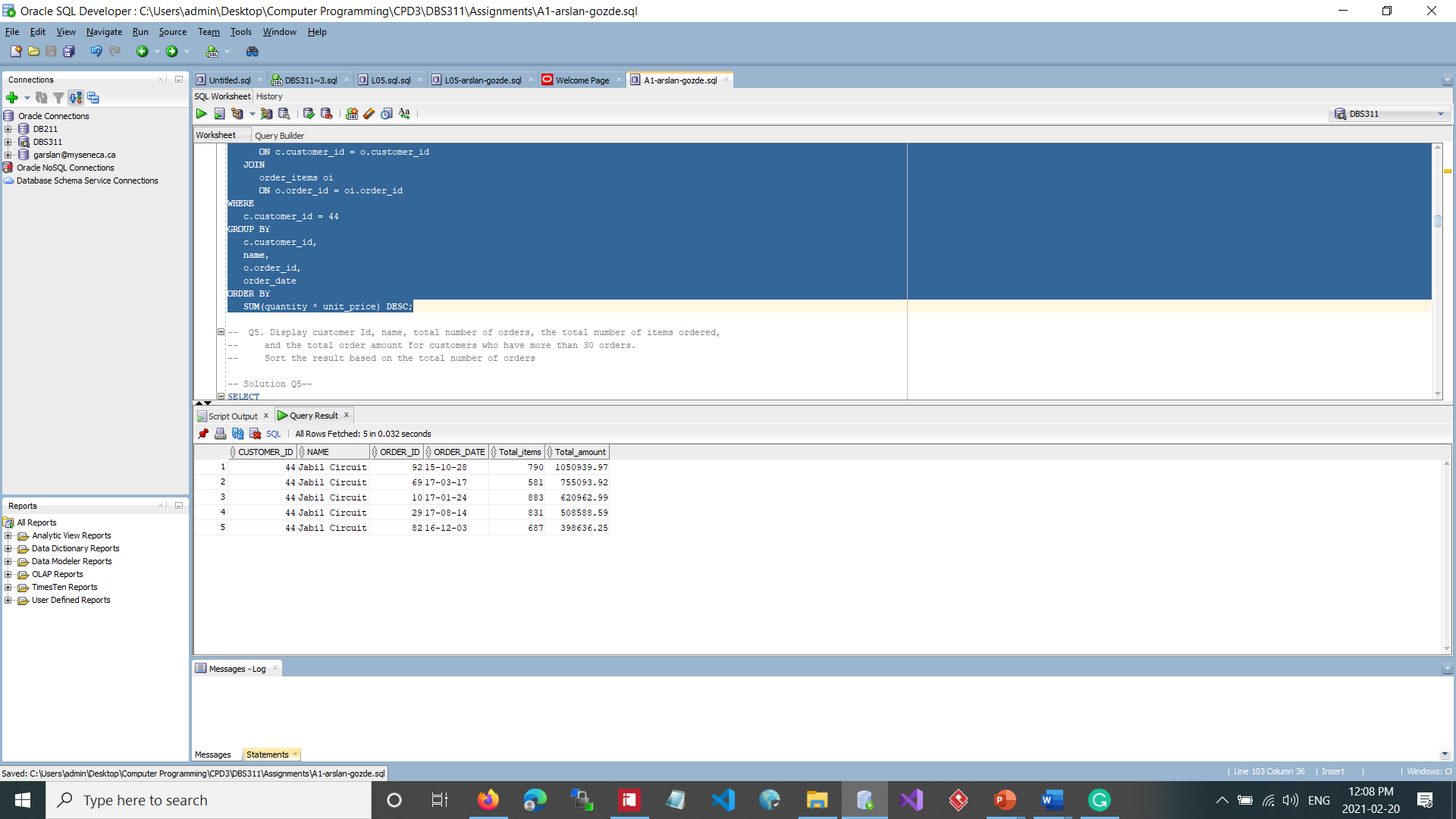
o.order\_id,

order\_date

ORDER BY

SUM(quantity \* unit\_price) DESC;

**-- Solution Q4--**



-- 5. Display customer Id, name, total number of orders, the total number of items ordered,

-- and the total order amount for customers who have more than 30 orders.

-- Sort the result based on the total number of orders

SELECT

c.customer\_id,

c.name,

COUNT(o.order\_id) AS "total number of orders",

SUM(oi.quantity) AS "Total Items",

SUM(oi.quantity \* oi.unit\_price) AS "Total Amount"

FROM

orders o,

customers c,

order\_items oi

WHERE

c.customer\_id = o.customer\_id

AND o.order\_id = oi.order\_id

GROUP BY

c.customer\_id,

c.name

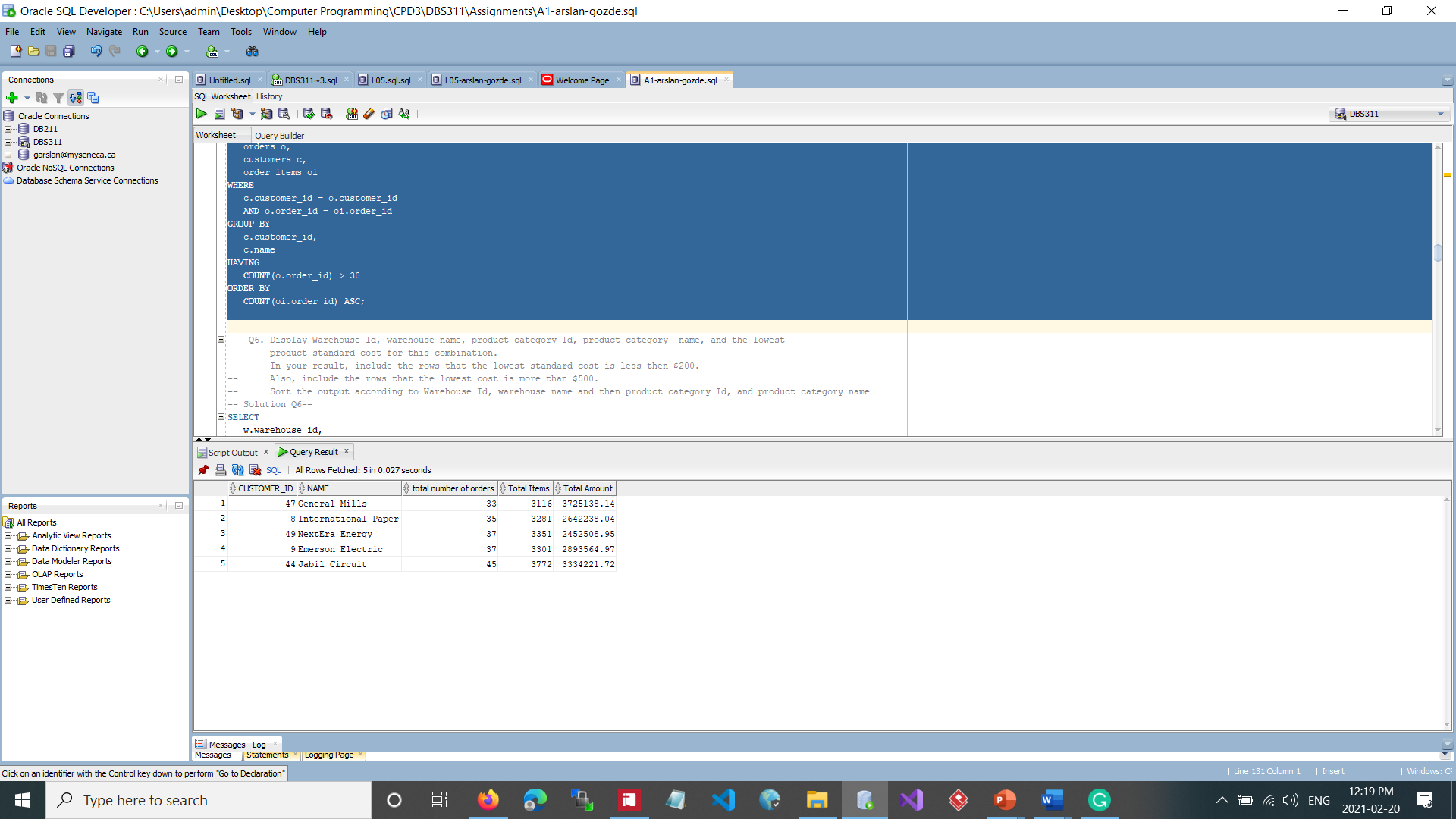
HAVING

COUNT(o.order\_id) > 30

ORDER BY

COUNT(oi.order\_id) ASC;

**-- Solution Q5--**



-- 6. Display Warehouse Id, warehouse name, product category Id, product category name, and the lowest

-- product standard cost for this combination.

-- ? In your result, include the rows that the lowest standard cost is less then $200.

-- ? Also, include the rows that the lowest cost is more than $500.

-- ? Sort the output according to Warehouse Id, warehouse name and then product category Id, and product category name

SELECT

w.warehouse\_id,

warehouse\_name,

p.category\_id,

category\_name,

MIN(standard\_cost) AS lowest\_cost

FROM

inventories i

JOIN

warehouses w

ON i.warehouse\_id = w.warehouse\_id

JOIN

products p

ON i.product\_id = p.product\_id

JOIN

product\_categories pc

ON p.category\_id = pc.category\_id

GROUP BY

w.warehouse\_id,

warehouse\_name,

p.category\_id,

category\_name

HAVING

MIN(standard\_cost) < 200

OR MIN(standard\_cost) > 500

ORDER BY

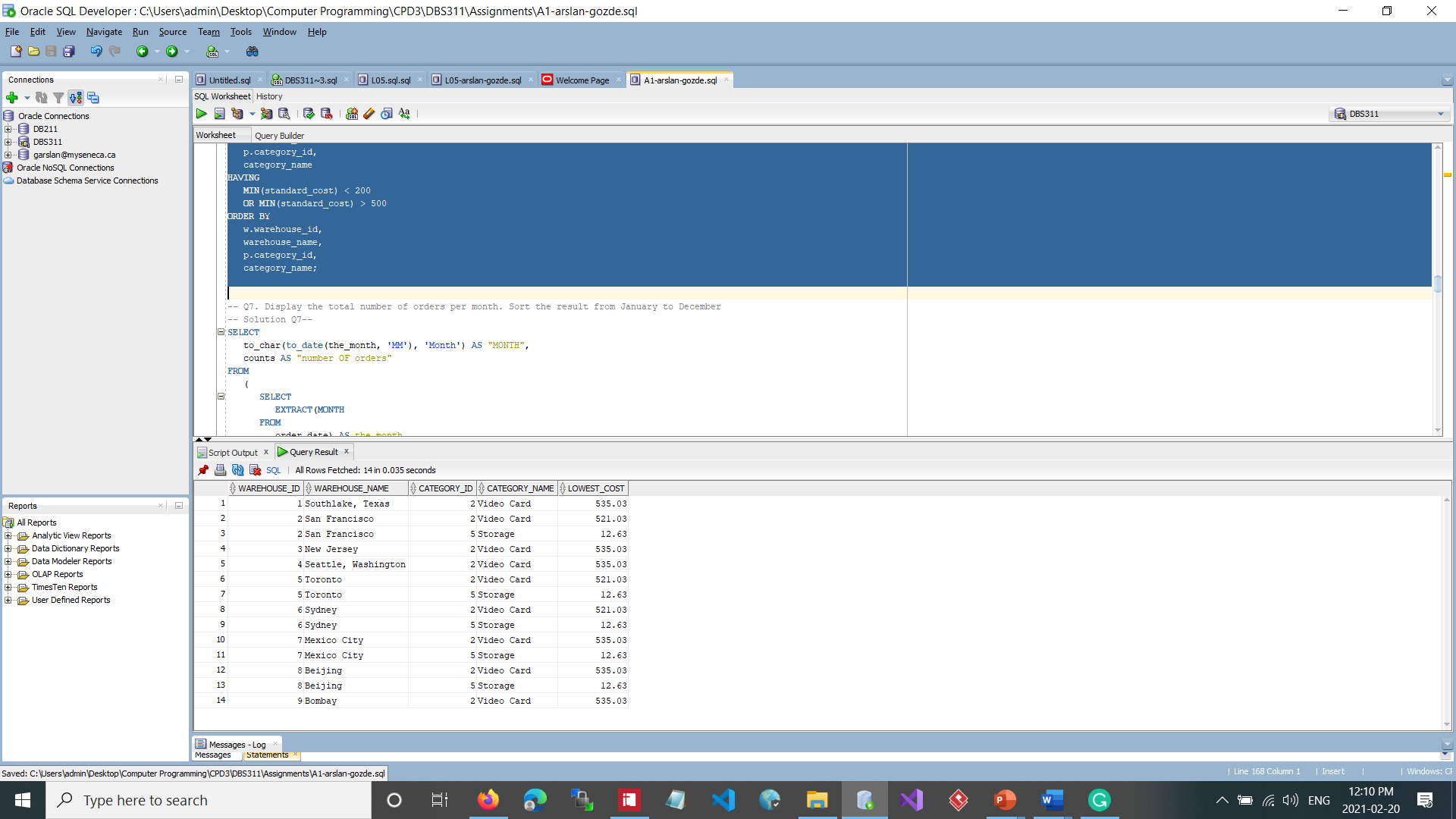
w.warehouse\_id,

warehouse\_name,

p.category\_id,

category\_name;

**-- Solution Q6--**



-- 7. Display the total number of orders per month. Sort the result from January to December

SELECT

to\_char(to\_date(the\_month, 'MM'), 'Month') AS "MONTH",

counts AS "number of orders"

FROM

(

SELECT

EXTRACT(MONTH

FROM

order\_date) AS the\_month,

COUNT(\*) AS counts

FROM

orders

GROUP BY

EXTRACT(MONTH

FROM

order\_date)

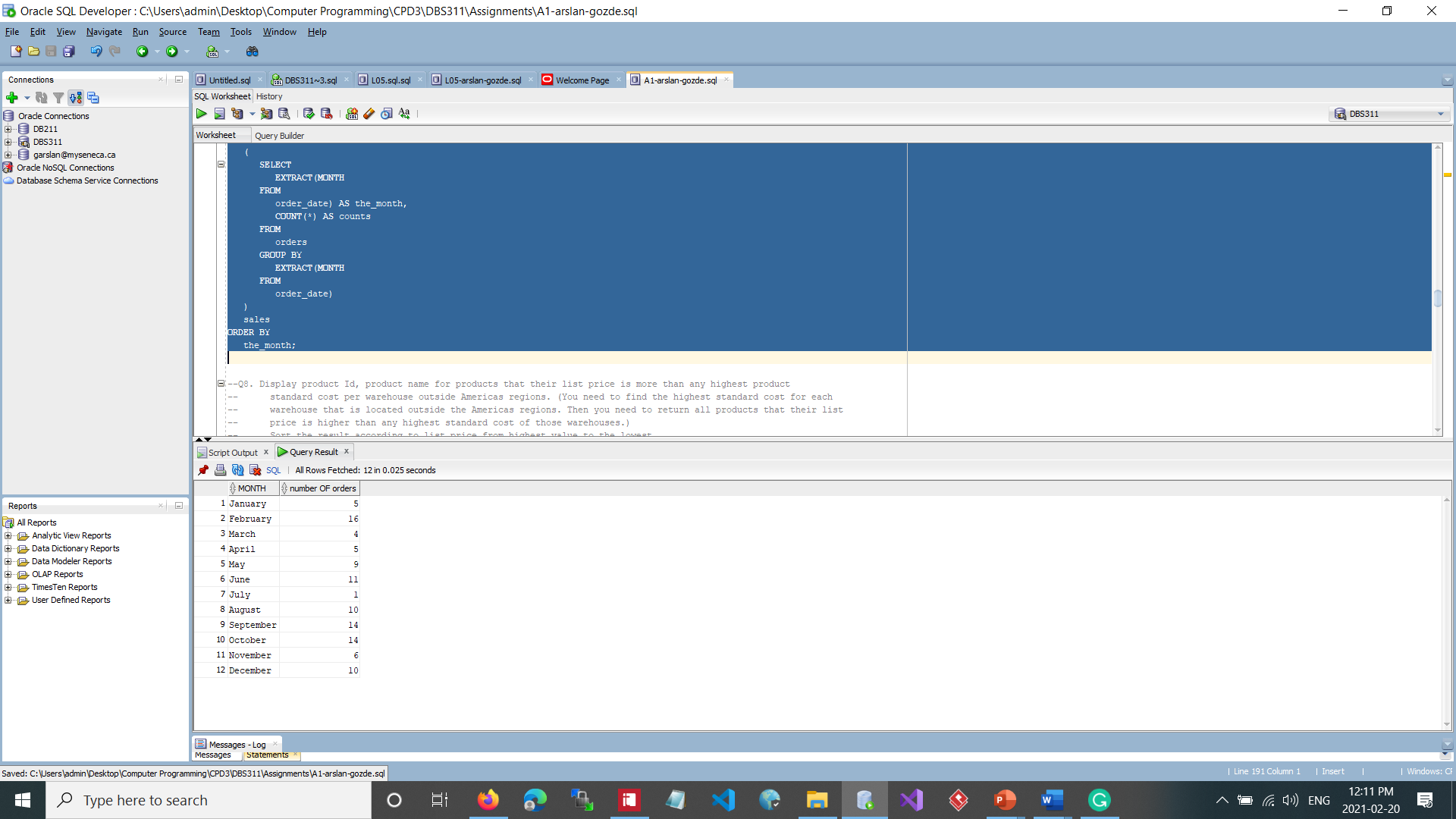
)

sales

ORDER BY

the\_month;

-- Solution Q7--



-- 8. Display product Id, product name for products that their list price is more than any highest product

-- standard cost per warehouse outside Americas regions. (You need to find the highest standard cost for each

-- warehouse that is located outside the Americas regions. Then you need to return all products that their list

-- price is higher than any highest standard cost of those warehouses.)

-- Sort the result according to list price from highest value to the lowest.

SELECT

product\_id AS "product id",

product\_name AS "product name",

to\_char(list\_price, '$999,999.99') AS "price"

FROM

products

WHERE

list\_price > ANY (

SELECT

MAX(standard\_cost)

FROM

locations l

JOIN

countries c

ON l.country\_id = c.country\_id

JOIN

regions r

ON r.region\_id = c.region\_id

JOIN

warehouses w

ON w.location\_id = l.location\_id

JOIN

inventories i

ON i.warehouse\_id = w.warehouse\_id

JOIN

products p

ON p.product\_id = i.product\_id

WHERE

region\_name NOT LIKE 'Americas'

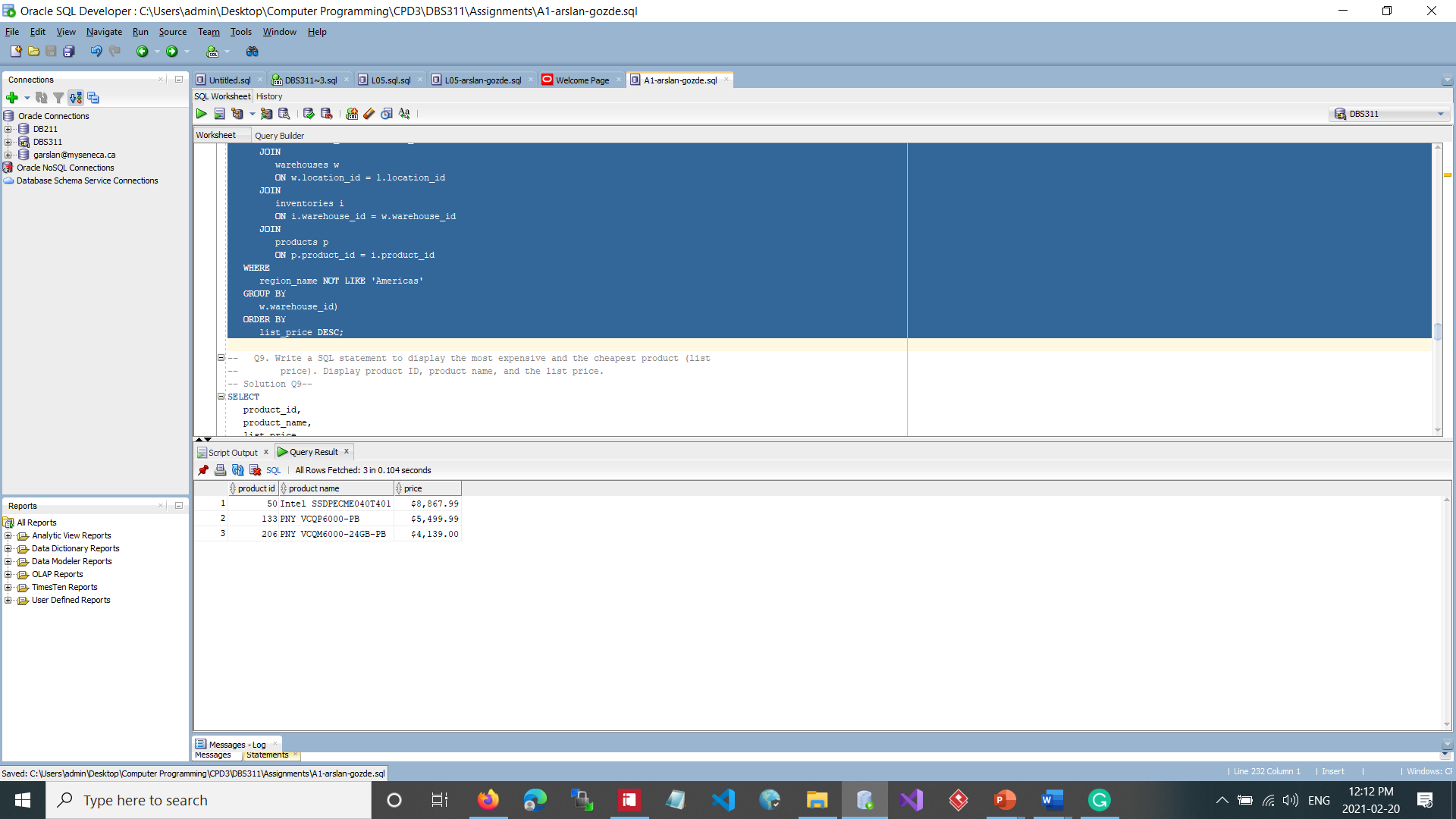
GROUP BY

w.warehouse\_id)

ORDER BY

list\_price DESC;

**-- Solution Q8--**



-- 9. Write a SQL statement to display the most expensive and the cheapest product (list

-- price). Display product ID, product name, and the list price.

SELECT

product\_id,

product\_name,

list\_price

FROM

products

WHERE

list\_price =

(

SELECT

MAX(list\_price)

FROM

products

)

OR list\_price =

(

SELECT

MIN(list\_price)

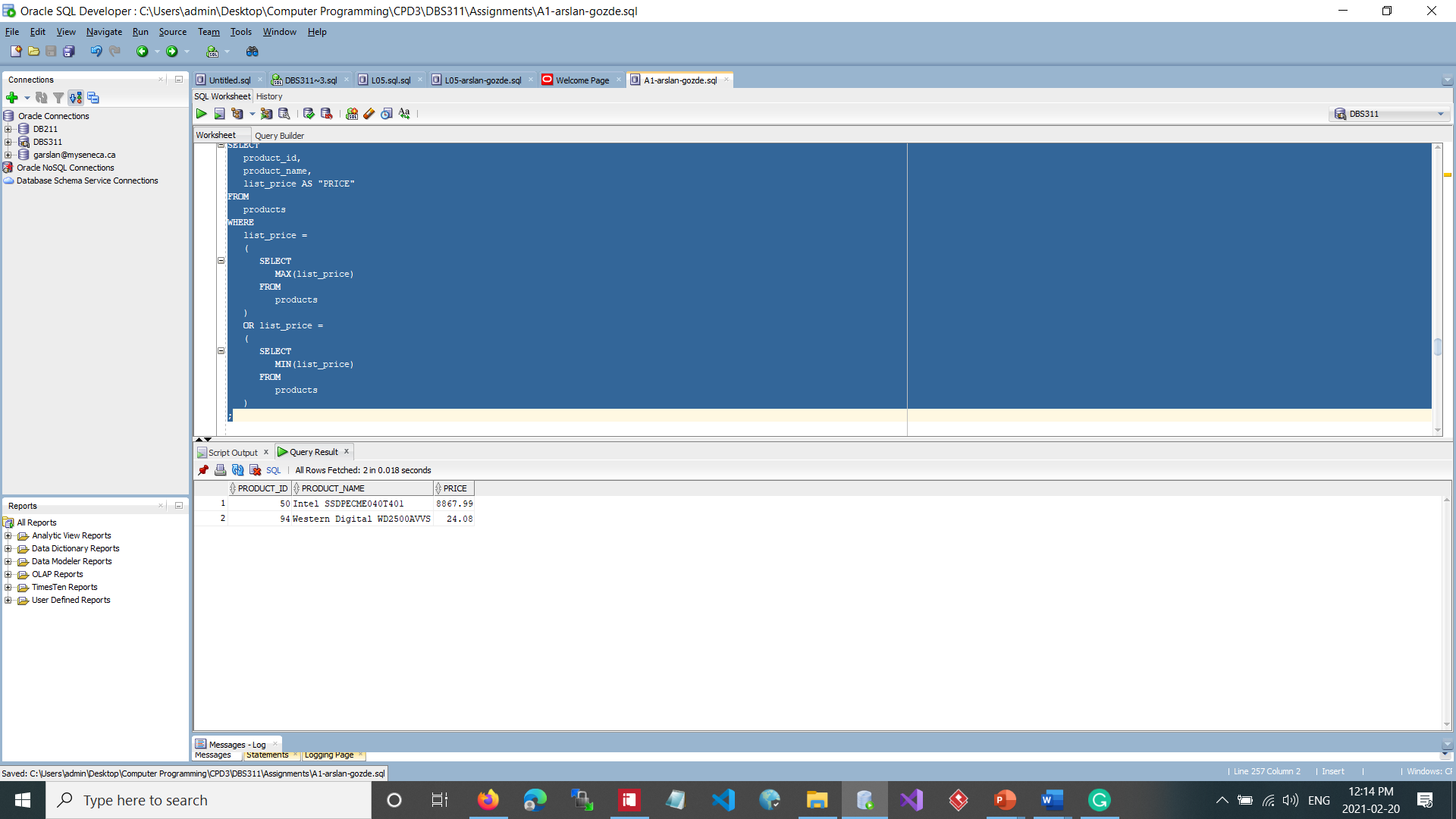
FROM

products

)

;

**-- Solution Q9--**



-- 10. Write a SQL query to display the number of customers with total order amount over the average amount

-- of all orders, the number of customers with total order amount under the average amount of all orders,

-- number of customers with no orders, and the total number of customers. See the format of the following result.

SELECT

'Number of customers with total purchase amount over average: ' || COUNT(\*) AS "customer report"

FROM

(

SELECT

c.customer\_id,

SUM(i.quantity\*i.unit\_price) AS total\_amount

FROM

customers c

INNER JOIN

orders o

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

INNER JOIN

order\_items i

ON i.order\_id = o.order\_id

GROUP BY

c.customer\_id

)

WHERE

total\_amount > (

SELECT

AVG(quantity\*unit\_price)

FROM

order\_items)

UNION ALL

SELECT

'Number of customers with total purchase amount below average: ' || COUNT(\*)

FROM

(

SELECT

c.customer\_id,

SUM(i.quantity\*i.unit\_price) AS total\_amount

FROM

customers c

INNER JOIN

orders o

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

INNER JOIN

order\_items i

ON i.order\_id = o.order\_id

GROUP BY

c.customer\_id

)

WHERE

total\_amount < (

SELECT

AVG(quantity\*unit\_price)

FROM

order\_items)

UNION ALL

SELECT

'Number of customers with no orders: ' || COUNT(\*)

FROM

(

SELECT

customer\_id

FROM

customers minus

SELECT

customer\_id

FROM

orders

)

UNION ALL

SELECT

'Total number of customers: ' || COUNT(\*)

FROM

(

SELECT

customer\_id

FROM

customers

UNION

SELECT

customer\_id

FROM

orders

)

;

**-- Solution Q10--**

