# Assignment 1, Section A Worth 20% Out of 100

# Due by Wednesday, February 24th by 9pm

# Group 6

# Name : Leonaire George

# Student ID = 012067153

# Name : Long Nguyen

# Student ID: 155176183

# Name : Rujal Tuladhar

# Student ID: 154594188

## Submission

***Your submission will be a Document file containing all 10 Select statements Plus their Outputs. Also you will attach a single text-based SQL file (.sql) with appropriate header and commenting.*** *Please ensure your file runs when the entire file is executed in SQL Developer or SQL Plus.*

Save these files as **Asgn1\_Gr6.docx** and **Asgn1\_Gr6.sql**, here is X your Group Number.  
 **Only one e-mail submission per group please.**

## Submission Checklist Use the following checklist, to make sure you have completed the assignment successfully.

|  |  |  |
| --- | --- | --- |
| **Tasks to be completed** | **Yes** | **No** |
| You have read the assignment group submission and completion policies and all instructions provided in the assignment document and have not missed a word. | YES |  |
| Student information and the assignment information have been added to the header of the submission. (Same as the template provided in the assignment documents) | YES |  |
| All questions are answered in a text file and are saved as a ***.sql*** file. | YES |  |
| Comments are included. (questions definition or any additional explanation) | YES |  |
| All SQL statements are executed successfully without errors. (Use "Run Script" or @scriptname to execute all statements together.) | YES |  |
| The result of your SQL statements contains the given sample row and the header in the assignment document. | YES |  |

## 

## Group Work

This assignment is to be completed in groups of 3 or 4. Please only one submission per group. The comment header MUST have all students’ name and student number.

It is suggested that you **ALL do it individually** and then meet to compare answers. Those not doing the work may be barred from your group resulting in a zero and incomplete on the assignment.

## Tasks

For each question, the Header and the First Sample Row (or Rows) must match the sample row or rows given in that question. If you are using SQL Plus, then you may format the width of certain items (columns) output with SUBSTR(item,1,length)

1. Display the employee number, full employee name, job title, and hire date of all employees hired in September, excluding employees with Administrative jobs (their title starts with **Admin**), with the most recently hired employees displayed first, followed by their last name ascending.

SELECT employee\_id AS "Emp Id", (last\_name || ',' || first\_name)

AS "Full Name", job\_title AS "Job",

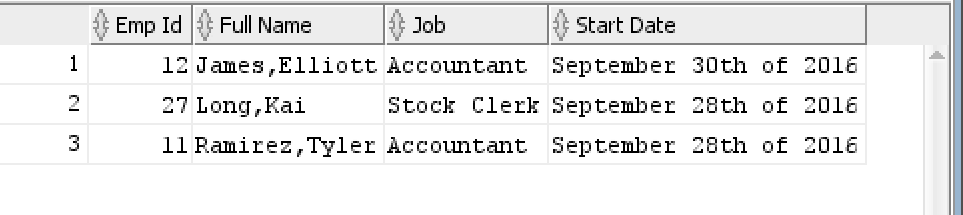
to\_char(hire\_date, 'fmMonth ddth "of "YYYY') AS "Start Date" FROM employees

WHERE UPPER(job\_title) NOT LIKE UPPER('Admin%')

AND extract(month from hire\_date)=extract(month from DATE '2020-09-01')

ORDER BY hire\_date DESC, last\_name ASC;

OUTPUT:



Emp Id Full Name Job Start Date

**---------- -------------------- -------------- ------------------------------**

**12 James, Elliott Accountant [September 30th of 2016]**

1. The company wants to see the total sale amount per sales person (salesman) for all orders. Assume that some online orders do not have any sales representative. For online orders (orders with no salesman ID), consider the salesman ID as 0. Display the salesman ID and the total sale amount for the employee for each employee.

Sort the result according to employee number.

SELECT COALESCE(employee\_id, 0) AS "Employee Id",

TO\_CHAR(SUM(unit\_price\*quantity),'$999,999,999.99') AS "Total Sale"

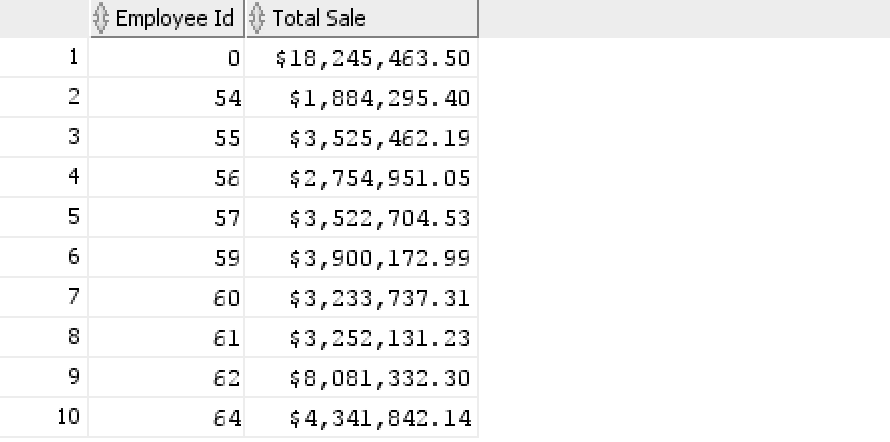
From (employees RIGHT JOIN orders ON (employees.employee\_id=orders.salesman\_id))

LEFT JOIN order\_items USING (order\_id)

GROUP BY employee\_id

ORDER BY "Employee Id" ASC;

OUTPUT:

1. Display customer Id, customer name and total number of orders for customers with their Id less than 200 and with name starting on F or J, but only if their total number of orders is less than 3. Include the customers with no orders in your report as well. Sort the result by the value of total orders ascending, followed by name ascending.

SELECT customer\_id AS "CustId", name AS "Name", COUNT(order\_id) AS "Total Orders"

FROM customers LEFT JOIN orders USING(customer\_id)

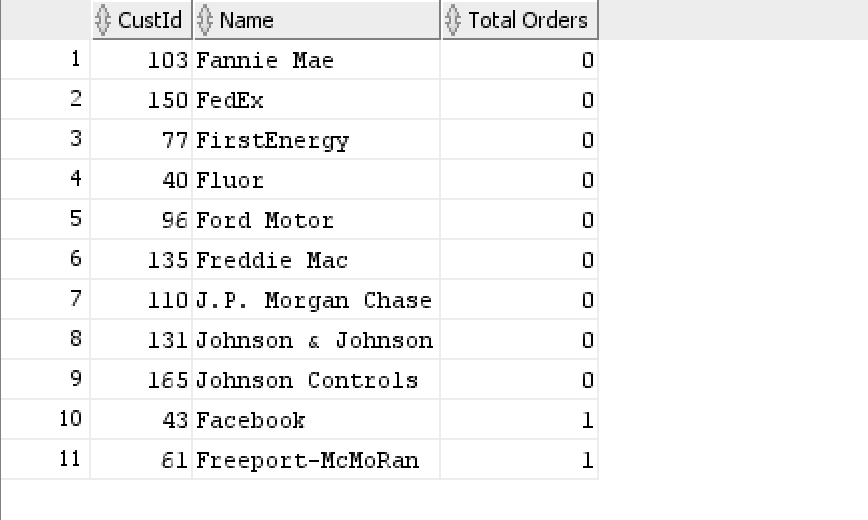
WHERE (UPPER(name) LIKE UPPER('F%') OR UPPER(name) LIKE UPPER('J%'))

GROUP BY customer\_id, name HAVING (customer\_id < 200

AND COUNT(order\_id) < 3)

ORDER BY "Total Orders" ASC, customers.name ASC;

OUTPUT:



**CustId Name Total Orders**

**---------- -------------------- --------------**

**103 Fannie Mae 0**

**150 FedEx 0**

1. Display customer Id, customer name, and the order id and the order date of all orders for customer whose ID is 44.
   1. Show also the total number of items ordered and the total amount of each customer’s order.
   2. Exclude Orders with the Total Amount exceeding 1 million dollars
   3. Sort the result from the highest to lowest total order amount.

SELECT customer\_id AS "Cust#", name AS "Name", order\_id AS "Order Id",

TO\_CHAR(order\_date, 'DD-MON-YY') AS "Order Dat",

SUM(quantity) AS "Total Items",

TO\_CHAR(SUM(unit\_price\*quantity),'$999,999,999.99') AS "Total Amount"

FROM (orders LEFT JOIN order\_items USING (order\_id))

JOIN customers USING(customer\_id)

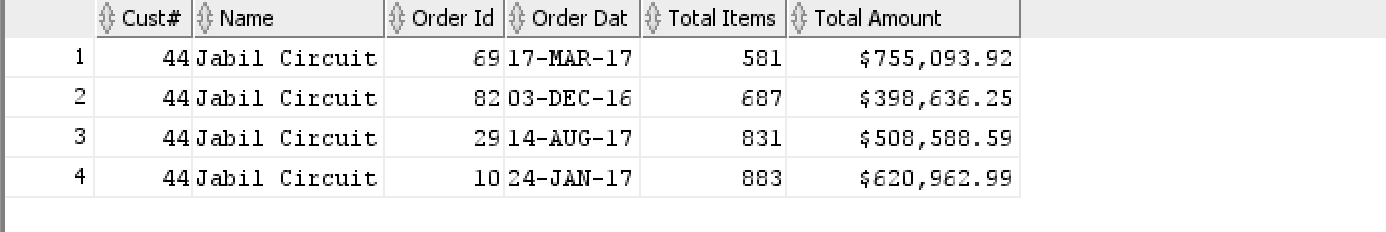
WHERE order\_id IN(SELECT order\_id FROM orders WHERE customer\_id = 44)

GROUP BY order\_id,order\_date,name,customer\_id

HAVING(SUM(unit\_price\*quantity) < 1000000)

ORDER BY "Total Items";

OUTPUT:



**Cust# Name Order Id Order Dat Total Items Total Amount**

**---------- ----------------- ---------- ------------- --------------- ----------------**

**44 Jabil Circuit 69 17-MAR-17 581 $755,093.92**

1. 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.

SOLUTION:

select customer\_id as "Cust#", name as "Name", count(order\_id) as "#of Orders",

sum(quantity) as "Total Items",

to\_char(sum(quantity\*unit\_price),'$9,999,999.99') as "Total Amount"

from orders

join customers using (customer\_id)

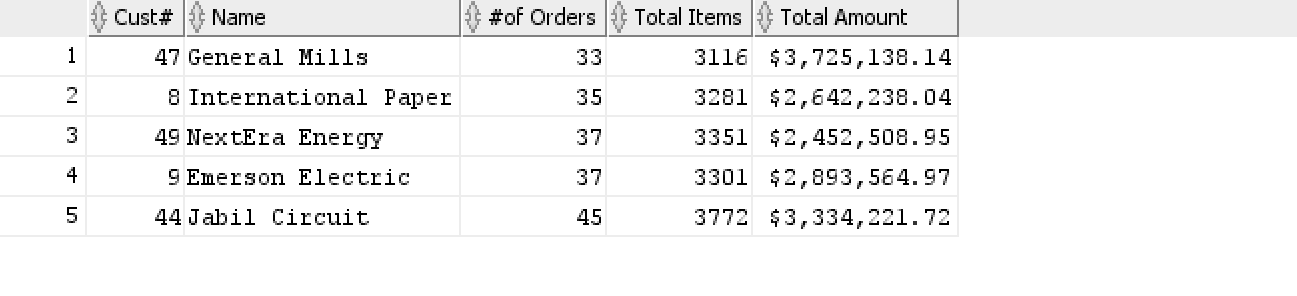
join order\_items using (order\_id)

group by customer\_id,name

having count(order\_id) > 30

order by 3;

OUTPUT:



Cust# Name # of Orders Total Items Total Amount

**---------- -------------------- ------------- ------------- --------------**

**47 General Mills 33 3116 $3,725,138.14**

1. 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 AS "Wrhs#",

warehouse\_name AS "Warehuse Name",

p.category\_id AS "Category ID",

category\_name AS "Category Name",

MIN (TO\_CHAR(standard\_cost, '$999.99')) 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

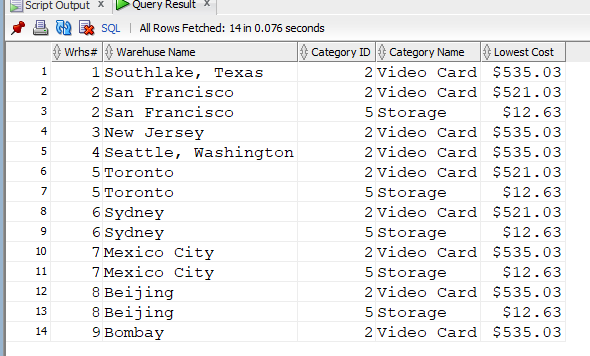
w.warehouse\_id,

warehouse\_name,

p.category\_id,

category\_name;

OUTPUT:



**Wrhs# Warehouse Name Category ID Category Name Lowest Cost**

**---------- --------------------- -------------- ------------------- --------------**

**1 Southlake, Texas 2 Video Card $535.03**

**2 San Francisco 2 Video Card $521.03 2 San Francisco 5 Storage $12**

1. Display product Id, name, and list Price for products that were purchased in orders handled by salesman Marshall and with list price greater than all average list prices per each category. Sort the output by Id ascending.

Solution:

select product\_id AS "ProdId", product\_name AS "Product Name", list\_price AS "LPrice"

from order\_items

join products using (product\_id)

join orders using (order\_id)

where salesman\_id =( select employee\_id

from employees

where upper(last\_name) like '%MARSHALL%')

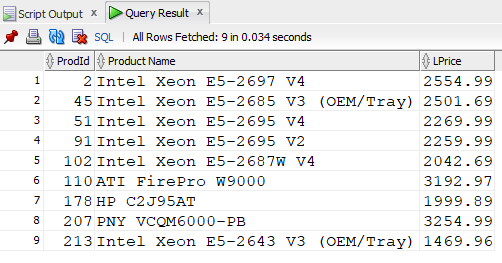
and list\_price >all (select avg(list\_price)

from products

group by category\_id)

order by 1 asc ;

OUTPUT:



**ProdId Product Name LPrice**

**---------- -------------------- --------**

**2 Intel Xeon E5-2697 V 2554.99**

**45 Intel Xeon E5-2685 V 2501.69**

1. Display customer Id, name, and total number of orders, for orders handled by salesman Marshall, but only if customer name begins on General or ends on Electric. Exclude customers who placed a single order, but include customers without orders as well. Sort the result based on the total number of orders descending and then by name ascending.

Do not use LIKE operator and do not join 3 tables.

select a.customer\_id AS "CUSTOMER\_ID" , a.name AS "Customer Name" ,count(b.order\_id) AS "# of Orders"

from customers a left

join orders b on a.customer\_id = b.customer\_id

and b.salesman\_id in (select b.salesman\_id

from employees

where last\_name = 'Marshall')

where (substr(a.name,1,7) = 'General'

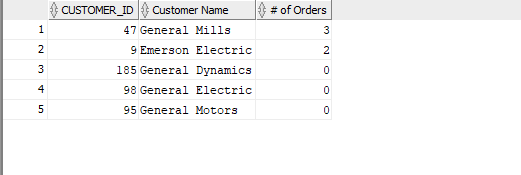
OR substr(a.name,-8,8) = 'Electric')

group by a.customer\_id, a.name

having count(b.order\_id)<>1

order by count(b.order\_id) desc, a.name;

OUTPUT:



**CUSTOMER\_ID Customer Name # of Orders**

**----------- -------------------- ------------**

**47 General Mills 3**

1. **Emerson Electric 2**

**185 General Dynamics 0**

1. Display product Id, name, and list Price 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.

SELECT

product\_id AS "Product iD",

product\_name AS "Product Name",

to\_char(list\_price, '$9,999.99') AS "List 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'

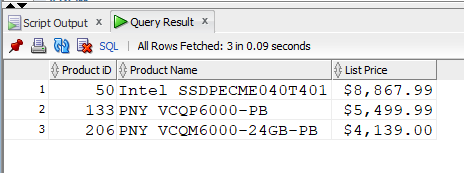
GROUP BY

w.warehouse\_id)

ORDER BY

list\_price DESC;

OUTPUT:



**Product ID Product Name List Price**

**---------- -------------------- ------------**

**50 Intel SSDPECME040T40 $8,867.99**

1. Display product Id, name, and list Price for the most expensive product, then the cheapest product and also for the product with the price closest to the average product price (rounded to the nearer ten). For the third row exclude products with name that starts on Intel. Here is the exact output.

SELECT product\_id AS "Product ID",

product\_name AS "Product Name",

TO\_CHAR(list\_price, '$9,999.99') AS "Price"

FROM products

WHERE product\_id = 50

UNION

SELECT product\_id AS "Product ID",

product\_name AS "Product Name",

TO\_CHAR(list\_price, '$9,999.99') AS "Price"

FROM products

WHERE list\_price = (

SELECT MIN(list\_price)

FROM products)

GROUP BY product\_id, product\_name, list\_price

UNION

SELECT product\_id AS "Product ID",

product\_name AS "Product Name",

TO\_CHAR(list\_price, '$9,999.99') AS "Price"

FROM products

WHERE ROUND(list\_price, -1) = (

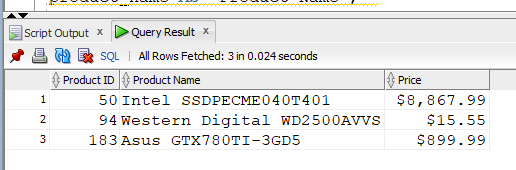
SELECT ROUND(AVG(list\_price), -1)

FROM products)

AND product\_name NOT LIKE 'Intel%'

GROUP BY product\_id, product\_name, list\_price;

OUTPUT:



**Product ID Product Name Price**

**---------- -------------------- -----------**

**50**   **Intel SSDPECME040T40 $8,867.99**

**94**   **Western Digital WD25 $15.55**

**183**   **Asus GTX780TI-3GD5 $899.99**