Week 5: SQL Queries

Query for the following information:

a. List all the customers from Ada, MI.

```
SELECT * FROM customer WHERE city='Ada' AND state='MI';
```

Output:

CUSTOMERID	LASTNAME	FIRSTNAME	STREETADDRESS	CITY	STATE	ZIPCODE	CURRENTBALANCE	CREDITLIMIT	SALESREPID
522	Nelson	Mary	108 Pine	Ada	MI	49441	98.75	1500	2
587	Galvez	Mara	512 Pine	Ada	MI	49441	114.6	1000	6

b. List all the customers in which the sales representative is from another city.

```
SELECT DISTINCT Customer.*,
SalesrepResentative.CITY as SalesRep_City
FROM
Customer
JOIN SalesrepResentative
ON Customer.SALESREPID=SalesrepResentative.SALESREPID
AND Customer.CITY <> SalesrepResentative.CITY;
```

Output:

CUSTOMERID	LASTNAME	FIRSTNAME	STREETADDRESS	CITY	STATE	ZIPCODE	CURRENTBALANCE	CREDITLIMIT	SALESREPID	SALESREP_CITY
256	Waters	Cheryl	215 Pete	Grant	MI	49219	21.5	1500	6	Ada
315	Daniels	Tom	914 Cherry	Kent	MI	48391	770.75	750	6	Ada
412	Adams	Sally	16 Elm	Lansing	MI	49224	1817.5	2000	3	Grant
567	Dinh	Tran	808 Ridge	Harper	MI	48421	402.4	750	6	Ada
311	Charles	Don	48 College	Ira	MI	49034	825.75	1000	12	Lansing
124	Mohnani	Payal	481 Oak	Lansing	MI	49224	818.75	1000	3	Grant
405	Williams	Al	519 Watson	Grant	MI	49219	402.75	1500	12	Lansing

c. List all the orders for customer 124.

```
SELECT Sales_Order.*
FROM
Sales_Order
WHERE
CustomerID=124;
```

Output:

ORDERID	ODATE	CUSTOMERID	SHIPPINGDATE
12489	02-JUL-11	124	22-JUL-11
12500	05-JUL-11	124	22-AUG-11

d. List all the parts orders by customer 124.

```
SELECT
DISTINCT  Sales_Order.CustomerId,
PART.*
FROM
Sales_Order
JOIN OrderLines
ON Sales_Order.orderid=OrderLines.orderid
JOIN Part
ON OrderLines.partid=Part.partid
WHERE
Sales_Order.CustomerID=124;
```

Output:

CUSTOMERID	PARTID	PARTDESCRIPTION	UNITEONHAND	PCLASS	WAREHOUSENUMBER	UNITPRICE
124	BT04	Gas Grill	11	AP	2	149.99
124	AX12	Iron	104	HW	3	24.95

e. For each sales representative, identify the number of customers, total sales of the customers and the number of parts ordered by the customers.

```
SELECT
TEMP_SALESREP.SALESREPID,
COUNT(DISTINCT NVL(Customer.CustomerID,0)) as SR_Customer_Count,
SUM(NVL(OrderLines.numberordered,0) * NVL(OrderLines.quotedprice,0)) as SR_Sales_D
ollars,
SUM(NVL(numberordered,0)) as SR_Parts_Sold

FROM
(SELECT SALESREPID FROM SalesrepResentative GROUP BY SALESREPID) TEMP_SALESREP
LEFT JOIN
customer ON TEMP_SALESREP.SALESREPID=Customer.SALESREPID
LEFT JOIN
Sales_Order ON Sales_Order.customerid=Customer.customerid
LEFT JOIN
OrderLines ON OrderLines.orderid=Sales_Order.orderid
GROUP BY TEMP_SALESREP.SALESREPID;
```

Output:

SALESREPID	SR_CUSTOMER_COUNT	SR_SALES_DOLLARS	SR_PARTS_SOLD
6	4	1165.86	6
12	2	549.98	2
3	3	391.44	12

Query one table and use WHERE to filter the results. The SELECT clause should have a column list, not an asterisk (*). State the purpose of the query; show the query and the output.

```
SELECT
   partid  , partdescription , uniteonhand ,
   pclass  , warehousenumber , unitprice
FROM
   PART
WHERE
   UnitPrice > 30;
```

Purpose of the Query is to retrieve records from PART table which have UnitPrice value greater than 30 Dollars.

Output:

PARTID	PARTDESCRIPTION	UNITEONHAND	PCLASS	WAREHOUSENUMBER	UNITPRICE
BT04	Gas Grill	11	AP	2	149.99
BZ66	Washer	52	AP	3	39.99
CA14	Griddle	78	HW	3	39.99
CB03	Bike	44	SG	1	299.99
CZ81	Treadmill	68	SG	2	349.95

Get information from at least 3 tables in one statement, and provide the output using the Join operator. Use ANSI Join syntax. State the purpose of the query; show the query and the output. Add a screen shot of SS Management Studio showing the query and results.

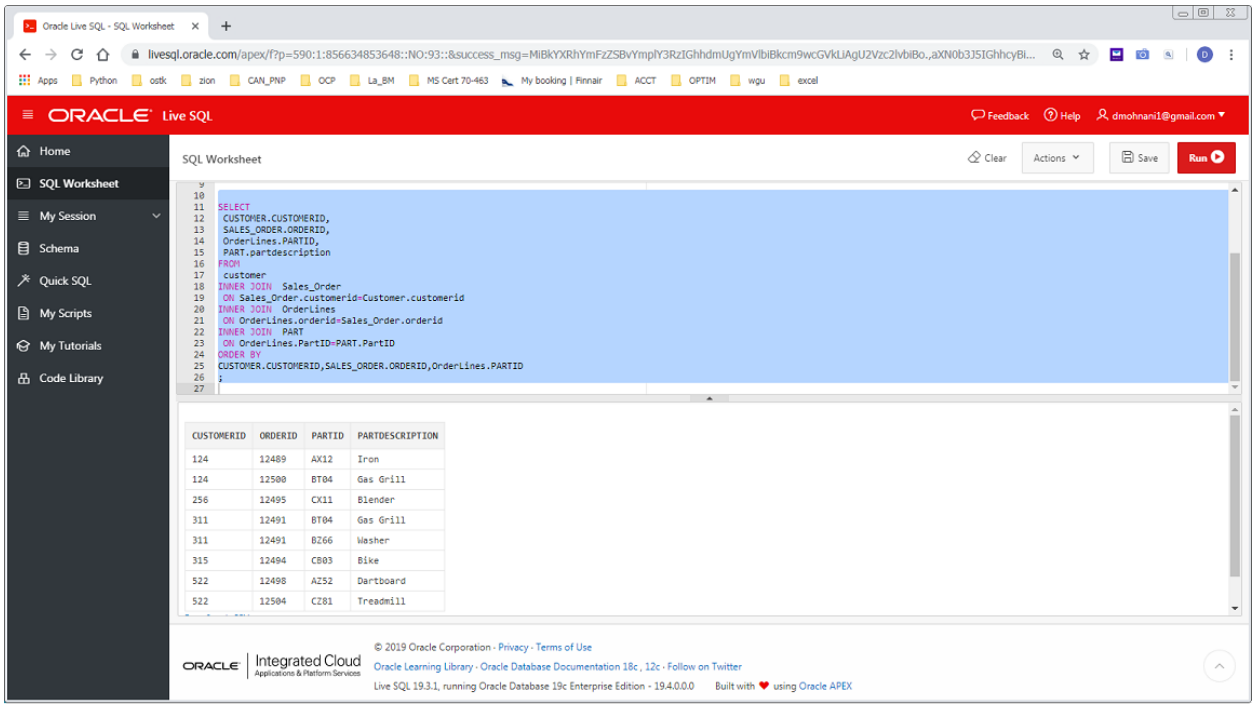
Purpose: Purpose of below Query is to display CustomerIDs, OrderIDs of the orders placed by each customer, the various products/parts in each sales order and the description of the part. This Information is pulled from 4 tables using three INNER JOIN statements.

```
SELECT
 CUSTOMER.CUSTOMERID,
 SALES_ORDER.ORDERID,
 OrderLines.PARTID,
 PART.partdescription
FROM
 customer
INNER JOIN  Sales_Order
 ON Sales_Order.customerid=Customer.customerid
INNER JOIN  OrderLines
 ON OrderLines.orderid=Sales_Order.orderid
INNER JOIN  PART
 ON OrderLines.PartID=PART.PartID
ORDER BY
CUSTOMER.CUSTOMERID,SALES_ORDER.ORDERID,OrderLines.PARTID
;
```

Output:

CUSTOMERID	ORDERID	PARTID	PARTDESCRIPTION
124	12489	AX12	Iron
124	12500	BT04	Gas Grill
256	12495	CX11	Blender
311	12491	BT04	Gas Grill
311	12491	BZ66	Washer
315	12494	CB03	Bike
522	12498	AZ52	Dartboard
522	12504	CZ81	Treadmill

Output Screenshot: Using Free Online Oracle Live SQL



Get information from **2** tables in one statement, and provide the output using the Left Outer Join operator. State the purpose of the query; show the query and the output. The outer join should be designed to retrieve information from the left table that has no matches in the right table. If that is not possible for your database, explain why.

Purpose: The Purpose of this (2 table) query is to list products in decreasing order of popularity.

Query:

```
SELECT
  PART.PARTID,
  PART.PARTDESCRIPTION,
  PART.PCLASS,
  SUM (NVL(OrderLines.numberordered,0)) AS PART_QTY_SOLD
FROM
  PART
LEFT JOIN OrderLines ON PART.PartID = OrderLines.PARTID
GROUP BY
  PART.PARTID,
  PART.PARTDESCRIPTION,
  PART.PCLASS
ORDER BY PART_QTY_SOLD DESC
```

Output:

PARTID	PARTDESCRIPTION	PCLASS	PART_QTY_SOLD
AX12	Iron	HW	11
CB03	Bike	SG	4
BT04	Gas Grill	AP	2
CZ81	Treadmill	SG	2
CX11	Blender	HW	2
AZ52	Dartboard	SW	2
BZ66	Washer	AP	1
BH22	Corn Popper	HW	0
CA14	Griddle	HW	0

Create a query using the IN keyword with a sub-query. State the purpose of the query; show the query and the output.

Purpose: The below Query utilized Correlated Sub query with IN clause to display Customer records having same zip code as their Sales Representative.

```
SELECT *
FROM
CUSTOMER
WHERE CUSTOMER.SalesrepID IN
(SELECT
   SalesRepID
 From
   SalesrepResentative
 WHERE
   Customer.Zipcode = SalesrepResentative.ZipCode);
```

Output:

CUSTOMERID	LASTNAME	FIRSTNAME	STREETADDRESS	CITY	STATE	ZIPCODE	CURRENTBALANCE	CREDITLIMIT	SALESREPID
587	Galvez	Mara	512 Pine	Ada	MI	49441	114.6	1000	6

6. Create a query using an aggregate function (i.e., min, max, avg, sum, count) and the GROUP BY command. State the purpose of the query; show the query and the output.

Purpose: The Purpose of below Query is to display aggregate of Sale Dollars (Total_Sale), Average Size of each order and Counts of the distinct Orders of each day, in the decreasing order of Total_Sale.

```
Select
S.ODATE,
SUM (L.QUOTEDPRICE * L.NUMBERORDERED) AS Total_Sale,
AVG(L.QUOTEDPRICE * L.NUMBERORDERED) AS Avreage_Order_Size,
COUNT(DISTINCT S.ORDERID) AS Total_Order_Count
FROM
  Sales_order S
JOIN
  OrderLines L
ON S.ORDERID=L.OrderID
GROUP BY S.ODATE
ORDER BY TOTAL_SALE DESC
;
```

Output:

ODATE	TOTAL_SALE	AVREAGE_ORDER_SIZE	TOTAL_ORDER_COUNT
04-JUL-11	1165.86	582.93	2
05-JUL-11	827.87	275.9567	3
02-JUL-11	791.43	263.81	2

7. Create a query using an aggregate function (i.e., min, max, avg, sum, count) and the GROUP BY command using the HAVING clause to filter the aggregate results. State the purpose of the query; show the query and the output.

Purpose: The Purpose of below Query is to display aggregate of Sale Dollars (Total_Sale), Average Size of each order and Counts of the distinct Orders of each day only for those days when average order size was more than 300 dollars.

```
Select
  S.ODATE,
  SUM (L.QUOTEDPRICE * L.NUMBERORDERED) AS Total_Sale,
  AVG(L.QUOTEDPRICE * L.NUMBERORDERED) AS Avreage_Order_Size,
  COUNT(DISTINCT S.ORDERID) AS Total_Order_Count
```

```
from
  Sales_order S
JOIN
  OrderLines L
ON S.ORDERID=L.OrderID
GROUP BY
S.ODATE
HAVING AVG(L.QUOTEDPRICE * L.NUMBERORDERED)   >= 300
;
```

Output:

ODATE	TOTAL_SALE	AVREAGE_ORDER_SIZE	TOTAL_ORDER_COUNT
04-JUL-11	1165.86	582.93	2

References:

Oppel, A. J. (2009). *Databases: A beginner's guide*. New York, NY: McGraw-Hill.

Laureate Education (Producer). (2011a). *Designing a database* [Video file]. Baltimore, MD: Author