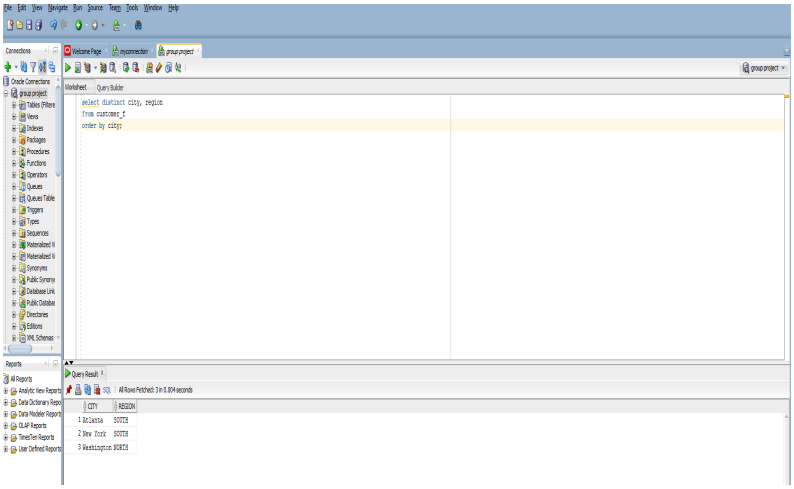
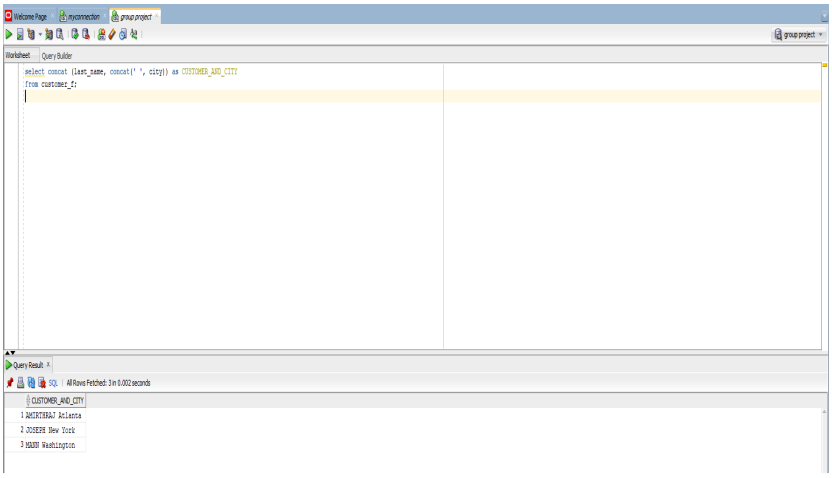
Database Management Systems Project

Garrett Kilcer

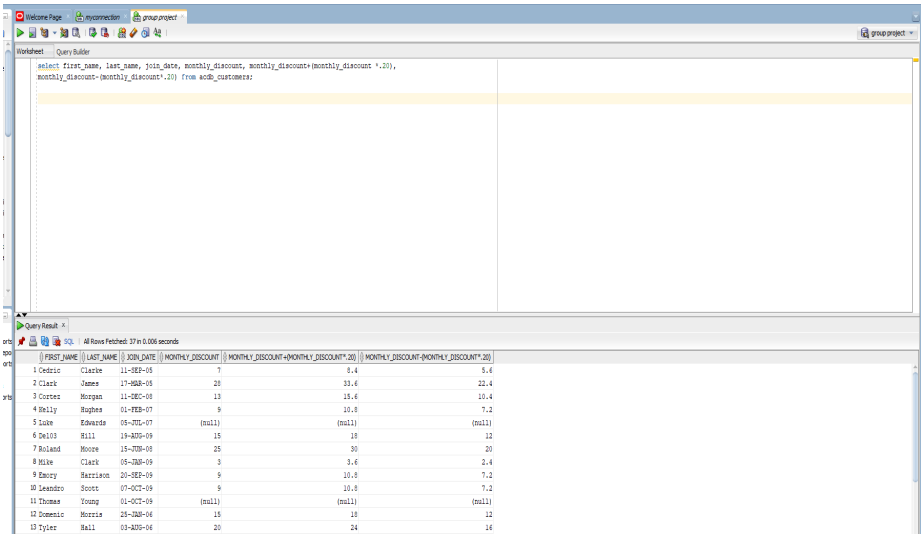
a. Create a query to display unique combination of cities and states from Customers table
select distinct city, region
from customer_f
order by city;



b. Create a query to display the last name concatenated with the city, separated by space. Name this column CUSTOMER_AND_CITY
select concat (last_name, concat(' ', city)) as CUSTOMER_AND_CITY
from customer_f;

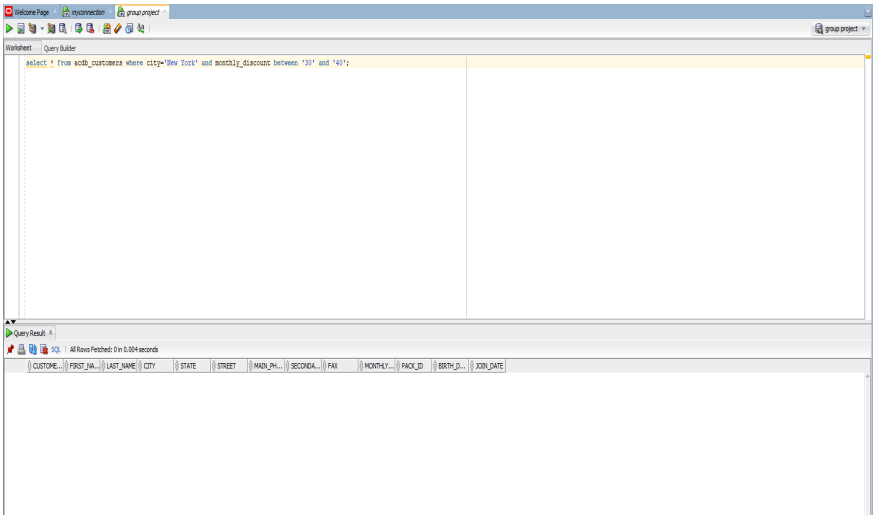


c. Create a query to display first name, last name, join date, monthly discount, monthly discount after an addition of 20% and monthly discount after a reduction of 20%.
select first_name, last_name, join_date, monthly_discount,
monthly_discount+(monthly_discount *.20),
monthly_discount-(monthly_discount*.20) from acdb_customers;



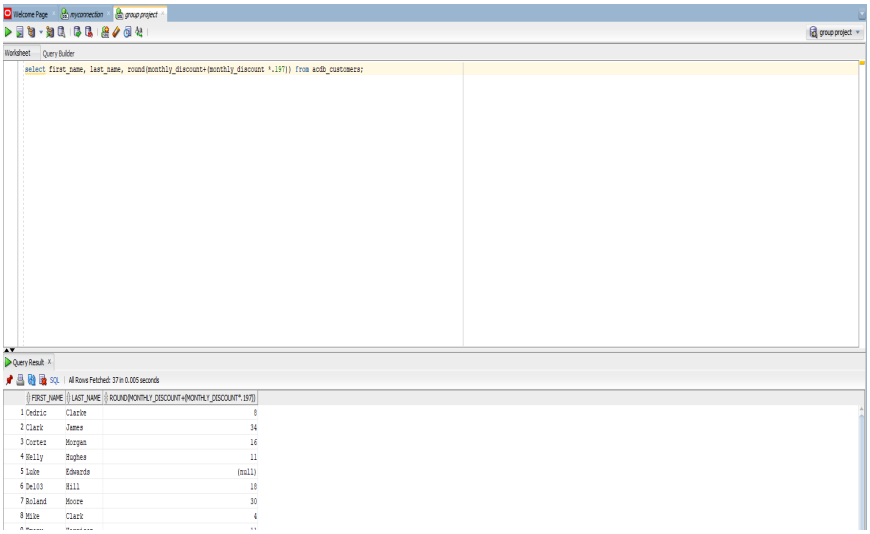
d. All customers who live in New York **and** whose monthly discount is in the range between 30 and 40

select * from acdb_customers where city='New York' and monthly_discount between '30' and '40';



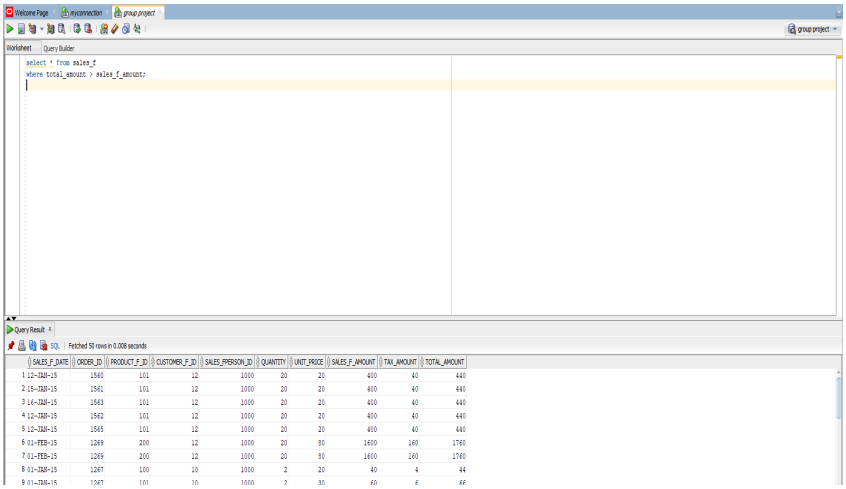
e. From customers table, display monthly discount after addition of 19.7%, expressed as a whole number

select first_name, last_name, round(monthly_discount+(monthly_discount *.197)) from acdb_customers;



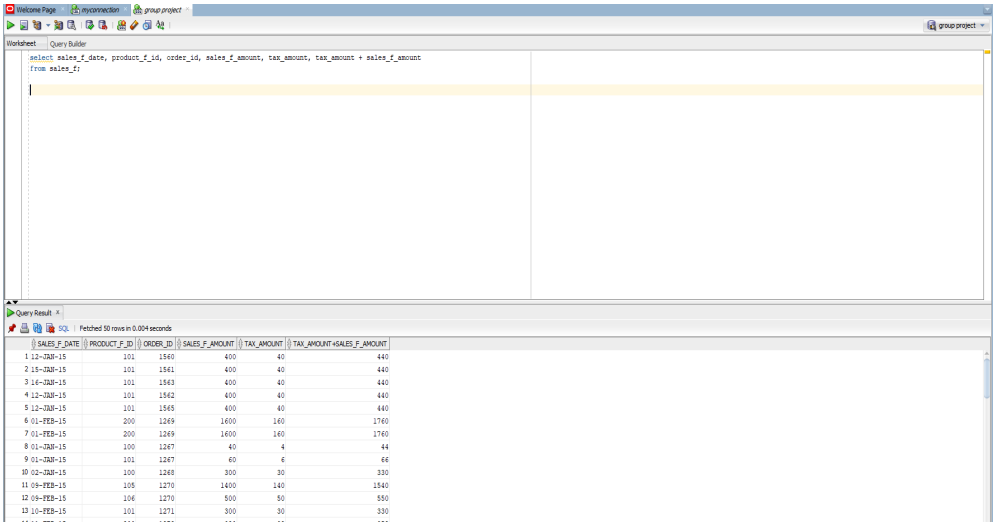
f. Produce a list of rows showing all columns from the SALES table where total amount is greater than sales amount.

select * from sales_f
where total_amount > sales_f_amount;



g. Produce a list of rows showing sales date, product number, order number, sales amount, tax amount and a calculated column (sum of sales amount and tax amount) from the SALES table

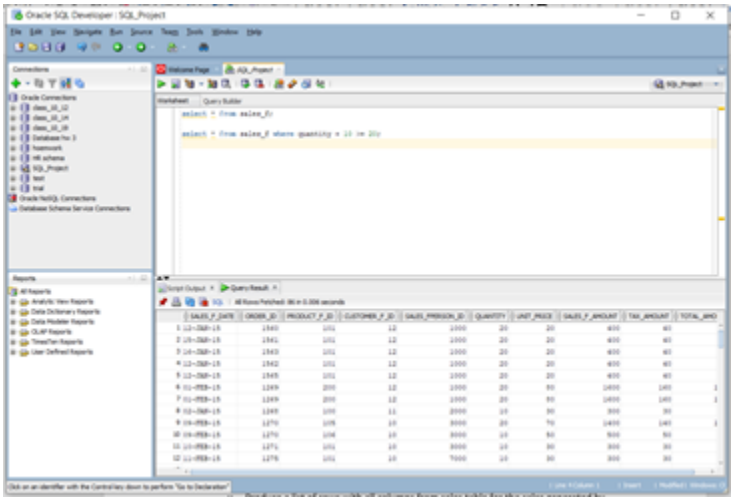
select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount, tax_amount + sales_f_amount
from sales_f;



	SALES_F_DATE	PRODUCT_F_ID	ORDER_ID	SALES_F_AMOUNT	TAX_AMOUNT	TAX_AMOUNT+SALES_F_AMOUNT
1	12-JAN-15	101	1560	400	40	440
2	15-JAN-15	101	1561	400	40	440
3	16-JAN-15	101	1563	400	40	440
4	12-JAN-15	101	1562	400	40	440
5	12-JAN-15	101	1565	400	40	440
6	01-FEB-15	200	1269	1600	160	1760
7	01-FEB-15	200	1269	1600	160	1760
8	01-JAN-15	100	1267	40	4	44
9	01-JAN-15	101	1267	60	6	66
10	02-JAN-15	100	1268	300	30	330
11	09-FEB-15	105	1270	1400	140	1540
12	09-FEB-15	106	1270	500	50	550
13	10-FEB-15	101	1271	300	30	330

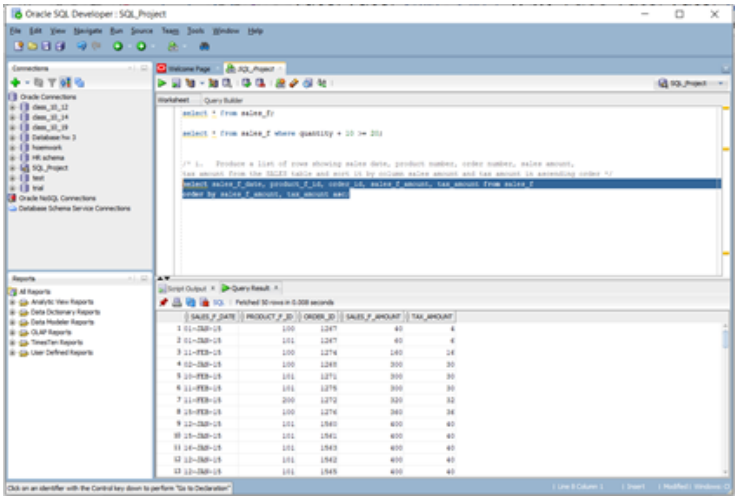
h. Produce a list of rows showing all columns from the SALES table where quantity plus 10 is greater than or equal to 20

select * from sales_f where quantity + 10 >= 20;

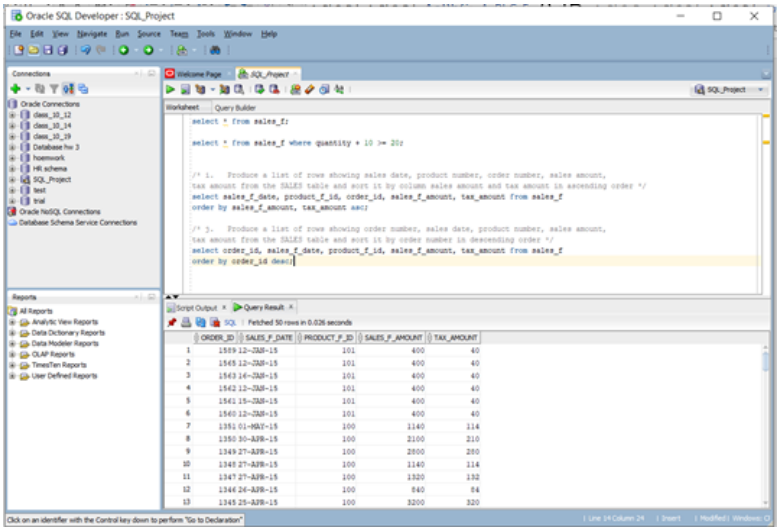


i. Produce a list of rows showing sales date, product number, order number, sales amount, tax amount from the SALES table and sort it by column sales amount and tax amount in ascending order

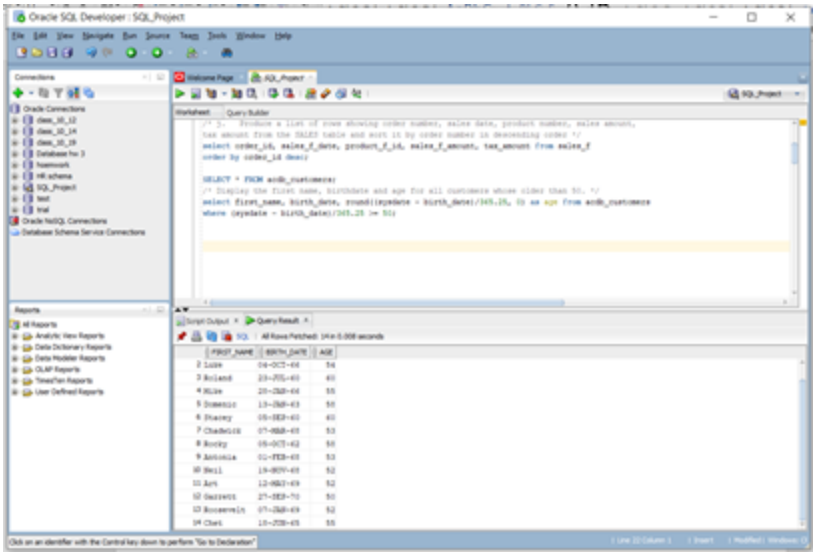
select sales_f_date, product_f_id, order_id, sales_f_amount, tax_amount from sales_f order by sales_f_amount, tax_amount asc;



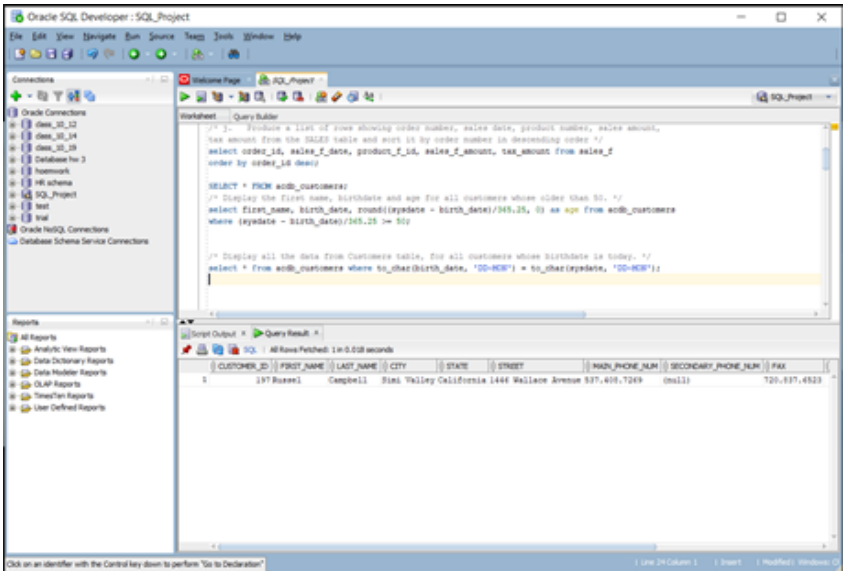
j. Produce a list of rows showing order number, sales date, product number, sales amount, tax amount from the SALES table and sort it by order number in descending order
select order_id, sales_f_date, product_f_id, sales_f_amount, tax_amount from sales_f order by order_id desc;



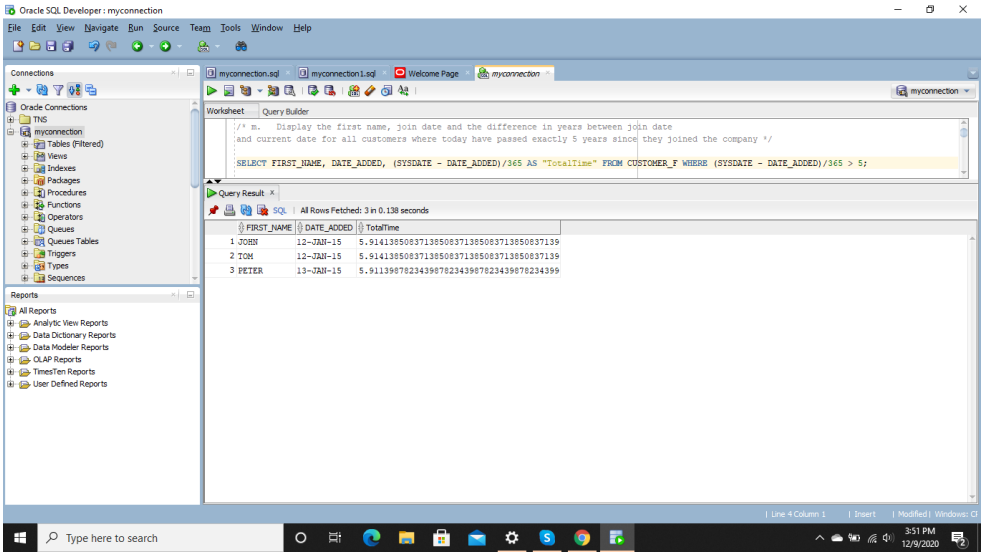
k. Display the first name, birthdate and age for all customers whose older than 50.
select first_name, birth_date, round((sysdate - birth_date)/365.25, 0) as age from acdb_customers where (sysdate - birth_date)/365.25 >= 50;



l. Display all the data from Customers table, for all customers whose birthdate is today.
select * from acdb_customers where to_char(birth_date, 'DD-MON') = to_char(sysdate, 'DD-MON');

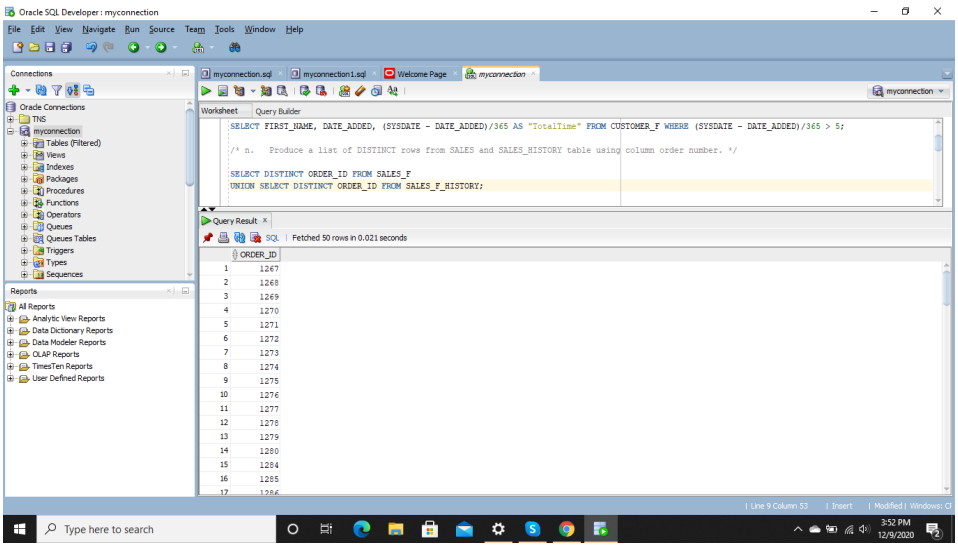


m. Display the first name, join date and the difference in years between join date and current date for all customers where today have passed exactly 5 years since they joined the company.



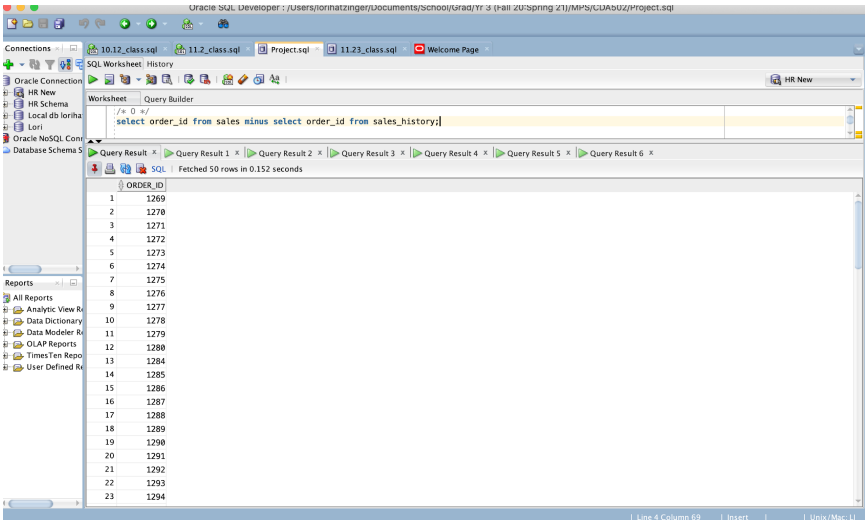
SELECT FIRST_NAME, DATE_ADDED, (SYSDATE - DATE_ADDED)/365 AS "TotalTime" FROM CUSTOMER_F WHERE (SYSDATE - DATE_ADDED)/365 > 5;

n. Produce a list of DISTINCT rows from SALES and SALES_HISTORY table using column order number.



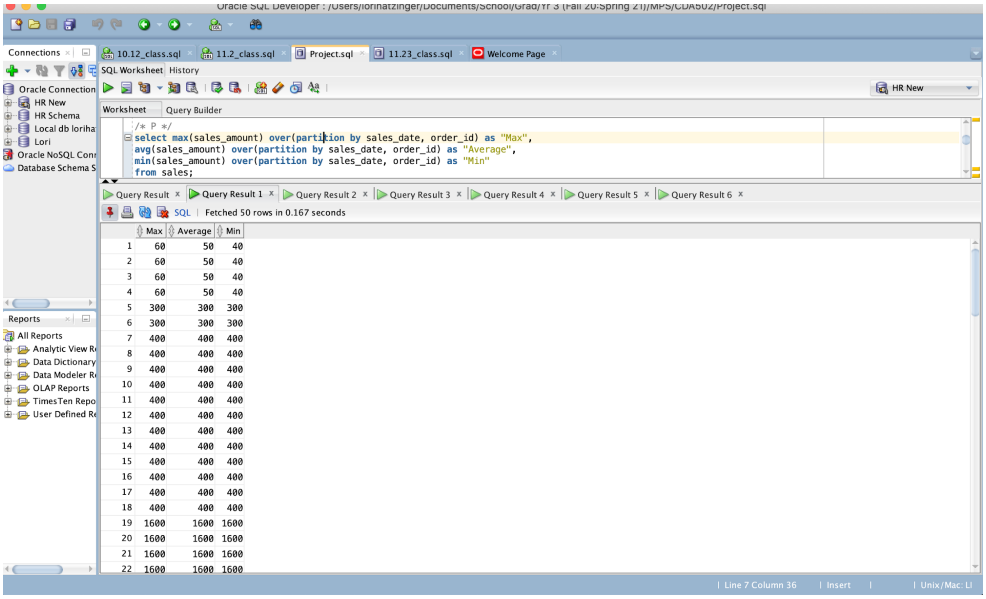
SELECT DISTINCT ORDER_ID FROM SALES_F UNION SELECT DISTINCT ORDER_ID FROM SALES_F_HISTORY;

o. Produce a list of rows which are present in SALES table and are not present in SALES_HISTORY table using column order number.
select order_id from sales minus select order_id from sales_history;



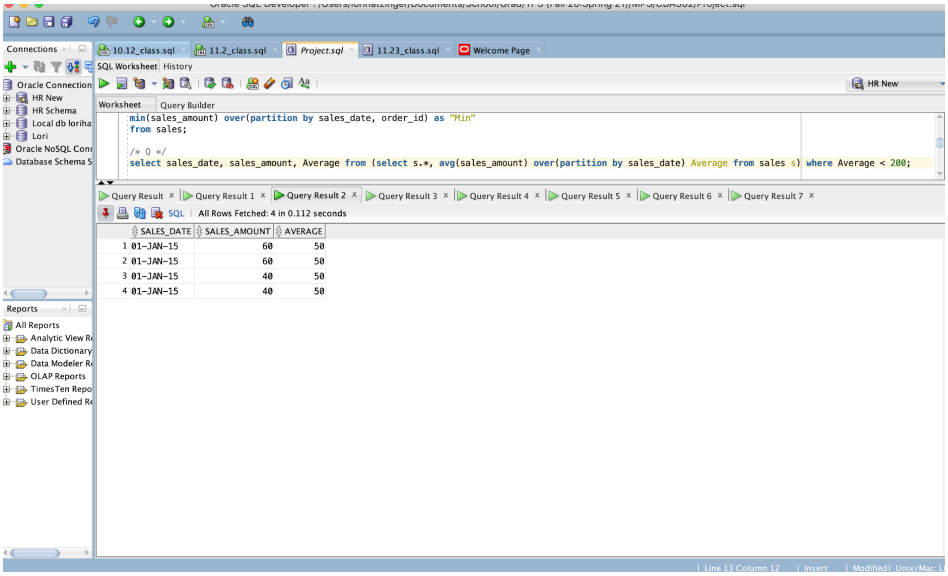
p. Produce a list which shows maximum, average, minimum sales volume broken by sales date and order number from SALES table

```
select max(sales_amount) over(partition by sales_date, order_id) as "Max",
avg(sales_amount) over(partition by sales_date, order_id) as "Average",
min(sales_amount) over(partition by sales_date, order_id) as "Min"
from sales;
```

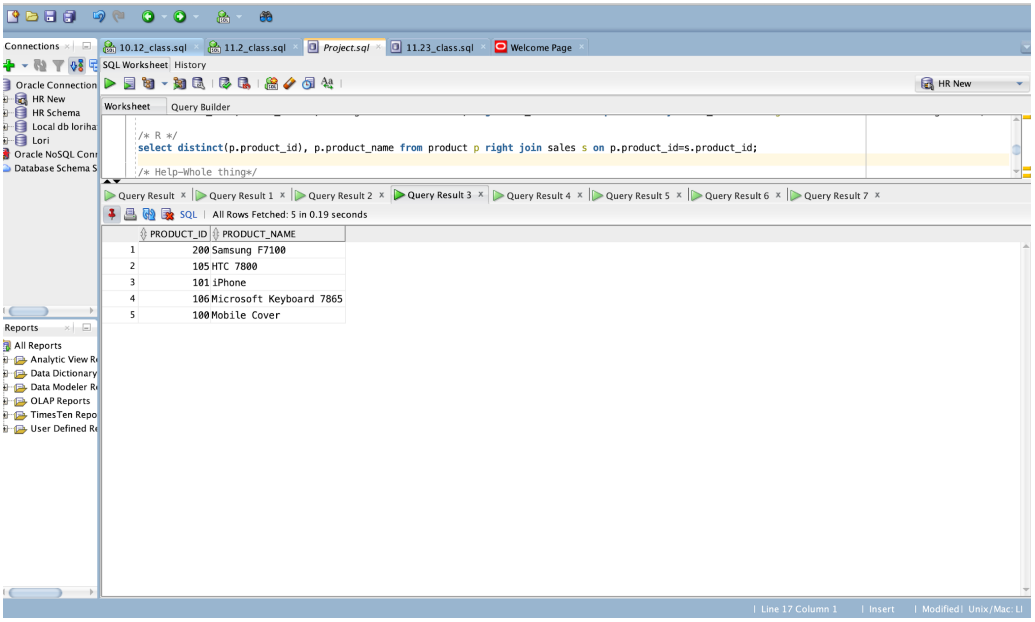


q. Produce a list which shows average sales volume broken by sales date from SALES table and average sales volume is less than 200.

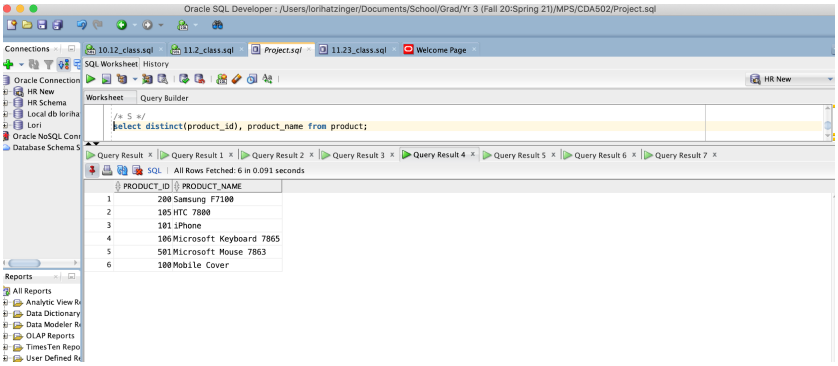
```
select sales_date, sales_amount, Average from (select s.*, avg(sales_amount)
over(partition by sales_date) Average from sales s) where Average < 200;
```



r. Produce a list which returns all products only those products that have had some sales
select distinct(p.product_id), p.product_name from product p right join sales s on p.product_id=s.product_id;

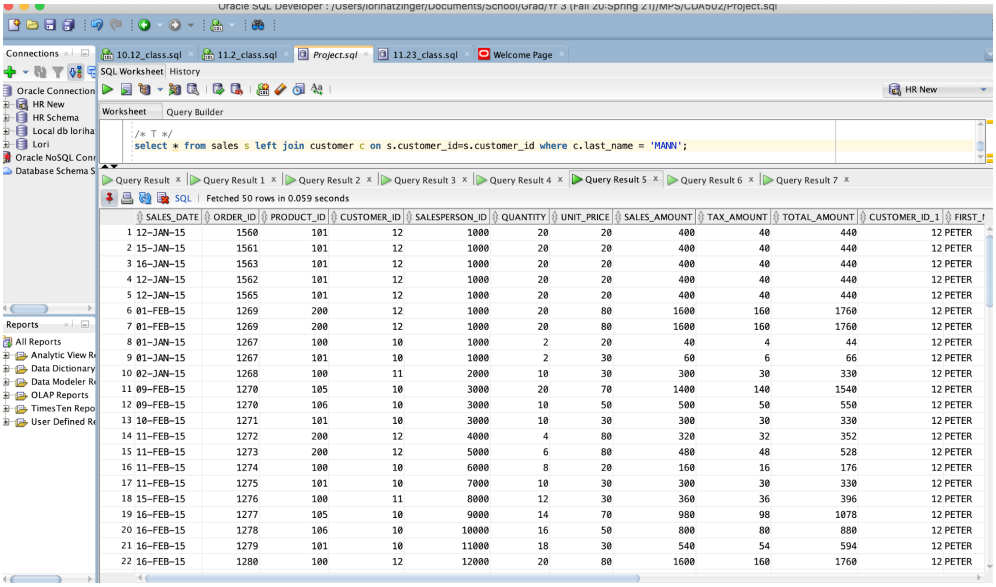


s. Produce a list which returns all products irrespective of whether the product has had any sales
select distinct(product_id), product_name from product;



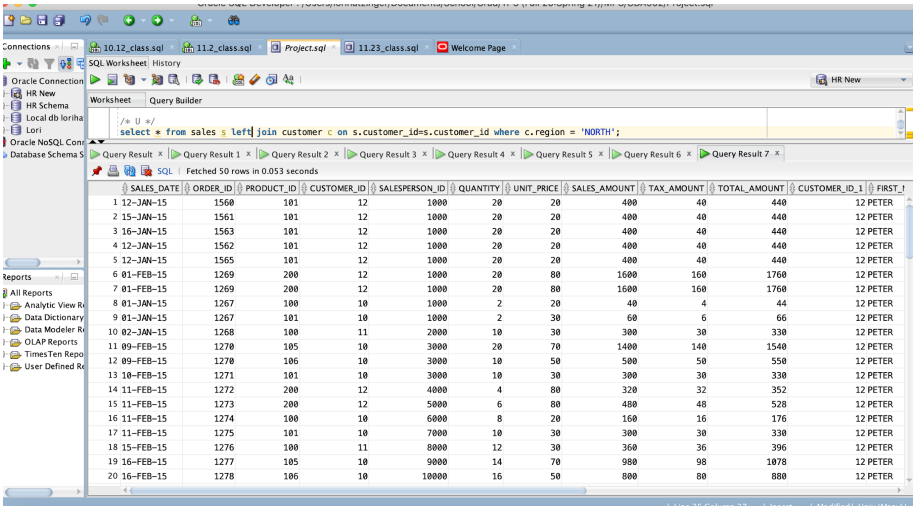
t. Produce a list of rows with all columns from sales table for the sales generated by customer MANN

select * from sales s left join customer c on s.customer_id=s.customer_id where c.last_name = 'MANN';



u. Produce a list of rows with all columns from sales table for the sales generated by customers in North region

select * from sales s left join customer c on s.customer_id=s.customer_id where c.region = 'NORTH';



v. Produce a list of rows from the sales table where the selected total_amount is greater than the average total_amount of their respective customers

```
select * from sales_f e1
where e1.total_amount>
(select avg(e2.total_amount) from sales_f e2 where e1.customer_f_id=e2.customer_f_id
group by e2.customer_f_id);
```

SALES_F_DATE	ORDER_ID	PRODUCT_F_ID	CUSTOMER_F_ID	SALES_FPERSON_ID	QUANTITY	UNIT_PRICE	SALES_F_AMOUNT	TAX_AMOUNT	TOTAL_AMOUNT
25-FEB-15	1288	106	12	9000	36	80	2880	288	3168
28-FEB-15	1289	101	12	10000	38	80	3040	304	3344
02-MAR-15	1293	106	10	1000	46	70	3220	322	3542
02-MAR-15	1294	101	10	2000	48	50	2400	240	2640
07-MAR-15	1296	105	12	9000	52	80	4160	416	4576
08-MAR-15	1297	106	12	10000	54	80	4320	432	4752
08-MAR-15	1301	105	10	1000	34	70	2380	238	2618
08-MAR-15	1302	106	10	1000	36	50	1800	180	1980
15-MAR-15	1304	101	10	2000	40	80	3200	320	3520
16-MAR-15	1305	100	12	3000	42	80	3360	336	3696
19-MAR-15	1309	106	11	2000	50	70	3500	350	3850
19-MAR-15	1310	101	12	3000	52	50	2600	260	2860
23-MAR-15	1312	100	10	10000	34	80	2720	272	2992

w. Write a query to bring the names of the salespersons and their manager

```
select e1.first_name ||''|| e1.last_name as salespersons,e2.manager from
sales_fperson e1 join sales_fperson e2 on e1.manager=e2.first_name where e1.manager
is not null;
```

SALESPERSONS	MANAGER
SonuAgarwal	Tom
PatelHari	Tom
JamesGavin	Tom
RehmanAhmed	Raj
SaraK	Raj
RajKishore	-
TomJoseph	-
BobMoris	Jeff
GregChappel	Jeff
AnilKrishna	Raj
PeterMann	Jeff
JohnKing	Jeff

Download CSV

x. Product the list of top 10 products based on the number of time they have been sold
select * from(
select distinct product_f_id, product_f_name, sum(sales_f.quantity) over(partition by product_f_id)
total_sale from product_f join sales_f using (product_f_id) order by total_sale desc)
where rownum <=10;

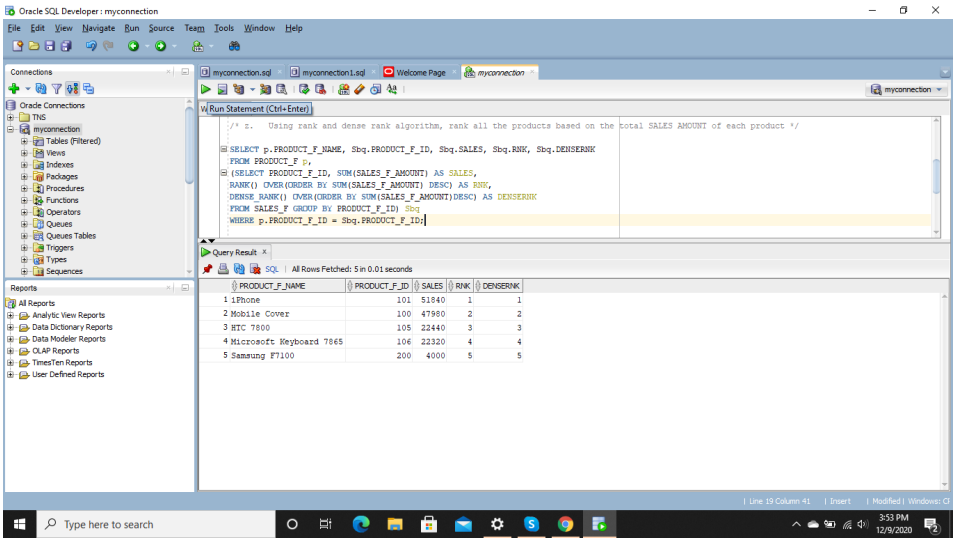
PRODUCT_F_ID	PRODUCT_F_NAME	TOTAL_SALE
101	iPhone	1236
100	Mobile Cover	944
105	HTC 7800	440
106	Microsoft Keyboard 7865	412
200	Samsung F7100	50

Download CSV

y. List all the products who price is greater than the average price of all the products in the respective categories
select * from product_f e1
where e1.list_price >
(select avg(e2.list_price) from product_f e2 where
e1.product_f_category=e2.product_f_category group by e2.product_f_category);

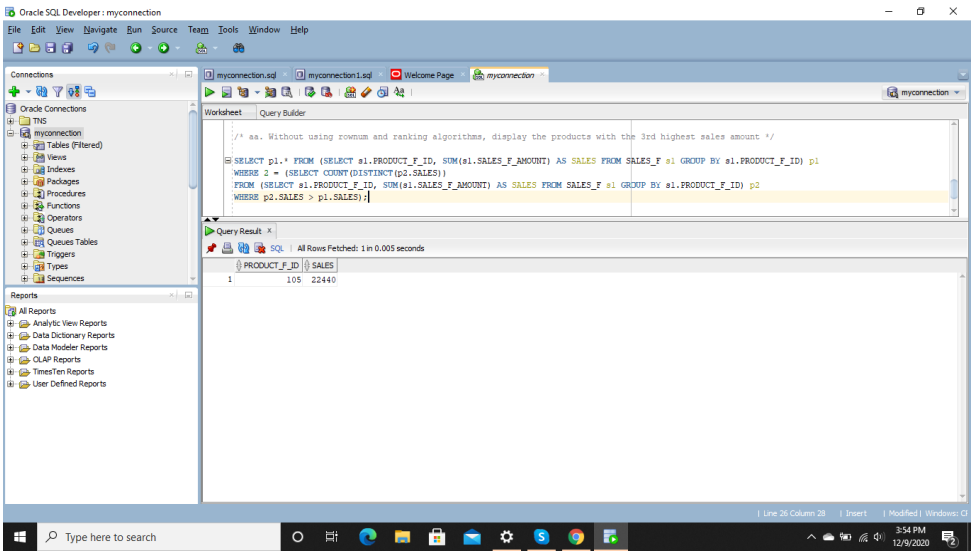
PRODUCT_F_ID	PRODUCT_F_NAME	STANDARD_COST	COLOR	LIST_PRICE	PRODUCT_F_SIZE	WEIGHT	PRODUCT_F_CATEGORY
106	Microsoft Keyboard 7865	50	BLACK	60	3	20	Computer
101	iPhone	500	GOLD	600	6	20	Mobile

z. Using rank and dense rank algorithm, rank all the products based on the total SALES AMOUNT of each product



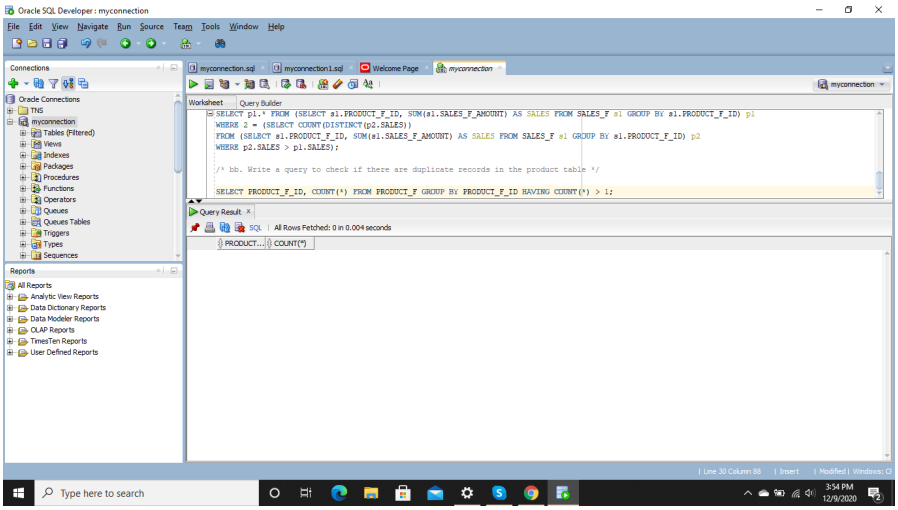
SELECT p.PRODUCT_F_NAME, Sbq.PRODUCT_F_ID, Sbq.SALES, Sbq.RNK, Sbq.DENSERNK FROM PRODUCT_F p, (SELECT PRODUCT_F_ID, SUM(SALES_F_AMOUNT) AS SALES, RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT) DESC) AS RNK, DENSE_RANK() OVER(ORDER BY SUM(SALES_F_AMOUNT)DESC) AS DENSERNK FROM SALES_F GROUP BY PRODUCT_F_ID) Sbq WHERE p.PRODUCT_F_ID = Sbq.PRODUCT_F_ID;

aa. Without using rownum and ranking algorithms, display the products with the 3rd highest sales amount



SELECT p1.* FROM (SELECT s1.PRODUCT_F_ID, SUM(s1.SALES_F_AMOUNT) AS SALES FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID) p1 WHERE 2 = (SELECT COUNT(DISTINCT(p2.SALES)) FROM (SELECT s1.PRODUCT_F_ID, SUM(s1.SALES_F_AMOUNT) AS SALES FROM SALES_F s1 GROUP BY s1.PRODUCT_F_ID) p2 WHERE p2.SALES > p1.SALES);

bb. Write a query to check if there are duplicate records in the product table



SELECT PRODUCT_F_ID, COUNT(*) FROM PRODUCT_F GROUP BY PRODUCT_F_ID HAVING COUNT(*) > 1;