Name: Emerson Drapac Test 3 (take Home Test) IT4/540

This test has the following parts

Part 1: Database (10%)

Part 2: Queries (40%)

Part 3: Relational Algebra, QBE, and SQL (15 %)

Part 4: Data Warehouse (15%)

Part 5: Object Relational Database (20%)

NOTE: ONLY graduate students or BS/MS students should answer the following Part

Part 6: ORDB extension

-------------------------------------------------------------------------------------------------------------------------------------------------

Part1 Database expansion and enhancement (10%)

P1.1 Existing Database (DB4)

P1.1.1 Submit the script for your database (it should run WITHOUT any error)

P1.1.2 Use Oracle Data modeler to develop current data model thru reverse engineering process DM\_DB4

P1.1.3 Develop a list to show how many records exists in every table of your database.

Note: Make sure your DB4 (ORDB) meets the following requirements

* + Have a Person type to include all person type entities in your database. Done
  + Have a type to hold different addresses that your person type may have. For this type, use nested table structure Done
  + Have a varray to hold different phone numbers. Done
  + Your data model to include object type as wells regular relational entities Done (update for REVIEW table and COLLEGE table)
  + Every object type should have a method. This method should a new one developed by you and it is different from in-class assignment and lecture notes. Done
  + Your scrip to include all types, methods, tables, and affected insert statements. Done
  + Develop a report to show number of records in each table Done

select

table\_name,

num\_rows counter

from

dba\_tables

where

owner = 'OO\_PAINT'

order by

table\_name;

P1.2 Modify your database to support the following new requirements

P1.2.1 Add employee work status to indicate if they are full time or seasonal. If they are season then what is the start and end date for each year of service (maybe make method to check if currently employed if time) Done

P1.2.2 Some of season employees are student intern from different colleges compare to regular seasonal employees. For college interns we want to keep track of their university and major. Done

P1.3 Modify your database as needed while solving parts 2, 3 and 4

if your database does not have the required entity and/or required attributes

if needs to have more records to produce results

* At the end of this part, you will have your final database script (DB5)
* Make sure your DB5 script works and does not have any error
* Use Oracle Data modeler to develop current data model thru reverse engineering process DM\_DB5
* Develop a list to show how many records exists in every table of your database.

-------------------------------------------------------------------------

Part2: Database Programming

P2.1 Using your database develop 5 queries to include the following requirements: (5%)

* Every query should join minimum of 3 tables
* All your queries should produce result
* Some of your queries should have some of the following commands. Your 5 queries, collectively, should cover all the following functions.
  + group by command, Done
  + RANK function, Done
  + NTILE function, Done
  + ROLL UP function with having, Done
  + SUBSTR function, Done
  + DECODE function to order based on Days of week, Done
  + CASE function Done
  + NVL function, Done
  + bottom and top number of records, Done
  + ratio-to percent, MONTH –BETWEEN, and NEXT-DAY Done

P2.1 Queries: Done

1. Show the top three customers according to their total sales.

With qty1 as

(

select CO.ORDER\_ID, SUM(P.PRICE) SUMS, OC.CUSTOMER\_ID

from PAINTING\_ORDER PO

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

JOIN CUSTOMER\_ORDER CO on PO.ORDER\_ID = CO.ORDER\_ID

JOIN O\_CUSTOMER OC on CO.CUSTOMER\_ID = OC.CUSTOMER\_ID

group by CO.ORDER\_ID, OC.CUSTOMER\_ID

)

SELECT t.rank,t.sums,t.cust\_id

FROM

(

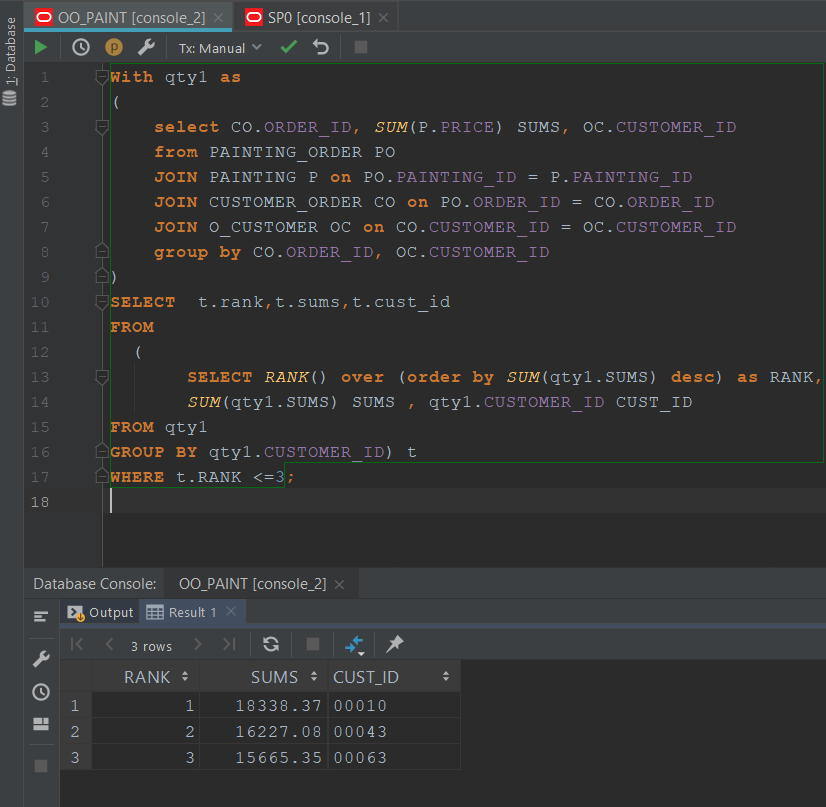
SELECT RANK() over (order by SUM(qty1.SUMS) desc) as RANK,

SUM(qty1.SUMS) SUMS , qty1.CUSTOMER\_ID CUST\_ID

FROM qty1

GROUP BY qty1.CUSTOMER\_ID) t

WHERE t.RANK <=3;



1. List top 5% of products (product id and number of items) sold in each month of 2016. Order your result based on month of the year from Jan to Dec.

select \*

from( select painting\_id, mon, count, Top\_Ranks

from(

select ol.painting\_id,substr(o.date\_created,4,3) mon, count(\*) count,

NTILE(20) OVER (ORDER BY count(\*) DESC) AS Top\_Ranks

from painting\_order ol join customer\_order o on o.order\_id=ol.order\_id

where substr(o.date\_created,8,2)=16 -- year 2016

group by ol.painting\_id, substr(o.date\_created,4,3))

where Top\_Ranks =1)

ORDER BY DECODE(mon,

'JAN',1,

'FEB',2,

'MAR',3,

'APR',4,

'MAY',5,

'JUN',6,

'JUL',7,

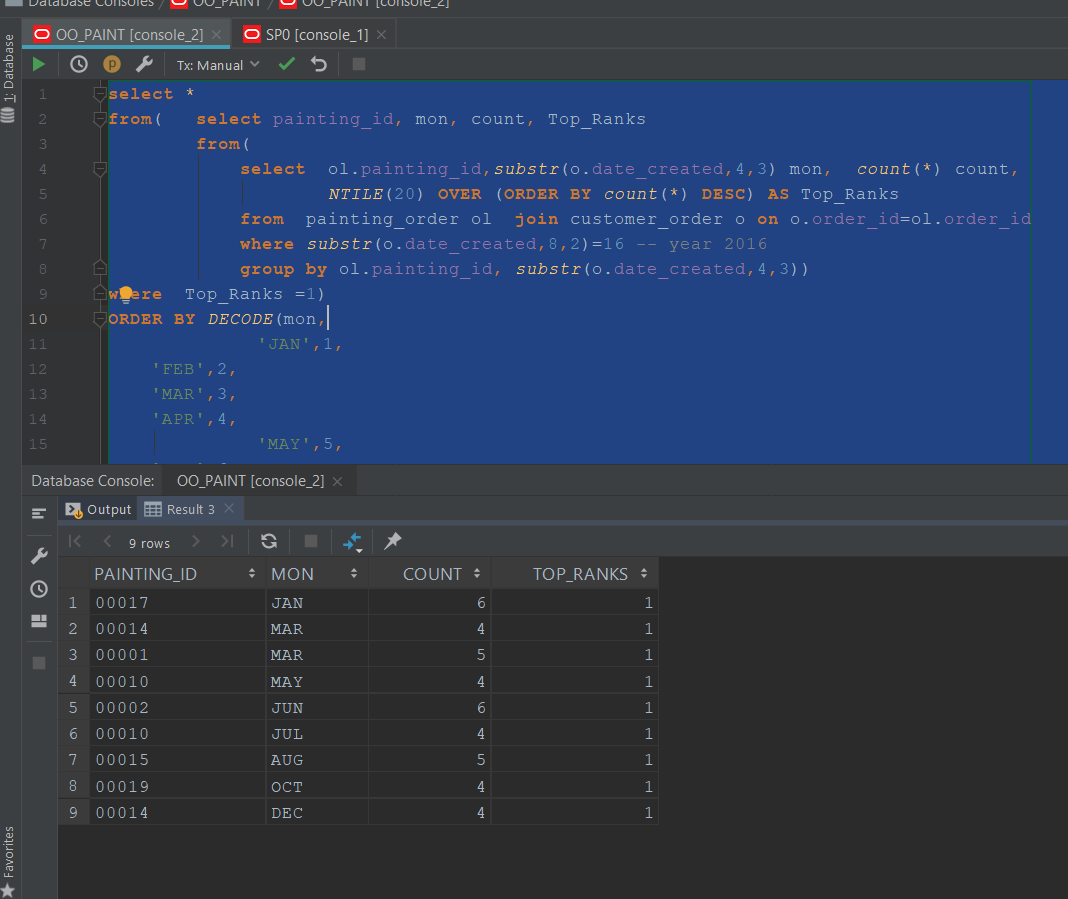
'AUG',8,

'SEP',9,

'OCT',10,

'NOV',11,

'DEC',12);



1. Get the subtotal of all Commissions of all the Employees who have a Commission listed, use NVL for those who don’t have a Commission to set their Commission Field to 0 and then exclude them from the query results.

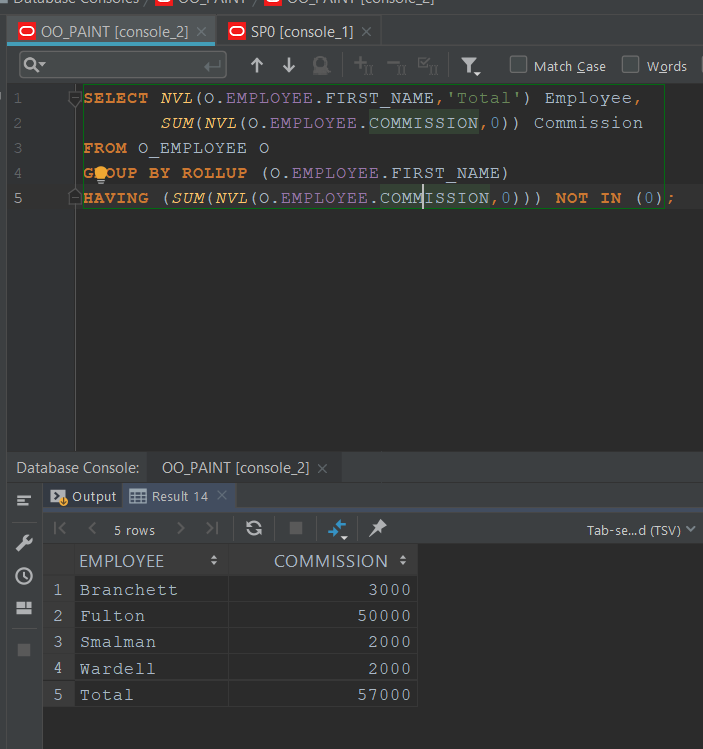
SELECT NVL(O.EMPLOYEE.FIRST\_NAME,'Total') Employee,

SUM(NVL(O.EMPLOYEE.COMMISSION,0)) Commission

FROM O\_EMPLOYEE O

GROUP BY ROLLUP (O.EMPLOYEE.FIRST\_NAME)

HAVING (SUM(NVL(O.EMPLOYEE.COMMISSION,0))) NOT IN (0);



1. Using CASE, Classify Employees by Status (use “Seasonal” and “Non-Seasonal”) and College status (“College Student” and “Non-College Student”). You should have 3 columns as a result: Firstname, Status, and College\_Status

SELECT O.EMPLOYEE.FIRST\_NAME,

(CASE WHEN O.EMPLOYEE.SEASONAL\_STATUS='T' THEN 'Seasonal'

WHEN O.EMPLOYEE.SEASONAL\_STATUS='F' THEN 'Non-Seasonal'

END) Status,

(CASE WHEN O.EMPLOYEE\_ID NOT IN (SELECT C.EMPLOYEE\_ID

FROM COLLEGE C)

THEN 'Non-College Student'

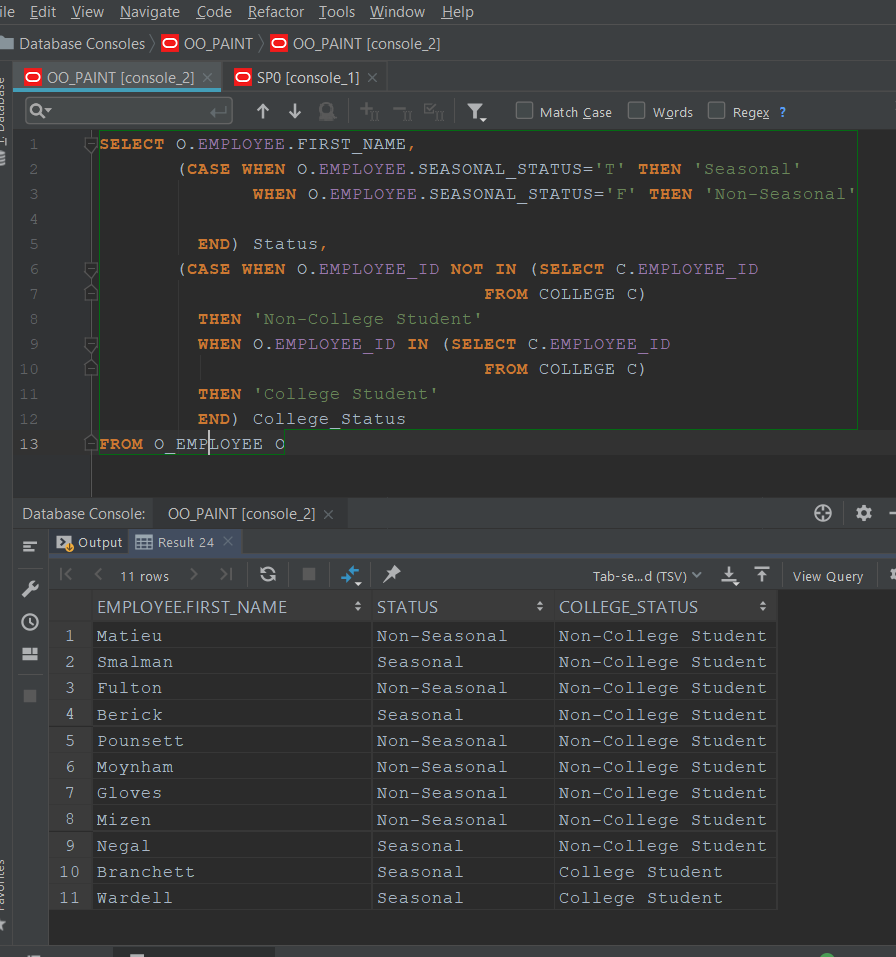
WHEN O.EMPLOYEE\_ID IN (SELECT C.EMPLOYEE\_ID

FROM COLLEGE C)

THEN 'College Student'

END) College\_Status

FROM O\_EMPLOYEE O



1. Using MONTHS\_BETWEEN and NEXT\_DAY, Select employee ID, Months an employee has been employed (if they are still employed display “Still Employed”) and the day that they started working for the company (use the next Monday after their start date as their first day on the job)

SELECT O.EMPLOYEE\_ID,(CASE WHEN NVL((MONTHS\_BETWEEN(JH.END\_DATE,JH.START\_DATE)),0)

= 0 THEN 'Still Employed'

WHEN NVL((MONTHS\_BETWEEN(JH.END\_DATE,JH.START\_DATE)),0)

NOT IN (0) THEN (SUBSTR(TO\_CHAR(NVL((MONTHS\_BETWEEN(JH.END\_DATE,JH.START\_DATE)),0)),1,4)

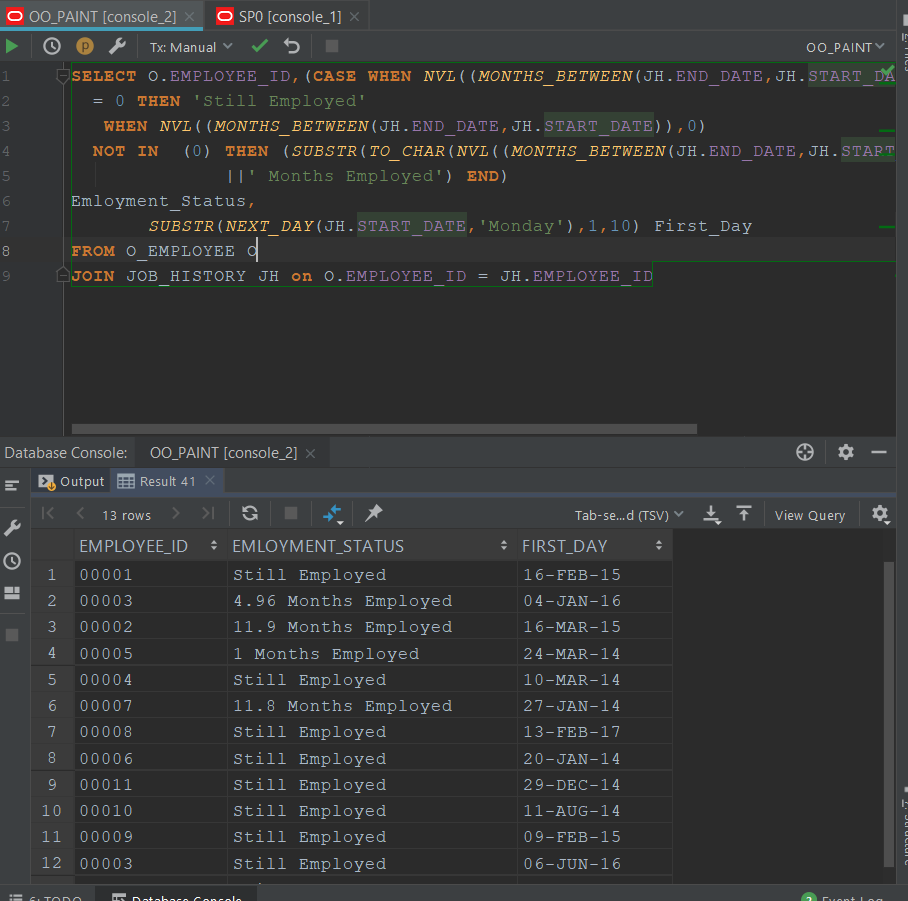
||' Months Employed') END)

Emloyment\_Status,

SUBSTR(NEXT\_DAY(JH.START\_DATE,'Monday'),1,10) First\_Day

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID



P2.2 Answer the following queries using your database (10%)

* + 1. List all product categories with their total sales and overall percent of their total sales

Product category name, total sales, percent-total

select p.painting\_id, sum(pl.PAINTING\_QTY \* p.price) total\_Prod\_Sales,

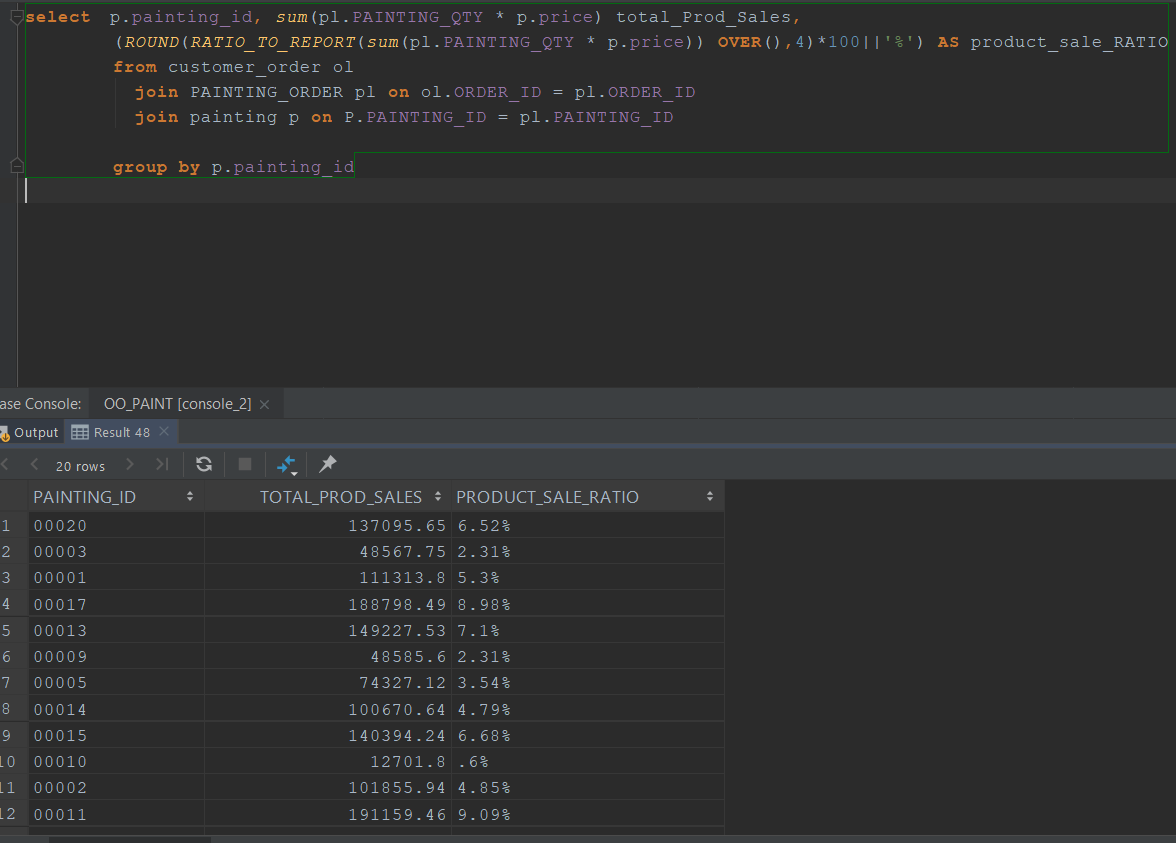
(ROUND(RATIO\_TO\_REPORT(sum(pl.PAINTING\_QTY \* p.price)) OVER(),4)\*100||'%') AS product\_sale\_RATIO

from customer\_order ol

join PAINTING\_ORDER pl on ol.ORDER\_ID = pl.ORDER\_ID

join painting p on P.PAINTING\_ID = pl.PAINTING\_ID

group by p.painting\_id



* + 1. List the top 25% percent of product in terms of total number of orders, number of orders, and number of customers visited them or used them
    2. List all products, name, category, monthly sales (for all years), number of monthly visitors (for all years)
    3. List every customer number, customer full name (last name, first name), each customer yearly purchases.

WITH t as (

select DISTINCT OC.CUSTOMER.FIRST\_NAME fname ,OC.CUSTOMER.LAST\_NAME lname ,CO.CUSTOMER\_ID id ,SUM(PO.PAINTING\_QTY\*P.PRICE) TOTAL\_SALES, SUBSTR(CO.DATE\_CREATED,8,3) YEAR

FROM CUSTOMER\_ORDER CO

JOIN O\_CUSTOMER OC on CO.CUSTOMER\_ID = OC.CUSTOMER\_ID

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

GROUP BY OC.CUSTOMER.FIRST\_NAME,OC.CUSTOMER.LAST\_NAME,CO.CUSTOMER\_ID,CO.DATE\_CREATED

ORDER BY CO.CUSTOMER\_ID)

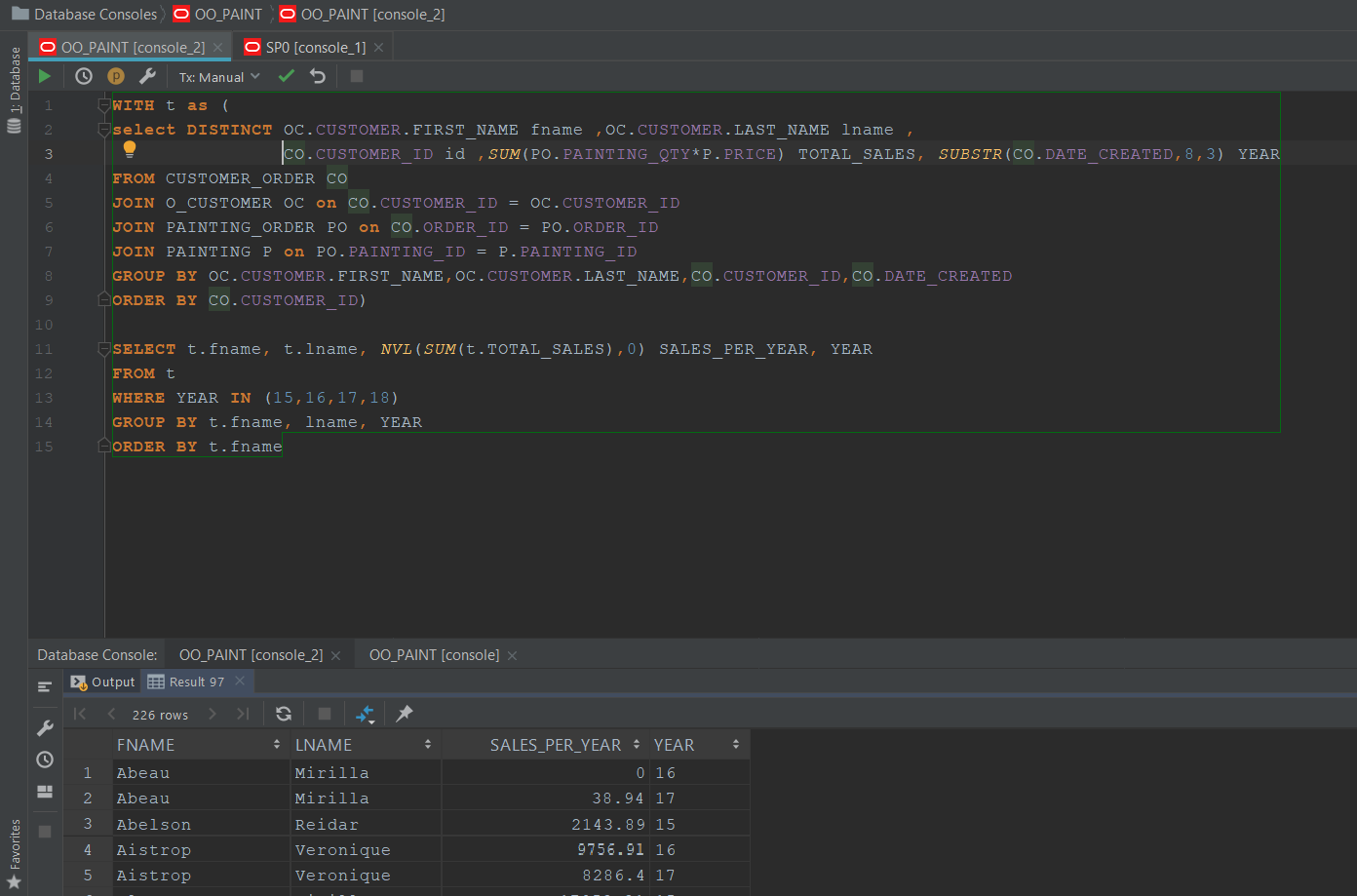
SELECT t.fname, t.lname, NVL(SUM(t.TOTAL\_SALES),0) SALES\_PER\_YEAR, YEAR

FROM t

WHERE YEAR IN (15,16,17,18)

GROUP BY t.fname, lname, YEAR

ORDER BY t.fname



* + 1. List those customers (customer number, last name, state) that their orders include more than one person.
    2. List all product categories, product name, average number of visitors per month, and average of visitors for every year

select p.painting\_id,p.TITLE, ROUND(COUNT(pl.PAINTING\_QTY)/12,2) monthly\_avg\_sold,

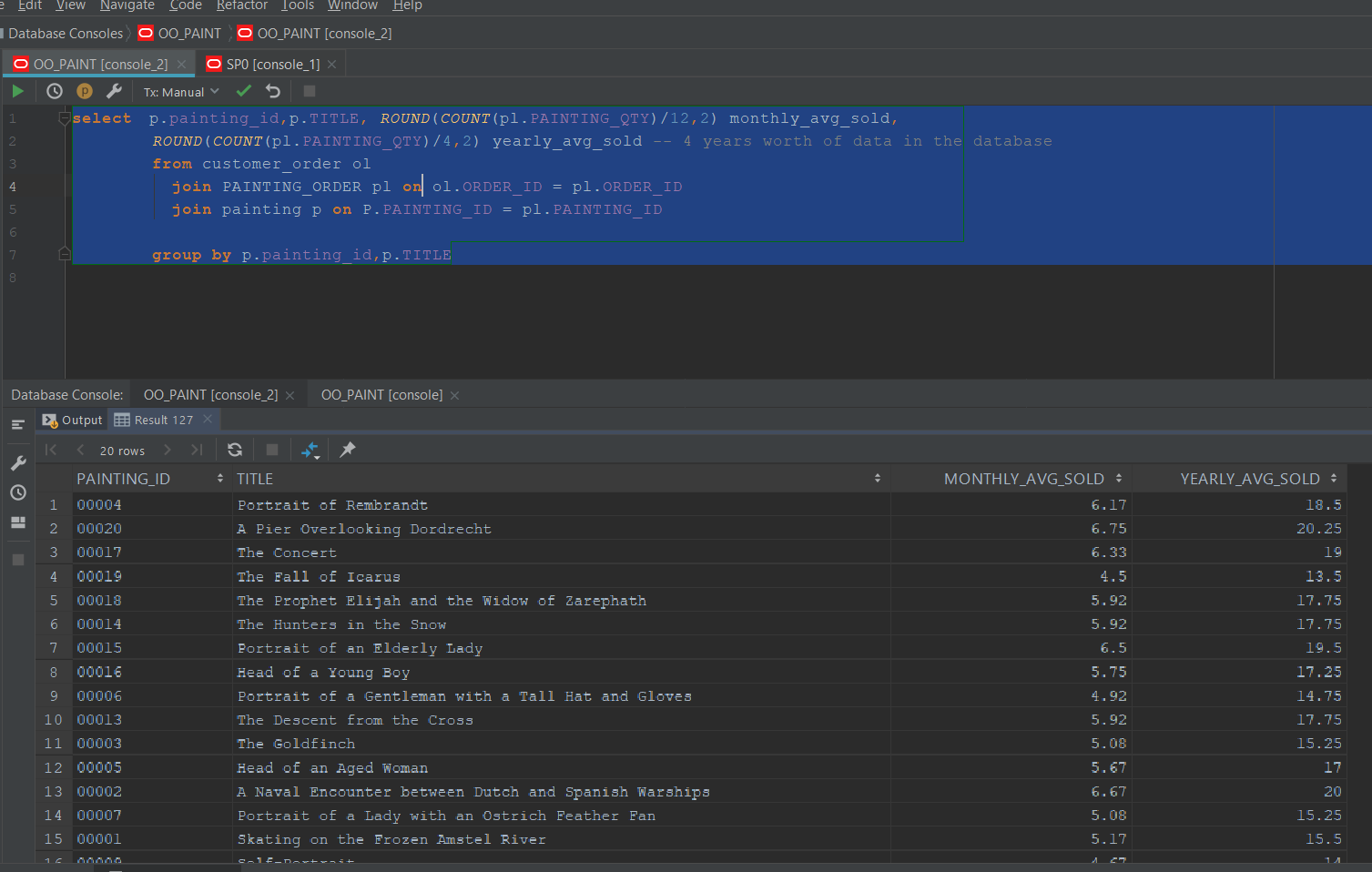
ROUND(COUNT(pl.PAINTING\_QTY)/4,2) yearly\_avg\_sold -- 4 years worth of data in the database

from customer\_order ol

join PAINTING\_ORDER pl on ol.ORDER\_ID = pl.ORDER\_ID

join painting p on P.PAINTING\_ID = pl.PAINTING\_ID

group by p.painting\_id,p.TITLE



* + 1. List all employee id, full name (last name, first name), job title, current salary, and number of years working for us.

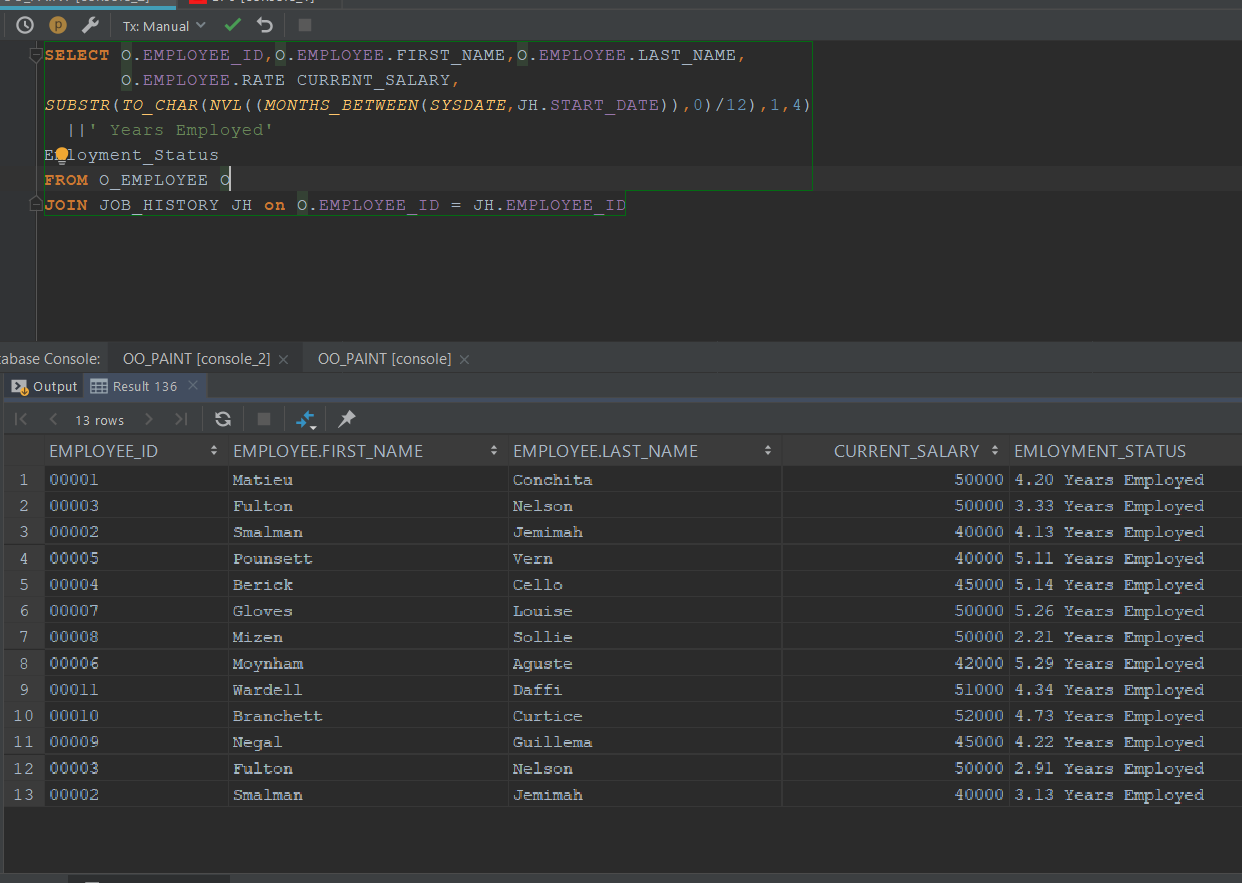
SELECT O.EMPLOYEE\_ID,O.EMPLOYEE.FIRST\_NAME,O.EMPLOYEE.LAST\_NAME,O.EMPLOYEE.RATE CURRENT\_SALARY,

SUBSTR(TO\_CHAR(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)/12),1,4)||' Years Employed'

Emloyment\_Status

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID;



* + 1. List all employee id last name, number of months working for us, worked on number of different product categories. List only those employees that worked on more than one product category.

SELECT DISTINCT O.EMPLOYEE\_ID,O.EMPLOYEE.LAST\_NAME,

O.EMPLOYEE.RATE CURRENT\_SALARY,

ROUND(SUM(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)),2)

Emloyment\_Status

FROM O\_EMPLOYEE O

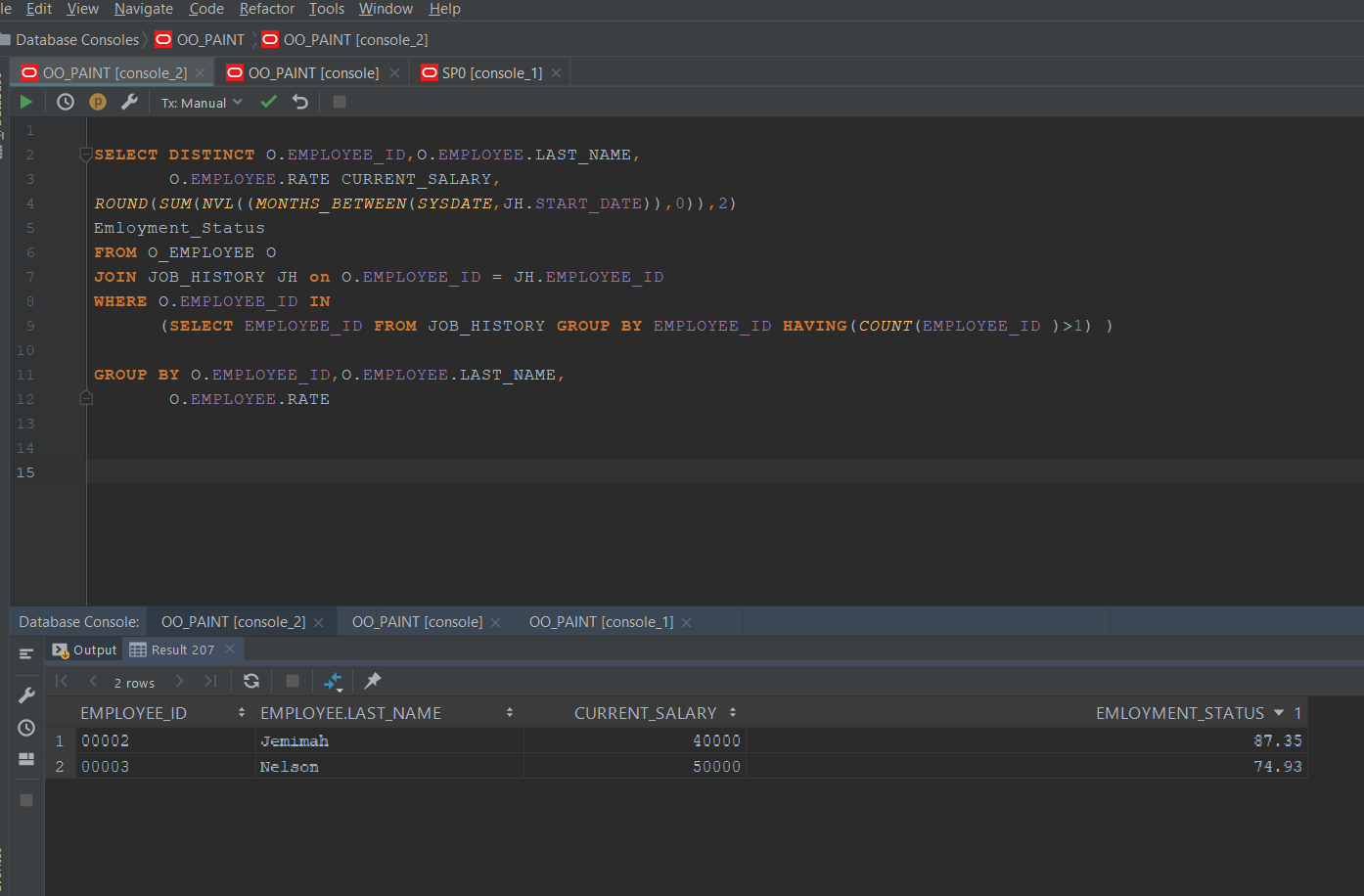
JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID

WHERE O.EMPLOYEE\_ID IN

(SELECT EMPLOYEE\_ID FROM JOB\_HISTORY GROUP BY EMPLOYEE\_ID HAVING(COUNT(EMPLOYEE\_ID )>1) )

GROUP BY O.EMPLOYEE\_ID,O.EMPLOYEE.LAST\_NAME,

O.EMPLOYEE.RATE



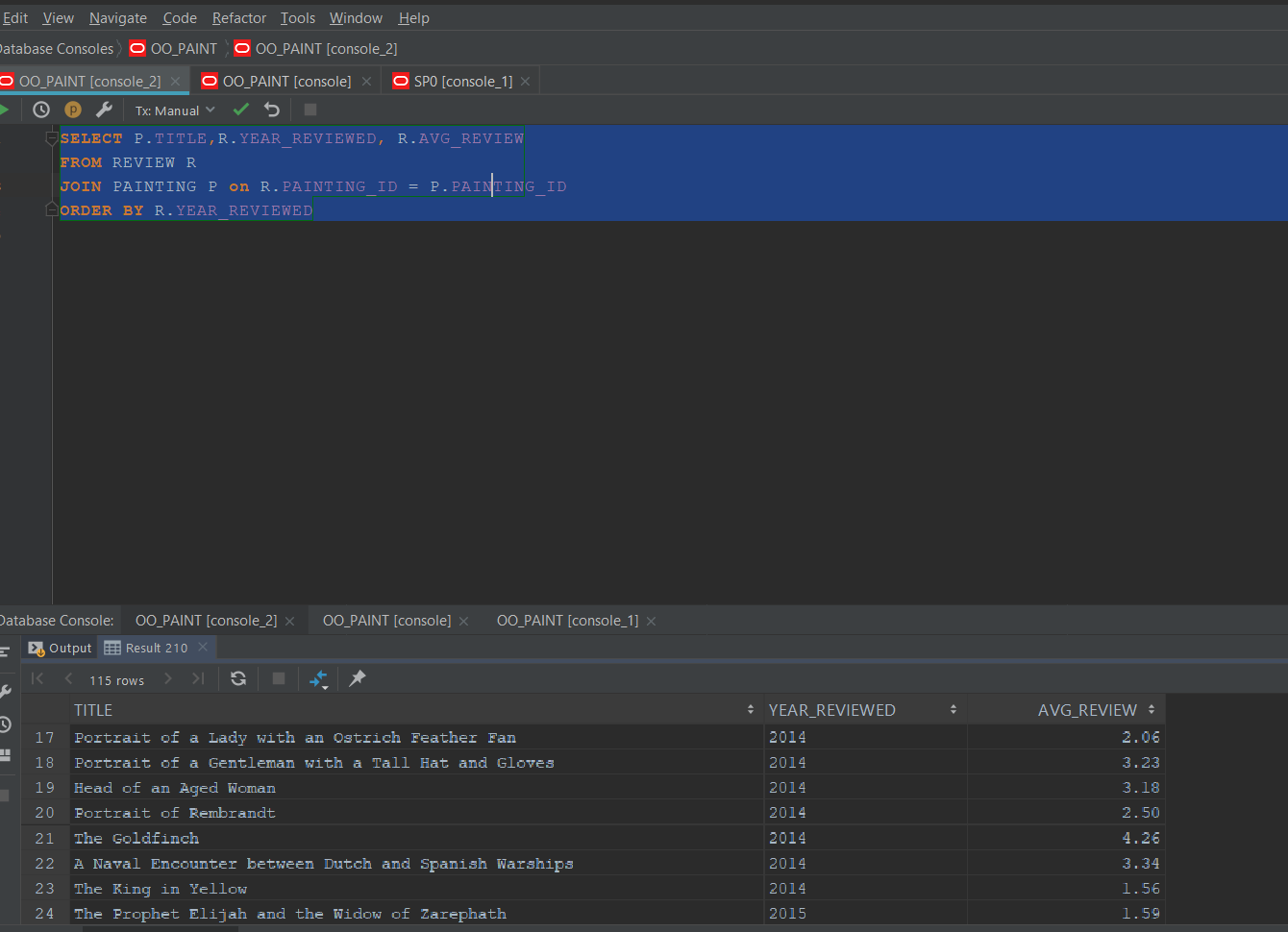
* + 1. List all vendor id, vendor name, vendor contact first name, vendor contact last name that have provided service to us more than once per year (only those years that provided more than once).
    2. For every product category, product name, list the average score (satisfaction/evaluation feedback) received from customers for every year that product is offered to our customers

SELECT P.TITLE,R.YEAR\_REVIEWED, R.AVG\_REVIEW

FROM REVIEW R

JOIN PAINTING P on R.PAINTING\_ID = P.PAINTING\_ID

ORDER BY R.YEAR\_REVIEWED



P2.3 Answer 5 of the above queries using PL/SQL. Your result should be formatted and produce the same result and output format as part 2.2 (5%)

2.2.1 (PLSQL) List all product categories with their total sales and overall percent of their total sales

Product category name, total sales, percent-total

SET serveroutput on

DECLARE

BEGIN

dbms\_output.put\_line('Painting ID'||' '||'Total Product Sales'||' '||'Product Sale Ratio');

for count\_var in (select p.painting\_id, sum(pl.PAINTING\_QTY \* p.price) total\_Prod\_Sales,

(ROUND(RATIO\_TO\_REPORT(sum(pl.PAINTING\_QTY \* p.price)) OVER(),4)\*100||'%') AS product\_sale\_RATIO

from customer\_order ol

join PAINTING\_ORDER pl on ol.ORDER\_ID = pl.ORDER\_ID

join painting p on P.PAINTING\_ID = pl.PAINTING\_ID

group by p.painting\_id)

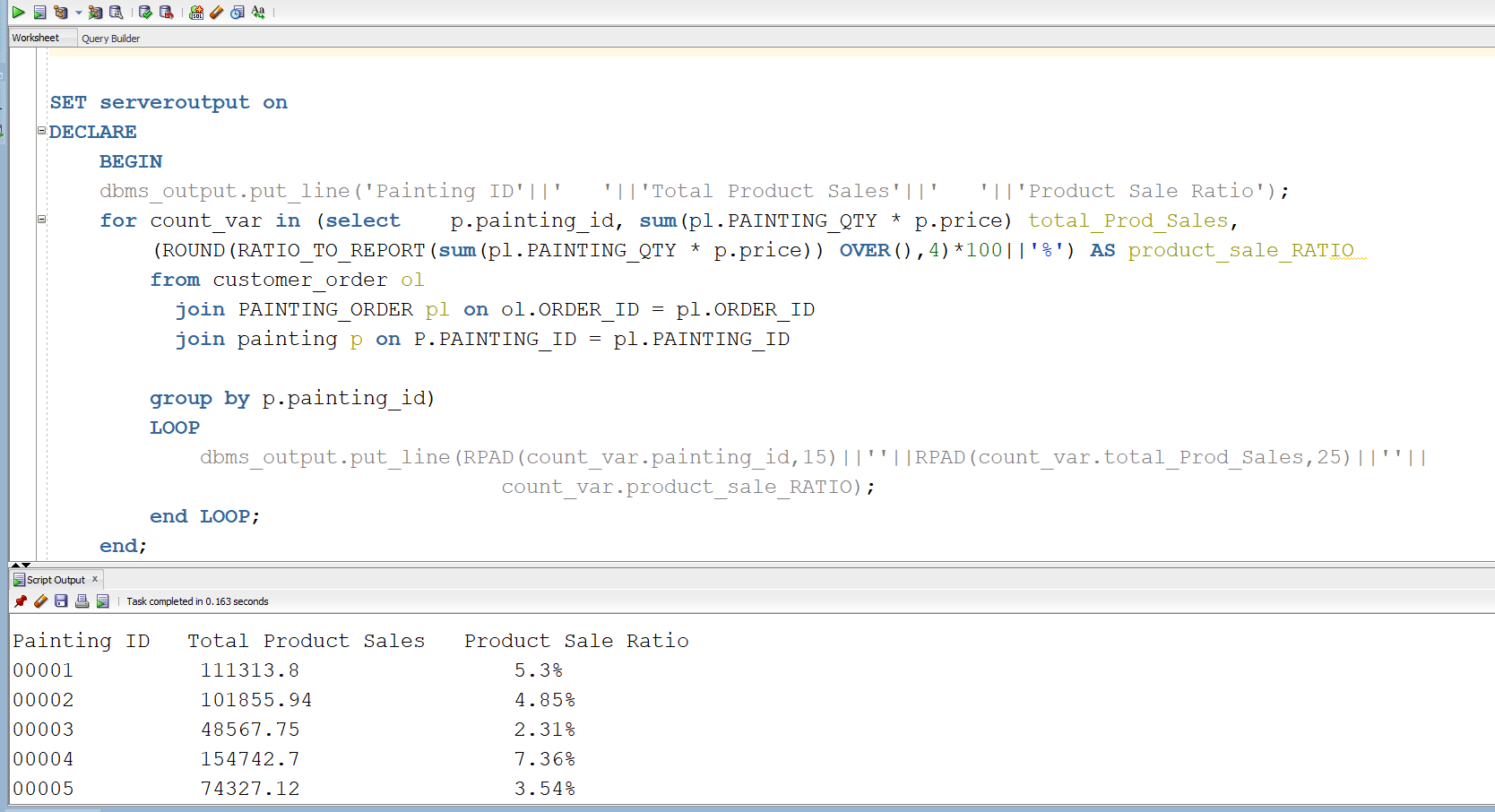
LOOP

dbms\_output.put\_line(RPAD(count\_var.painting\_id,15)||''||RPAD(count\_var.total\_Prod\_Sales,25)||''||

count\_var.product\_sale\_RATIO);

end LOOP;

end;



2.2.4 (PLSQL) List every customer number, customer full name (last name, first name), each customer yearly purchases.

set serveroutput on;

declare

BEGIN

dbms\_output.put\_line('First Name'||' '||'Last Name'||' '||'Sales Per Year'

||' '||'Year');

for cnt\_var in(

WITH t as (

select DISTINCT OC.CUSTOMER.FIRST\_NAME fname ,OC.CUSTOMER.LAST\_NAME lname ,CO.CUSTOMER\_ID id ,SUM(PO.PAINTING\_QTY\*P.PRICE) TOTAL\_SALES, SUBSTR(CO.DATE\_CREATED,8,3) YEAR

FROM CUSTOMER\_ORDER CO

JOIN O\_CUSTOMER OC on CO.CUSTOMER\_ID = OC.CUSTOMER\_ID

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

GROUP BY OC.CUSTOMER.FIRST\_NAME,OC.CUSTOMER.LAST\_NAME,CO.CUSTOMER\_ID,CO.DATE\_CREATED

ORDER BY CO.CUSTOMER\_ID)

SELECT t.fname, t.lname, NVL(SUM(t.TOTAL\_SALES),0) SALES\_PER\_YEAR, YEAR

FROM t

WHERE YEAR IN (15,16,17,18)

GROUP BY t.fname, lname, YEAR

ORDER BY t.fname)

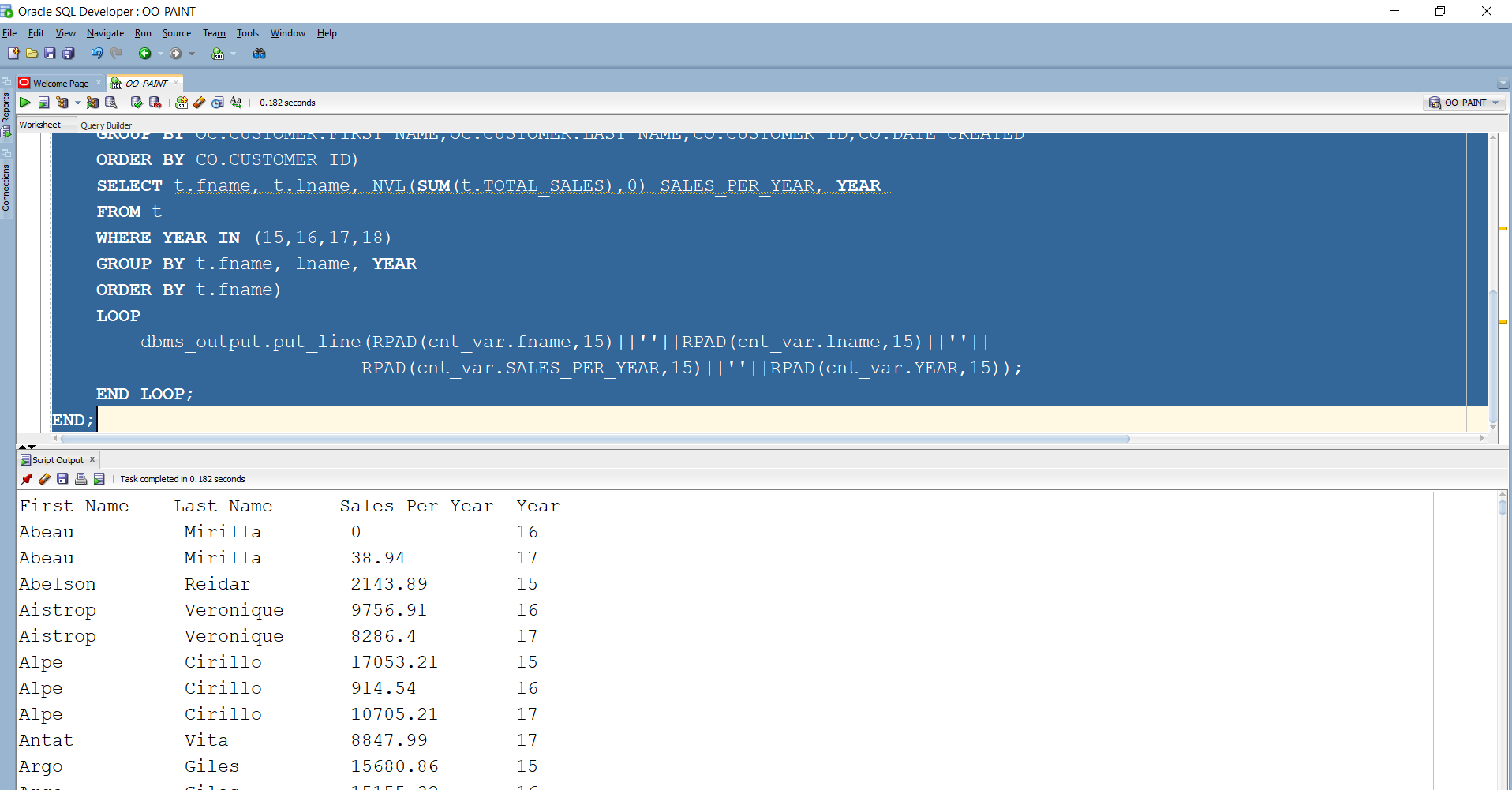
LOOP

dbms\_output.put\_line(RPAD(cnt\_var.fname,15)||''||RPAD(cnt\_var.lname,15)||''||

RPAD(cnt\_var.SALES\_PER\_YEAR,15)||''||RPAD(cnt\_var.YEAR,15));

END LOOP;

END;



2.2.6(PLSQL) List all product categories, product name, average number of visitors (Customers) per month, and average of visitors for every year

set serveroutput on;

declare

BEGIN

dbms\_output.put\_line('Painting\_ID'||' '||'Painting Title'||' '||'Monthly Avg Sold'

||' '||'Yearly Avg Sold');

for cnt\_var in (

select p.painting\_id,p.TITLE, ROUND(COUNT(pl.PAINTING\_QTY)/12,2) monthly\_avg\_sold,

ROUND(COUNT(pl.PAINTING\_QTY)/4,2) yearly\_avg\_sold -- 4 years worth of data in the database

from customer\_order ol

join PAINTING\_ORDER pl on ol.ORDER\_ID = pl.ORDER\_ID

join painting p on P.PAINTING\_ID = pl.PAINTING\_ID

group by p.painting\_id,p.TITLE

)

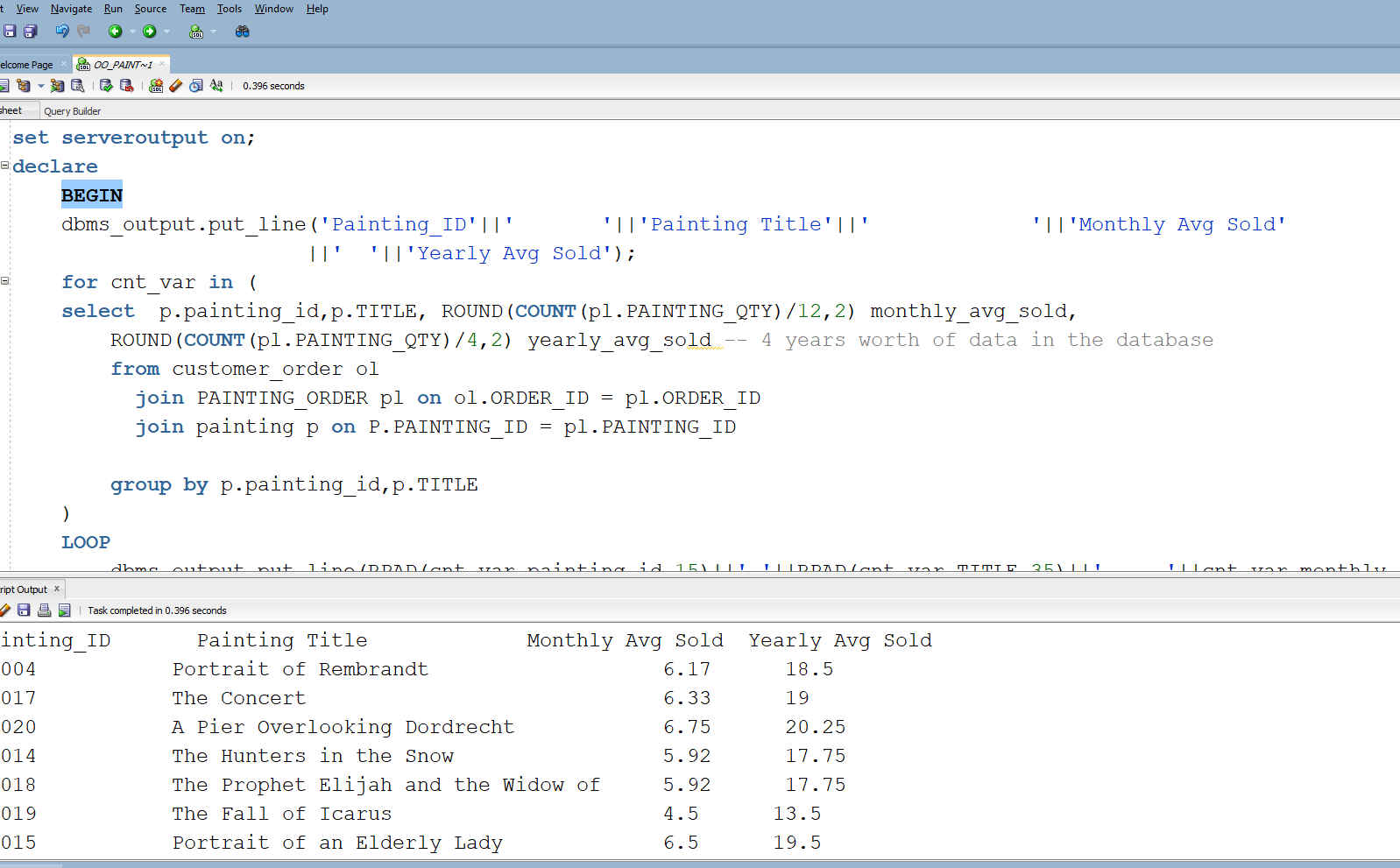
LOOP

dbms\_output.put\_line(RPAD(cnt\_var.painting\_id,15)||' '||RPAD(cnt\_var.TITLE,35)||' '||cnt\_var.monthly\_avg\_sold

||' ' ||cnt\_var.yearly\_avg\_sold);

END LOOP;

END;



* + 1. (PLSQL) List all employee id, full name (last name, first name), job title, current salary, and number of years working for us.

set serveroutput on

declare

begin

dbms\_output.put\_line('Employee ID'||' '||'First Name'||' '||'Last Name'||

' '||'Salary'||' '||'Employment Status');

for cnt\_var in ( SELECT

O.EMPLOYEE\_ID,O.EMPLOYEE.FIRST\_NAME fname,O.EMPLOYEE.LAST\_NAME lname,O.EMPLOYEE.RATE CURRENT\_SALARY,

SUBSTR(TO\_CHAR(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)/12),1,4)||' Years Employed'

Employment\_Status

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID);

LOOP

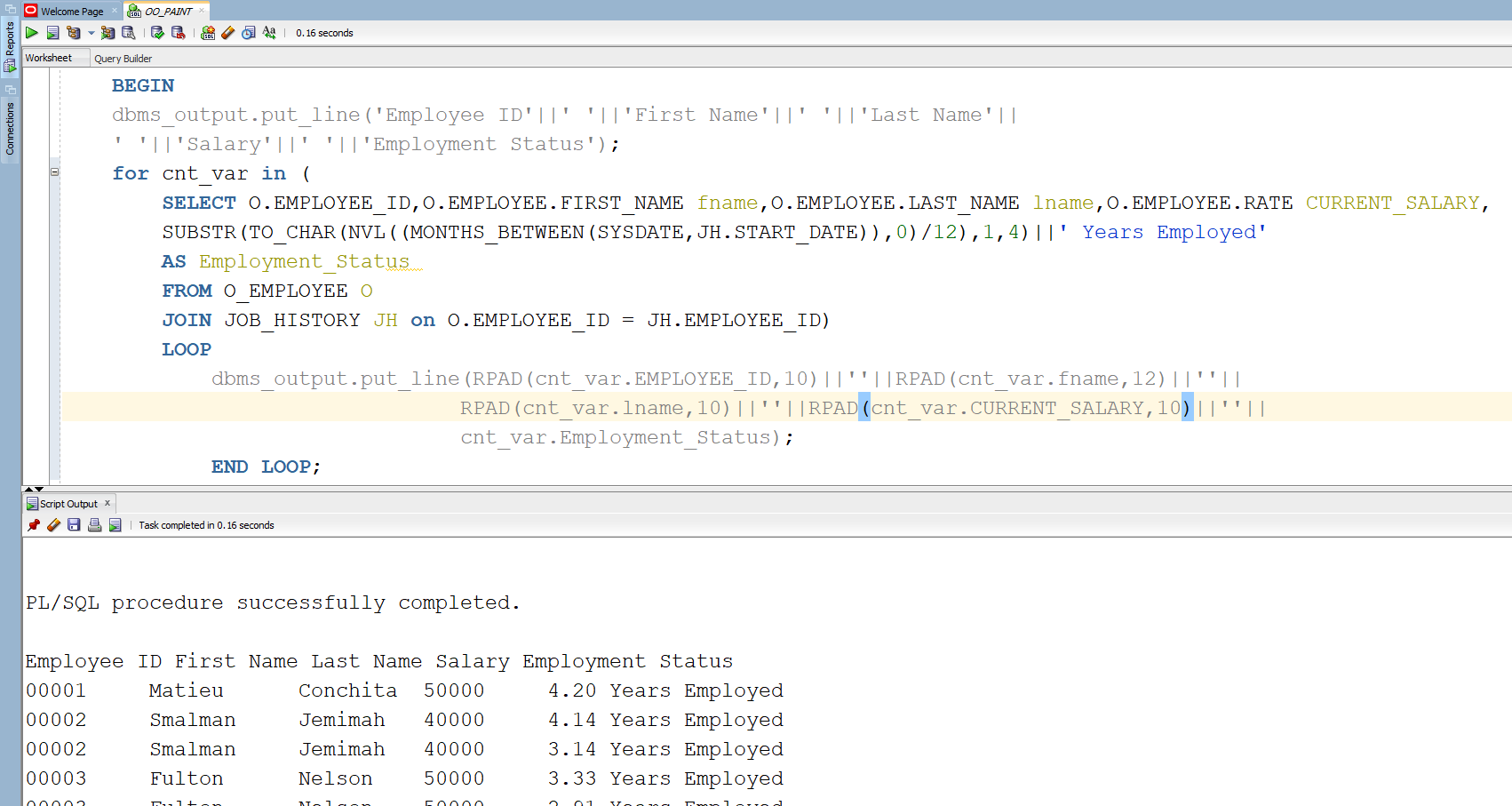
dbms\_output.put\_line(RPAD(cnt\_var.EMPLOYEE\_ID,10)||''||RPAD(cnt\_var.fname,12)||''||

RPAD(cnt\_var.lname,10)||''||RPAD(cnt\_var.CURRENT\_SALARY,10)||''||

cnt\_var.Employment\_Status);

END LOOP;

END;



2.2.8 (PLSQL) List all employee id last name, number of months working for us, worked on number of different product categories. List only those employees that worked on more than one product category

set serveroutput on

declare

begin

dbms\_output.put\_line('Employee ID'||' '||'First Name'||' '||'Salary'||' '||'Employment Status');

for cnt\_var in (

SELECT DISTINCT O.EMPLOYEE\_ID,O.EMPLOYEE.LAST\_NAME fname,O.EMPLOYEE.RATE CURRENT\_SALARY,

ROUND(SUM(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)),2)||' Months Employed'

Employment\_Status

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID

WHERE O.EMPLOYEE\_ID IN

(SELECT EMPLOYEE\_ID FROM JOB\_HISTORY GROUP BY EMPLOYEE\_ID HAVING(COUNT(EMPLOYEE\_ID )>1) )

GROUP BY O.EMPLOYEE\_ID,O.EMPLOYEE.LAST\_NAME,O.EMPLOYEE.RATE)

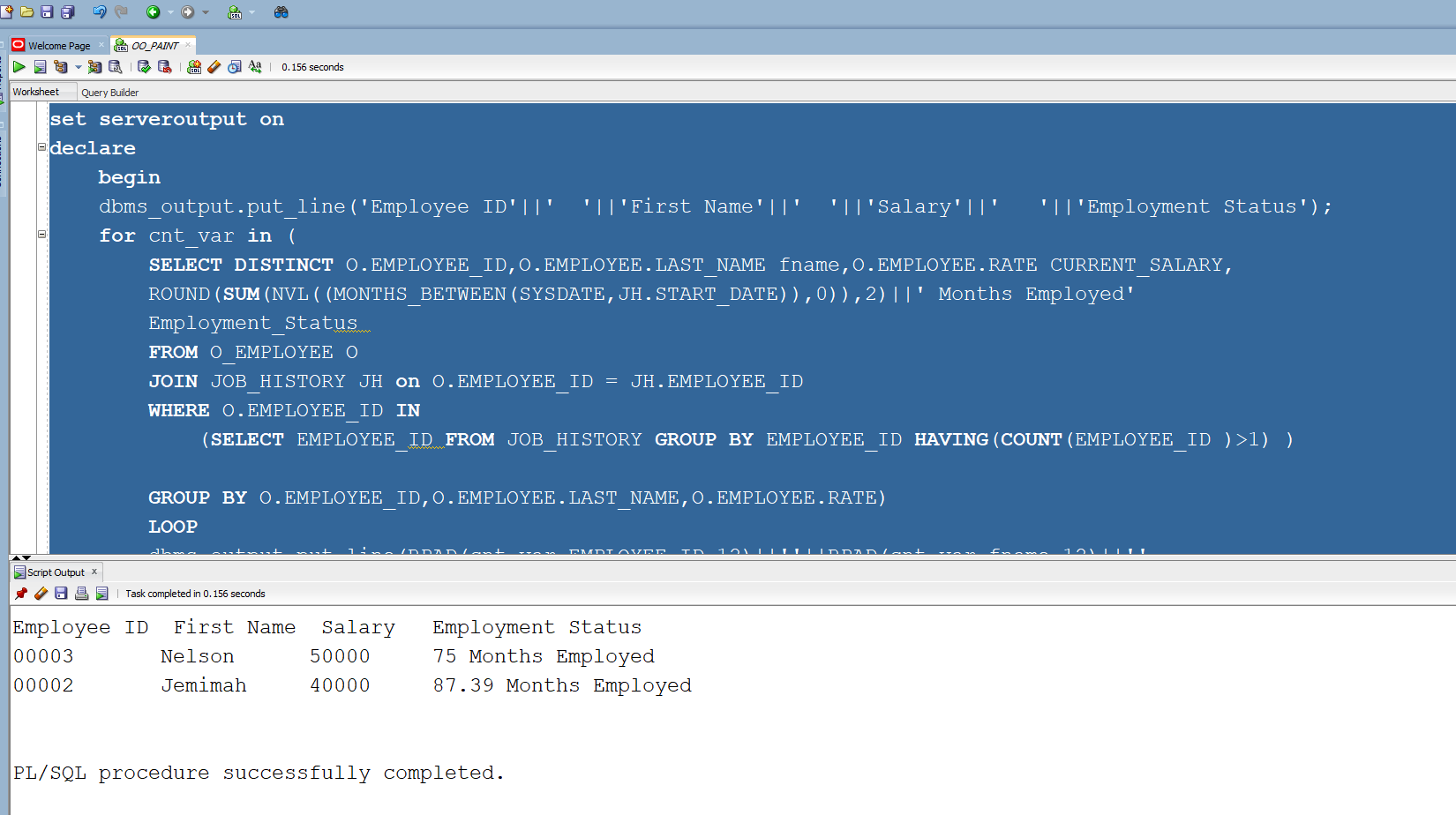
LOOP

dbms\_output.put\_line(RPAD(cnt\_var.EMPLOYEE\_ID,12)||''||RPAD(cnt\_var.fname,12)||''

||RPAD(cnt\_var.CURRENT\_SALARY,10)||''||cnt\_var.Employment\_Status);

END LOOP;

END;



P2.4: Using your current database, develop a user defined function for the following logics. (10%) (Reformat with ROWNUM showing)

* + 1. Develop a user defined functions that will accept product id, year and will return yearly sales for that product.

set serveroutput on;

CREATE OR REPLACE FUNCTION PROD\_YEARLY\_SALES(product\_id in varchar,

year\_sale in number)

return number is

cursor a is (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

WHERE P.PAINTING\_ID = product\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale); --input 2

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

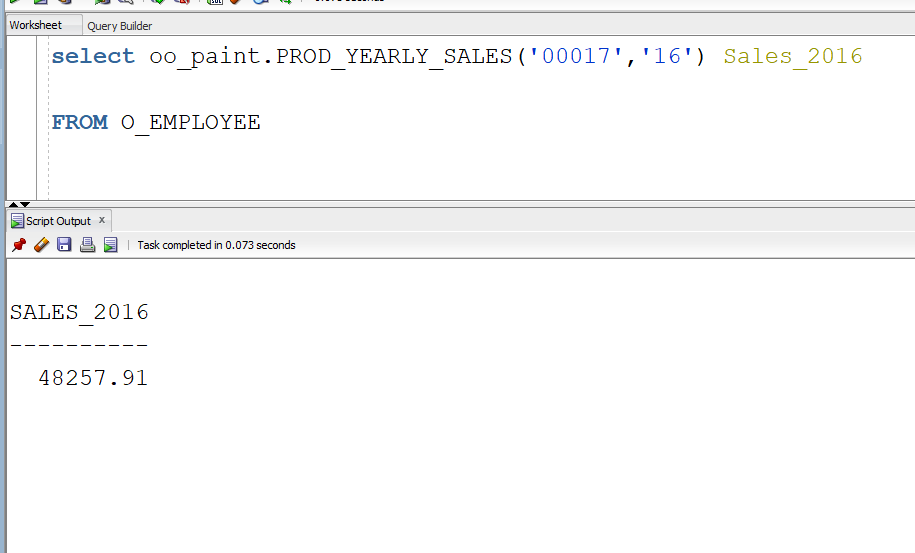
return 'No Such year or Product ID';

else

return a\_var.Prod\_Sales;

end if;

end;



* + 1. Develop a user defined functions that will accept product id, month, and year. This function then will return total sales for that product for that month of that year.

create or replace FUNCTION PROD\_MONTHLY\_SALES(product\_id in varchar,

month\_sale varchar)

return number is

cursor a is (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

WHERE P.PAINTING\_ID = product\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

HAVING SUBSTR(DATE\_CREATED,4,3)=month\_sale

); --input 2

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

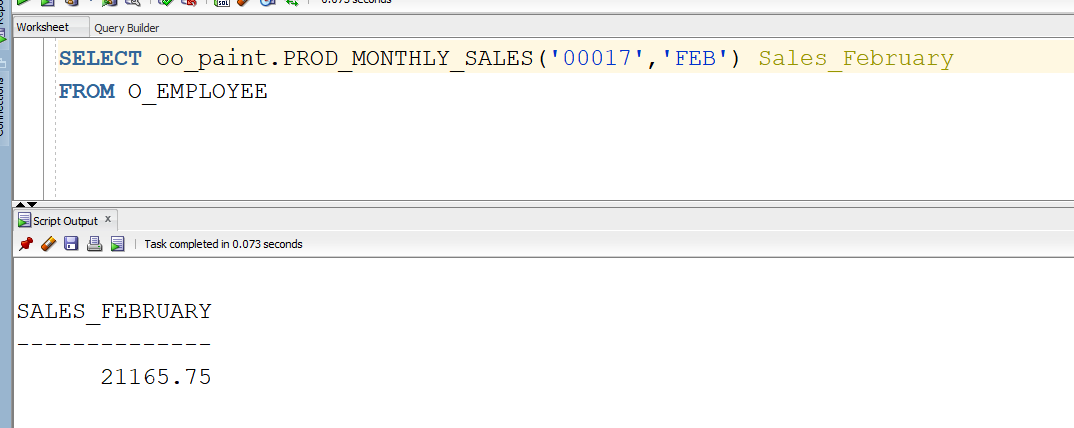
return 'No Such month or Product ID';

else

return a\_var.Prod\_Sales;

end if;

end;



* + 1. Develop a user defined functions that will accept product category id and will return yearly sales for that product category. (Since all products are paintings we will be treating category as “style id” instead, which returns the ID for a specific style of painting ie impressionism)

create or replace FUNCTION STYLE\_YEARLY\_SALES(style\_id in varchar,

year\_sale in number)

return number is

cursor a is (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

JOIN PAINTING\_STYLE PS on P.PAINTING\_ID = PS.PAINTING\_ID

WHERE PS.STYLE\_ID = style\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale

); --input 2

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

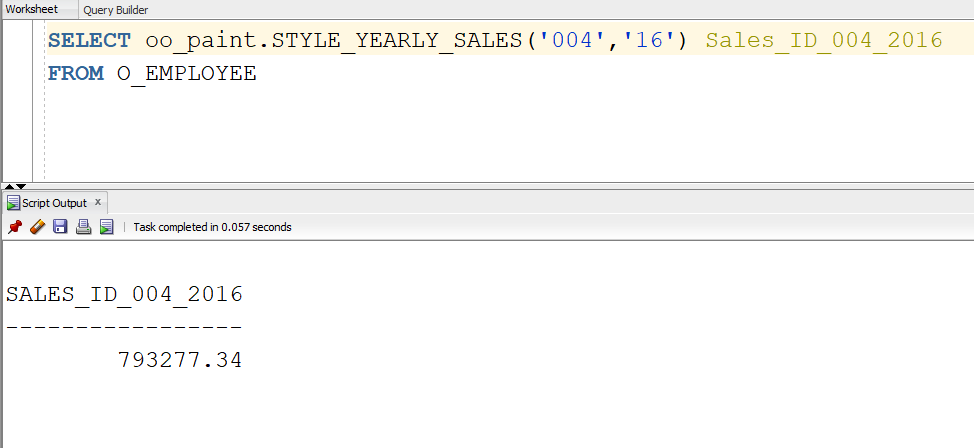
return 'No Such year or Product ID';

else

return a\_var.Prod\_Sales;

end if;

end;



* + 1. Develop a user defined functions that will accept product category id, month, and year. This function then will return total sales for that product. (Since all products are paintings we will be treating category as “style id” instead, which returns the ID for a specific style of painting ie impressionism)

create or replace FUNCTION STYLE\_YEARLY\_MONTHLY\_SALES(style\_id in varchar,

year\_sale in number,

month\_sale in varchar)

return number is

cursor a is (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

JOIN PAINTING\_STYLE PS on P.PAINTING\_ID = PS.PAINTING\_ID

WHERE PS.STYLE\_ID = style\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale

AND SUBSTR(CO.DATE\_CREATED,4,3) = month\_sale

); --input 2 &3

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

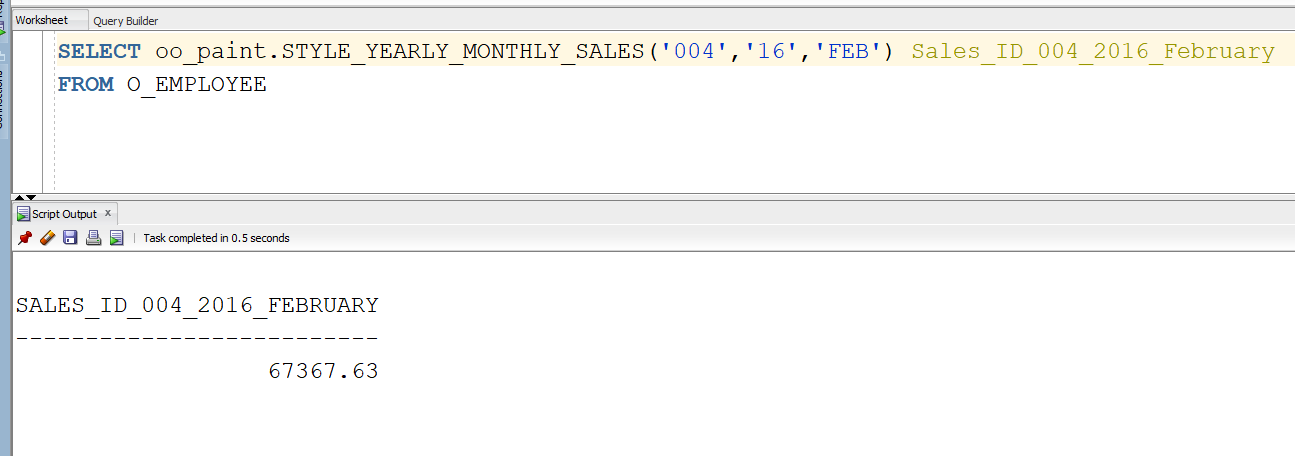
return 'No Such year or Product ID';

else

return a\_var.Prod\_Sales;

end if;

end;



* + 1. Develop a user defined function that will accept a vendor id and year and it will return number if services they have provided to us. (To fit my database, this function accepts employee\_id and year, and returns how many orders they have worked on that year)

create or replace FUNCTION YEARLY\_EMP\_ORDERS(employee\_identification in varchar,

year\_sale in number

)

return number is

cursor a is (

SELECT OE.EMPLOYEE\_ID, COUNT(CO.EMPLOYEE\_ID) Num\_Orders

FROM CUSTOMER\_ORDER CO

JOIN O\_EMPLOYEE OE on CO.EMPLOYEE\_ID = OE.EMPLOYEE\_ID

WHERE (SUBSTR(CO.DATE\_CREATED,8,2))=year\_sale -- input 1

AND OE.EMPLOYEE\_ID = employee\_identification --input 2

GROUP BY OE.EMPLOYEE\_ID

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

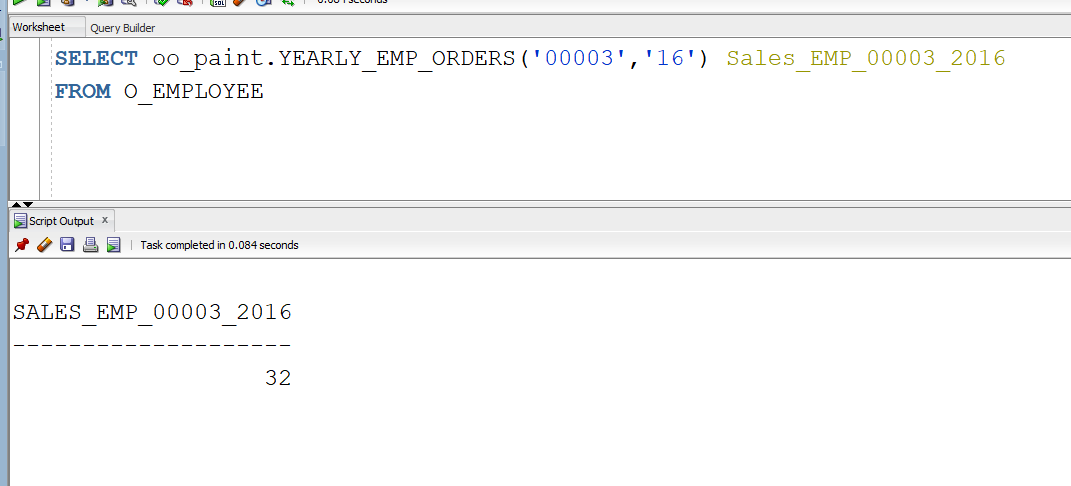
return 'No Such year or Product ID';

else

return a\_var.Num\_Orders;

end if;

end;



* + 1. Develop a user defined function that will accept an order number and will return number of people in that order. (My database has only 1 person per order so this function will accept an order number and return number of products in that order)

create or replace FUNCTION NUM\_PRODUCTS(prod\_id in varchar

)

return number is

cursor a is (

SELECT COUNT(PO.PAINTING\_ID) Paintings

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

-- input 1

WHERE PO.ORDER\_ID = prod\_id --input 2

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

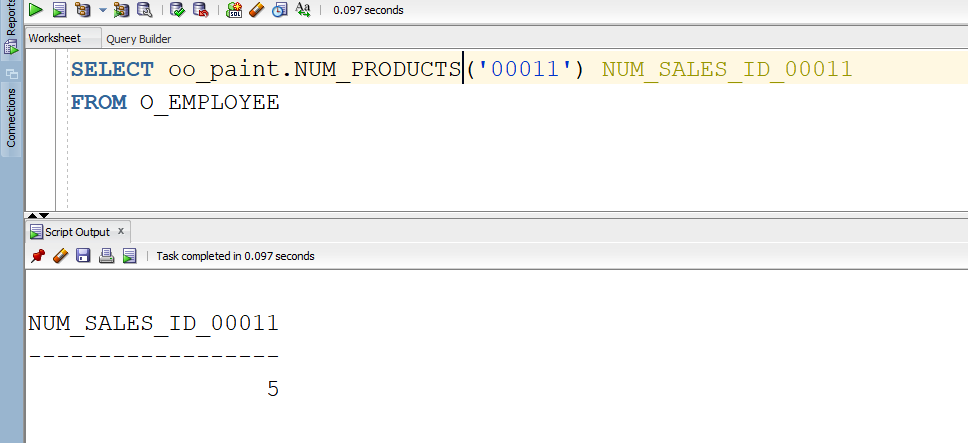
return 'No Such year or Product ID';

else

return a\_var.Paintings;

end if;

end;



* + 1. Develop a user defined function that will accept employee number and will return the current salary.

create or replace FUNCTION EMP\_SALARY(emp\_id in varchar

)

return number is

cursor a is (

SELECT e.EMPLOYEE.RATE SALARY

FROM O\_EMPLOYEE e

WHERE e.EMPLOYEE\_ID = emp\_id --input 2

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

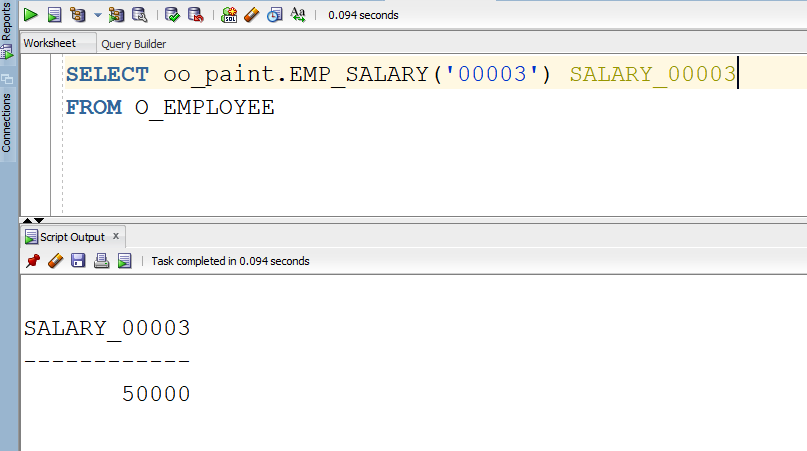
return 'No Such year or Product ID';

else

return a\_var.SALARY;

end if;

end;



* + 1. Develop a user defined function that will accept employee number and will return the current job title.

create or replace FUNCTION EMP\_POSITION(emp\_id in varchar

)

return varchar is

cursor a is (

SELECT e.EMPLOYEE.POSITION POSITION

FROM O\_EMPLOYEE e

WHERE e.EMPLOYEE\_ID = emp\_id --input 2

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

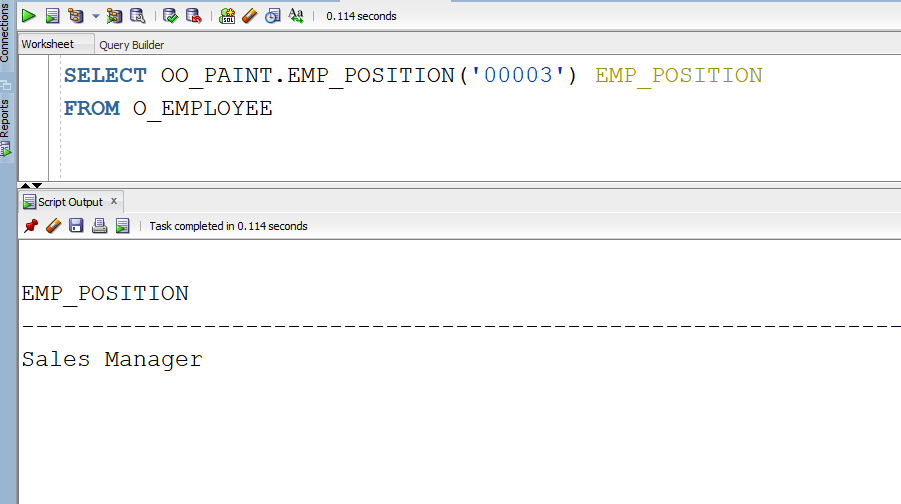
return 'No Emp ID';

else

return a\_var.POSITION;

end if;

end;



* + 1. Develop a user defined function that will accept product number and will return the average customer satisfaction/or evaluation rating for that product.

create or replace FUNCTION PAINTING\_RATING(paint\_id in varchar )

return number is

cursor a is (

SELECT AVG(R.AVG\_REVIEW) AVG\_Review

FROM REVIEW R

WHERE R.PAINTING\_ID = paint\_id

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

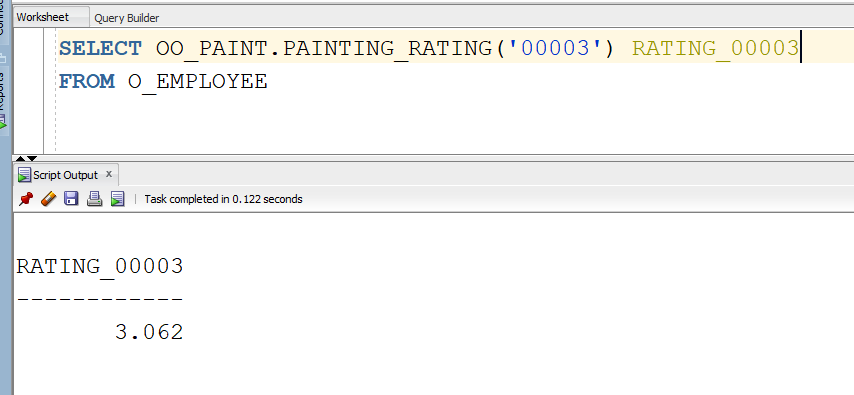
return 'No Such year or Product ID';

else

return a\_var.AVG\_Review;

end if;

end;



* + 1. Develop a user defined function that will accept employee id last name and will return number of months that employee is working on number of different product categories.

create or replace FUNCTION EMP\_MONTHS(emp\_id in varchar

)

return varchar is

cursor a is (

SELECT O.EMPLOYEE\_ID,O.EMPLOYEE.FIRST\_NAME,O.EMPLOYEE.LAST\_NAME,O.EMPLOYEE.RATE CURRENT\_SALARY,

SUBSTR(TO\_CHAR(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)),1,4)||' Months Employed'

Emloyment\_Status

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID

WHERE O.EMPLOYEE\_ID = emp\_id

);

a\_var a%rowtype;

begin

open a;

fetch a into a\_var;

if a%notfound then

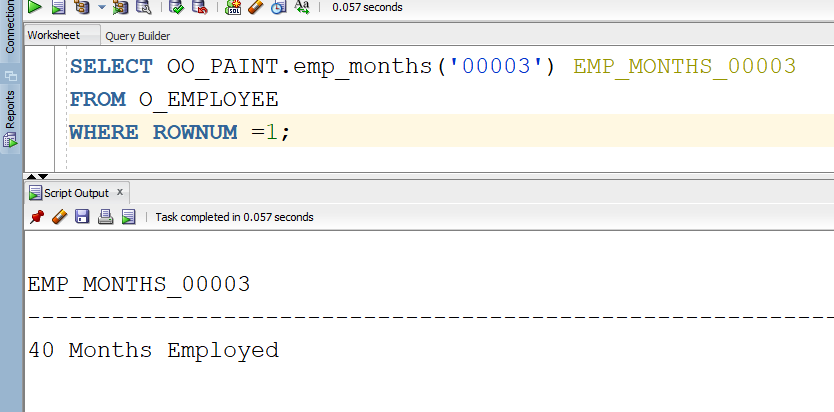
return 'No Such year or Product ID';

else

return a\_var.Emloyment\_Status;

end if;

end;



P2.5: Redo the P2.4 question using Stored Procedures. You need a driver program to make the call to stored procedures. All the result should be displayed by the driver program. The output format should be exactly the same as result displayed in P2.5 (10%) (Fix output “Prod months”)

P2.5.1 Develop a stored procedure that will accept product id, year and will return yearly sales for that product.

set serveroutput on;

CREATE OR REPLACE PROCEDURE YEARLY\_SALES\_PROD (product\_id in varchar,

year\_sale in number)

IS

BEGIN

dbms\_output.put\_line('Year sales of product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

WHERE P.PAINTING\_ID = product\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

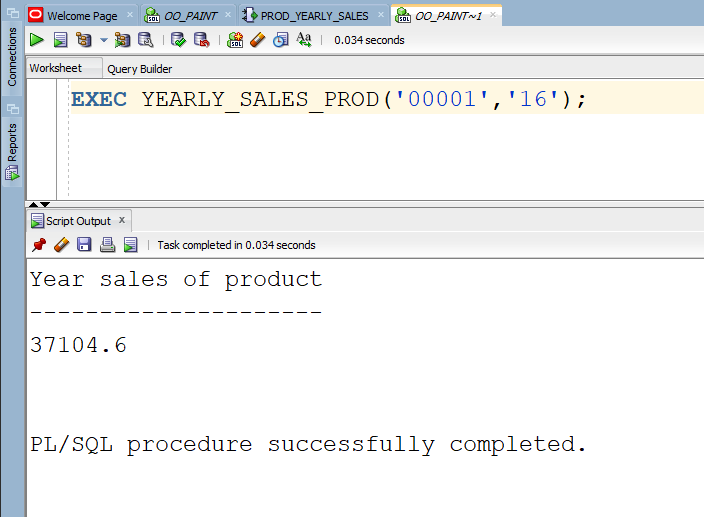
HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale)

LOOP

dbms\_output.put\_line(a.Prod\_Sales);

END LOOP;

END;



P2.5.2 Develop a stored procedure that will accept product id, month, and year. This function then will return total sales for that product for that month of that year.

set serveroutput on;

CREATE OR REPLACE PROCEDURE MONTHLY\_SALES\_PROD (product\_id in varchar,

month\_sale in varchar)

IS

BEGIN

dbms\_output.put\_line('Month sales of product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

WHERE P.PAINTING\_ID = product\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

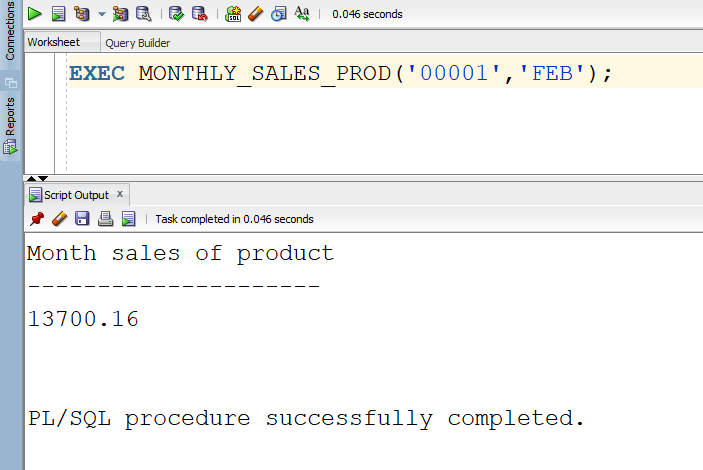
HAVING SUBSTR(DATE\_CREATED,4,3)=month\_sale)

LOOP

dbms\_output.put\_line(a.Prod\_Sales);

END LOOP;

END;



2.5.3 Develop a stored procedure that will accept product category id and will return yearly sales for that product category. (Since all products are paintings we will be treating category as “style id” instead, which returns the ID for a specific style of painting ie impressionism)

set serveroutput on;

CREATE OR REPLACE PROCEDURE P\_STYLE\_YEARLY\_SALES(style\_id in varchar,

year\_sale in number)

IS

BEGIN

dbms\_output.put\_line('Month sales of product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

JOIN PAINTING\_STYLE PS on P.PAINTING\_ID = PS.PAINTING\_ID

WHERE PS.STYLE\_ID = style\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

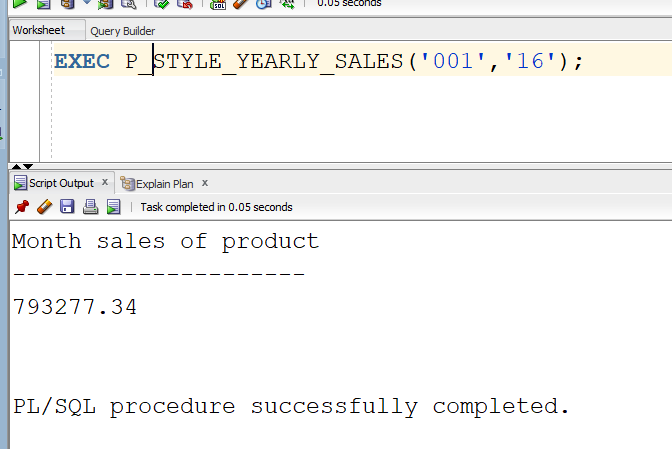
HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale)

LOOP

dbms\_output.put\_line(a.Prod\_Sales);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept product category id, month, and year. This function then will return total sales for that product. (Since all products are paintings we will be treating category as “style id” instead, which returns the ID for a specific style of painting ie impressionism)

set serveroutput on;

CREATE OR REPLACE PROCEDURE P\_STYLE\_YEARLY\_MONTHLY\_SALES(style\_id in varchar,

year\_sale in number,

month\_sale in varchar)

IS

BEGIN

dbms\_output.put\_line('Month sales of product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT DISTINCT SUM(SUM(PO.PAINTING\_QTY\*P.PRICE)) Prod\_Sales

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

JOIN PAINTING\_STYLE PS on P.PAINTING\_ID = PS.PAINTING\_ID

WHERE PS.STYLE\_ID = style\_id -- input 1

GROUP BY CO.DATE\_CREATED, P.PAINTING\_ID

HAVING SUBSTR(Co.DATE\_CREATED,8,2)=year\_sale

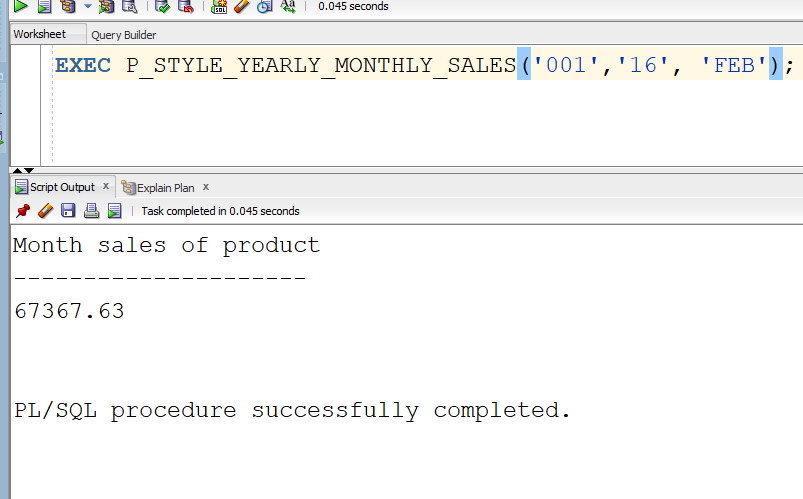
AND SUBSTR(CO.DATE\_CREATED,4,3) = month\_sale)

LOOP

dbms\_output.put\_line(a.Prod\_Sales);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept a vendor id and year and it will return number if services they have provided to us. (To fit my database, this function accepts employee\_id and year, and returns how many orders they have worked on that year)

set serveroutput on;

CREATE OR REPLACE PROCEDURE P\_YEARLY\_EMP\_ORDERS(employee\_identification in varchar,

year\_sale in number

)

IS

BEGIN

dbms\_output.put\_line('Month sales of product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT OE.EMPLOYEE\_ID, COUNT(CO.EMPLOYEE\_ID) Num\_Orders

FROM CUSTOMER\_ORDER CO

JOIN O\_EMPLOYEE OE on CO.EMPLOYEE\_ID = OE.EMPLOYEE\_ID

WHERE (SUBSTR(CO.DATE\_CREATED,8,2))=year\_sale -- input 1

AND OE.EMPLOYEE\_ID = employee\_identification --input 2

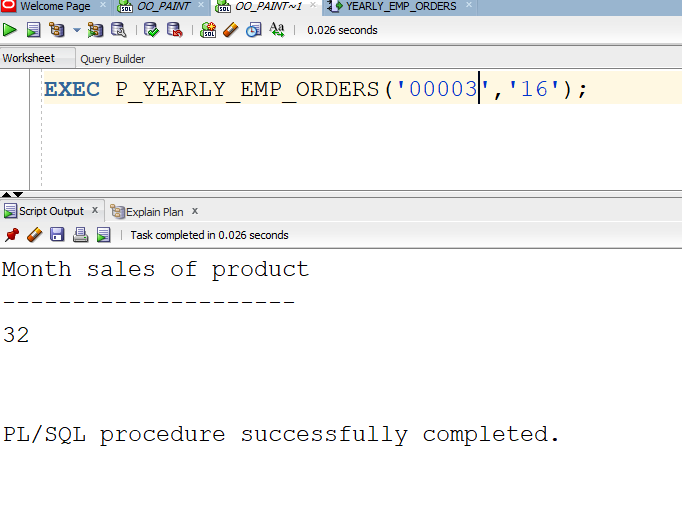
GROUP BY OE.EMPLOYEE\_ID)

LOOP

dbms\_output.put\_line(a.Num\_Orders);

END LOOP;

END;



* + 1. Develop a procedure that will accept an order number and will return number of people in that order. (My database has only 1 person per order so this function will accept an order number and return number of products in that order)

set serveroutput on;

create or replace procedure P\_NUM\_PRODUCTS(prod\_id in varchar

)

IS

BEGIN

dbms\_output.put\_line(‘NUM\_Product');

dbms\_output.put\_line('---------------------');

for a in (

SELECT COUNT(PO.PAINTING\_ID) Paintings

FROM CUSTOMER\_ORDER CO

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

-- input 1

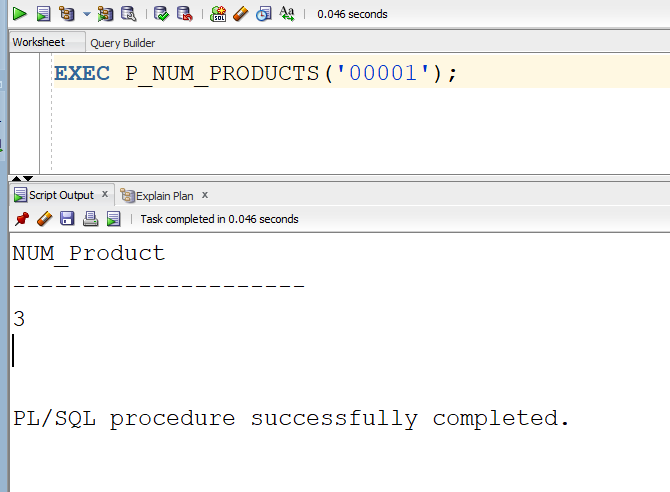
WHERE PO.ORDER\_ID = prod\_id)

LOOP

dbms\_output.put\_line(a.Paintings);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept employee number and will return the current salary.

set serveroutput on;

create or replace procedure P\_EMP\_SALARY(emp\_id in varchar

)

IS

BEGIN

dbms\_output.put\_line('Salary');

dbms\_output.put\_line('---------------------');

for a in (

SELECT e.EMPLOYEE.RATE SALARY

FROM O\_EMPLOYEE e

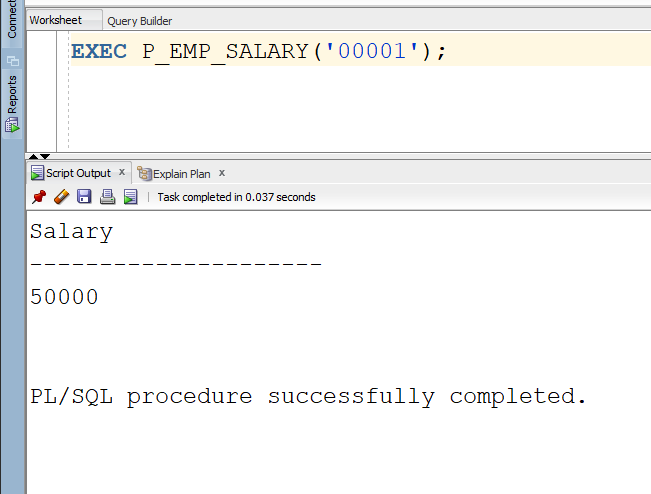
WHERE e.EMPLOYEE\_ID = emp\_id) --input 2)

LOOP

dbms\_output.put\_line(a.Salary);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept employee number and will return the current job title.

set serveroutput on;

create or replace procedure P\_EMP\_POSITION(emp\_id in varchar

)

IS

BEGIN

dbms\_output.put\_line('POSITION');

dbms\_output.put\_line('---------------------');

for a in (

SELECT e.EMPLOYEE.POSITION POSITION

FROM O\_EMPLOYEE e

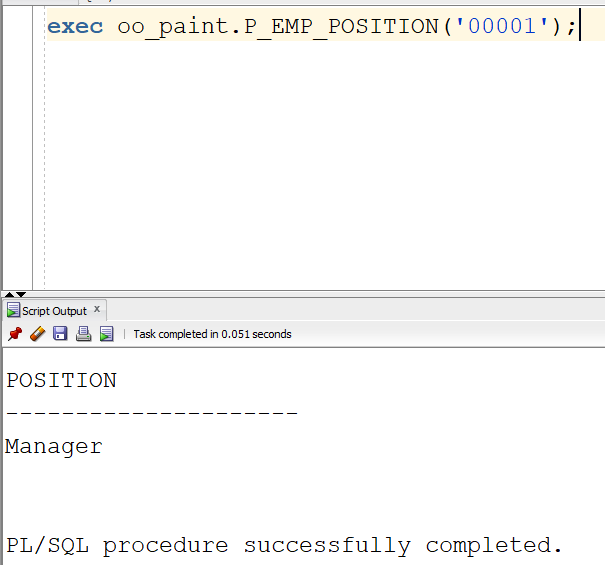
WHERE e.EMPLOYEE\_ID = emp\_id) --input 2)

LOOP

dbms\_output.put\_line(a.POSITION);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept product number and will return the average customer satisfaction/or evaluation rating for that product.

set serveroutput on;

create or replace procedure P\_EMP\_POSITION(emp\_id in varchar

)

IS

BEGIN

dbms\_output.put\_line('POSITION');

dbms\_output.put\_line('---------------------');

for a in (

SELECT e.EMPLOYEE.POSITION POSITION

FROM O\_EMPLOYEE e

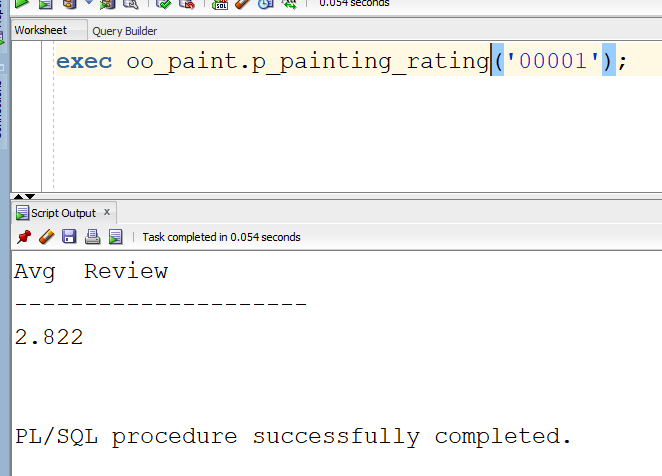
WHERE e.EMPLOYEE\_ID = emp\_id) --input 2)

LOOP

dbms\_output.put\_line(a.POSITION);

END LOOP;

END;



* + 1. Develop a stored procedure that will accept employee id last name and will return number of months that employee is working on number of different product categories

set serveroutput on;

create or replace procedure P\_EMP\_MONTHS(emp\_id in varchar

)

IS

BEGIN

dbms\_output.put\_line('Status');

dbms\_output.put\_line('---------------------');

for a in (

SELECT O.EMPLOYEE\_ID,O.EMPLOYEE.FIRST\_NAME,O.EMPLOYEE.LAST\_NAME,O.EMPLOYEE.RATE CURRENT\_SALARY,

SUBSTR(TO\_CHAR(NVL((MONTHS\_BETWEEN(SYSDATE,JH.START\_DATE)),0)),1,4)||' Months Employed'

Employment\_Status

FROM O\_EMPLOYEE O

JOIN JOB\_HISTORY JH on O.EMPLOYEE\_ID = JH.EMPLOYEE\_ID

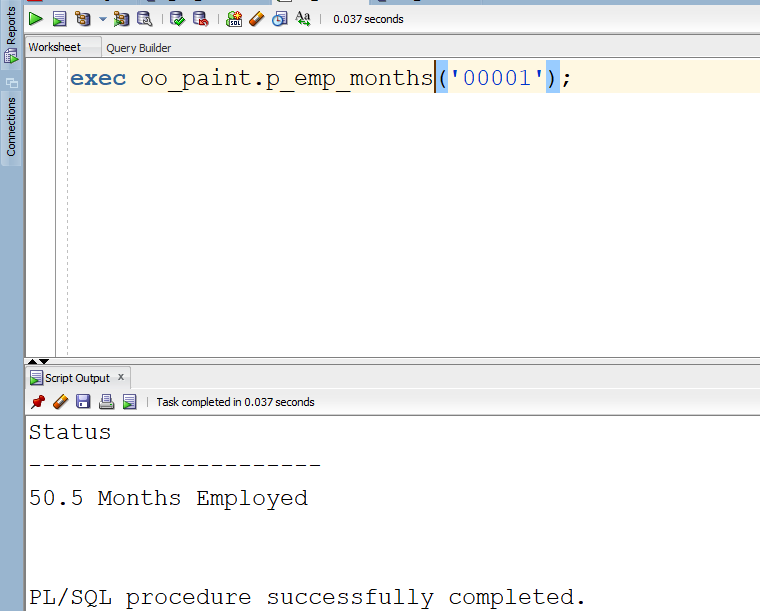
WHERE O.EMPLOYEE\_ID = emp\_id)

LOOP

dbms\_output.put\_line(a.Employment\_Status);

END LOOP;

END;



Note: For every query, you should have the following format:

* Query in English, SQL code, and result.
* Result should be readable. If the result is more than 20 lines, then show only 20 rows (selecting 20 rows should be part of your SQL program). Hint think about ROWNUM

-------------------------------------------------------------------------

Part 3: Relational Algebra, QBE, and SQL Relational Algebra: *Answer the following query using (15% points)*

* *TIME, JOIN, Left JOIN, Full JOIN, and Natural join of relational algebra*
* *TIME, JOIN, Left JOIN, Full JOIN, and Natural join of SQL*
* *QBE*

*List visitor id, visitor last name, date of visit, employee last name, employee id, product id, and product name.*

*Here “visitor” implies “Customer” and ‘date of visit” is “date sale created”*

1. *You can handwrite the queries (CLEARLY WRITTEN), except the SQL version of it, take picture of it and include the images in your report*

*SQL for the requirement*

SELECT C.CUSTOMER\_ID, C.CUSTOMER.FIRST\_NAME, CO.DATE\_CREATED, OE.EMPLOYEE.LAST\_NAME,

OE.EMPLOYEE\_ID,PO.PAINTING\_ID, P.TITLE

FROM O\_CUSTOMER C

JOIN CUSTOMER\_ORDER CO on C.CUSTOMER\_ID = CO.CUSTOMER\_ID

JOIN O\_EMPLOYEE OE on CO.EMPLOYEE\_ID = OE.EMPLOYEE\_ID

JOIN PAINTING\_ORDER PO on CO.ORDER\_ID = PO.ORDER\_ID

JOIN PAINTING P on PO.PAINTING\_ID = P.PAINTING\_ID

1. *Make sure not to submit images individual but insert them in your report*
2. *Make sure it is readable and nicely written*

-------------------------------------------------------------------------

Part 4: Data warehouse implementation using Microsoft Pivot table (15%).

Using **your DB5, create two out of three star schema strategies**

**PLEASE SEE ATTACHED FILES: 4.1 PIVOT CHART.XLSX, 4.2 PIVOT CHART.XLSX, 4.1 STAR SCHEMA, 4.2 STAR SCHEMA**

1. Sales analysis considering customer data, order data, and product data, product category, month of the customer order, and year of customer visit
   1. Draw the star schema diagram
   2. Analyze (report and chart) the sales from the different customer state divided into different month
   3. Analyze (report and chart) the sales from the different customer gender divided into different year
   4. Analyze (report and chart) the sales from the different employee (employee ID and last name) divided into different year
   5. Analyze (report and chart) the sales from the different product divided into different year
   6. Analyze (report and chart) the sales from the different product category divided into different year
2. Feedback Analysis considering customer data, feedback, product data, and product category, month of visit, and year of visit
   1. Draw the star schema diagram Need to do
   2. Analyze (report and chart) the sales from the different customer state divided into different year
   3. Analyze (report and chart) the sales from the different customer gender Done
   4. Analyze (report and chart) the sales from the different employee (employee ID and last name)
   5. Analyze (report and chart) the sales from the different product divided into different year
   6. Analyze (report and chart) the sales from the different product category divided into different month
3. Vendor service analysis by considering vendor name, date of service, amount service charge, nature of service, month of service, date of service. Your task is to develop a pivot table analysis report (report and chart) for the following inquiries:
4. Draw the star schema diagram
5. Analyze (report and chart) vendor services charged from different service category
6. Analyze (report and chart) vendor services charged from different service category for each year
7. Analyze (report and chart) vendor services charged from different service category for each month

-------------------------------------------------------------------------

Part 5: Object Relational Database (20%).

P5.1 ORDB Queries:

P5.1.1 Develop a report to show employee id, last name, home address, home phone, cell phone, job title, number of years worked for us.

P5.1.2 Develop a report to show vendor id, vendor name, vendor contact ID, vendor contact last name, home phone, cell phone, job title, and total amount billed us by vendor

P5.1.3 Answer the following queries in Part2.2

2.2.4, 2.2.5, 2.2.7, 2.2.8, and 2.2.9

-------------------------------------------------------------------------

Part 6: Graduate student question (ONLY grad students) (10%)

* Create one additional method for each object type
* Write a report (500 words) on how oracle type/ORDB is used in industries in addition to what has been discussed in class. Your report should include examples

------------------------------------------------------------

Deliverables

* Script for DB4
* Script for DB5
* Report for all parts
* Part1 Should include
  + data models derived from reverse engineering using Oracle data modeler after DB4, and ORDB
  + Report showing number of records DB4 and ORDB
* Report showing all queries in every part.
  + Query in English
  + SQL code or stored procedure or UDF
* Schema design for each strategy (total three diagrams)
* Excel document with a worksheet for each star schema strategies
* ZIP all documents with the following name   
  440-01-T3-your last name
* Submit your zip file to D2L by 11:59pm on Saturday 5/4/2019