Roll No: 2057
Subject: Database management System

Report of Factory Management System

Introduction

Factory Management System (FMS) is a computerised management and manufacturing system that automates distribution of work, from detailers to production managers through to the factory floor.

Function 1: The ability for a truss plant to operate without any paperwork

- Almost all factory paperwork can be removed when using FMS, each station shows both text details and a graphic display of the work required to be produced (just as the paperwork does now)
- Eliminates the chance of cutting and manufacturing sheets being lost or misplaced (therefore all trusses should be produced - none missed)

Function 2: The ability to control truss plant productivity and workflow

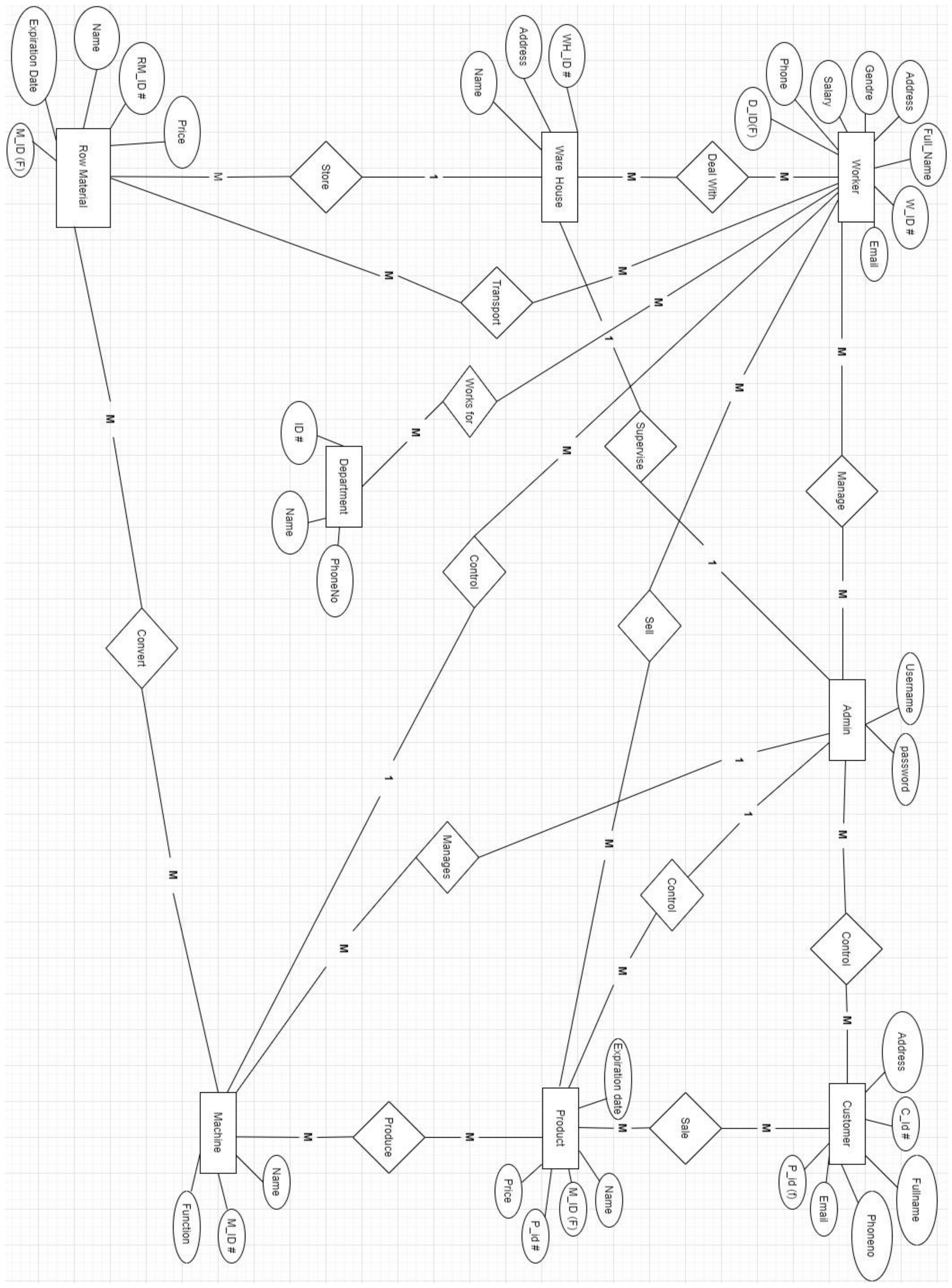
- Set and determine truss plant production scheduling
- Set the order of production - job by job
- Allocate production to stations
- Define the order/priority of production - item by item
- Monitor factory performance
- Production completed is displayed instantly to the controller
- Problems notified instantly from each station to the controller
- Downtime identified as it happens
- Review and update factory and truss costings

Requirements

SYSTEM REQUIREMENTS

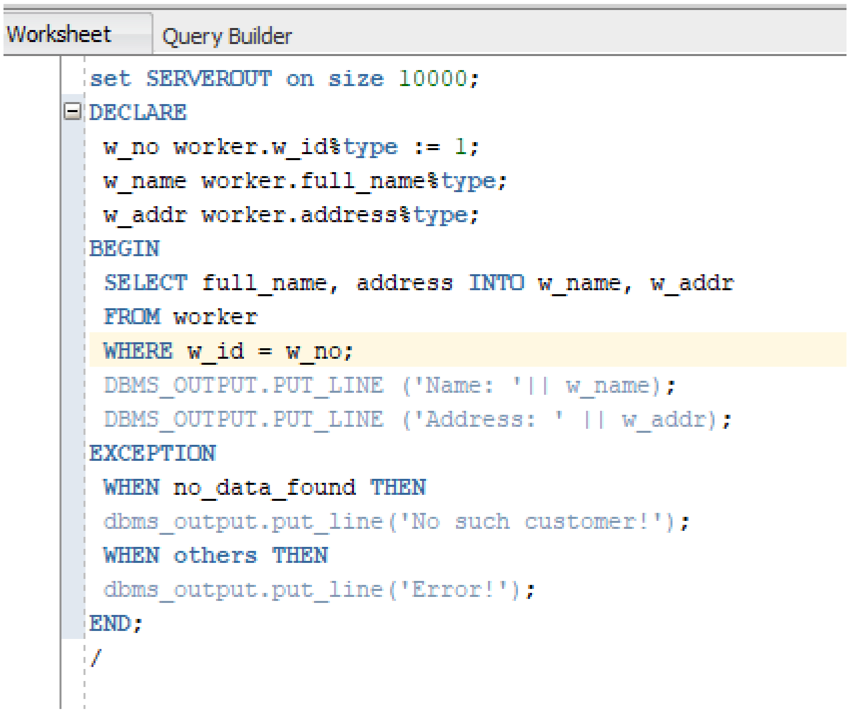
Requirements	
Windows	
Operating System	Windows XP SP3, Windows Vista SP2, Windows 10,11
CPU	Core 2 Duo at 2.4 GHz
Memory	4 GB RAM
Free Space	8 GB of free, 23.8 GB + 1 GB Swap File Space
Graphics hardware	DirectX 9.0c compatible video card. 3D Hardware Accelerator- 256MB of memory minimum
Sound hardware	DirectX 9.0c compatible sound card

ER Diagram


Expiration Date
Name
RM_ID #
Price
M_ID (F)
Row Material
Store
Name
Address
WH_ID #
Ware House
Deal With
Phone
Salary
Gendre
Address
D_ID(F)
Full_Name
W_ID #
Email
Worker
Transport
Works for
ID #
Department
Name
PhoneNo
Supervise
Manage
Control
Sell
Convert
Admin
Username
password
Manages
Control
Control
Machine
Function
M_ID #
Name
Produce
Product
Expiration date
Price
P_id #
M_ID (F)
Name
Sale
Customer
Address
C_Id #
Fullname
Phoneno
Email
P_id (f)

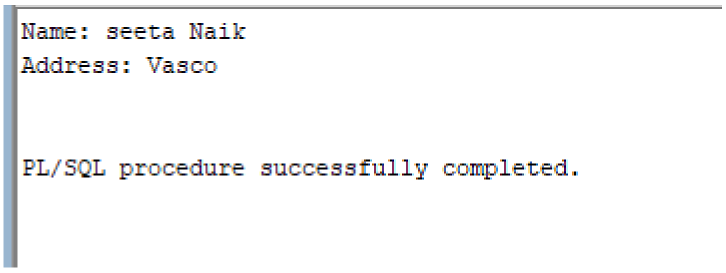
15 Reports

1. In the below code we have used procedure and passed select query which will display full name of workers and address too.



```
set SERVEROUT on size 10000;
DECLARE
 w_no worker.w_id%type := 1;
 w_name worker.full_name%type;
 w_addr worker.address%type;
BEGIN
 SELECT full_name, address INTO w_name, w_addr
 FROM worker
 WHERE w_id = w_no;
 DBMS_OUTPUT.PUT_LINE ('Name: '|| w_name);
 DBMS_OUTPUT.PUT_LINE ('Address: ' || w_addr);
EXCEPTION
 WHEN no_data_found THEN
 dbms_output.put_line('No such customer!');
 WHEN others THEN
 dbms_output.put_line('Error!');
END;
/
```

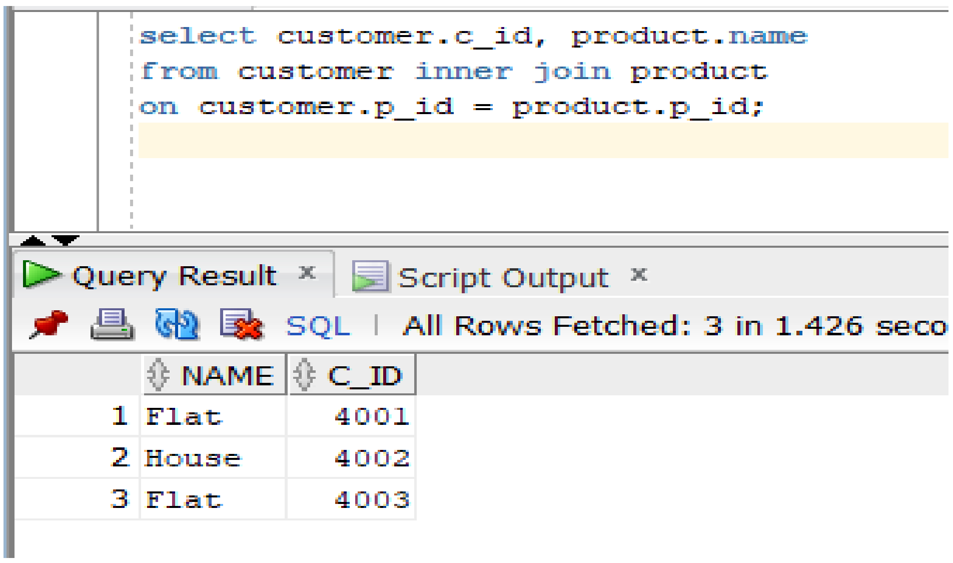
Output of the above Query



```
Name: seeta Naik
Address: Vasco


PL/SQL procedure successfully completed.
```

2. The below code will display the name of the product and customer ID which is govern by the customer.

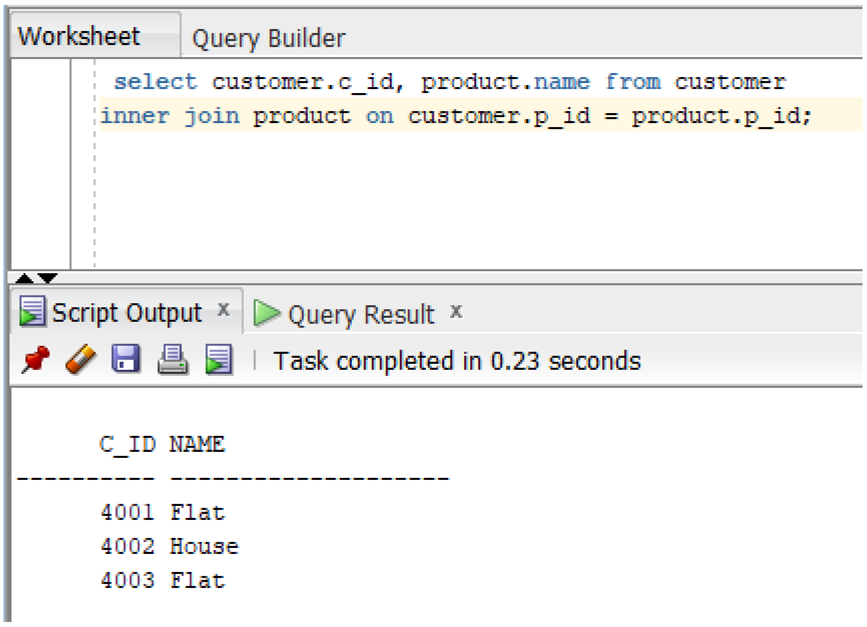


```
select customer.c_id, product.name
from customer inner join product
on customer.p_id = product.p_id;
```

All Rows Fetched: 3 in 1.426 seco

	NAME	C_ID
1	Flat	4001
2	House	4002
3	Flat	4003

3. The below code will display the name of the customer and product name which is govern by the customer ID.

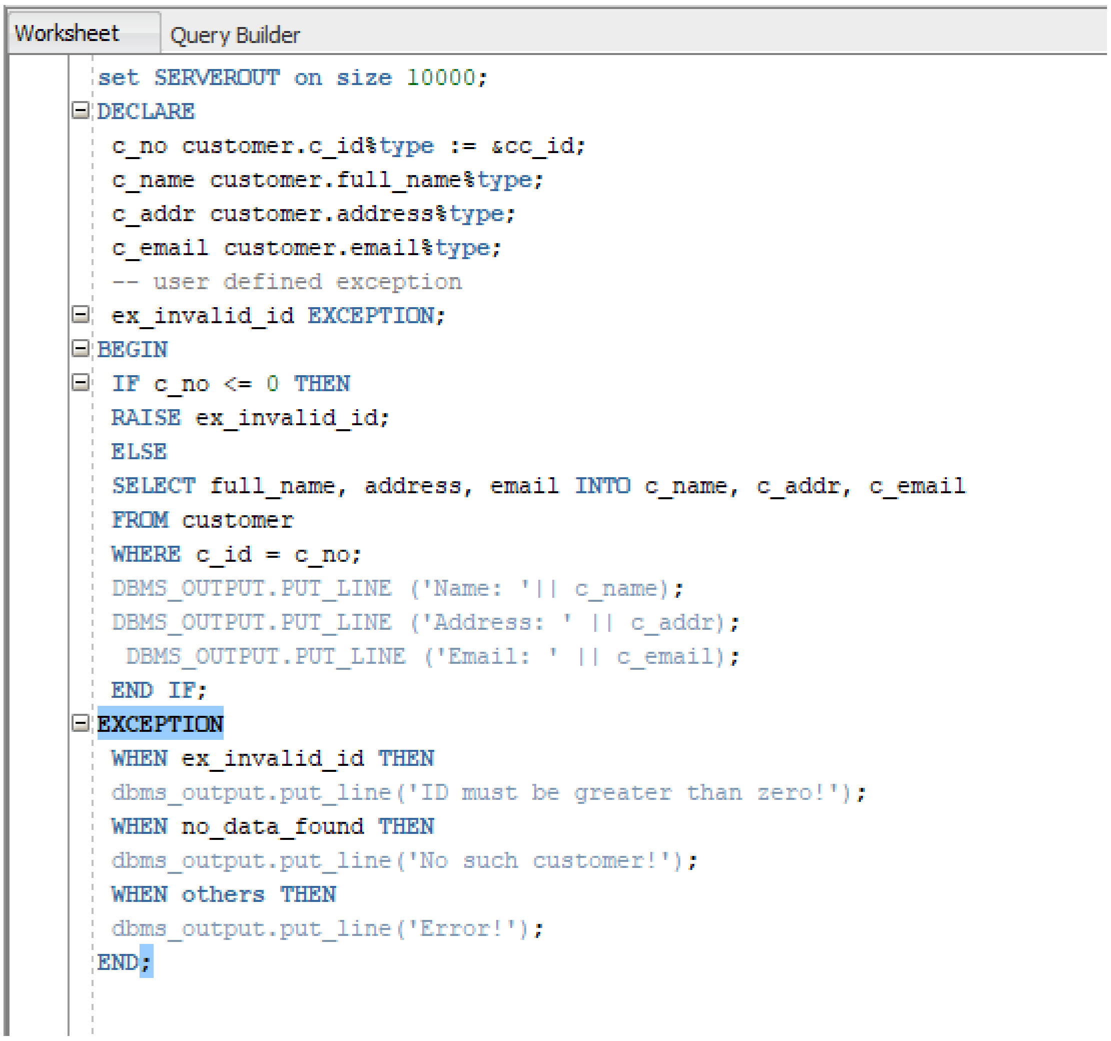


```
 select customer.c_id, product.name from customer
inner join product on customer.p_id = product.p_id;
```

Task completed in 0.23 seconds

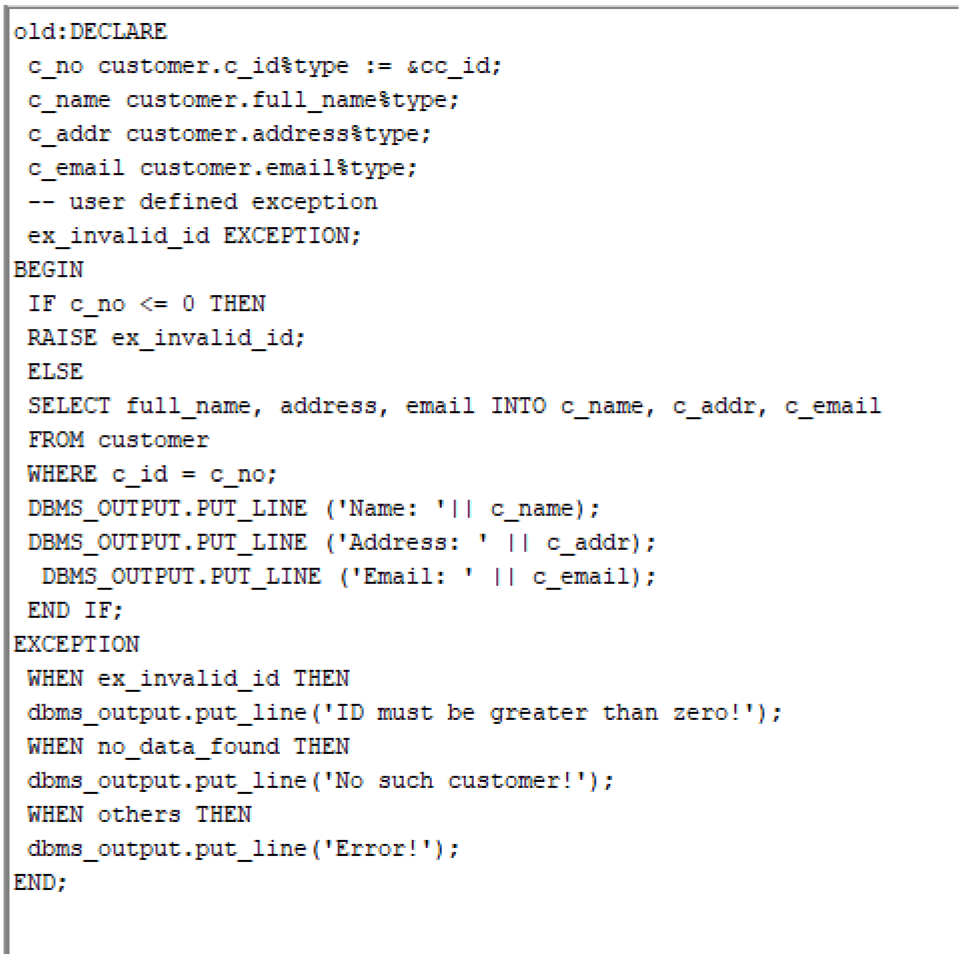
```
      C_ID NAME
---------- --------------------
      4001 Flat
      4002 House
      4003 Flat
```

4. In this query we have used Exception, if customer ID less than 0 than it will run the select query which will display the full name, address and email of the customer or else it will display the error message.



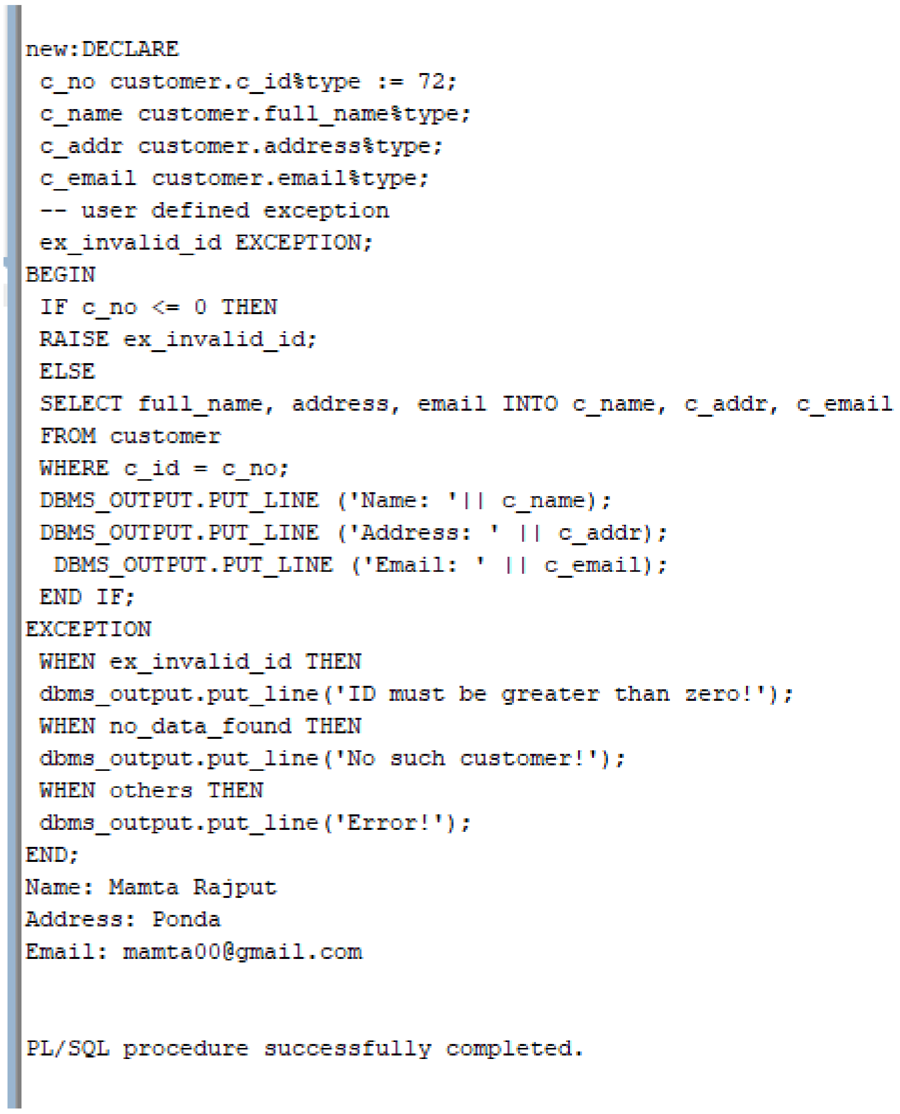
```
set SERVEROUT on size 10000;
DECLARE
 c_no customer.c_id%type := &cc_id;
 c_name customer.full_name%type;
 c_addr customer.address%type;
 c_email customer.email%type;
 -- user defined exception
 ex_invalid_id EXCEPTION;
BEGIN
 IF c_no <= 0 THEN
 RAISE ex_invalid_id;
 ELSE
 SELECT full_name, address, email INTO c_name, c_addr, c_email
 FROM customer
 WHERE c_id = c_no;
 DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
 DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
  DBMS_OUTPUT.PUT_LINE ('Email: ' || c_email);
 END IF;
EXCEPTION
 WHEN ex_invalid_id THEN
 dbms_output.put_line('ID must be greater than zero!');
 WHEN no_data_found THEN
 dbms_output.put_line('No such customer!');
 WHEN others THEN
 dbms_output.put_line('Error!');
END;
```

Output of old declaration



```
old:DECLARE
 c_no customer.c_id%type := &cc_id;
 c_name customer.full_name%type;
 c_addr customer.address%type;
 c_email customer.email%type;
 -- user defined exception
 ex_invalid_id EXCEPTION;
BEGIN
 IF c_no <= 0 THEN
 RAISE ex_invalid_id;
 ELSE
 SELECT full_name, address, email INTO c_name, c_addr, c_email
 FROM customer
 WHERE c_id = c_no;
 DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
 DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
  DBMS_OUTPUT.PUT_LINE ('Email: ' || c_email);
 END IF;
EXCEPTION
 WHEN ex_invalid_id THEN
 dbms_output.put_line('ID must be greater than zero!');
 WHEN no_data_found THEN
 dbms_output.put_line('No such customer!');
 WHEN others THEN
 dbms_output.put_line('Error!');
END;
```

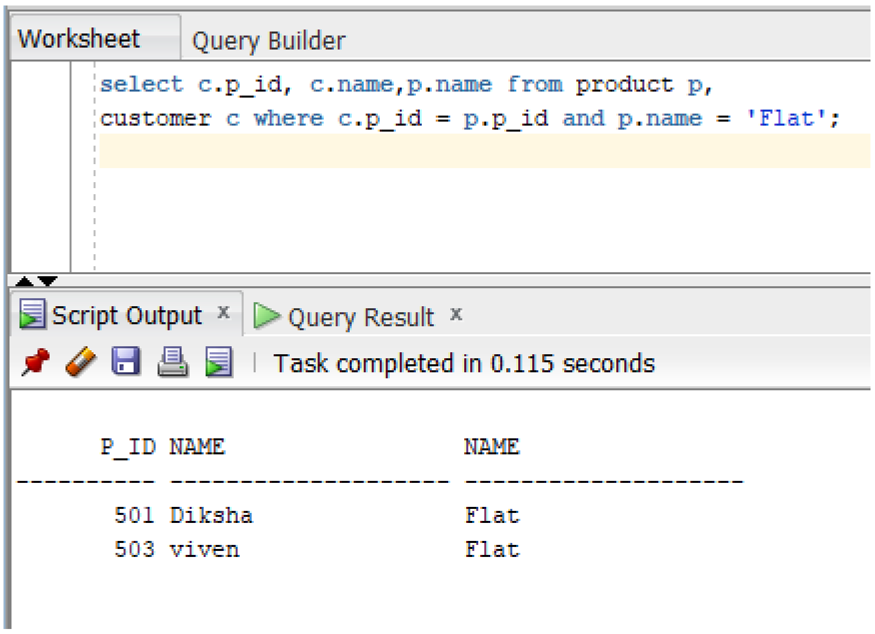
Output of new declaration after execution of query



```
new:DECLARE
 c_no customer.c_id%type := 72;
 c_name customer.full_name%type;
 c_addr customer.address%type;
 c_email customer.email%type;
 -- user defined exception
 ex_invalid_id EXCEPTION;
BEGIN
 IF c_no <= 0 THEN
 RAISE ex_invalid_id;
 ELSE
 SELECT full_name, address, email INTO c_name, c_addr, c_email
 FROM customer
 WHERE c_id = c_no;
 DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
 DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
  DBMS_OUTPUT.PUT_LINE ('Email: ' || c_email);
 END IF;
EXCEPTION
 WHEN ex_invalid_id THEN
 dbms_output.put_line('ID must be greater than zero!');
 WHEN no_data_found THEN
 dbms_output.put_line('No such customer!');
 WHEN others THEN
 dbms_output.put_line('Error!');
END;
Name: Mamta Rajput
Address: Ponda
Email: mamta00@gmail.com


PL/SQL procedure successfully completed.
```

5. Find’s all those customers who took flat. Return customer ID, name.



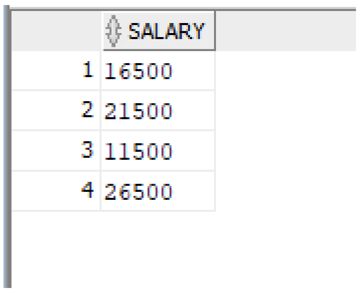
```
select c.p_id, c.name,p.name from product p,
customer c where c.p_id = p.p_id and p.name = 'Flat';
```

Task completed in 0.115 seconds

```
      P_ID NAME                 NAME
---------- -------------------- --------------------
       501 Diksha               Flat
       503 viven                Flat
```

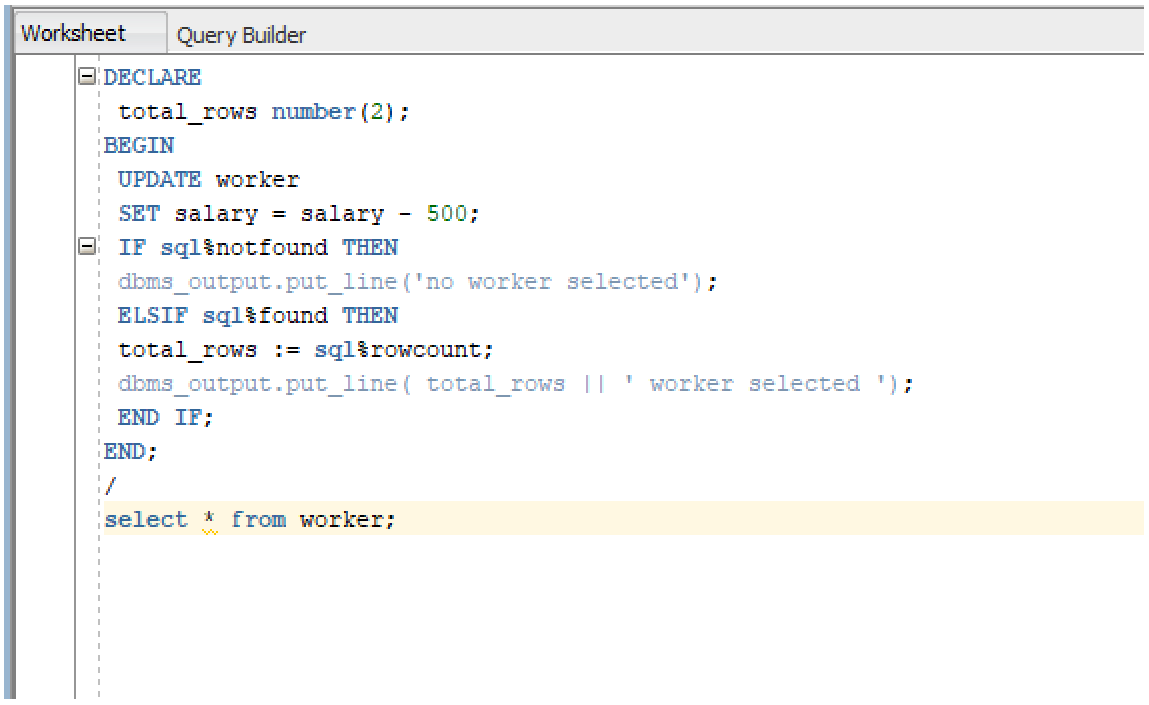
6. In this query we have used implicit Cursor and passed the update query which will decrease the actual salary by 500.

Salary Before implementing the query



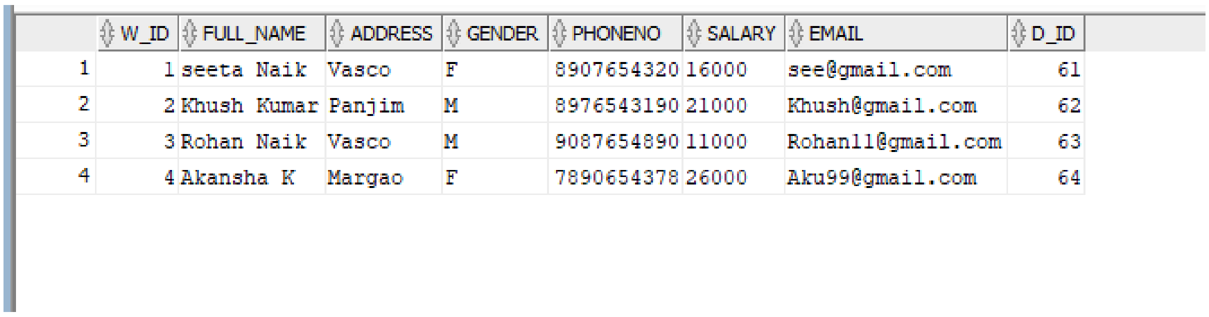
	SALARY
1	16500
2	21500
3	11500
4	26500

Code



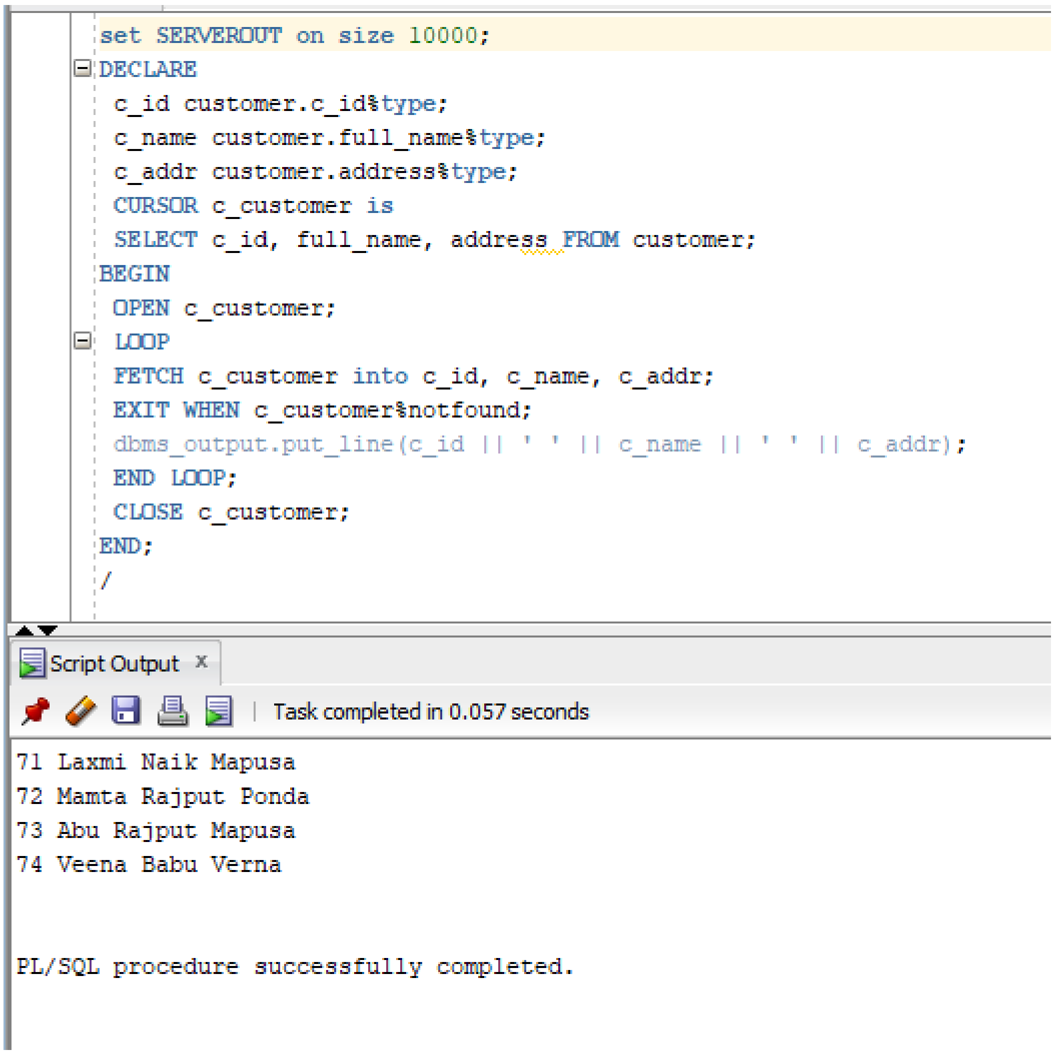
```
DECLARE
 total_rows number(2);
BEGIN
 UPDATE worker
 SET salary = salary - 500;
 IF sql%notfound THEN
 dbms_output.put_line('no worker selected');
 ELSIF sql%found THEN
 total_rows := sql%rowcount;
 dbms_output.put_line( total_rows || ' worker selected ');
 END IF;
END;
/
select * from worker;
```

Output after implementing the query salary got dedicated.



	W_ID	FULL_NAME	ADDRESS	GENDER	PHONENO	SALARY	EMAIL	D_ID
1	1	seeta Naik	Vasco	F	8907654320	16000	see@gmail.com	61
2	2	Khush Kumar	Panjim	M	8976543190	21000	Khush@gmail.com	62
3	3	Rohan Naik	Vasco	M	9087654890	11000	Rohan11@gmail.com	63
4	4	Akansha K	Margao	F	7890654378	26000	Aku99@gmail.com	64

7. In the query we have used explicit cursor and displayed customer id, full name and address from customer table by fetching the value in new variable declared.

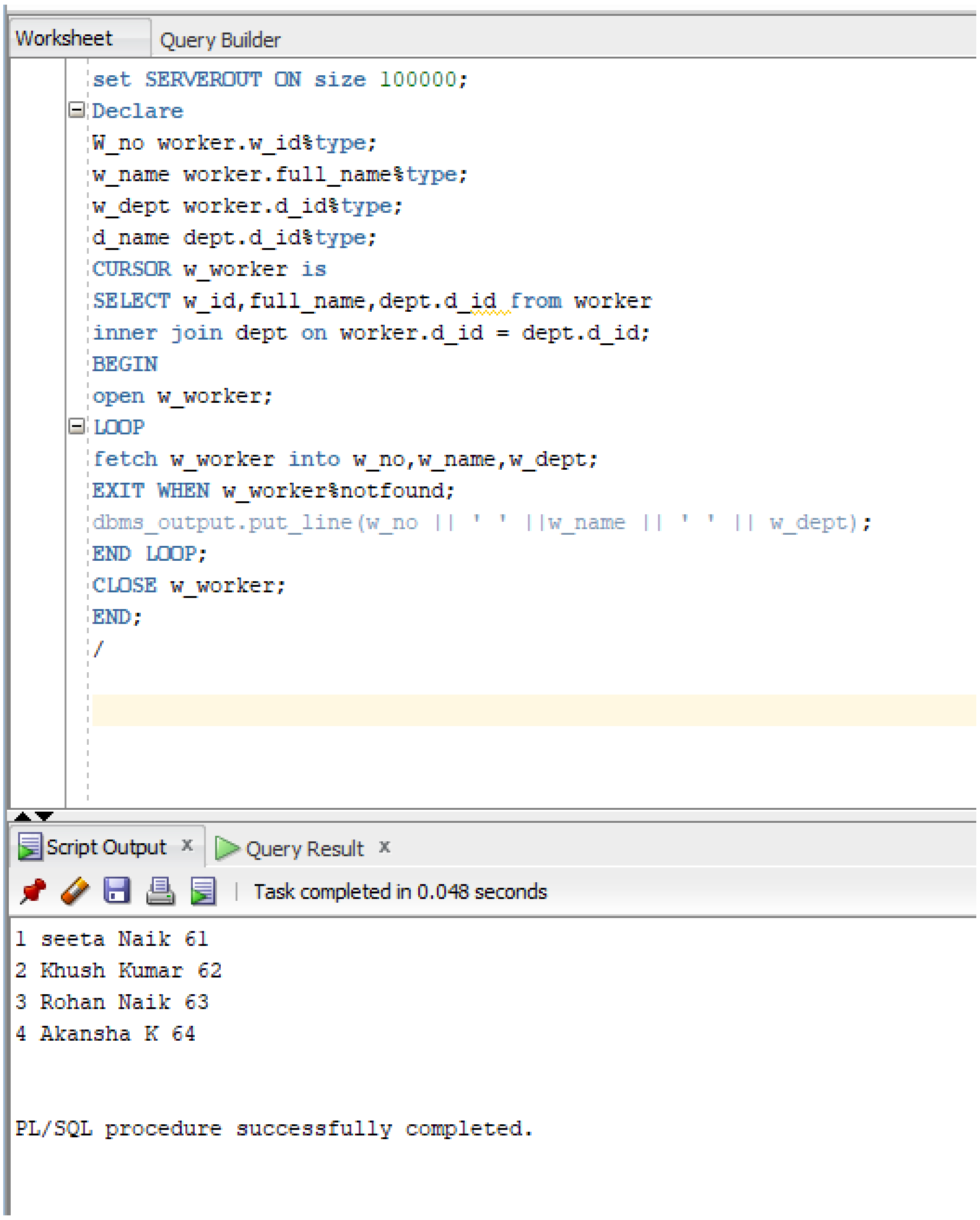


```
set SERVEROUT on size 10000;
DECLARE
 c_id customer.c_id%type;
 c_name customer.full_name%type;
 c_addr customer.address%type;
 CURSOR c_customer is
 SELECT c_id, full_name, address FROM customer;
BEGIN
 OPEN c_customer;
 LOOP
 FETCH c_customer into c_id, c_name, c_addr;
 EXIT WHEN c_customer%notfound;
 dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
 END LOOP;
 CLOSE c_customer;
END;
/
```

```
71 Laxmi Naik Mapusa
72 Mamta Rajput Ponda
73 Abu Rajput Mapusa
74 Veena Babu Verna


PL/SQL procedure successfully completed.
```

8. In the query we have used explicit cursor and displayed worker id, full name and department ID from worker table by using inner join with department, fetched the value in new variable declared.

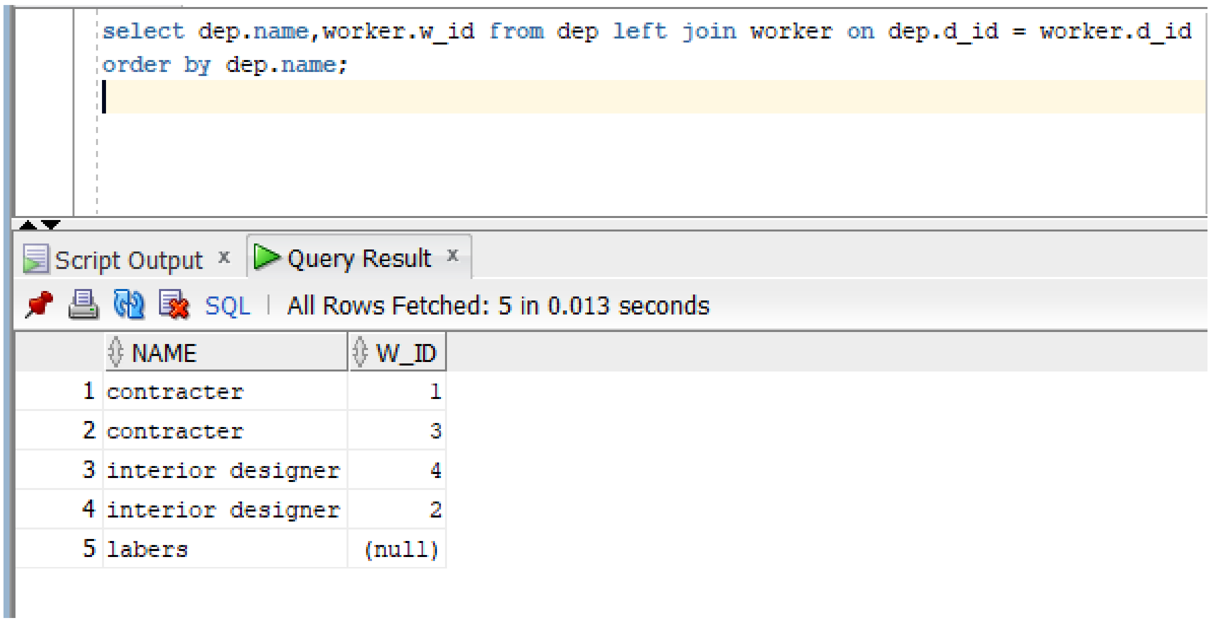


```
set SERVEROUT ON size 100000;
Declare
W_no worker.w_id%type;
w_name worker.full_name%type;
w_dept worker.d_id%type;
d_name dept.d_id%type;
CURSOR w_worker is
SELECT w_id,full_name,dept.d_id from worker
inner join dept on worker.d_id = dept.d_id;
BEGIN
open w_worker;
LOOP
fetch w_worker into w_no,w_name,w_dept;
EXIT WHEN w_worker%notfound;
dbms_output.put_line(w_no || ' ' ||w_name || ' ' || w_dept);
END LOOP;
CLOSE w_worker;
END;
/
```

```
1 seeta Naik 61
2 Khush Kumar 62
3 Rohan Naik 63
4 Akansha K 64


PL/SQL procedure successfully completed.
```

9. The below query is used to display department name and worker ID from department which has been left joined to workers and the department name will be displayed in the ascending order.

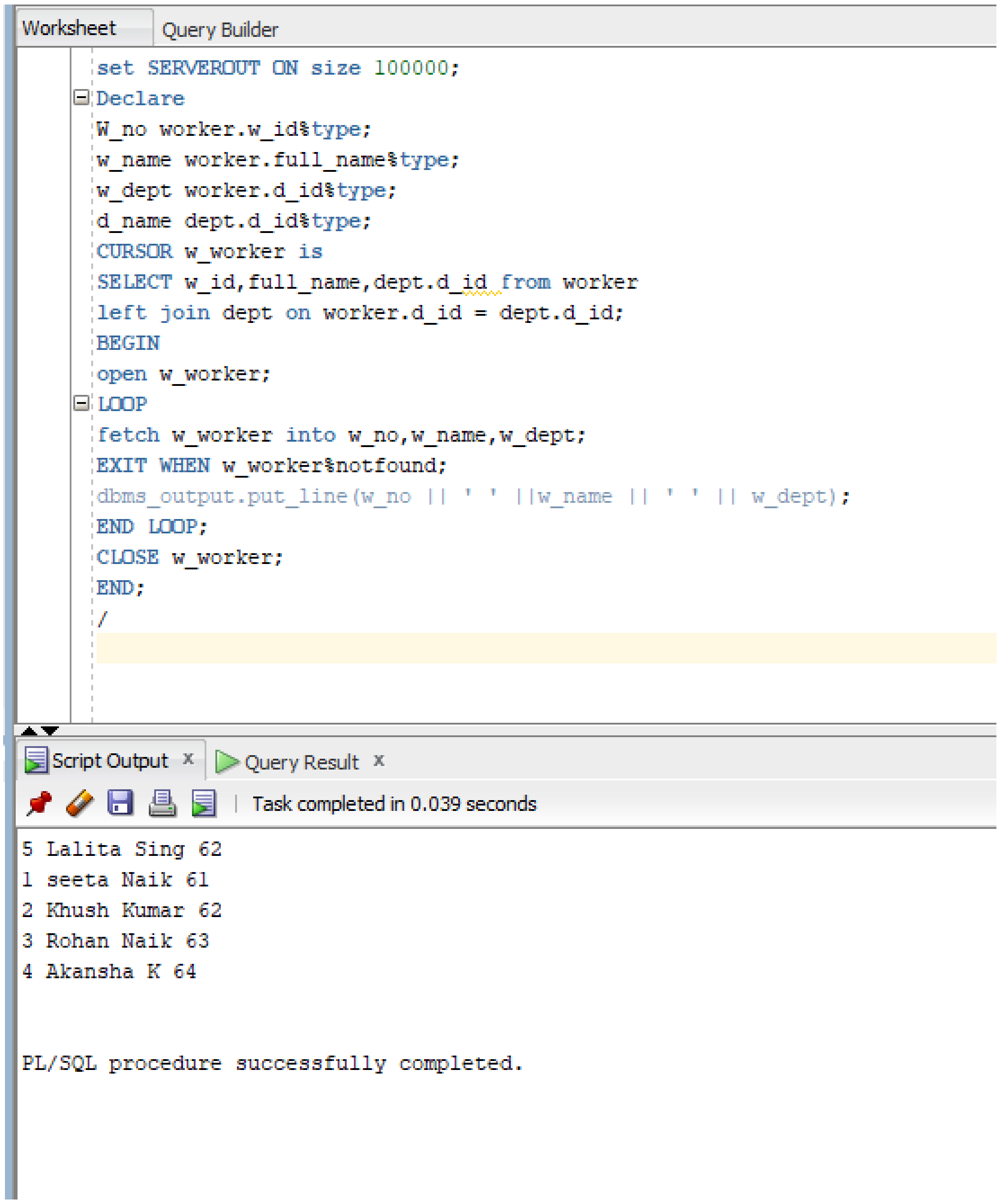


```
select dep.name,worker.w_id from dep left join worker on dep.d_id = worker.d_id
order by dep.name;
```

All Rows Fetched: 5 in 0.013 seconds

	NAME	W_ID
1	contracter	1
2	contracter	3
3	interior designer	4
4	interior designer	2
5	labers	(null)

10. In the query we have used explicit cursor and displayed worker id, full name and department ID from worker table by using left join with department, fetched the value in new variable declared.



Worksheet | Query Builder

```
set SERVEROUT ON size 100000;
Declare
W_no worker.w_id%type;
w_name worker.full_name%type;
w_dept worker.d_id%type;
d_name dept.d_id%type;
CURSOR w_worker is
SELECT w_id,full_name,dept.d_id from worker
left join dept on worker.d_id = dept.d_id;
BEGIN
open w_worker;
LOOP
fetch w_worker into w_no,w_name,w_dept;
EXIT WHEN w_worker%notfound;
dbms_output.put_line(w_no  ||  ' '  ||w_name  ||  ' '  || w_dept);
END LOOP;
CLOSE w_worker;
END;
/
```

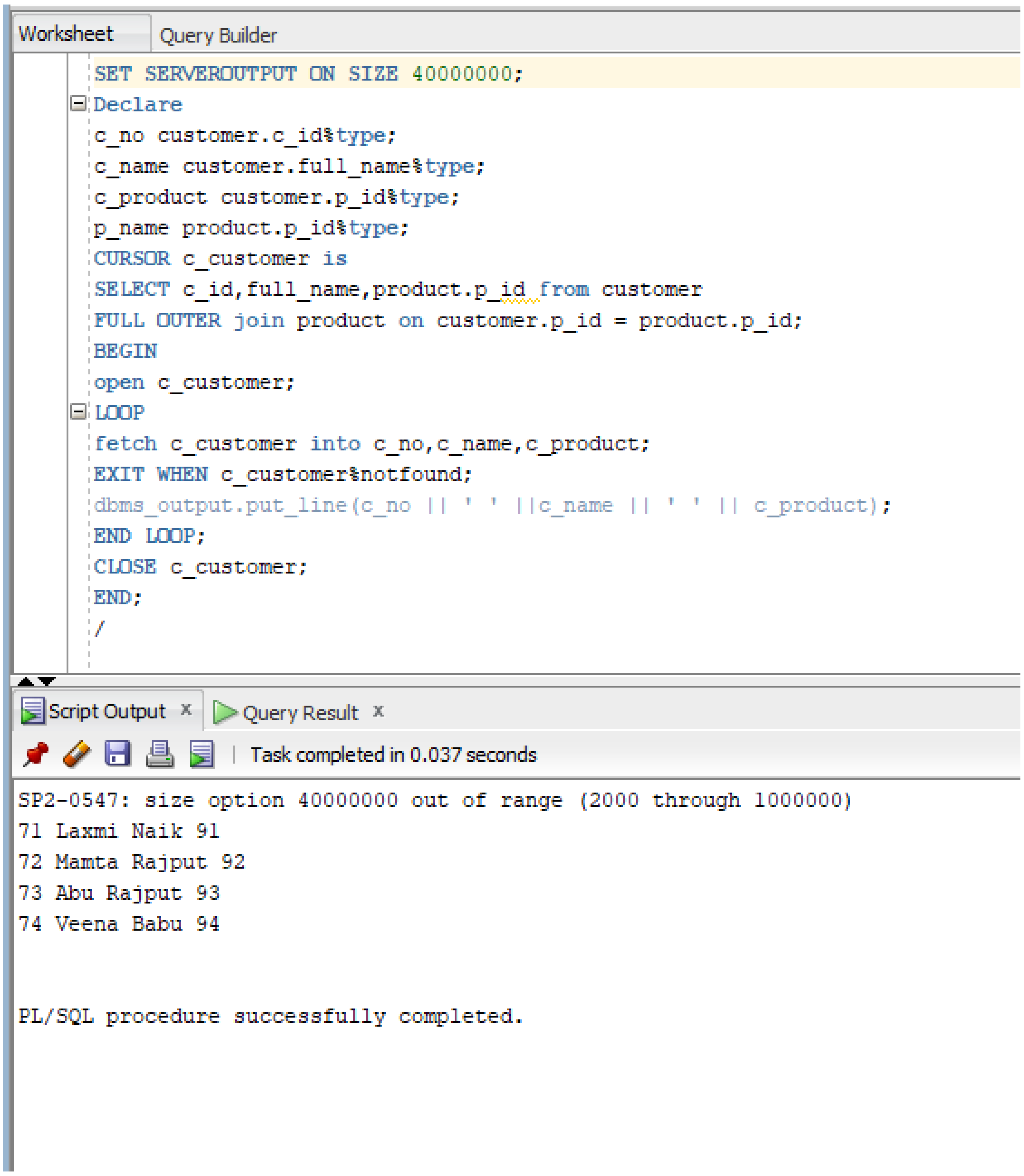
Script Output | Query Result

Task completed in 0.039 seconds

```
5 Lalita Sing 62
1 seeta Naik 61
2 Khush Kumar 62
3 Rohan Naik 63
4 Akansha K 64


PL/SQL procedure successfully completed.
```

11. In the query we have used explicit cursor and displayed customer id, full name and product ID from customer table by using full outer join with product, fetched the value in new variable declared.

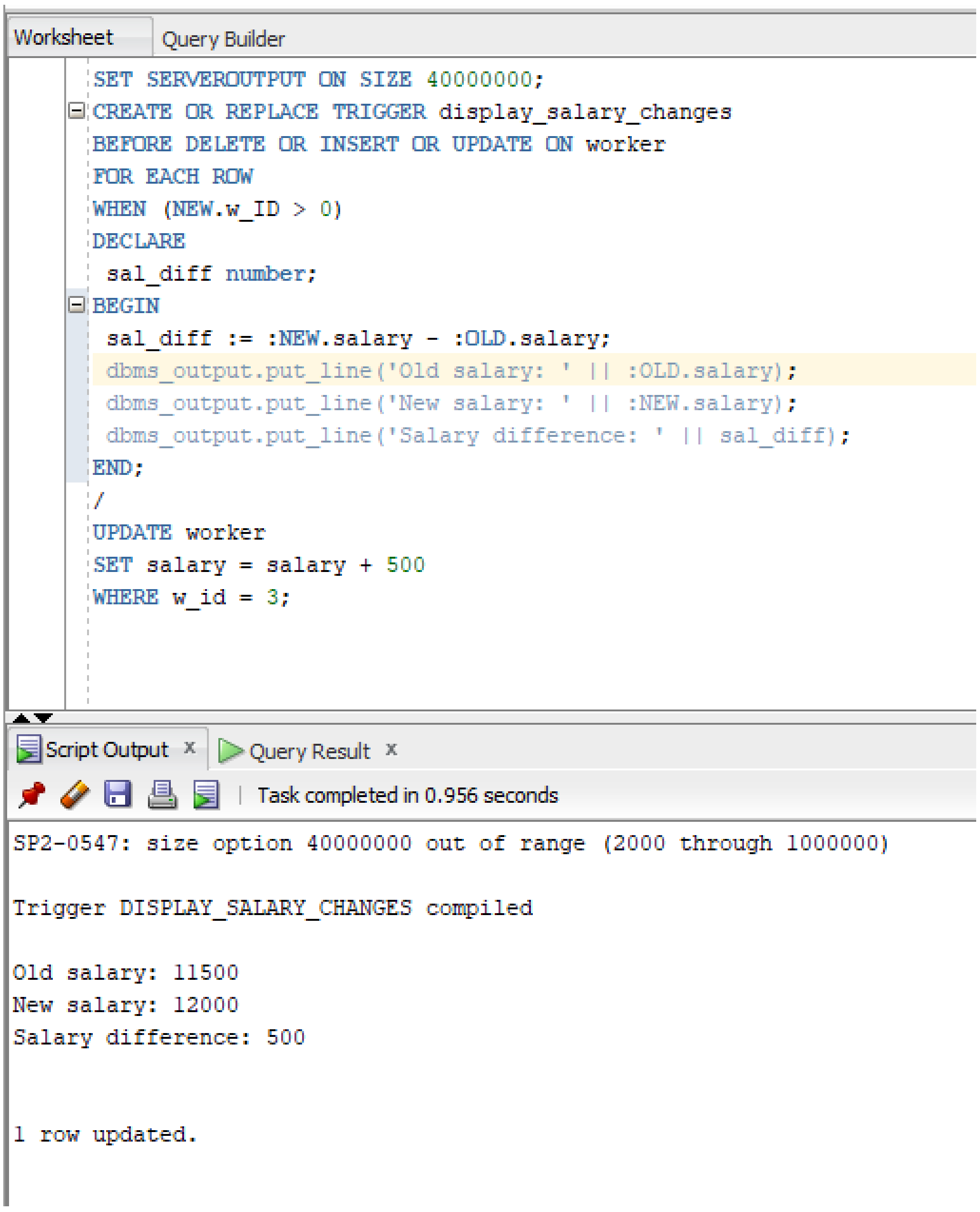


```
SET SERVEROUTPUT ON SIZE 40000000;
Declare
c_no customer.c_id%type;
c_name customer.full_name%type;
c_product customer.p_id%type;
p_name product.p_id%type;
CURSOR c_customer is
SELECT c_id,full_name,product.p_id from customer
FULL OUTER join product on customer.p_id = product.p_id;
BEGIN
open c_customer;
LOOP
fetch c_customer into c_no,c_name,c_product;
EXIT WHEN c_customer%notfound;
dbms_output.put_line(c_no || ' ' ||c_name || ' ' || c_product);
END LOOP;
CLOSE c_customer;
END;
/
```

```
SP2-0547: size option 40000000 out of range (2000 through 1000000)
71 Laxmi Naik 91
72 Mamta Rajput 92
73 Abu Rajput 93
74 Veena Babu 94


PL/SQL procedure successfully completed.
```

12. In the below query we have used trigger and updated the value of salary in the table worker.



```
SET SERVEROUTPUT ON SIZE 40000000;
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON worker
FOR EACH ROW
WHEN (NEW.w_ID > 0)
DECLARE
 sal_diff number;
BEGIN
 sal_diff := :NEW.salary - :OLD.salary;
 dbms_output.put_line('Old salary: ' || :OLD.salary);
 dbms_output.put_line('New salary: ' || :NEW.salary);
 dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
UPDATE worker
SET salary = salary + 500
WHERE w_id = 3;
```

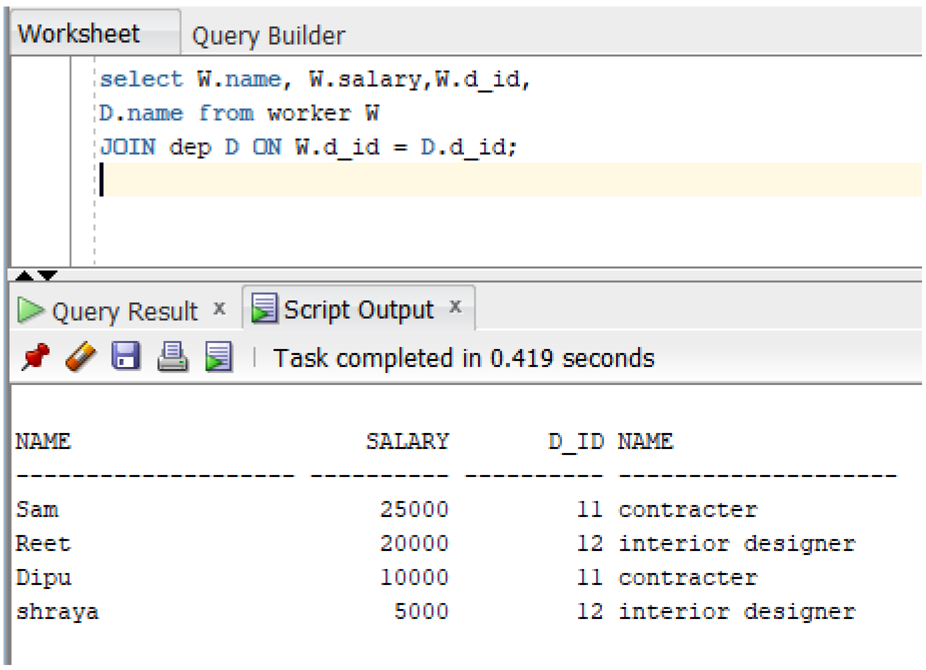
```
SP2-0547: size option 40000000 out of range (2000 through 1000000)

Trigger DISPLAY_SALARY_CHANGES compiled

Old salary: 11500
New salary: 12000
Salary difference: 500


1 row updated.
```

13. The below command will display the name, salary of workers and department ID and Name correspond to workers.

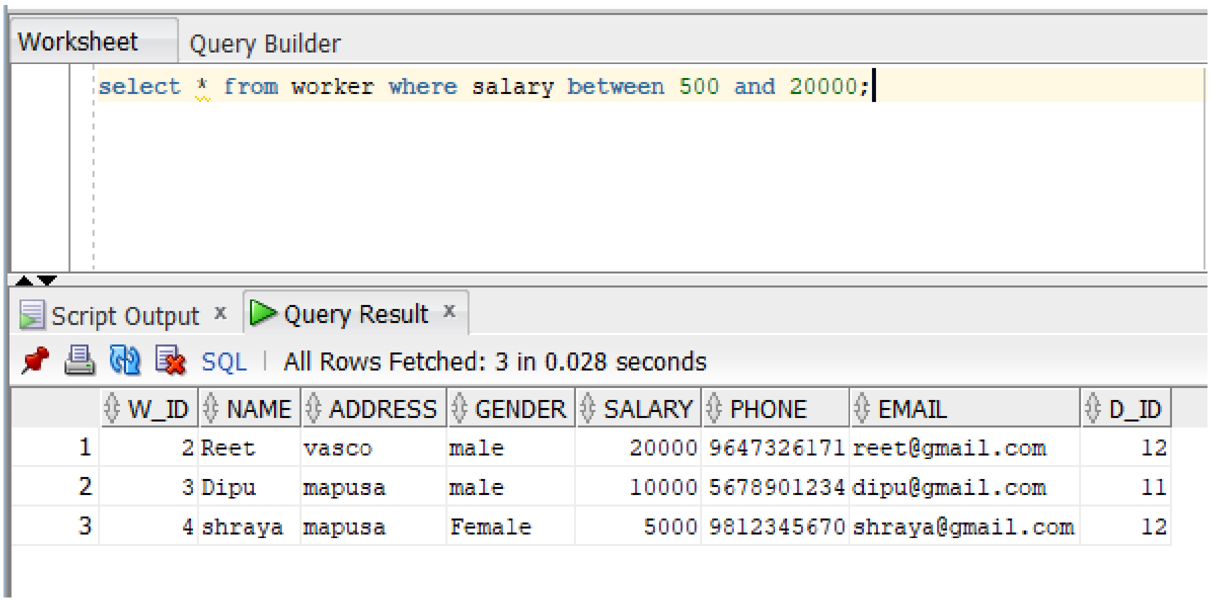


```
select W.name, W.salary,W.d_id,
D.name from worker W
JOIN dep D ON W.d_id = D.d_id;
```

Task completed in 0.419 seconds

```
NAME                     SALARY       D_ID NAME
-------------------- ---------- ---------- --------------------
Sam                       25000         11 contracter
Reet                      20000         12 interior designer
Dipu                      10000         11 contracter
shraya                     5000         12 interior designer
```

14. The below query will display worker salary ranging from 500 to 20000.

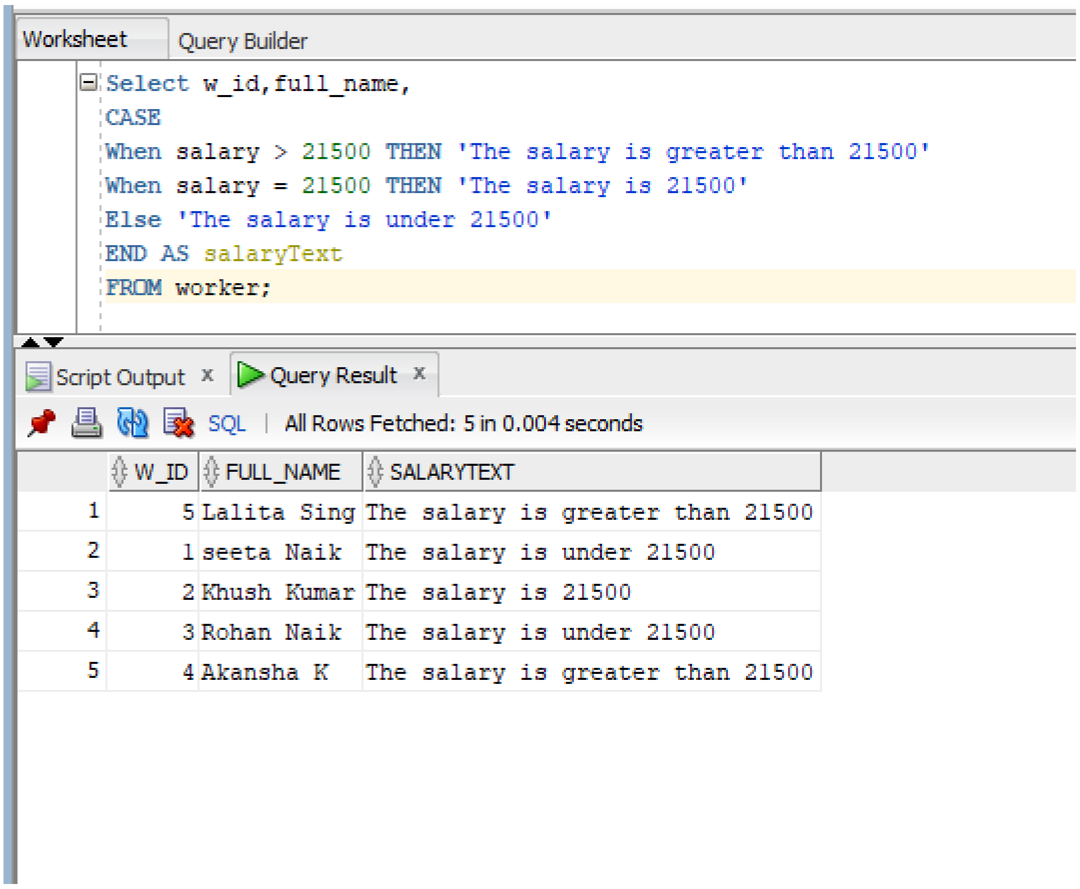


```
select * from worker where salary between 500 and 20000;
```

All Rows Fetched: 3 in 0.028 seconds

	W_ID	NAME	ADDRESS	GENDER	SALARY	PHONE	EMAIL	D_ID
1	2	Reet	vasco	male	20000	9647326171	reet@gmail.com	12
2	3	Dipu	mapusa	male	10000	5678901234	dipu@gmail.com	11
3	4	shraya	mapusa	Female	5000	9812345670	shraya@gmail.com	12

15. In this query we have used SQL case and displayed the message in salarytext column accordingly.



```
Select w_id,full_name,
CASE
When salary > 21500 THEN 'The salary is greater than 21500'
When salary = 21500 THEN 'The salary is 21500'
Else 'The salary is under 21500'
END AS salaryText
FROM worker;
```

	W_ID	FULL_NAME	SALARYTEXT
1	5	Lalita Sing	The salary is greater than 21500
2	1	seeta Naik	The salary is under 21500
3	2	Khush Kumar	The salary is 21500
4	3	Rohan Naik	The salary is under 21500
5	4	Akansha K	The salary is greater than 21500

Conclusion:

This factory information management system is developed in such a way that future modifications can be done easily. A searching option could be added such that one can directly search to the particular product factory from this.

It has been noted that automation of the entire FMS system improves efficiency, providing a more user-friendly graphical interface. Currently the system gives access to all authorized users and effectively overcomes the delay in communications. System security, data security, and reliability are all very good.