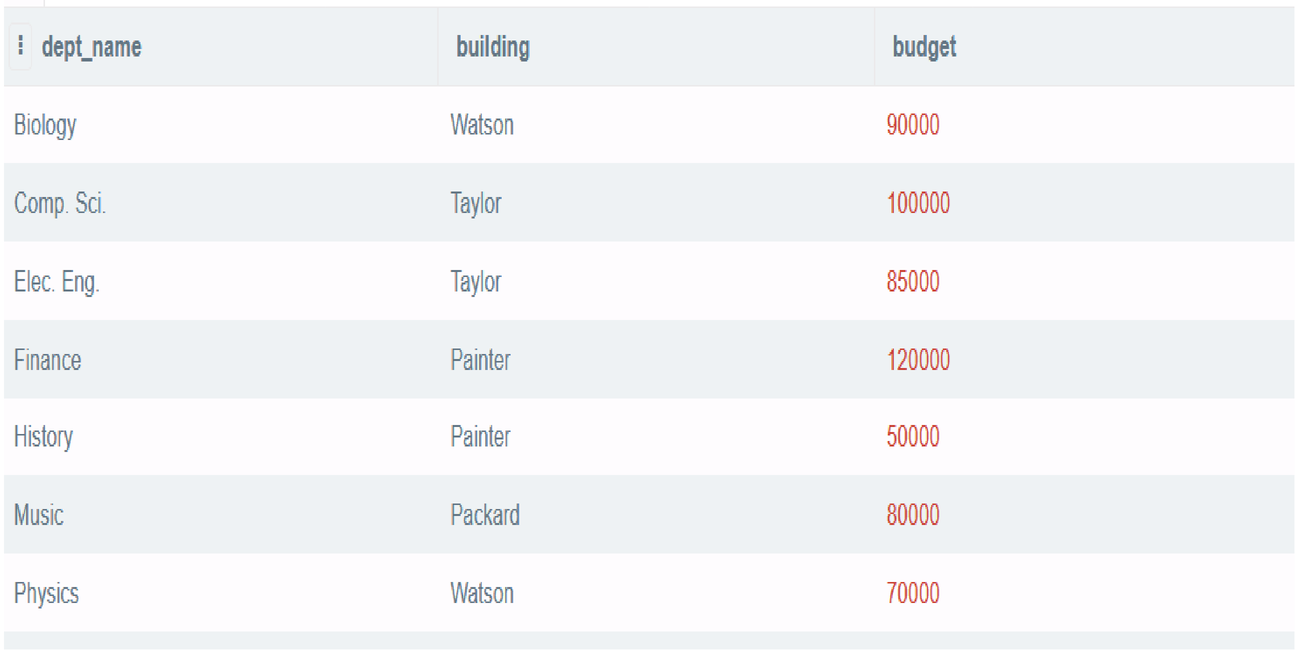
WEEK 1

1) Table department

```
CREATE TABLE department(
dept_name VARCHAR(100),
building VARCHAR(100),
budget DOUBLE
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Biology', 'Watson', 90000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Comp. Sci.', 'Taylor', 100000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Elec. Eng.', 'Taylor', 85000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Finance', 'Painter', 120000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'History', 'Painter', 50000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Music', 'Packard', 80000
);

INSERT INTO department(dept_name, building, budget)
VALUES
(
'Physics', 'Watson', 70000
);

select * from department;
```



dept_name	building	budget
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

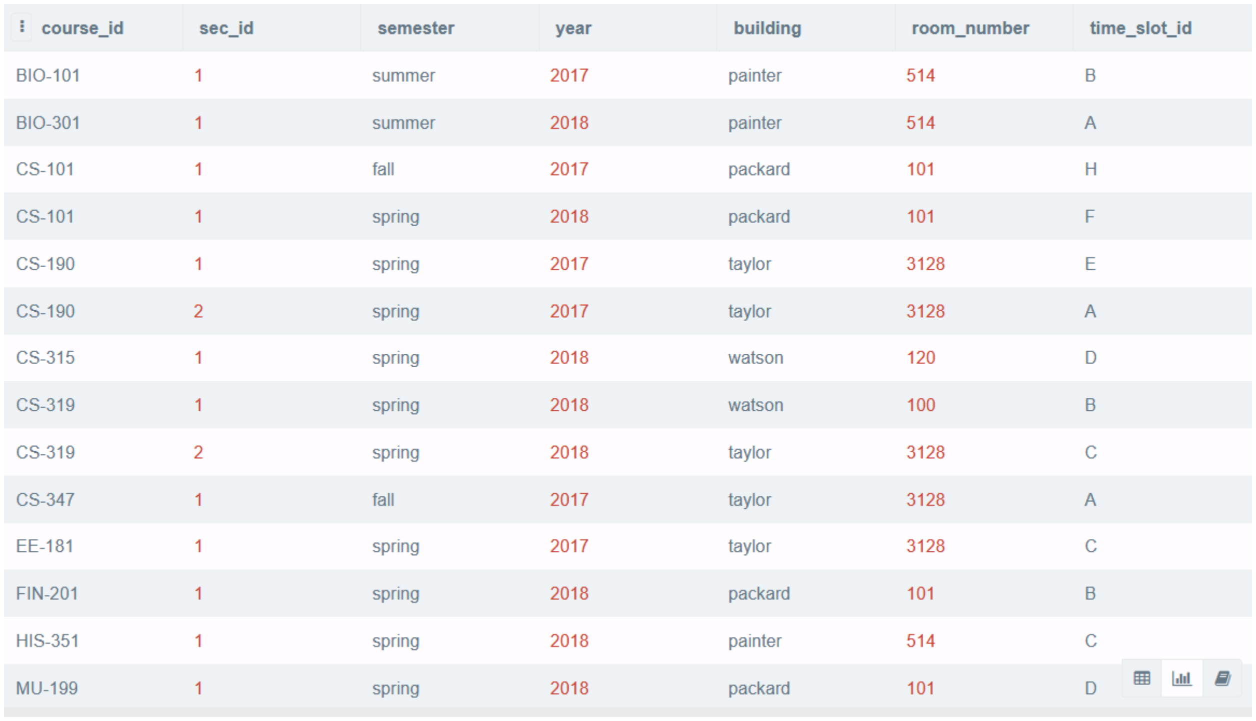
Table section

```
CREATE TABLE section(
course_id VARCHAR(100),
sec_id DOUBLE,
semester VARCHAR(100),
year DOUBLE,
building VARCHAR(100),
room_number DOUBLE,
time_slot_id VARCHAR(100)
);
```

```
INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'BIO-101', 1, 'summer', 2017, 'painter', 514, 'B'
);

INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'BIO-301', 1, 'summer', 2018, 'painter', 514, 'A'
);

INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-101', 1, 'fall', 2017, 'packard', 101, 'H'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-101', 1, 'spring', 2018, 'packard', 101, 'F'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-190', 1, 'spring', 2017, 'taylor', 3128, 'E'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-190', 2, 'spring', 2017, 'taylor', 3128, 'A'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-315', 1, 'spring', 2018, 'watson', 120, 'D'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-319', 1, 'spring', 2018, 'watson', 100, 'B'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-319', 2, 'spring', 2018, 'taylor', 3128, 'C'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'CS-347', 1, 'fall', 2017, 'taylor', 3128, 'A'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'EE-181', 1, 'spring', 2017, 'taylor', 3128, 'C'
);


INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'FIN-201', 1, 'spring', 2018, 'packard', 101, 'B'
);

INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'HIS-351', 1, 'spring', 2018, 'painter', 514, 'C'
);
```

INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'MU-199', 1, 'spring', 2018, 'packard', 101, 'D'
);

INSERT INTO section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'PHY-101', 1, 'fall', 2017, 'watson', 100, 'A'
);



course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	summer	2017	painter	514	B
BIO-301	1	summer	2018	painter	514	A
CS-101	1	fall	2017	packard	101	H
CS-101	1	spring	2018	packard	101	F
CS-190	1	spring	2017	taylor	3128	E
CS-190	2	spring	2017	taylor	3128	A
CS-315	1	spring	2018	watson	120	D
CS-319	1	spring	2018	watson	100	B
CS-319	2	spring	2018	taylor	3128	C
CS-347	1	fall	2017	taylor	3128	A
EE-181	1	spring	2017	taylor	3128	C
FIN-201	1	spring	2018	packard	101	B
HIS-351	1	spring	2018	painter	514	C
MU-199	1	spring	2018	packard	101	D

2)ALTER TABLE section
ADD (course_id, sec_id, semester, year, building, room_number, time_slot_id)
VALUES
(
'PHY-102', 2, 'summerl', 2021, 'watson', 90, 'A'
);

3)
syntax
DESCRIBE table_name

Example
DESCRIBE section

4) syntax
RENAME TABLE old_table TO new_table;

Example
ALTER TABLE section
RENAME to class;

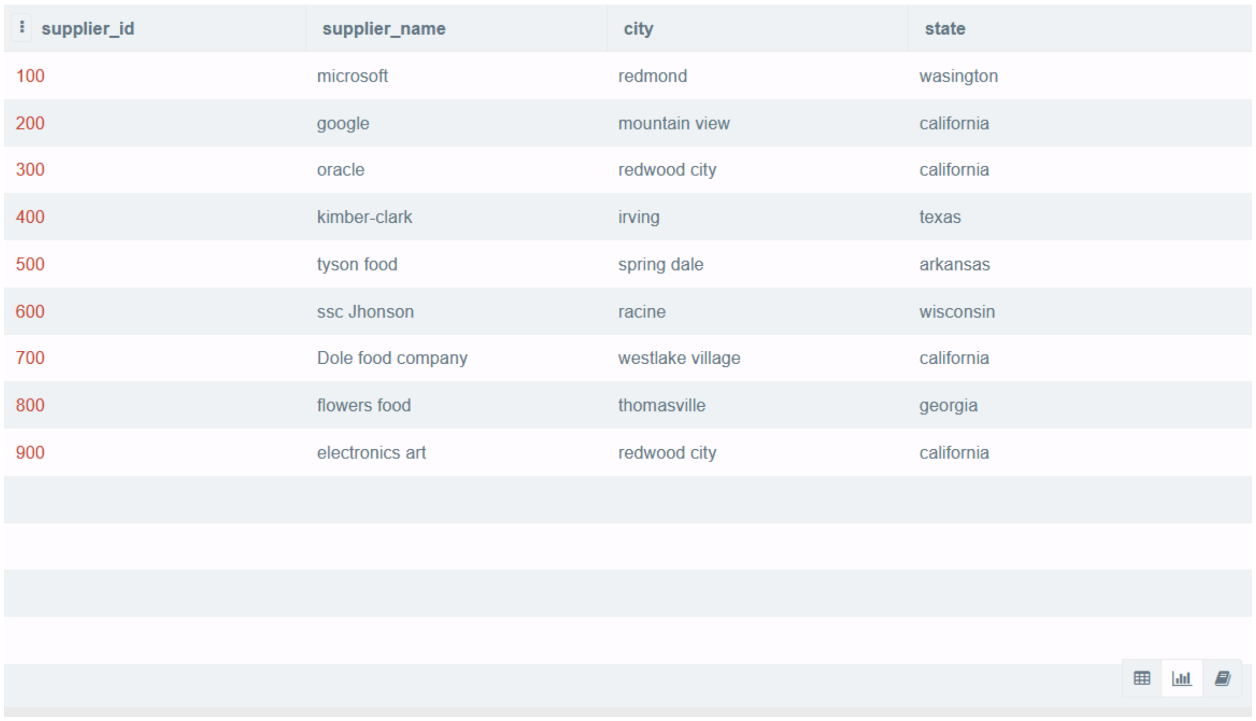
5)**TRUNCATE**
TRUNCATE Command is a Data Definition Language operation. It is used to remove all the records from a table. It deletes all the records from an existing table but not the table itself.

DROP
DROP statement is a Data Definition Language(DDL) Command which is used to delete existing database objects. It can be used to delete databases

6)
- Column can’t be deleted with alter command.
- Column can’t be renamed a column.
- Column can’t be added in between of the existing columns.
- When a column is added, it will be added at the end of the table.

```
CREATE TABLE suppliers(
supplier_id DOUBLE,
supplier_name VARCHAR(100),
city VARCHAR(100),
state VARCHAR(100)
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
100, 'microsoft', 'redmond', 'wasington'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
200, 'google', 'mountain view', 'california'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
300, 'oracle', 'redwood city', 'california'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
400, 'kimber-clark', 'irving', 'texas'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
500, 'tyson food', 'spring dale', 'arkansas'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
```

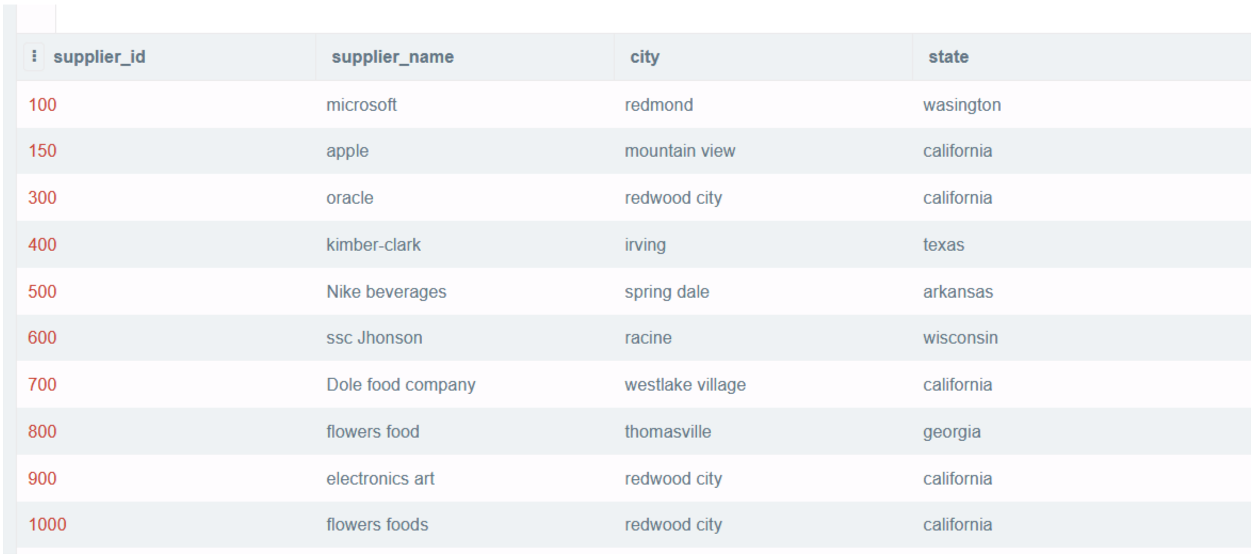
```
VALUES
(
600, 'ssc Jhonson', 'racine', 'wisconsin'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
700, 'Dole food company', 'westlake village', 'california'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
800, 'flowers food', 'thomasville', 'georgia'
);


INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
900, 'electronics art', 'redwood city', 'california'
);
```



supplier_id	supplier_name	city	state
100	microsoft	redmond	wasington
200	google	mountain view	california
300	oracle	redwood city	california
400	kimber-clark	irving	texas
500	tyson food	spring dale	arkansas
600	ssc Jhonson	racine	wisconsin
700	Dole food company	westlake village	california
800	flowers food	thomasville	georgia
900	electronics art	redwood city	california

Exercise 2

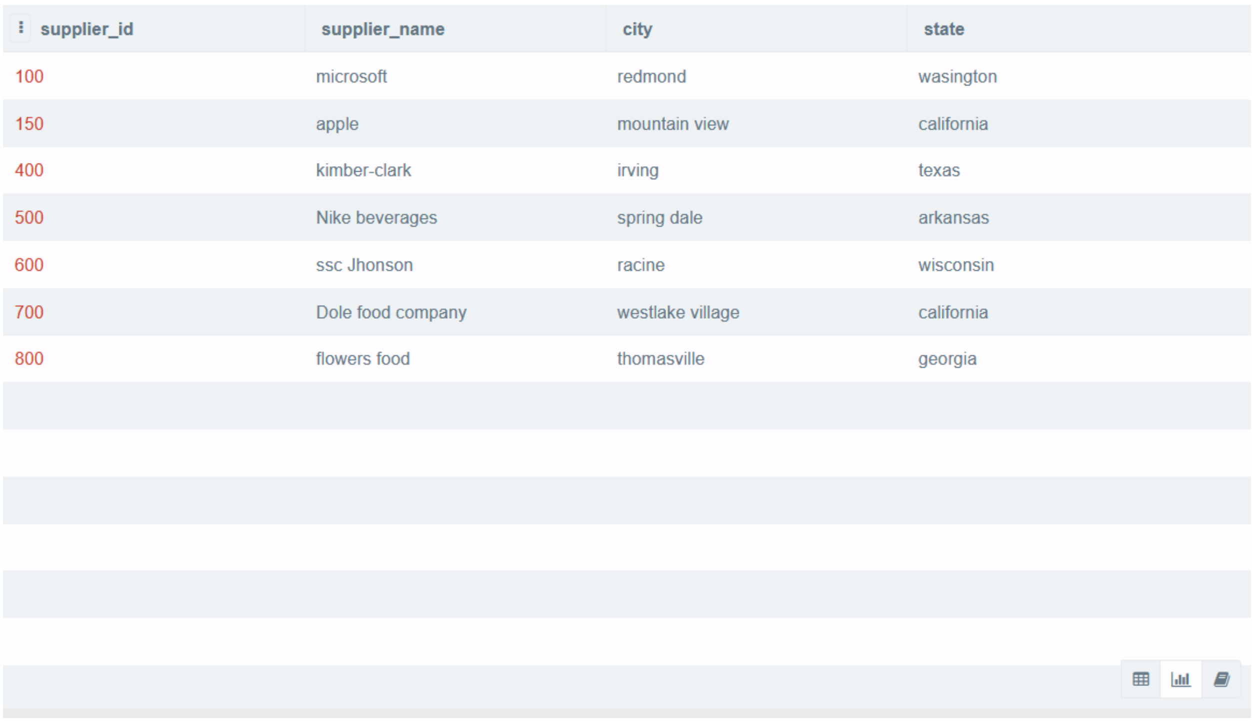
```
UPDATE suppliers set supplier_name='Nike beverages' where city='arkansas';
UPDATE suppliers set supplier_name='apple',supplier_id=150 where supplier_name='google';
INSERT INTO suppliers(supplier_id, supplier_name, city, state)
VALUES
(
1000, 'flowers foods', 'redwood city', 'california'
);
```



supplier_id	supplier_name	city	state
100	microsoft	redmond	wasington
150	apple	mountain view	california
300	oracle	redwood city	california
400	kimber-clark	irving	texas
500	Nike beverages	spring dale	arkansas
600	ssc Jhonson	racine	wisconsin
700	Dole food company	westlake village	california
800	flowers food	thomasville	georgia
900	electronics art	redwood city	california
1000	flowers foods	redwood city	california

Exercise 3

```
DELETE FROM suppliers WHERE city = 'Redwood City';
DELETE FROM suppliers WHERE supplier_name = 'Kimberly-Clark';
```



supplier_id	supplier_name	city	state
100	microsoft	redmond	wasington
150	apple	mountain view	california
400	kimber-clark	irving	texas
500	Nike beverages	spring dale	arkansas
600	ssc Jhonson	racine	wisconsin
700	Dole food company	westlake village	california
800	flowers food	thomasville	georgia

1. Write the various datatypes that are support by SQL

1. Numeric data types such as int, float etc.
2. Date and Time data types such as Date, Time, Datetime etc.
3. Character and String data types such as char, varchar, etc.
4. Binary data types such as binary, varbinary etc.
5. Miscellaneous data types – clob, blob, xml, cursor, table etc.

2)Justify whether pseudo table can be created using select statement
you can select from Pseudocolumns,Pseudocolumns are not actual columns in a table but they behave like columns.you cannot insert into, update, or delete from a pseudocolumn.

3) Is it possible to insert only few values in the table?
Yes it is possible to insert few values ,it fills the place with 'NULL' .

4).Enumerate the possible constraints applicable to the field/column_name

NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

5) Whether constraints can be enforced to the attributes once the table/relation that is created already.

We can use alter statement to define a constraint

ALTER statement to create a **primary key**

Week 3 (RA1811029010015)

1)GRANT UPDATE ON Supplier_015 TO Ashok WITH GRANT OPTION;

2)GRANT DELETE ON Supplier_015 TO Akshat WITH GRANT OPTION;

3)REVOKE UPDATE ON Supplier_015 FROM Ashok;

4)REVOKE DELETE ON Supplier_015 FROM Akshat;

QUESTIONS IN GCR:

1) Necessary for DCL and TCL commands

Ans) DCL (Data Control Language) includes commands such as GRANT and REVOKE which mainly Deals with the rights, permissions and other controls of the database system.

TCL (Transaction Control Language) commands deal with the transaction within the database.Transactions group a set of tasks into a single execution unit.

2) Enumerate the privileges for the user

Ans) ALTER - alter the structure of the database

DELETE – removing one or more rows from a table/relation

INDEX –Creates an index on a table. Duplicate values are allowed

INSERT – inserting data into the row of a table

SELECT – used to select data from a database

UPDATE - modify or update the value of a column in a table

3) Explain the commands in DCL

Ans) DCL (data control language) includes commands such as GRANT and REVOKE which mainly deals with the rights, permission and other controls of the database system.

Examples of DCL commands:

* GRANT-gives user's access privileges to database.

* REVOKE-withdraw user's access privileges given by using the GRANT command.

4) Explain the commands in TCL

Ans) TCL commands deals with the transaction within the database.

Examples of TCL commands:

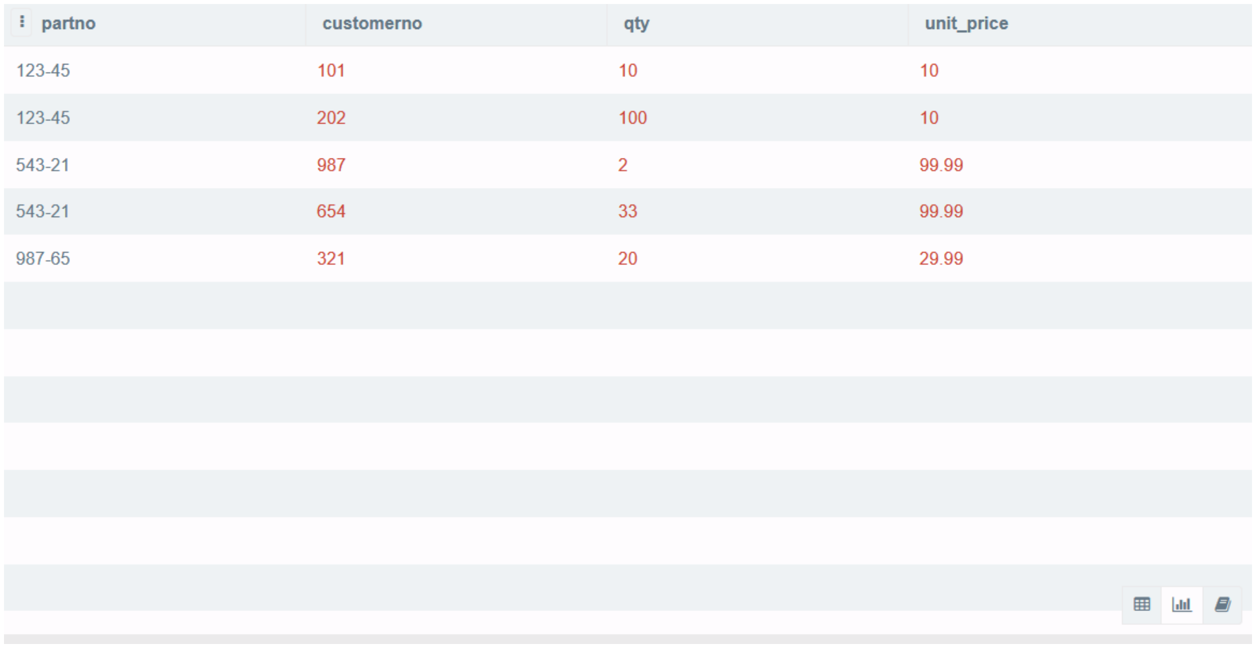
COMMIT– commits a Transaction.

ROLLBACK– rollbacks a transaction in case of any error occurs.

SAVEPOINT–sets a savepoint within a transaction.

RA1811029010015
M.karthik vikram

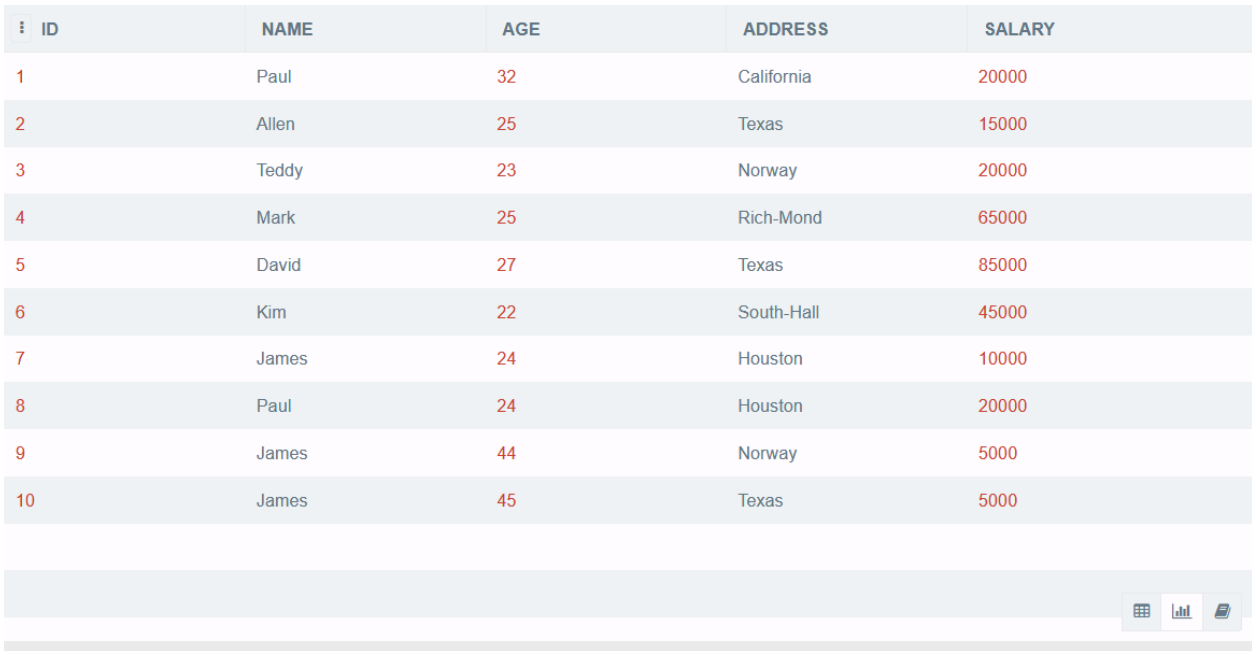
```
CREATE TABLE order_015(
partno VARCHAR(100),
customerno DOUBLE,
qty DOUBLE,
unit_price DOUBLE
);

INSERT INTO order_015(partno, customerno, qty, unit_price)
VALUES
(
'123-45', 101, 10, 10
);

INSERT INTO order_015(partno, customerno, qty, unit_price)
VALUES
(
'123-45', 202, 100, 10
);

INSERT INTO order_015(partno, customerno, qty, unit_price)
VALUES
(
'543-21', 987, 2, 99.99
);

INSERT INTO order_015(partno, customerno, qty, unit_price)
VALUES
(
'543-21', 654, 33, 99.99
);

INSERT INTO order_015(partno, customerno, qty, unit_price)
VALUES
(
'987-65', 321, 20, 29.99
);
```



partno	customerno	qty	unit_price
123-45	101	10	10
123-45	202	100	10
543-21	987	2	99.99
543-21	654	33	99.99
987-65	321	20	29.99

```
CREATE TABLE company_015(
ID DOUBLE,
NAME VARCHAR(100),
AGE DOUBLE,
ADDRESS VARCHAR(100),
SALARY DOUBLE
);
INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
1, 'Paul', 32, 'California', 20000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
2, 'Allen', 25, 'Texas', 15000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
3, 'Teddy', 23, 'Norway', 20000
);
INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
```

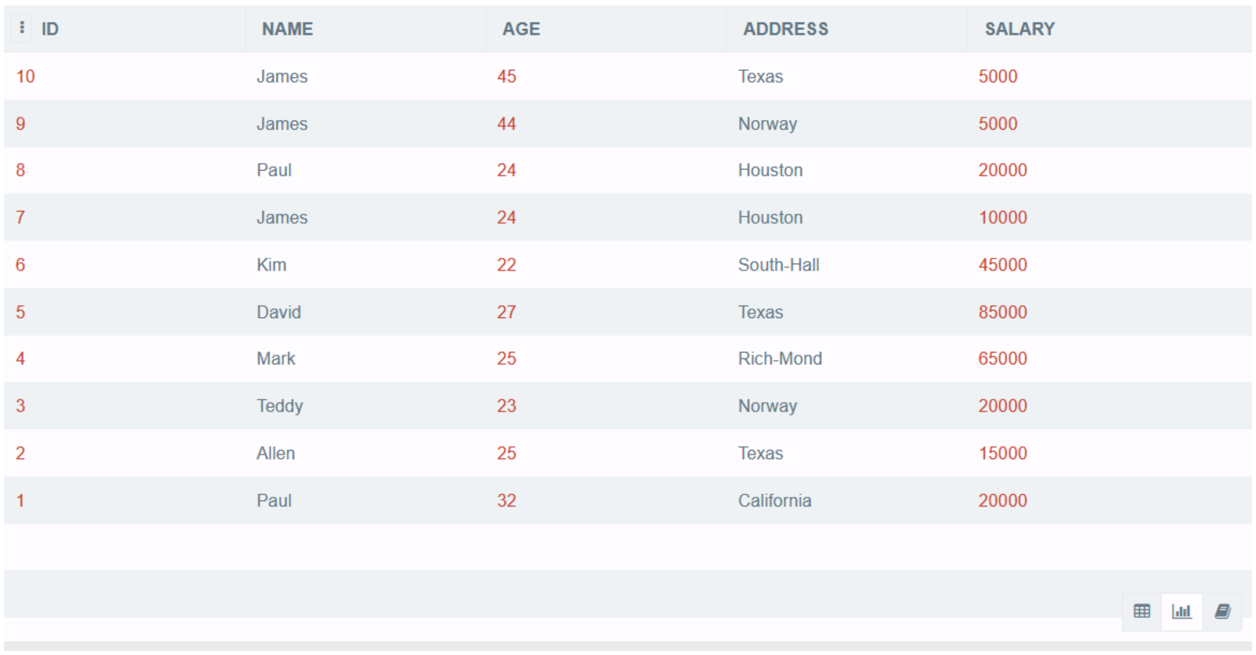
```
(
4, 'Mark', 25, 'Rich-Mond', 65000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
5, 'David', 27, 'Texas', 85000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
6, 'Kim', 22, 'South-Hall', 45000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
7, 'James', 24, 'Houston', 10000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
8, 'Paul', 24, 'Houston', 20000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
9, 'James', 44, 'Norway', 5000
);

INSERT INTO company_015(ID, NAME, AGE, ADDRESS, SALARY)
VALUES
(
10, 'James', 45, 'Texas', 5000
);
```



ID	NAME	AGE	ADDRESS	SALARY
1	Paul	32	California	20000
2	Allen	25	Texas	15000
3	Teddy	23	Norway	20000
4	Mark	25	Rich-Mond	65000
5	David	27	Texas	85000
6	Kim	22	South-Hall	45000
7	James	24	Houston	10000
8	Paul	24	Houston	20000
9	James	44	Norway	5000
10	James	45	Texas	5000

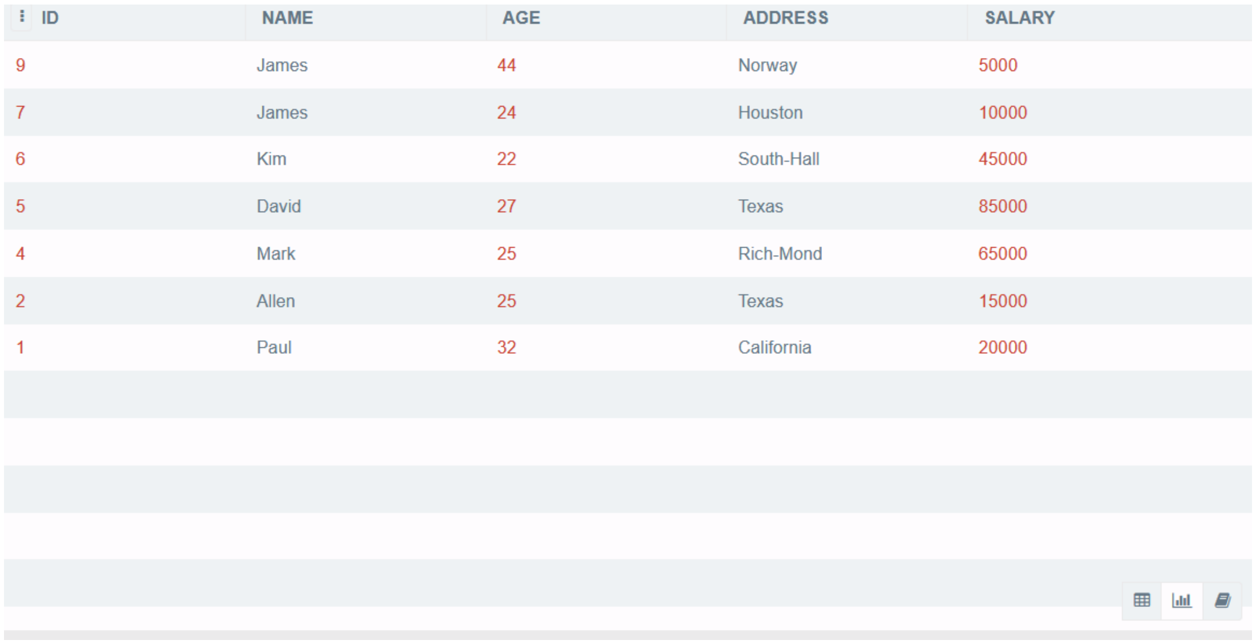
Queries

1)select * from company_015 order by id DESC;



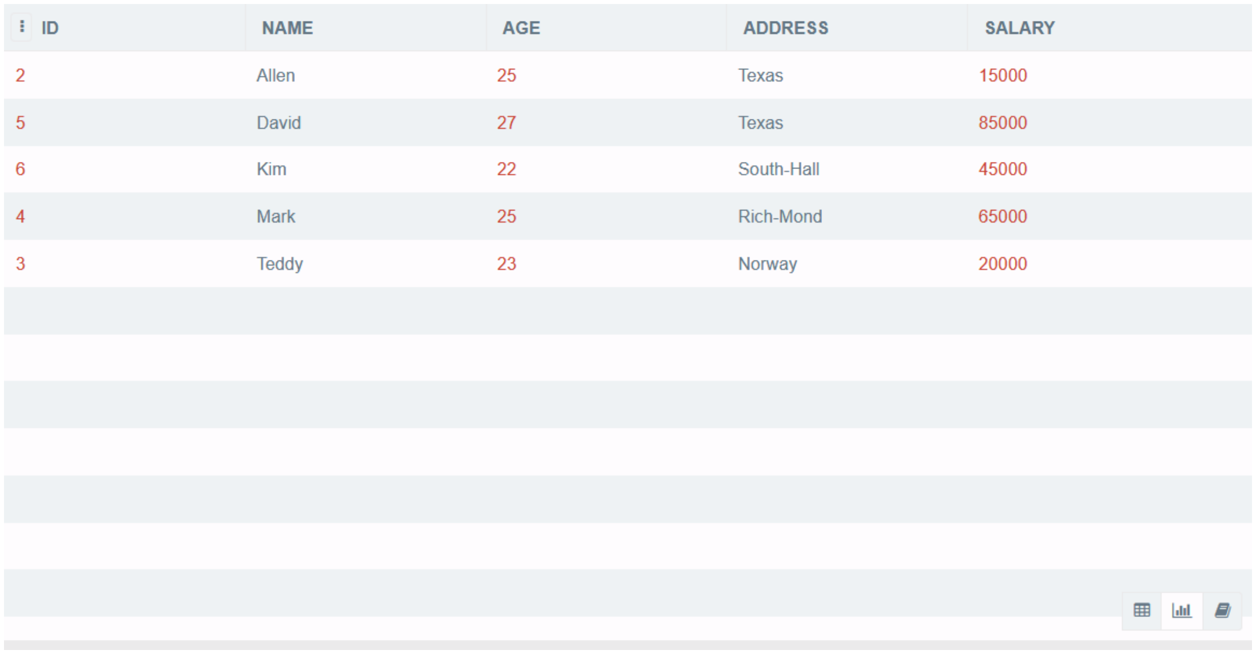
ID	NAME	AGE	ADDRESS	SALARY
10	James	45	Texas	5000
9	James	44	Norway	5000
8	Paul	24	Houston	20000
7	James	24	Houston	10000
6	Kim	22	South-Hall	45000
5	David	27	Texas	85000
4	Mark	25	Rich-Mond	65000
3	Teddy	23	Norway	20000
2	Allen	25	Texas	15000
1	Paul	32	California	20000

2)select * from company_015 Group by SALARY order by id DESC



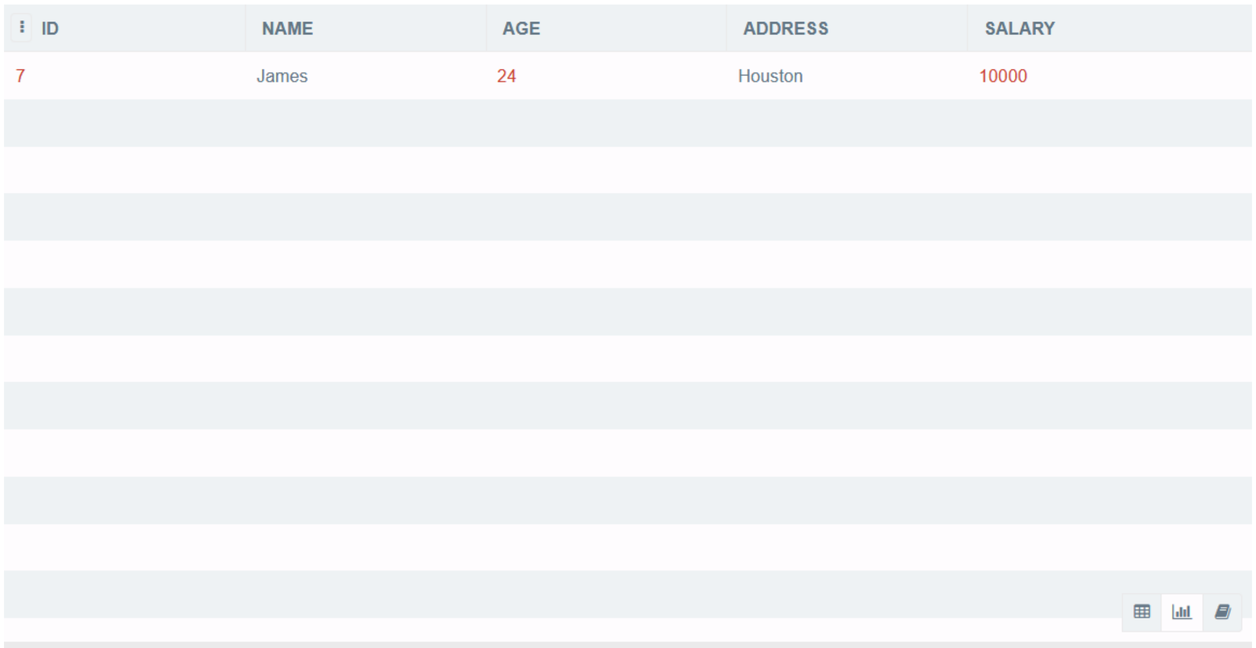
ID	NAME	AGE	ADDRESS	SALARY
9	James	44	Norway	5000
7	James	24	Houston	10000
6	Kim	22	South-Hall	45000
5	David	27	Texas	85000
4	Mark	25	Rich-Mond	65000
2	Allen	25	Texas	15000
1	Paul	32	California	20000

3)select * from company_015 Group by NAME having COUNT(*)<2;



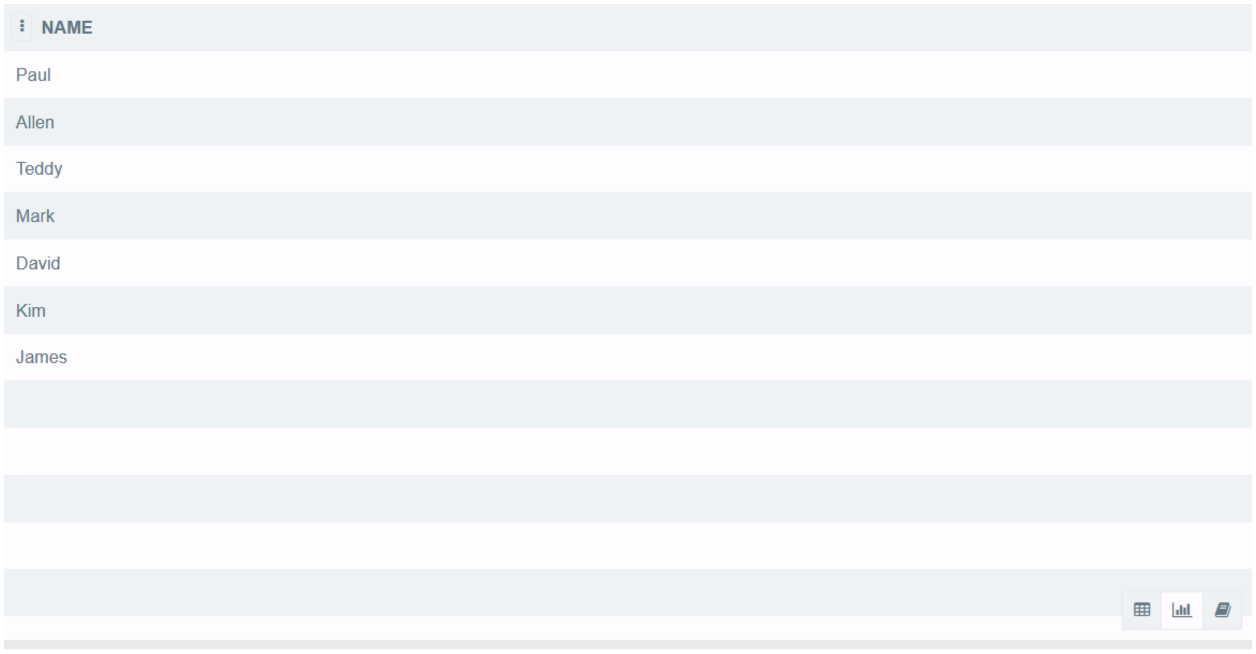
ID	NAME	AGE	ADDRESS	SALARY
2	Allen	25	Texas	15000
5	David	27	Texas	85000
6	Kim	22	South-Hall	45000
4	Mark	25	Rich-Mond	65000
3	Teddy	23	Norway	20000

4)select * from company_015 Group by NAME having COUNT(*)>2;



ID	NAME	AGE	ADDRESS	SALARY
7	James	24	Houston	10000

5)select DISTINCT name from company_015



NAME
Paul
Allen
Teddy
Mark
David
Kim
James

1. List all the built-in functions

Numberic Functions

String Functions

Number Conversion Functions

Group Functions

Date and Time Functions

Date Conversion Functions

Date Formats

2) describe the commands

ABS (m)	m = value	Absolute value of m
MOD (m, n)	m = value, n = divisor	Remainder of m divided by n
POWER (m, n)	m = value, n = exponent	m raised to the nth power
ROUND (m [, n])	m = value, n = number of decimal places, default 0	m rounded to the nth decimal place
TRUNC (m [, n])	m = value, n = number of decimal places, default 0	m truncated to the nth decimal place
SIN (n)	n = angle expressed in radians	sine (n)

COS (n)	n = angle expressed in radians	cosine (n)

SQRT (n)	n = value	positive square root of n
EXP (n)	n = value	e raised to the power n
LN (n)	n > 0	natural logarithm of n

INITCAP (s)	s = character string	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	s = character string	All letters are changed to lowercase.
UPPER (s)	s = character string	All letters are changed to uppercase.
CONCAT (s1, s2)	s1 and s2 are character strings	Concatenation of s1 and s2. Equivalent to *s1* || *s2*

AVG ([DISTINCT | ALL] col)	col = column name	The average value of that column
COUNT (*)	none	Number of rows returned including duplicates and NULLs
COUNT ([DISTINCT | ALL] col)	col = column name	Number of rows where the value of the column is not NULL

MAX ([DISTINCT | ALL] col)	col = column name	Maximum value in the column
MIN ([DISTINCT | ALL] col)	col = column name	Minimum value in the column
SUM ([DISTINCT | ALL] col)	col = column name	Sum of the values in the column

3)What are the commands that can be executed in the online editor

1 **SQLite COUNT Function**

COUNT aggregate function is used to count the number of rows in a database table.

2 **MAX Function**

MAX aggregate function allows us to select the highest (maximum) value for a certain column.

3 **MIN Function**

MIN aggregate function allows us to select the lowest (minimum) value for a certain column.

4 **AVG Function**

AVG aggregate function selects the average value for certain table column.

5 **SUM Function**

SUM aggregate function allows selecting the total for a numeric column.

6 **RANDOM Function**

RANDOM function returns a pseudo-random integer between -9223372036854775808 and +9223372036854775807.

7 **ABS Function**

ABS function returns the absolute value of the numeric argument.

8 **UPPER Function**

UPPER function converts a string into upper-case letters.

9 **LOWER Function**

LOWER function converts a string into lower-case letters.

10 **LENGTH Function**

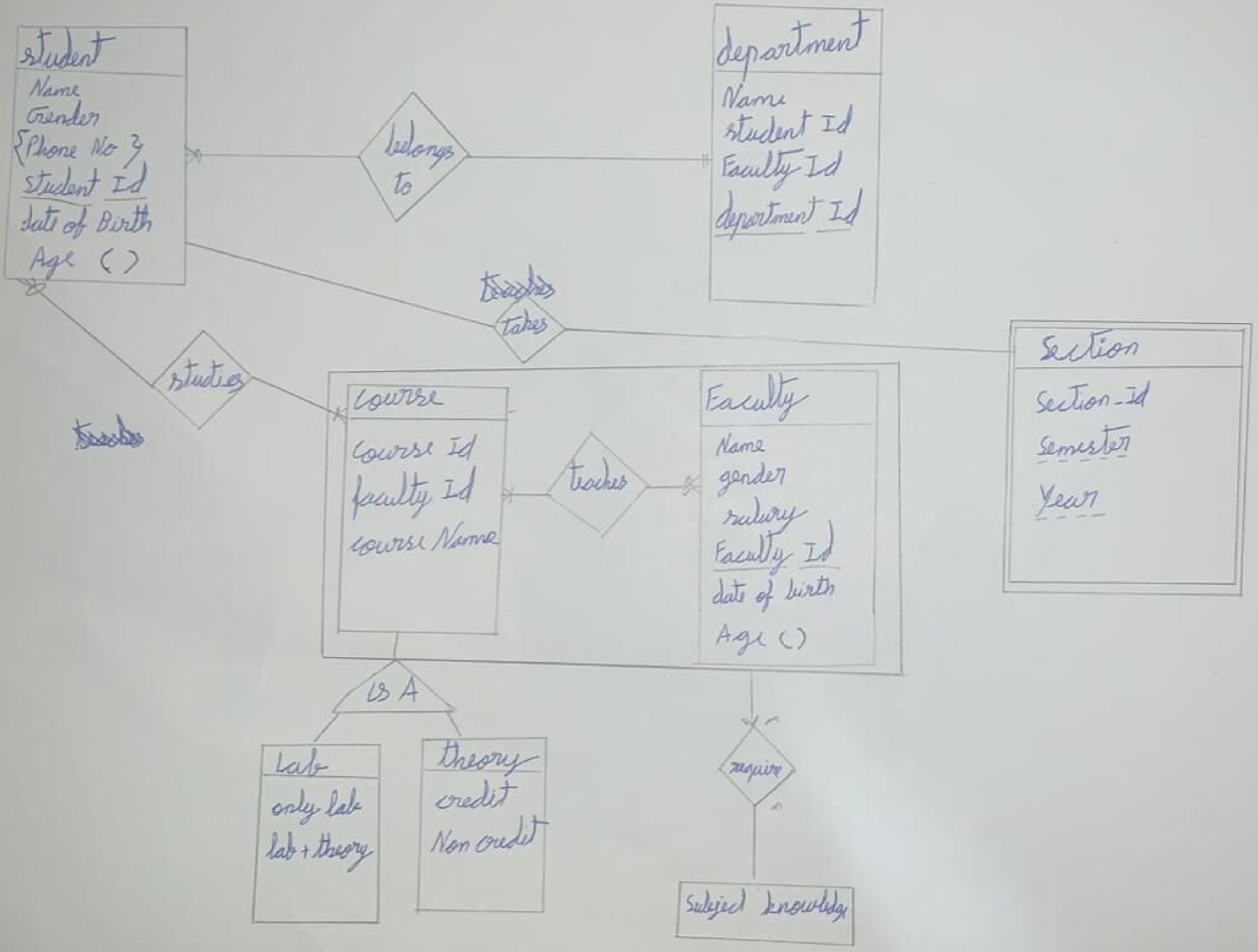
LENGTH function returns the length of a string.

Date and time functions

11) date(time string,modifiers)

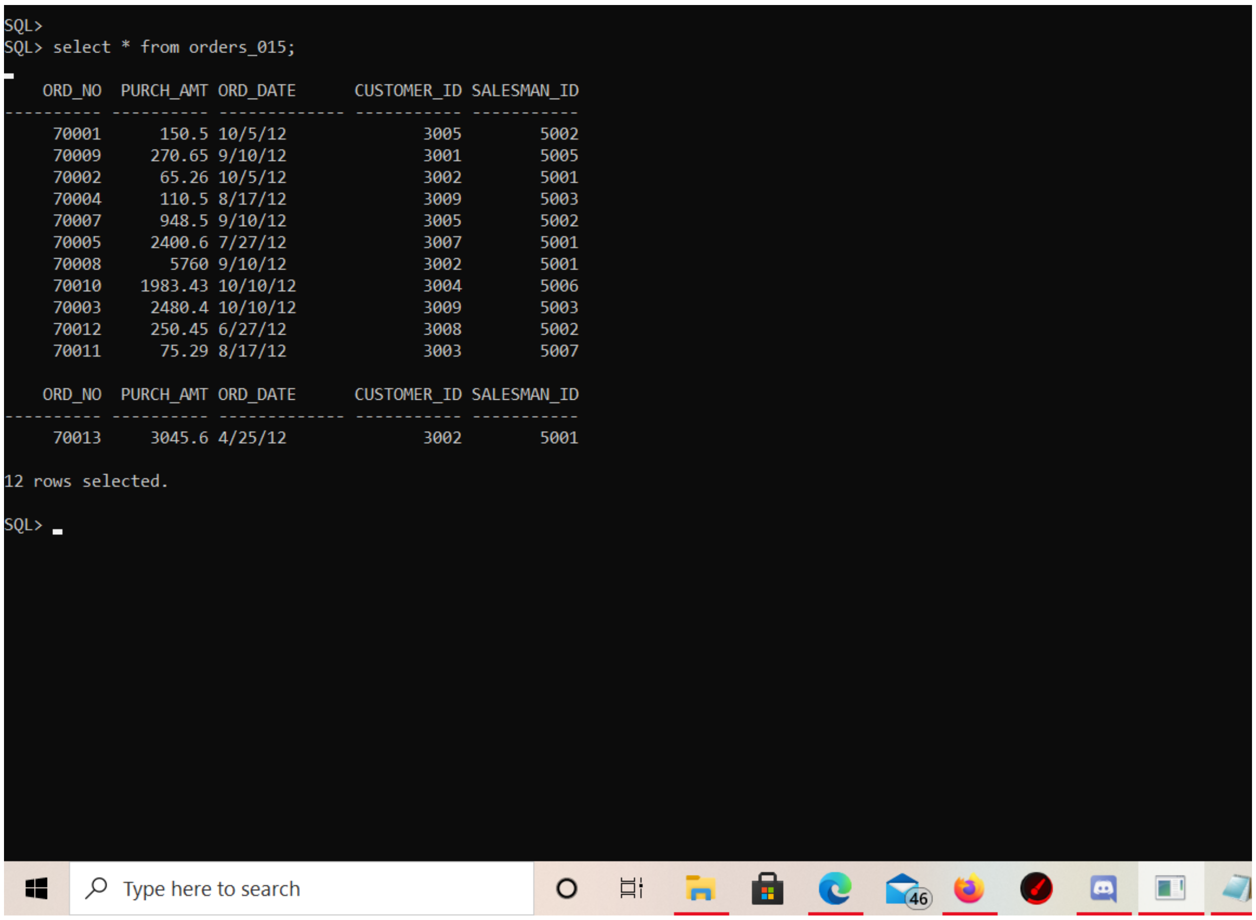
12)time(time string,modifiers)

13)datetime(time string,modifiers)


student
Name
Grender
{Phone No }
student Id
date of Birth
Age ( )
belongs to
department
Name
student Id
Faculty Id
department Id
~~teaches~~
takes
Section
Section_Id
Semester
Year
studies
~~teaches~~
course
course Id
faculty Id
course Name
teaches
Faculty
Name
gender
salary
Faculty Id
date of birth
Age ( )
IS A
Lab
only lab
lab + theory
theory
credit
Non credit
require
Subject knowledge

Week 7

Table orders



```
SQL>
SQL> select * from orders_015;

    ORD_NO  PURCH_AMT ORD_DATE      CUSTOMER_ID SALESMAN_ID
---------- ---------- ------------- ----------- -----------
     70001      150.5 10/5/12              3005        5002
     70009     270.65 9/10/12              3001        5005
     70002      65.26 10/5/12              3002        5001
     70004      110.5 8/17/12              3009        5003
     70007      948.5 9/10/12              3005        5002
     70005     2400.6 7/27/12              3007        5001
     70008       5760 9/10/12              3002        5001
     70010    1983.43 10/10/12             3004        5006
     70003     2480.4 10/10/12             3009        5003
     70012     250.45 6/27/12              3008        5002
     70011      75.29 8/17/12              3003        5007

    ORD_NO  PURCH_AMT ORD_DATE      CUSTOMER_ID SALESMAN_ID
---------- ---------- ------------- ----------- -----------
     70013     3045.6 4/25/12              3002        5001

12 rows selected.

SQL>
```

Table grade

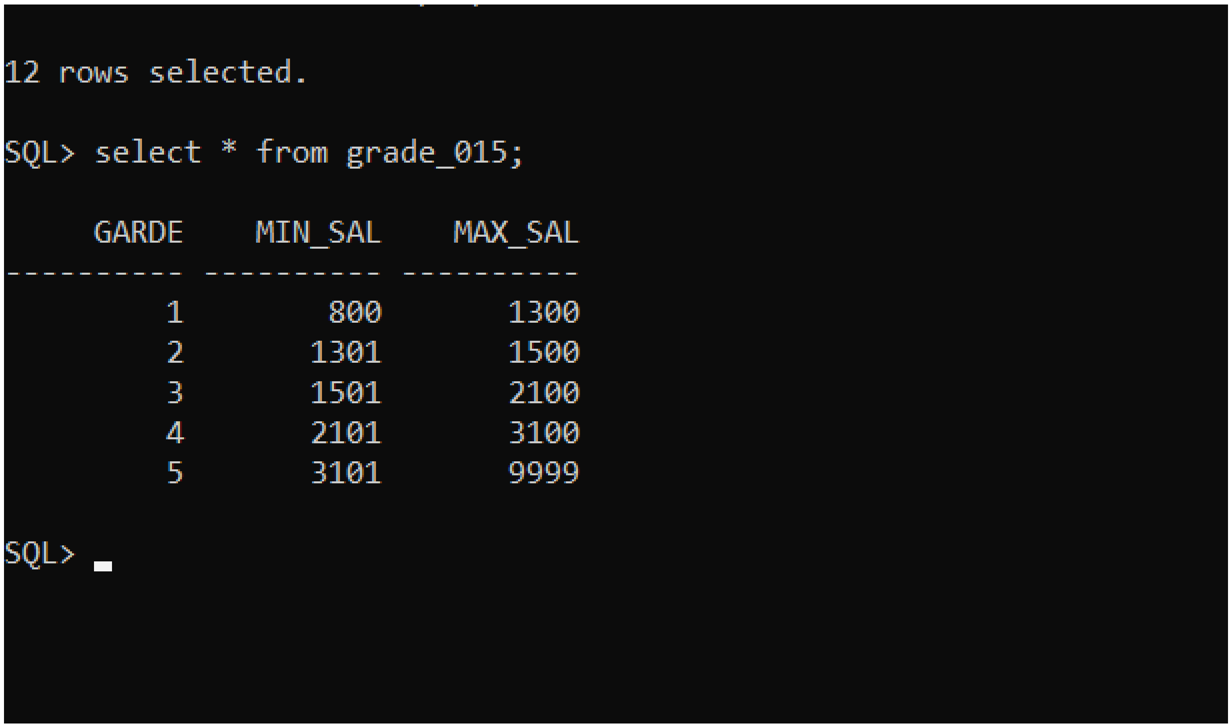

```
12 rows selected.

SQL> select * from grade_015;

     GARDE    MIN_SAL    MAX_SAL
---------- ---------- ----------
         1        800       1300
         2       1301       1500
         3       1501       2100
         4       2101       3100
         5       3101       9999

SQL> _
```

Table department

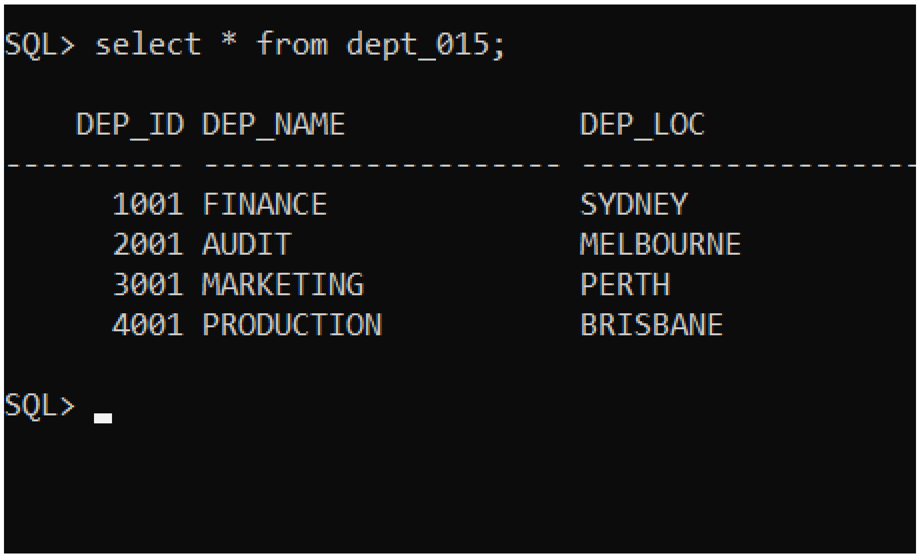
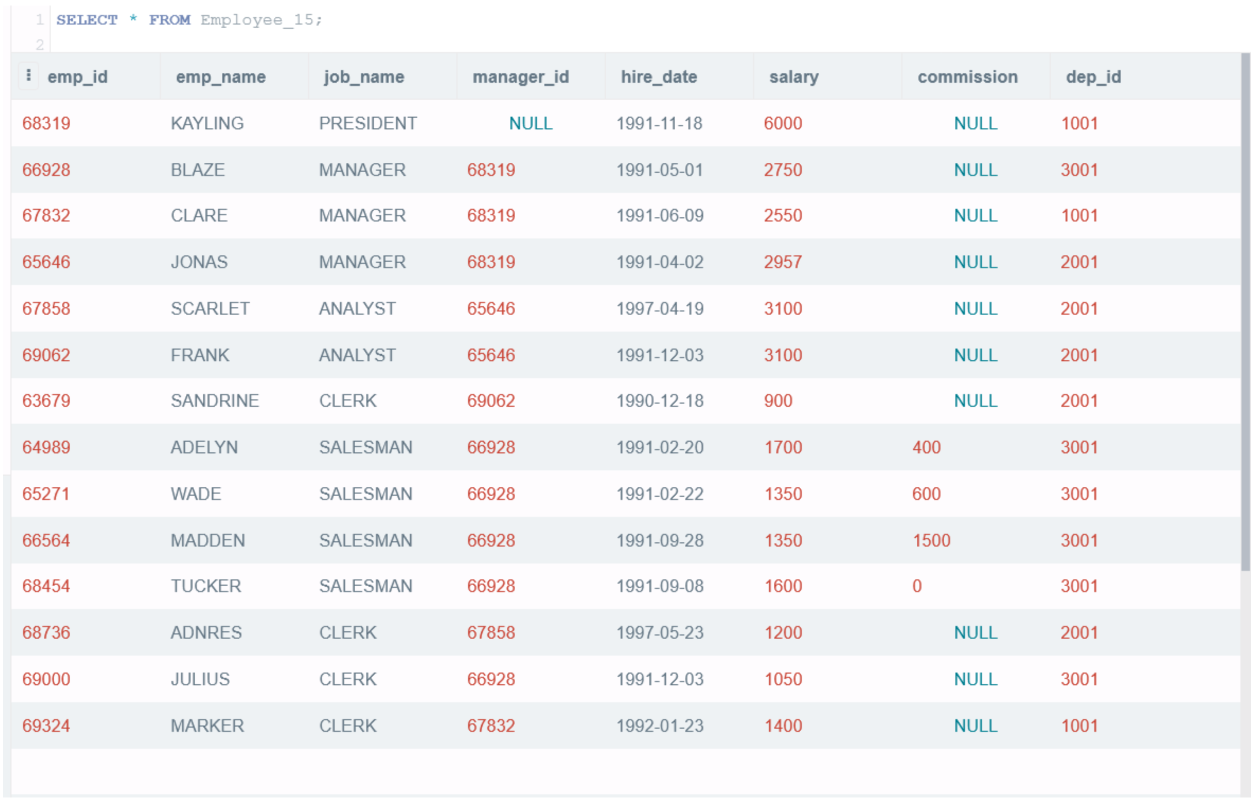

```
SQL> select * from dept_015;

    DEP_ID DEP_NAME             DEP_LOC
---------- -------------------- -------------------
      1001 FINANCE              SYDNEY
      2001 AUDIT                MELBOURNE
      3001 MARKETING            PERTH
      4001 PRODUCTION           BRISBANE

SQL> _
```

Table employee



```
SELECT * FROM Employee_15;
```

emp_id	emp_name	job_name	manager_id	hire_date	salary	commission	dep_id
68319	KAYLING	PRESIDENT	NULL	1991-11-18	6000	NULL	1001
66928	BLAZE	MANAGER	68319	1991-05-01	2750	NULL	3001
67832	CLARE	MANAGER	68319	1991-06-09	2550	NULL	1001
65646	JONAS	MANAGER	68319	1991-04-02	2957	NULL	2001
67858	SCARLET	ANALYST	65646	1997-04-19	3100	NULL	2001
69062	FRANK	ANALYST	65646	1991-12-03	3100	NULL	2001
63679	SANDRINE	CLERK	69062	1990-12-18	900	NULL	2001
64989	ADELYN	SALESMAN	66928	1991-02-20	1700	400	3001
65271	WADE	SALESMAN	66928	1991-02-22	1350	600	3001
66564	MADDEN	SALESMAN	66928	1991-09-28	1350	1500	3001
68454	TUCKER	SALESMAN	66928	1991-09-08	1600	0	3001
68736	ADNRES	CLERK	67858	1997-05-23	1200	NULL	2001
69000	JULIUS	CLERK	66928	1991-12-03	1050	NULL	3001
69324	MARKER	CLERK	67832	1992-01-23	1400	NULL	1001

Week 7 queries

```
1)SELECT e.emp_id,e.emp_name,e.salary,d.dep_name FROM employee_015 e,dept_015 d
WHERE d.dep_location IN ('SYDNEY','PERTH')  AND e.dep_id = d.dep_id  AND e.emp_id IN
   (SELECT e.emp_id FROM employee_015 e
    WHERE e.job_name IN ('MANAGER', 'ANALYST')
      AND (DATE_PART('year', CURRENT_DATE)-DATE_PART('year', hire_date))> 5
      AND e.commission IS NULL)
      ORDER BY d.dep_location ASC;
```

```
2)SELECT E.emp_id,
      E.emp_name,
      E.salary,
      D.dep_name,
      D.dep_location,
      E.dep_id,
      E.job_name
FROM employee_015 E,
    dept_015 D
WHERE (D.dep_location = 'SYDNEY'
      OR D.dep_name = 'FINANCE')
  AND E.dep_id=D.dep_id
  AND E.emp_id IN
    (SELECT emp_id
    FROM employee_015 E
    WHERE (12*E.salary) > 28000
      AND E.salary NOT IN (3000,
                           2800)
      AND E.job_name !='MANAGER'
      AND (trim(to_char(emp_id,'99999')) LIKE '__3%'
           OR trim(to_char(emp_id,'99999')) LIKE '__7%'))
ORDER BY E.dep_id ASC,
         E.job_name DESC;
```

```
3)FROM employee_015 e,salary_grade_015 s
WHERE e.salary BETWEEN s.min_sal AND s.max_sal
  AND s.grade IN (4,5) AND e.emp_id IN
 (SELECT e.emp_id  FROM employee_015 e  WHERE e.job_name IN ('MANAGER',
'ANALYST'));
```

```
4)SELECT department_name AS 'Department Name',
COUNT(*) AS 'No of Employees'
FROM dept_015
INNER JOIN employee_015
ON employees.dept_id = dept_015.dept_id
GROUP BY dept_015.dept_id, dept_015_name
ORDER BY dep_name;
```

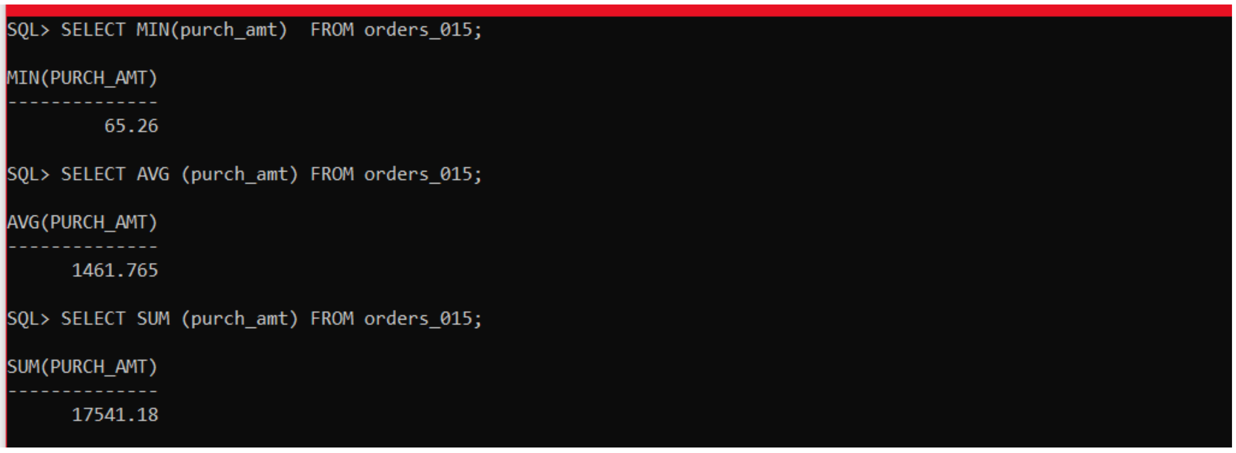
```
5)SELECT customer_id,MAX(purch_amt) FROM orders_015 GROUP BY customer_id;
```

```
6)SELECT COUNT(*) FROM order_015  WHERE ord_date='2012-08-17';
```

```
7)SELECT MIN(purch_amt)  FROM orders_015;
```

```
8)SELECT AVG (purch_amt) FROM orders_015;
```

```
9)SELECT SUM (purch_amt) FROM orders_015;
```

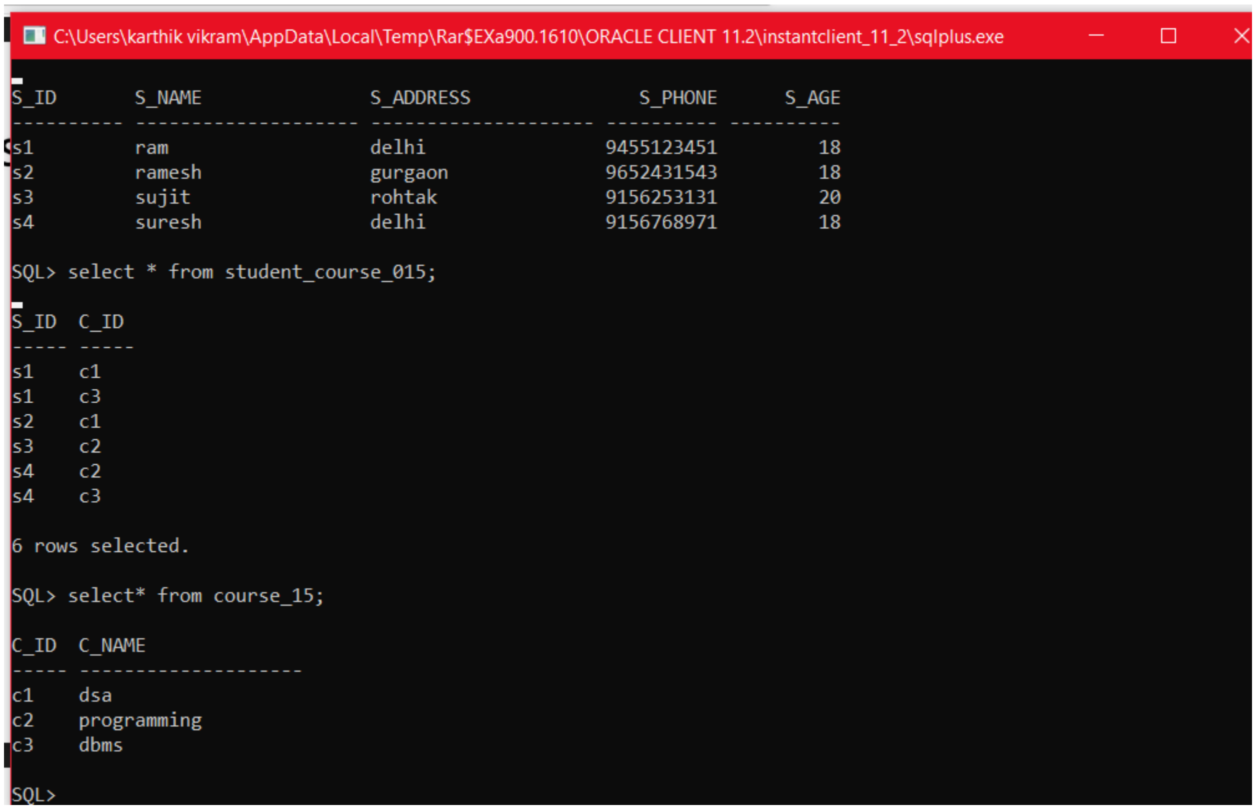


```
SQL> SELECT MIN(purch_amt)  FROM orders_015;

MIN(PURCH_AMT)
--------------
         65.26

SQL> SELECT AVG (purch_amt) FROM orders_015;

AVG(PURCH_AMT)
--------------
      1461.765

SQL> SELECT SUM (purch_amt) FROM orders_015;

SUM(PURCH_AMT)
--------------
      17541.18
```

```
s3
Enter value for s_name: sujit
Enter value for s_address: rohtak
Enter value for s_phone: 9156253131
Enter value for s_age: 20
old   1:  insert into student_015 values ('&s_id','&s_name','&s_address','&s_phone','&s_age')
new   1:  insert into student_015 values ('s3','sujit','rohtak','9156253131','20')

1 row created.

SQL>  insert into student_015 values ('&s_id','&s_name','&s_address','&s_phone','&s_age');
Enter value for s_id: s4
Enter value for s_name: suresh
Enter value for s_address: delhi
Enter value for s_phone: 9156768971
Enter value for s_age: 18
old   1:  insert into student_015 values ('&s_id','&s_name','&s_address','&s_phone','&s_age')
new   1:  insert into student_015 values ('s4','suresh','delhi','9156768971','18')

1 row created.

SQL> select* from student_015;

S_ID    S_NAME               S_ADDRESS            S_PHONE      S_AGE
---------- -------------------- -------------------- ---------- ----------
s1      ram             delhi           9455123451          18
s2      ramesh          gurgaon         9652431543          18
s3      sujit           rohtak          9156253131          20
s4      suresh          delhi           9156768971          18

SQL> create table course_015 (c_id int,c_name varchar(20));

Table created.

SQL> insert into course_015 values(c1,dsa);
insert into course_015 values(c1,dsa)
                               *
ERROR at line 1:
ORA-00984: column not allowed here


SQL> insert into course_015 values('&c_id','&c-name');
Enter value for c_id: c1
Enter value for c: dsa
```

```
old   1: insert into course_015 values('&c_id','&c-name')
new   1: insert into course_015 values('c1','dsa-name')
insert into course_015 values('c1','dsa-name')
                              *
ERROR at line 1:
ORA-01722: invalid number


SQL>  insert into course_015 values('&c_id','&c_name');
Enter value for c_id: c1
Enter value for c_name: dsa
old   1:  insert into course_015 values('&c_id','&c_name')
new   1:  insert into course_015 values('c1','dsa')
 insert into course_015 values('c1','dsa')
                              *
ERROR at line 1:
ORA-01722: invalid number


SQL> create table course_15 (c_id varchar(5),c_name varchar(20));

Table created.

SQL>  insert into course_015 values('&c_id','&c_name');
Enter value for c_id: c1
Enter value for c_name: dsa
old   1:  insert into course_015 values('&c_id','&c_name')
new   1:  insert into course_015 values('c1','dsa')
 insert into course_015 values('c1','dsa')
                              *
ERROR at line 1:
ORA-01722: invalid number


SQL>  insert into course_15 values('&c_id','&c_name');
Enter value for c_id: c1
Enter value for c_name: dsa
old   1:  insert into course_15 values('&c_id','&c_name')
new   1:  insert into course_15 values('c1','dsa')

1 row created.

SQL>  insert into course_15 values('&c_id','&c_name');
Enter value for c_id: c2
Enter value for c_name: programming
old   1:  insert into course_15 values('&c_id','&c_name')
new   1:  insert into course_15 values('c2','programming')

1 row created.

SQL>  insert into course_15 values('&c_id','&c_name');
Enter value for c_id: c3
Enter value for c_name: dbms
old   1:  insert into course_15 values('&c_id','&c_name')
new   1:  insert into course_15 values('c3','dbms')

1 row created.

SQL> create table student_course_015(s_id varchar(5),c_id varchar (5));

Table created.

SQL> insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s1
Enter value for c_id: c1
old   1: insert into student_course_015 values( '&s_id','&c_id')
new   1: insert into student_course_015 values( 's1','c1')

1 row created.

SQL> create table student_course_015(s_id varchar(5),c_id varchar (5));
create table student_course_015(s_id varchar(5),c_id varchar (5))
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL>  insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s1
Enter value for c_id: c3
old   1:  insert into student_course_015 values( '&s_id','&c_id')
new   1:  insert into student_course_015 values( 's1','c3')

1 row created.

SQL>  insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s2
Enter value for c_id: c1
```

```
old   1:  insert into student_course_015 values( '&s_id','&c_id')
new   1:  insert into student_course_015 values( 's2','c1')

1 row created.

SQL>  insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s3
Enter value for c_id: c2
old   1:  insert into student_course_015 values( '&s_id','&c_id')
new   1:  insert into student_course_015 values( 's3','c2')

1 row created.

SQL>  insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s4
Enter value for c_id: c2
old   1:  insert into student_course_015 values( '&s_id','&c_id')
new   1:  insert into student_course_015 values( 's4','c2')

1 row created.

SQL>  insert into student_course_015 values( '&s_id','&c_id');
Enter value for s_id: s4
Enter value for c_id: c3
old   1:  insert into student_course_015 values( '&s_id','&c_id')
new   1:  insert into student_course_015 values( 's4','c3')

1 row created.
```

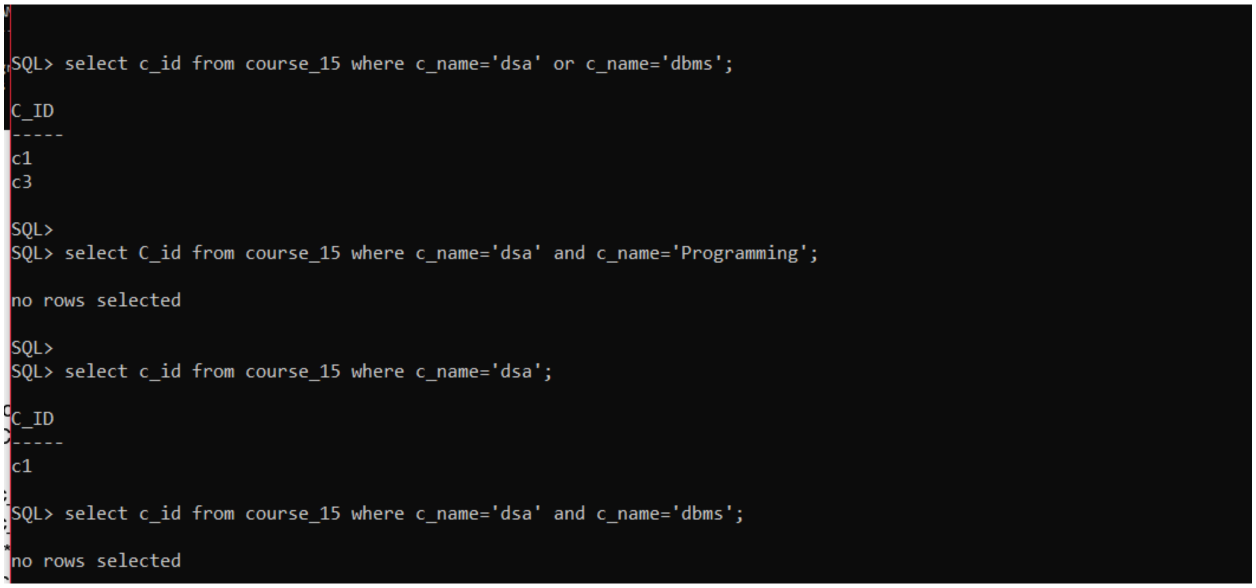


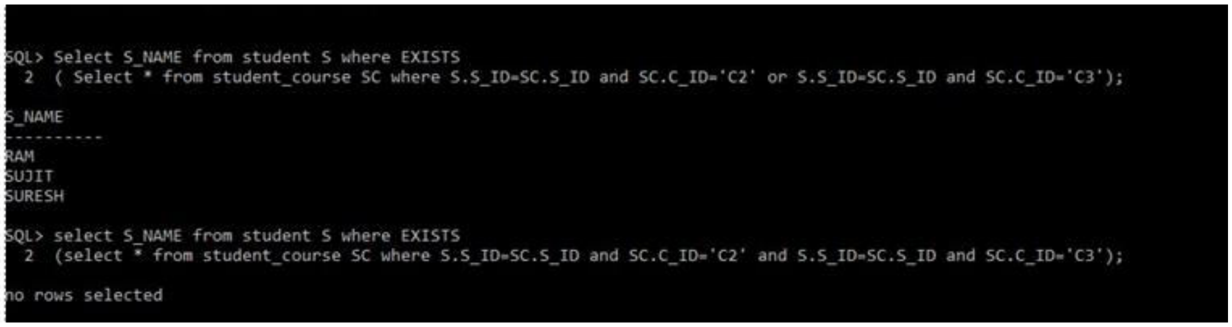
```
C:\Users\karthik vikram\AppData\Local\Temp\Rar$EXa900.1610\ORACLE CLIENT 11.2\instantclient_11_2\sqlplus.exe

S_ID       S_NAME               S_ADDRESS               S_PHONE      S_AGE
---------- -------------------- -------------------- ---------- ----------
s1         ram                  delhi                9455123451         18
s2         ramesh               gurgaon              9652431543         18
s3         sujit                rohtak               9156253131         20
s4         suresh               delhi                9156768971         18

SQL> select * from student_course_015;

S_ID  C_ID
----- -----
s1    c1
s1    c3
s2    c1
s3    c2
s4    c2
s4    c3

6 rows selected.

SQL> select* from course_15;

C_ID  C_NAME
----- --------------------
c1    dsa
c2    programming
c3    dbms

SQL>
```

Queries

1) select c_id from course_15 where c_name='dsa' or c_name='dbms';
2)select C_id from course_15 where c_name='dsa' and c_name='Programming';

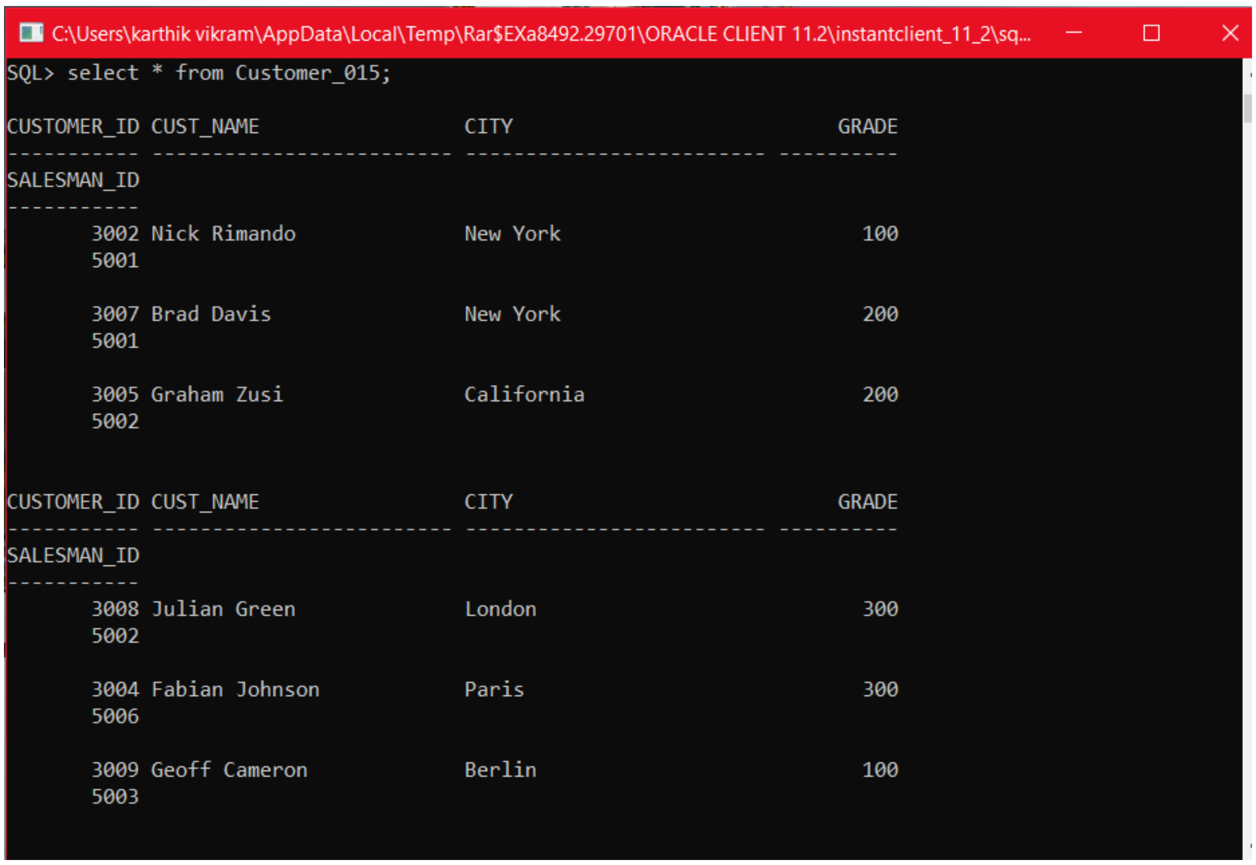
3)select c_id from course_15 where c_name='dsa';
4)select c_id from course_15 where c_name='dsa' and c_name='dbms';
5)(select * from student_015 sc where s.s_id=sc.c_id and sc.cid='c2' or s.s_id=sc.c_id and sc.c_id='c3');
6)(select * from student_015 sc where s.s_id=sc.c_id and sc.cid='c2' and s.s_id=sc.c_id and sc.c_id='c3');


```
SQL> select c_id from course_15 where c_name='dsa' or c_name='dbms';

C_ID
-----
c1
c3

SQL>
SQL> select C_id from course_15 where c_name='dsa' and c_name='Programming';

no rows selected

SQL>
SQL> select c_id from course_15 where c_name='dsa';

C_ID
-----
c1

SQL> select c_id from course_15 where c_name='dsa' and c_name='dbms';

no rows selected
```


```
SQL> Select S_NAME from student S where EXISTS
  2  ( Select * from student_course SC where S.S_ID=SC.S_ID and SC.C_ID='C2' or S.S_ID=SC.S_ID and SC.C_ID='C3');

S_NAME
----------
RAM
SUJIT
SURESH

SQL> select S_NAME from student S where EXISTS
  2  (select * from student_course SC where S.S_ID=SC.S_ID and SC.C_ID='C2' and S.S_ID=SC.S_ID and SC.C_ID='C3');

no rows selected
```

Week 8

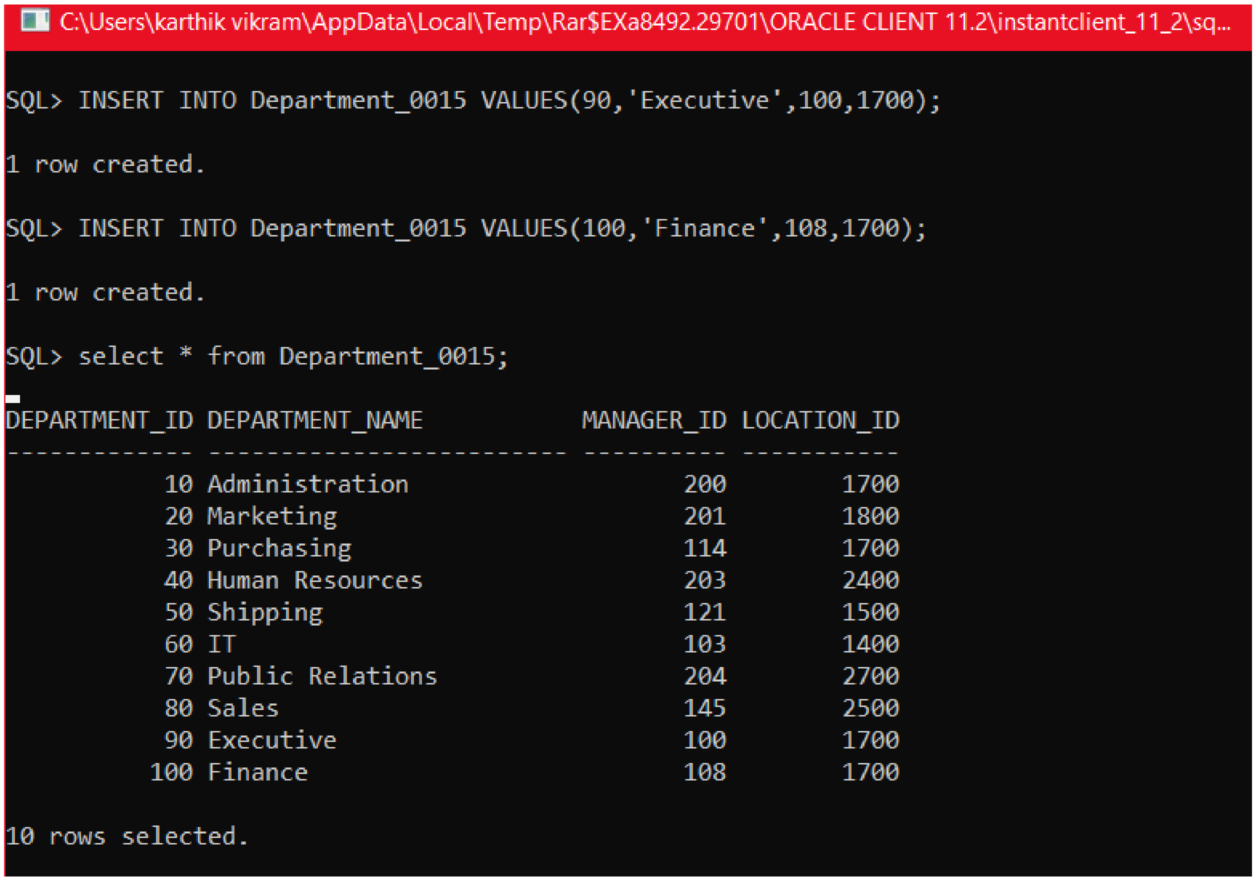
Table customer



C:\Users\karthik vikram\AppData\Local\Temp\Rar$EXa8492.29701\ORACLE CLIENT 11.2\instantclient_11_2\sq...

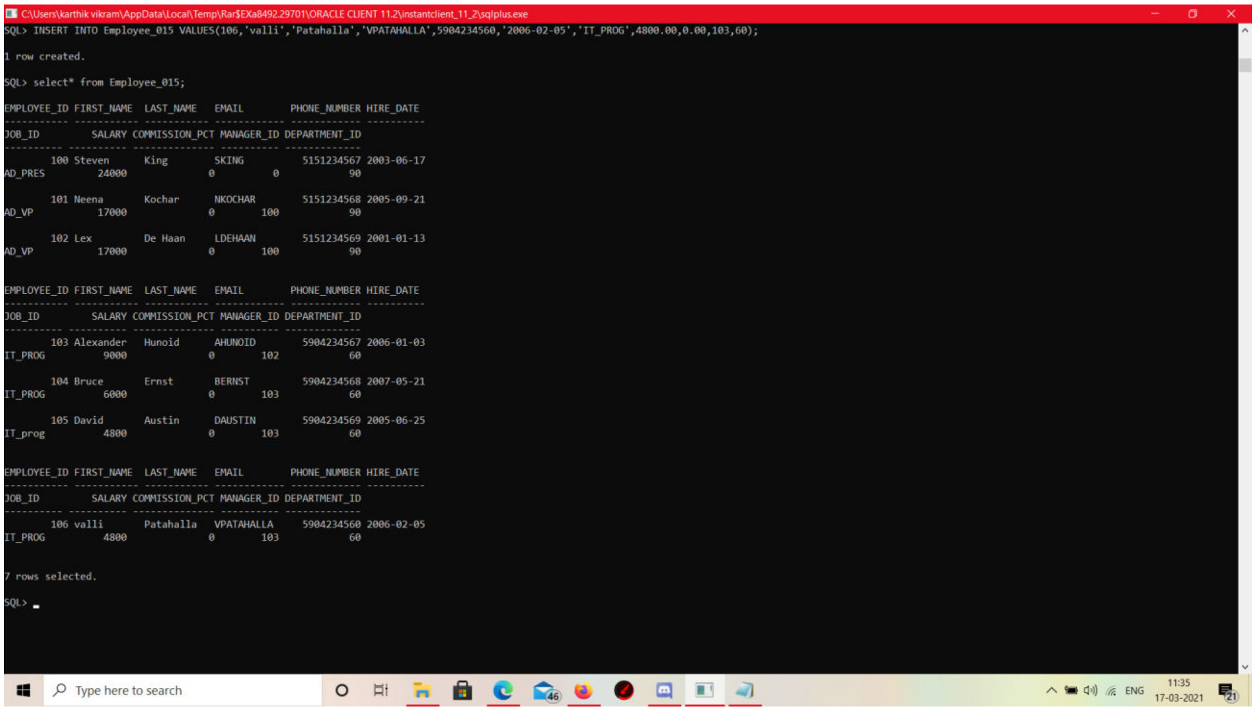
```
SQL> select * from Customer_015;

CUSTOMER_ID CUST_NAME                 CITY                           GRADE
----------- ------------------------- ------------------------- ----------
SALESMAN_ID
-----------
       3002 Nick Rimando              New York                         100
       5001

       3007 Brad Davis                New York                         200
       5001

       3005 Graham Zusi               California                       200
       5002


CUSTOMER_ID CUST_NAME                 CITY                           GRADE
----------- ------------------------- ------------------------- ----------
SALESMAN_ID
-----------
       3008 Julian Green              London                           300
       5002

       3004 Fabian Johnson            Paris                            300
       5006

       3009 Geoff Cameron             Berlin                           100
       5003
```

Table department


C:\Users\karthik vikram\AppData\Local\Temp\Rar$EXa8492.29701\ORACLE CLIENT 11.2\instantclient_11_2\sq...

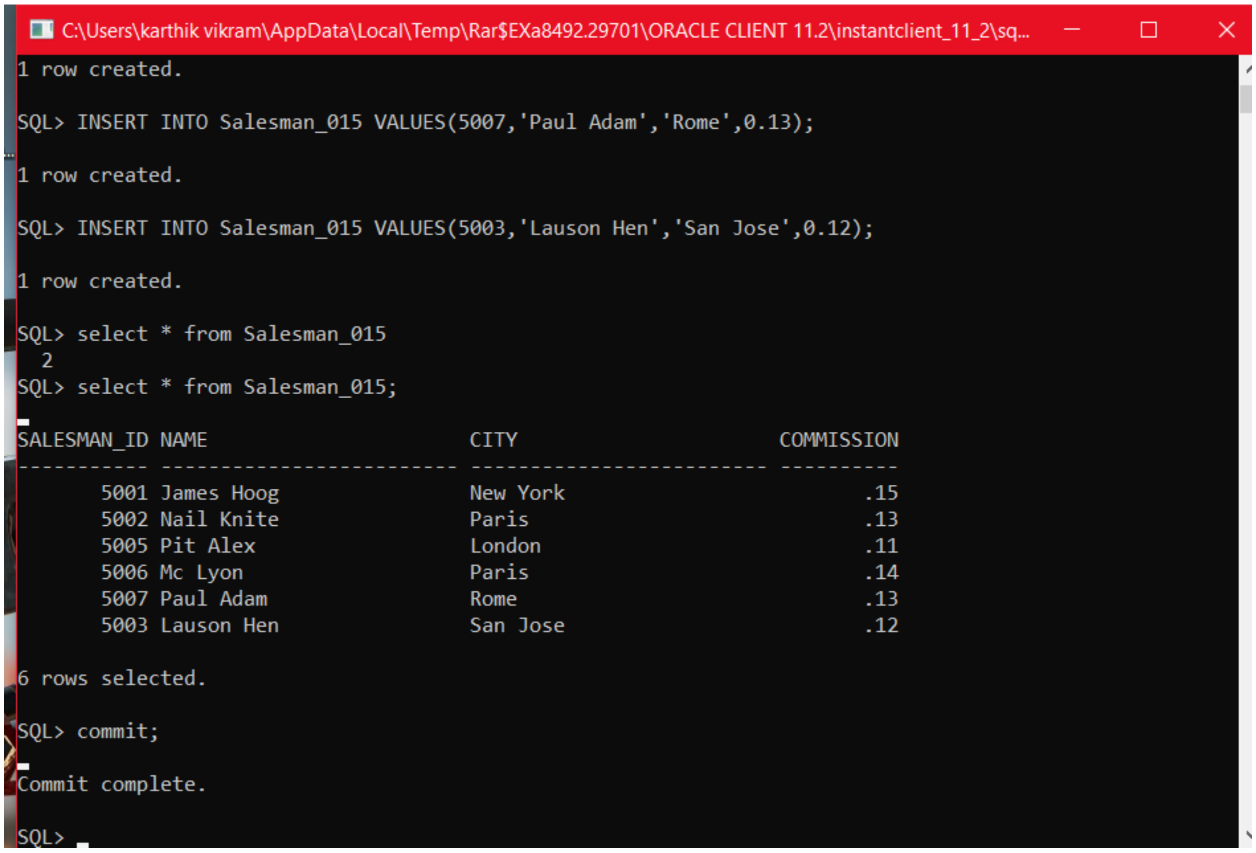
```
SQL> INSERT INTO Department_0015 VALUES(90,'Executive',100,1700);

1 row created.

SQL> INSERT INTO Department_0015 VALUES(100,'Finance',108,1700);

1 row created.

SQL> select * from Department_0015;

DEPARTMENT_ID DEPARTMENT_NAME           MANAGER_ID LOCATION_ID
------------- ------------------------- ---------- -----------
           10 Administration                   200        1700
           20 Marketing                        201        1800
           30 Purchasing                       114        1700
           40 Human Resources                  203        2400
           50 Shipping                         121        1500
           60 IT                               103        1400
           70 Public Relations                 204        2700
           80 Sales                            145        2500
           90 Executive                        100        1700
          100 Finance                          108        1700

10 rows selected.
```

Table employee



```
SQL> INSERT INTO Employee_015 VALUES(106,'valli','Patahalla','VPATAHALLA',5904234560,'2006-02-05','IT_PROG',4800.00,0.00,103,60);

1 row created.

SQL> select* from Employee_015;

EMPLOYEE_ID FIRST_NAME  LAST_NAME   EMAIL        PHONE_NUMBER HIRE_DATE
----------- ---------- ---------- ----------- ------------ ---------
JOB_ID         SALARY COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
---------- --------- -------------- ---------- -------------
        100 Steven      King        SKING          5151234567 2003-06-17
AD_PRES         24000              0          0            90

        101 Neena       Kochar      NKOCHAR        5151234568 2005-09-21
AD_VP           17000              0        100            90

        102 Lex         De Haan     LDEHAAN        5151234569 2001-01-13
AD_VP           17000              0        100            90


EMPLOYEE_ID FIRST_NAME  LAST_NAME   EMAIL        PHONE_NUMBER HIRE_DATE
----------- ---------- ---------- ----------- ------------ ---------
JOB_ID         SALARY COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
---------- --------- -------------- ---------- -------------
        103 Alexander   Hunoid      AHUNOID        5904234567 2006-01-03
IT_PROG          9000              0        102            60

        104 Bruce       Ernst       BERNST         5904234568 2007-05-21
IT_PROG          6000              0        103            60

        105 David       Austin      DAUSTIN        5904234569 2005-06-25
IT_prog          4800              0        103            60


EMPLOYEE_ID FIRST_NAME  LAST_NAME   EMAIL        PHONE_NUMBER HIRE_DATE
----------- ---------- ---------- ----------- ------------ ---------
JOB_ID         SALARY COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
---------- --------- -------------- ---------- -------------
        106 valli       Patahalla   VPATAHALLA     5904234560 2006-02-05
IT_PROG          4800              0        103            60


7 rows selected.

SQL>
```

Table salesman



```
1 row created.

SQL> INSERT INTO Salesman_015 VALUES(5007,'Paul Adam','Rome',0.13);

1 row created.

SQL> INSERT INTO Salesman_015 VALUES(5003,'Lauson Hen','San Jose',0.12);

1 row created.

SQL> select * from Salesman_015
  2
SQL> select * from Salesman_015;

SALESMAN_ID NAME                           CITY                       COMMISSION
----------- ------------------------------ ------------------------- ----------
       5001 James Hoog                     New York                         .15
       5002 Nail Knite                     Paris                            .13
       5005 Pit Alex                       London                           .11
       5006 Mc Lyon                        Paris                            .14
       5007 Paul Adam                      Rome                             .13
       5003 Lauson Hen                     San Jose                         .12

6 rows selected.

SQL> commit;

Commit complete.

SQL>
```

queries

1)

```
SELECT salesman_id , name FROM salesman_015 WHERE city='New York' UNION (SELECT
customer_id , cust_name FROM customer_015 WHERE city='New York');
```

2)

```
SELECT salesman_id, city FROM customer_015 UNION (SELECT salesman_id, city FROM
salesman_015);
```

3)

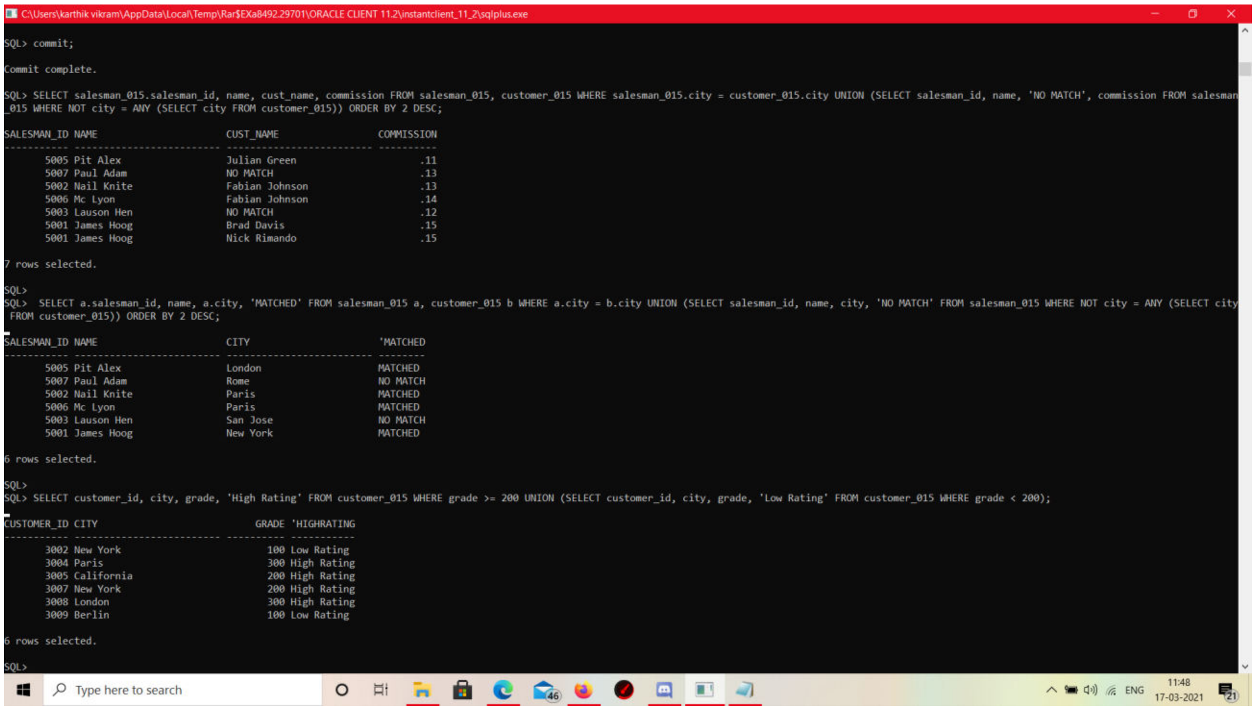
```
SELECT salesman_015.salesman_id, name, cust_name, commission FROM salesman_015,
customer_015 WHERE salesman_015.city = customer_015.city UNION (SELECT salesman_id,
name, 'NO MATCH', commission FROM salesman_015 WHERE NOT city = ANY (SELECT city
FROM customer_015)) ORDER BY 2 DESC;
```

4)

```
 SELECT a.salesman_id, name, a.city, 'MATCHED' FROM salesman_015 a, customer_015 b
WHERE a.city = b.city UNION (SELECT salesman_id, name, city, 'NO MATCH' FROM
salesman_015 WHERE NOT city = ANY (SELECT city FROM customer_015)) ORDER BY 2
DESC;
```

5)

```
SELECT customer_id, city, grade, 'High Rating' FROM customer_015 WHERE grade >= 200
UNION (SELECT customer_id, city, grade, 'Low Rating' FROM customer_015 WHERE grade <
200);
```



```
SQL> commit;

Commit complete.

SQL> SELECT salesman_015.salesman_id, name, cust_name, commission FROM salesman_015, customer_015 WHERE salesman_015.city = customer_015.city UNION (SELECT salesman_id, name, 'NO MATCH', commission FROM salesman
_015 WHERE NOT city = ANY (SELECT city FROM customer_015)) ORDER BY 2 DESC;

SALESMAN_ID NAME                     CUST_NAME                COMMISSION
----------- ------------------------ ------------------------ ----------
       5005 Pit Alex                 Julian Green                    .11
       5007 Paul Adam                NO MATCH                        .13
       5002 Nail Knite               Fabian Johnson                  .13
       5006 Mc Lyon                  Fabian Johnson                  .14
       5003 Lauson Hen               NO MATCH                        .12
       5001 James Hoog               Brad Davis                      .15
       5001 James Hoog               Nick Rimando                    .15

7 rows selected.

SQL>
SQL>  SELECT a.salesman_id, name, a.city, 'MATCHED' FROM salesman_015 a, customer_015 b WHERE a.city = b.city UNION (SELECT salesman_id, name, city, 'NO MATCH' FROM salesman_015 WHERE NOT city = ANY (SELECT city
 FROM customer_015)) ORDER BY 2 DESC;

SALESMAN_ID NAME                     CITY                     'MATCHED
----------- ------------------------ ------------------------ --------
       5005 Pit Alex                 London                   MATCHED
       5007 Paul Adam                Rome                     NO MATCH
       5002 Nail Knite               Paris                    MATCHED
       5006 Mc Lyon                  Paris                    MATCHED
       5003 Lauson Hen               San Jose                 NO MATCH
       5001 James Hoog               New York                 MATCHED

6 rows selected.

SQL>
SQL> SELECT customer_id, city, grade, 'High Rating' FROM customer_015 WHERE grade >= 200 UNION (SELECT customer_id, city, grade, 'Low Rating' FROM customer_015 WHERE grade < 200);

CUSTOMER_ID CITY                          GRADE 'HIGHRATING
----------- ------------------------ ---------- -----------
       3002 New York                        100 Low Rating
       3004 Paris                           300 High Rating
       3005 California                      200 High Rating
       3007 New York                        200 High Rating
       3008 London                          300 High Rating
       3009 Berlin                          100 Low Rating

6 rows selected.

SQL>
```

6)SELECT E.first_name , E.last_name , E.department_id , D.department_name
   FROM employee_015 E   JOIN dept_015 D ON E.department_id = D.department_id;

 7)SELECT E.first_name , E.last_name , E.department_id , D.department_name
       FROM employee_015 E JOIN dept_015 D ON E.department_id = D.department_id
AND E.department_id IN (90 , 60) ORDER BY E.last_name;

8)SELECT E.first_name, E.last_name, D.department_id, D.department_name
 FROM employee_015 E   RIGHT OUTER JOIN dept_015 D
ON E.department_id=D.department_id;

9)SELECT E.first_name, E.last_name, E.salary
  FROM employee_015 E JOIN employee_015 S ON E.salary < S.salary
AND S.employee_id = 182;

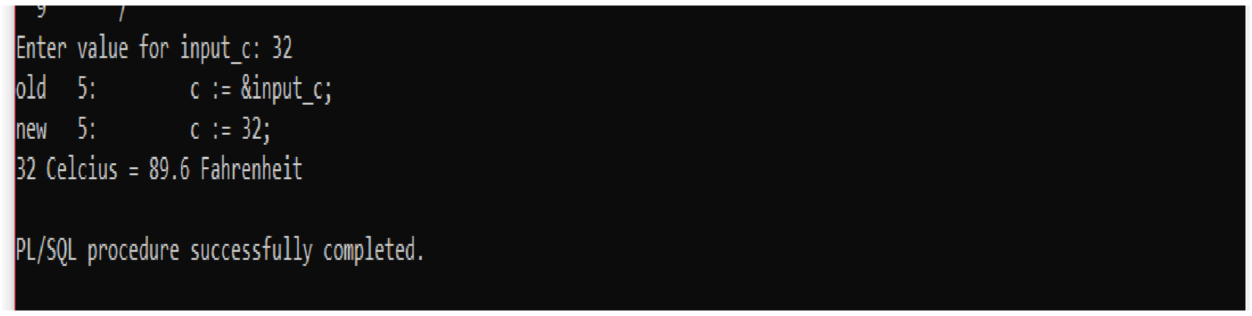
10)SELECT E.first_name, E.last_name, D.department_id, D.department_name
 FROM employee_015 E   LEFT OUTER JOIN dept_015 D
ON E.department_id=D.department_id;

```
11)SELECT department_name AS 'Department Name',
COUNT(*) AS 'No of Employees' FROM dept_015 INNER JOIN employee_015
ON employees.department_id = departments.department_id
GROUP BY departments.department_id, department_name
ORDER BY department_name;

12)SELECT department_name, first_name || ' ' || last_name AS name_of_manager, city
FROM dept_015 D JOIN employee_015 E ON (D.manager_id=E.employee_id)
JOIN locations L USING (location_id);

13)SELECT department_name, AVG(salary), COUNT(commission_pct) FROM dept_015 JOIN
employee_015 USING (department_id) GROUP BY department_name;

14)SELECT E.first_name, E.last_name, E.department_id  FROM employee_015 E
  JOIN employee_015 S    ON E.department_id = S.department_id  AND S.last_name = 'Ernst';

15)SELECT E.first_name AS "Employee Name",   M.first_name AS "Manager"
FROM employee-015 E  LEFT OUTER JOIN employee_015 M ON E.manager_id =
M.employee_id;

16)CREATE VIEW newyorkstaff
AS SELECT *FROM salesman_015 WHERE city = 'New York';

17)CREATE VIEW salesown AS SELECT salesman_id, name, city FROM salesman_015;

18)SELECT *FROM newyorkstaff WHERE commission > .13;

19)CREATE VIEW gradecount (grade, number) AS SELECT grade, COUNT(*)FROM
customer_015  GROUP BY grade;

20)CREATE VIEW mcustomer S SELECT *FROM salesman_015 a WHERE 1 < (SELECT
COUNT(*)  FROM customer b WHERE a.salesman_id = b.salesman_id);

21)CREATE VIEW highgrade
  AS SELECT *FROM customer_015WHERE grade =    (SELECT MAX (grade) FROM
customer);

22)CREATE VIEW citynum AS SELECT city, COUNT (DISTINCT salesman_id)
FROM salesman_015 GROUP BY city;
```

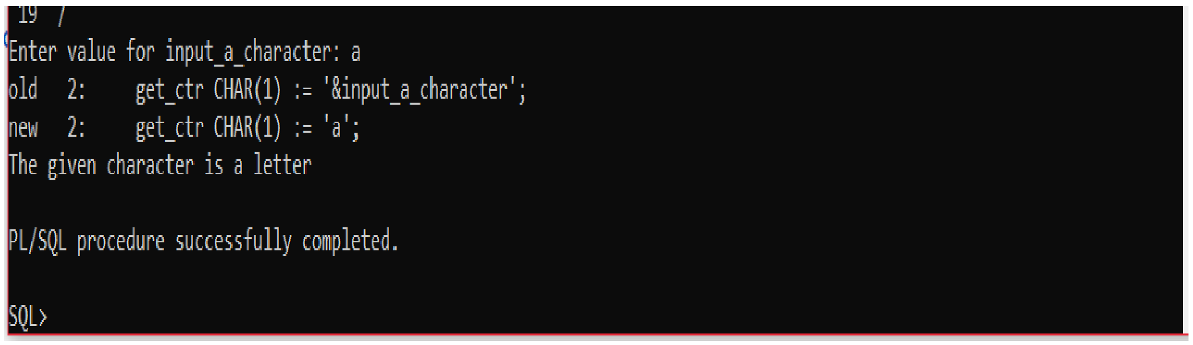
```
23)CREATE VIEW incentiveAS SELECT DISTINCT salesman_id, name
FROM salesman_015 aWHERE 3 <=(SELECT COUNT (*) FROM salesman_015 b
   WHERE a.salesman_id = b.salesman_id);
```

WEEK 9

RA18110290010015
M.Karthik vikram

1)
```
SET SERVEROUTPUT ON

DECLARE
        c NUMBER;
        f NUMBER;
BEGIN
        c := &input_c;
        f := 9/5 * c + 32;
        DBMS_OUTPUT.PUT_LINE (c ||' Celcius = '||f|| ' Fahrenheit');
        END;
        /
```

```
Enter value for input_c: 32
old   5:        c := &input_c;
new   5:        c := 32;
32 Celcius = 89.6 Fahrenheit

PL/SQL procedure successfully completed.
```

2)
```
SET SERVEROUTPUT ON



DECLARE
        get_ctr CHAR(1) := '&input_a_character';
BEGIN
        IF ( get_ctr >= 'A'
        AND get_ctr <= 'Z' )
        OR ( get_ctr >= 'a'
        AND get_ctr <= 'z' ) THEN
        dbms_output.Put_line ('The given character is a letter');
        ELSE
        dbms_output.Put_line ('The given character is not a letter');

        IF get_ctr BETWEEN '0' AND '9' THEN
        dbms_output.Put_line ('The given character is a number');
        ELSE
        dbms_output.Put_line ('The given character is not a number');
        END IF;
        END IF;
END;
/
```

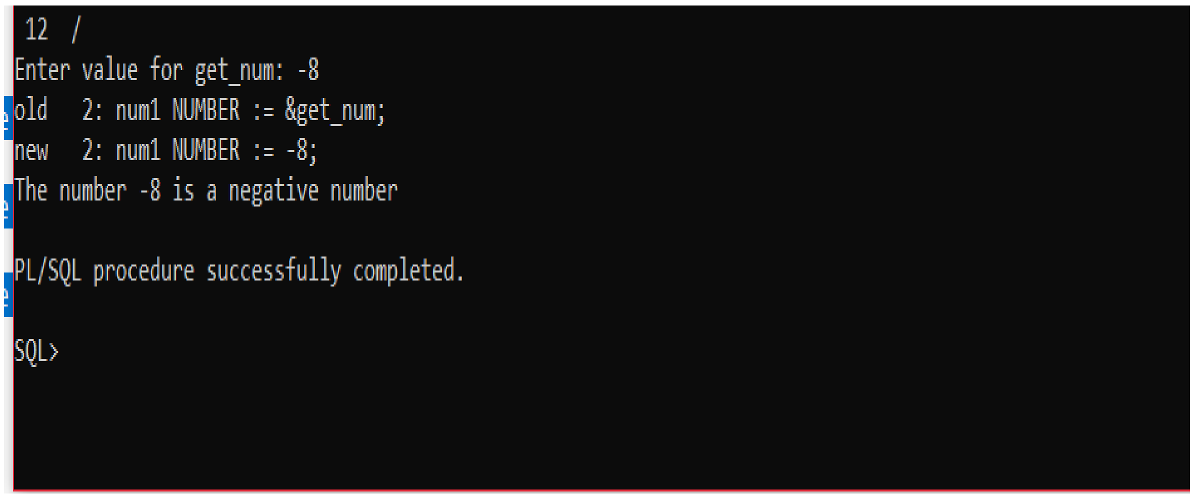


```
 19  /
Enter value for input_a_character: a
old   2:     get_ctr CHAR(1) := '&input_a_character';
new   2:     get_ctr CHAR(1) := 'a';
The given character is a letter

PL/SQL procedure successfully completed.

SQL>
```

3)SET SERVEROUTPUT ON

```
DECLARE
num1 NUMBER := &get_num1;
BEGIN
IF num1 < 0 THEN
DBMS_OUTPUT.PUT_LINE ('The number '||num1||' is a negative number');
ELSIF num1 = 0 THEN
DBMS_OUTPUT.PUT_LINE ('The number '||num1||' is equal to zero');
ELSE
DBMS_OUTPUT.PUT_LINE ('The number '||num1||' is a positive number');
END IF;
```

```
END;
/
```



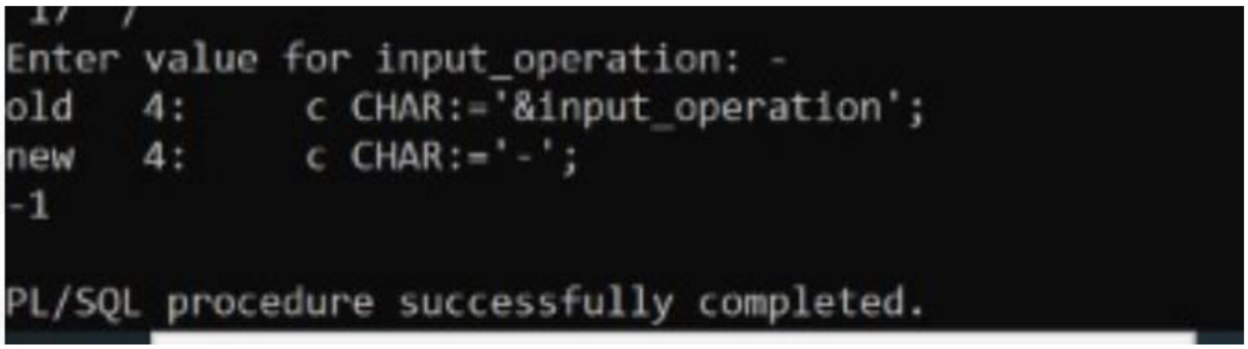
```
 12  /
Enter value for get_num: -8
old   2: num1 NUMBER := &get_num;
new   2: num1 NUMBER := -8;
The number -8 is a negative number

PL/SQL procedure successfully completed.

SQL>
```

4)SET SERVEROUTPUT ON

```
DECLARE
  grade CHAR(1);
BEGIN

grade CHAR(1) := ‘&get_grade’;

  IF grade = 'A' THEN
          DBMS_OUTPUT.PUT_LINE('Excellent');
  ELSIF grade = 'B' THEN
          DBMS_OUTPUT.PUT_LINE('Very Good');
  ELSIF grade = 'C' THEN
          DBMS_OUTPUT.PUT_LINE('Good');
  ELSIF grade = 'D' THEN
          DBMS_OUTPUT. PUT_LINE('Fair');
  ELSIF grade = 'F' THEN
          DBMS_OUTPUT.PUT_LINE('Poor');
  ELSE
          DBMS_OUTPUT.PUT_LINE('No such grade');
  END IF;
END;
/
```

5)
SET SERVEROUTPUT ON

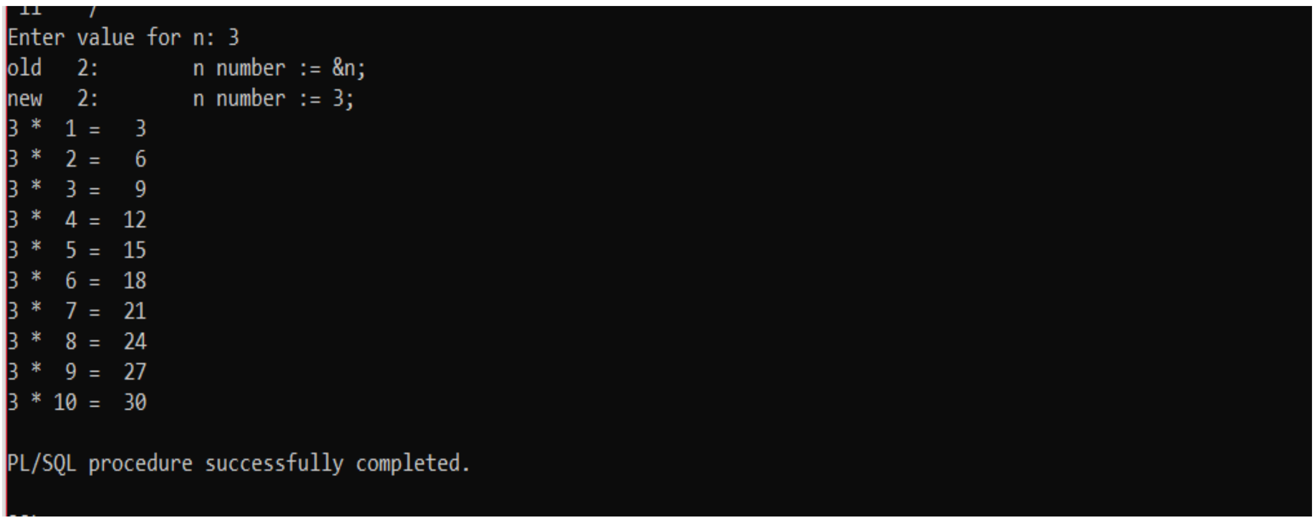
```
DECLARE
a NUMBER :=&get_a;
b NUMBER :=&get_b;
arth_operation CHAR(20);
BEGIN

arth_operation := &input_arth_operation;

dbms_output.put_line('Program started.');
CASE
WHEN arth_operation = 'ADD'
THEN dbms_output.put_line('Addition of the numbers are: '||a+b );
WHEN arth_operation = 'SUBTRACT'
THEN dbms_output.put_line('Subtraction of the numbers are: '|| a-b);
WHEN arth_operation = 'MULTIPLY'
THEN dbms_output.put_line('Multiplication of the numbers are: '|| a*b );
WHEN arth_operation = 'DIVIDE'
THEN dbms_output.put_line('Division of the numbers are: '|| a/b ):
 ELSE dbms_output.put_line('No operation action defined. Invalid operation');
END CASE;
dbms_output.put_line('Program completed.' );
END;
/
```



```
 17  /
Enter value for input_operation: -
old   4:      c CHAR:='&input_operation';
new   4:      c CHAR:='-';
-1

PL/SQL procedure successfully completed.
```

6)SET SERVEROUTPUT ON

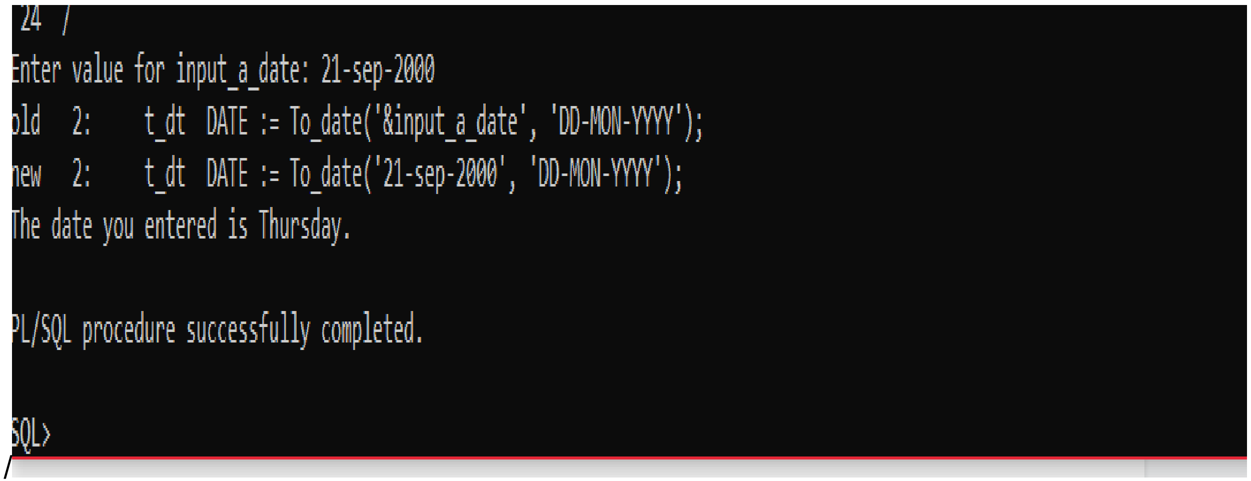
```
DECLARE
        n number := &n;
        prod number;
        BEGIN
        for i in 1..10 loop
        prod := n * i;
        dbms_output.put_line(n||' * '||lpad(i,2,' ')
        ||' = '||lpad(prod,3,' '));
        end loop;
  END;
 /
```



```
Enter value for n: 3
old   2:        n number := &n;
new   2:        n number := 3;
3 *  1 =   3
3 *  2 =   6
3 *  3 =   9
3 *  4 =  12
3 *  5 =  15
3 *  6 =  18
3 *  7 =  21
3 *  8 =  24
3 *  9 =  27
3 * 10 =  30

PL/SQL procedure successfully completed.
```

```
7)SET SERVEROUTPUT ON

DECLARE
        t_dt  DATE := To_date('&input_a_date', 'DD-MON-YYYY');
        t_day VARCHAR2(1);
BEGIN
        t_day := To_char(t_dt, 'D');

        CASE t_day
        WHEN '1' THEN
        dbms_output.Put_line ('The date you entered is Sunday.');
        WHEN '2' THEN
        dbms_output.Put_line ('The date you entered is Monday.');
        WHEN '3' THEN
```

```
          dbms_output.Put_line ('The date you entered is Tuesday.');
          WHEN '4' THEN
          dbms_output.Put_line ('The date you entered is Wednesday.');
          WHEN '5' THEN
          dbms_output.Put_line ('The date you entered is Thursday.');
          WHEN '6' THEN
          dbms_output.Put_line ('The date you entered is Friday.');
          WHEN '7' THEN
          dbms_output.Put_line ('The date you entered is Saturday.');
          END CASE;
END;
```



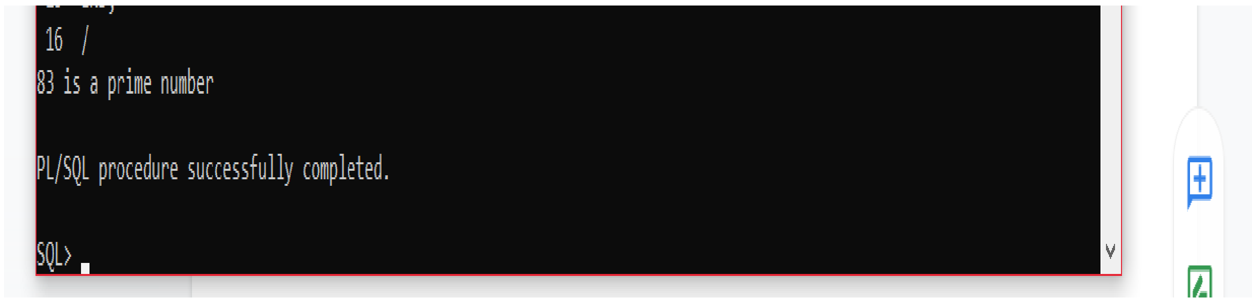
```
 24  /
Enter value for input_a_date: 21-sep-2000
old   2:     t_dt  DATE := To_date('&input_a_date', 'DD-MON-YYYY');
new   2:     t_dt  DATE := To_date('21-sep-2000', 'DD-MON-YYYY');
The date you entered is Thursday.

PL/SQL procedure successfully completed.

SQL>
```

```
/
```

8)SET SERVEROUTPUT ON

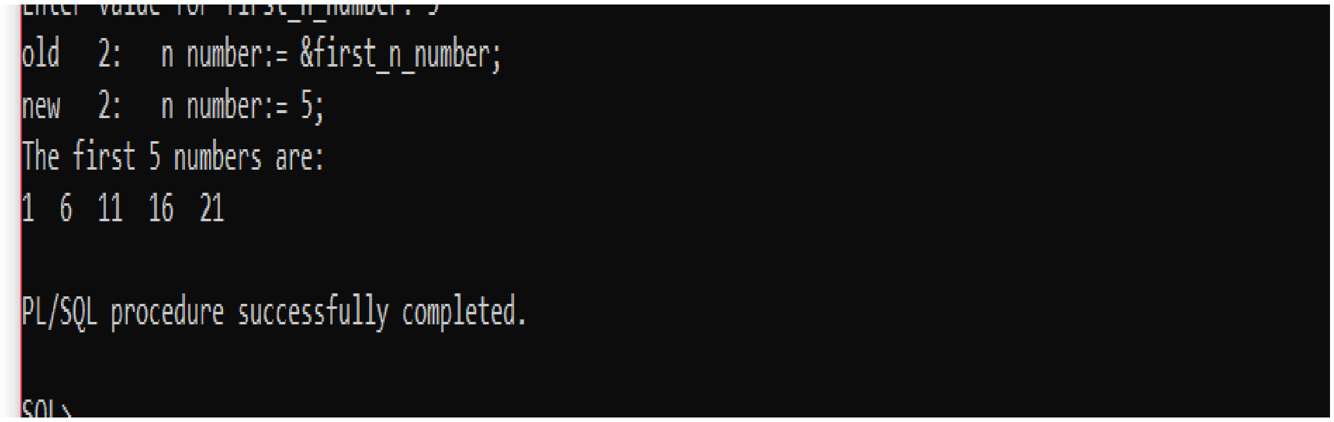
```
DECLARE
  msg  VARCHAR2(30);
  n  PLS_INTEGER := 83;
BEGIN
  FOR i in 2..ROUND(SQRT(n)) LOOP
          IF n MOD i = 0 THEN
          msg := ' is not a prime number';
          GOTO when_prime;
          END IF;
  END LOOP;

  msg := ' is a prime number';
  <<when_prime>>
   DBMS_OUTPUT.PUT_LINE(TO_CHAR(n) || msg);
```

END;
/



```
 16  /
83 is a prime number

PL/SQL procedure successfully completed.

SQL>
```

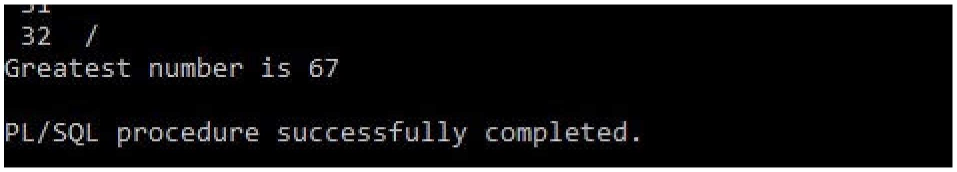
```
9)SET SERVEROUTPUT ON

DECLARE
  n number:= &first_n_number;
  i number:=1;
  m number:=1;
BEGIN
 DBMS_OUTPUT.PUT_LINE ('The first '||n||' numbers are: ');
  DBMS_OUTPUT.PUT (i||'  ');
           for i in 1..n-1 loop
           m:=m+5;
           dbms_output.put(m||'  ');
           END LOOP;
           dbms_output.new_line;
 END;
/
```



```
old   2:   n number:= &first_n_number;
new   2:   n number:= 5;
The first 5 numbers are:
1  6  11  16  21

PL/SQL procedure successfully completed.

SQL>
```

10)SET SERVEROUTPUT ON

```
declare
A NUMBER:=&get_a;
B NUMBER:=&get_b;
C NUMBER:=&get_c;
D NUMBER:=&get_d;
E NUMBER:=&get_e;
begin
   dbms_output.put_line('A='||A||' B='||B||' C='||C||' D='||D||' E='||E');
GREATEST(A,B,C,D,E);

   end;
/
```

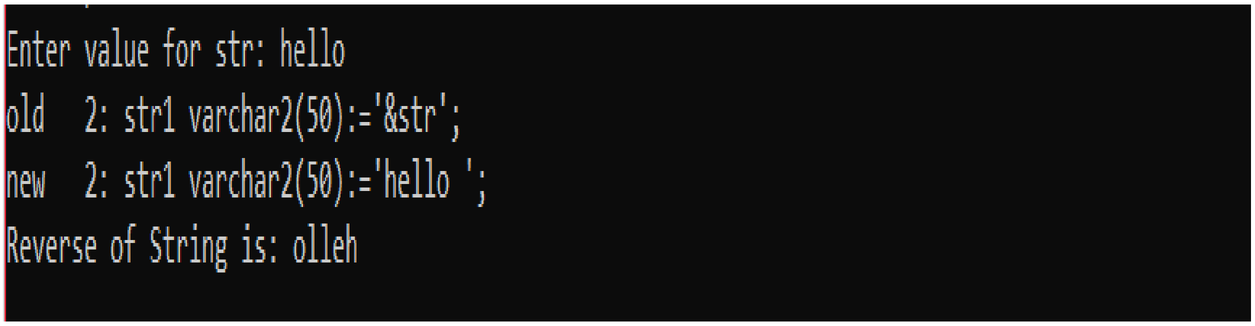


```
 32  /
Greatest number is 67

PL/SQL procedure successfully completed.
```

DBMS week 10
Procedure:-

1)

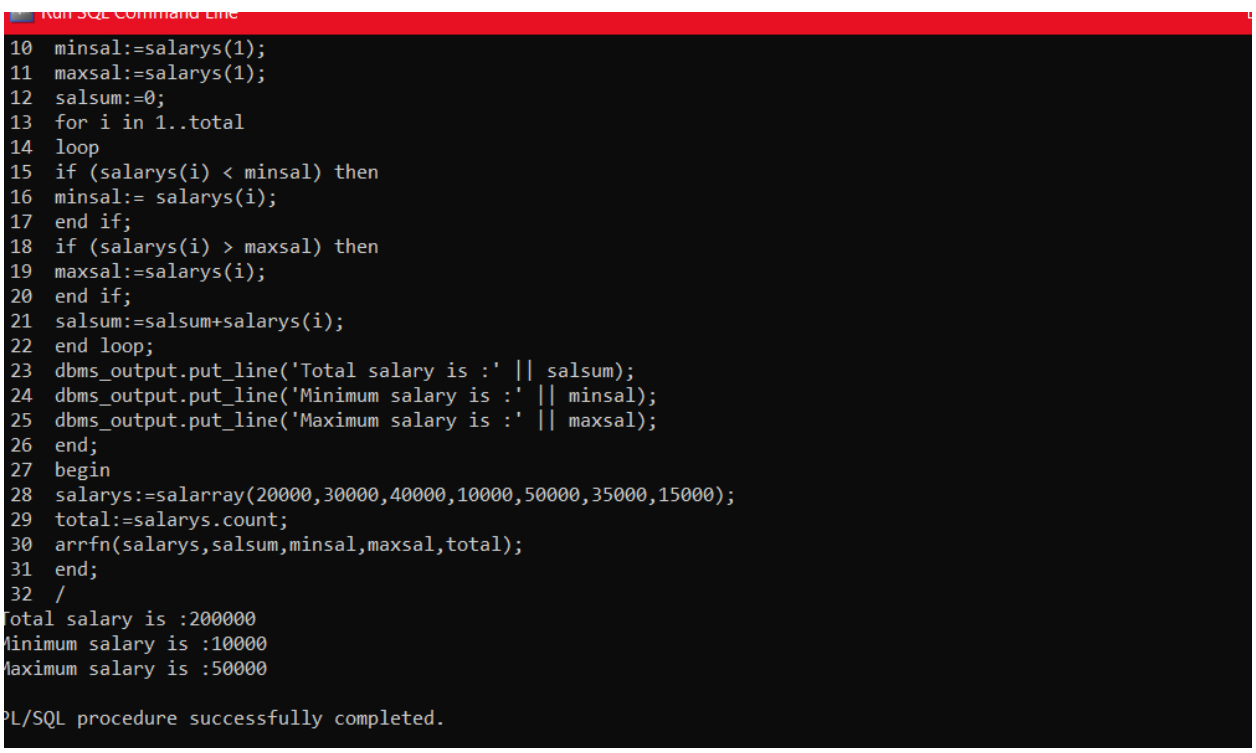
```
declare
str1 varchar2(50):='&str';
str2 varchar2(50);
        len number;
        i number;
 Begin
        len:=length(str1);
for i in reverse 1..len
loop
str2:=str2 || substr(str1,i,1);
end loop;
dbms_output.put_line('Reverse of String is:'||str2);
End;
 /
```



```
Enter value for str: hello
old   2: str1 varchar2(50):='&str';
new   2: str1 varchar2(50):='hello ';
Reverse of String is: olleh
```

2)declare

```
   type salarray is varray(20) of number;
   salarys salarray;
   salsum number;
   minsal number;
   maxsal number;
   total number;
   procedure arrfn(salarys in salarray,salsum in out number,minsal in out
number,maxsal in out number,total in number) is
   begin
         minsal:=salarys(1);
         maxsal:=salarys(1);
         salsum:=0;
         for i in 1..total
         loop
                 if (salarys(i) < minsal) then
                         minsal:= salarys(i);
                 end if;
                 if (salarys(i) > maxsal) then
                         maxsal:=salarys(i);
                 end if;
                 salsum:=salsum+salarys(i);
         end loop;
         dbms_output.put_line('Total salary is :' || salsum);
         dbms_output.put_line('Minimum salary is :' || minsal);
         dbms_output.put_line('Maximum salary is :' || maxsal);
   end;
begin
   salarys:=salarray(20000,30000,40000,10000,50000,35000,15000);
   total:=salarys.count;
   arrfn(salarys,salsum,minsal,maxsal,total);
end;
/
```

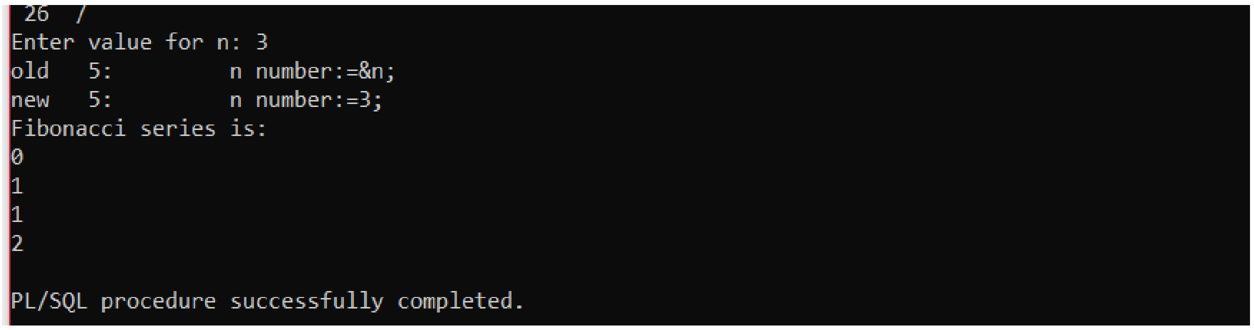


```
minsal:=salarys(1);
maxsal:=salarys(1);
salsum:=0;
for i in 1..total
loop
if (salarys(i) < minsal) then
minsal:= salarys(i);
end if;
if (salarys(i) > maxsal) then
maxsal:=salarys(i);
end if;
salsum:=salsum+salarys(i);
end loop;
dbms_output.put_line('Total salary is :' || salsum);
dbms_output.put_line('Minimum salary is :' || minsal);
dbms_output.put_line('Maximum salary is :' || maxsal);
end;
begin
salarys:=salarray(20000,30000,40000,10000,50000,35000,15000);
total:=salarys.count;
arrfn(salarys,salsum,minsal,maxsal,total);
end;
/
otal salary is :200000
inimum salary is :10000
aximum salary is :50000

L/SQL procedure successfully completed.
```

3) declare

```
first number:=0;
        second number:=1;
        third number;
        n number:=&n;
        i number;
 PROCEDURE fib(id IN NUMBER)
is
  begin
        dbms_output.put_line('Fibonacci series is:');
        dbms_output.put_line(first);
        dbms_output.put_line(second);

        for i in 2..n
        loop
        third:=first+second;
        first:=second;
```

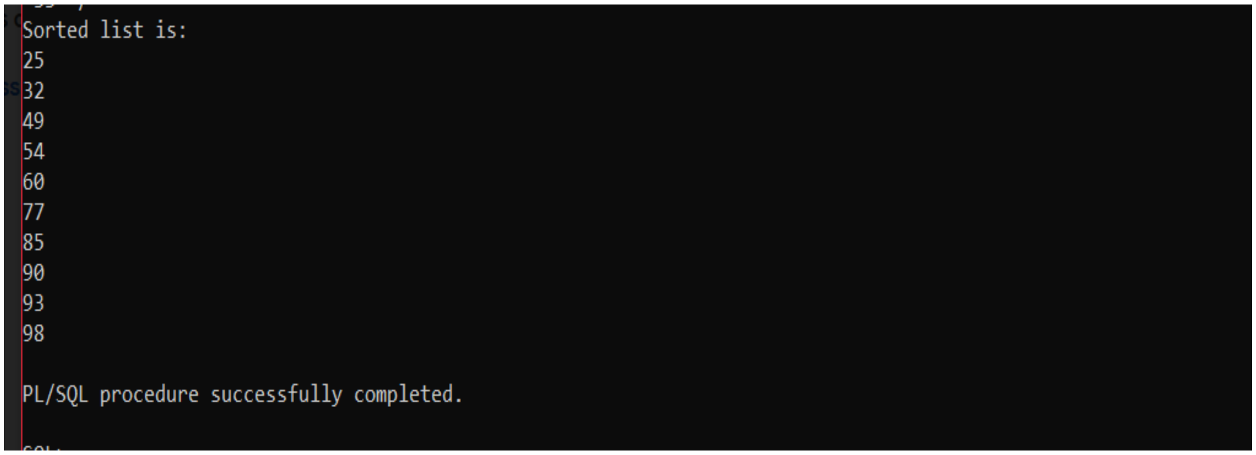
```
          second:=third;
          dbms_output.put_line(third);
          end loop;

          end;
begin
          fib(n);
end;
 /
Enter value for n: 3
old   5:          n number:=&n;
new   5:          n number:=3;
Fibonacci series is:
0
1
1
2
```



```
 26  /
Enter value for n: 3
old   5:         n number:=&n;
new   5:         n number:=3;
Fibonacci series is:
0
1
1
2

PL/SQL procedure successfully completed.
```

```
4)declare
   type arr is varray(20) of number;
   numarr arr;
   i number;
   j number;
   k number;
   temp number;
   length number;
   procedure bubbsort(numarr in out arr, length in number, temp in out number) is
   begin
         for i in 1..length
         loop
                   for j in 1..length-i
```

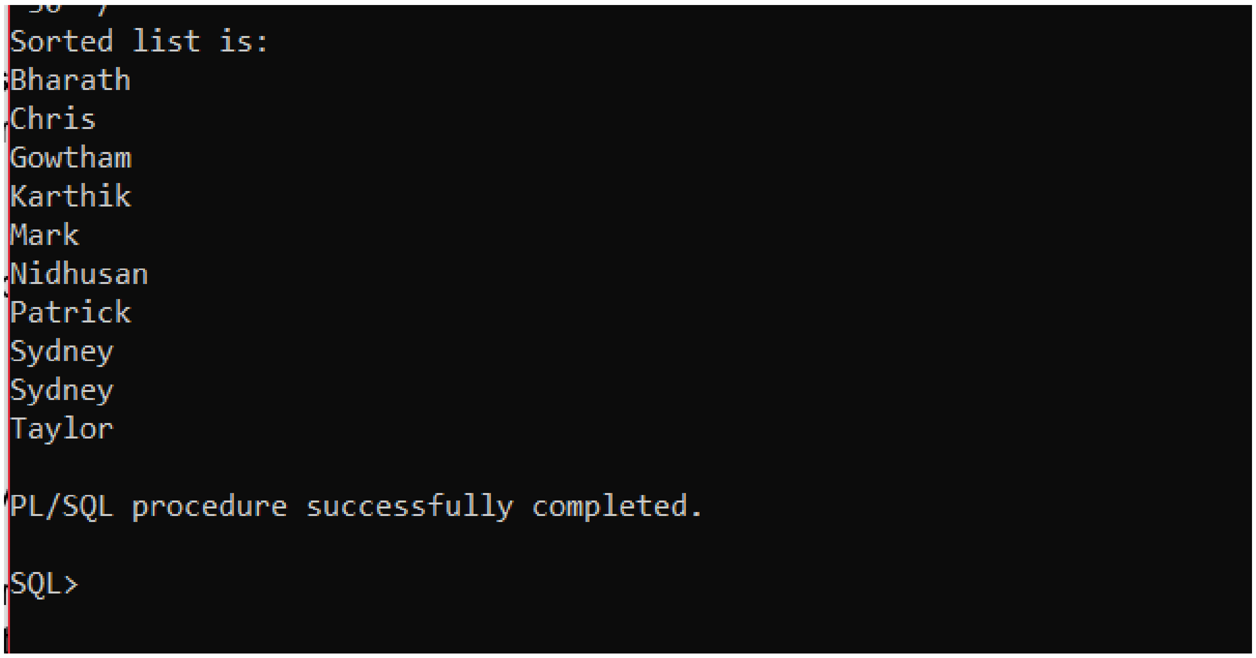
```
                loop
                        if (numarr(j)>numarr(j+1)) then
                                temp:= numarr(j);
                                numarr(j):=numarr(j+1);
                                numarr(j+1):=temp;
                        end if;
                end loop;
        end loop;
        dbms_output.put_line('Sorted list is:');
        for i in 1..length
        loop
                dbms_output.put_line(numarr(i));
        end loop;
   end;
begin
   numarr:=arr(60,90,49,32,98,25,54,77,93,85);
   length:=numarr.count;
   bubbsort(numarr,length,temp);
end;
/
```



```
Sorted list is:
25
32
49
54
60
77
85
90
93
98

PL/SQL procedure successfully completed.
```

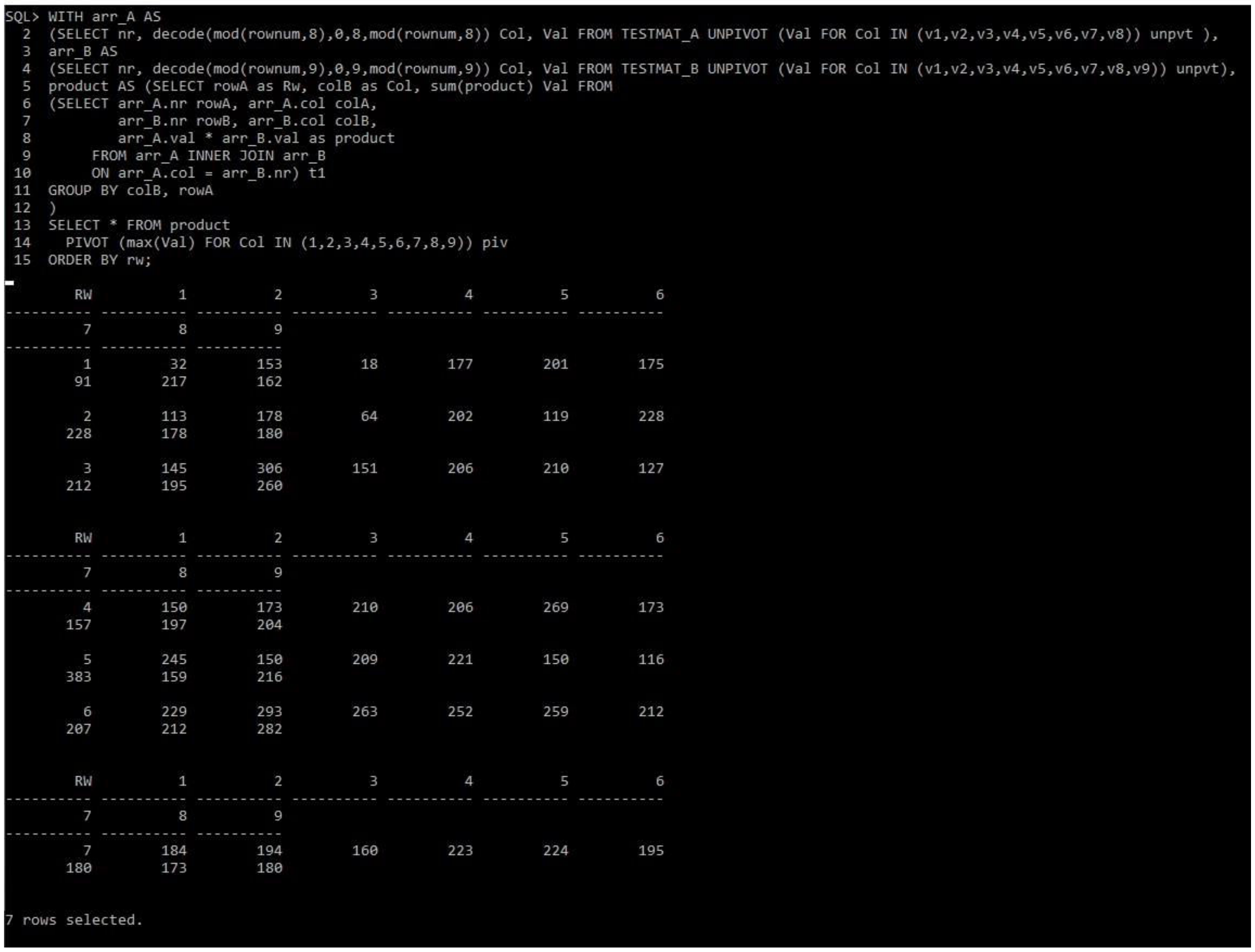
```
5)declare
   type strarray is varray(20) of varchar2(20);
   strs strarray;
   temp varchar2(20);
   length number;
   procedure alphsort(strs in out strarray, length in number, temp in out varchar2) is
   begin
         for i in 1..length
```

```
        loop
                for j in 1..length-i
                loop
                        if (strs(j)>strs(j+1)) then
                                temp:= strs(j);
                                strs(j):=strs(j+1);
                                strs(j+1):=temp;
                        end if;
                end loop;
        end loop;
        dbms_output.put_line('Sorted list is:');
        for i in 1..length
        loop
                dbms_output.put_line(strs(i));
        end loop;
   end;
begin

strs:=strarray('Nidhusan','Karthik','Bharath','Sydney','Sydney','Mark','Taylor','Chris','Patri
ck','Gowtham');
   length:=strs.count;
   alphsort(strs,length,temp);
end;
/
```



```
Sorted list is:
Bharath
Chris
Gowtham
Karthik
Mark
Nidhusan
Patrick
Sydney
Sydney
Taylor

PL/SQL procedure successfully completed.

SQL>
```

6)

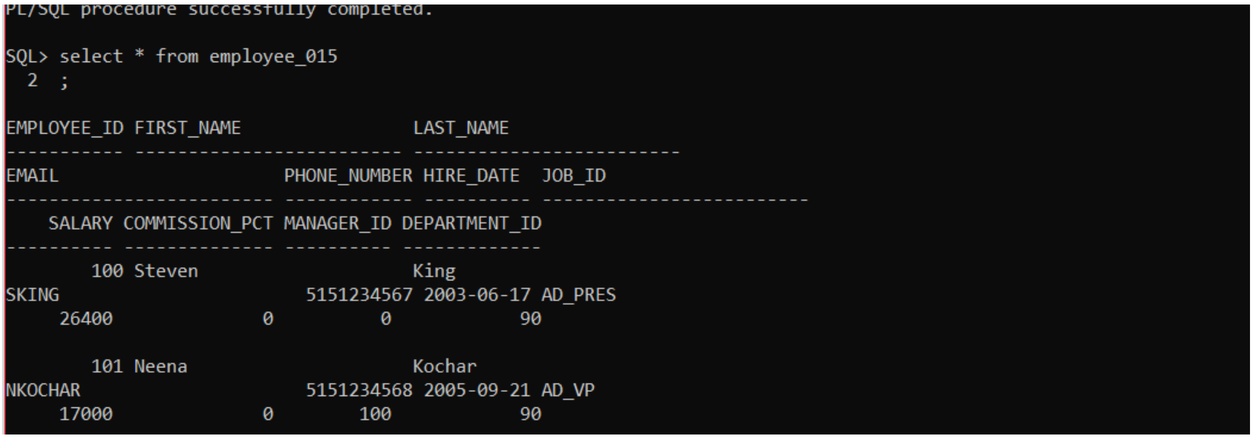


```
SQL> WITH arr_A AS
(SELECT nr, decode(mod(rownum,8),0,8,mod(rownum,8)) Col, Val FROM TESTMAT_A UNPIVOT (Val FOR Col IN (v1,v2,v3,v4,v5,v6,v7,v8)) unpvt ),
arr_B AS
(SELECT nr, decode(mod(rownum,9),0,9,mod(rownum,9)) Col, Val FROM TESTMAT_B UNPIVOT (Val FOR Col IN (v1,v2,v3,v4,v5,v6,v7,v8,v9)) unpvt),
product AS (SELECT rowA as Rw, colB as Col, sum(product) Val FROM
(SELECT arr_A.nr rowA, arr_A.col colA,
        arr_B.nr rowB, arr_B.col colB,
        arr_A.val * arr_B.val as product
     FROM arr_A INNER JOIN arr_B
     ON arr_A.col = arr_B.nr) t1
GROUP BY colB, rowA
)
SELECT * FROM product
  PIVOT (max(Val) FOR Col IN (1,2,3,4,5,6,7,8,9)) piv
ORDER BY rw;

        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         1         32        153         18        177        201        175
        91        217        162

         2        113        178         64        202        119        228
       228        178        180

         3        145        306        151        206        210        127
       212        195        260


        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         4        150        173        210        206        269        173
       157        197        204

         5        245        150        209        221        150        116
       383        159        216

         6        229        293        263        252        259        212
       207        212        282


        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         7        184        194        160        223        224        195
       180        173        180


7 rows selected.
```

7)DECLARE

```
CURSOR employee_cur IS
SELECT employee_id,
salary,
first_name
FROM   employee_015
WHERE  employee_id = 100
FOR UPDATE;
```

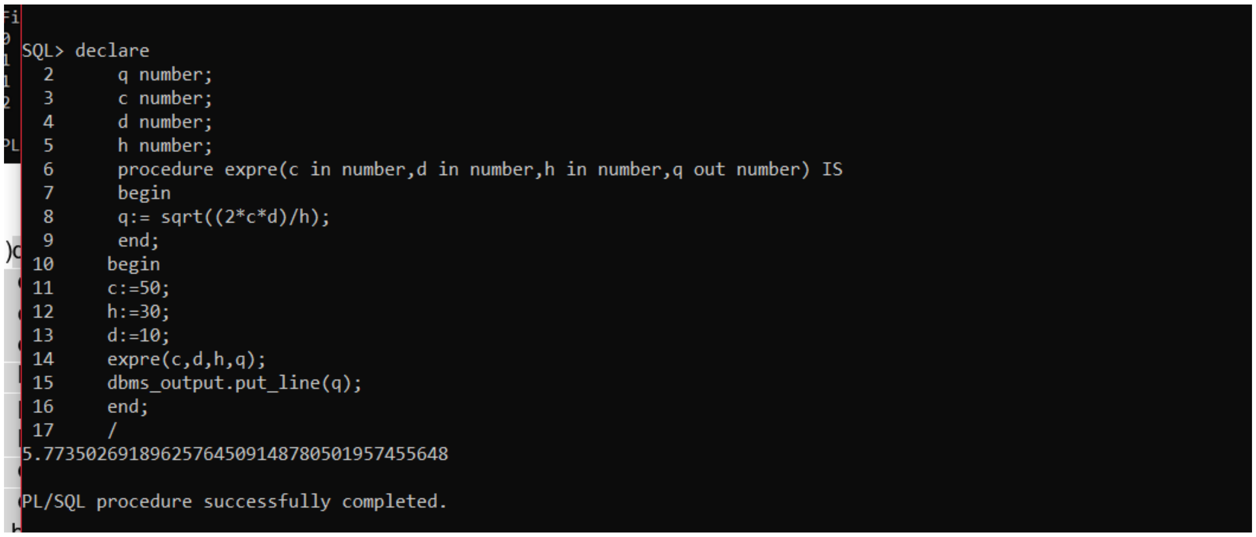
```
        incr_sal NUMBER;
BEGIN
        FOR employee_rec IN employee_cur LOOP
        incr_sal := .10;
UPDATE employee_015
        SET    salary = salary + salary * incr_sal
        WHERE  CURRENT OF employee_cur;
        END LOOP;
END;
/
```



```
PL/SQL procedure successfully completed.

SQL> select * from employee_015
  2  ;

EMPLOYEE_ID FIRST_NAME                LAST_NAME
----------- ------------------------- -------------------------
EMAIL                     PHONE_NUMBER HIRE_DATE  JOB_ID
------------------------- ------------ ---------- -------------------------
    SALARY COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
---------- -------------- ---------- -------------
        100 Steven                    King
SKING                       5151234567 2003-06-17 AD_PRES
     26400              0          0            90

        101 Neena                     Kochar
NKOCHAR                     5151234568 2005-09-21 AD_VP
     17000              0        100            90
```

8)declare

```
   q number;
   c number;
   d number;
   h number;
   procedure expre(c in number,d in number,h in number,q out number) IS
   begin
   q:= sqrt((2*c*d)/h);
   end;
  begin
  c:=50;
  h:=30;
  d:=10;
  expre(c,d,h,q);
  dbms_output.put_line(q);
  end;
  /
```



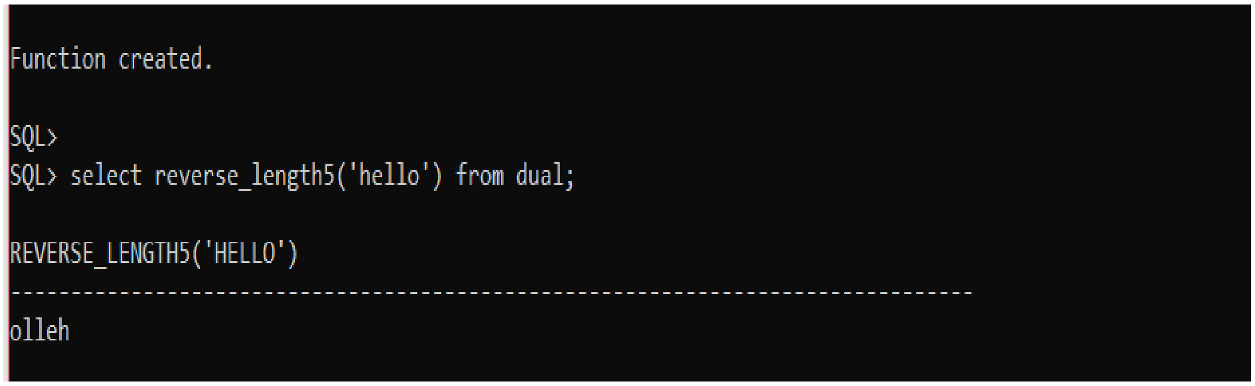
```
SQL> declare
      q number;
      c number;
      d number;
      h number;
      procedure expre(c in number,d in number,h in number,q out number) IS
      begin
      q:= sqrt((2*c*d)/h);
      end;
     begin
     c:=50;
     h:=30;
     d:=10;
     expre(c,d,h,q);
     dbms_output.put_line(q);
     end;
     /
5.77350269189625764509148780501957455648

PL/SQL procedure successfully completed.
```

Week 11

FUNCTIONS:-

```
1)create function reverse_length14(x varchar)
return varchar
is
len int;
str1 varchar(25);
begin
len:= Length(x);
   FOR i IN REVERSE 1.. len LOOP
          -- assigning the reverse string in str1
          str1 := str1
                    || Substr(x, i, 1);
          END LOOP;
return(str1);
end ;
/
```

select reverse_length5('hello') from dual;

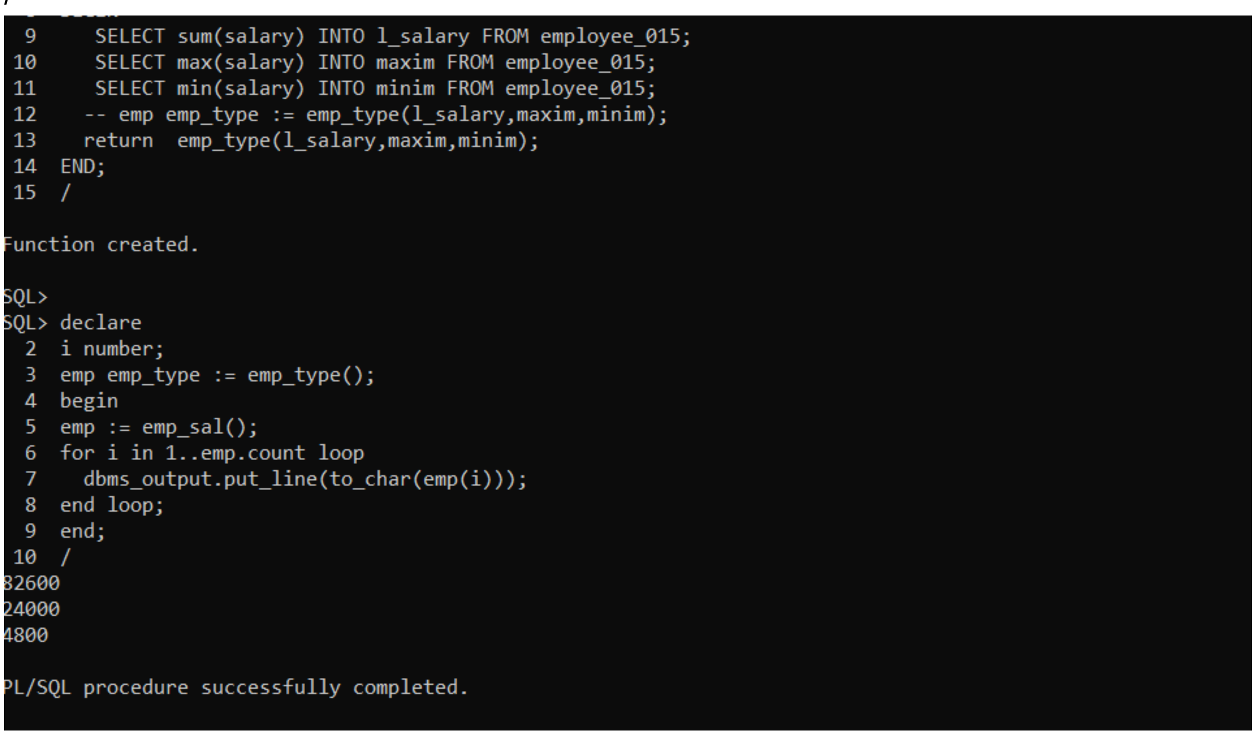


```
Function created.

SQL>
SQL> select reverse_length5('hello') from dual;

REVERSE_LENGTH5('HELLO')
--------------------------------------------------------------------------------
olleh
```

2)create or replace type emp_type AS VARRAY(25) OF VARCHAR(10);
/

```
create or replace function emp_sal
return emp_type
is
  emp emp_type := emp_type();
  l_salary number(10);
  maxim number(10);
  minim number(10);
BEGIN
   SELECT sum(salary) INTO l_salary FROM employee_015;
   SELECT max(salary) INTO maxim FROM employee_015;
   SELECT min(salary) INTO minim FROM employee_015;
  -- emp emp_type := emp_type(l_salary,maxim,minim);
  return  emp_type(l_salary,maxim,minim);
END;
/

declare
i number;
emp emp_type := emp_type();
begin
emp := emp_sal();
for i in 1..emp.count loop
  dbms_output.put_line(to_char(emp(i)));
end loop;
end;
```

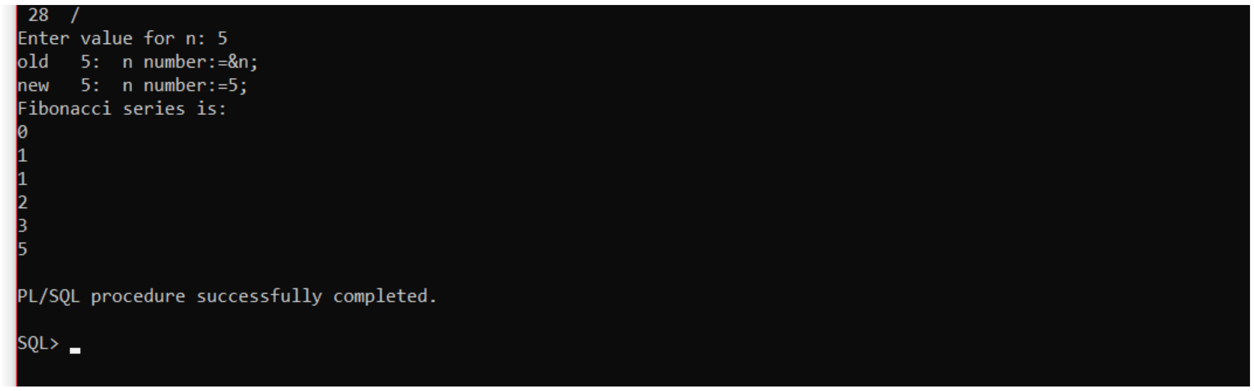
/



```
    SELECT sum(salary) INTO l_salary FROM employee_015;
    SELECT max(salary) INTO maxim FROM employee_015;
    SELECT min(salary) INTO minim FROM employee_015;
   -- emp emp_type := emp_type(l_salary,maxim,minim);
   return  emp_type(l_salary,maxim,minim);
 END;
 /

Function created.

SQL>
SQL> declare
 i number;
 emp emp_type := emp_type();
 begin
 emp := emp_sal();
 for i in 1..emp.count loop
   dbms_output.put_line(to_char(emp(i)));
 end loop;
 end;
 /
82600
24000
4800

PL/SQL procedure successfully completed.
```

```
3)declare
 first number:=0;
second number:=1;
third number;
 n number:=&n;
 i number;
fibonacci number;
FUNCTION fib (first in out number,second in out number,third in out number,n in out
number)
RETURN number
IS
BEGIN
dbms_output.put_line('Fibonacci series is:');
dbms_output.put_line(first);
dbms_output.put_line(second);
for i in 2..n
loop
third:=first+second;
first:=second;
second:=third;
```

```
dbms_output.put_line(third);
end loop;
return null;
end;
Begin
fibonacci:=fib(first,second,third,n);
dbms_output.put_line(fibonacci);
END ;
/
```



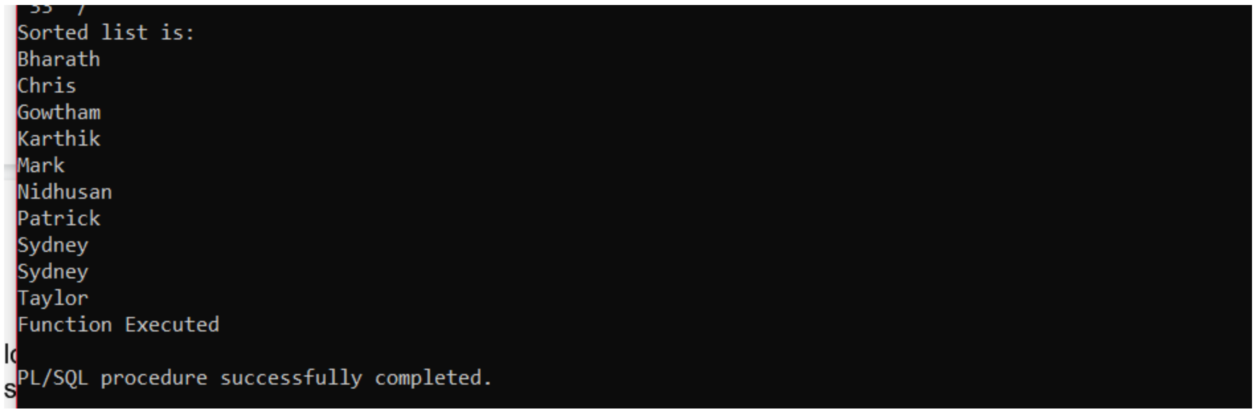
```
 28  /
Enter value for n: 5
old   5:  n number:=&n;
new   5:  n number:=5;
Fibonacci series is:
0
1
1
2
3
5

PL/SQL procedure successfully completed.

SQL>
```

```
4)declare
   type arr is varray(20) of number;
   numarr arr;
   i number;
   j number;
   k number;
   temp number;
   length number;
   bubblesort varchar2(20);
   function bubbsort(numarr in out arr, length in number, temp in out number) return
varchar2 is
   begin
         for i in 1..length
         loop
                 for j in 1..length-i
                 loop
                         if (numarr(j)>numarr(j+1)) then
                                 temp:= numarr(j);
```

```
                    numarr(j):=numarr(j+1);
                    numarr(j+1):=temp;
                end if;
            end loop;
        end loop;
        dbms_output.put_line('Sorted list is:');
        for i in 1..length
        loop
            dbms_output.put_line(numarr(i));
        end loop;
        return 'Function Executed';
    end;
begin
    numarr:=arr(64,56,49,32,98,25,19,77,93,85);
    length:=numarr.count;
    bubblesort:=bubbsort(numarr,length,temp);
    dbms_output.put_line(bubblesort);
end;
/
```

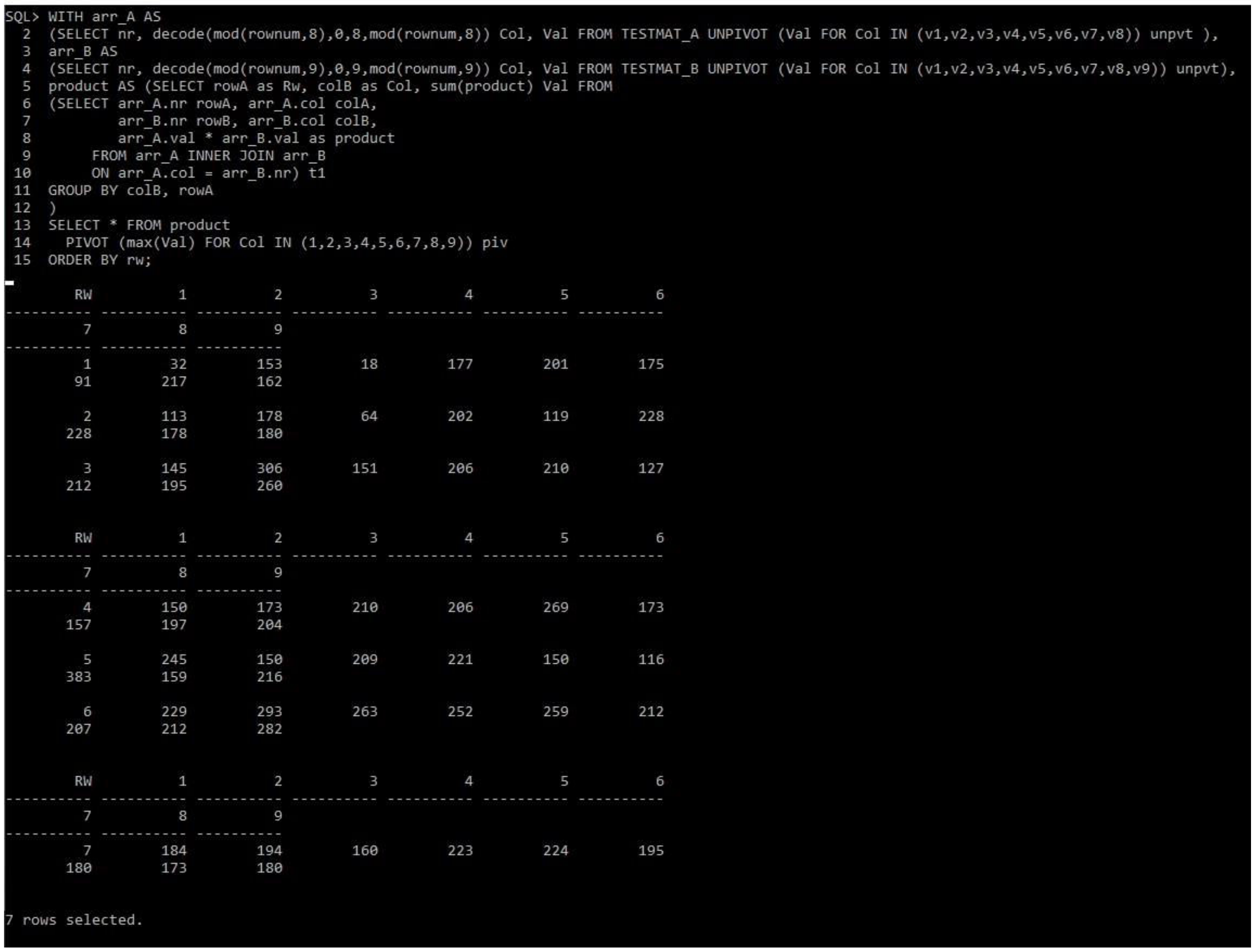
```
5)declare
    type strarray is varray(20) of varchar2(20);
    strs strarray;
    temp varchar2(20);
    length number;
    stringsort varchar2(20);
    function alphsort(strs in out strarray, length in number, temp in out varchar2) return
varchar2 is
    begin
        for i in 1..length
        loop
            for j in 1..length-i
            loop
                if (strs(j)>strs(j+1)) then
                    temp:= strs(j);
                    strs(j):=strs(j+1);
                    strs(j+1):=temp;
                end if;
            end loop;
        end loop;
```

```
            dbms_output.put_line('Sorted list is:');
            for i in 1..length
            loop
                    dbms_output.put_line(strs(i));
            end loop;
            return 'Function Executed';
    end;
begin

strs:=strarray('Nidhusan','Karthik','Bharath','Sydney','Sydney','Mark','Taylor','Chris','Patri
ck','Gowtham');
    length:=strs.count;
    stringsort:=alphsort(strs,length,temp);
    dbms_output.put_line(stringsort);
end;
/
```



```
Sorted list is:
Bharath
Chris
Gowtham
Karthik
Mark
Nidhusan
Patrick
Sydney
Sydney
Taylor
Function Executed

PL/SQL procedure successfully completed.
```

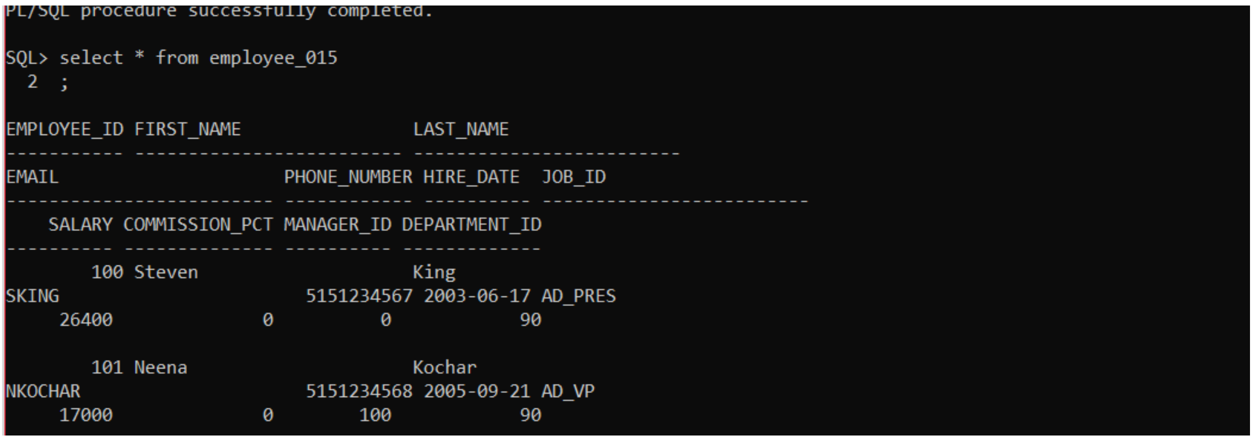
6)



```
SQL> WITH arr_A AS
(SELECT nr, decode(mod(rownum,8),0,8,mod(rownum,8)) Col, Val FROM TESTMAT_A UNPIVOT (Val FOR Col IN (v1,v2,v3,v4,v5,v6,v7,v8)) unpvt ),
arr_B AS
(SELECT nr, decode(mod(rownum,9),0,9,mod(rownum,9)) Col, Val FROM TESTMAT_B UNPIVOT (Val FOR Col IN (v1,v2,v3,v4,v5,v6,v7,v8,v9)) unpvt),
product AS (SELECT rowA as Rw, colB as Col, sum(product) Val FROM
(SELECT arr_A.nr rowA, arr_A.col colA,
        arr_B.nr rowB, arr_B.col colB,
        arr_A.val * arr_B.val as product
     FROM arr_A INNER JOIN arr_B
     ON arr_A.col = arr_B.nr) t1
GROUP BY colB, rowA
)
SELECT * FROM product
  PIVOT (max(Val) FOR Col IN (1,2,3,4,5,6,7,8,9)) piv
ORDER BY rw;

        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         1         32        153         18        177        201        175
        91        217        162

         2        113        178         64        202        119        228
       228        178        180

         3        145        306        151        206        210        127
       212        195        260


        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         4        150        173        210        206        269        173
       157        197        204

         5        245        150        209        221        150        116
       383        159        216

         6        229        293        263        252        259        212
       207        212        282


        RW          1          2          3          4          5          6
---------- ---------- ---------- ---------- ---------- ---------- ----------
         7          8          9
---------- ---------- ----------
         7        184        194        160        223        224        195
       180        173        180


7 rows selected.
```

```
7)DECLARE
        CURSOR employee_cur IS
        SELECT employee_id,
        salary,
        first_name
        FROM    employee_015
        WHERE  employee_id = 100
        FOR UPDATE;
        incr_sal NUMBER;
BEGIN
        FOR employee_rec IN employee_cur LOOP
        incr_sal := .10;
```

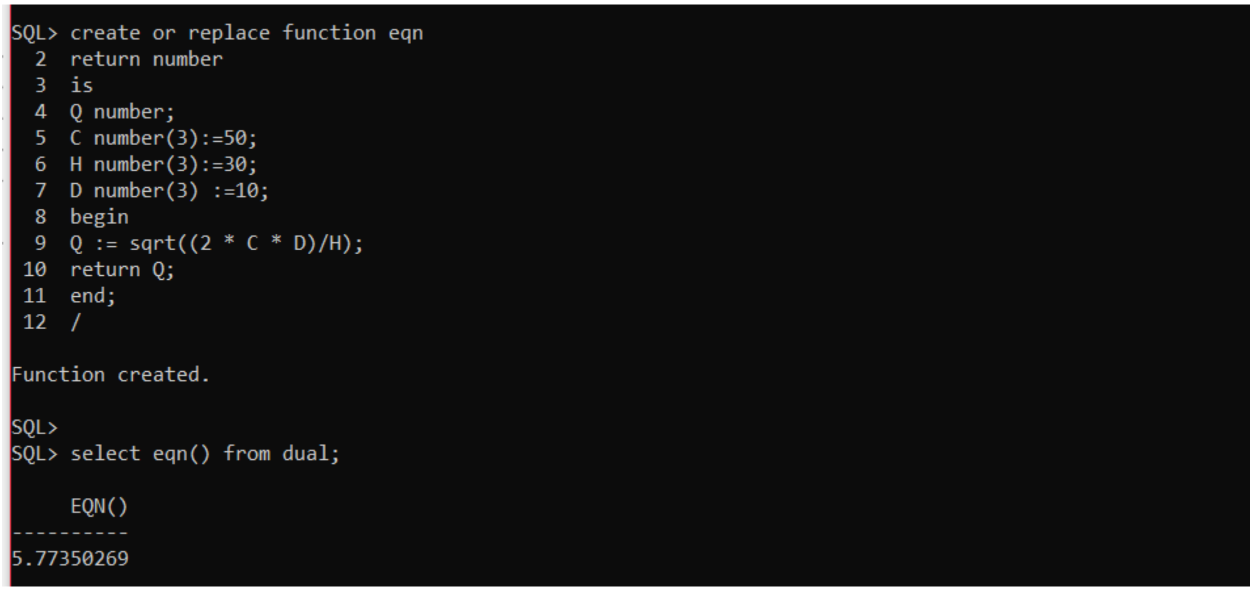
```
UPDATE employee_015
        SET    salary = salary + salary * incr_sal
        WHERE  CURRENT OF employee_cur;
        END LOOP;
END;
/
```



```
PL/SQL procedure successfully completed.

SQL> select * from employee_015
  2  ;

EMPLOYEE_ID FIRST_NAME                LAST_NAME
----------- ------------------------- -------------------------
EMAIL                     PHONE_NUMBER HIRE_DATE  JOB_ID
------------------------- ------------ ---------- -------------------------
    SALARY COMMISSION_PCT MANAGER_ID DEPARTMENT_ID
---------- -------------- ---------- -------------
        100 Steven                    King
SKING                     5151234567 2003-06-17 AD_PRES
     26400              0          0            90

        101 Neena                     Kochar
NKOCHAR                   5151234568 2005-09-21 AD_VP
     17000              0        100            90
```

8)

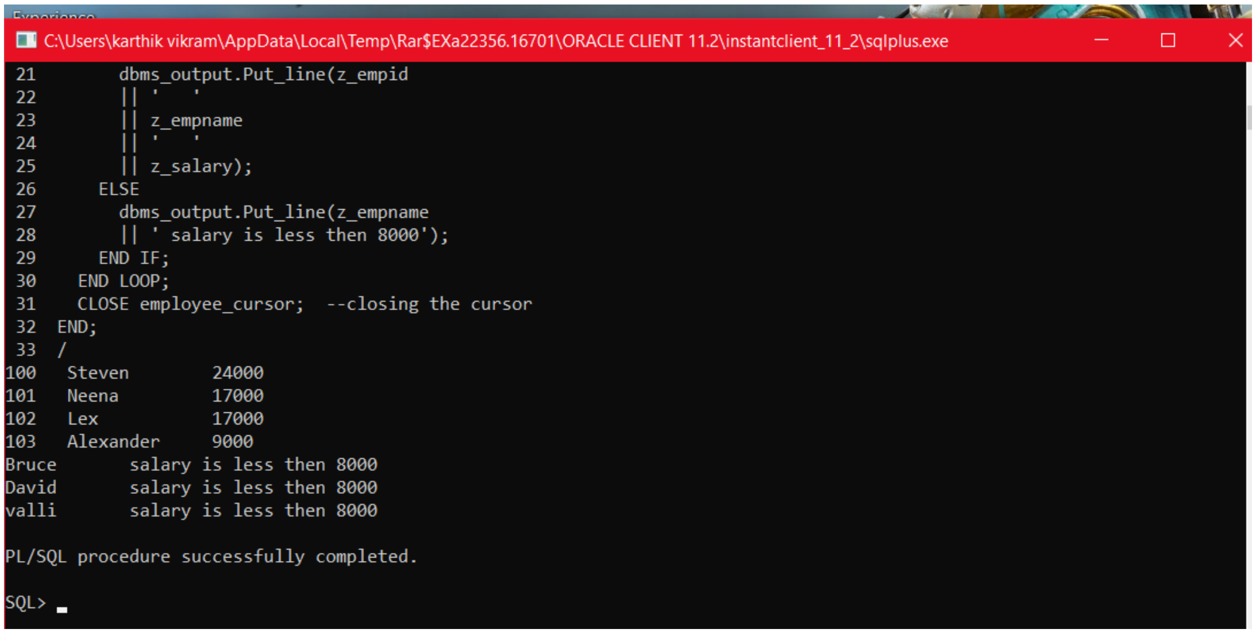
```
create or replace function eqn
return number
is
Q number;
C number(3):=50;
H number(3):=30;
D number(3) :=10;
begin
Q := sqrt((2 * C * D)/H);
return Q;
end;
/

select eqn() from dual;
```



```
SQL> create or replace function eqn
return number
is
Q number;
C number(3):=50;
H number(3):=30;
D number(3) :=10;
begin
Q := sqrt((2 * C * D)/H);
return Q;
end;
/

Function created.

SQL>
SQL> select eqn() from dual;

     EQN()
----------
5.77350269
```

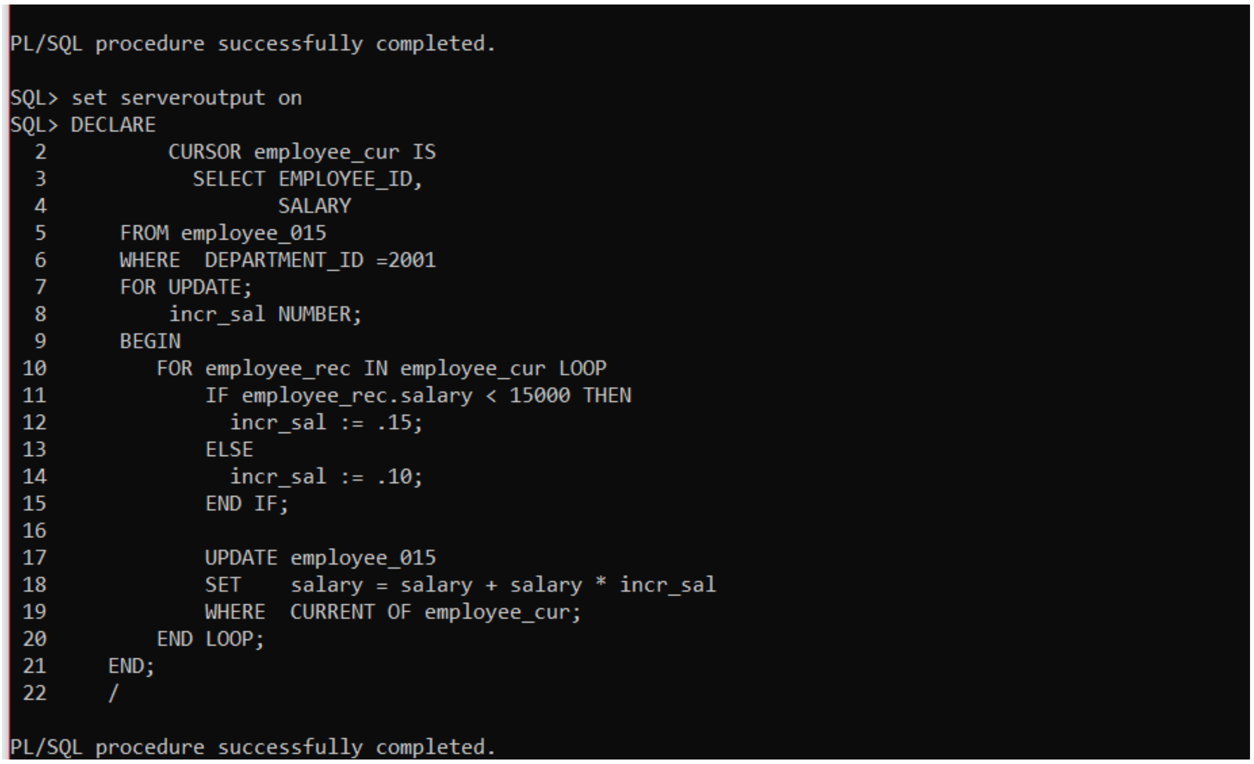
Week 12 PL/SQL

```
1)DECLARE
  z_empid employee_015.EMPLOYEE_ID%TYPE;
  z_empname employee_015.FIRST_NAME%TYPE;
  z_salary employee_015.SALARY%TYPE;
  CURSOR employee_cursor IS  -- declaring a cursor
    SELECT EMPLOYEE_ID,
           FIRST_NAME,
           SALARY
    FROM   employee_015;

BEGIN
  OPEN employee_cursor;    -- opening the cursor
  LOOP
    FETCH employee_cursor  -- fetching records from the cursor
    INTO  z_empid,
          z_empname,
          z_salary;
    EXIT
  WHEN employee_cursor%NOTFOUND;
    IF (z_salary > 8000) THEN
      dbms_output.Put_line(z_empid
      || '   '
      || z_empname
      || '   '
      || z_salary);
    ELSE
      dbms_output.Put_line(z_empname
      || ' salary is less then 8000');
    END IF;
  END LOOP;
  CLOSE employee_cursor;  --closing the cursor
END;
/
```


C:\Users\karthik vikram\AppData\Local\Temp\Rar$EXa22356.16701\ORACLE CLIENT 11.2\instantclient_11_2\sqlplus.exe

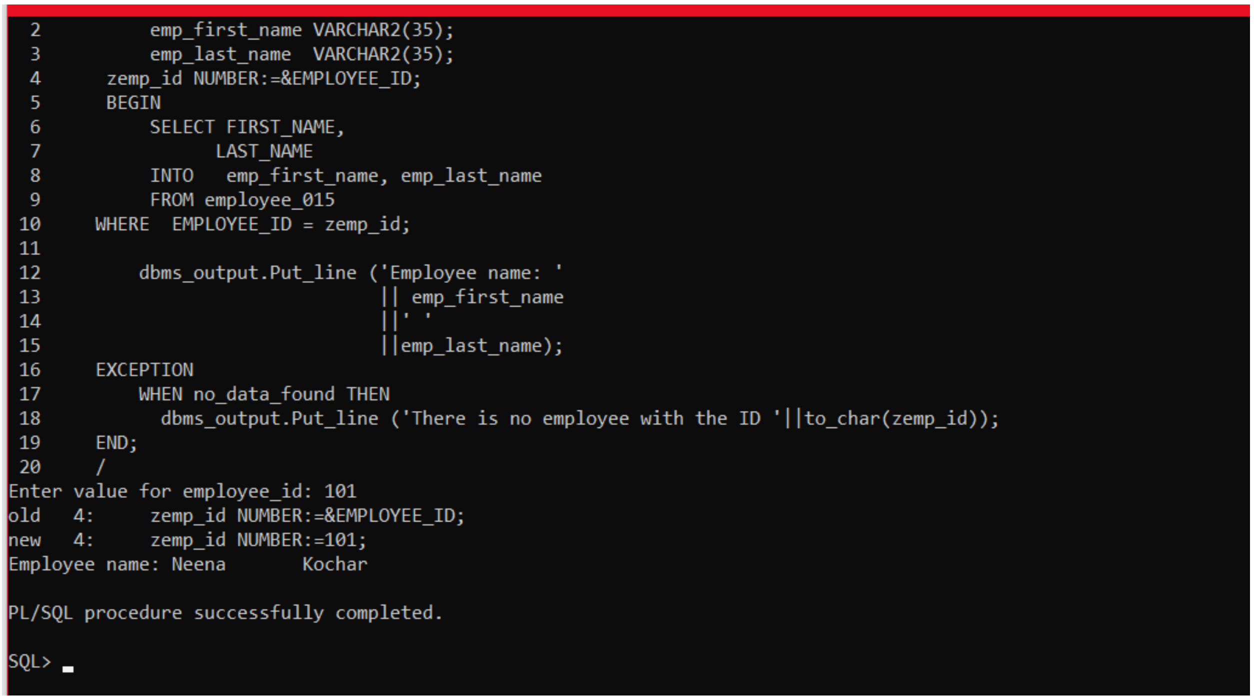
```
      dbms_output.Put_line(z_empid
      || '   '
      || z_empname
      || '   '
      || z_salary);
    ELSE
      dbms_output.Put_line(z_empname
      || ' salary is less then 8000');
    END IF;
  END LOOP;
  CLOSE employee_cursor;  --closing the cursor
END;
/
100   Steven        24000
101   Neena         17000
102   Lex           17000
103   Alexander     9000
Bruce       salary is less then 8000
David       salary is less then 8000
valli       salary is less then 8000

PL/SQL procedure successfully completed.

SQL>
```

```
2)DECLARE
         CURSOR employee_cur IS
         SELECT EMPLOYEE_ID,
                  SALARY
         FROM employee_015
         WHERE  DEPARTMENT_ID =2001
         FOR UPDATE;
         incr_sal NUMBER;
         BEGIN
         FOR employee_rec IN employee_cur LOOP
         IF employee_rec.salary < 15000 THEN
         incr_sal := .15;
         ELSE
         incr_sal := .10;
         END IF;

         UPDATE employee_015
         SET     salary = salary + salary * incr_sal
         WHERE  CURRENT OF employee_cur;
         END LOOP;
  END;
  /
```



```
PL/SQL procedure successfully completed.

SQL> set serveroutput on
SQL> DECLARE
           CURSOR employee_cur IS
             SELECT EMPLOYEE_ID,
                    SALARY
       FROM employee_015
       WHERE  DEPARTMENT_ID =2001
       FOR UPDATE;
           incr_sal NUMBER;
       BEGIN
          FOR employee_rec IN employee_cur LOOP
              IF employee_rec.salary < 15000 THEN
                incr_sal := .15;
              ELSE
                incr_sal := .10;
              END IF;

              UPDATE employee_015
              SET    salary = salary + salary * incr_sal
              WHERE  CURRENT OF employee_cur;
          END LOOP;
      END;
      /

PL/SQL procedure successfully completed.
```

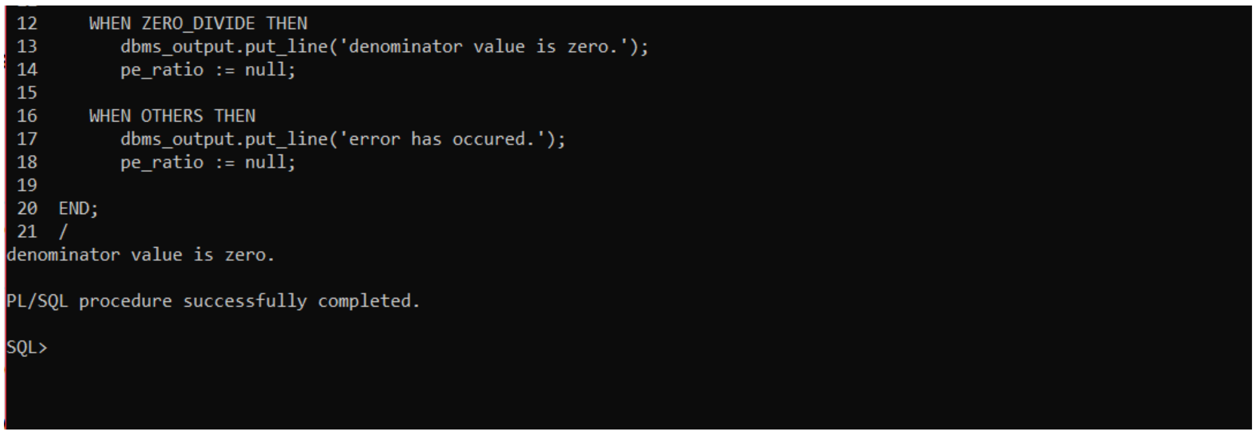
```
3)DECLARE
        emp_first_name VARCHAR2(35);
        emp_last_name  VARCHAR2(35);
        zemp_id NUMBER:=&EMPLOYEE_ID;
        BEGIN
        SELECT FIRST_NAME,
        LAST_NAME
        INTO   emp_first_name, emp_last_name
        FROM employee_015
  WHERE  EMPLOYEE_ID = zemp_id;

        dbms_output.Put_line ('Employee name: '
                        || emp_first_name
                        ||' '
                        ||emp_last_name);
  EXCEPTION
        WHEN no_data_found THEN
        dbms_output.Put_line ('There is no employee with the ID '||to_char(zemp_id));
  END;
  /
```



```
  2         emp_first_name VARCHAR2(35);
  3         emp_last_name  VARCHAR2(35);
  4     zemp_id NUMBER:=&EMPLOYEE_ID;
  5     BEGIN
  6         SELECT FIRST_NAME,
  7               LAST_NAME
  8         INTO   emp_first_name, emp_last_name
  9         FROM employee_015
 10    WHERE  EMPLOYEE_ID = zemp_id;
 11
 12        dbms_output.Put_line ('Employee name: '
 13                              || emp_first_name
 14                              ||' '
 15                              ||emp_last_name);
 16    EXCEPTION
 17        WHEN no_data_found THEN
 18          dbms_output.Put_line ('There is no employee with the ID '||to_char(zemp_id));
 19    END;
 20    /
Enter value for employee_id: 101
old   4:     zemp_id NUMBER:=&EMPLOYEE_ID;
new   4:     zemp_id NUMBER:=101;
Employee name: Neena       Kochar

PL/SQL procedure successfully completed.

SQL>
```

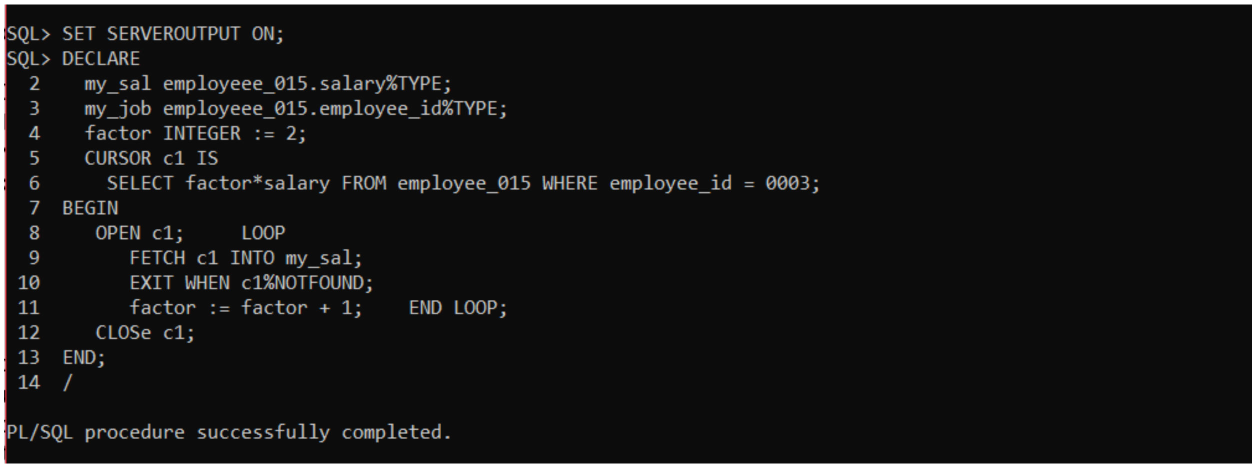
1)SET SERVEROUTPUT ON;

```
DECLARE
  num NUMBER := 9.73;
  den NUMBER := 0;
  pe_ratio NUMBER;
BEGIN

  pe_ratio := num / den;
  dbms_output.put_line('num/den ratio = ' || pe_ratio);

EXCEPTION

  WHEN ZERO_DIVIDE THEN
        dbms_output.put_line('denominator value is zero.');
        pe_ratio := null;

  WHEN OTHERS THEN
        dbms_output.put_line('error has occured.');
        pe_ratio := null;

END;
/
```



```
    WHEN ZERO_DIVIDE THEN
       dbms_output.put_line('denominator value is zero.');
       pe_ratio := null;

    WHEN OTHERS THEN
       dbms_output.put_line('error has occured.');
       pe_ratio := null;

 END;
 /
denominator value is zero.

PL/SQL procedure successfully completed.

SQL>
```

```
2)SET SERVEROUTPUT ON;
DECLARE
  my_sal employeee_015.salary%TYPE;
  my_job employeee_015.employee_id%TYPE;
  factor INTEGER := 2;
  CURSOR c1 IS
          SELECT factor*salary FROM employee_015 WHERE employee_id = 0003;
BEGIN
   OPEN c1;     LOOP
          FETCH c1 INTO my_sal;
          EXIT WHEN c1%NOTFOUND;
          factor := factor + 1;    END LOOP;
   CLOSe c1;
END;
/
```



```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
    my_sal employeee_015.salary%TYPE;
    my_job employeee_015.employee_id%TYPE;
    factor INTEGER := 2;
    CURSOR c1 IS
      SELECT factor*salary FROM employee_015 WHERE employee_id = 0003;
  BEGIN
     OPEN c1;     LOOP
        FETCH c1 INTO my_sal;
        EXIT WHEN c1%NOTFOUND;
        factor := factor + 1;    END LOOP;
     CLOSe c1;
  END;
  /

PL/SQL procedure successfully completed.
```

3) Write a PL/SQL block to select the name of the employee with a given salary value. Use the DEFINE command to provide the salary. Pass the value to the PL/SQL block through a iSQL*Plus substitution variable. If the salary entered returns more than one row, handle the exception with an appropriate exception handler and insert into the MESSAGES table the message “More than one employee with a salary of .”

a. Test this with input as 3000

b. If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert into the MESSAGES table the message “No employee with a salary of .”

c. If the salary entered returns only one row, insert into the MESSAGES table the employee’s name and the salary amount.

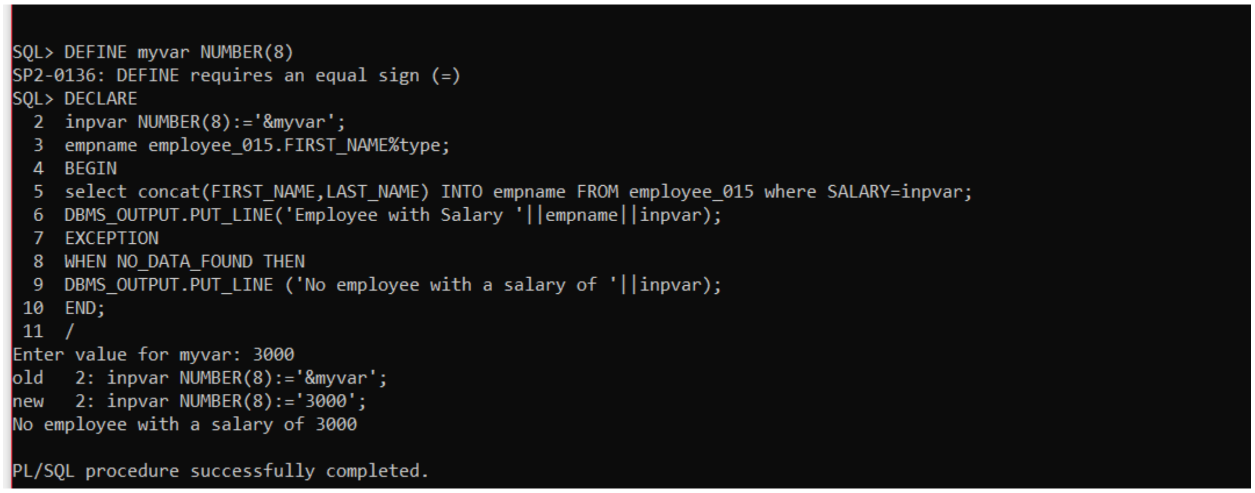
d. Handle any other exception with an appropriate exception handler and insert into the MESSAGES table the message “Some other error occurred.”

e. Test the block for a variety of test cases. Display the rows from the MESSAGES table to check whether the PL/SQL block has executed successfully.

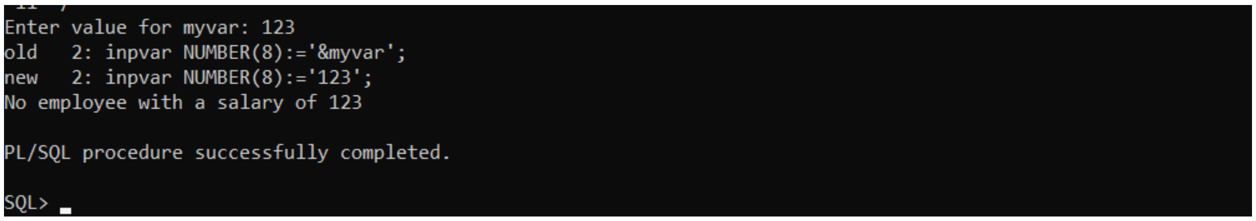
f. Test with random salary such as 123 then with 3000.There are two records with salary 3000 but since we are not handling it.

a)

```
SET SERVEROUTPUT ON
DEFINE myvar NUMBER(8)
DECLARE
inpvar NUMBER(8):='&myvar';
empname employee_015.FIRST_NAME%type;
BEGIN
select concat(FIRST_NAME,LAST_NAME) INTO empname FROM employee_015 where
SALARY=inpvar;
DBMS_OUTPUT.PUT_LINE('Employee with Salary '||empname||inpvar);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE ('No employee with a salary of '||inpvar);
END;
/
```

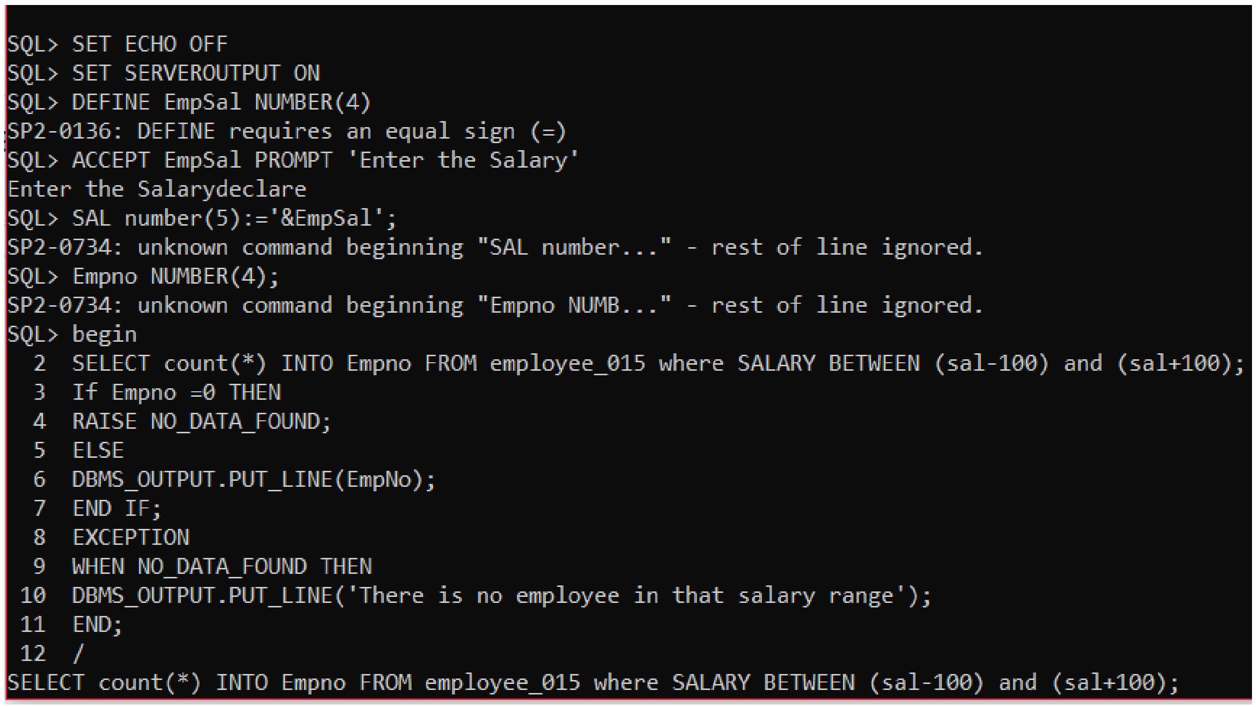


```
SQL> DEFINE myvar NUMBER(8)
SP2-0136: DEFINE requires an equal sign (=)
SQL> DECLARE
  2  inpvar NUMBER(8):='&myvar';
  3  empname employee_015.FIRST_NAME%type;
  4  BEGIN
  5  select concat(FIRST_NAME,LAST_NAME) INTO empname FROM employee_015 where SALARY=inpvar;
  6  DBMS_OUTPUT.PUT_LINE('Employee with Salary '||empname||inpvar);
  7  EXCEPTION
  8  WHEN NO_DATA_FOUND THEN
  9  DBMS_OUTPUT.PUT_LINE ('No employee with a salary of '||inpvar);
 10  END;
 11  /
Enter value for myvar: 3000
old   2: inpvar NUMBER(8):='&myvar';
new   2: inpvar NUMBER(8):='3000';
No employee with a salary of 3000

PL/SQL procedure successfully completed.
```



```
Enter value for myvar: 123
old   2: inpvar NUMBER(8):='&myvar';
new   2: inpvar NUMBER(8):='123';
No employee with a salary of 123

PL/SQL procedure successfully completed.

SQL>
```

```
4)SET SERVEROUTPUT ON
DEFINE myvar1 NUMBER(8)
DEFINE myvar2 VARCHAR(15)
ACCEPT myvar1 PROMPT 'Enter department no'
ACCEPT myvar2 PROMPT 'Enter department Location'
DECLARE
inpvar1 NUMBER(8):='&myvar1';
inpvar2 VARCHAR(15):='&myvar2';
empname employee_015.FIRST_NAME%type;
TestDisplay Varchar2(50) :='Department Doesnot exist';
BEGIN
select concat(FIRST_NAME,LAST_NAME) INTO empname FROM employee_015 where
DEPARTMENT_ID=inpvar1;
DBMS_OUTPUT.PUT_LINE('Employee with Salary '||empname||inpvar1);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE (TestDisplay);
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE ('Some other exception occurred');
END;
/
```



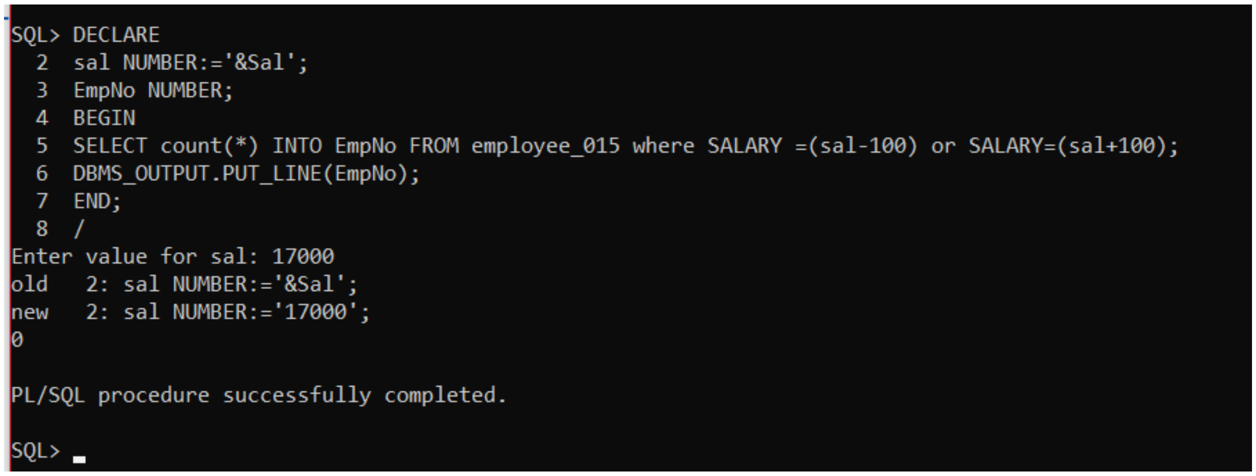
```
SQL> SET ECHO OFF
SQL> SET SERVEROUTPUT ON
SQL> DEFINE EmpSal NUMBER(4)
SP2-0136: DEFINE requires an equal sign (=)
SQL> ACCEPT EmpSal PROMPT 'Enter the Salary'
Enter the Salarydeclare
SQL> SAL number(5):='&EmpSal';
SP2-0734: unknown command beginning "SAL number..." - rest of line ignored.
SQL> Empno NUMBER(4);
SP2-0734: unknown command beginning "Empno NUMB..." - rest of line ignored.
SQL> begin
  2  SELECT count(*) INTO Empno FROM employee_015 where SALARY BETWEEN (sal-100) and (sal+100);
  3  If Empno =0 THEN
  4  RAISE NO_DATA_FOUND;
  5  ELSE
  6  DBMS_OUTPUT.PUT_LINE(EmpNo);
  7  END IF;
  8  EXCEPTION
  9  WHEN NO_DATA_FOUND THEN
 10  DBMS_OUTPUT.PUT_LINE('There is no employee in that salary range');
 11  END;
 12  /
SELECT count(*) INTO Empno FROM employee_015 where SALARY BETWEEN (sal-100) and (sal+100);
```

5). Write a PL/SQL block that prints the number of employees who earn plus or minus $100 of the salary value set for an iSQL*Plus substitution variable.

Use the DEFINE command to provide the salary value. Pass the value to the PL/SQL block through a iSQL*Plus substitution variable.

a. If there is no employee within that salary range, print a message to the user indicating that is the case. Use an exception for this case.

```
SET SERVEROUTPUT ON
DEFINE EmpSal NUMBER(4)
DECLARE
sal NUMBER:='&Sal';
EmpNo NUMBER;
BEGIN
SELECT count(*) INTO EmpNo FROM employee_015 where SALARY =(sal-100) or
SALARY=(sal+100);
DBMS_OUTPUT.PUT_LINE(EmpNo);
END;
/
```

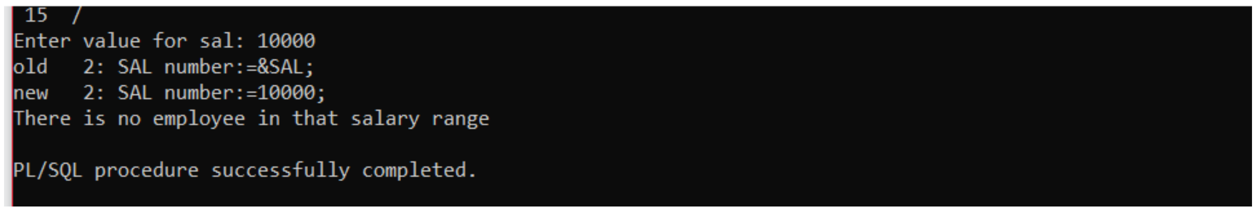


```
SQL> DECLARE
sal NUMBER:='&Sal';
EmpNo NUMBER;
BEGIN
SELECT count(*) INTO EmpNo FROM employee_015 where SALARY =(sal-100) or SALARY=(sal+100);
DBMS_OUTPUT.PUT_LINE(EmpNo);
END;
/
Enter value for sal: 17000
old   2: sal NUMBER:='&Sal';
new   2: sal NUMBER:='17000';
0

PL/SQL procedure successfully completed.

SQL>
```

b. If there are one or more employees within that range, the message should indicate how many employees are in that salary range.

```
SET ECHO OFF
SET SERVEROUTPUT ON
declare
SAL number:=&SAL;
Empno NUMBER;
begin
```

SELECT count(*) INTO Empno FROM employee_015 where SALARY BETWEEN (sal-100) and (sal+100);
If Empno =0 THEN
RAISE NO_DATA_FOUND;
ELSE
DBMS_OUTPUT.PUT_LINE(EmpNo);
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('There is no employee in that salary range');
END;
/

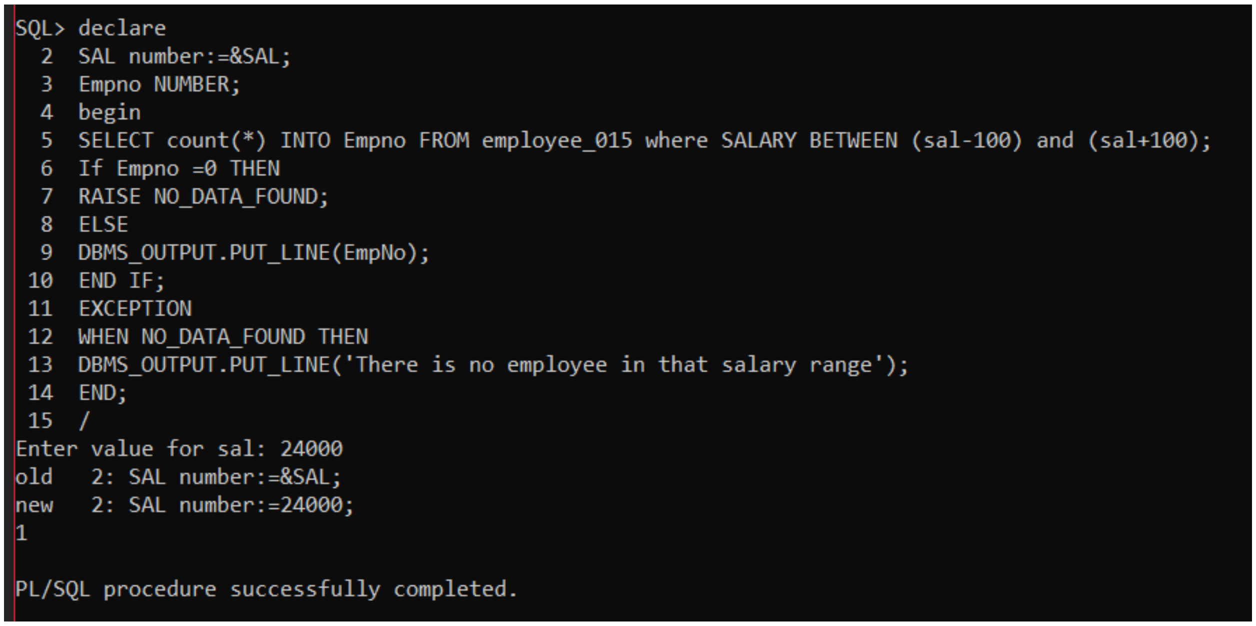


```
 15  /
Enter value for sal: 10000
old   2: SAL number:=&SAL;
new   2: SAL number:=10000;
There is no employee in that salary range

PL/SQL procedure successfully completed.
```

c. Handle any other exception with an appropriate exception handler. The message should indicate that some other error occurred.

SET ECHO OFF
SET SERVEROUTPUT ON
DEFINE EmpSal NUMBER(4)
ACCEPT EmpSal PROMPT 'Enter the Salary'
declare
NO_DATA_FOUNDING EXCEPTION;
sal NUMBER(5):='&EmpSal';
Empno NUMBER(3);
begin
SELECT count(*) INTO Empno FROM employee_015 where SALARY BETWEEN (sal-100) and (sal+100);
if EMPNO =0 then
RAISE NO_DATA_FOUNDING;
ELSE
DBMS_OUTPUT.PUT_LINE(EmpNo||' Employees are in the salary range');
END IF;
EXCEPTION
WHEN NO_DATA_FOUNDING THEN
DBMS_OUTPUT.PUT_LINE('There is no employee in that salary range');
WHEN OTHERS THEN

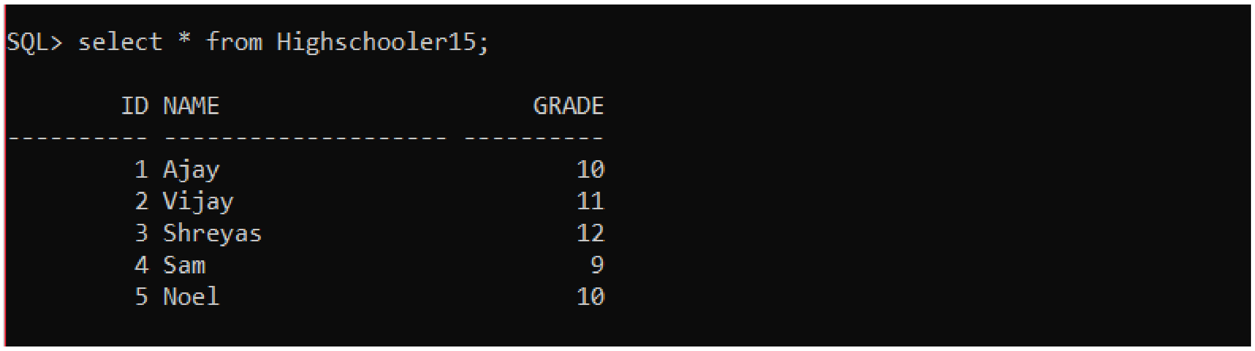
DBMS_OUTPUT.PUT_LINE('SOME OTHER ERRORS');
END;



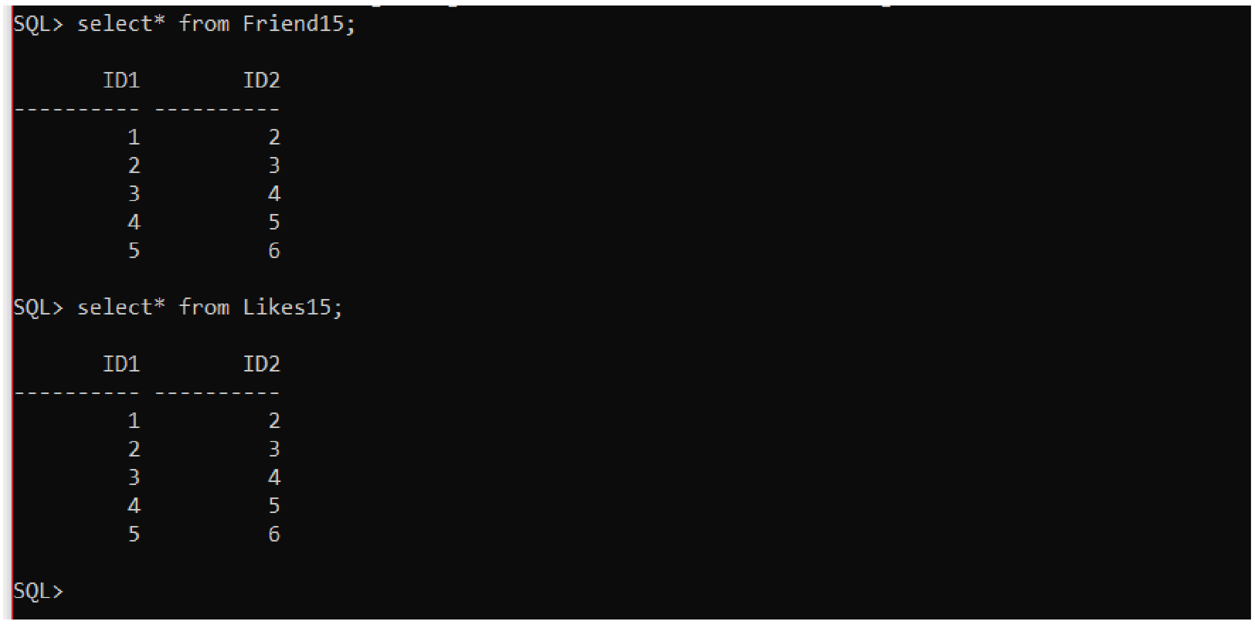
```
SQL> declare
SAL number:=&SAL;
Empno NUMBER;
begin
SELECT count(*) INTO Empno FROM employee_015 where SALARY BETWEEN (sal-100) and (sal+100);
If Empno =0 THEN
RAISE NO_DATA_FOUND;
ELSE
DBMS_OUTPUT.PUT_LINE(EmpNo);
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('There is no employee in that salary range');
END;
/
Enter value for sal: 24000
old   2: SAL number:=&SAL;
new   2: SAL number:=24000;
1

PL/SQL procedure successfully completed.
```

Week - 14
RA1811029010015
M.karthik vikram

```
1)create table Highschooler15(ID int, name varchar(20), grade int)

insert into Highschooler15 values(01,'Ajay',10);

insert into Highschooler15 values(02,'Vijay',11);

insert into Highschooler15 values(03,'Shreyas',12);

insert into Highschooler15 values(04,'Sam',9);

insert into Highschooler15 values(05,'Noel',10);

create table Friend15(ID1 int, ID2 int);

insert into Friend15 values(1, 2);

insert into Friend15 values(2, 3);

insert into Friend15 values(3, 4);

insert into Friend15 values(4, 5);

insert into Friend15 values(5, 6);
```



```
SQL> select * from Highschooler15;

        ID NAME                      GRADE
---------- -------------------- ----------
         1 Ajay                         10
         2 Vijay                        11
         3 Shreyas                      12
         4 Sam                           9
         5 Noel                         10
```



```
SQL> select* from Friend15;

       ID1        ID2
---------- ----------
         1          2
         2          3
         3          4
         4          5
         5          6

SQL> select* from Likes15;

       ID1        ID2
---------- ----------
         1          2
         2          3
         3          4
         4          5
         5          6

SQL>
```

create table Likes15(ID1 int, ID2 int);

insert into Likes15 values(1, 2);

insert into Likes15 values(2, 3);

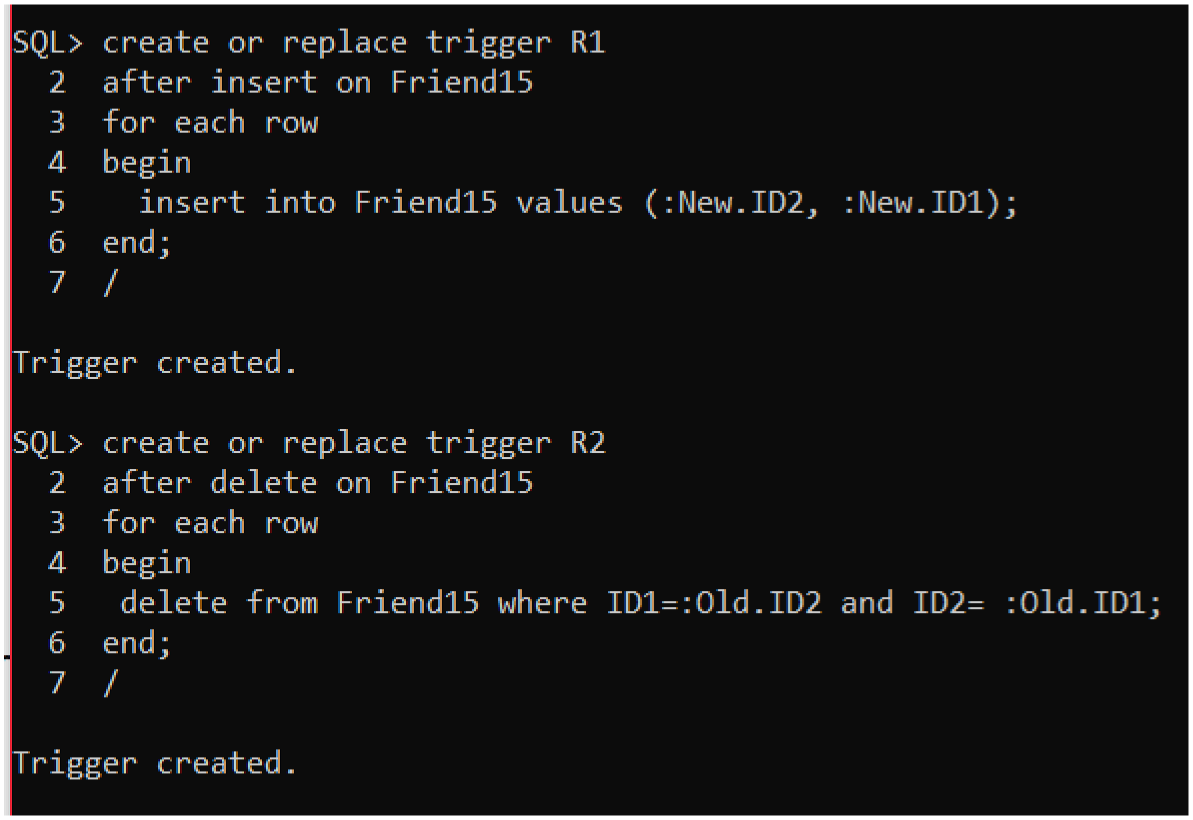
insert into Likes15 values(3, 4);

insert into Likes15 values(4, 5);

insert into Likes15 values(5, 6);

create or replace trigger R1

```
after insert on Friend15

for each row

begin

  insert into Friend15 values (:New.ID2, :New.ID1);

End;

/


create or replace trigger R2

after delete on Friend15

for each row

begin

 delete from Friend15 where ID1=:Old.ID2 and ID2= :Old.ID1;

End;

/
```



```
SQL> create or replace trigger R1
  2  after insert on Friend15
  3  for each row
  4  begin
  5    insert into Friend15 values (:New.ID2, :New.ID1);
  6  end;
  7  /

Trigger created.

SQL> create or replace trigger R2
  2  after delete on Friend15
  3  for each row
  4  begin
  5   delete from Friend15 where ID1=:Old.ID2 and ID2= :Old.ID1;
  6  end;
  7  /

Trigger created.
```

create or replace trigger R3

after update of grade on Highschooler15

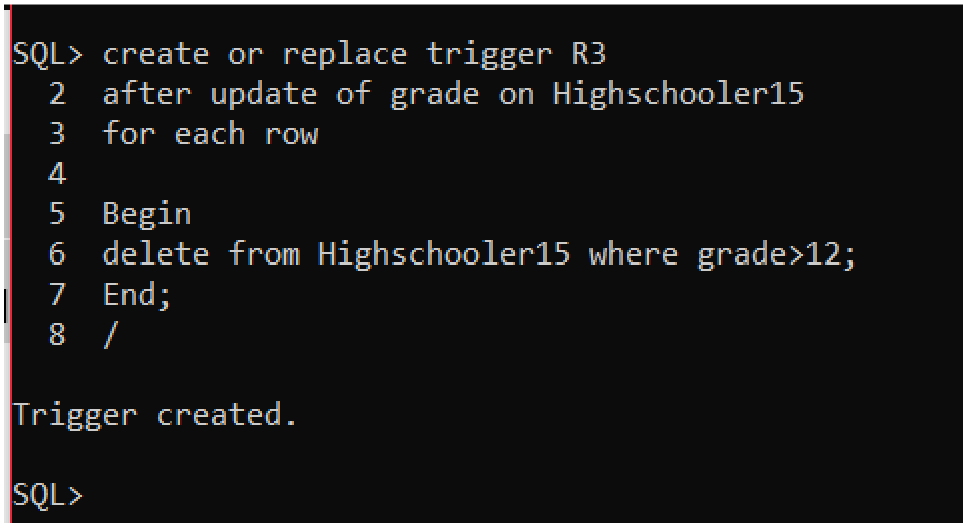
for each row

Begin

delete from Highschooler15 where grade>12;

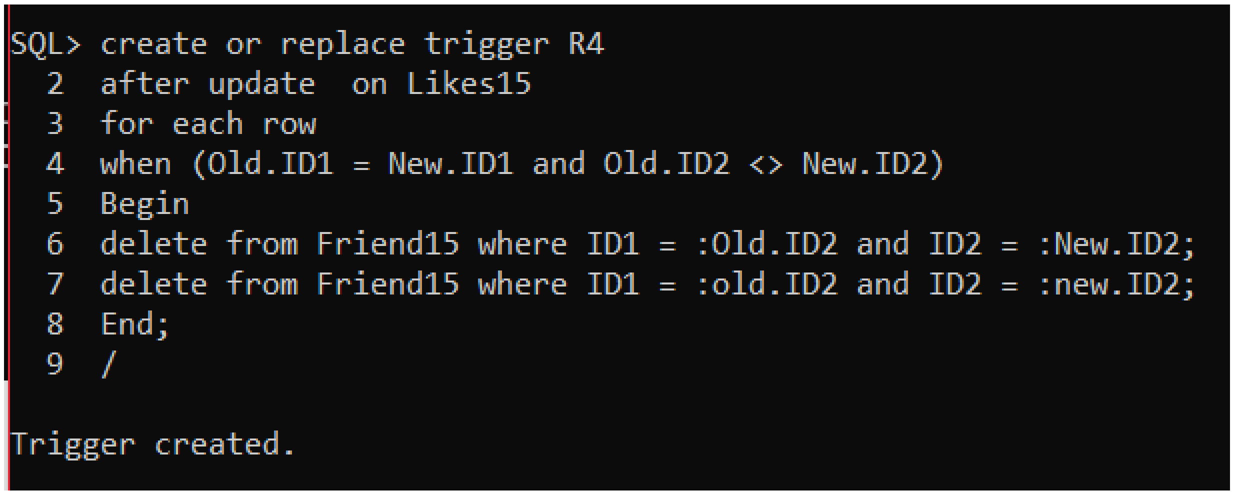
End;

/



```
SQL> create or replace trigger R3
  2  after update of grade on Highschooler15
  3  for each row
  4
  5  Begin
  6  delete from Highschooler15 where grade>12;
  7  End;
  8  /

Trigger created.

SQL>
```

```
create or replace trigger R4

after update  on Likes15

for each row

when (Old.ID1 = New.ID1 and Old.ID2 <> New.ID2)

Begin

delete from Friend15 where ID1 = :Old.ID2 and ID2 = :New.ID2;

delete from Friend15 where ID1 = :old.ID2 and ID2 = :new.ID2;

End;

/
```



```
SQL> create or replace trigger R4
  2  after update  on Likes15
  3  for each row
  4  when (Old.ID1 = New.ID1 and Old.ID2 <> New.ID2)
  5  Begin
  6  delete from Friend15 where ID1 = :Old.ID2 and ID2 = :New.ID2;
  7  delete from Friend15 where ID1 = :old.ID2 and ID2 = :new.ID2;
  8  End;
  9  /

Trigger created.
```

5)V. Consider the following relational schema for the european volleyball tournament: PLAYER (PlayerId, Name, Team, Height, Birthday, PlayedMatches) TEAM (Team, Coach, WonGames) MATCH (MatchId, Date, Team1, Team2, WonSetsTeam1, WonSetsTeam2, Referee) PLAYED (MatchId, PlayerId, Role, ScoredPoints)

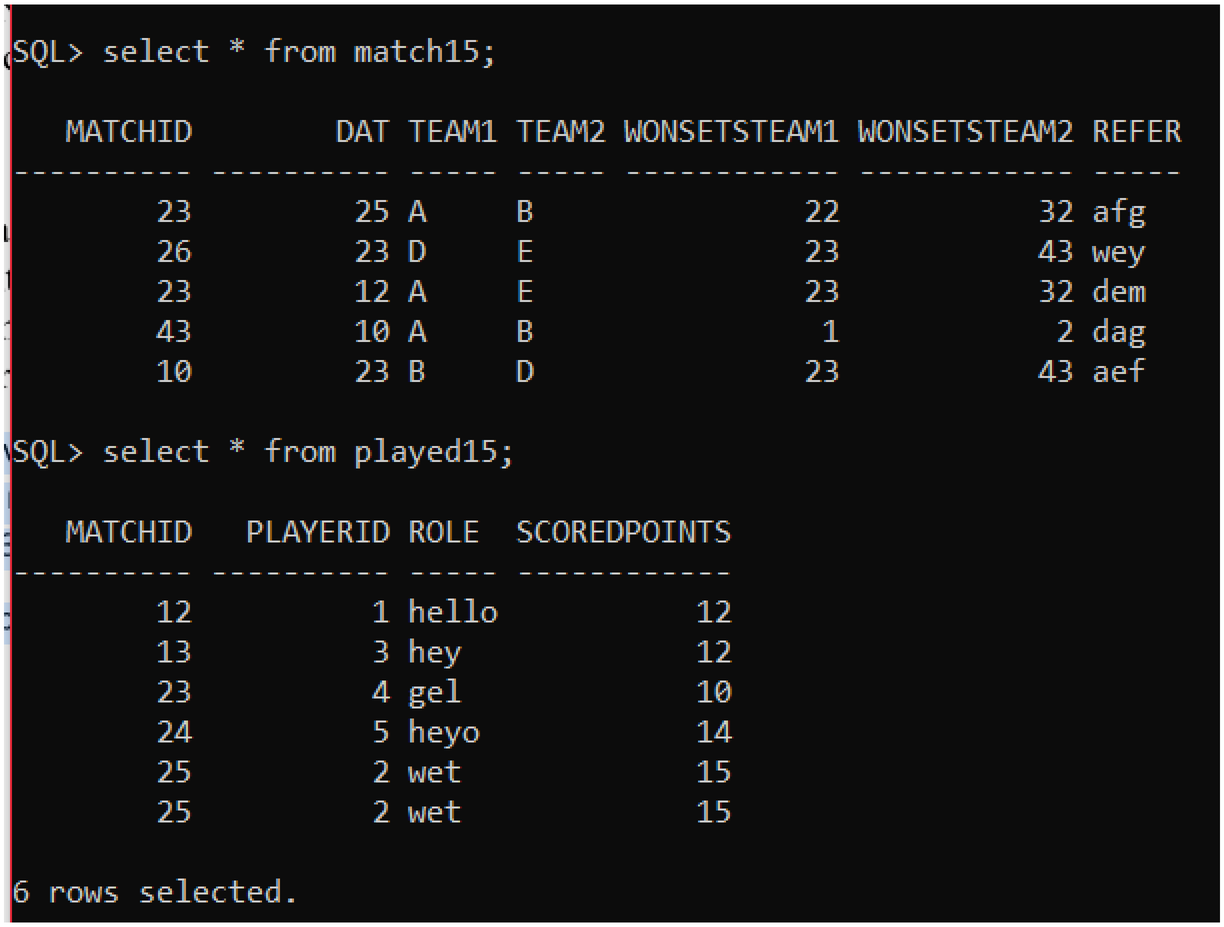
create table player15(playerid number(5), name varchar(5), team varchar2(5), height number(5), birthday number, playedmatches number(5));
INSERT INTO player15(playerid, name, team, height, birthday, playedmatches)
VALUES(1, 'abc', 'A', 5, 25, 13);

INSERT INTO player15(playerid, name, team, height, birthday, playedmatches)
VALUES(2, 'def', 'B', 6, 26, 12);

INSERT INTO player15(playerid, name, team, height, birthday, playedmatches)
VALUES(3, 'ghi', 'C', 7, 23, 11);

INSERT INTO player15(playerid, name, team, height, birthday, playedmatches)
VALUES(4, 'jkl', 'D', 4, 23, 5);

INSERT INTO player15(playerid, name, team, height, birthday, playedmatches)
VALUES(5, 'mno', 'E', 6, 20, 6);
create table match15(matchid number(5), dat number, team1 varchar2(5), team2 varchar2(5), wonsetsteam1 number(5), wonsetsteam2 number(5), referee varchar2(5));



```
SQL> select * from match15;

   MATCHID        DAT TEAM1 TEAM2 WONSETSTEAM1 WONSETSTEAM2 REFER
---------- ---------- ----- ----- ------------ ------------ -----
        23         25 A     B               22           32 afg
        26         23 D     E               23           43 wey
        23         12 A     E               23           32 dem
        43         10 A     B                1            2 dag
        10         23 B     D               23           43 aef

SQL> select * from played15;

   MATCHID   PLAYERID ROLE  SCOREDPOINTS
---------- ---------- ----- ------------
        12          1 hello           12
        13          3 hey             12
        23          4 gel             10
        24          5 heyo            14
        25          2 wet             15
        25          2 wet             15

6 rows selected.
```

INSERT INTO match15 VALUES(23,25,'A','B',22,32,'afg');
INSERT INTO match15 VALUES(26,23,'D','E',23,43,'wey');
INSERT INTO match15 VALUES(23,12,'A','E',23,32,'dem');
INSERT INTO match15 VALUES(43,10,'A','B',1,2,'dag');
INSERT INTO match15 VALUES(10,23,'B','D',23,43,'aef');

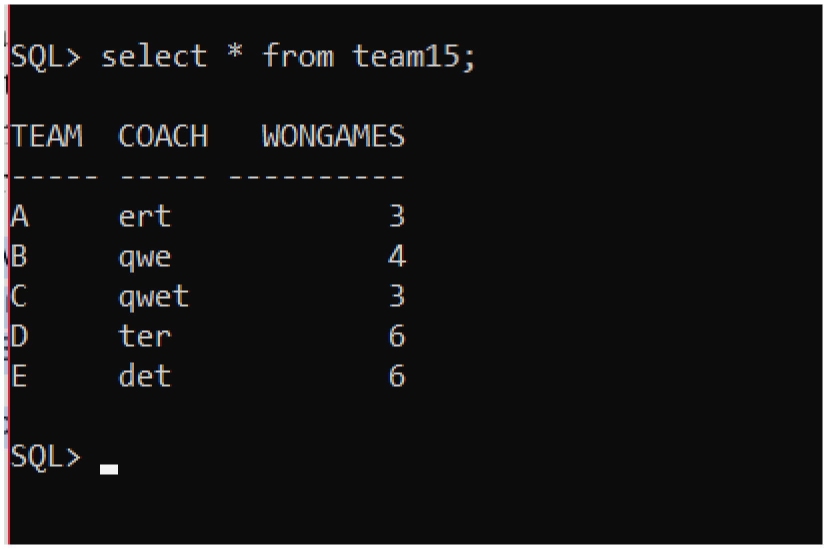
create table team15(team varchar2(5),coach varchar2(5), wongames number(5));
INSERT INTO team15(team, coach, wongames)
VALUES('A', 'ert', 3);

INSERT INTO team15(team, coach, wongames)
VALUES('B', 'qwe', 4);

INSERT INTO team15(team, coach, wongames)
VALUES
('C', 'qwet', 3);

INSERT INTO team15(team, coach, wongames)
VALUES
('D', 'ter', 6);

INSERT INTO team15(team, coach, wongames)
VALUES
('E', 'det', 6);

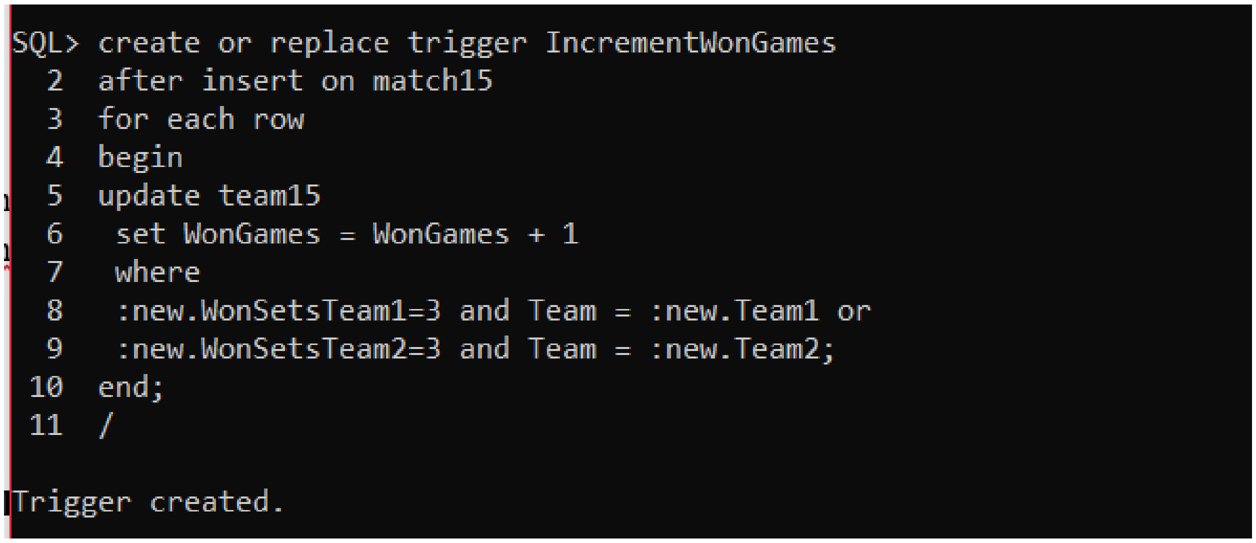


```
SQL> select * from team15;

TEAM  COACH   WONGAMES
----- ----- ----------
A     ert            3
B     qwe            4
C     qwet           3
D     ter            6
E     det            6

SQL>
```

create table played15(matchid number(5), playerid number(5), role varchar2(5), scoredpoints number(5));
INSERT INTO played15 VALUES (12,1,'hello',12);
INSERT INTO played15 VALUES (13,3,'hey',12);
INSERT INTO played15 VALUES (23,4,'gel',10);
INSERT INTO played15 VALUES (24,5,'heyo',14);
INSERT INTO played15 VALUES (25,2,'wet',15);

1. Build a trigger that keeps the value of WonGames after insertions in GAME taking into account that
WonGames is relative to the entire history of the team, not only to the current tournament, and that a
team wins a game when he wins 3 sets.

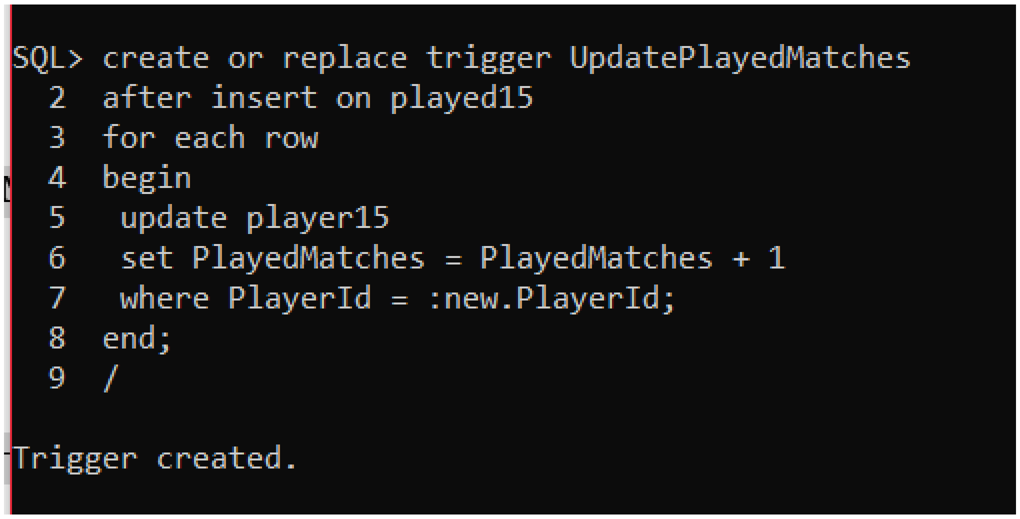
```
create or replace trigger IncrementWonGames
after insert on match15
for each row
begin
update team15
 set WonGames = WonGames + 1
 where
 :new.WonSetsTeam1=3 and Team = :new.Team1 or
 :new.WonSetsTeam2=3 and Team = :new.Team2;
end;
/
```



```
SQL> create or replace trigger IncrementWonGames
  2  after insert on match15
  3  for each row
  4  begin
  5  update team15
  6   set WonGames = WonGames + 1
  7   where
  8   :new.WonSetsTeam1=3 and Team = :new.Team1 or
  9   :new.WonSetsTeam2=3 and Team = :new.Team2;
 10  end;
 11  /

Trigger created.
```

2. Building also a trigger that keeps PlayedMatches of PLAYER updated after insertions in PLAYED

```
create or replace trigger UpdatePlayedMatches
after insert on played15
for each row
begin
 update player15
 set PlayedMatches = PlayedMatches + 1
 where PlayerId = :new.PlayerId;
end;
/
```



```
SQL> create or replace trigger UpdatePlayedMatches
  2  after insert on played15
  3  for each row
  4  begin
  5   update player15
  6   set PlayedMatches = PlayedMatches + 1
  7   where PlayerId = :new.PlayerId;
  8  end;
  9  /

Trigger created.
```