**SECTION-1 : PROGRAMMING FOUNDATION WITH PSEUDOCODE**

1.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROGRAM NAME: WITHDRAWAL FUNCTION FOR ATM SYSTEM

AUTHOR : SRAVYA

DESCRIPTION : To verify the withdraws function of ATM

VERSION :

DATE : 17/01/2017

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BEGIN

DECLARE AccountNumber, CurrentBalance,

WithdrawalAmount, RemainingBalance AS INTEGER

//Variable are declared according the required data types as integer and string

DECLARE CustomerName AS STRING

PRINT “ENTER CUSTOMER NAME:”

ACCEPT CustomerName // the entered customer name

IF ( isCHAR(CustomerName))

//isChar is an inbuilt function to check whether entered value is character or not

THEN

PRINT “ENTER ACCOUNT NUMBER:”

ACCEPT AccountNumber

IF(isValid(AccountNumber))

//isValid Function is redirected to a sub block where it checks if the entered number is valid or not

THEN

PRINT” ENTER CURRENT BALANCE:”

ACCEPT CurrentBalance

// the below block checks if the entered balance is in the acceptable integer format

IF(isValid(CurrentBalance))

PRINT “THE CURRENT BALANCE IS UPDATED.”

ELSE

PRINT”CURRENT BALANCE SHOULD CONTAIN   
 ONLY INTEGERS”

ENDIF

ELSE

// executes when an invalid account number format is forced that contains characters

PRINT”ACCOUNT NUMBER SHOULD CONTAIN INTEGERS ONLY”

ENDIF

ELSE

PRINT “CUSTOMER NAME SHOULD CONTAIN ONLY CHARACTERS”

ENDIF

//sub-block to implement withdrawal function

SUB WITHDRAWAL

// Account number is fetched

getAccountNumber(AccountNumber);

EXCEPTION

// if it doesnot match with existing database then exception is raised

WHEN NOSuchElement THEN

PRINT “ERROR MESSAGE: NO SUCH RECORD FOUND”

END SUB

// sub-block to check for a valid account number that is directed from getAccountNumber

SUB getAccountNumber( AccountNumber)

IF(ElementFound(AccountNumber)) THEN

PRINT “ENTER AMOUNT TO BE WITHDRAWN:”

ACCEPT WithdrawalAmount

IF (CurrentBalance>WithdrawalAmount)

PRINT”YOUR WITHDRAWAL TRANSACTION IS BEING PROCESSED”

RemainingBalance=(CurrentBalance-WithdrawalAmount)

Details(RemainingBal)

ELSE IF(CurrentBalance<WithdrawalAmount)

PRINT “ENTER THE VALID WITHDRAWAL AMOUNT LOWER THAN CURRENT   
 BALANCE,YOU HAVE ENTERED HIGHER AMOUNT”

ELSE

PRINT ”Withdraw Amount should be an integer”

ENDIF

ELSE

RAISE ”ACCOUNT NUMBER DOESNOT EXIST”

END IF

END SUB

// sub-block to print the remaining balance after the withdrawal operation is complete

SUB Details(RemainingBalance)

PRINT “TRANSACTION COMPLETED SUCCESSFULLY”

PRINT CustomerName

PRINT AccountNumber

PRINT RemainingBalance

END SUB

//Sub-block to check if the entered data is in the integer format

SUB isValid(DATA)

IF isDigits(DATA)

THEN RETURN TRUE

ELSE

RETURN FALSE

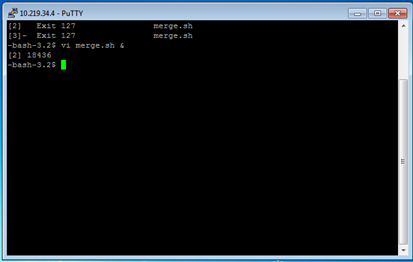
END IF

END SUB

END

**SECTION-2- UNIX**

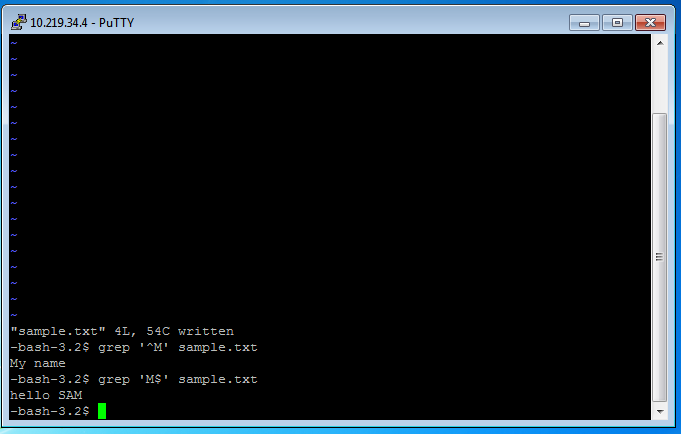
**Q1**

****

To run a process in the background ‘&’ operator is used

The shell script command to run “merge.sh” in the background is : vi merge.sh &

2.



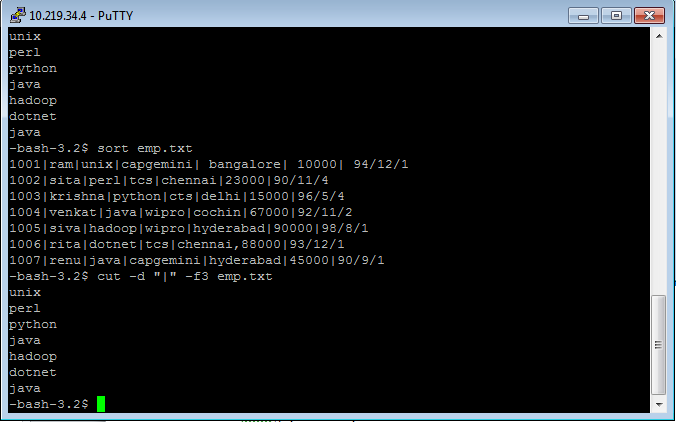
To find out the lines containing ^M characters at the end of every line grep command is used

^M - gives the lines starting with M

M$ - gives the lines ending with character M

Syntax – grep ‘^character\_to\_be\_searched/character\_to\_be\_searched$’ filename.txt

Command used : grep ‘^M’ sample.txt

3.

The command used to sort a pipe de-limited file as per the third column is :

sort emp.txt | cut –d “|” –f3 emp.txt

cut command is used to cut the fields vertically according to the delimiter specified

syntax: cut command\_keyword “delimiter” –field\_number filename.txt

Q2

4. #!/bin/bash

echo "enter username:"

read username

x=`date`

echo "enter any filename :"

read filename

if [ -e $filename ]

then

cat $filename

echo "file created by: $username"

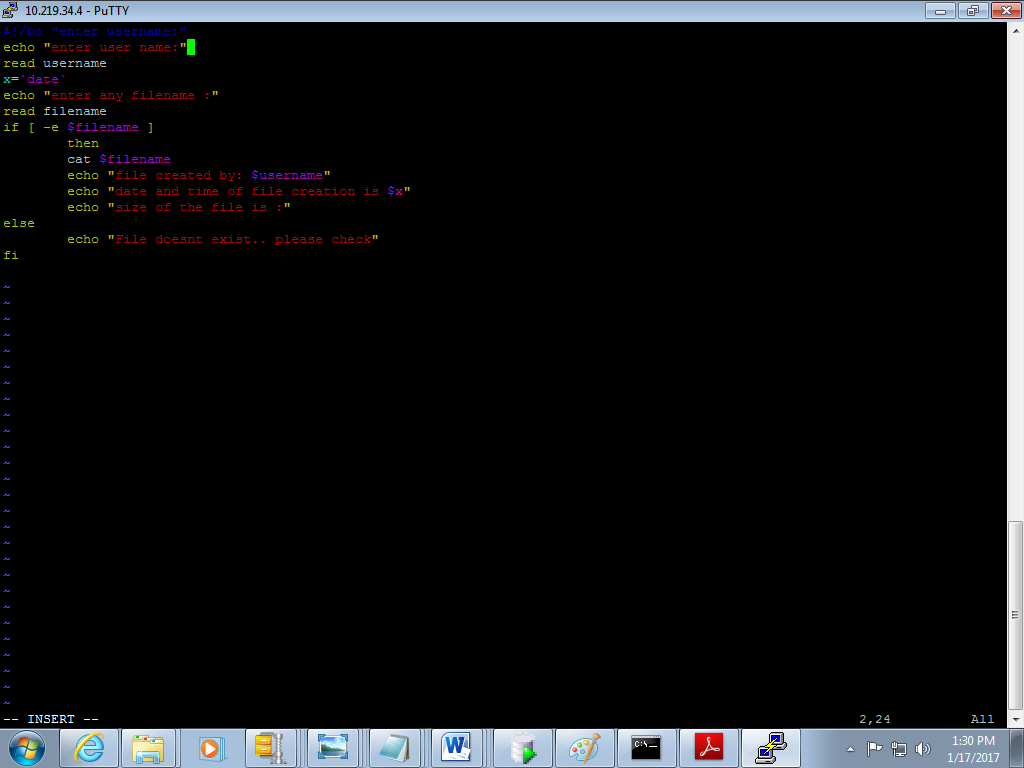
echo "date and time of file creation is $x"

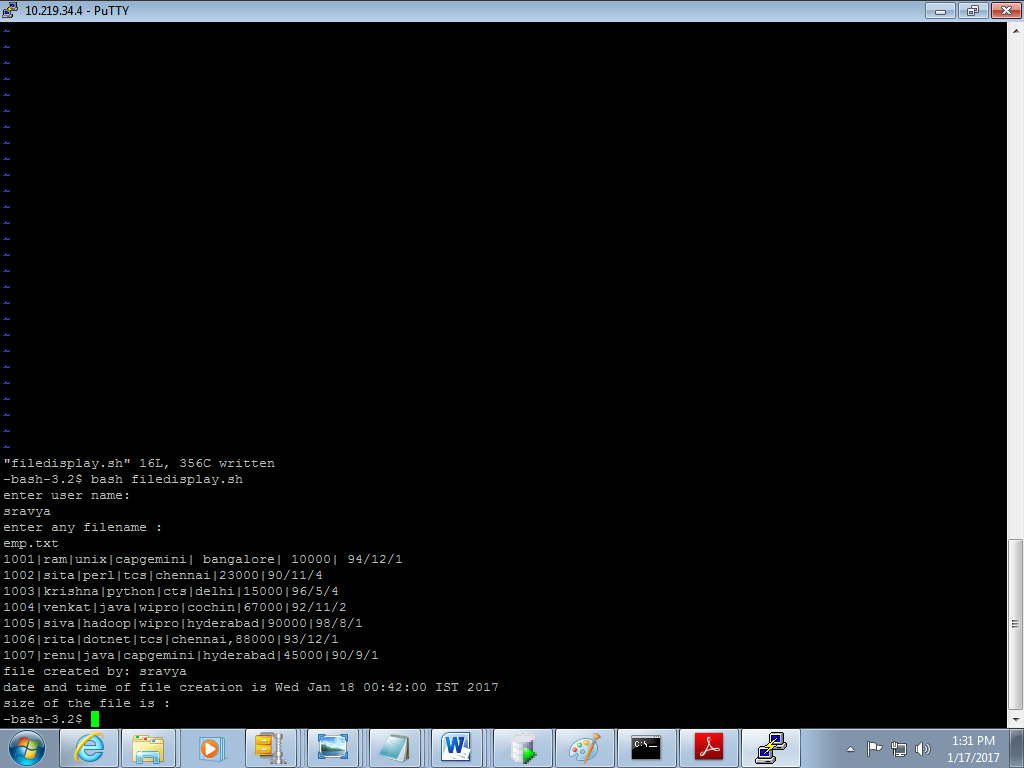
echo "size of the file is :"

else

echo "File doesnt exist.. please check"

fi





**SECTION 3 – ORACLE**

Q1

1.

SELECT s.staff\_name,

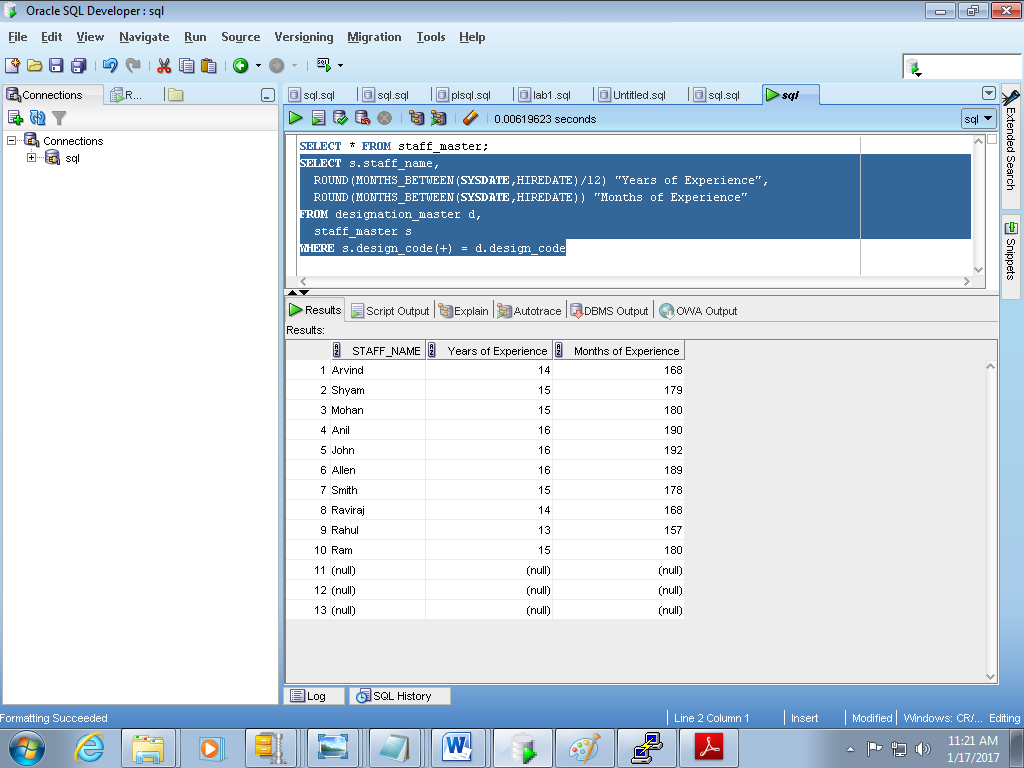
ROUND(MONTHS\_BETWEEN(SYSDATE,HIREDATE)/12) "Years of Experience",

ROUND(MONTHS\_BETWEEN(SYSDATE,HIREDATE)) "Months of Experience"

FROM designation\_master d,

staff\_master s

WHERE s.design\_code(+) = d.design\_code



2.

SELECT staff\_code "manager code",

staff\_name "staff name",

staff\_sal "salary of staff"

FROM staff\_master

WHERE

staff\_code NOT IN

(SELECT m.mgr\_code

FROM staff\_master s, staff\_master m

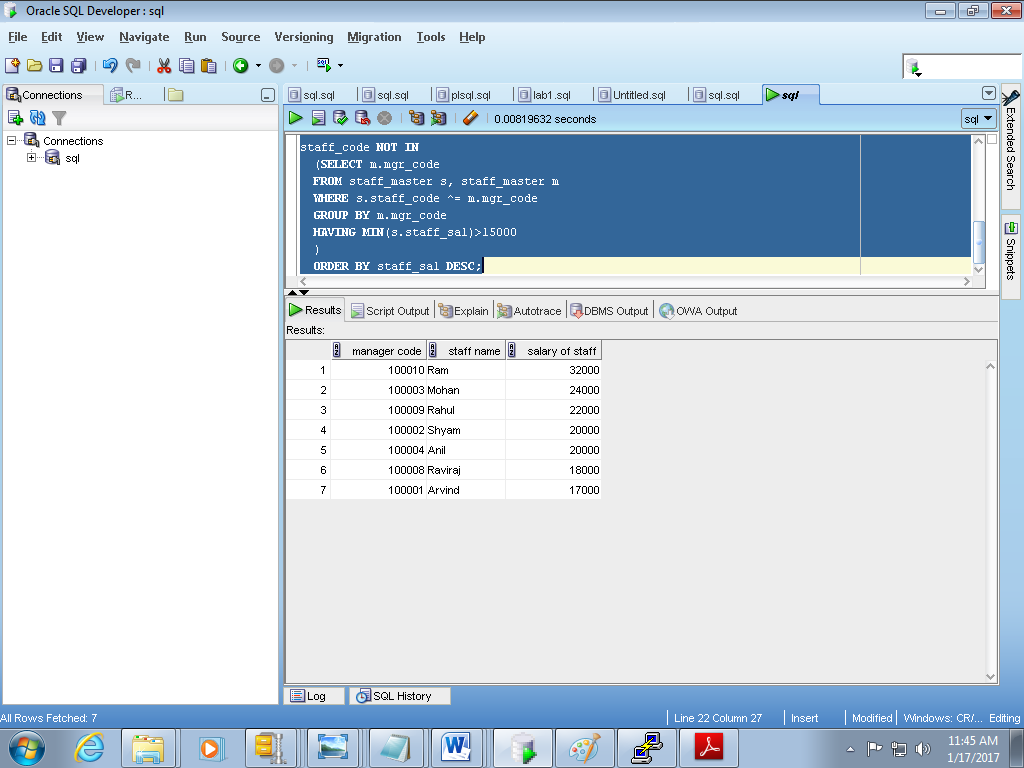
WHERE s.staff\_code ^= m.mgr\_code

GROUP BY m.mgr\_code

HAVING MIN(s.staff\_sal)>15000

)

ORDER BY staff\_sal DESC;

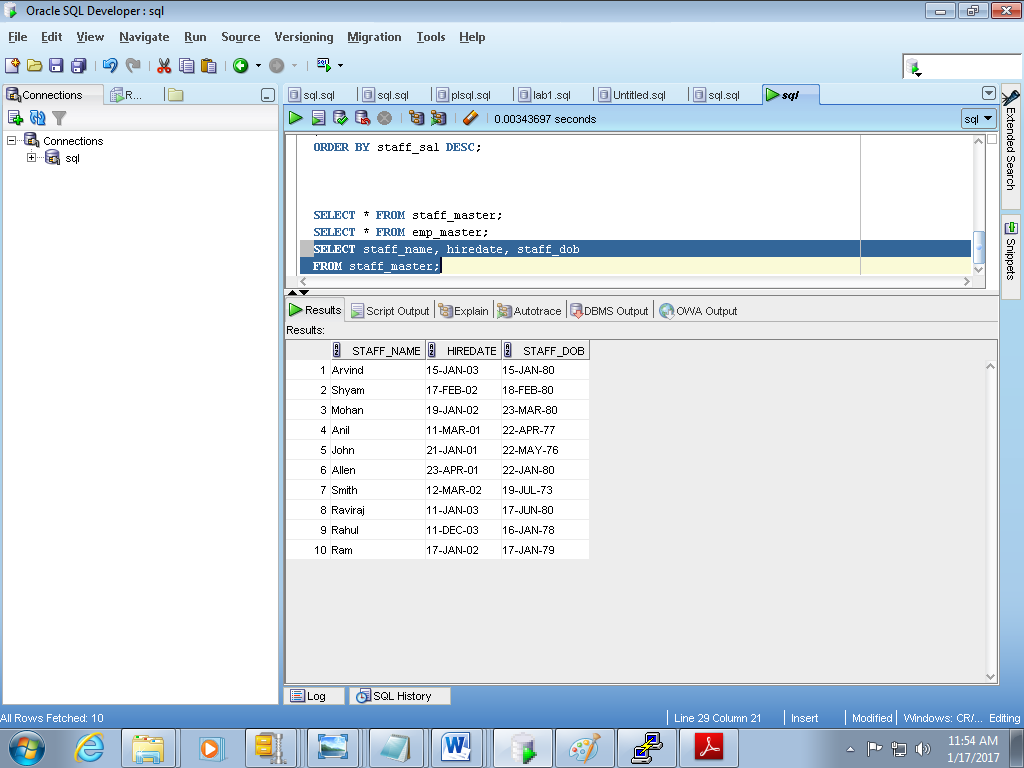


3.Since the staff\_master table doesnot contain first name, last name and date of joining

Staff name, hire date and staff date of birth are considered

SELECT staff\_name, hiredate, staff\_dob

FROM staff\_master;

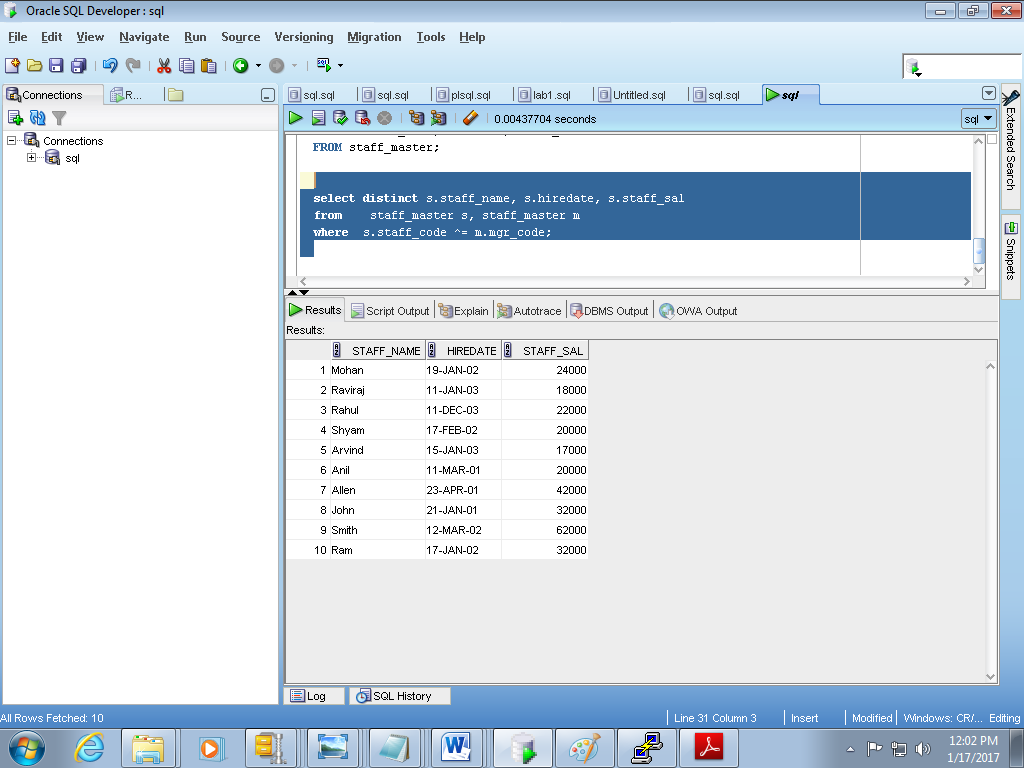


4.

select distinct s.staff\_name, s.hiredate, s.staff\_sal

from staff\_master s, staff\_master m

where s.staff\_code ^= m.mgr\_code;



Q2

create or replace procedure salary\_proc(p\_deptno dept.deptno%TYPE) /\*creating procedure\*/

IS

v\_deptno dept.deptno%TYPE;

v\_sal emp.sal%TYPE;

v\_exception EXCEPTION;

CURSOR c\_sal IS SELECT sal FROM emp WHERE deptno = p\_deptno;

BEGIN

OPEN c\_sal;

LOOP

FETCH c\_sal INTO v\_sal; /\*cursor value is fetched into v\_sal \*/

IF c\_sal%FOUND AND v\_sal < 3000 THEN

UPDATE test\_emp SET sal=sal+(sal\*20/200) /\*salary updated by 20% \*/

WHERE deptno =p\_deptno;

ELSIF v\_sal > 3000 THEN

UPDATE test\_emp SET sal=sal+(sal\*10/100) /\*salary updated by 10% \*/

WHERE deptno = p\_deptno;

ELSIF c\_sal%ROWCOUNT =0 THEN

RAISE v\_exception; /\*raises an exception if c\_sal%rowcount=0 \*/

END IF;

EXIT WHEN c\_sal%NOTFOUND;

END LOOP;

EXCEPTION

WHEN v\_exception THEN

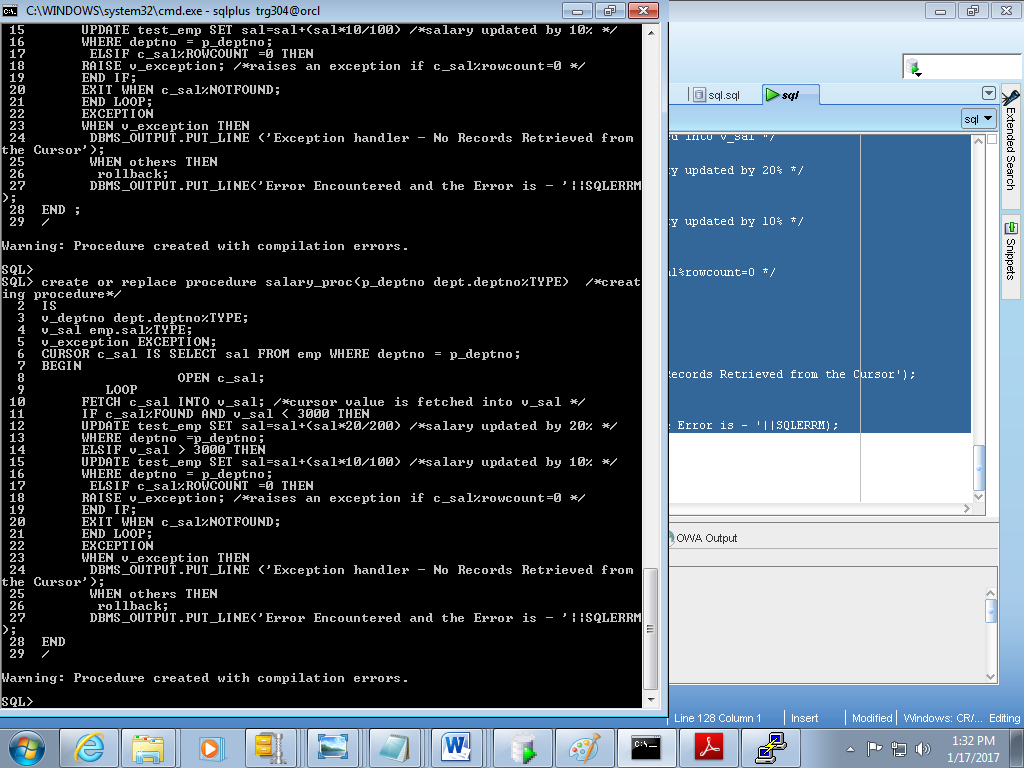
DBMS\_OUTPUT.PUT\_LINE ('Exception handler - No Records Retrieved from the Cursor');

WHEN others THEN

rollback;

DBMS\_OUTPUT.PUT\_LINE('Error Encountered and the Error is - '||SQLERRM);

END



Q3.

CREATE OR REPLACE function sal\_update\_func(v\_empno IN emp.empno%TYPE)

RETURN NUMBER AS

v\_commission emp.comm%TYPE;

v\_salary emp.sal%TYPE;

v\_updated\_sal emp.sal%TYPE;

BEGIN

SELECT comm,sal INTO v\_commission,v\_salary FROM emp

WHERE empno= v\_empno;

IF(v\_commission=0 OR v\_commission IS NULL) THEN

v\_commission:=((v\_salary\*5)/100);

v\_updated\_sal:=v\_salary+v\_commission;

ELSE

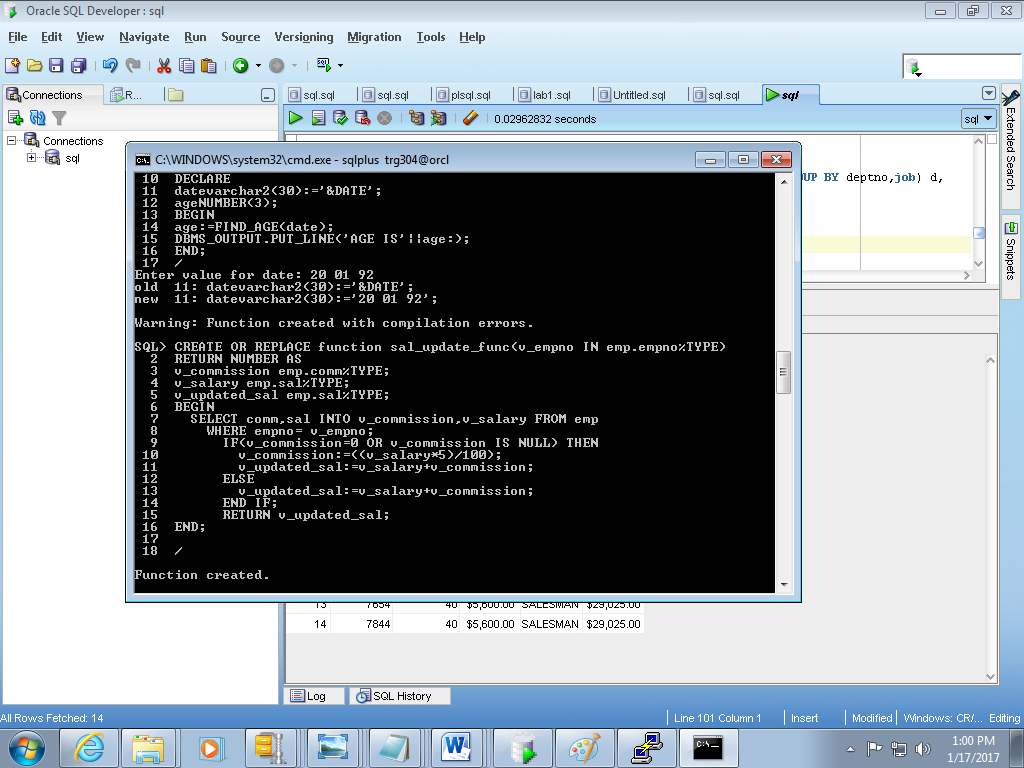
v\_updated\_sal:=v\_salary+v\_commission;

END IF;

RETURN v\_updated\_sal;

END;

EOF



Q5.

SELECT e.empno,d.deptno,d.total,d.job,t.tot FROM emp e,

(SELECT TO\_CHAR(SUM(SAL), '$99,999.00') AS total,deptno,job FROM emp GROUP BY deptno,job) d,

(SELECT TO\_CHAR(SUM(SAL),'$99,999.00') AS tot FROM EMP) t

WHERE e.deptno=d.deptno and e.job=d.job

ORDER BY deptno;

