

Name: Suraj Chandramauli

Roll No: 40

Exp. No: 9

Date: 26/06/2021

Procedures

Aim:

Demonstration of Procedures in SQL:

I.

- a) Create a table called 'employee' with the following attributes

Employee	
Field names	Type
ID	Number (3)
name	Varchar (20)

- b) Show the structure of the table.
c) Insert the following values into employee table.

id	name
101	Nithya
102	Maya

- d) Display all the values in 'employee' table.
e) Create a procedure to insert a number as 'id' to the 'employee' table. 'name' field of the table should accept default value as the 'user' of the given computer.
f) Display all the values in 'employee' table.
g) Create a procedure to include a new field called 'age' to the 'employee' table and also insert values for this field in all the existing rows as 20, 30, 18 respectively.
h) Create a procedure for employee to get the employee details where age>20.
i) Create a procedure to get employee details using 'id'.

II. Create a procedure to add 2 numbers.

III. Create a procedure to find largest among the given numbers.

Theory:

- A Procedure in PL/SQL is a subprogram unit that consists of a group of PL/SQL statements that can be called by name. Each procedure in PL/SQL has its own unique name by which it can be referred to and called. This subprogram unit in the Oracle database is stored as a database object.
- Procedures are standalone blocks of a program that can be stored in the database
- Call to these PLSQL procedures can be made by referring to their name, to execute the PL/SQL statements.
- It is mainly used to execute a process in PL/SQL.
- It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
- It contains declaration part (optional), execution part, exception handling part (optional).
- The values can be passed into Oracle procedure or fetched from the procedure through parameters.
- These parameters should be included in the calling statement.
- A Procedure in SQL can have a RETURN statement to return the control to the calling block, but it cannot return any values through the RETURN statement.
- Procedures cannot be called directly from SELECT statements. They can be called from another block or through EXEC keyword.
- Syntax for Procedures in SQL:
- CREATE OR REPLACE PROCEDURE <procedure_name> (<parameter1 IN/OUT
- <datatype> ..) [IS | AS] <declaration_part> BEGIN <execution part> EXCEPTION
- <exception handling part> END;
- Example for Procedures in SQL is:
- CREATE OR REPLACE PROCEDURE welcome_msg (p_name IN VARCHAR2) IS
BEGIN dbms_output.put_line ('Welcome '|| p_name); END; / EXEC welcome_msg
('Friends');

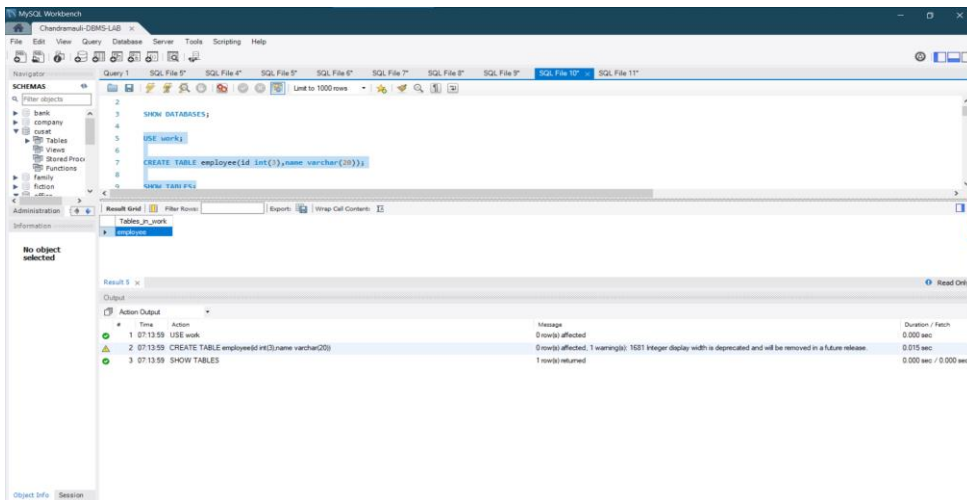
Result: As a result procedures in SQL is familiarized and output is verified.

Remarks:(To be filled by faculty)

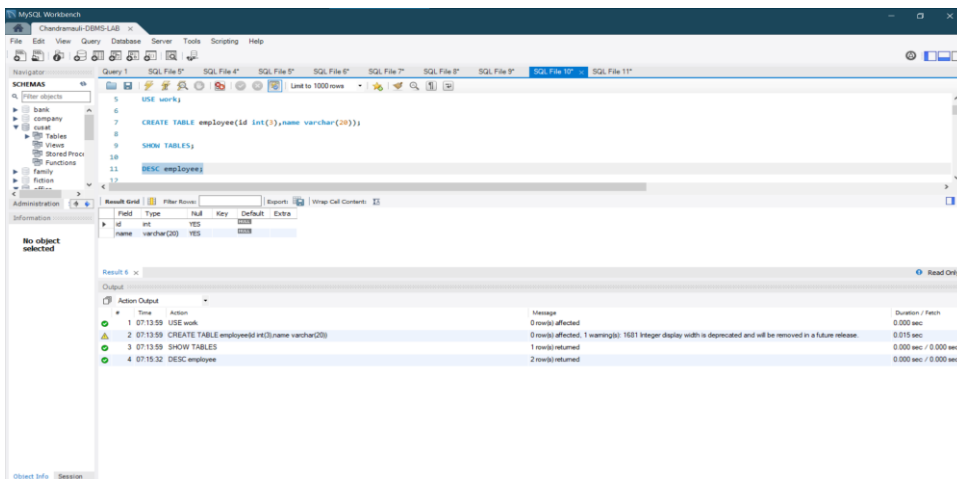
OUTPUT:

I)

a) Create a table called 'employee' with attributes



b)) Show the structure of the table



c) Insert the values into employee table.

MySQL Workbench

Chandramauli-DBMS-LAB

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1 SQL File 5* SQL File 4* SQL File 5* SQL File 5* SQL File 7* SQL File 8* SQL File 9* SQL File 10* SQL File 11*

SCHEMAS

Filter objects

- bank
- company
- cust
- Tables
- views
- Stored Proc
- Functions
- Family
- Relation

Administration

Information

No object selected

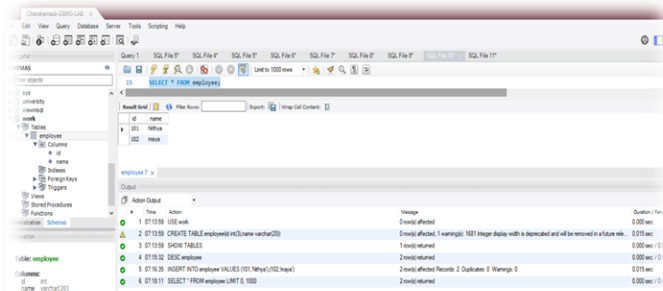
Output

Action Output

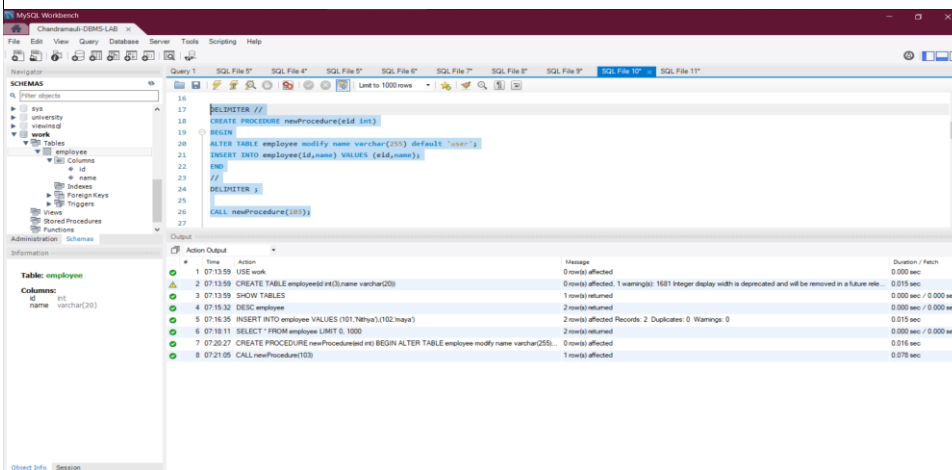
#	Time	Action	Message	Duration / Patch
1	07:13:59	USE work	0 row(s) affected	0.000 sec
2	07:13:59	CREATE TABLE employee(id int(3),name varchar(20))	0 row(s) affected, 1 warning(s): 1661 Integer display width is deprecated and will be removed in a future release.	0.015 sec
3	07:13:59	SHOW TABLES	1 row(s) returned	0.000 sec / 0.000 sec
4	07:15:32	DESC employee	2 row(s) returned	0.000 sec / 0.000 sec
5	07:16:35	INSERT INTO employee VALUES (101,'Nithya'),(102,'naya')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.015 sec

Object Info Session

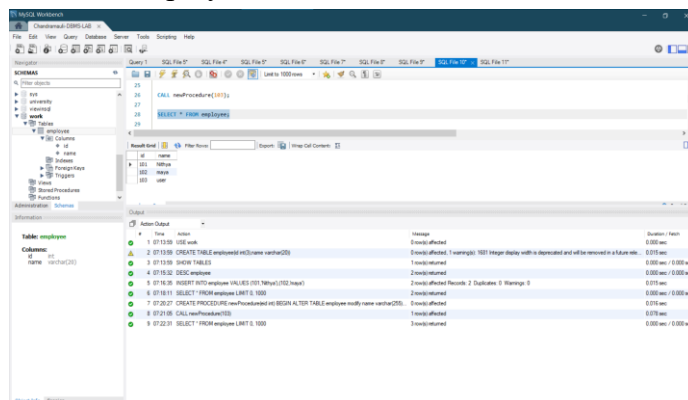
d) Display all the values in 'employee' table



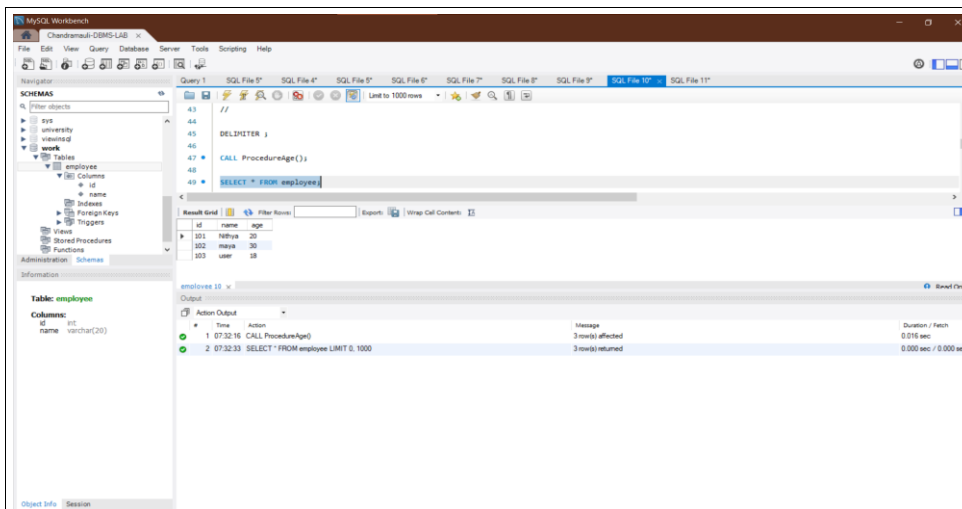
e)) Create a procedure to insert a number as 'id' to the 'employee' table. 'name' field of the table should accept default value as the 'user' of the given computer.



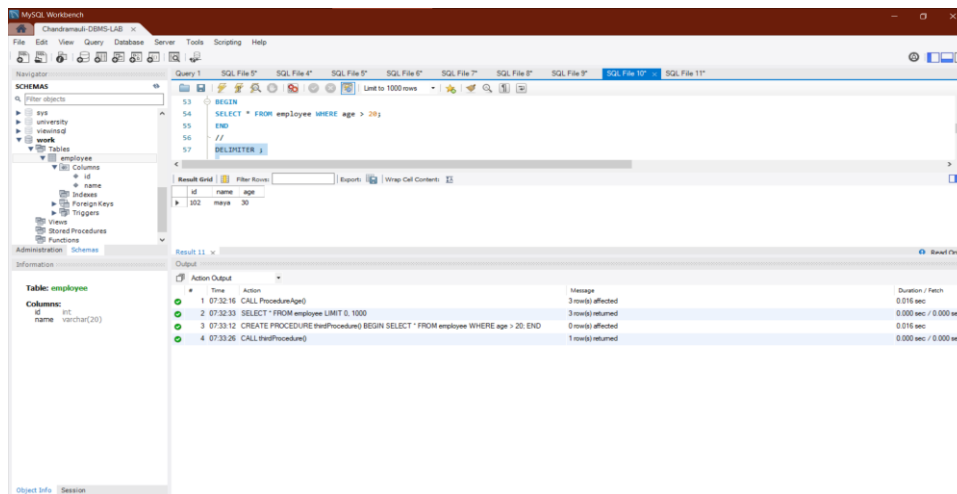
f) Display all the values in 'employee' table



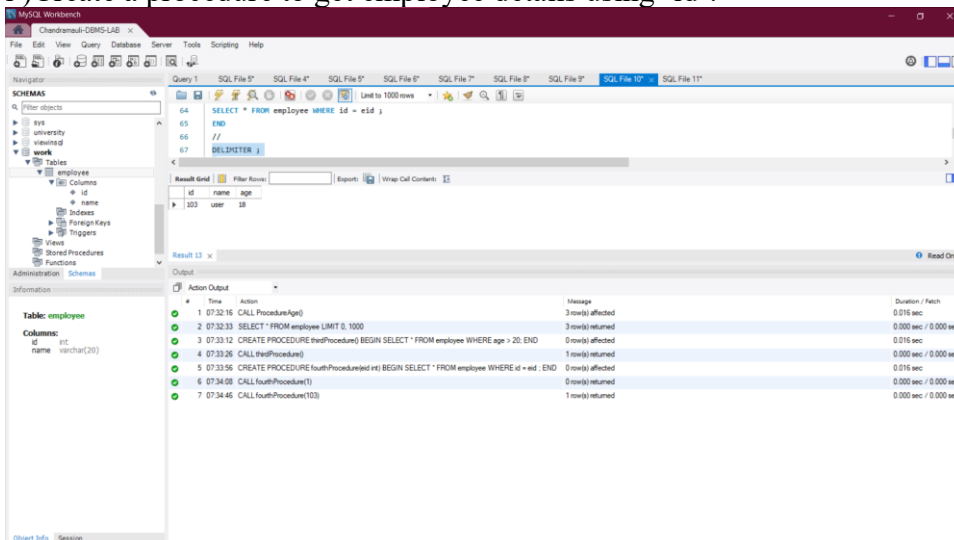
g) Create a procedure to include a new field called 'age' to the 'employee' table and also insert values for this field in all the existing rows as 20, 30, 18 respectively.



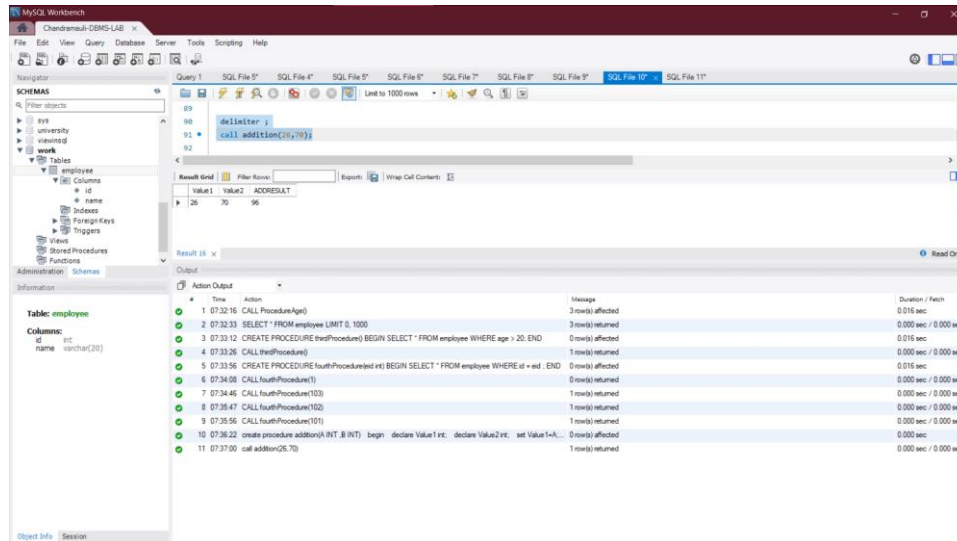
h) Create a procedure for employee to get the employee details where age>20.



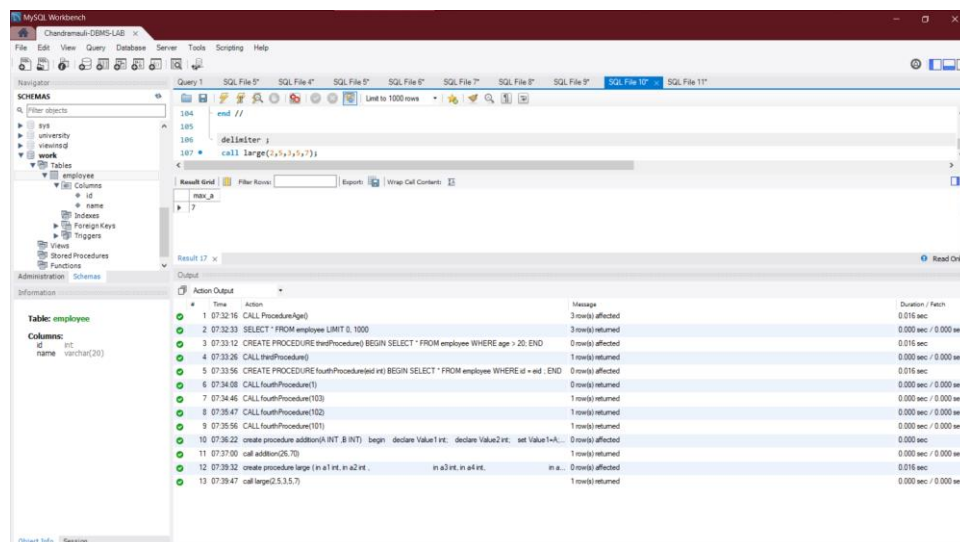
i) Create a procedure to get employee details using 'id'.



(II) Create a procedure to add 2 numbers.



(III) Create a procedure to find largest among the given numbers.



(LEFT SIDE OF A PAGE)

Name: Suraj Chandramauli

Roll No:40

Exp. No:10

Date: 26/06/2021

Functions

Aim:

Demonstration of Functions in SQL:

Consider the 'bank' table and do the following operations.

BANK		
Acc_no	B_name	balance
101	SBI	25000
102	SBI	5000
103	FEDRAL	10000
104	AXIS	15000
105	CANARA	50000

- Create a function for withdrawing money from an account in a bank management system which uses bank table. The minimum balance the account should hold is 500.
- Display all the values in 'bank' table.
- Create a function for depositing money to an account in a bank management system which uses bank table.
- Display all the values in 'bank' table.

Theory:

- Functions is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.
- Functions are a standalone block that is mainly used for calculation purpose. Function use RETURN keyword to return the value, and the datatype of this is defined at the time of creation. A Function should either return a value or raise the exception, i.e. return is mandatory in functions.
- Function with no DML statements can be directly called in SELECT query whereas the function with DML operation can only be called from other PL/SQL blocks. It can have nested blocks, or it can be defined and nested inside the other blocks or packages. It contains declaration part (optional), execution part, exception handling part (optional).
- The values can be passed into the function or fetched from the procedure through the parameters. These parameters should be included in the calling statement.
- A PLSQL function can also return the value through OUT parameters other than using RETURN. Since it will always return the value, in calling statement it always accompanies with assignment operator to populate the variables.
- Syntax for Functions in SQL is:

```
CREATE OR REPLACE FUNCTION <procedure_name> ( <parameter1 IN/OUT  
<datatype> ) RETURN <datatype> [ IS | AS ] <declaration_part> BEGIN <execution part>  
EXCEPTION <exception handling part> END;
```

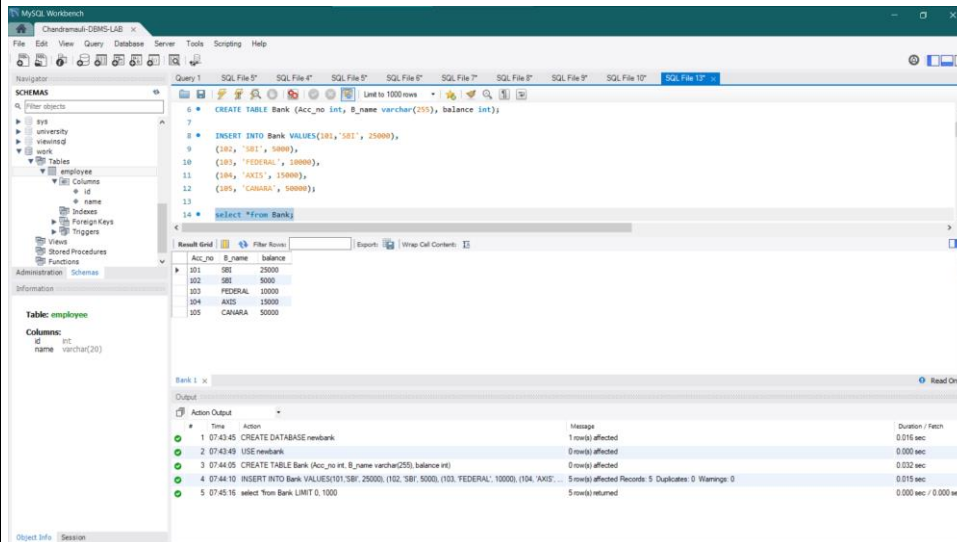
- Example for Functions in SQL is:

```
CREATE OR REPLACE FUNCTION welcome_msgJune ( p_name IN VARCHAR2)  
RETURN VAR.CHAR2 IS BEGIN RETURN ('Welcome '|| p_name); END; / DECLARE  
lv_msg VARCHAR2(250); BEGIN lv_msg := welcome_msg_func ('TRIKKZ');  
dbms_output.put_line(lv_msg); END; SELECT welcome_msg_func('TRIKKZ:') FROM  
DUAL;
```

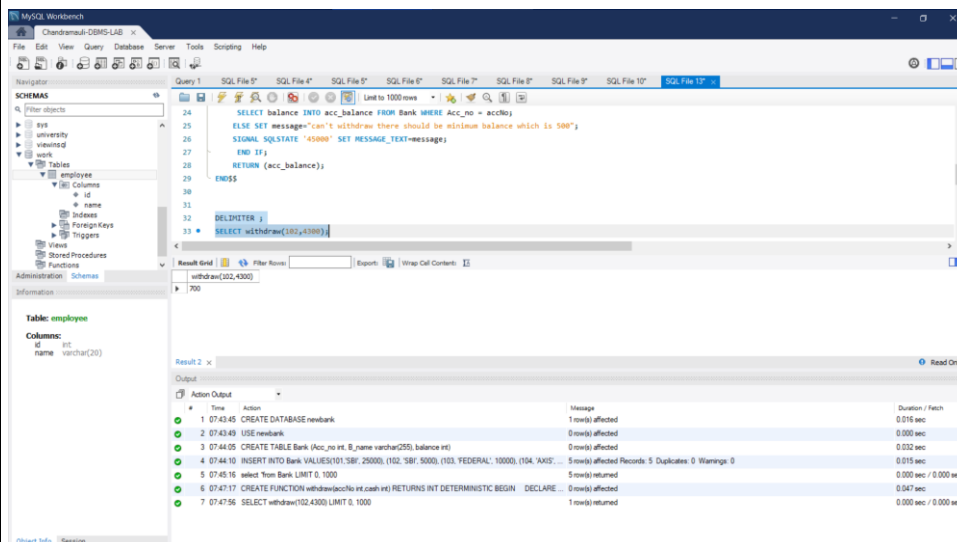
Result: As a result functions in SQL are familiarized and output is verified.

Remarks:(To be filled by faculty)

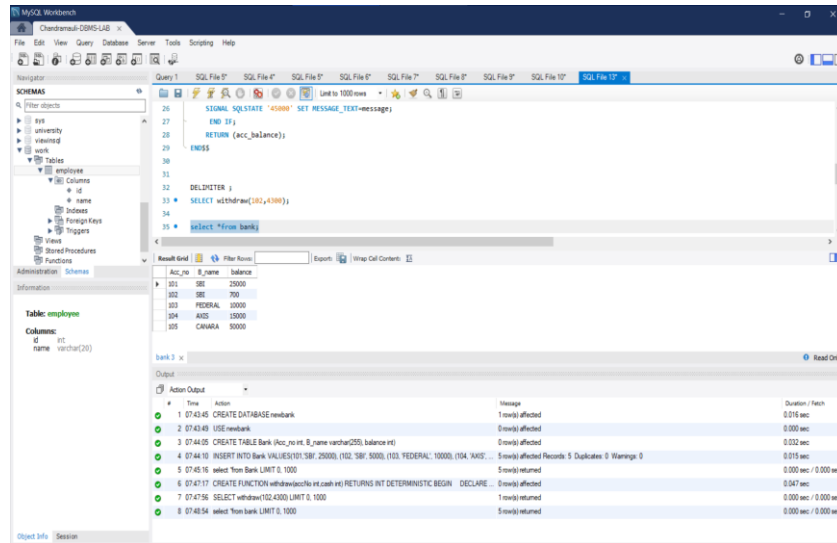
OUTPUT:



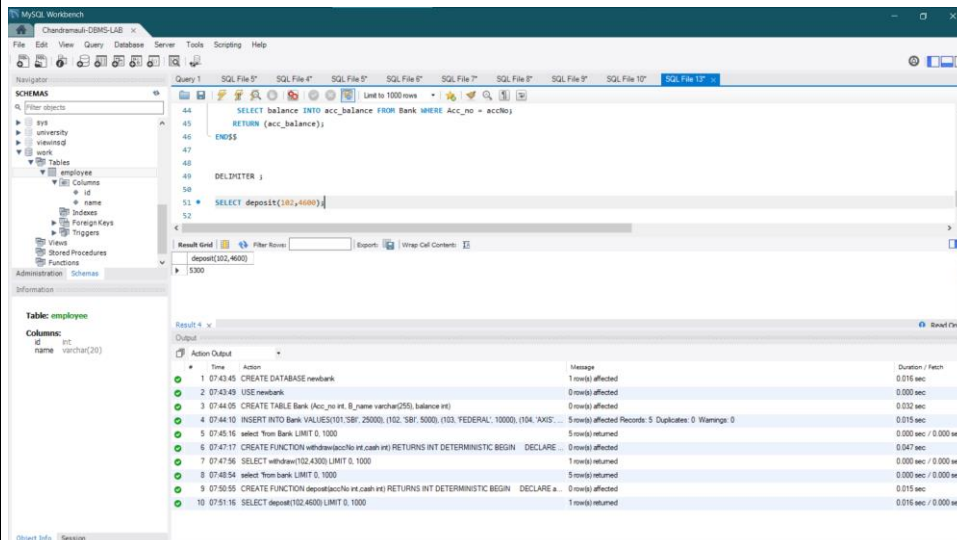
a) Create a function for withdrawing money from an account in a bank management system which uses bank table. The minimum balance the account should hold is 500.



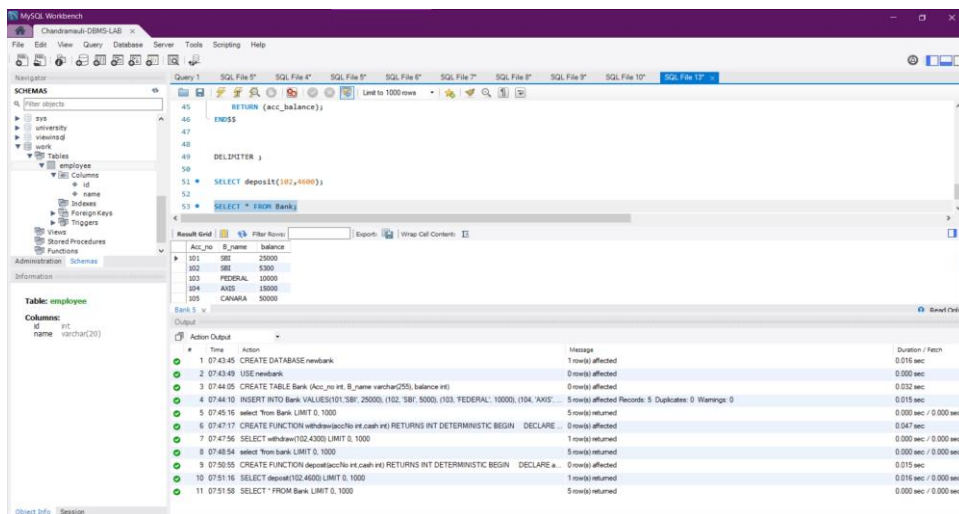
b) Display all the values in 'bank' table



c)) Create a function for depositing money to an account in a bank management system which uses bank table



d) Display all the values in 'bank' table.



(LEFT SIDE OF A PAGE)

Name: Suraj Chandramauli

Roll No:40

Exp. No:11

Date: 26/06/2021

Cursor

Aim:

Demonstration of Cursors in SQL:

Consider the 'student table' of a particular class and do the following operations.

Roll_No	Name	M1	M2	M3	Total	Percentage	grade
15	Jenny	20	30	20	0	0	0
41	Reena	98	90	85	0	0	0
22	Leena	40	45	60	0	0	0
23	Boban	50	30	20	0	0	0

I. Explicit cursor

a) Create a cursor for calculating the total marks and percentage of the student, grade of the student in a student management system which uses a student table. Grade the student according to the following rules.

Total Marks	Grade
≥ 250	Distinction
180-250	First class
120-179	Second class
80-180	Third class
< 80	fail

b) Display all the details of student table.

c) Create a cursor to find student who got the highest mark from the 'student' table and display the particular student details in the following format. Remark for a highest scored student is 'topper of the class'.

Roll no:

Student name:

Total marks:

Grade:

Remarks:

d) Display all the details of student table.

Theory:

- A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set. There are 2 types of cursors: Implicit cursor and Explicit cursor.
- Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
- Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.
- In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK_ROWCOUNT and %BULK_EXCEPTIONS, designed for use with the FORALL statement.
- Explicit Cursors are created by users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row manner.
- Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row
- Declaring the cursor simply means to create one named context area for the 'SELECT' statement that is defined in the declaration part. The name of this context area is same as the cursor name.
- Opening the cursor will instruct the PL/SQL to allocate the memory for this cursor. It will make the cursor ready to fetch the records.
- In this process, the 'SELECT' statement is executed and the rows fetched is stored in the allocated memory. These are now called as active sets. Fetching data from the cursor is a record-level activity that means we can access the data in a record-by-record way. Each fetch statement will fetch one active set and holds the information of that particular record. This statement is same as 'SELECT' statement that fetches the record and assigns to the variable in the 'INTO' clause, but it will not throw any exceptions
- Once all the record is fetched now, we need to close the cursor so that the memory allocated to this context area will be released.
- Syntax for cursor is:

```
DECLARE CURSOR <cursor_name> IS <SELECT statement^>  
<cursor_variable declaration> BEGIN OPEN <cursor_name>; FETCH  
<cursor_name> INTO
```

<cursor_variable>; . . CLOSE <cursor_name>; END;

- Example for cursor is:

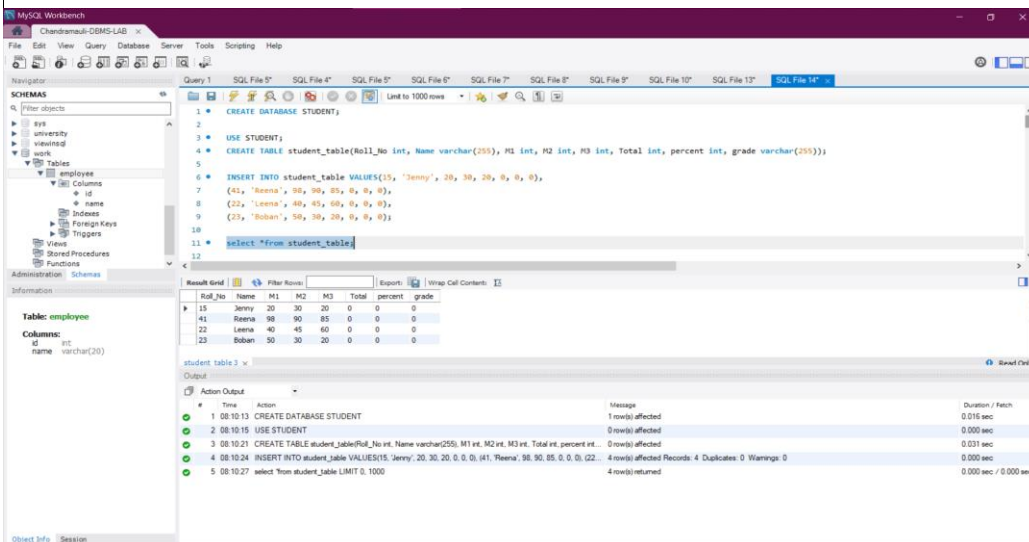
```
DECLARE CURSOR TRIKKZ_det IS SELECT emp_name FROM emp;  
lv_emp_name emp.emp_name%type; BEGIN OPEN TRIKKZ_det; LOOP  
FETCH TRIKKZ_det INTO lv_emp_name; IF TRIKKZ_det%NOTFOUND  
THEN EXIT; END IF; Dbms_output.put_line('Employee  
Fetched:'||lv_emp_name); END LOOP; Dbms_output.put_line('Total rows  
fetched is'||guru99_det%ROWCOUNT); CLOSE TRIKKZ_det; END: /
```

- "FOR LOOP" statement can be used for working with cursors. We can give the cursor name instead of range limit in the FOR loop statement so that the loop will work from the first record of the cursor to the last record of the cursor. The cursor variable, opening of cursor, fetching and closing of the cursor will be done implicitly by the FOR loop.
- Syntax for cursor using FOR LOOP is:
- DECLARE CURSOR <cursor_name> IS <SELECT statement>; BEGIN FOR I IN
- <cursor_name> LOOP . . END LOOP; END;
- Example for cursor using FOR LOOP is:
- DECLARE CURSOR TRIKKZ_det IS SELECT emp_name FROM emp; BEGIN FOR
- lv_emp_name IN TRIKKZ_det LOOP Dbms_output.put_line('Employee
Fetched:'||lv_emp_name.emp_name); END LOOP; END; /

Result: As a result cursors in SQL are familiarized and output is verified.

Remarks:(To be filled by faculty)

OUTPUT:



MySQL Workbench

Chandramauli-DBMS-LAB

Query 1

```
1 * CREATE DATABASE STUDENT;
2
3 * USE STUDENT;
4 * CREATE TABLE student_table(Roll_No int, Name varchar(255), M1 int, M2 int, M3 int, Total int, percent int, grade varchar(255));
5
6 * INSERT INTO student_table VALUES(15, 'Jenny', 20, 30, 20, 0, 0, 0);
7 * (1, 'Reena', 90, 90, 85, 0, 0, 0);
8 * (21, 'Leena', 40, 45, 60, 0, 0, 0);
9 * (23, 'Boban', 50, 30, 20, 0, 0, 0);
10
11 * select *from student_table;
```

Result Grid

Roll_No	Name	M1	M2	M3	Total	percent	grade
15	Jenny	20	30	20	0	0	0
41	Reena	90	90	85	0	0	0
21	Leena	40	45	60	0	0	0
23	Boban	50	30	20	0	0	0

student_table 3

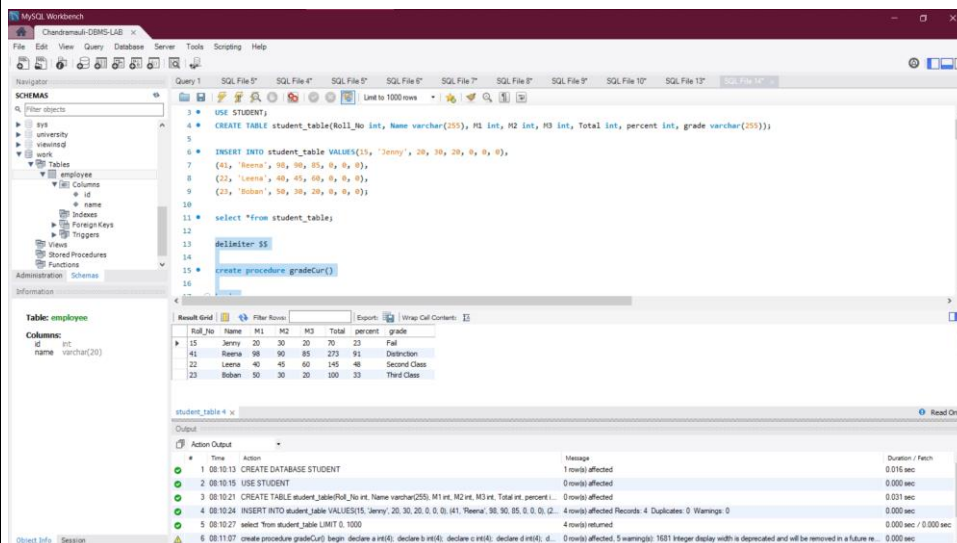
Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	08:10:13	CREATE DATABASE STUDENT	1 row(s) affected	0.016 sec
2	08:10:15	USE STUDENT	0 row(s) affected	0.000 sec
3	08:10:21	CREATE TABLE student_table(Roll_No int, Name varchar(255), M1 int, M2 int, M3 int, Total int, percent int, grade varchar(255));	0 row(s) affected	0.031 sec
4	08:10:24	INSERT INTO student_table VALUES(15, 'Jenny', 20, 30, 20, 0, 0, 0); (1, 'Reena', 90, 90, 85, 0, 0, 0); (21, 'Leena', 40, 45, 60, 0, 0, 0); (23, 'Boban', 50, 30, 20, 0, 0, 0);	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
5	08:10:27	select *from student_table LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

I) Explicit cursor

a) Create a cursor for calculating the total marks and percentage of the student, grade of the student in a student management system which uses a student table.



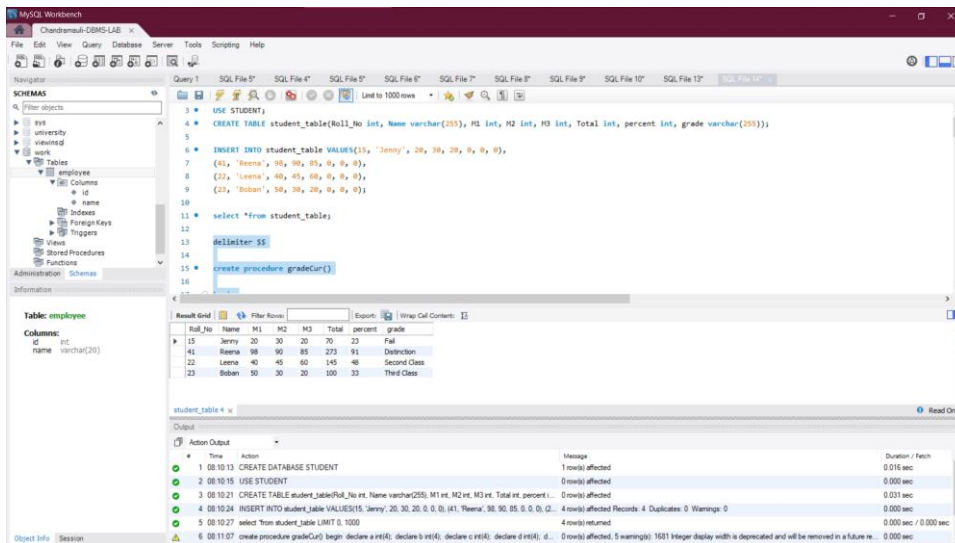
MySQL Workbench

Chandramauli-DBMS-LAB

Query 1

```
3 * USE STUDENT;
4 * CREATE TABLE student_table(Roll_No int, Name varchar(255), M1 int, M2 int, M3 int, Total int, percent int, grade varchar(255));
5
6 * INSERT INTO student_table VALUES(15, 'Jenny', 20, 30, 20, 0, 0, 0);
7 * (1, 'Reena', 90, 90, 85, 0, 0, 0);
8 * (21, 'Leena', 40, 45, 60, 0, 0, 0);
9 * (23, 'Boban', 50, 30, 20, 0, 0, 0);
10
11 * select *from student_table;
12
13 * delimiter $$
14
15 * create procedure gradeCur()
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *
1001 *
1002 *
1003 *
1004 *
1005 *
1006 *
1007 *
1008 *
1009 *
1010 *
1011 *
1012 *
1013 *
1014 *
1015 *
1016 *
1017 *
1018 *
1019 *
1020 *
1021 *
1022 *
1023 *
1024 *
1025 *
1026 *
1027 *
1028 *
1029 *
1030 *
1031 *
1032 *
1033 *
1034 *
1035 *
1036 *
1037 *
1038 *
1039 *
1040 *
1041 *
1042 *
1043 *
1044 *
1045 *
1046 *
1047 *
1048 *
1049 *
1050 *
1051 *
1052 *
1053 *
1054 *
1055 *
1056 *
1057 *
1058 *
1059 *
1060 *
1061 *
1062 *
1063 *
1064 *
1065 *
1066 *
1067 *
1068 *
1069 *
1070 *
1071 *
1072 *
1073 *
1074 *
1075 *
1076 *
1077 *
1078 *
1079 *
1080 *
1081 *
1082 *
1083 *
1084 *
1085 *
1086 *
1087 *
1088 *
1089 *
1090 *
1091 *
1092 *
1093 *
1094 *
1095 *
1096 *
1097 *
1098 *
1099 *
1100 *
1101 *
1102 *
1103 *
1104 *
1105 *
1106 *
1107 *
1108 *
1109 *
1110 *
1111 *
1112 *
1113 *
1114 *
1115 *
1116 *
1117 *
1118 *
1119 *
1120 *
1121 *
1122 *
1123 *
1124 *
1125 *
1126 *
1127 *
1128 *
1129 *
1130 *
1131 *
1132 *
1133 *
1134 *
1135 *
1136 *
1137 *
1138 *
1139 *
1140 *
1141 *
1142 *
1143 *
1144 *
1145 *
1146 *
1147 *
1148 *
1149 *
1150 *
1151 *
1152 *
1153 *
1154 *
1155 *
1156 *
1157 *
1158 *
1159 *
1160 *
1161 *
1162 *
1163 *
1164 *
1165 *
1166 *
1167 *
1168 *
1169 *
1170 *
1171 *
1172 *
1173 *
1174 *
1175 *
1176 *
1177 *
1178 *
1179 *
1180 *
1181 *
1182 *
1183 *
1184 *
1185 *
1186 *
1187 *
1188 *
1189 *
1190 *
1191 *
1192 *
1193 *
1194 *
1195 *
1196 *
1197 *
1198 *
1199 *
1200 *
1201 *
1202 *
1203 *
1204 *
1205 *
1206 *
1207 *
1208 *
1209 *
1210 *
1211 *
1212 *
1213 *
1214 *
1215 *
1216 *
1217 *
1218 *
1219 *
1220 *
1221 *
1222 *
1223 *
1224 *
1225 *
1226 *
1227 *
1228 *
1229 *
1230 *
1231 *
1232 *
1233 *
1234 *
1235 *
1236 *
1237 *
1238 *
1239 *
1240 *
1241 *
1242 *
1243 *
1244 *
1245 *
1246 *
1247 *
1248 *
1249 *
1250 *
1251 *
1252 *
1253 *
1254 *
1255 *
1256 *
1257 *
1258 *
1259 *
1260 *
1261 *
1262 *
1263 *
1264 *
1265 *
1266 *
1267 *
1268 *
1269 *
1270 *
1271 *
1272 *
1273 *
1274 *
1275 *
1276 *
1277 *
1278 *
1279 *
1280 *
1281 *
1282 *
1283 *
1284 *
1285 *
1286 *
1287 *
1288 *
1289 *
1290 *
1291 *
1292 *
1293 *
1294 *
1295 *
1296 *
1297 *
1298 *
1299 *
1300 *
1301 *
1302 *
1303 *
1304 *
1305 *
1306 *
1307 *
1308 *
1309 *
1310 *
1311 *
1312 *
1313 *
1314 *
1315 *
1316 *
1317 *
1318 *
1319 *
1320 *
1321 *
1322 *
1323 *
1324 *
1325 *
1326 *
1327 *
1328 *
1329 *
1330 *
1331 *
1332 *
1333 *
1334 *
1335 *
1336 *
1337 *
1338 *
1339 *
1340 *
1341 *
1342 *
1343 *
1344 *
1345 *
1346 *
1347 *
1348 *
1349 *
1350 *
1351 *
1352 *
1353 *
1354 *
1355 *
1356 *
1357 *
1358 *
1359 *
1360 *
1361 *
1362 *
1363 *
1364 *
1365 *
1366 *
1367 *
1368 *
1369 *
1370 *
1371 *
1372 *
1373 *
1374 *
1375 *
1376 *
1377 *
1378 *
1379 *
1380 *
1381 *
1382 *
1383 *
1384 *
1385 *
1386 *
1387 *
1388 *
1389 *
1390 *
1391 *
1392 *
1393 *
1394 *
1395 *
1396 *
1397 *
1398 *
1399 *
1400 *
1401 *
1402 *
1403 *
1404 *
1405 *
1406 *
1407 *
1408 *
1409 *
1410 *
1411 *
1412 *
1413 *
1414 *
1415 *
1416 *
1417 *
1418 *
1419 *
1420 *
1421 *
1422 *
1423 *
1424 *
1425 *
1426 *
1427 *
1428 *
1429 *
1430 *
1431 *
1432 *
1433 *
1434 *
1435 *
1436 *
1437 *
1438 *
1439 *
1440 *
1441 *
1442 *
1443 *
1444 *
1445 *
1446 *
1447 *
1448 *
1449 *
1450 *
1451 *
1452 *
1453 *
1454 *
1455 *
1456 *
1457 *
1458 *
1459 *
1460 *
1461 *
1462 *
1463 *
1464 *
1465 *
1466 *
1467 *
1468 *
1469 *
1470 *
1471 *
1472 *
1473 *
1474 *
1475 *
1476 *
1477 *
1478 *
1479 *
1480 *
1481 *
1482 *
1483 *
1484 *
1485 *
1486 *
1487 *
1488 *
1489 *
1490 *
1491 *
1492 *
1493 *
1494 *
1495 *
1496 *
1497 *
1498 *
1499 *
1500 *
1501 *
1502 *
1503 *
1504 *
1505 *
1506 *
1507 *
1508 *
1509 *
1510 *
1511 *
1512 *
1513 *
1514 *
1515 *
1516 *
1517 *
1518 *
1519 *
1520 *
1521 *
1522 *
1523 *
1524 *
1525 *
1526 *
1527 *
1528 *
1529 *
1530 *
1531 *
1532 *
1533 *
1534 *
1535 *
1536 *
1537 *
1538 *
1539 *
1540 *
1541 *
1542 *
1543 *
1544 *
1545 *
1546 *
1547 *
1548 *
1549 *
1550 *
1551 *
1552 *
1553 *
1554 *
1555 *
1556 *
1557 *
1558 *
1559 *
1560 *
1561 *
1562 *
1563 *
1564 *
1565 *
1566 *
1567 *
1568 *
1569 *
1570 *
1571 *
1572 *
1573 *
1574 *
1575 *
1576 *
1577 *
1578 *
1579 *
1580 *
1581 *
1582 *
1583 *
1584 *
1585 *
1586 *
1587 *
1588 *
1589 *
1590 *
1591 *
1592 *
1593 *
1594 *
1595 *
1596 *
1597 *
1598 *
1599 *
1600 *
1601 *
1602 *
1603 *
1604 *
1605 *
1606 *
1607 *
1608 *
1609 *
1610 *
1611 *
1612 *
1613 *
1614 *
1615 *
1616 *
1617 *
1618 *
1619 *
1620 *
1621 *
1622 *
1623 *
1624 *
1625 *
1626 *
1627 *
1628 *
1629 *
1630 *
1631 *
1632 *
1633 *
1634 *
1635 *
1636 *
1637 *
1638 *
1639 *
1640 *
1641 *
1642 *
1643 *
1644 *
1645 *
1646 *
1647 *
1648 *
1649 *
1650 *
1651 *
1652 *
1653 *
1654 *
1655 *
1656 *
1657 *
1658 *
1659 *
1660 *
1661 *
1662 *
1663 *
1664 *
1665 *
1666 *
1667 *
1668 *
1669 *
1670 *
1671 *
1672 *
1673 *
1674 *
1675 *
1676 *
1677 *
1678 *
1679 *
1680 *
1681 *
1682 *
1683 *
1684 *
1685 *
1686 *
1687 *
1688 *
1689 *
1690 *
1691 *
1692 *
1693 *
1694 *
1695 *
1696 *
1697 *
1698 *
1699 *
1700 *
1701 *
1702 *
1703 *
1704 *
1705 *
1706 *
1707 *
1708 *
1709 *
1710 *
1711 *
1712 *
1713 *
1714 *
1715 *
1716 *
1717 *
1718 *
1719 *
1720 *
1721 *
1722 *
1723 *
1724 *
1725 *
1726 *
1727 *
1728 *
1729 *
1730 *
1731 *
1732 *
1733 *
1734 *
1735 *
1736 *
1737 *
1738 *
1739 *
1740 *
1741 *
1742 *
1743 *
1744 *
1745 *
1746 *
1747 *
1748 *
1749 *
1750 *
1751 *
1752 *
1753 *
1754 *
1755 *
1756 *
1757 *
1758 *
1759 *
1760 *
1761 *
1762 *
1763 *
1764 *
1765 *
1766 *
1767 *
1768 *
1769 *
1770 *
1771 *
1772 *
1773 *
1774 *
1775 *
1776 *
1777 *
1778 *
1779 *
1780 *
1781 *
1782 *
1783 *
1784 *
1785 *
1786 *
1787 *
1788 *
1789 *
1790 *
1791 *
1792 *
1793 *
1794 *
1795 *
1796 *
1797 *
1798 *
1799 *
1800 *
1801 *
1802 *
1803 *
1804 *
1805 *
1806 *
1807 *
1808 *
1809 *
1810 *
1811 *
1812 *
1813 *
1814 *
1815 *
1816 *
1817 *
1818 *
1819 *
1820 *
1821 *
1822 *
1823 *
1824 *
1825 *
1826 *
1827 *
1828 *
1829 *
1830 *
1831 *
1832 *
1833 *
1834 *
1835 *
1836 *
1837 *
1838 *
1839 *
1840 *
1841 *
1842 *
1843 *
1844 *
1845 *
1846 *
1847 *
1848 *
1849 *
1850 *
1851 *
1852 *
1853 *
1854 *
1855 *
1856 *
1857 *
1858 *
1859 *
1860 *
1861 *
1862 *
1863 *
1864 *
1865 *
1866 *
1867 *
1868 *
1869 *
1870 *
1871 *
1872 *
1873 *
1874 *
1875 *
1876 *
1877 *
1878 *
1879 *
1880 *
1881 *
1882 *
1883 *
1884 *
1885 *
1886 *
1887 *
1888 *
1889 *
1890 *
1891 *
1892 *
1893 *
1894 *
1895 *
1896 *
1897 *
1898 *
1899 *
1900 *
1901 *
1902 *
1903 *
1904 *
1905 *
1906 *
1907 *
1908 *
1909 *
1910 *
1911 *
1912 *
1913 *
1914 *
1915 *
1916 *
1917 *
1918 *
1919 *
1920 *
1921 *
1922 *
1923 *
1924 *
1925 *
1926 *
1927 *
1928 *
1929 *
1930 *
1931 *
1932 *
1933 *
1934 *
1935 *
1936 *
1937 *
1938 *
1939 *
1940 *
1941 *
1942 *
1943 *
1944 *
1945 *
1946 *
1947 *
1948 *
1949 *
1950 *
1951 *
1952 *
1953 *
1954 *
1955 *
1956 *
1957 *
1958 *
1959 *
1960 *
1961 *
1962 *
1963 *
1964 *
1965 *
1966 *
1967 *
1968 *
1969 *
1970 *
1971 *
1972 *
1973 *
1974 *
1975 *
1976 *
1977 *
1
```

b) Display all the details of student table.



c) Create a cursor to find student who got the highest mark from the 'student' table and display the particular student details in the following format. Remark for a highest scored student is 'topper of the class'.

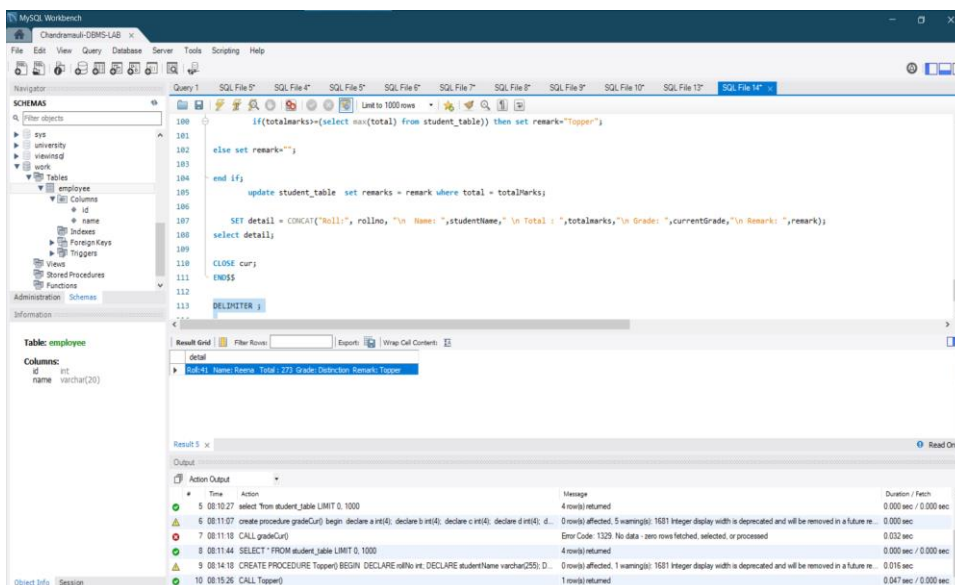
Roll no:

Student name:

Total marks:

Grade:

Remarks:



d) Display all the details of student table

MySQL Workbench

Chandimal-DBMS-LAB

File Edit View Query Database Server Tools Scripting Help

Schemas

- Filter objects
- mysql
- university
- viewmodel
- work
- employee
 - Columns
 - id
 - name
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions

Administration Schemas

Table: employee

Columns:

- id int
- name varchar(20)

Query 1

```

183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Result Grid

Roll_No	Name	M1	M2	M3	Total	percent	grade	remarks
15	Jerry	20	30	20	70	23	Fail	
41	Keema	98	98	88	273	91	Distinction	Topper
22	Lenna	40	45	60	145	48	Second Class	
23	Boban	50	30	20	100	33	Third Class	

student_table 6 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
6	08:11:07	create procedure gradeCur() begin declare a int(4); declare b int(4); declare c int(4); declare d int(4); d := 0; while a <= b do update student_table set remarks = remark where total = totalMarks; end if; update student_table set remarks = remark where total = totalMarks; SET detail = CONCAT("RollNo:", rollno, " Name: ", studentName, " Total : ", totalmarks, " Grade: ", currentGrade, " Remark: ", remark); select detail; CLOSE cur; END\$\$	0 row(s) affected. 5 warning(s): 1681 Integer display width is deprecated and will be removed in a future re...	0.000 sec
7	08:11:18	CALL gradeCur()	Error Code: 1329 No data - zero rows fetched, selected, or processed	0.032 sec
8	08:11:44	SELECT * FROM student_table LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
9	08:14:18	CREATE PROCEDURE Topper() BEGIN DECLARE rollNo int; DECLARE studentName varchar(255); D...	0 row(s) affected. 1 warning(s): 1681 Integer display width is deprecated and will be removed in a future re...	0.016 sec
10	08:15:26	CALL Topper()	1 row(s) returned	0.047 sec / 0.000 sec
11	08:16:17	select * from student_table LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

(LEFT SIDE OF A PAGE)

Name: Suraj Chandramauli

Roll No: 40

Exp. No: 12

Date: 26/06/2021

Triggers

Aim:

Demonstration of trigger in SQL is:

a) Create a table called 'reservations' with following fields.

reservations		
name	Null	Type
Fight_id	Not null	Char(6)
Customer_Phone	Not null	number

b) Create another table called 'flights' with the following fields. Make 'flight_id' the primary key in 'flights' table.

flights		
Flight_id	Not null	Char(6)
Seats	Not null	number

c) Show the structure of 'reservations' table.

d) Show the structure of 'flights' table.

e) Insert the following rows into 'flights' table.

Flight_id	Seats
AC0529	120
Aco530	0

f) Display all the details of 'flights' table.

g) Create a trigger **RES_TRG** that will ensure that when a new row is inserted into the **RESERVATIONS** table, the flight id is in the **FLIGHTS** table and that the number of seats on this flight, SEATS is greater than 0.

Here are the details of how the trigger should behave:

- If flight id is not in the flights table it should raise application error 'Invalid flight id'.
- If flight id is in the flights table, (for example AC0529) but SEATS = 0, then it should raise application error 'Flight AC0529 has no seats left'.
- If flight id is in the flights table and SEATS > 0, then it should update the appropriate row in flights table by setting SEATS = SEATS – 1 for this flight.

Theory:

- Triggers are stored programs that are fired by Oracle engine automatically when DML Statements like insert, update, delete are executed on the table or some events occur. The code to be executed in case of a trigger can be defined as per the requirement. You can choose the event upon which the trigger needs to be fired and the timing of the execution. The purpose of trigger is to maintain the integrity of information on the database.
- Uses of Triggers are generating some derived column values automatically ,enforcing referential integrity, event logging and storing information on table access, auditing, synchronous replication of tables ,imposing security authorizations ,preventing invalid transactions.
- Syntax for creating a trigger in SQL is:

```
CREATE [OR REPLACE ] TRIGGER trigger_name { BEFORE | AFTER | INSTEAD OF }  
{ INSERT [OR] | UPDATE [OR] | DELETE } [OF col_name] ON table_name  
[REFERENCING OLD AS o NEW AS n] [FOR EACH ROW] WHEN (condition)  
DECLARE Declaration-statements BEGIN Executable-statements EXCEPTION Exception-  
handling-statements END;
```
- CREATE [OR REPLACE] TRIGGER trigger_name – Creates or replaces an existing trigger with the trigger_name.
- {BEFORE | AFTER | INSTEAD OF} – This specifies when the trigger will be executed.
- The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} – This specifies the DML operation. [OF col_name] – This specifies the column name that will be updated. [ON table_name] – This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] – This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE. [FOR EACH ROW] – This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) – This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.
- Example for triggers in SQL is:

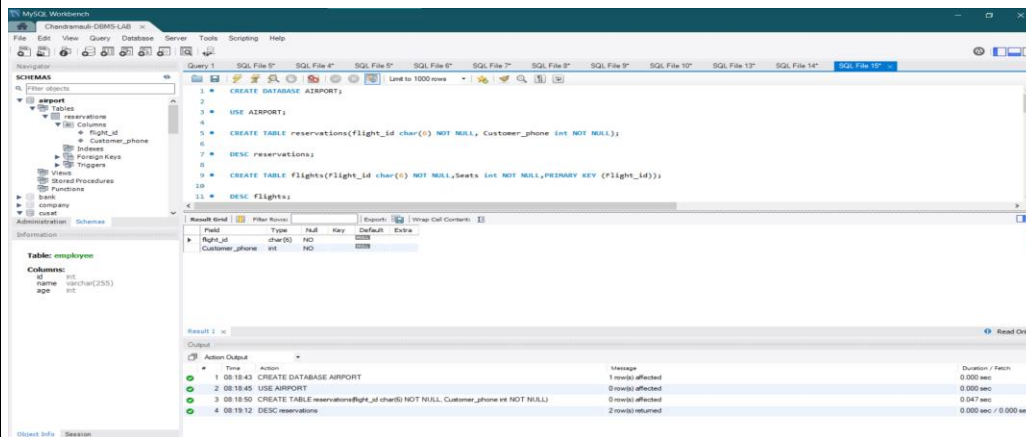
```
CREATE OR REPLACE TRIGGER display_salary_changes BEFORE DELETE OR  
INSERT OR UPDATE ON customers FOR EACH ROW WHEN (NEW.ID > 0)  
DECLARE sal_diff number; BEGIN sal_diff := :NEW.salary - :OLD.salary;  
dbms_output.put_line('Old salary: ' || :OLD.salary); dbms_output.put_line('New salary: '  
|| :NEW.salary); dbms_output.put_line('Salary difference: ' || sal_diff); END; /
```

Result: As a result triggers in SQL is familiarized and output is verified.

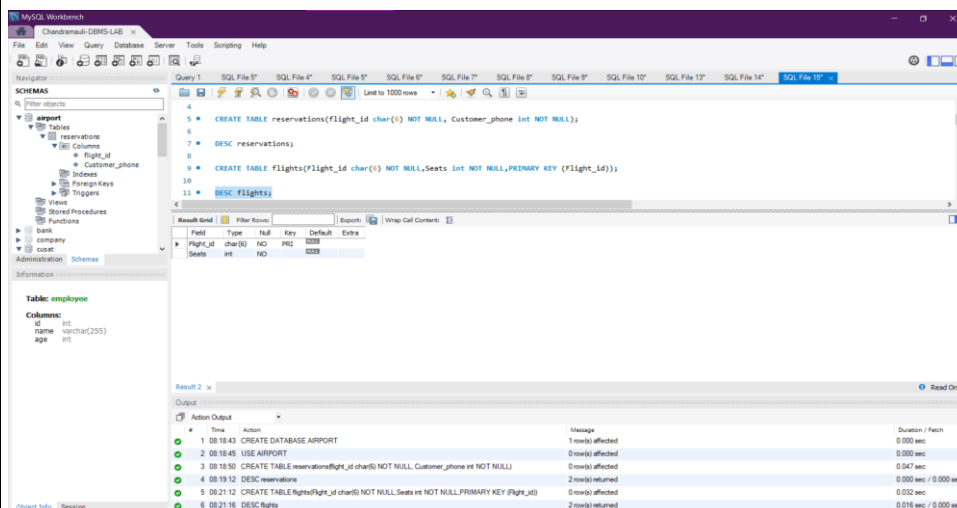
Remarks:(To be filled by faculty)

OUTPUT:

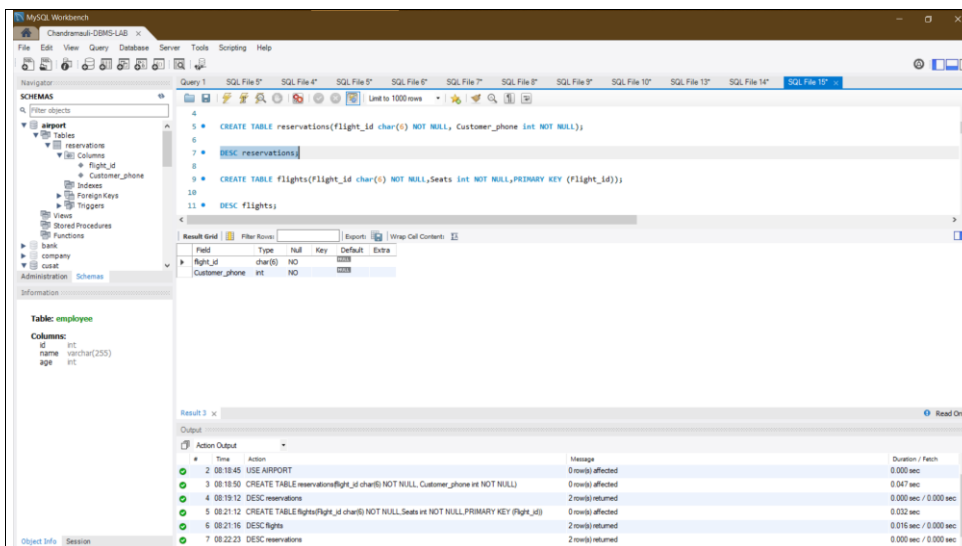
a) Create a table called ‘reservations’ with fields.



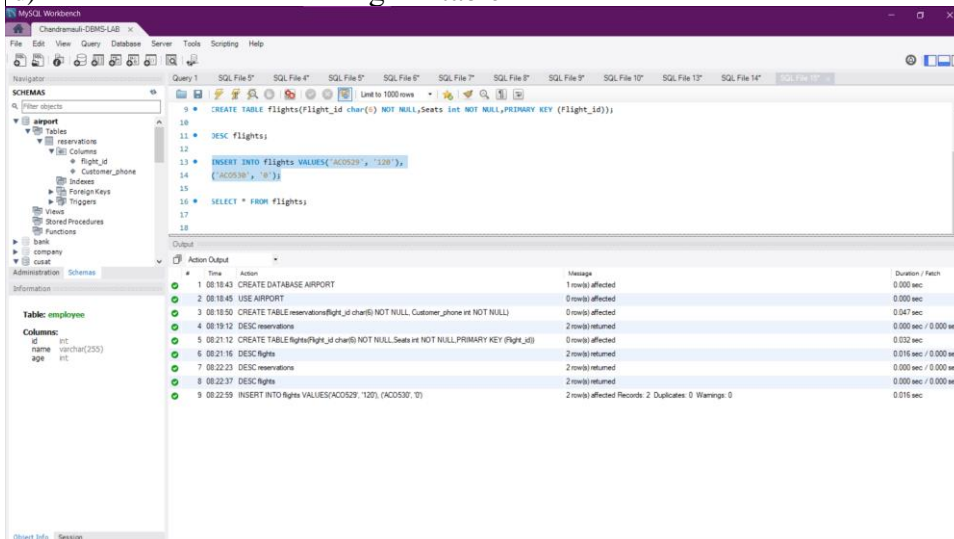
b) Create another table called ‘flights’ with fields. Make ‘flight_id’ the primary key in ‘flights’ table.



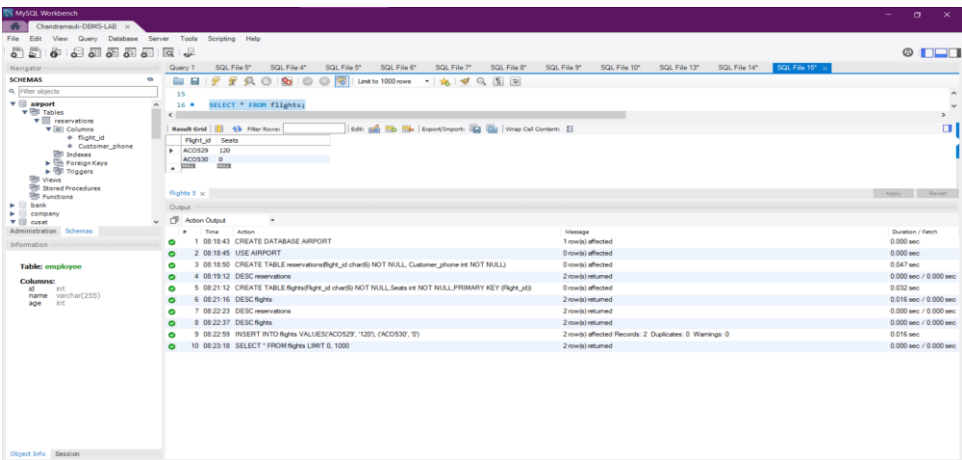
c) Show the structure of ‘reservations’ table



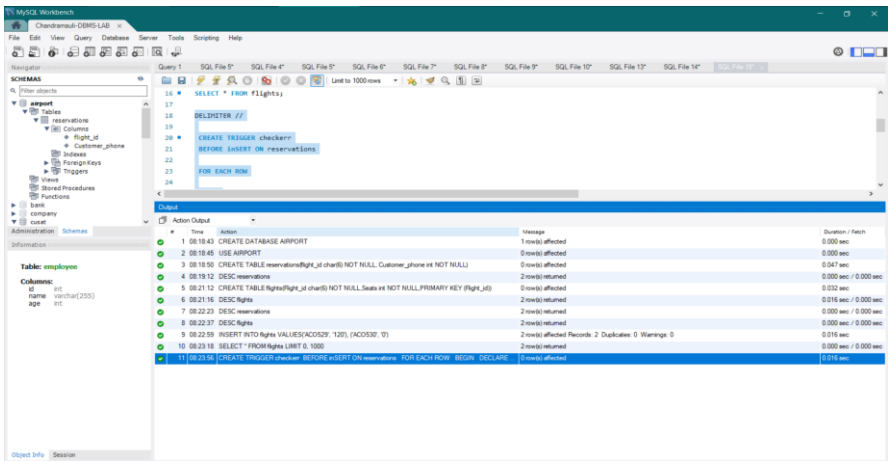
d) Show the structure of 'flights' table



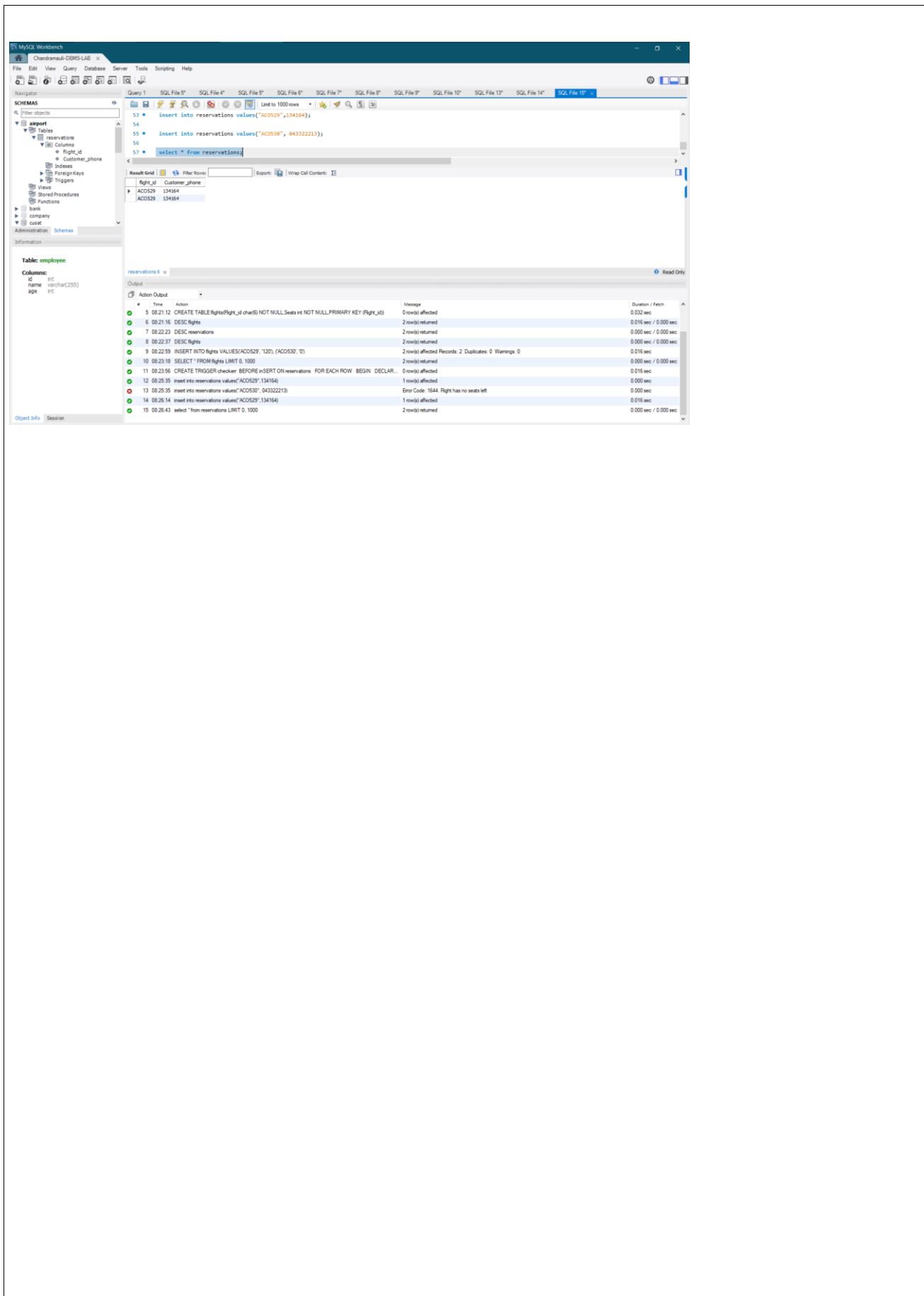
e) Insert the rows into ‘flights’ table.



f) Display all the details of ‘flights’ table



g) Create a trigger **RES_TRG** that will ensure that when a new row is inserted into the **RESERVATIONS** table, the flight id is in the **FLIGHTS** table and that the number of seats on this flight, **SEATS** is greater than 0.



(LEFT SIDE OF A PAGE)