Assignment - 6

Date: | | | |

Title: Write a PL/SQL block of code using parameterized cursor, that will merge the data available in the newly created table.

Problem Statement: Write a PL/SQL block of code using parameterized cursor, that will merge data available in newly created table N_empid with data available in O_empid. If the data in first table already exists then that data should be skipped.

Objectives:
To understand and implement types of cursors with PL/SQL code.

Outcomes:
Students will be able to:
- implement PL/SQL block code.
- implement types of cursors

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 SQL statement. The set of rows the cursor hold is referred as an active set.

Date :

Implicit Cursors:
These are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for statemen Programmers cannot control implicit cursors and the information in it.

In PL/SQL, you can refer to most recent implicit cursors as SQLCursors, which always has attributes such as %.FOUND, %.ISOPEN, %.NOTFOUND, and %.ROWCOUNT. Any SQL cursor attribute will be accessed as sql%.attribute_name.

Explicit Cursors:
Explicit cursors are programmer defined cursors for gaining more control over the context area. Any explicit cursor should be defined in the declaration ~~block~~ ~~of~~ section of PL/SQL block.

Working with an explicit cursor includes the following steps:

- Declaring the cursor for initializing the memory
- Opening the cursor for allocating the memory.
- Fetching the cursor for retrieving data.
- Closing the cursor to release the allocated.

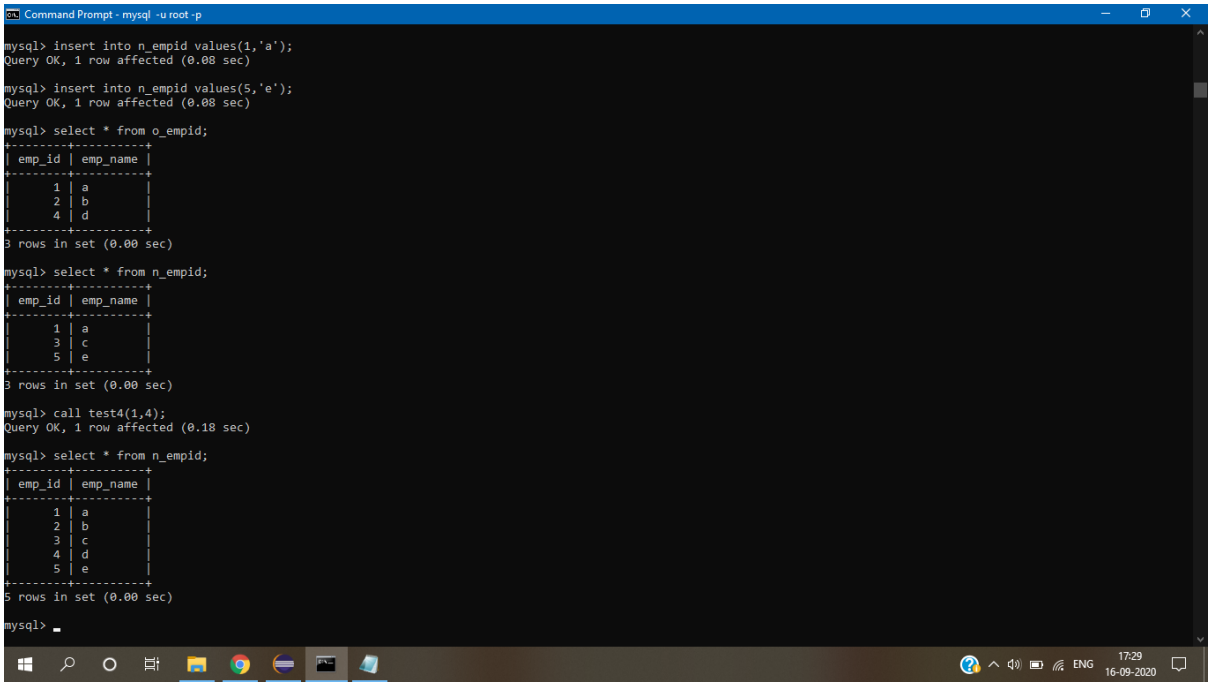
Date :

Procedure:

```
delimiter //
create procedure test4 (min int, max int)
begin
declare id integer;
declare nid integer;
declare name varchar(100) default " ";
declare is_done integer default 0;
declare is_present integer default 0;
declare cpy_cursor cursor for select emp_id,
emp_name from o_empid where emp_id between
min and max;
declare verify_cursor cursor for seleet emp_id
from n_empid;
declare continue handler for not found set
is_done = 1;
open cpy_cursor;
cpy : loop
fetch cpy_cursor into id, name;
if is_done = 1 then leave cpy;
end if;
open verify_cursor into nid;
if is_done = 1 then
set is_done = 0;
close verify_cursor;
leave verify;
end if;
if id = nid then
```

Date:

```
set is_present = 1;
set is_done = 1;
end if;
end loop verify;
if is_present = 0 then
insert into n_enapid values (id, name);
elseif is_present = 1 then
set is_present = 0;
end if;
end loop cpy;
close cpy_cursor;
end //
```



Conclusion:

We have understood concepts of cursors and implemented it successfully using stored procedures.