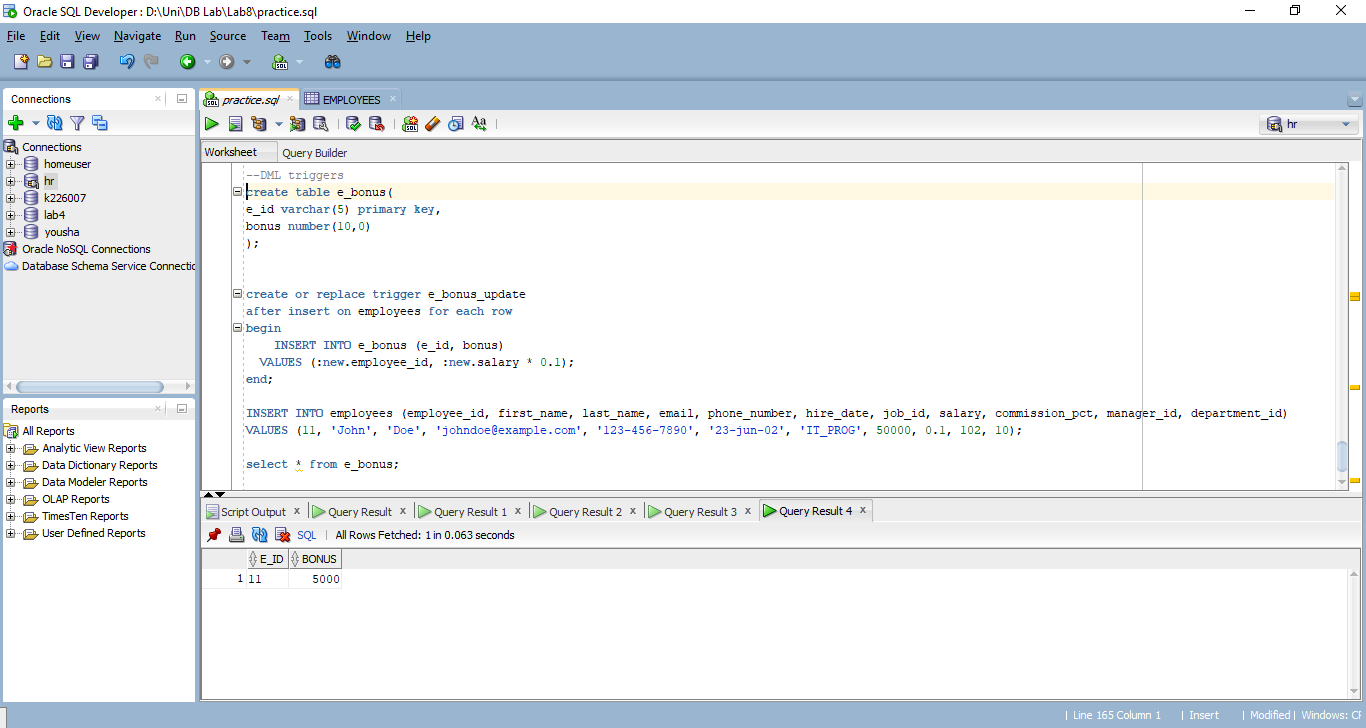
K226007

DBMS LAB 9

BCS-5L

DML TRIGGERS:



create table e\_bonus(

e\_id varchar(5) primary key,

bonus number(10,0)

);

create or replace trigger e\_bonus\_update

after insert on employees for each row

begin

INSERT INTO e\_bonus (e\_id, bonus)

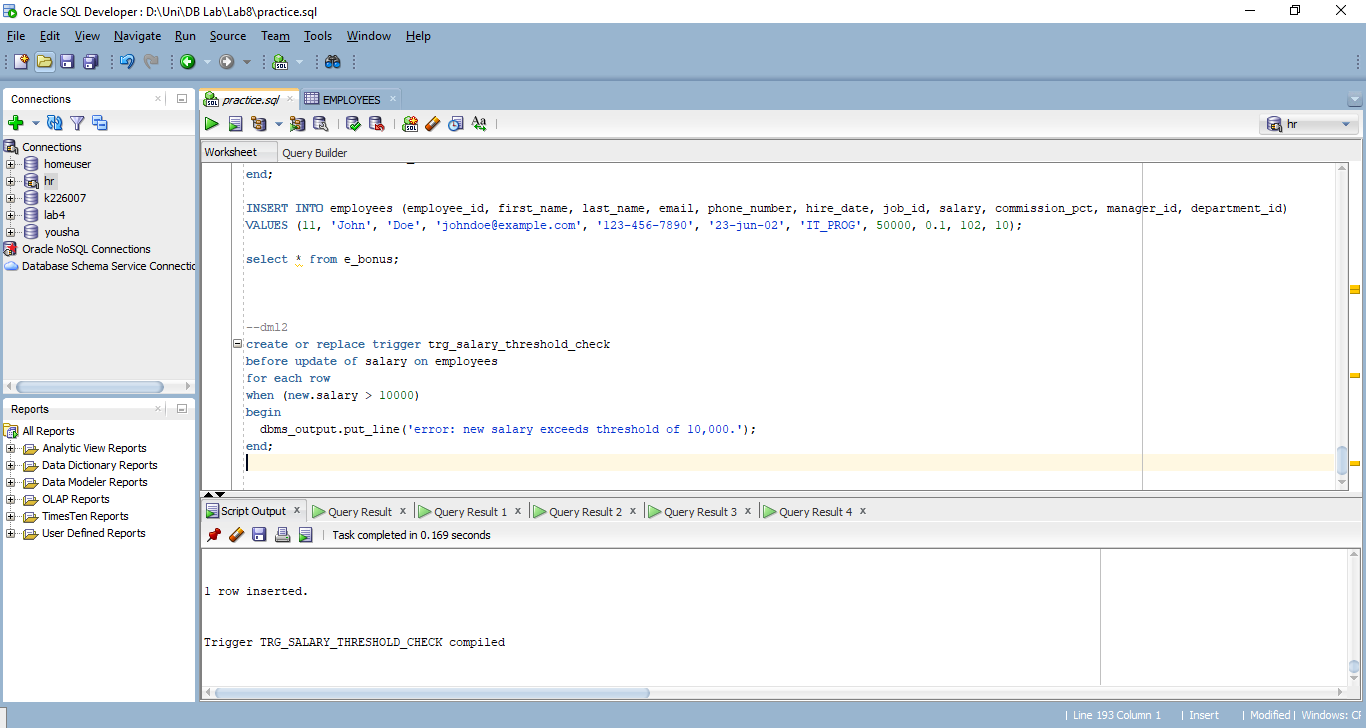
VALUES (:new.employee\_id, :new.salary \* 0.1);

end;

INSERT INTO employees (employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, department\_id)

VALUES (11, 'John', 'Doe', 'johndoe@example.com', '123-456-7890', '23-jun-02', 'IT\_PROG', 50000, 0.1, 102, 10);

select \* from e\_bonus;



create or replace trigger trg\_salary\_threshold\_check

before update of salary on employees

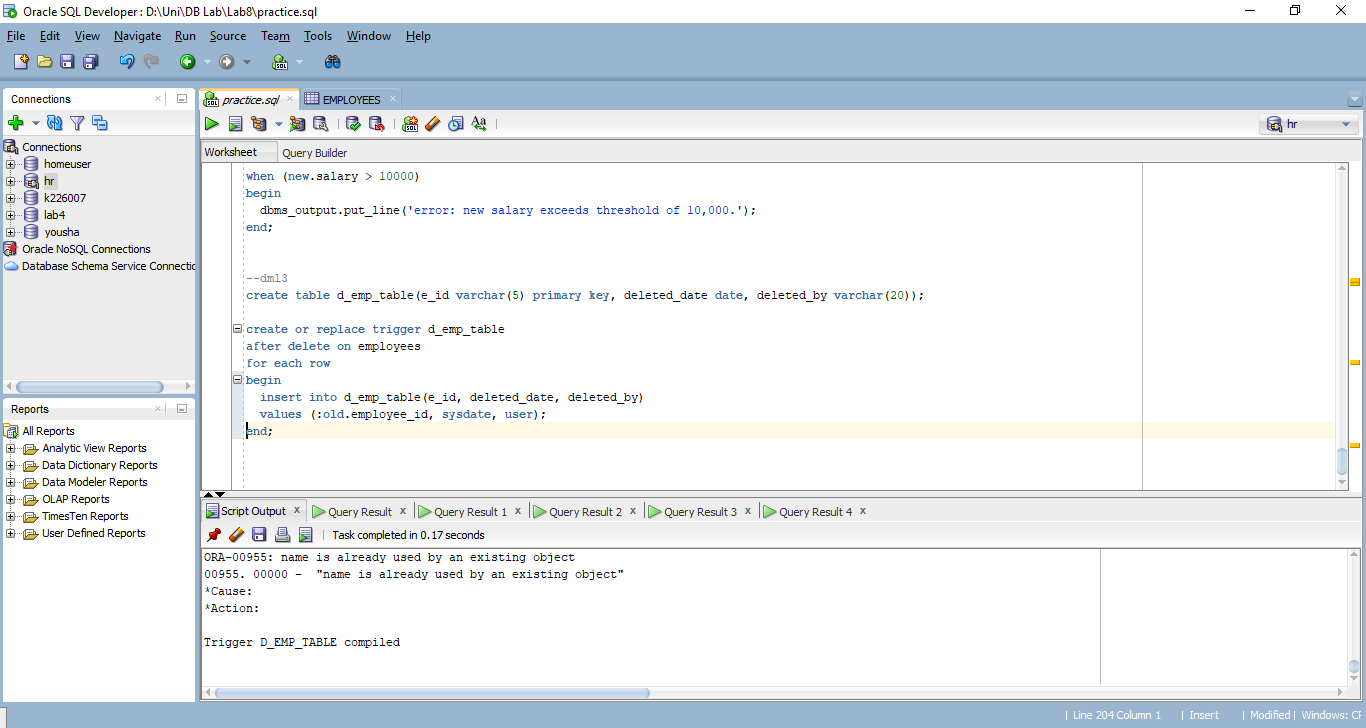
for each row

when (new.salary > 10000)

begin

dbms\_output.put\_line('error: new salary exceeds threshold of 10,000.');

end;



create table d\_emp\_table(e\_id varchar(5) primary key, deleted\_date date, deleted\_by varchar(20));

create or replace trigger d\_emp\_table

after delete on employees

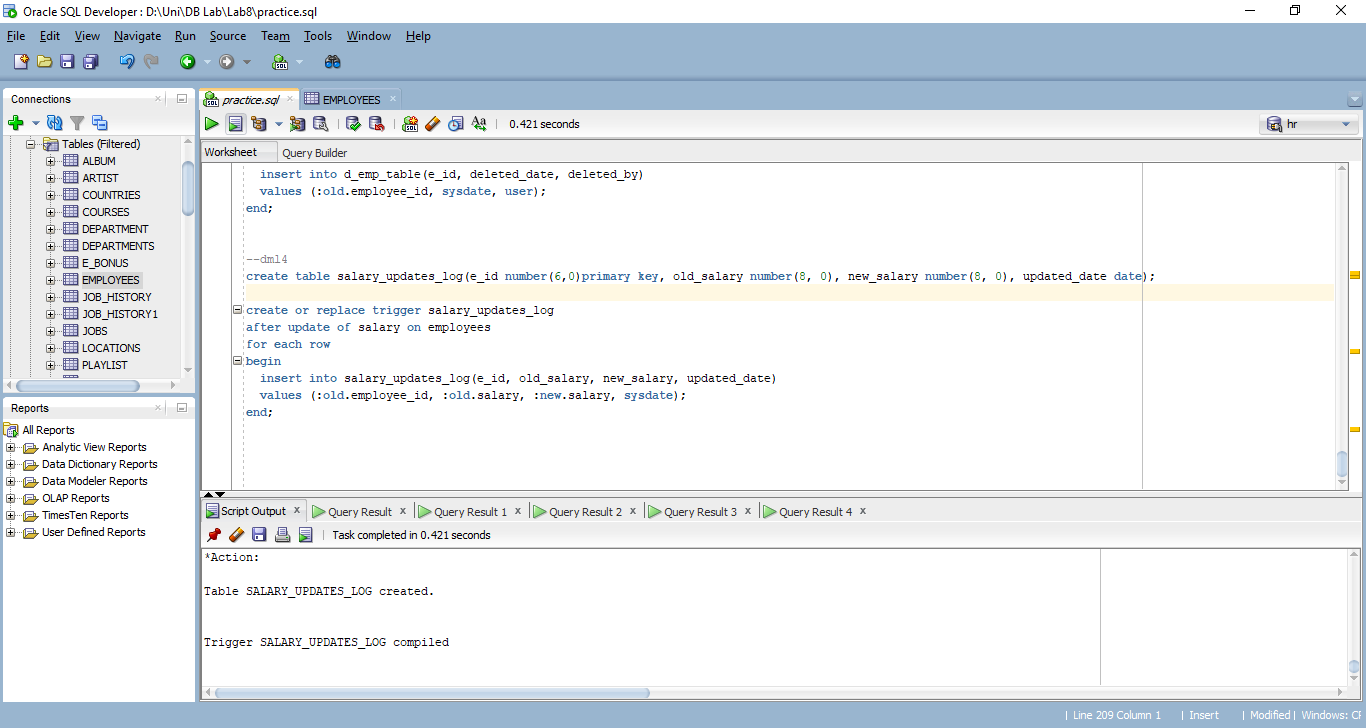
for each row

begin

insert into d\_emp\_table(e\_id, deleted\_date, deleted\_by)

values (:old.employee\_id, sysdate, user);

end;

  
  
create table salary\_updates\_log(e\_id number(6,0)primary key, old\_salary number(8, 0), new\_salary number(8, 0), updated\_date date);

create or replace trigger salary\_updates\_log

after update of salary on employees

for each row

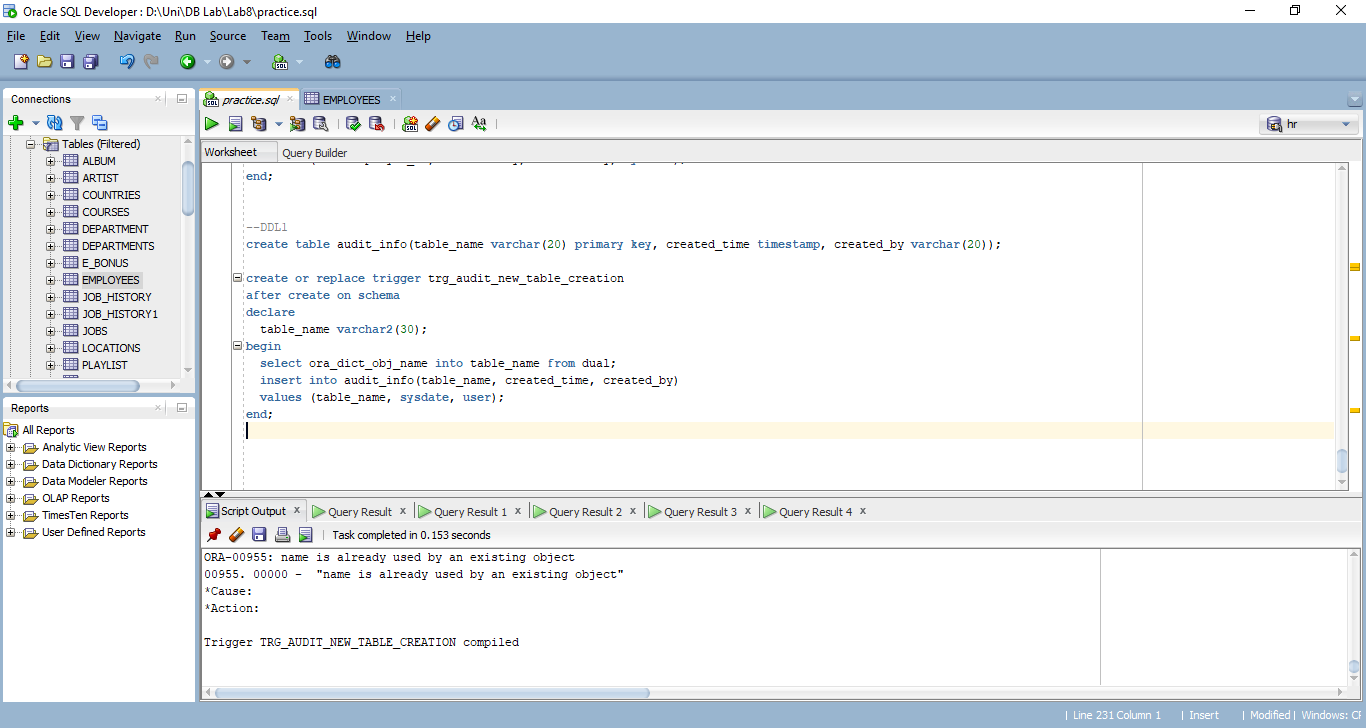
begin

insert into salary\_updates\_log(e\_id, old\_salary, new\_salary, updated\_date)

values (:old.employee\_id, :old.salary, :new.salary, sysdate);

end;

DDL Commands:



create table audit\_info(table\_name varchar(20) primary key, created\_time timestamp, created\_by varchar(20));

create or replace trigger trg\_audit\_new\_table\_creation

after create on schema

declare

table\_name varchar2(30);

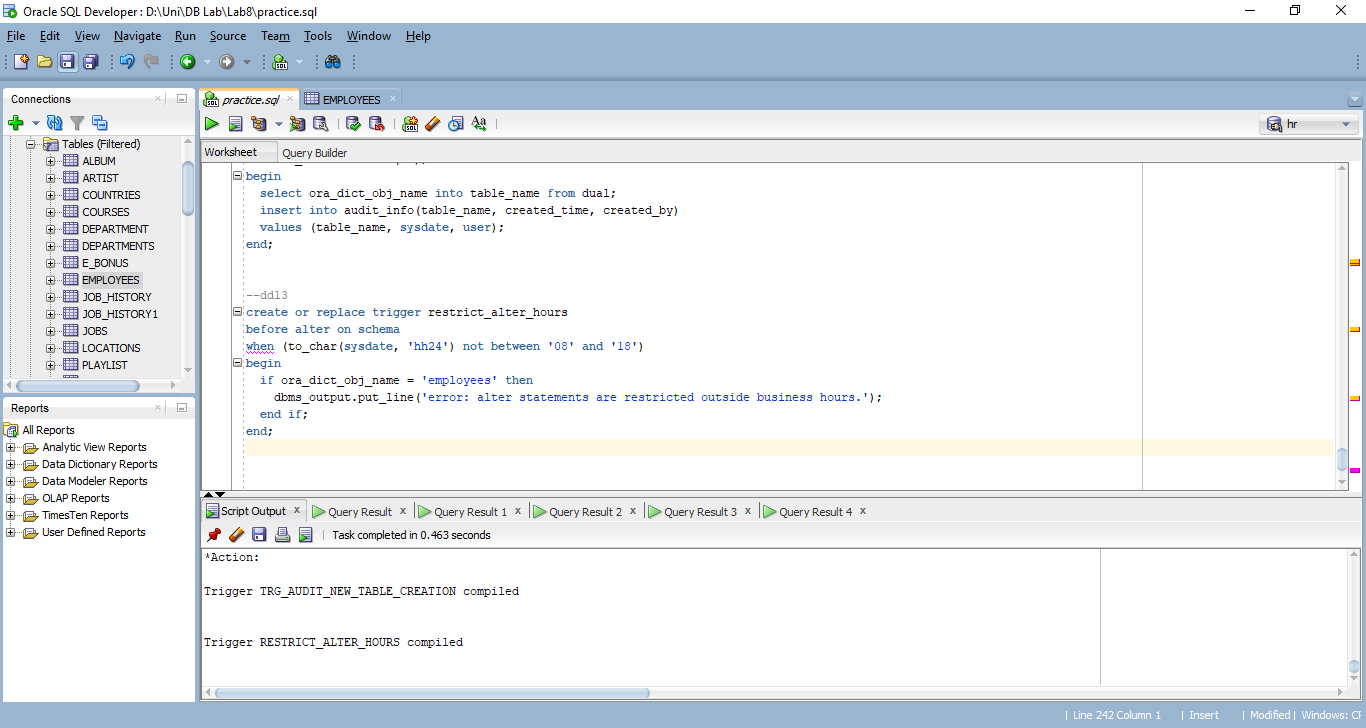
begin

select ora\_dict\_obj\_name into table\_name from dual;

insert into audit\_info(table\_name, created\_time, created\_by)

values (table\_name, sysdate, user);

end;



create or replace trigger restrict\_alter\_hours

before alter on schema

when (to\_char(sysdate, 'hh24') not between '08' and '18')

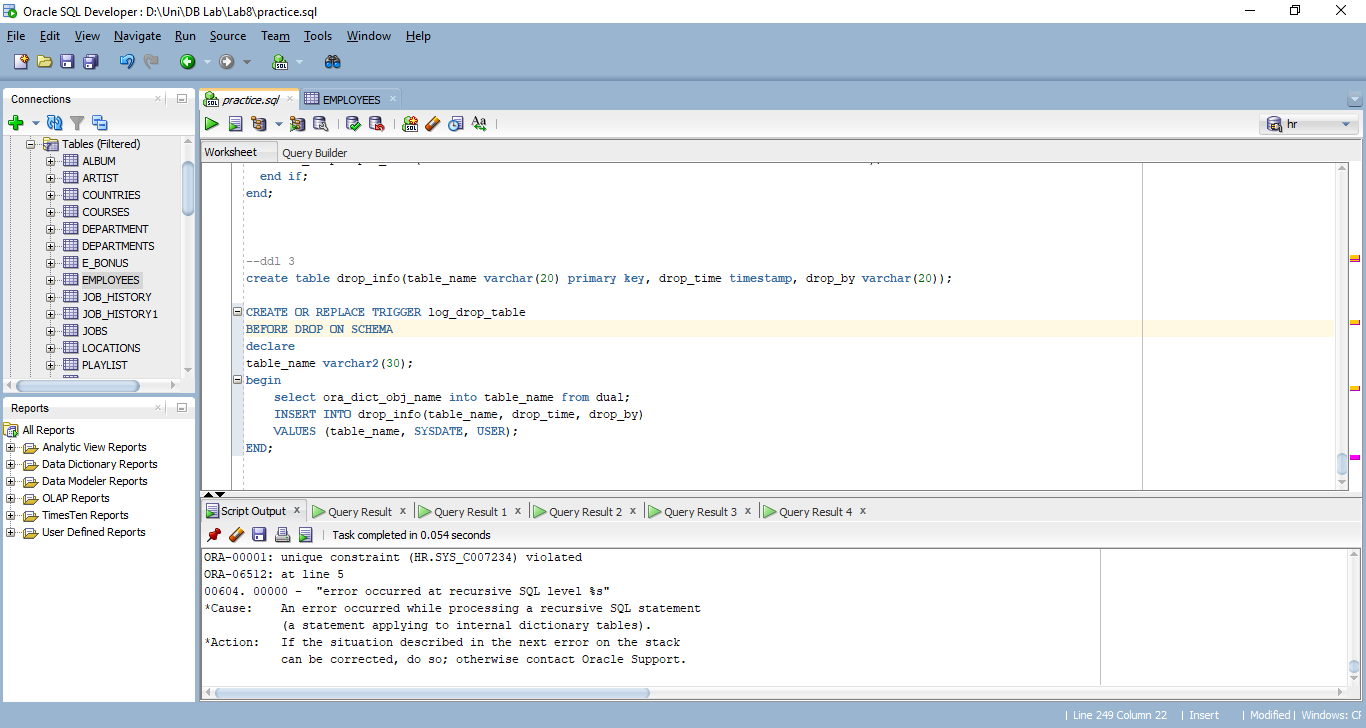
begin

if ora\_dict\_obj\_name = 'employees' then

dbms\_output.put\_line('error: alter statements are restricted outside business hours.');

end if;

end;



create table drop\_info(table\_name varchar(20) primary key, drop\_time timestamp, drop\_by varchar(20));

CREATE OR REPLACE TRIGGER log\_drop\_table

BEFORE DROP ON SCHEMA

declare

table\_name varchar2(30);

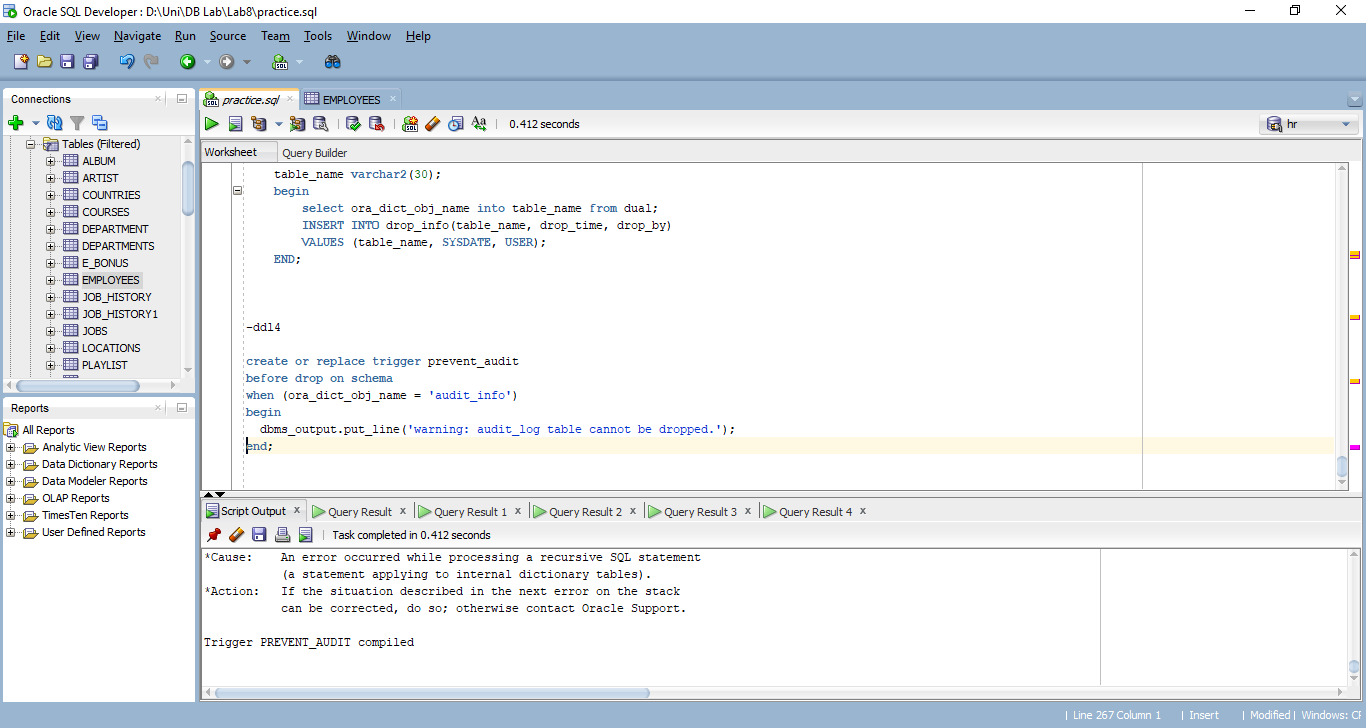
begin

select ora\_dict\_obj\_name into table\_name from dual;

INSERT INTO drop\_info(table\_name, drop\_time, drop\_by)

VALUES (table\_name, SYSDATE, USER);

END;



create or replace trigger prevent\_audit

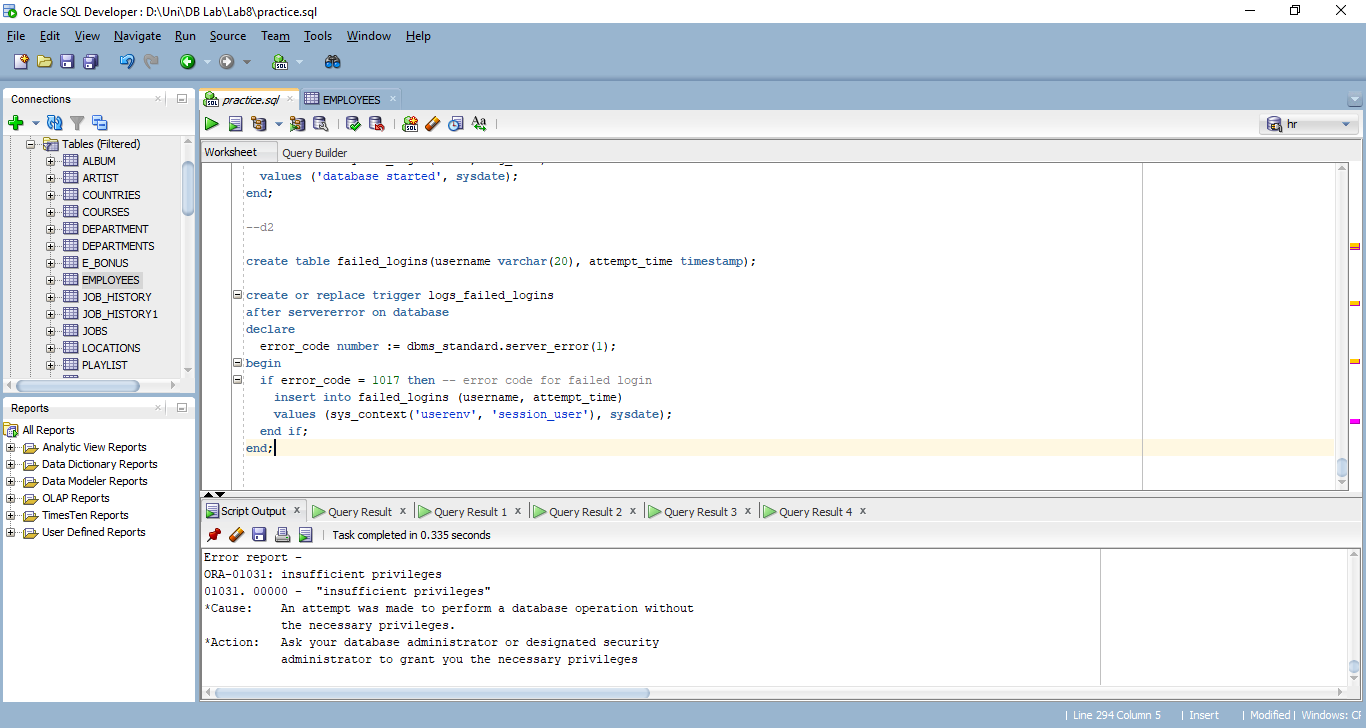
before drop on schema

when (ora\_dict\_obj\_name = 'audit\_info')

begin

dbms\_output.put\_line('warning: audit\_log table cannot be dropped.');

end;



create table failed\_logins(username varchar(20), attempt\_time timestamp);

create or replace trigger logs\_failed\_logins

after servererror on database

declare

error\_code number := dbms\_standard.server\_error(1);

begin

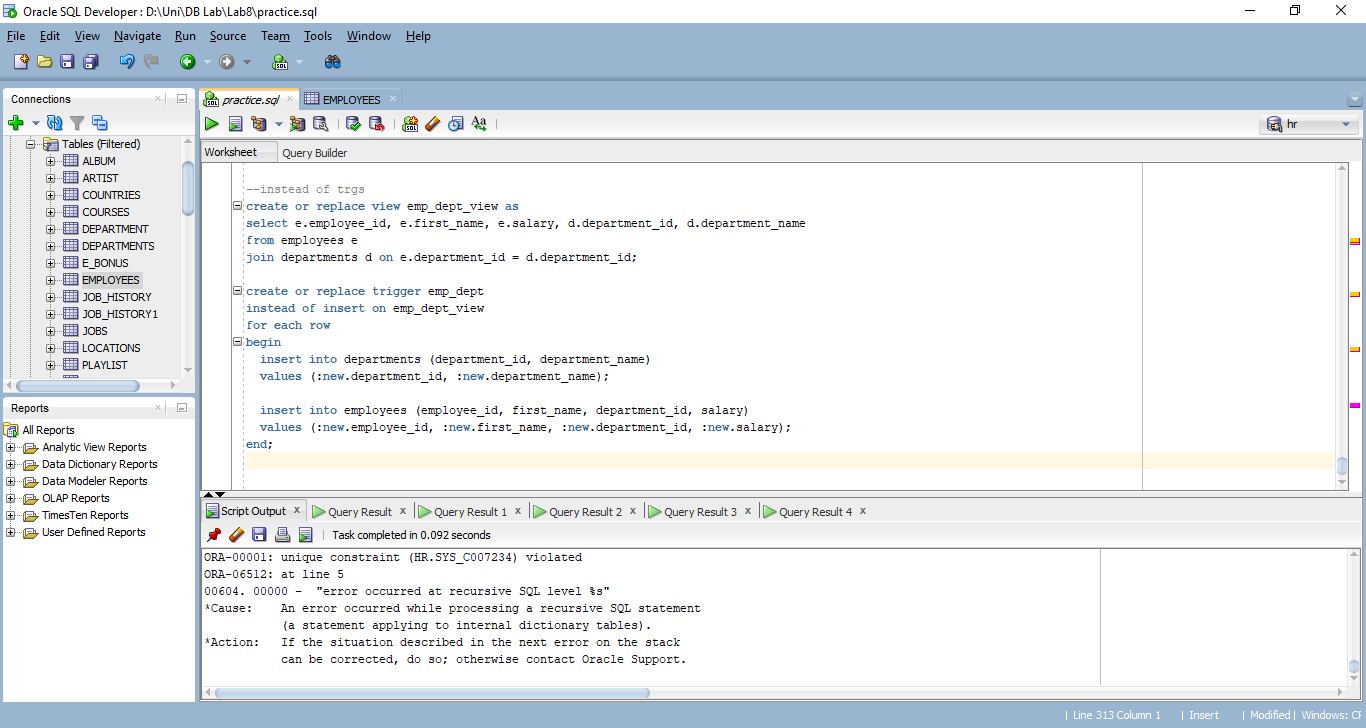
if error\_code = 1017 then -- error code for failed login

insert into failed\_logins (username, attempt\_time)

values (sys\_context('userenv', 'session\_user'), sysdate);

end if;

end;



create or replace view emp\_dept\_view as

select e.employee\_id, e.first\_name, e.salary, d.department\_id, d.department\_name

from employees e

join departments d on e.department\_id = d.department\_id;

create or replace trigger emp\_dept

instead of insert on emp\_dept\_view

for each row

begin

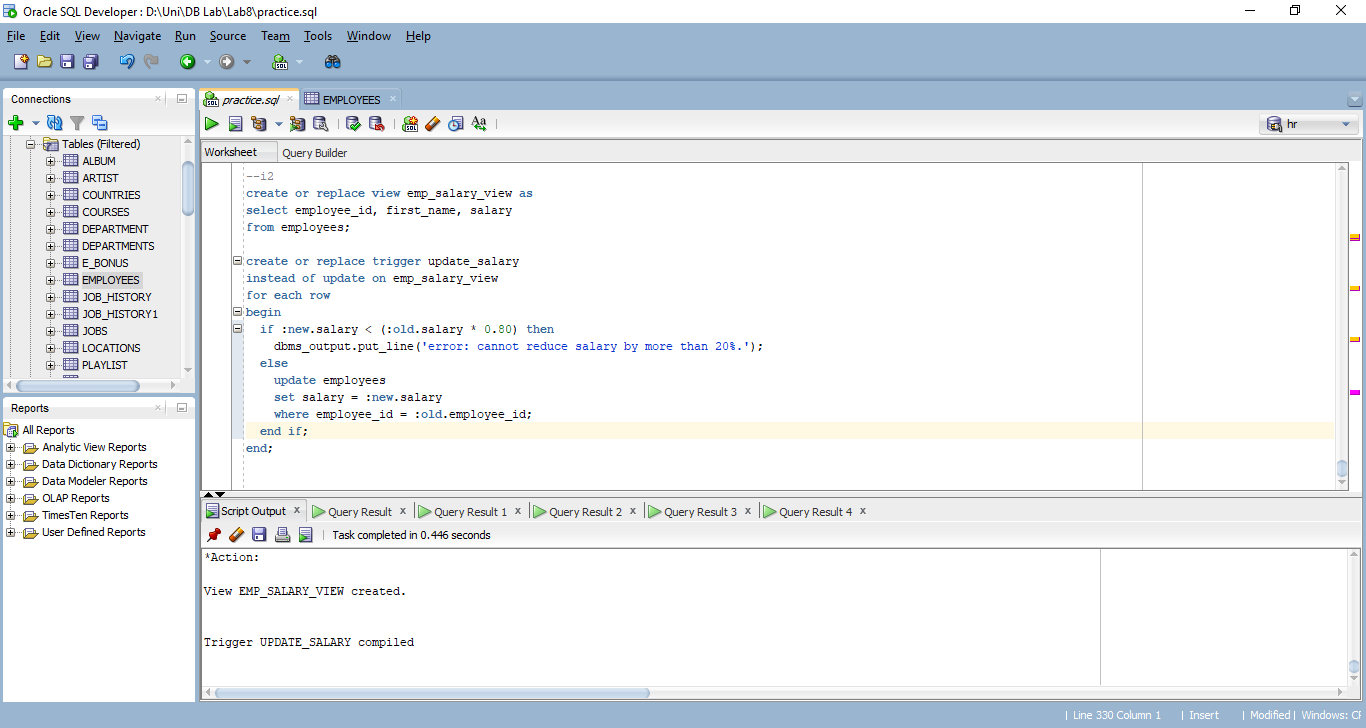
insert into departments (department\_id, department\_name)

values (:new.department\_id, :new.department\_name);

insert into employees (employee\_id, first\_name, department\_id, salary)

values (:new.employee\_id, :new.first\_name, :new.department\_id, :new.salary);

end;



create or replace view emp\_salary\_view as

select employee\_id, first\_name, salary

from employees;

create or replace trigger update\_salary

instead of update on emp\_salary\_view

for each row

begin

if :new.salary < (:old.salary \* 0.80) then

dbms\_output.put\_line('error: cannot reduce salary by more than 20%.');

else

update employees

set salary = :new.salary

where employee\_id = :old.employee\_id;

end if;

end;