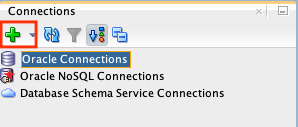
# Using SQL Developer

1. Open SQL Developer and click on + icon at upper left.



1. On new database connection window provide connection name, username, password, hostname and service name.

# 

# 

1. Click on Test and check if it is a success or not.

# 

# Click on Save, You will notice connection Name and connection details on left panel. Click on connect.

# 

# Connection window will be popped up and provide user/pass and click on OK.

# 

# 

# 

# Workspace panel will be displayed once you login to the database.

# 

# DDL SQL Commands

-- tbl\_course

CREATE TABLE tbl\_course(

course\_id number constraint pk\_cid primary key,

course\_name varchar2(32) not null,

course\_duration number not null,

course\_fee number(7,2) not null,

pass\_mark number(2) not null

);

comment on table tbl\_course is 'Table to store course information';

-- tbl\_teacher

CREATE TABLE tbl\_teacher(

teacher\_id number constraint pk\_tid primary key,

teacher\_name varchar2(32) not null

);

comment on table tbl\_teacher is 'Table to store teacher information';

-- tbl\_student

CREATE TABLE tbl\_student(

student\_id number constraint pk\_sid primary key,

student\_name varchar2(32) not null

);

comment on table tbl\_student is 'Table to store student information';

-- tbl\_course\_enrollment

CREATE TABLE tbl\_course\_enrollment(

course\_id number,

teacher\_id number,

student\_id number,

start\_date date,

completion\_date date,

constraint fk\_cid\_enroll foreign key(course\_id) references tbl\_course(course\_id),

constraint fk\_tid\_enroll foreign key(teacher\_id) references tbl\_teacher(teacher\_id),

constraint fk\_sid\_enroll foreign key(student\_id) references tbl\_student(student\_id)

);

comment on table tbl\_course\_enrollment is 'Table to store student enrolled for particular course';

-- tbl\_exam

CREATE TABLE tbl\_exam(

course\_id number,

student\_id number,

scored\_mark number(3,1),

constraint fk\_cid\_exam foreign key(course\_id) references tbl\_course(course\_id),

constraint fk\_sid\_exam foreign key(student\_id) references tbl\_student(student\_id)

);

comment on table tbl\_exam is 'Table to store student exam information';

-- tbl\_payment

CREATE TABLE tbl\_payment(

student\_id number,

paid\_date date not null,

paid\_amount number(7,2) not null,

constraint fk\_sid\_payment foreign key(student\_id) references tbl\_student(student\_id)

);

comment on table tbl\_payment is 'Table to store student payment information';

-- tbl\_fee

CREATE TABLE tbl\_fee(

student\_id number,

due\_amount number(7,2) not null,

due\_date date not null,

constraint fk\_sid\_fee foreign key(student\_id) references tbl\_student(student\_id)

);

comment on table tbl\_fee is 'Table to store students due fee information';

--Trig\_fee\_deduct

Create or replace trigger trig\_fee\_deduct

After insert on tbl\_payment

For each row

Begin

Update tbl\_fee set due\_amount=due\_amount - :new.paid\_amount where student\_id=:new.student\_id;

End;

/

--Trig\_fee\_Add

create or replace trigger trig\_fee\_add

before insert on tbl\_course\_enrollment

For each row

Declare

PRAGMA AUTONOMOUS\_TRANSACTION;

v\_fee number(7,2);

v\_duration number;

Begin

Select course\_fee,course\_duration into v\_fee,v\_duration from tbl\_course where course\_id=:new.course\_id;

merge into tbl\_fee f

using (select :new.student\_id student\_id from dual) s

on(s.student\_id=f.student\_id)

when matched then

Update set due\_amount=due\_amount + v\_fee

when not matched then

insert (student\_id,due\_amount,due\_date)

values(:new.student\_id,v\_fee,sysdate+3);

:new.completion\_date:=:new.start\_date + v\_duration;

commit;

End;

/

-- seq\_teacher

Create sequence seq\_teacher start with 1;

-- seq\_course

Create sequence seq\_course start with 1;

-- seq\_student

Create sequence seq\_student start with 1;

-- proc\_due\_date

create or replace procedure proc\_due\_date

as

cursor cur is select student\_id, student\_name,due\_amount from tbl\_fee join tbl\_student using(student\_id) where due\_date >= sysdate;

begin

for c in cur

loop

dbms\_output.put\_line('Student Name: ' || c.student\_name || ' Due Amount: ' || c.due\_amount);

end loop;

end;

/

-- func\_result

Create or replace function func\_result(v\_student\_id number,v\_course\_id number)

Return varchar2

as

v\_marks number(3);

v\_pass\_marks number(2);

v\_result varchar2(4);

Begin

Select scored\_mark into v\_marks from tbl\_exam where course\_id=v\_course\_id and student\_id=v\_student\_id;

Select pass\_mark into v\_pass\_marks from tbl\_course where course\_id=v\_course\_id;

If v\_marks >=v\_pass\_marks then

v\_result:='PASS';

else

v\_result:='FAIL';

End if;

Return v\_result;

End;

/

-- vw\_coruse\_enrolled

Create view vw\_course\_enrolled as

select s.student\_name,c.course\_name,e.start\_date

from tbl\_student s join tbl\_course\_enrollment e

using(student\_id)

join tbl\_course c

using(course\_id);

select \* from vw\_course\_enrolled;

# DML Insert commands.

insert into tbl\_course(course\_id,course\_name,course\_duration,course\_fee,pass\_mark)

values(SEQ\_COURSE.nextval,'ORACLE',90,7000,60);

insert into tbl\_course(course\_id,course\_name,course\_duration,course\_fee,pass\_mark)

values(SEQ\_COURSE.nextval,'JAVA',120,9000,50);

insert into tbl\_course(course\_id,course\_name,course\_duration,course\_fee,pass\_mark)

values(SEQ\_COURSE.nextval,'C/C++',45,6000,50);

select \* from tbl\_course;

commit;

insert into tbl\_teacher(teacher\_id,teacher\_name)

values(SEQ\_TEACHER.nextval,'Ram Shrestha');

insert into tbl\_teacher(teacher\_id,teacher\_name)

values(SEQ\_TEACHER.nextval,'Laxman Pandey');

select \* from tbl\_teacher;

commit;

insert into tbl\_student(student\_id,student\_name)

values(SEQ\_STUDENT.nextval,'Raju Bajracharya');

insert into tbl\_student(student\_id,student\_name)

values(SEQ\_STUDENT.nextval,'Krishna Gurung');

select \* from tbl\_student;

commit;

insert into tbl\_course\_enrollment(course\_id,teacher\_id,student\_id,start\_date,completion\_date)

values(1,1,1,sysdate,null);

insert into tbl\_course\_enrollment(course\_id,teacher\_id,student\_id,start\_date,completion\_date)

values(2,1,2,sysdate,null);

select \* from tbl\_course\_enrollment;

commit;

select \* from tbl\_fee;

insert into tbl\_payment(student\_id,paid\_date,paid\_amount)

values(1,sysdate,1000);

insert into tbl\_payment(student\_id,paid\_date,paid\_amount)

values(2,sysdate,2000);

select \* from tbl\_payment;

select \* from tbl\_fee;

commit;

insert into tbl\_exam(course\_id,student\_id,scored\_mark)

values(1,1,50);

insert into tbl\_exam(course\_id,student\_id,scored\_mark)

values(2,2,50);

commit;

select \* from tbl\_exam;

select func\_result(2,2) from dual;

# SQL Join

select s.student\_name,c.course\_name, e.scored\_mark,func\_result(e.student\_id,e.course\_id)

from tbl\_student s join tbl\_exam e

on(s.student\_id=e.student\_id)

join tbl\_course c

on (e.course\_id=c.course\_id);

# Create user commands.

create user dilli identified by dilli

Quota unlimited on users;

grant connect, resource to dilli;

grant create view, create synonym, create trigger to dilli;

grant DATAPUMP\_EXP\_FULL\_DATABASE, DATAPUMP\_IMP\_FULL\_DATABASE to dilli;

Grant select any table to dilli;

Grant exp\_full\_database, imp\_full\_database to dilli;

Create directory dir\_iims as '/home/oracle/iims';

Grant read,write on directory dir\_iims to public;

drop table tbl\_course\_enrollment cascade constraint purge;

drop table tbl\_exam cascade constraint purge;

drop table tbl\_fee cascade constraint purge;

drop table tbl\_payment cascade constraint purge;

drop table tbl\_student cascade constraint purge;

drop table tbl\_teacher cascade constraint purge;

drop table tbl\_course cascade constraint purge;