**Function call in Hibernate**

Creating Function In Postgres

CREATE OR REPLACE FUNCTION public.**fn\_insertuniv1**(

university\_id bigint,

country character varying,

name character varying)

**RETURNS integer AS**

$BODY$

DECLARE

err\_context text;

BEGIN

INSERT INTO university

(university\_id,country, name)

VALUES (university\_id,country, name);

**return university\_id;**

exception

when others then

GET STACKED DIAGNOSTICS err\_context = PG\_EXCEPTION\_CONTEXT;

RAISE INFO 'Error Name:%' ,SQLERRM;

RAISE INFO 'Error State:%' ,SQLSTATE;

RAISE INFO 'Error Context:%', err\_context;

return -1;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DaoImpl\*\*\*\*\*\*\*\*

@SuppressWarnings("deprecation")

@Override

public void createUniversity(University univ) {

//Session session= (Session)entityManager.getDelegate();

Session session= entityManager.unwrap(Session.class);

Object i=sessionFactory.getCurrentSession().createSQLQuery("select public.fn\_applcnt\_reg('"

+applicant.getApplicant\_name()+"','"+applicant.getEmail\_id()+"',"+applicant.getMobile\_number()+",'"+applicant.getPasswd()+

"',"+applicant.getState().getState\_id()+","+applicant.getCity().getCity\_id()+

")").getSingleResult();

System.out.println(i);

}

\*\*\*\*\*\*\*\*\*\*\*Controller\*\*\*\*\*\*\*\*\*\*\*

@RequestMapping(value = "/createUniversity", method = RequestMethod.POST, produces = "application/json", consumes = "application/json")

public void createUniversity(@RequestBody University univ) {

studentServiceImpl.createUniversity(univ);}

\*\*\*\*\*\*\*\*ServiceImpl\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public void createUniversity(University univ) {

studentDao.createUniversity(univ);

}

http://wiki.eclipse.org/EclipseLink/Examples/JPA/StoredProcedures

Multiple table Insert through Function

CREATE OR REPLACE FUNCTION public.fn\_applcnt\_reg(applicant\_name text, email text, phone\_no bigint, passwd text, state\_id integer, city\_id integer) RETURNS integer

LANGUAGE plpgsql

AS $function$

/\*

\* Author : Ram

\* Date : July 24th 2018

\* Version: 1.0

\* Purpose: Sample code to validate Registration screen in PostgreSQL.

\*/

DECLARE

v\_id integer;

p\_id integer;

err\_context text;

BEGIN

INSERT INTO public.applicant\_registration(applicant\_name, email\_id, mobile\_number, state\_id, city\_id)

VALUES(applicant\_name, email, phone\_no, state\_id, city\_id) RETURNING user\_id INTO v\_id;

INSERT INTO public.passwords(user\_id, passwd, password\_question, password\_answer)

VALUES(v\_id, passwd, '', '') RETURNING password\_id INTO p\_id;

return p\_id;

exception

when others then

GET STACKED DIAGNOSTICS err\_context = PG\_EXCEPTION\_CONTEXT;

RAISE INFO 'Error Name:%' ,SQLERRM;

RAISE INFO 'Error State:%' ,SQLSTATE;

RAISE INFO 'Error Context:%', err\_context;

return -1;

END;

$function$;

**EntityManager is JPA Specific**

**Session factory is hibernate based**

Creating Session From

ex:

Session session = (Session) entityManager.getDelegate();

OR

Session session = entityManager.unwrap(Session.class);

Select Query Function :

CREATE OR REPLACE FUNCTION get\_stuff(stateid INT)

RETURNS TABLE(st\_id INT,st\_name character varying) AS $$

BEGIN

RETURN QUERY

SELECT \* from state

WHERE state\_id=stateid;

END;

$$ LANGUAGE plpgsql;

SELECT st\_name from get\_stuff(1);

create or replace function fn\_ins\_app(applicantid bigint,username varchar,email varchar,phone bigint,city varchar,state varchar,password varchar)

returns integer as

$body$

declare

begin

insert into applicant(applicantid,username,email,phone,city,state,password)

values(applicantid,username,email,phone,city,state,password);

return applicantid;

end;

$body$

language plpgsql volatile;

select from fn\_ins\_app(10,'sri','sri@sr.com',4566774,'atp','ap','sri');

---------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION fn\_app()

RETURNS SETOF applicant AS

$BODY$

DECLARE

BEGIN

RETURN QUERY

SELECT \* FROM APPLICANT;

END;

$BODY$

LANGUAGE plpgsql VOLATILE;

select \* from fn\_app();

--------------------------------------------------------------------------

create or replace function fn\_del\_app3(appid bigint)

returns integer as

$body$

begin

delete from applicant where applicantid=appid;

return appid;

end

$body$

language plpgsql volatile;

select fn\_del\_app3(90);