DBS1 S17 COURSE ASSIGNMENT – PART 2

JAKUB LEMKA – 249817

2.1.POPULATING THE DATABASE

According to the command most of the data was taken from three IMDB sites concerning three given movies. The following INSERT statements were used to insert the data in the database:

TABLE PERSON:

```
INSERT INTO person (person id,
                    name,
                    birthday,
                     sex,
                    birhplace)
     VALUES (1,
             'Matt Damon',
                   '1970-10-08',
             'M',
             'Cambridge');
INSERT INTO person (person_id,
                    name,
                    birthday,
                     sex,
                    birhplace)
     VALUES (2,
             'Julia Stiles',
                   '1981-03-28',
             'F',
             'New York ');
INSERT INTO person (person_id,
                    name,
                    birthday,
                     sex,
                    birhplace)
     VALUES (3,
             'Alicia Vikander',
                   '1988-10-03',
             'F',
             'Gothenburg');
INSERT INTO person (person id,
                    name,
                    birthday,
                     sex,
                    birhplace)
     VALUES (4,
             'Jennifer Lawrence',
                   '1990-08-15',
             'F'
             'Louisville');
```

```
INSERT INTO person (person_id,
                    name,
                    birthday,
                    sex,
                    birhplace)
    VALUES (5,
            'Gary Ross',
                   '1956-11-03',
             'M',
             'Los Angeles');
                                TABLE STATUS:
INSERT INTO status (status id,
                   name)
     VALUES (1,
             'Released');
INSERT INTO status (status id,
                   name)
    VALUES (2,
             'Completed');
INSERT INTO status (status id,
                    name)
    VALUES (3,
             'Post-production');
INSERT INTO status (status_id,
                   name)
     VALUES (4,
             'Filming');
INSERT INTO status (status_id,
                    name)
     VALUES (5,
             'Announced');
                          TABLE PRODUCTION COMPANY
INSERT INTO production_company (production_id,
                   name,
                             country)
     VALUES (1,
             'MP BETA Productions',
                   'USA');
INSERT INTO production company (production id,
                    name,
                             country)
```

'The Kennedy/Marshall Company',

'USA');

VALUES (2,

```
INSERT INTO production_company (production_id,
                    name,
                              country)
     VALUES (3,
             'Ludlum Entertainment',
                   'USA');
INSERT INTO production_company (production_id,
                    name,
                              country)
     VALUES (4,
             'Perfect World Pictures',
                   'USA');
INSERT INTO production company (production id,
                    name,
                              country)
     VALUES (5,
             'Captivate Entertainment',
                   'USA');
INSERT INTO production company (production id,
                     name,
                              country)
     VALUES (6,
             'Pearl Street',
                    'USA');
INSERT INTO production company (production id,
                    name,
                              country)
     VALUES (7,
             'Color Force',
                    'USA');
                                  TABLE MOVIE
INSERT INTO movie (movie id,
                     title,
                              production_year,
                              status id)
     VALUES (1,
             'The Bourne Ultimatum',
                   2007,
                   1);
INSERT INTO movie (movie id,
                     title,
                              production_year,
                              status_id)
     VALUES (2,
              'Jason Bourne',
                   2016,
                   1);
```

```
INSERT INTO movie (movie id,
                    title,
                              production year,
                              status id)
     VALUES (3,
             'The Hunger Games',
                   2012,
                   1);
                                 TABLE IMAGES:
INSERT INTO images (url,
                              description,
                              movie id,
                    person id)
     VALUES ('http://www.hungergamesdwtc.net/wp-
content/uploads/2014/02/The-Hunger-Games-Poster.jpg',
             'The Hunger Games Poster',
                   3,
                   4);
INSERT INTO images (url,
                              description,
                              movie id,
                    person id)
     VALUES
('http://www.impawards.com/2007/posters/bourne ultimatum ver4.jpg',
             'The Bourne Ultimatum Poster',
                   1,
                   1);
INSERT INTO images (url,
                              description,
                              movie id,
                    person id)
     VALUES ('http://cdn1-www.comingsoon.net/assets/uploads/gallery/jason-
bourne/jason_bourne_ver3_xlg.jpg',
             'Jason Bourne Poster',
                   2,
                   1);
INSERT INTO images (url,
                              description,
                              movie id,
                    person_id)
     VALUES ('http://static.srcdn.com/wp-content/uploads/Matt-Damon-as-
Jason-Bourne-in-The-Bourne-Ultimatum.jpg',
             'Matt Damon in Bourne Ultimatum',
                   1,
                   1);
INSERT INTO images (url,
                              description,
                              movie id,
                    person id)
     VALUES
('http://i.dailymail.co.uk/i/pix/2015/05/10/11/122DB1D8000005DC-3075487-
image-a-1 1431253487957.jpg',
             'Jennifer Lawrence in Hunger Games',
                   3,
                   4);
```

TABLE MOVIE COMPANY

```
INSERT INTO movie_company (movie_id,
                company_id)
    VALUES (1,
           1);
INSERT INTO movie_company (movie_id,
            company_id)
    VALUES (1,
           2);
INSERT INTO movie_company (movie_id,
           company id)
    VALUES (1,
           3);
INSERT INTO movie company (movie id,
                  company id)
    VALUES (2,
           2);
INSERT INTO movie company (movie id,
                 company id)
    VALUES (2,
           4);
INSERT INTO movie_company (movie_id,
                 company id)
    VALUES (1,
           5);
INSERT INTO movie_company (movie_id,
            company_id)
    VALUES (2,
         6);
INSERT INTO movie company (movie id,
                company id)
    VALUES (3,
           7);
                               TABLE ROLE
INSERT INTO role (role id,
                role name)
    VALUES (1,
           'Director');
INSERT INTO role (role id,
            role name)
    VALUES (2,
            'Actor');
```

TABLE PERSON MOVIE

```
INSERT INTO person_movie (movie_id,
                    person_id,
                              role_id)
     VALUES (1,
             1,
                   2);
INSERT INTO person_movie (movie_id,
                    person_id,
                             role_id)
     VALUES (1,
             2,
                   2);
INSERT INTO person movie (movie id,
                   person id,
                             role id)
     VALUES (2,
                   2);
INSERT INTO person movie (movie id,
                   person id,
                             role id)
     VALUES (2,
             2,
                   2);
INSERT INTO person_movie (movie_id,
                   person_id,
                              role_id)
     VALUES (2,
             3,
                   2);
INSERT INTO person_movie (movie_id,
                    person_id,
                             role_id)
     VALUES (3,
             4,
                   2);
```

TABLE USERS

```
insert into Users (username, name, mail, password)
VALUES ('jacoob', 'Jakub', 'jacoob@pc.com', 'ManUnited');
insert into Users (username, name, mail, password)
VALUES ('ann', 'Anna', 'anna@pc.com', 'HaloHalo');
insert into Users (username, name, mail, password)
VALUES ('cris', 'Krzysiu', 'cris@pc.com', 'GGMU');
insert into Users (username, name, mail, password)
VALUES ('mama', 'Mama', 'mama@pc.com', '123mama');
insert into Users (username, name, mail, password)
VALUES ('tata', 'Tata', 'tata@pc.com', '123tata');
insert into Users (username, name, mail, password)
VALUES ('babciaidziadek', 'Babcia i Dziadek', 'babciaidziadek@pc.com', '123babciaidziadek');
```

TABLE USERS_FOLLOWERS

```
insert into users followers (username follower, username followed)
VALUES ('ann', 'jacoob');
insert into users followers (username follower, username followed)
VALUES ('mama', 'jacoob');
insert into users followers (username follower, username followed)
VALUES ('tata', 'jacoob');
insert into users followers (username follower, username followed)
VALUES ('mama', 'ann');
insert into users followers (username follower, username followed)
VALUES ('babciaidziadek', 'ann');
insert into users followers (username follower, username followed)
VALUES ('mama', 'tata');
insert into users followers (username follower, username followed)
VALUES ('tata', 'mama');
insert into users_followers (username_follower, username followed)
VALUES ('jacoob', 'babciaidziadek');
insert into users followers (username follower, username followed)
VALUES ('cris', 'babciaidziadek');
```

```
insert into users_followers (username_follower, username_followed)
VALUES ('ann', 'cris');
```

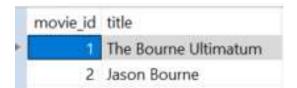
TABLE REVIEW

```
insert into Review (username, movie id, rate, text)
VALUES ('jacoob', 1, 5, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('ann', 1, 3, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('tata', 1, 4, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('jacoob', 2, 1, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('cris', 2, 5, 'XXX');
insert into Review (username, movie_id, rate, text)
VALUES ('mama', 2, 2, 'XXX');
insert into Review (username, movie_id, rate, text)
VALUES ('tata', 3, 5, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('jacoob', 3, 5, 'XXX');
insert into Review (username, movie id, rate, text)
VALUES ('ann', 3, 1, 'XXX');
```

2.2.SELECT statements

2.2.1. In which movies has Matt Damon participated?

```
SELECT distinct p.movie_id, m.title
from movie m, person_movie p
where p.person_id = (select person_id from
person where name='Matt Damon')
and m.movie id = p.movie id;
```



2.2.2. When was The Hunger Games released?

```
select production_year
from movie
where title = 'The Hunger Games';
```

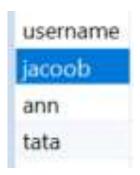
2.2.3. What is the average review score of all reviews?

```
select AVG(rate)
from review;
```



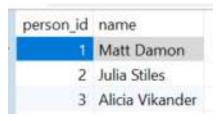
2.2.4. Who reviewed the Bourne Ultimatum?

```
select username
from review r
where r.movie_id = (SELECT movie_id from movie m where
m.title='The Bourne Ultimatum');
```



2.2.5. Which actors appeared in Jason Bourne?

```
select pm.person_id, p.name
from person_movie pm, person p
where pm.movie_id = (select movie_id from movie
m where m.title='Jason Bourne')
AND pm.person_id = p.person_id;
```



2.2.6. Which actors have appeared in Movies from Lionsgate? /Here Lionsgate is changed to 'Pearl Street' because Lionsgate doesn't appear in our tables/

```
select distinct p.name --, m.title, pc.name
from person p, movie m, production_company pc,
person_movie pm
where pc.name = 'Pearl Street' and p.person_id =
pm.person_id
and pm.role_id = '2' and m.movie_id = pm.movie_id;
```



2.2.7. In which movies has the director also performed as an actor?

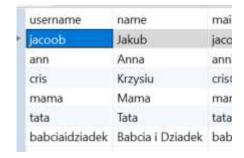
```
select distinct pm1.movie_id from
person_movie pm1, person_movie pm2
where pm1.movie_id = pm2.movie_id and
pm1.role_id=(SELECT Role.role_id FROM Role WHERE
Role_name='Director') and
pm2.role_id=(SELECT Role.role_id FROM Role WHERE
Role name='Actor');
```



2.2.8. Which reviewers have not reviewed Hunger Games?

```
select username
from users EXCEPT (select username from review r
where r.movie_id = (SELECT movie_id from movie where title='The Hunger
Games'));
```





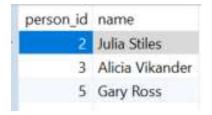


2.2.9. What is the average rating of all reviews?

```
select AVG(rate)
from review;
```

2.2.10. What is the average rating of all reviews?

```
select p.person_id, p.name
from person p EXCEPT (select distinct y.
person_id, x.name
from person x, images y
where x.person id = y.person id);
```



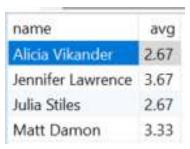
2.2.11. What is the average review score from each reviewer?

```
select username, avg (rate)
from review
group by username
order by username;
```

username	avg
ann	2.00
cris	5.00
jacoob	3.67
mama	2.00
tata	4.50

2.2.12. What is the average movie review score for each actor? (The average review for the movies, they've played in)

```
select p.name, avg(r.rate)
from person_movie pm, review r, person p
where pm.role_id=2
AND pm.movie_id = r.movie_id
and p.person_id = pm.person_id
group by p.name;
```



2.2.13. What is the average number of followers for the reviewers?

```
select ROUND(CAST(count(username_follower) as
numeric),2) /
(select count(username)from users) as
avg_num_of_followers
from users_followers;
```



2.3.Update DERIVED ATTRIBUTES.

2.3.1. Attribute *age* in 'Person' table

person_id	name	birthday	sex	birhplace	age
1	Matt Damon	10/08/1970	Μ	Cambridge	{null}
2	Julia Stiles	03/28/1981	F	New York	{null}
3	Alicia Vikander	10/03/1988	F	Gothenburg	{null}
5	Gary Ross	11/03/1956	Μ	Los Angeles	{null}
4	Jennifer Lawrence	08/15/1990	F	Louisville	{null}

update person p set age = ROUND(CAST(DATE_PART('year', now()) DATE_PART('year', p.birthday) as numeric), 0);

person_id	name	birthday	sex	birhplace	age
1	Matt Damon	10/08/1970	Μ	Cambridge	47
2	Julia Stiles	03/28/1981	F	New York	36
3	Alicia Vikander	10/03/1988	F	Gothenburg	29
5	Gary Ross	11/03/1956	М	Los Angeles	61
4	Jennifer Lawrence	08/15/1990	F	Louisville	27

2.3.2. Average review (avg_review) in 'Movie' table

movie_id	title	avg_review	production_year	status_id
1	The Bourne Ultimatum	{null}	2,007	1
2	Jason Bourne	{null}	2,016	1
3	The Hunger Games	{null}	2,012	1

```
update movie m set avg_review = (select AVG(rate)
from review r
where r.movie id = m.movie id);
```

movie_id	title	avg_review	production_year	status_id
1	The Bourne Ultimatum	4.00	2,007	1.
3	The Hunger Games	3.67	2,012	1
2	Jason Bourne	2.67	2.016	1

2.3.3. Number of movies (*num_of_movies*) in 'Person' table

person_id	name	birthday	sex	birhplace	age	num_of_movies
97.	Matt Damon	10/08/1970	M	Cambridge	47	
2	Julia Stiles	03/28/1981	F	New York	36	(null)
3	Alicia Vikander	10/03/1988	F	Gothenburg	29	(null)
4	Jennifer Lawrence	08/15/1990	F	Louisville	27	(null)
5	Gary Ross	11/03/1956	M	Los Angeles	61	(null)

update person p set num_of_movies = (select COUNT(distinct (movie_id))
from person_movie pm
where pm.person_id=p.person_id);

person_id	name	birthday	sex	birhplace	age	num_of_movies
. 1	Matt Damon	10/08/1970	M	Cambridge	47	2
2	Julia Stiles	03/28/1981	F	New York	36	2
3	Alicia Vikander	10/03/1988	F	Gothenburg	29	1
5	Gary Ross	11/03/1956	M	Los Angeles	61	1
4	Jennifer Lawrence	08/15/1990	F	Louisville	27	1

2.4.VIEWS

2.4.1. Create a simple view for 'works_on' table in the company database – all turples should be in this view!

```
create view workson_view as
select empl_ssn, fname, lname, proj_number, pname, hours
from works_on w, employee e, project p
where w.proj_number = p.pnumber AND w.empl_ssn=e.ssn;
```

100 select * from workson_view 11 order by proj number;

12

<

empl_ssn	fname	Iname	proj_number	pname	hours
123456789	John	Smith	1	ProductX	33
999887777	Alicia	Zelaya	1	ProductX	30
453453453	Joyce	English	1	ProductX	20
123456789	John	Smith	2	ProductY	8
453453453	Joyce	English	2	ProductY	20
333445555	Franklin	Wong	2	ProductY	10
123456789	John	Smith	3	ProductZ	3
CCCOOAAAA	Damaah	Marayan	ว	Draduat7	40

2.4.2. Create a view with a sum of hours for each project.

create view sum_of_hours as
select proj_number, pname as
proj_name, sum(hours) as
sum_of_hours
from works_on w, project p
where w.proj_number =
p.pnumber
group by proj_number, pname
order by proj_number;

180 select	* from sum_c	of_hours;
<		
proj_number	proj_name	sum_of_hours
1	ProductX	83
2	ProductY	38
3	ProductZ	53
10	Computerization	55
20	Reorganization	25
30	Newbenefits	55

2.4.3. Create a view with a sum of hours for each combination of employee and project, add names for employee and project and calculate the cost – the cost for each hour are 300 DDK.

```
create view project_cost as
select empl_ssn, fname, lname, proj_number, pname as proj_name, hours,
hours*300 AS Cost
from works_on w, employee e, project p
where w.proj_number = p.pnumber AND w.empl_ssn=e.ssn
order by proj number;
```

30⊖select * from project_cost;

empl_ssn	fname	Iname	proj_number	proj_name	hours	cost
123456789	John	Smith	1	ProductX	33	9,900
999887777	Alicia	Zelaya	1	ProductX	30	9,000
453453453	Joyce	English	1	ProductX	20	6,000
123456789	John	Smith	2	ProductY	8	2,400
453453453	Joyce	English	2	ProductY	20	6,000
333445555	Franklin	Wong	2	ProductY	10	3,000
123456789	John	Smith	3	ProductZ	3	900
666884444	Ramesh	Naravan	3	Product7	40	12 000

2.4.4. Create a view of you own choice.

/this view represents all the managers of some projects and include also the amount of years as a manager of the project/

```
create view managers as
select ssn, fname, lname, dnumber, dname, startdate,
ROUND(CAST(DATE_PART('year', now()) - DATE_PART('year', startdate) as
numeric), 0) as years_as_manager
from employee e, manages m, department d
where e.ssn = m.empl_ssn AND d.dnumber = m.dno;
```

44@select * from managers;

•						
ssn	fname	Iname	dnumber	dname	startdate	years_as_manager
123456789	John	Smith	1	Headquarters	04/07/2001	16
333445555	Franklin	Wong	4	Administration	12/24/2014	3
999887777	Alicia	Zelaya	5	Research	06/16/1993	24

2.5.TRIGGERS

2.5.1. Create a log trigger for the 'works_on' table in the company database – for insert, update and delete. The new logtable should contain a serial id and a time stamp.

```
create table logworkson (
log id serial primary key,
empl ssn char(9),
proj_number integer,
hours INTEGER,
current text,
time time);
create or replace function set log workson() returns trigger
      $$
BEGIN
      if (tg_op = 'INSERT')
            insert into logworkson (
         empl ssn,
         proj number,
         hours,
         current,
         time)
         values (new.empl ssn, new.proj number, new.hours,
current_timestamp, now());
         return new;
   end if;
   if(tg_op = 'UPDATE')
   then
           insert into logworkson (
         empl ssn,
         proj_number,
         hours,
         current,
         time)
         values (new.empl ssn, new.proj number, new.hours,
current timestamp, now());
        return new;
  end if;
  return null;
  end;
  language plpgsql;
create trigger set log workson trigger
before insert or update or delete
on works on
for each row
execute procedure set log workson();
INSERT INTO WORKS_ON (empl_ssn, proj_number, hours)
     VALUES (999887777, 1, 30.0);
```

```
84@select * from logworkson;

85

| log_id empl_ssn proj_number hours current | time |
| 1 999887777 | 1 30 2017-04-15 16:26:46.867354+02 16:26:46
```

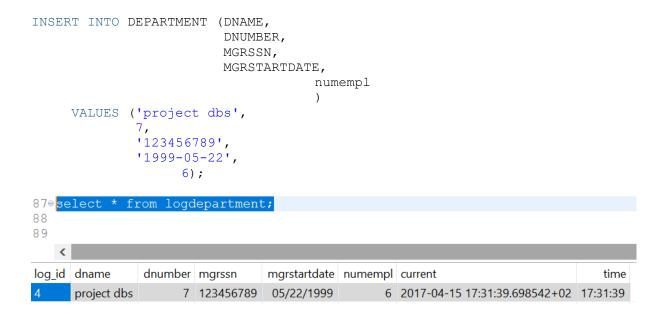
2.5.2. Create a trigger for the 'works_on' table who raise an exception if the employee is assigned more than 4 projects.

create or replace function workson max() returns trigger AS \$\$

```
if (select count(empl ssn) from works on
where new.empl ssn=empl ssn) \geq= 4 THEN
RAISE EXCEPTION 'employee cannot work for more than 4 projects';
end if;
return new;
end;
$$
language plpgsql;
CREATE TRIGGER workson max trigger BEFORE INSERT OR UPDATE ON works on
    FOR EACH ROW EXECUTE PROCEDURE workson max();
select count(empl ssn) from works on
where empl ssn='333445555';
INSERT INTO WORKS ON (empl ssn, proj number, hours)
     VALUES (333445555, 1, 20.0);
select count(empl ssn) from works on
where empl ssn='123456789';
INSERT INTO WORKS_ON (empl_ssn, proj_number, hours)
     VALUES (123456789, 3, 3.0);
107⊜
    INSERT INTO WORKS_ON (empl_ssn, proj_number, hours)
108
          VALUES (333445555, 1, 20.0);
109
     <
Result
@ Quick Doc 👺 DBMS Output 🖳 SQL Monitor 🗗 SQL Recall ध DB Problems ∺ 🛅 DB Console 📮 Console 🗗 Ju JUnit
ERROR: employee cannot work for more than 4 projects Where: PL/pgSQL function workson max() line 5 at RAISE
Executed Statement:
```

2.5.3. Create a log trigger for department table in the company database ...the content in the log table should be readable!

```
create table logdepartment (
log id serial primary key,
              CHARACTER VARYING (20),
 dname
                integer,
   dnumber
  mgrssn
                 character (9),
  mgrstartdate date,
                 integer,
   numempl
current text,
time time);
create or replace function set log department() returns trigger
      $$
BEGIN
      if (tg op = 'INSERT')
      then
            insert into logdepartment (
         dname,
         dnumber,
         mgrssn,
         mgrstartdate,
         numempl,
         current,
         time)
         values (new.dname, new.dnumber, new.mgrssn, new.mgrstartdate,
new.numempl, current timestamp, now());
         return new;
   end if;
   if(tg op = 'UPDATE')
   then
            insert into logdepartment (
         dname,
         dnumber,
         mgrssn,
         mgrstartdate,
         numempl,
         current,
         time)
         values (new.dname, new.dnumber, new.mgrssn, new.mgrstartdate,
new.numempl, current timestamp, now());
         return new;
   end if;
  return null;
  end;
  $$
  language plpgsql;
create trigger set log department trigger
before insert or update or delete
on department
for each row
execute procedure set log department();
```



2.5.4. Create a trigger of your own choice.

/This trigger unables to add an employee ssn as someone's supervisor if he or she already supervises at 2 employees/

```
create or replace function super ssn max() returns trigger AS $$
BEGIN
if (select count(superssn) from employee
where new.superssn=superssn) >= 2 THEN
RAISE EXCEPTION 'employee cannot supervise more than 2 other employees';
end if;
return new;
end;
$$
language plpgsql;
CREATE TRIGGER super ssn max trigger BEFORE INSERT OR UPDATE ON employee
    FOR EACH ROW EXECUTE PROCEDURE super ssn max();
select count(superssn) from employee
where superssn='123456789';
INSERT INTO EMPLOYEE (FNAME,
                      MINIT,
                       LNAME,
                       SSN,
                       BDATE,
                       ADDRESS,
                       SEX,
                       SALARY,
                       SUPERSSN,
                       DNO)
     VALUES ('Robert',
             'Y',
             'Dyane',
             '647647647',
             '1999-08-27'
             'Horsens, DK',
             'M',
```

```
28000,
                 '123456789',
                 4);
225
226-select count(superssn) from employee
227 where superssn='123456789';
228
      <
 count
229 INSERT INTO EMPLOYEE (FNAME,
230
                                MINIT,
231
                                LNAME,
232
233
234
235
                                SSN,
                                BDATE,
                                ADDRESS,
                                SEX,
236
                                SALARY,
237
                                SUPERSSN,
238
                                DNO)
239
240
241
242
           VALUES ('Robert',
                     'Y',
                     'Dyane',
                     '647647647',
'1999-08-27',
243
244
                     'Horsens, DK',
                     'M',
245
                     28000,
246
247
248
                     '123456789',
                     4);
249
250
      <

    Quick Doc    □ DBMS Output   □ SQL Monitor    □ SQL Recall    □ DB Problems    □ DB Console    □ Console    □ Unit
```

ERROR: employee cannot supervise more than 2 other employees Where: PL/pgSQL function super-ssn-max() line 5 at RAISE