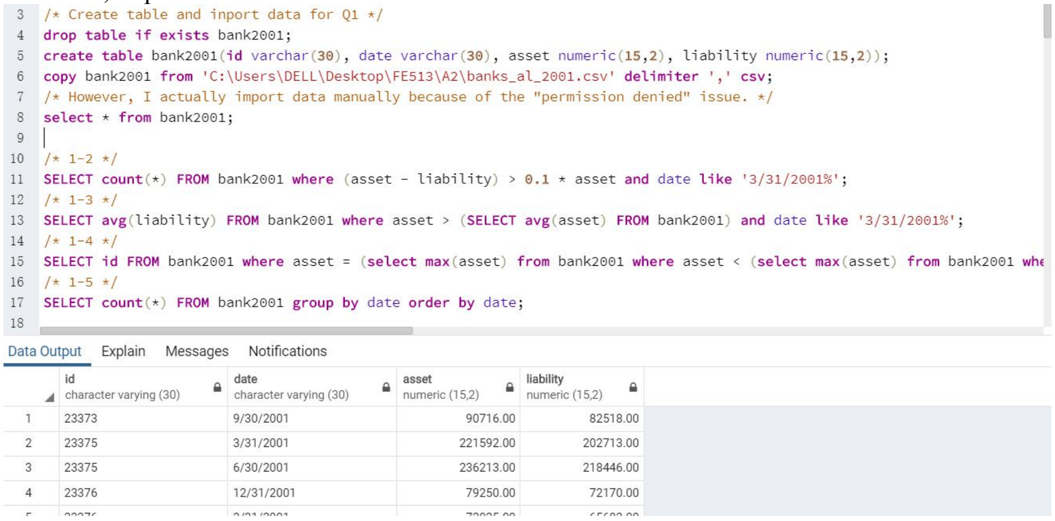
Mostly I used only one query but in order to make it easier to read I split them to several lines.

Question 1:

1-1

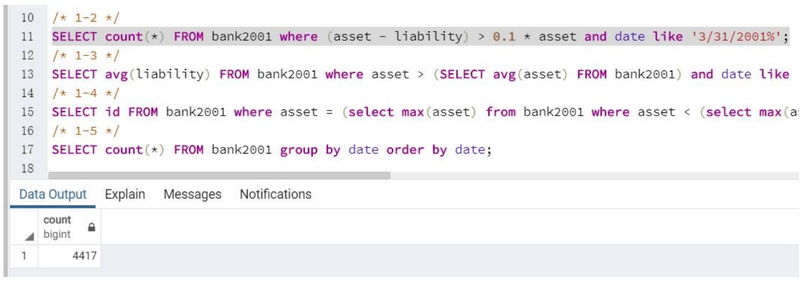
Create table, import data and view the head:



```
/* Create table and inport data for Q1 */
drop table if exists bank2001;
create table bank2001(id varchar(30), date varchar(30), asset numeric(15,2), liability numeric(15,2));
copy bank2001 from 'C:\Users\DELL\Desktop\FE513\A2\banks_al_2001.csv' delimiter ',' csv;
/* However, I actually import data manually because of the "permission denied" issue. */
select * from bank2001;

/* 1-2 */
SELECT count(*) FROM bank2001 where (asset - liability) > 0.1 * asset and date like '3/31/2001%';
/* 1-3 */
SELECT avg(liability) FROM bank2001 where asset > (SELECT avg(asset) FROM bank2001) and date like '3/31/2001%';
/* 1-4 */
SELECT id FROM bank2001 where asset = (select max(asset) from bank2001 where asset < (select max(asset) from bank2001 whe
/* 1-5 */
SELECT count(*) FROM bank2001 group by date order by date;
```

```
Data Output   Explain   Messages   Notifications

     id                        date                       asset           liability
     character varying (30)    character varying (30)     numeric (15,2)  numeric (15,2)
1    23373                     9/30/2001                       90716.00        82518.00
2    23375                     3/31/2001                      221592.00       202713.00
3    23375                     6/30/2001                      236213.00       218446.00
4    23376                     12/31/2001                      79250.00        72170.00
```

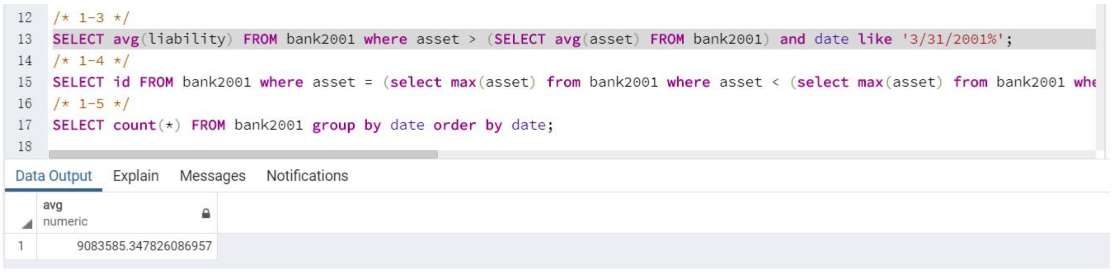
1-2



```
/* 1-2 */
SELECT count(*) FROM bank2001 where (asset - liability) > 0.1 * asset and date like '3/31/2001%';
/* 1-3 */
SELECT avg(liability) FROM bank2001 where asset > (SELECT avg(asset) FROM bank2001) and date like
/* 1-4 */
SELECT id FROM bank2001 where asset = (select max(asset) from bank2001 where asset < (select max(a
/* 1-5 */
SELECT count(*) FROM bank2001 group by date order by date;
```

```
Data Output   Explain   Messages   Notifications

     count
     bigint
1    4417
```

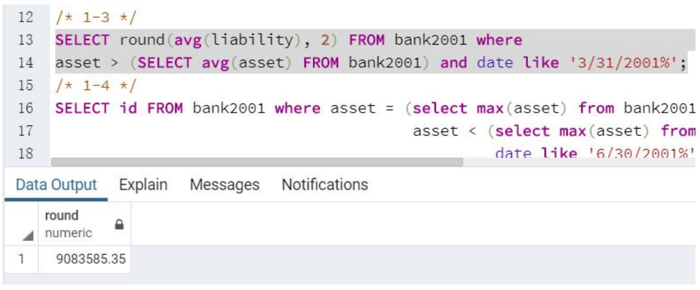
1-3



```
/* 1-3 */
SELECT avg(liability) FROM bank2001 where asset > (SELECT avg(asset) FROM bank2001) and date like '3/31/2001%';
/* 1-4 */
SELECT id FROM bank2001 where asset = (select max(asset) from bank2001 where asset < (select max(asset) from bank2001 whe
/* 1-5 */
SELECT count(*) FROM bank2001 group by date order by date;
```

```
Data Output   Explain   Messages   Notifications

     avg
     numeric
1    9083585.347826086957
```

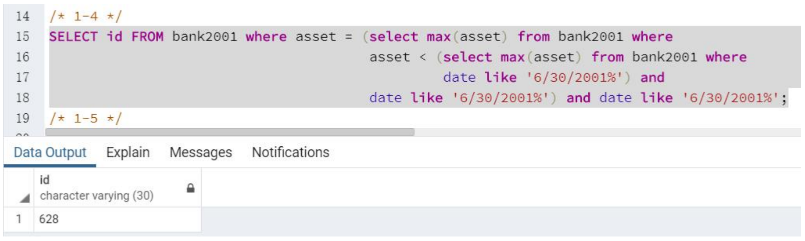
Also I can change the decimal number by using “round()”:



```
/* 1-3 */
SELECT round(avg(liability), 2) FROM bank2001 where
asset > (SELECT avg(asset) FROM bank2001) and date like '3/31/2001%';
/* 1-4 */
SELECT id FROM bank2001 where asset = (select max(asset) from bank2001
                                       asset < (select max(asset) from
                                           date like '6/30/2001%'
```

```
Data Output   Explain   Messages   Notifications

     round
     numeric
1    9083585.35
```

1-4

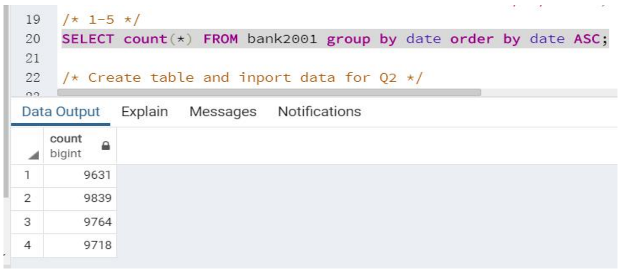


```
/* 1-4 */
SELECT id FROM bank2001 where asset = (select max(asset) from bank2001 where
                                        asset < (select max(asset) from bank2001 where
                                                 date like '6/30/2001%') and
                                        date like '6/30/2001%') and date like '6/30/2001%';
/* 1-5 */
```

Data Output   Explain   Messages   Notifications

```
   id
   character varying (30)
1  628
```

1-5
The order here is the forth quarter, first quarter, second quarter and third quarter.



```
/* 1-5 */
SELECT count(*) FROM bank2001 group by date order by date ASC;

/* Create table and inport data for Q2 */
```

Data Output   Explain   Messages   Notifications

```
   count
   bigint
1  9631
2  9839
3  9764
4  9718
```

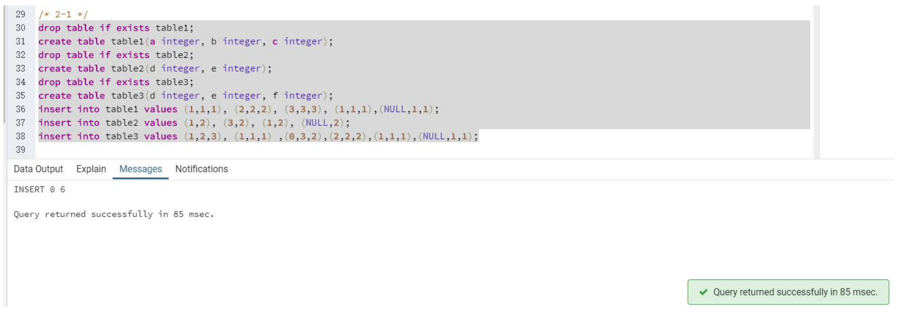
Question 2:

```
/* Create table and inport data for Q2 */
drop table if exists bank2002_sec;
create table bank2002_sec(id varchar(30), date varchar(30), security numeric(15,2));
copy bank2002_sec from 'C:\Users\DELL\Desktop\FE513\A2\banks_sec_2002.csv' delimiter ',' csv;
drop table if exists bank2002_al;
create table bank2002_al(id varchar(30), date varchar(30), asset numeric(15,2), liability numeric(15,2));
copy bank2002_al from 'C:\Users\DELL\Desktop\FE513\A2\banks_al_2002.csv' delimiter ',' csv;
/* However, I actually import data manually because of the "permission denied" issue. */
```

2-1

Intersect is an operator and Inner join is a type of join.
Intersect requires the same number of fields while inner join does not.
The result of intersect is groups of values in the first table that also appears in the second table, while inner join return a combination of 2 tables.
Intersect can return matching null values but inner join can't.
Intersect doesn't return any duplicate values but inner join will not delete duplicate values.

Here I prepare 3 tables for test:



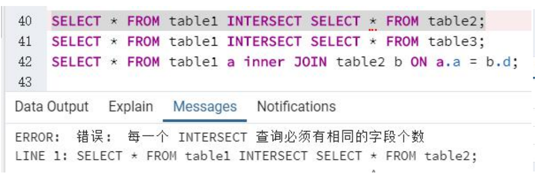
```
/* 2-1 */
drop table if exists table1;
create table table1(a integer, b integer, c integer);
drop table if exists table2;
create table table2(d integer, e integer);
drop table if exists table3;
create table table3(d integer, e integer, f integer);
insert into table1 values (1,1,1), (2,2,2), (3,3,3), (1,1,1),(NULL,1,1);
insert into table2 values (1,2), (3,2), (1,2), (NULL,2);
insert into table3 values (1,2,3), (1,1,1) ,(0,3,2),(2,2,2),(1,1,1),(NULL,1,1);
```

Data Output   Explain   Messages   Notifications

```
INSERT 0 6

Query returned successfully in 85 msec.
```

✓ Query returned successfully in 85 msec.

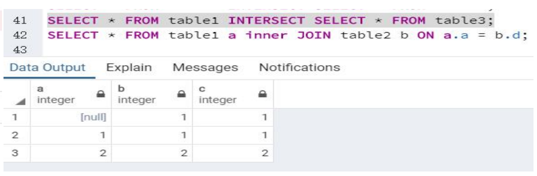
And then comes the queries:



```
SELECT * FROM table1 INTERSECT SELECT * FROM table2;
SELECT * FROM table1 INTERSECT SELECT * FROM table3;
SELECT * FROM table1 a inner JOIN table2 b ON a.a = b.d;
```

Data Output　Explain　Messages　Notifications

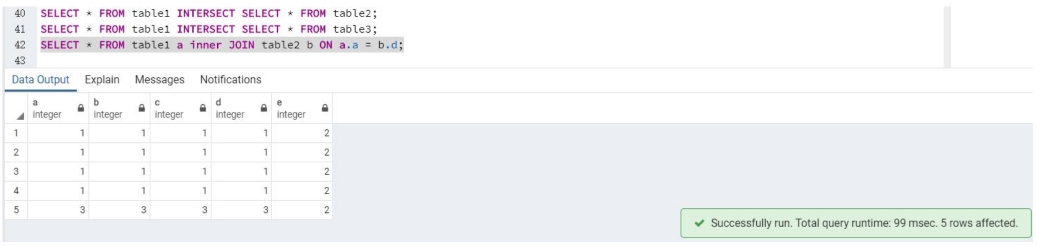
```
ERROR:  错误:  每一个 INTERSECT 查询必须有相同的字段个数
LINE 1: SELECT * FROM table1 INTERSECT SELECT * FROM table2;
```



```
SELECT * FROM table1 INTERSECT SELECT * FROM table3;
SELECT * FROM table1 a inner JOIN table2 b ON a.a = b.d;
```

Data Output　Explain　Messages　Notifications

	a integer	b integer	c integer
1	[null]	1	1
2	1	1	1
3	2	2	2



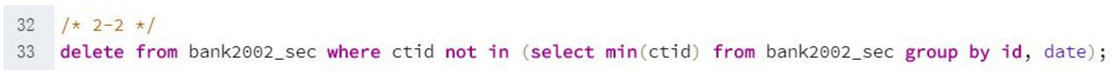
```
SELECT * FROM table1 INTERSECT SELECT * FROM table2;
SELECT * FROM table1 INTERSECT SELECT * FROM table3;
SELECT * FROM table1 a inner JOIN table2 b ON a.a = b.d;
```

Data Output　Explain　Messages　Notifications

	a integer	b integer	c integer	d integer	e integer
1	1	1	1	1	2
2	1	1	1	1	2
3	1	1	1	1	2
4	1	1	1	1	2
5	3	3	3	3	2

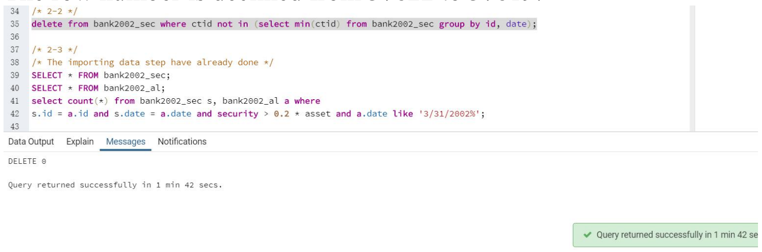
✓ Successfully run. Total query runtime: 99 msec. 5 rows affected.

2-2



```
/* 2-2 */
delete from bank2002_sec where ctid not in (select min(ctid) from bank2002_sec group by id, date);
```

The row number is declined from 37822 to 37819.



```
/* 2-2 */
delete from bank2002_sec where ctid not in (select min(ctid) from bank2002_sec group by id, date);

/* 2-3 */
/* The importing data step have already done */
SELECT * FROM bank2002_sec;
SELECT * FROM bank2002_al;
select count(*) from bank2002_sec s, bank2002_al a where
s.id = a.id and s.date = a.date and security > 0.2 * asset and a.date like '3/31/2002%';
```

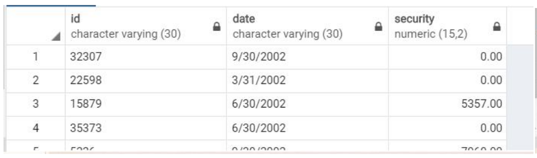
Data Output　Explain　Messages　Notifications

```
DELETE 0

Query returned successfully in 1 min 42 secs.
```

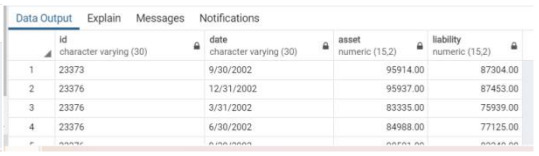
✓ Query returned successfully in 1 min 42 se

2-3

The importing step has already been down. View data:

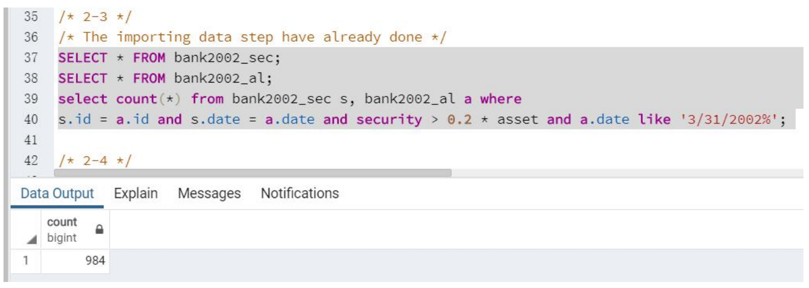


	id character varying (30)	date character varying (30)	security numeric (15,2)
1	32307	9/30/2002	0.00
2	22598	3/31/2002	0.00
3	15879	6/30/2002	5357.00
4	35373	6/30/2002	0.00



Data Output　Explain　Messages　Notifications

	id character varying (30)	date character varying (30)	asset numeric (15,2)	liability numeric (15,2)
1	23373	9/30/2002	95914.00	87304.00
2	23376	12/31/2002	95937.00	87453.00
3	23376	3/31/2002	83335.00	75939.00
4	23376	6/30/2002	84988.00	77125.00

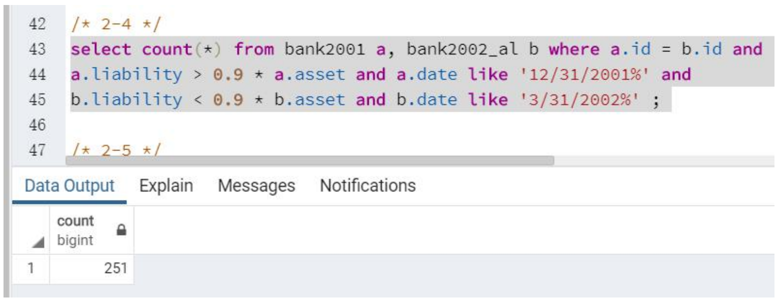


```
/* 2-3 */
/* The importing data step have already done */
SELECT * FROM bank2002_sec;
SELECT * FROM bank2002_al;
select count(*) from bank2002_sec s, bank2002_al a where
s.id = a.id and s.date = a.date and security > 0.2 * asset and a.date like '3/31/2002%';

/* 2-4 */
```

Data Output　Explain　Messages　Notifications

	count bigint
1	984

2-4

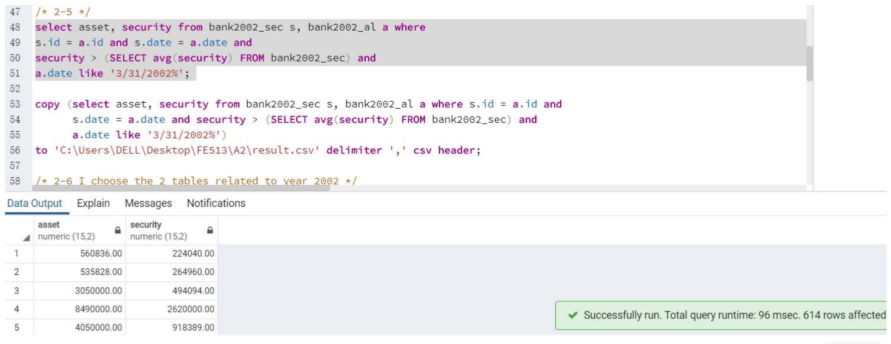


```
/* 2-4 */
select count(*) from bank2001 a, bank2002_al b where a.id = b.id and
a.liability > 0.9 * a.asset and a.date like '12/31/2001%' and
b.liability < 0.9 * b.asset and b.date like '3/31/2002%' ;

/* 2-5 */
```

Data Output　Explain　Messages　Notifications

	count bigint
1	251

2-5

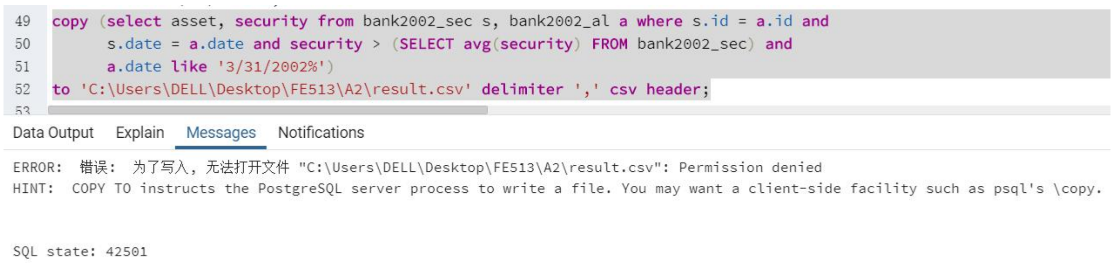


```
/* 2-5 */
select asset, security from bank2002_sec s, bank2002_al a where
s.id = a.id and s.date = a.date and
security > (SELECT avg(security) FROM bank2002_sec) and
a.date like '3/31/2002%';

copy (select asset, security from bank2002_sec s, bank2002_al a where s.id = a.id and
      s.date = a.date and security > (SELECT avg(security) FROM bank2002_sec) and
      a.date like '3/31/2002%')
to 'C:\Users\DELL\Desktop\FE513\A2\result.csv' delimiter ',' csv header;

/* 2-6 I choose the 2 tables related to year 2002 */
```

Data Output | Explain | Messages | Notifications

	asset numeric (15,2)	security numeric (15,2)
1	560836.00	224040.00
2	535828.00	264960.00
3	3050000.00	494094.00
4	8490000.00	2620000.00
5	4050000.00	918389.00

```
Successfully run. Total query runtime: 96 msec. 614 rows affected
```

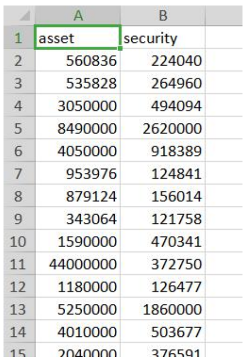


```
copy (select asset, security from bank2002_sec s, bank2002_al a where s.id = a.id and
      s.date = a.date and security > (SELECT avg(security) FROM bank2002_sec) and
      a.date like '3/31/2002%')
to 'C:\Users\DELL\Desktop\FE513\A2\result.csv' delimiter ',' csv header;
```

Data Output | Explain | Messages | Notifications

```
ERROR:  错误:  为了写入，无法打开文件 "C:\Users\DELL\Desktop\FE513\A2\result.csv": Permission denied
HINT:  COPY TO instructs the PostgreSQL server process to write a file. You may want a client-side facility such as psql's \copy.


SQL state: 42501
```

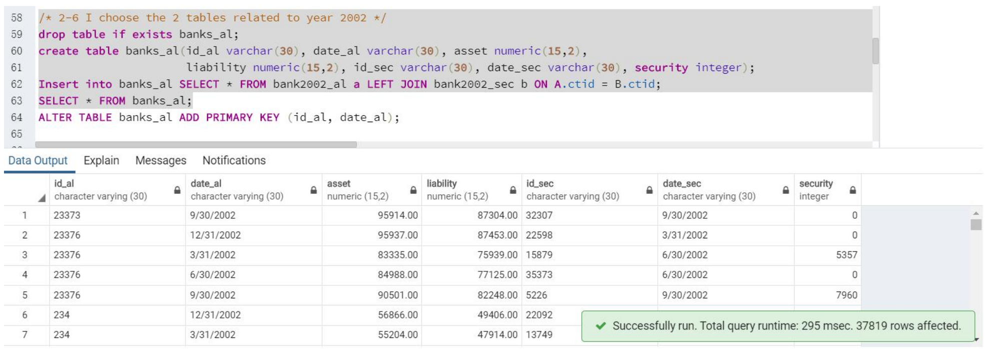
Still, because of the “Permission denied” issue I cannot export with commands, so I use the download bottom . Here is part of the csv file I got.



	A	B
1	asset	security
2	560836	224040
3	535828	264960
4	3050000	494094
5	8490000	2620000
6	4050000	918389
7	953976	124841
8	879124	156014
9	343064	121758
10	1590000	470341
11	44000000	372750
12	1180000	126477
13	5250000	1860000
14	4010000	503677
15	2040000	376591

2-6

Here I created a new table and then combined 2 tables into one and view what I got:



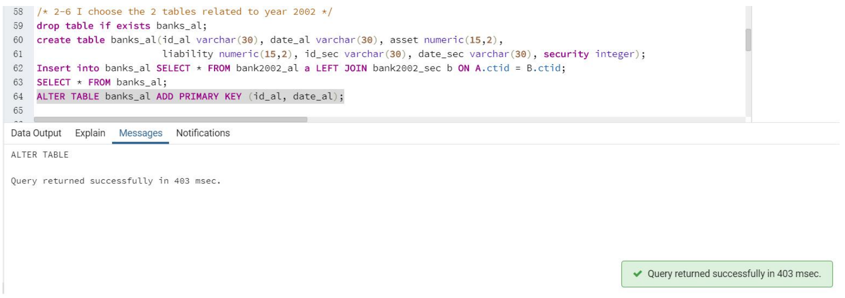
```
/* 2-6 I choose the 2 tables related to year 2002 */
drop table if exists banks_al;
create table banks_al(id_al varchar(30), date_al varchar(30), asset numeric(15,2),
                      liability numeric(15,2), id_sec varchar(30), date_sec varchar(30), security integer);
Insert into banks_al SELECT * FROM bank2002_al a LEFT JOIN bank2002_sec b ON A.ctid = B.ctid;
SELECT * FROM banks_al;
ALTER TABLE banks_al ADD PRIMARY KEY (id_al, date_al);
```

Data Output | Explain | Messages | Notifications

	id_al character varying (30)	date_al character varying (30)	asset numeric (15,2)	liability numeric (15,2)	id_sec character varying (30)	date_sec character varying (30)	security integer
1	23373	9/30/2002	95914.00	87304.00	32307	9/30/2002	0
2	23376	12/31/2002	95937.00	87453.00	22598	3/31/2002	0
3	23376	3/31/2002	83335.00	75939.00	15879	6/30/2002	5357
4	23376	6/30/2002	84988.00	77125.00	35373	6/30/2002	0
5	23376	9/30/2002	90501.00	82248.00	5226	9/30/2002	7960
6	234	12/31/2002	56866.00	49406.00	22092		
7	234	3/31/2002	55204.00	47914.00	13749		

```
Successfully run. Total query runtime: 295 msec. 37819 rows affected.
```

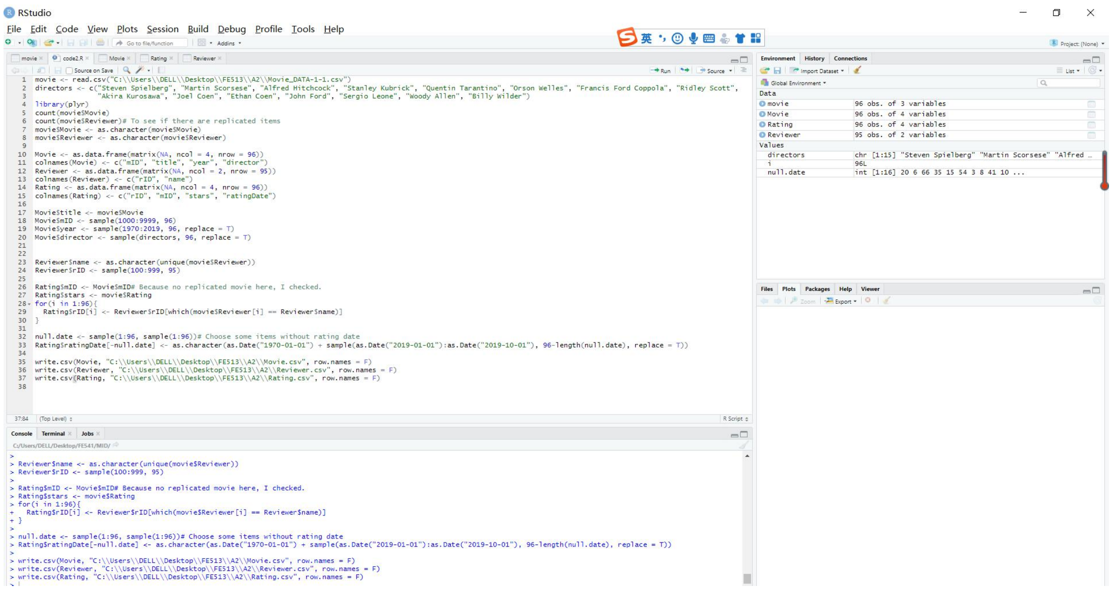
And then set primary key as below:


```
/* 2-6 I choose the 2 tables related to year 2002 */
drop table if exists banks_al;
create table banks_al(id_al varchar(30), date_al varchar(30), asset numeric(15,2),
                   liability numeric(15,2), id_sec varchar(30), date_sec varchar(30), security integer);
Insert into banks_al SELECT * FROM bank2002_al a LEFT JOIN bank2002_sec b ON A.ctid = B.ctid;
SELECT * FROM banks_al;
ALTER TABLE banks_al ADD PRIMARY KEY (id_al, date_al);
```

```
ALTER TABLE

Query returned successfully in 403 msec.
```

Question 3:

For the third question, I first use R to fake data and export the tables as csv files. Here is the screenshot contains all commands and tables I created.

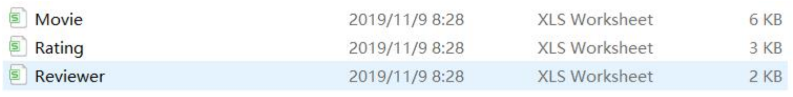


```
movie <- read.csv("C:\\Users\\DELL\\Desktop\\FE513\\A2\\Movie_DATA-1-1.csv")
directors <- c("Steven Spielberg", "Martin Scorsese", "Alfred Hitchcock", "Stanley Kubrick", "Quentin Tarantino", "Orson Welles", "Francis Ford Coppola", "Ridley Scott",
               "Akira Kurosawa", "Joel Coen", "Ethan Coen", "John Ford", "Sergio Leone", "Woody Allen", "Billy Wilder")
library(plyr)
count(movie$Movie)
count(movie$Reviewer)# To see if there are replicated items
movie$Movie <- as.character(movie$Movie)
movie$Reviewer <- as.character(movie$Reviewer)

Movie <- as.data.frame(matrix(NA, ncol = 4, nrow = 96))
colnames(Movie) <- c("mID", "title", "year", "director")
Reviewer <- as.data.frame(matrix(NA, ncol = 2, nrow = 95))
colnames(Reviewer) <- c("rID", "name")
Rating <- as.data.frame(matrix(NA, ncol = 4, nrow = 96))
colnames(Rating) <- c("rID", "mID", "stars", "ratingDate")

Movie$title <- movie$Movie
Movie$mID <- sample(1000:9999, 96)
Movie$year <- sample(1970:2019, 96, replace = T)
Movie$director <- sample(directors, 96, replace = T)


Reviewer$name <- as.character(unique(movie$Reviewer))
Reviewer$rID <- sample(100:999, 95)

Rating$mID <- Movie$mID# Because no replicated movie here, I checked.
Rating$stars <- movie$Rating
for(i in 1:96){
  Rating$rID[i] <- Reviewer$rID[which(movie$Reviewer[i] == Reviewer$name)]
}

null.date <- sample(1:96, sample(1:96))# Choose some items without rating date
Rating$ratingDate[-null.date] <- as.character(as.Date("1970-01-01") + sample(as.Date("2019-01-01"):as.Date("2019-10-01"), 96-length(null.date), replace = T))

write.csv(Movie, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Movie.csv", row.names = F)
write.csv(Reviewer, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Reviewer.csv", row.names = F)
write.csv(Rating, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Rating.csv", row.names = F)
```

```
37:84   (Top Level)                                                     R Script
```

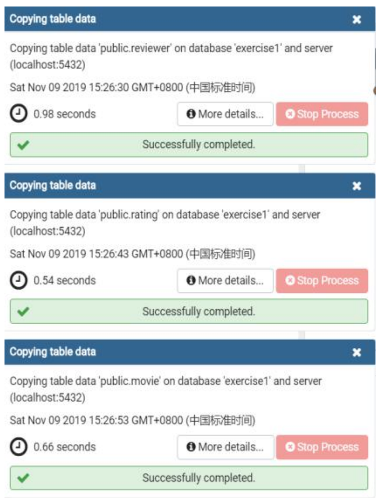
```
Data
movie        96 obs. of 3 variables
Movie        96 obs. of 4 variables
Rating       96 obs. of 4 variables
Reviewer     95 obs. of 2 variables
Values
directors    chr [1:15] "Steven Spielberg" "Martin Scorsese" "Alfred …
i            96L
null.date    int [1:16] 20 6 66 35 15 54 3 8 41 10 ...
```

```
C:/Users/DELL/Desktop/FE541/MID/
>
> Reviewer$name <- as.character(unique(movie$Reviewer))
> Reviewer$rID <- sample(100:999, 95)
>
> Rating$mID <- Movie$mID# Because no replicated movie here, I checked.
> Rating$stars <- movie$Rating
> for(i in 1:96){
+   Rating$rID[i] <- Reviewer$rID[which(movie$Reviewer[i] == Reviewer$name)]
+ }
>
> null.date <- sample(1:96, sample(1:96))# Choose some items without rating date
> Rating$ratingDate[-null.date] <- as.character(as.Date("1970-01-01") + sample(as.Date("2019-01-01"):as.Date("2019-10-01"), 96-length(null.date), replace = T))
>
> write.csv(Movie, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Movie.csv", row.names = F)
> write.csv(Reviewer, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Reviewer.csv", row.names = F)
> write.csv(Rating, "C:\\Users\\DELL\\Desktop\\FE513\\A2\\Rating.csv", row.names = F)
>
```

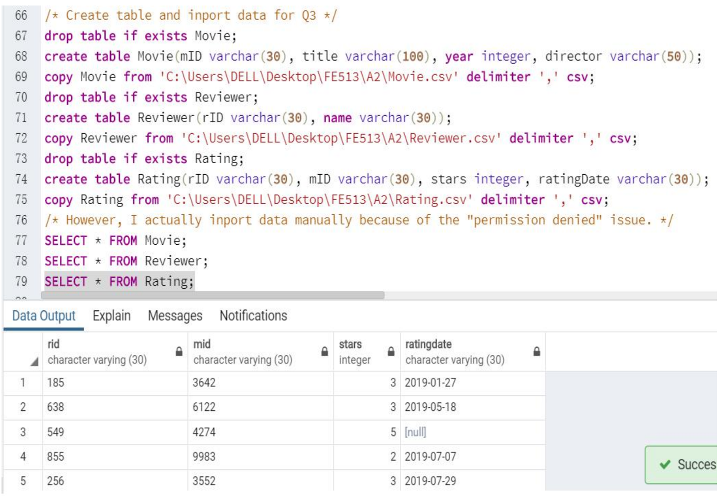


Movie	2019/11/9 8:28	XLS Worksheet	6 KB
Rating	2019/11/9 8:28	XLS Worksheet	3 KB
Reviewer	2019/11/9 8:28	XLS Worksheet	2 KB

Then I created tables, imported data and viewed what I got:

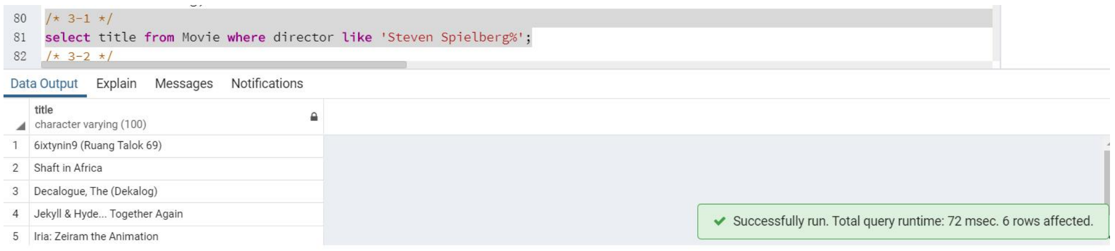


```
Copying table data
Copying table data 'public.reviewer' on database 'exercise1' and server
(localhost:5432)
Sat Nov 09 2019 15:26:30 GMT+0800 (中国标准时间)
0.98 seconds
Successfully completed.

Copying table data
Copying table data 'public.rating' on database 'exercise1' and server
(localhost:5432)
Sat Nov 09 2019 15:26:43 GMT+0800 (中国标准时间)
0.54 seconds
Successfully completed.

Copying table data
Copying table data 'public.movie' on database 'exercise1' and server
(localhost:5432)
Sat Nov 09 2019 15:26:53 GMT+0800 (中国标准时间)
0.66 seconds
Successfully completed.
```


```
/* Create table and inport data for Q3 */
drop table if exists Movie;
create table Movie(mID varchar(30), title varchar(100), year integer, director varchar(50));
copy Movie from 'C:\Users\DELL\Desktop\FE513\A2\Movie.csv' delimiter ',' csv;
drop table if exists Reviewer;
create table Reviewer(rID varchar(30), name varchar(30));
copy Reviewer from 'C:\Users\DELL\Desktop\FE513\A2\Reviewer.csv' delimiter ',' csv;
drop table if exists Rating;
create table Rating(rID varchar(30), mID varchar(30), stars integer, ratingDate varchar(30));
copy Rating from 'C:\Users\DELL\Desktop\FE513\A2\Rating.csv' delimiter ',' csv;
/* However, I actually inport data manually because of the "permission denied" issue. */
SELECT * FROM Movie;
SELECT * FROM Reviewer;
SELECT * FROM Rating;
```

	rid character varying (30)	mid character varying (30)	stars integer	ratingdate character varying (30)
1	185	3642	3	2019-01-27
2	638	6122	3	2019-05-18
3	549	4274	5	[null]
4	855	9983	2	2019-07-07
5	256	3552	3	2019-07-29

3-1



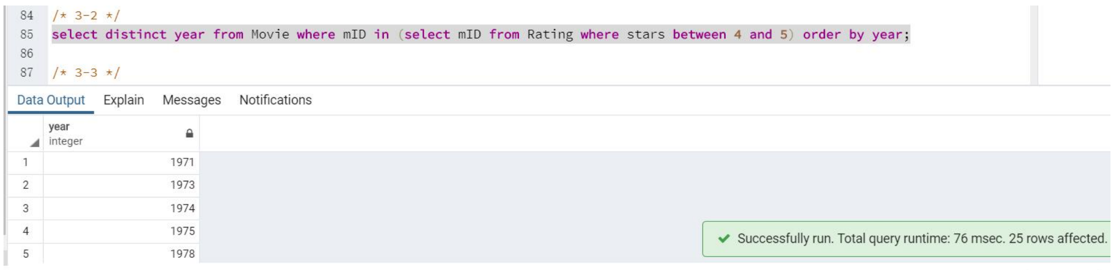
```sql
/* 3-1 */
select title from Movie where director like 'Steven Spielberg%';
/* 3-2 */
```

Data Output | Explain | Messages | Notifications

	title character varying (100)
1	6ixtynin9 (Ruang Talok 69)
2	Shaft in Africa
3	Decalogue, The (Dekalog)
4	Jekyll & Hyde... Together Again
5	Iria: Zeiram the Animation

✓ Successfully run. Total query runtime: 72 msec. 6 rows affected.

3-2



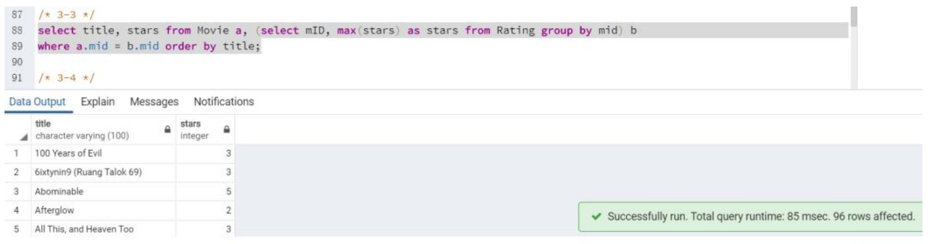
```sql
/* 3-2 */
select distinct year from Movie where mID in (select mID from Rating where stars between 4 and 5) order by year;

/* 3-3 */
```

Data Output | Explain | Messages | Notifications

	year integer
1	1971
2	1973
3	1974
4	1975
5	1978

✓ Successfully run. Total query runtime: 76 msec. 25 rows affected.

3-3



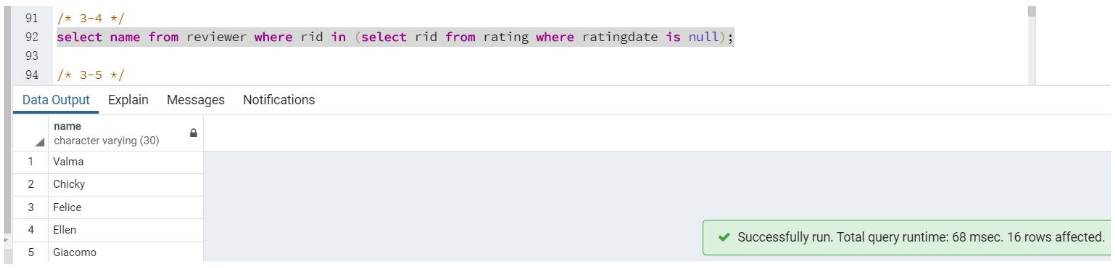
```sql
/* 3-3 */
select title, stars from Movie a, (select mID, max(stars) as stars from Rating group by mid) b
where a.mid = b.mid order by title;

/* 3-4 */
```

Data Output | Explain | Messages | Notifications

	title character varying (100)	stars integer
1	100 Years of Evil	3
2	6ixtynin9 (Ruang Talok 69)	3
3	Abominable	5
4	Afterglow	2
5	All This, and Heaven Too	3

✓ Successfully run. Total query runtime: 85 msec. 96 rows affected.

3-4



```sql
/* 3-4 */
select name from reviewer where rid in (select rid from rating where ratingdate is null);

/* 3-5 */
```

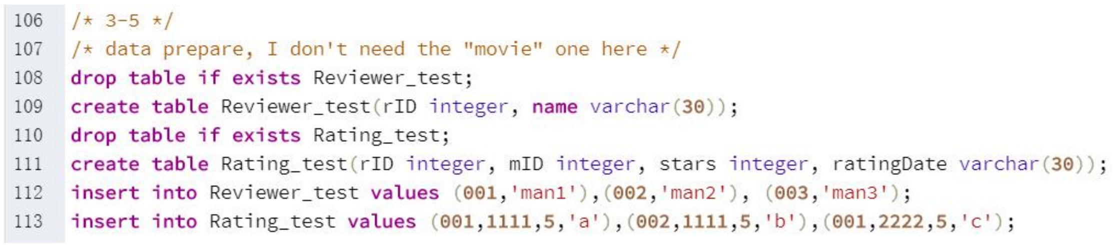
Data Output | Explain | Messages | Notifications

	name character varying (30)
1	Valma
2	Chicky
3	Felice
4	Ellen
5	Giacomo

✓ Successfully run. Total query runtime: 68 msec. 16 rows affected.

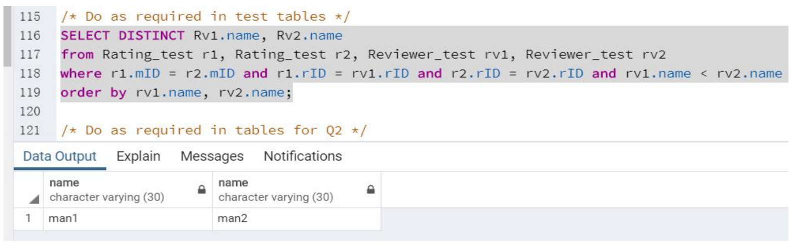
3-5

There is no pair in my data so I created test tables for this question:



```sql
/* 3-5 */
/* data prepare, I don't need the "movie" one here */
drop table if exists Reviewer_test;
create table Reviewer_test(rID integer, name varchar(30));
drop table if exists Rating_test;
create table Rating_test(rID integer, mID integer, stars integer, ratingDate varchar(30));
insert into Reviewer_test values (001,'man1'),(002,'man2'), (003,'man3');
insert into Rating_test values (001,1111,5,'a'),(002,1111,5,'b'),(001,2222,5,'c');
```

And then attempt with test tables:

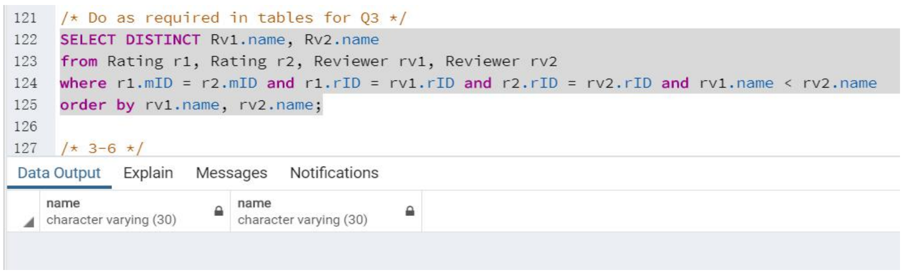


```
/* Do as required in test tables */
SELECT DISTINCT Rv1.name, Rv2.name
from Rating_test r1, Rating_test r2, Reviewer_test rv1, Reviewer_test rv2
where r1.mID = r2.mID and r1.rID = rv1.rID and r2.rID = rv2.rID and rv1.name < rv2.name
order by rv1.name, rv2.name;

/* Do as required in tables for Q2 */
```

Data Output | Explain | Messages | Notifications

	name character varying (30)	name character varying (30)
1	man1	man2

It works. Also I write query for tables in Q3



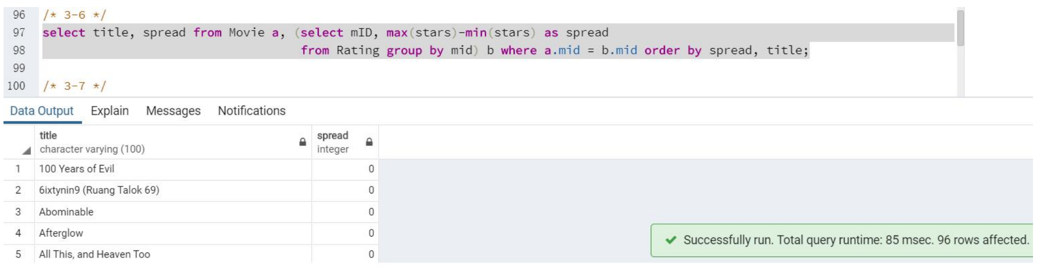
```
/* Do as required in tables for Q3 */
SELECT DISTINCT Rv1.name, Rv2.name
from Rating r1, Rating r2, Reviewer rv1, Reviewer rv2
where r1.mID = r2.mID and r1.rID = rv1.rID and r2.rID = rv2.rID and rv1.name < rv2.name
order by rv1.name, rv2.name;

/* 3-6 */
```

Data Output | Explain | Messages | Notifications

	name character varying (30)	name character varying (30)

Well, there is no pair.

3-6



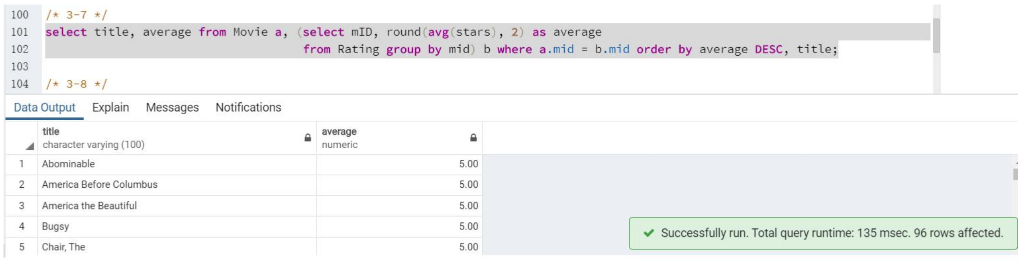
```
/* 3-6 */
select title, spread from Movie a, (select mID, max(stars)-min(stars) as spread
                                   from Rating group by mid) b where a.mid = b.mid order by spread, title;

/* 3-7 */
```

Data Output | Explain | Messages | Notifications

	title character varying (100)	spread integer
1	100 Years of Evil	0
2	6ixtynin9 (Ruang Talok 69)	0
3	Abominable	0
4	Afterglow	0
5	All This, and Heaven Too	0

✓ Successfully run. Total query runtime: 85 msec. 96 rows affected.

3-7



```
/* 3-7 */
select title, average from Movie a, (select mID, round(avg(stars), 2) as average
                                    from Rating group by mid) b where a.mid = b.mid order by average DESC, title;

/* 3-8 */
```

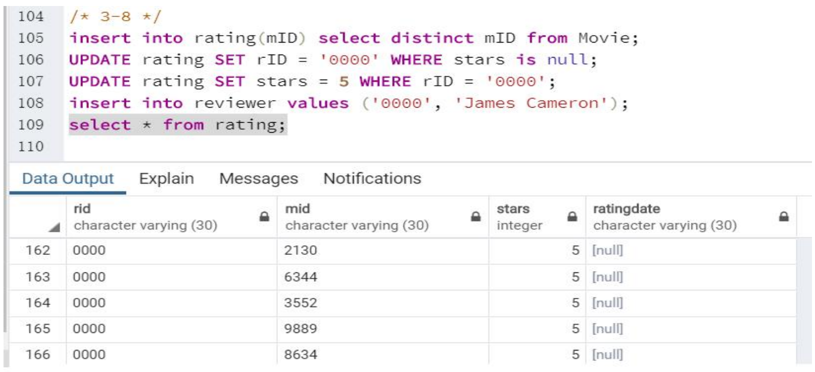
Data Output | Explain | Messages | Notifications

	title character varying (100)	average numeric
1	Abominable	5.00
2	America Before Columbus	5.00
3	America the Beautiful	5.00
4	Bugsy	5.00
5	Chair, The	5.00

✓ Successfully run. Total query runtime: 135 msec. 96 rows affected.

3-8

Set “0000” to represent James Cameron in the “Reviewer” table. Here is the adjusted “Rating” table.



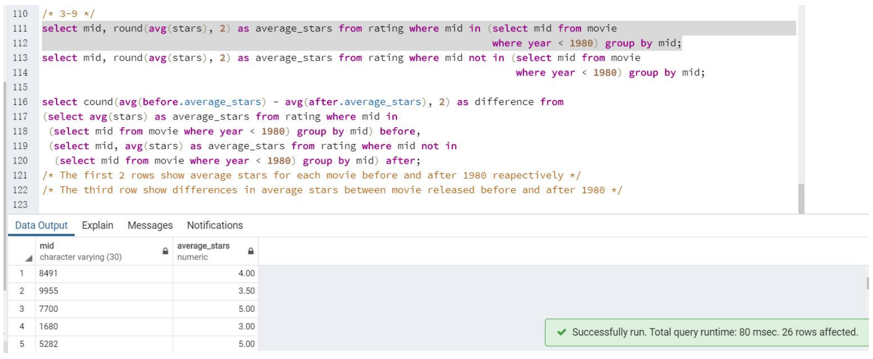
```
/* 3-8 */
insert into rating(mID) select distinct mID from Movie;
UPDATE rating SET rID = '0000' WHERE stars is null;
UPDATE rating SET stars = 5 WHERE rID = '0000';
insert into reviewer values ('0000', 'James Cameron');
select * from rating;
```

Data Output | Explain | Messages | Notifications

	rid character varying (30)	mid character varying (30)	stars integer	ratingdate character varying (30)
162	0000	2130	5	[null]
163	0000	6344	5	[null]
164	0000	3552	5	[null]
165	0000	9889	5	[null]
166	0000	8634	5	[null]

3-9

The first 2 pictures show the average stars for each movie before and after 1980 respectively, and the third screenshot shows differences in average stars between movie released before and after 1980

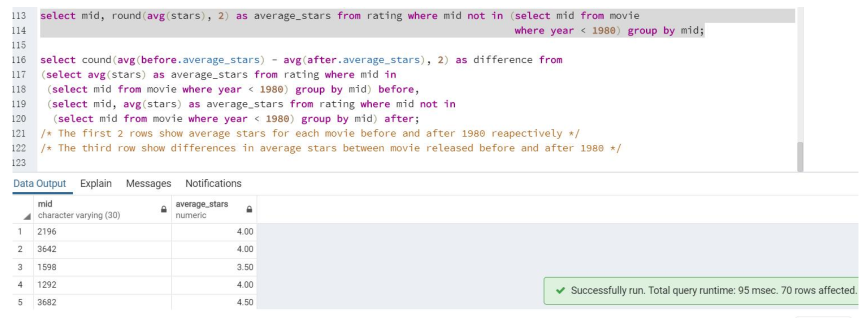


```
/* 3-9 */
select mid, round(avg(stars), 2) as average_stars from rating where mid in (select mid from movie
                                                                   where year < 1980) group by mid;
select mid, round(avg(stars), 2) as average_stars from rating where mid not in (select mid from movie
                                                                       where year < 1980) group by mid;

select cound(avg(before.average_stars) - avg(after.average_stars), 2) as difference from
(select avg(stars) as average_stars from rating where mid in
 (select mid from movie where year < 1980) group by mid) before,
 (select mid, avg(stars) as average_stars from rating where mid not in
  (select mid from movie where year < 1980) group by mid) after;
/* The first 2 rows show average stars for each movie before and after 1980 reapectively */
/* The third row show differences in average stars between movie released before and after 1980 */
```

Data Output | Explain | Messages | Notifications

```
   mid                         average_stars
   character varying (30)      numeric
1  8491                        4.00
2  9955                        3.50
3  7700                        5.00
4  1680                        3.00
5  5282                        5.00
```

✓ Successfully run. Total query runtime: 80 msec. 26 rows affected.

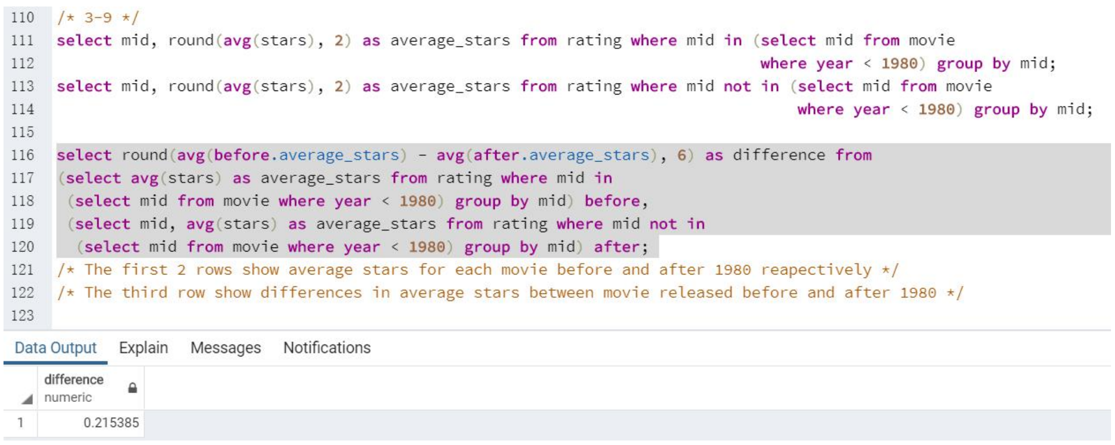


```
select mid, round(avg(stars), 2) as average_stars from rating where mid not in (select mid from movie
                                                                       where year < 1980) group by mid;

select cound(avg(before.average_stars) - avg(after.average_stars), 2) as difference from
(select avg(stars) as average_stars from rating where mid in
 (select mid from movie where year < 1980) group by mid) before,
 (select mid, avg(stars) as average_stars from rating where mid not in
  (select mid from movie where year < 1980) group by mid) after;
/* The first 2 rows show average stars for each movie before and after 1980 reapectively */
/* The third row show differences in average stars between movie released before and after 1980 */
```

Data Output | Explain | Messages | Notifications

```
   mid                         average_stars
   character varying (30)      numeric
1  2196                        4.00
2  3642                        4.00
3  1598                        3.50
4  1292                        4.00
5  3682                        4.50
```

✓ Successfully run. Total query runtime: 95 msec. 70 rows affected.



```
/* 3-9 */
select mid, round(avg(stars), 2) as average_stars from rating where mid in (select mid from movie
                                                                   where year < 1980) group by mid;
select mid, round(avg(stars), 2) as average_stars from rating where mid not in (select mid from movie
                                                                       where year < 1980) group by mid;

select round(avg(before.average_stars) - avg(after.average_stars), 6) as difference from
(select avg(stars) as average_stars from rating where mid in
 (select mid from movie where year < 1980) group by mid) before,
 (select mid, avg(stars) as average_stars from rating where mid not in
  (select mid from movie where year < 1980) group by mid) after;
/* The first 2 rows show average stars for each movie before and after 1980 reapectively */
/* The third row show differences in average stars between movie released before and after 1980 */
```

Data Output | Explain | Messages | Notifications

```
   difference
   numeric
1  0.215385
```