Primeira Avaliação (Remota) de SCC-240 – Bases de Dados
Primeiro semestre de 2020

Nome: Sidnei Gazola Junior
nº. USP: 9378888

Respostas

postgres/postgres@PostgreSQL 12

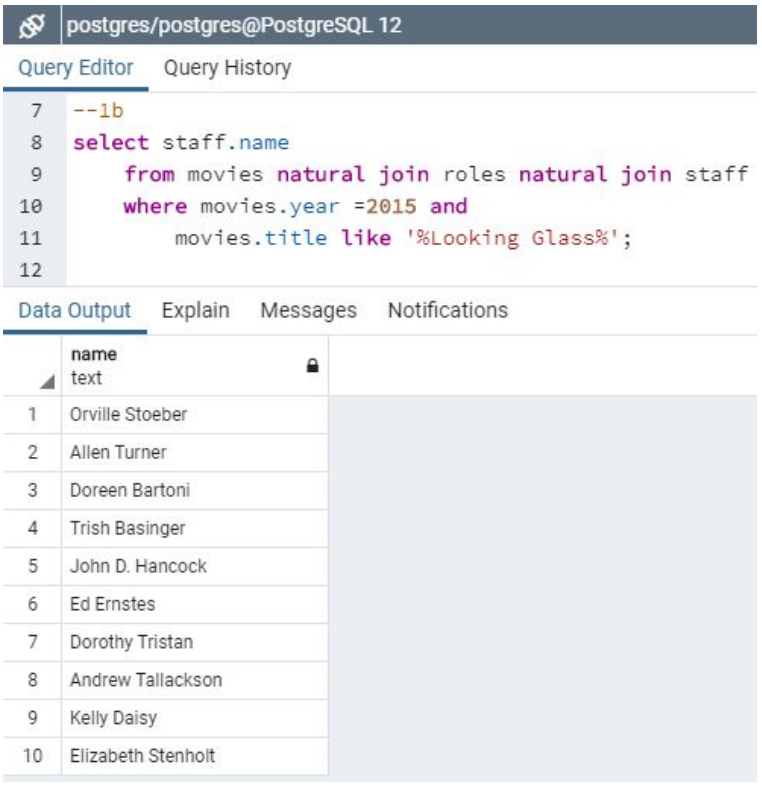
Query Editor | Query History

```
1  --1a
2  select roles.category, count(roles.personid)
3      from roles natural join movies
4      where movies.year =2019
5      group by roles.category;
```

Data Output | Explain | Messages | Notifications

	category text	count bigint
1	actor	14391
2	actress	8979
3	archive_foot...	142
4	cinematogra...	4114
5	composer	3599
6	director	7198
7	editor	2568
8	producer	6987
9	production_d...	527
10	self	2958
11	writer	4416

1.a) Primeiro, seleciona-se as categorias e suas respectivas contagens de atores pelo comando select. Para isso, cria-se uma junção da tabela roles com a tabela movies. Como o atributo movieid tem o mesmo domínio nas duas tabelas, pode-se usar natural join. Logo após, filtra-se com o comando where somente os filmes do ano de 2019. Por fim, usa-se o group by para agrupar as categorias para fazer a contagem de atores.



1.b) Primeiro, seleciona-se os nomes dos atores da tabela staff pelo comando select. Para isso, cria-se uma junção da tabela movies com a tabela roles, para pegar todos os personid de cada filme, pode-se usar natural join por meio do atributo movieid=movieid presente nas duas tabelas. Depois, cria-se novamente uma junção com a tabela staff para pegar o nome referente a cada personid, pode-se usar natural join por meio da coluna personid=personid presente nas duas tabelas. Depois, filtra-se os dados por meio do comando where: primeiro, os filmes que são de 2015, e, depois os que têm "Looking Glass" no nome.

1.c) Primeiro, seleciona-se somente o título do filme da tabela movies. Para isso, cria-se junção da tabela movies com a tabela roles, pode-se usar natural join por meio do atributo movieid=movieid presente nas duas tabelas, depois cria-se novamente uma junção com a tabela staff, pode-se usar natural join por meio da coluna personid=personid presente nas duas tabelas. Agora que se tem a lista de todos os nomes dos filmes, filtra-se pelo comando where os filmes que "Frances Conroy" ou "Edward Norton" tem participação, agora com essa lista de filmes que os dois participaram, verifica-se (com group by e having) quais filmes aparecem mais de uma vez porque nesse caso quer dizer que os dois participaram, que no caso é somente o filme “Stone”.

postgres/postgres@PostgreSQL 12

Query Editor | Query History

```
21  --1d
22  select count(distinct(title)) from movies
23      where title in(
24          select title from movies
25              group by title
26                  having count(title)>3
27      );
```

Data Output | Explain | Messages | Notifications

	count bigint
1	658

1.d) Seleciona-se (2º select) os nomes de filmes na tabela movies que aparecem mais de 3 vezes depois cria-se a contagem (1º select) de quantos nomes tem nessa tabela para mostrar quantos são, para assim ter quantos nomes de filmes são nome de mais do que 3 filmes.

postgres/postgres@PostgreSQL 12

Query Editor | Query History

```
29  --1e
30  select roles.category, movies.year, count(roles.personid)
31      from roles natural join movies
32      group by roles.category,movies.year;
33
```

Data Output | Explain | Messages | Notifications

	category text	year integer	count bigint
1	actor	2010	21162
2	actor	2011	23056
3	actor	2012	24474
4	actor	2013	26155
5	actor	2014	27512
6	actor	2015	27604
7	actor	2016	28942
8	actor	2017	30105
9	actor	2018	30230
10	actor	2019	14391
11	actor	2020	1105

1.e) Primeiro, seleciona-se todas as categorias de papéis nos filmes para cada ano, e depois selecionamos a contagem de participações. Para isso, cria-se junção da tabela movies com a tabela roles, por meio do atributo movieid=movieid, podemos fazer natural join. Usa-se o comando group by para agrupar cada categoria para cada ano, para fazer a contagem desejada.

1 2.a

$$\pi_{\{Title\}}(Movies \bowtie_{(MovieID=MovieID)}((\sigma_{Category=\text{“}producer\text{”}})Roles) \bowtie_{(PersonID=PersonID)}(\sigma(Name = \text{“}NicolasCage\text{”})Staff))$$

2 2.b

$$\pi_{\{Name\}}(Staff \bowtie_{(PersonID=PersonID)}((\sigma_{Category=\text{”}actress\text{”}})Roles) \bowtie_{(MovieID=MovieID)}(\sigma(Name = \text{”}RobertdeNiro\text{”})Staff) \bowtie_{(PersonID=PersonID)} Roles)))$$

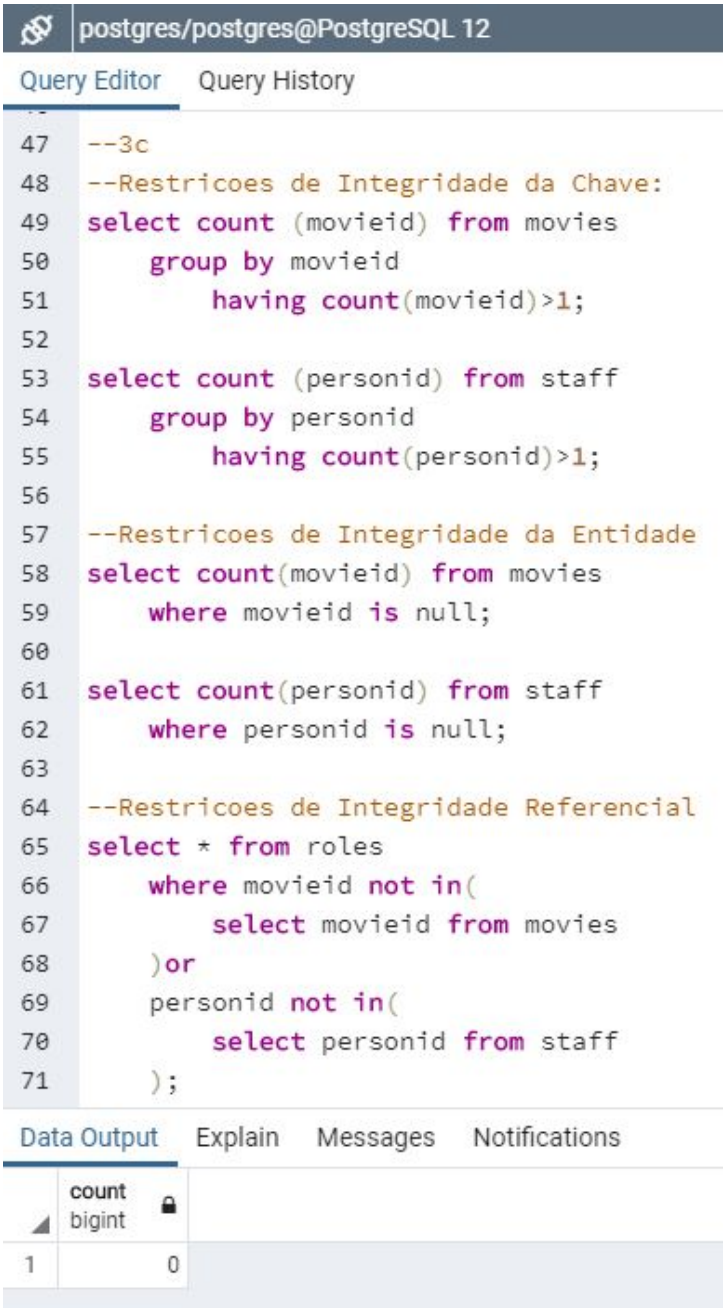
2.a) Primeiro, se faz a junção da tabela staff, filtrada pelo atributo name = “Nicolas Cage”, com a tabela roles, filtrada pelo atributo category = “producer, faz a junção pelo atributo personid=personid presente nas duas tabelas e com o mesmo domínio. Com a tabela resultante até agora e faz junção com a tabela movies, por meio do atributo movieid=movieid presente nas duas tabelas. Com a tabela final resultante filtra-se pelo atributo title, para mostrar a lista de títulos dos filmes em que "Nicolas Cage" participa como produtor.

2.b) Primeiro, se faz a junção da tabela staff, filtrada pelo atributo name = “Robert De Niro”, com a tabela roles, pelo atributo personid=personid presente nas duas tabelas e com o mesmo domínio. Agora nessa tabela resultante se tem todos os movieid que Robert De Niro participou, então faz-se novamente junção com a tabela roles original filtrada pelo atributo category = “actress”, fazemos junção pelo atributo movieid=movieid presente nas duas tabelas. Agora nessa tabela resultante temos todos os personid das atrizes que contracenaram com Robert De Niro, então, fazemos junção com a tabela staff, pelo atributo movieid=movieid. Agora selecionamos somente o atributo name, para mostrar somente o nome das atrizes.

Obs: As expressões foram escritas da esquerda para a direita,assim com na aula e nos slides de álgebra relacional.

3.a) Os atributos MovieID em Movies e PersonID em Staff se justificam o uso pelo fato de serem atributos identificadores e chaves primárias nas suas respectivas relações, sendo atributos que não possuem valores nulos ou duplicados. Eles fazem uma “ligação” dessas duas tabelas com a tabela roles.

3.b) Não há a necessidade de haver um atributo identificador na tabela roles, pois não há uma quarta tabela que necessite saber o papel que cada ator fez em um filme. Além de que há linhas repetidas em roles. E pelo fato de que com os atributos MovieID e PersonID já é possível fazer uma correta identificação.



3.c) Para garantir que as tabelas geradas são relações, tem-se que verificar se garantem as restrições de integridade do Modelo Relacional, para isso primeiro verifica-se as restrições de integridade da chave como mostra os comandos ao lado, nas chaves movieid da tabela movies e a chave personid da tabela staff, como nenhuma possui tuplas repetidas temos que essa restrição foi cumprida. Depois se verifica as restrições de integridade da entidade, para as duas mesmas chaves, como mostra o código ao lado, como nenhuma tem algum valor nulo, essa restrição também é cumprida. Depois verifica-se as restrições de integridade referencial da tabela roles, como mostra o código ao lado, como o domínio dos atributos movieid e personid bate com o domínio das tabelas movies e staff, e não há valores nesses atributos em roles que não esteja referenciado nas outras tabelas, podemos concluir que a restrição é cumprida. Portanto como temos todas as restrições cumpridas podemos garantir que as tabelas geradas são relações. **Obs:** Não foi printado a saída de cada comando porque todos tem essa mesma saída que já está nessa print.