

1. الف) دیتابیس PostgreSQL که شروع پروژه آن به سال 1986 برمیگردد اما در سال 2001 به صورت ACID atomicity, consistency, integrity, durability سازگار درآمد و اکثر سیستم عاملها را پشتیبانی میکند.
ب) بله میتوان از pgadmin استفاده کرد.
ج) uber, netflix, spotify, instagram, reddit
د) ترجیح به استفاده از زبان C بوده و استفاده به فریم‌ورک خاصی محدود نبوده.
2. الف) ایجاد سطوح مختلف انتزاع در سیستم کمک میکند تا بتوان کارها را راحت‌تر انجام داد و میزان کاری که باید در هر لایه انجام شود را کاهش می‌دهد. به عنوان مثال فرض کنید که در کار ذخیره یک رکورد جدید و ایجاد مدل جدول جدید در یک لایه باشد که در این صورت سرعت ذخیره کاهش یافته و باعث ایجاد سربار برای سیستم می‌شود.
ب) لایه فیزیکی مشخص میکند که یک رکورد چگونه در سیستم ذخیره می‌شود و لایه منطقی داده‌های ذخیره شده در دیتابیس و رابطه میان آنها را مشخص می‌کند.
ج) پایگاه داده رابطه‌ای: به این صورت که داده‌ها در جدول‌های مختلف ذخیره شده و برای دسترسی باید جداول را بررسی کنیم.
پایگاه داده‌های شی‌گرا: به این صورت که داده‌ها به صورت شی ذخیره میشوند. در این مدل داده‌های به صورت پیچیده ذخیره شده و رابطه میان داده‌ها به صورت مستقیم ذخیره می‌شود بر خلاف مدل رابطه‌ای که رابطه میان سطرها و ستون‌ها ذخیره میشد لذا این مدل برای برنامه‌ها با داده‌های پیچیده مناسب است. اشیا رابطه‌های many to many دارند و از طریق پوینترها رابطه‌ها قابل پیاده‌سازی است.
3. الف) ۱. فراوانی و ناسازگاری داده‌ها ۲. دسترسی سخت به اطلاعات ۳. سختی برای اضافه کردن محدودیت و تغییر داده‌های موجود ۴. محدودیت‌ها برای دسترسی همزمان چند کاربر
ب) بله یکی از عیب‌هایی که به این ساختار وارد است این است که در صورتیکه دچار سردرگمی شویم ممکن است جدول‌هایی بدون استفاده در ساختار ایجاد کنیم و باعث هدر رفتگی فضا شود هم چنین بعضی از دیتابیس‌های رابطه‌ای برای طول فیلدها دارای محدودیت‌هایی میباشند.
4. در مرحله اول primary key یک ویژگی یا مجموعه‌ای از ویژگی‌هاست که به صورت یکتا یک سطر جدول را مشخص میکند اما در candidate key عبارت است از ویژگی یا مجموعه ویژگی‌هایی که به عنوان primary key انتخاب نشده‌اند و در مورد super key هم مجموعه‌ای از ویژگی‌هاست که برای توصیف یک رکورد جدول مورد استفاده قرار میگیرند. به عنوان مثال برای یک کتاب فرض کنید موارد نام و شابک و نویسنده را داریم. Primary key میتواند نام شابک باشد یا ترکیب شابک و دو مورد باقی مانده از بین مواردی که باقی میمانند همگی candidate key میباشند.

5. برای مقداردهی مقادیری که گم شده اند به عنوان مثال فرض کنید که یک ستون جدید به جدول اضافه کنیم در این صورت رکوردهای قبلی برای این ستون مقداری نخواهند داشت که باعث مشکل خواهد شد. و دلیل دوم اینکه گاهی اوقات مقدار یک ستون برای یک رکورد نامشخص است و در این صورت هم باید نال برای عدم ایجاد مشکل تعریف شود.

6. عبارت ALL در صورتی نتیجه درست برمیگرداند که تمام مقادیر subquery ما شرایط مورد نیاز را برطرف کنند از طرفی ALL <> یک شرط متناقض را بیان میکند که عبارت هم باید کوچکتر هم باید بزرگتر از نتیجه ALL باشد پس نتیجه حاصل در ALL نیست که این معادل با عبارت not in میباشد که نتیجه ای که برمیگرداند در مقادیر subquery نیست.

به عنوان مثال در جدول زیر دو query زیر نتیجه یکسانی خواهند داشت:

```
Select name from instructor
Where name not in ('mozart' and 'einstein') ;

Select name from instructor
Where name <> ALL ('mozart' and 'einstein') ;
```

7. الف) برای انتخاب از جدول department میبایست جدول department را با جدول instructor که دارای department name است join کنیم که در اینجا ارور مربوط به join میباشد.

ب) نام و آیدی دانشجو هایی را برمیگرداند که درس زیست را پاس کرده اند.

```
Select S.ID, S.name
From student as S
Where ALL(Select course id
From course
Where dept_name='biology' AND course id IN
(Select T.course id
From takes As T
Where S.id=T.id));
```

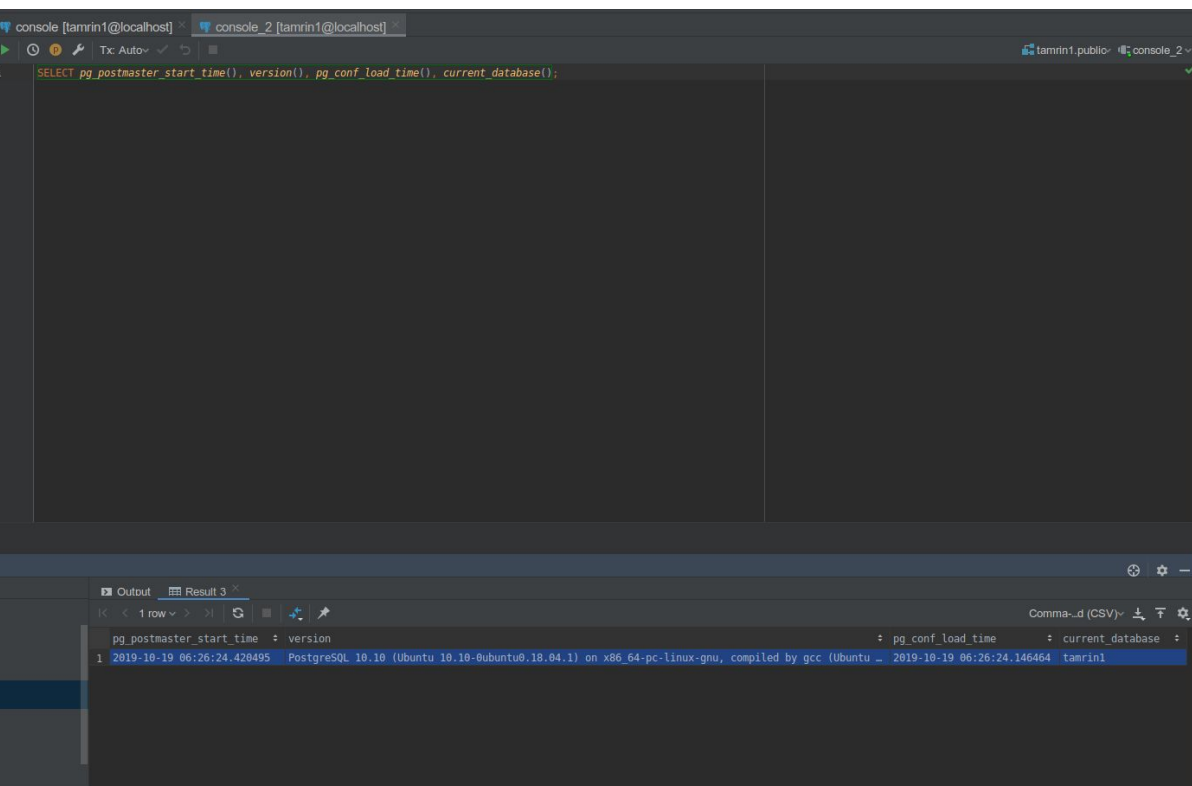
8.

الف) جدول هایی همچون staff, payment, inventory, rental, payment دارای اطلاعات پایه می باشند زیرا بقیه جدول ها دارای foreign key به مقادیری که در این جدول ها است می باشند.

ب) جدول هایی همچون customer, film, address, city در طول کار روزانه مرتباً رشد می کنند زیرا در طول روز مشتری های مختلفی به مراکز ما مراجعه میکنند همچنین در طول یک روز ممکن فیلم های زیادی

وارد مراکز ما شود.

.9



The screenshot shows a PostgreSQL console window with two tabs: 'console [tamrin1@localhost]' and 'console_2 [tamrin1@localhost]'. The active tab 'console_2' contains the following SQL query:

```
SELECT pg_postmaster_start_time(), version(), pg_conf_load_time(), current_database();
```

The result of the query is displayed in the 'Output' pane at the bottom, showing a single row of data:

pg_postmaster_start_time	version	pg_conf_load_time	current_database
2019-10-19 06:26:24.420495	PostgreSQL 10.10 (Ubuntu 10.10-0ubuntu0.18.04.1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 7.4.0-1ubuntu1~18.04) 7.4.0, 64-bit	2019-10-19 06:26:24.146464	tamrin1

10. a)

```
select customer.first_name, customer.last_name, address.city_id  
from customer  
inner join address on customer.address_id = address.address_id where address.address_id in  
(select address_id from address where city_id in  
(select city_id from city where country_id in  
(select country_id from country where country.country='Iran')));
```

b)

```
select first_name, last_name from actor where actor_id in (  
select actor_id from film_actor where film_id in (  
select film_id from inventory where inventory_id in(  
select inventory_id from rental where customer_id in(  
select customer_id  
from customer  
inner join address on customer.address_id = address.address_id where address.address_id in  
(select address_id from address where city_id in  
(select city_id from city where country_id in  
(select country_id from country where country.country='Iran'))))));
```

c)

```
select customer.first_name, customer.last_name, rental.return_date, rental.rental_date  
from customer  
inner join address on customer.address_id = address.address_id  
inner join rental on customer.customer_id = rental.customer_id
```

```

where address.address_id in
    (select address_id from address where city_id in
        (select city_id from city where country_id in
            (select country_id from country where country.country='Iran')))) and exists(select rental.customer_id from rental where
rental_date=return_date);

```

d)

```

select first_name, last_name from actor where actor_id in
    (select actor_id from film_actor where film_id in
        (select film_id from film where length>100 or rental_rate>4.00));

```

e)

```

select name from category inner join film_category fc on category.category_id = fc.category_id
where film_id in(
    select film_id from film where film_id in
        (select film_id from inventory where inventory_id=2) and rental_duration>90
        and film_id not in (select film_id from inventory where inventory_id=1));

```

f)

```

select film.title from film where lower(title) like '%g' or lower(title) like '%s%s%';

```

g)

```

select count(r.customer_id), r.staff_id from customer
inner join rental r on customer.customer_id = r.customer_id
inner join staff s on r.staff_id = s.staff_id
group by customer.customer_id, r.staff_id;

```

h)

```

create table category_rating(
    name character varying(25) NOT NULL,
    rental_rate numeric(4,2) DEFAULT 4.99 NOT NULL,
    length smallint,
    category_id smallint
);
insert into category_rating
select c.name, avg(rental_rate), max(length), c.category_id
from film
inner join film_category fc on film.film_id = fc.film_id
inner join category c on fc.category_id = c.category_id
group by (c.name, c.category_id)
order by avg(rental_rate) asc;

```

i)

```

with counts as(
    select film.rating, count(film.rating) as counter, category.name as cname, category.category_id
    from film
    inner join film_category on (film_category.film_id = film.film_id)
    inner join category on (category.category_id = film_category.category_id)
    group by category.category_id, film.rating
), maxs as (
    select cname, max(counter) as maxCount from counts group by cname
), maxReady as (
    select distinct on (maxs.cname) maxs.cname, counts.rating, counts.counter, category_id from counts
    inner join maxs on (maxs.cname = counts.cname and maxs.maxCount = counts.counter)
)
update category_rating

```

```

set age_group = (case
    when (select maxReady.rating from maxReady where maxReady.category_id = category_rating.category_id) = 'G' then 1
    when (select maxReady.rating from maxReady where maxReady.category_id = category_rating.category_id) = 'PG' then 2
    when (select maxReady.rating from maxReady where maxReady.category_id = category_rating.category_id) = 'PG-13' then 3
    else 4
end
)

```

j)

```

with categoryAvg as(
select avg(film.length) as len,category.name,category.category_id from film
inner join film_category on (film_category.film_id = film.film_id)
inner join category on (category.category_id = film_category.category_id)
group by category.name,category.category_id
),topThree as (
select * from category_rating order by category_rating.rental_rate desc limit 3
), averageTopThree as(
select avg(film.length) as len from film
inner join film_category on (film_category.film_id = film.film_id)
inner join category on (category.category_id = film_category.category_id)
inner join topThree on (topThree.category_id = category.category_id)
)

```

```

delete from category_rating using categoryAvg,averageTopThree where category_rating.category_id = categoryAvg.category_id
and averageTopThree.len < categoryAvg.len

```