



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)

به نام خدا

آزمایشگاه پایگاه داده  
جلسه سوم

## دوره‌ی مباحث پایگاه‌داده‌ی موردنیاز

► Joins: دستور Join در SQL برای اتصال دو جدول استفاده می‌شود یعنی از هر جدول یک ستون انتخاب می‌کنیم تا به هم الحاق کنیم و این دو ستون شامل اعدادی هستند که درمیان دو جدول مشترک هستند.

► Union clause: عملگر UNION برای ترکیب نتایج دو یا چند دستور SELECT استفاده می‌شود.

► Sub Queries: منظور از Subquery یک query است که درون یک عبارت مانند UPDATE، INSERT، SELECT یا DELETE نوشته می‌شود.

► Group by: با استفاده از دستور group by می‌توان دسته بندی یا گروه بندی ستون ها را انجام داد.

## تمرین پرس و جویهای پیچیده تر برای شروع

► **SELECT DISTINCT**: چنانچه در ستون های مورد جستجو ، موارد تکراری وجود داشته باشد در نتیجه خروجی نمایش داده خواهند شد . برای جلوگیری از چنین موردی و عدم نمایش موارد تکراری پس از دستور **Select** عبارت **DISTINCT** نوشته می شود.

► **WHERE CONDITION**: دستور **where** برای اضافه کردن شرط یا شرط هایی جهت محدود کردن نتایج جستجو و یا استخراج نتایج دقیقتر برای داشتن خروجی که در ذهن ما وجود دارد استفاده می شود . این دستور باید پس از دستور **select** و تعیین ستون ها از جدول مورد نظر به کار رود.

► با استفاده از عملگرهای **and** و **or** پرانتز می توان چندین شرط را با هم ترکیب کرد . خروجی برنامه با شرط هایی که روی دستور داده شده است مطابقت داده خواهد شد .

نکته: همتای کلید واژه **Distinct** ، **All** می باشد که **SQL Server** را برای بازگرداندن همه سطرها آگاه می سازد خواه آن واحد باشد یا خیر **All** . پیش فرض دستور **select** است ، پس نیازی به نوشتن آن نیست.

## تمرین پرس و جویهای پیچیده تر برای شروع

► AND و OR = برای ترکیب شرط ها در بخش Where در sql استفاده می شود .

► گاهی اوقات خروجی که ما می خواهیم بایستی چند شرط مختلف داشته باشد . به طور مثال افرادی را می خواهیم که سن بالای ۲۳ سال و مدرک تحصیلی بالای لیسانس داشته باشند . در این حالت بایستی هر کدام از شرط ها را جداگانه تعریف کرده و سپس آنها را با هم ترکیب کنیم . برنامه هر کدام از شرط ها را بررسی میکند و خروجی را نمایش میدهد .

► عملگر and برای اجرای دستور نیاز دارد تا تمام شرط های تعیین شده برای آن درست باشد .

► عملگر or فقط نیاز دارد که حداقل یکی از شرط ها درست باشد .

► ترکیب قیود AND و OR و = در شرط where :

- SELECT \*
- FROM employees
- WHERE (city = 'Miami' AND first\_name = 'Sarah')
- OR (employee\_id <= 2000);

نکته: همتای کلید واژه Distinct ، All می باشد که SQL Server را برای بازگرداندن همه سطرها آگاه می سازد خواه آن واحد باشد یا خیر All . پیش فرض دستور select است ، پس نیازی به نوشتن آن نیست .

## تمرین پرس وجوهای پیچیده تر برای شروع

### ▶ انواع JOIN

▶ **INNER JOIN:** در این دستور تنها سطریهایی برمی گردند که حداقل یک ردیف در هر دو جدول باشد که شرط پرس وجو را داشته باشد.

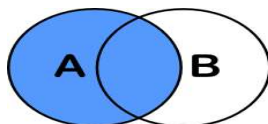
▶ **LEFT JOIN (LEFT OUTER JOIN):** در این دستور سطریهایی برمی گردند که داده هایشان در جدول چپ باشند، هر چند در جدول راست هیچ سطر منطبقی وجود نداشته باشد.

▶ **RIGHT JOIN (RIGHT OUTER JOIN):** در این دستور سطریهایی برمی گردند که داده هایی در جدول راست داشته باشند، هر چند در جدول چپ هیچ مورد منطبقی وجود نداشته باشد.

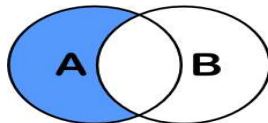
▶ **FULL JOIN:** در این دستور همه سطرها تا زمانی که در یکی از جدول ها شرط پرس وجو برقرار باشد بازگشت داده می شوند.

## تمرین پرس و جوهای پیچیده تر برای شروع

- **Select** column1\_name, column2\_name, column3\_name
- **From** table1\_name t
- **inner/left/right/full** join table2\_name b
- **on** t.pk\_column = b.pk\_column

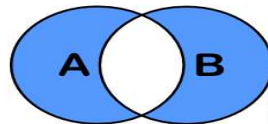
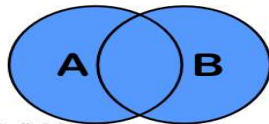


```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```

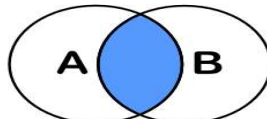


```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```

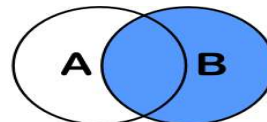
```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```



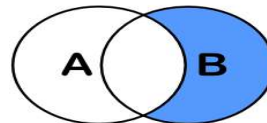
CHEATSHEET  
**SQL**  
JOINS



```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

## تمرین پرس و جویهای پیچیده تر برای شروع

► **ORDER BY** : اطلاعاتی که در دستور **select** در **sql** به عنوان خروجی نمایش داده میشود بی نظم و یا بهتر بگوییم بدون نظم مد نظر ما است. مقادیر خروجی در ستون های جدول بر اساس مقدار هیچ ستونی مرتب نمی شوند. با دستور **order by** میتوان اطلاعات جدول را بر اساس مقادیر یک یا چند ستون بر حسب شاخص هایی مثل ترتیب حروف الفبا ، بزرگتر یا کوچکتر بودن اعداد و ... مرتب کرد.

- **SELECT column\_name(s)**
- **FROM table\_name**
- **[Where]**
- **ORDER BY column\_name(s) [ASC|DESC]**

► نکته: همچنین این نیز امکان پذیرست که مرتب سازی را بر مبنای بیش از یک ستون انجام دهید. **ASC** به معنای صعودی بودن (**a to z**) و **DESC** به معنای نزولی بودن است. (**z to a**) پیش فرض **ACS** است.

## تمرین پرس و جوهای پیچیده تر برای شروع

- ▶ **GROUP By**: با استفاده از دستور **group by** می توان دسته بندی یک ستون بر حسب مقادیر مشابه فیلدهای یک ستون دیگر را انجام داد. در هنگام استفاده از برخی از توابع درون ساخته SQL که عمل محاسبه ( مثل مجموع و میانگین ) را بر روی داده ها انجام می دهند ، این مشکل وجود دارد که این توابع قادر به جدا کردن و متمایز کردن اطلاعات موجود در دو ستون نسبت به هم نیستند و نتایج محاسبات را به صورت کلی برای همه آنها در نظر می گیرند . در این مواقع از دستور **group by** استفاده می کنیم.
- ▶ نکته: گروه بندی را نباید روی ستون کلید اصلی جدول قرار داد چون تمام مقادیر یکتا هستند و آنگاه هر سطر را به یک گروه تبدیل می کند.

- ▶ **SELECT** column1, column2
- ▶ **FROM** table\_name
- ▶ **WHERE** [ conditions ]
- ▶ **GROUP BY** column1, column2
- ▶ **ORDER BY** column1, column2



## تمرین پرس و جوهای پیچیده تر برای شروع

► **HAVING** : دستور **having** برای افزودن شرط به توابع درون ساخته **SQL** استفاده می شود ، زیرا از دستور **where** نمی توان برای کار با مقادیر خروجی توابع درون ساخته **SQL** استفاده کرد . به عبارت دیگر دستور **having** در **sql** برای اعمال شرط به ستون ها اعمال می شود و همان کاری را می کند که **Where** در رکوردها انجام می دهد.

► دستور **having** معمولا با دستور **group by** می آید.

► از دستور **HAVING** به شکل زیر استفاده می شود:

- **SELECT** column\_name, aggregate\_function(column\_name)
- **FROM** table\_name
- **WHERE** column\_name operator value
- **GROUP BY** column\_name
- **HAVING** aggregate\_function(column\_name) operator value

## تمرین پرس و جوهای پیچیده تر برای شروع

► مثال دیگری برای درک بهتر استفاده از HAVING همراه با تابع sum و اعمال شرط بیشتر از ۲۰۰:

- **SELECT**
- customer\_id,
- SUM (amount)
- **FROM**
- payment
- **GROUP BY**
- customer\_id
- **HAVING**
- SUM (amount) > 200;

## تمرین پرس و جوهای پیچیده تر برای شروع

WITH CLAUSE ►

With همان دستور CTE است و CTE همان جدول مجازی.

CTE = Common Table Expression ►

برای ساخت CTE از کلمه کلیدی WITH استفاده می کنیم و یک نام برای CTE می گذاریم و SELECT مدنظر را اجرا می کنیم.

► WITH query\_name (column\_name1, ...) AS

► (SELECT ...)

## تمرین پرس و جوهای پیچیده تر برای شروع

```
► WITH cte_film AS (  
►     SELECT  
►         film_id, title,  
►         (CASE  
►             WHEN length < 30 THEN 'Short'  
►             WHEN length < 90 THEN 'Medium'  
►             ELSE 'Long'  
►         END) length  
►     FROM film  
► )  
► SELECT  
►     film_id,  
►     title,  
►     length  
► FROM cte_film  
► WHERE length = 'Long'  
► ORDER BY title;
```

## تمرین پرس و جوهای پیچیده تر برای شروع

```
▶ WITH cte_rental AS (  
▶     SELECT staff_id,  
▶           COUNT(rental_id) rental_count  
▶     FROM   rental  
▶     GROUP BY staff_id  
▶ )  
▶ SELECT s.staff_id,  
▶       first_name,  
▶       last_name,  
▶       rental_count  
▶ FROM   staff s  
▶       INNER JOIN cte_rental USING (staff_id);
```

▶ در این مثال ابتدا CTE نتایج شامل staff id و تعداد rental را برمی گرداند سپس جدول staff را با دستور cte با استفاده از ستون staff id ترکیب می کند.

# تمرین پرس و جوهای پیچیده تر برای شروع

## NESTED QUERY یا SUBQUERY ▶

▶ وقتی یک ساختار SQL داخل یک ساختار SQL قرار می گیرد Subquery نامیده می شود.

▶ نکته : به SQL که داخل آن یک SQL دیگر قرار دارد Parent statement هم می گویند.

- ▶ **SELECT** column\_name [, column\_name ]
- ▶ **FROM** table1 [, table2 ]
- ▶ **WHERE** column\_name OPERATOR
- ▶ ( **SELECT** column\_name [, column\_name ]
- ▶ **FROM** table1 [, table2 ]
- ▶ [**WHERE**])

# تمرین پرس و جوهای پیچیده تر برای شروع

▶ نمونه یک پرس و جو در یک پرس و جوی دیگر:

```
▶ SELECT
▶     column_1
▶     , column_2
▶     , column_3
▶ FROM
▶     tbl_data
▶ WHERE
▶     column_1 IN -- this can also be "NOT IN", "EXISTS, an operator like "=", "<", and others.
▶     (
▶     SELECT
▶         column_1
▶     FROM
▶         tbl_data
▶     WHERE
▶         [condition]
▶     )
▶ ORDER BY column_1
```

## دستورات قابل استفاده در پرس و جو ها:

EXIST/NOT EXIST ►

- **SELECT** \*
- **FROM** products
- **WHERE** NOT EXISTS (**SELECT** 1
- **FROM** inventory
- **WHERE** products.product\_id = inventory.product\_id);
- **SELECT** count(aid),bid **FROM** pgbench\_accounts **WHERE**
- bid <> ALL(**SELECT** bid **FROM** pgbench\_branches **WHERE** bbalance > 0)
- **GROUP BY** bid;

ALL ►



## دستورات قابل استفاده در پرس و جو ها:

▶ **LIKE**: دستور LIKE برای پیدا کردن تشابه در متن و مقدار STRING در پرس و جو استفاده می شود و با دو کاراکتر % و \_ می آید.

▶ “%” نشان دهنده صفر یا یک یا چند کاراکتر است و “\_” نشان دهنده تک کاراکتر است و می توانند باهم نیز استفاده شوند

```
▶ SELECT FROM table_name
▶ WHERE column LIKE 'XXXX%'
▶ or
▶ SELECT FROM table_name
▶ WHERE column LIKE '%XXXX%'
▶ or
▶ SELECT FROM table_name
▶ WHERE column LIKE 'XXXX_'
▶ or
▶ SELECT FROM table_name
▶ WHERE column LIKE '_XXXX'
▶ or
▶ SELECT FROM table_name
```

## پروژه

- ▶ گروه‌ها، دو یا سه نفره باشد
- ▶ طراحی جداول و پرس و جوها به صورت مناسب، کاربردی و منطقی از اهمیت ویژه‌ای برخوردار است
- ▶ تعداد جداول اصلی دستکم ۵ عدد باشد ( برای گروه‌های سه نفره، ۶ جدول)
- ▶ موجودیت کاربران حتما وجود داشته باشد
- ▶ نحوه‌ی عملکرد کاربران باید به گونه‌ای باشد که بر یکدیگر اثری نداشته باشند؛ به طور مثال تاثیر انتقال پول در یک بانک بر روی موجودی افراد و یا تاثیر حرکت بازیکن اول بر بازیکن دوم، لذا در انتخاب موضوع پروژه دقت شود.
- ▶ حداقل ۵ کاربر وجود داشته باشد
- ▶ بصورت سلسله مراتبی مجوزهای مختلفی باید داشته باشند. بطور مثال یکی از کاربران در نقش مدیریت، همه مجوزها را داشته باشد و افراد دیگر متناسب با نقششان محدود تر باشند (کاربران دیگر می‌توانند همگی یک یا تعداد بیشتری نقش داشته باشند)
- ▶ در رابطه با دسترسی کاربران در این پایگاه داده تمهیداتی صورت گرفته باشد به طور مثال کاربران نباید بتوانند برخی از جداولی که نباید دسترسی داشته باشند را کنترل کنند.
- ▶ پایگاه داده حداقل ۳ رویه (procedure) داشته باشد ( برای گروه‌های ۳ نفره، دستکم ۴ رویه داشته باشد)
- ▶ پایگاه داده حداقل ۳ آغازگر (trigger) داشته باشد ( برای گروه‌های ۳ نفره، دستکم ۴ آغازگر داشته باشد)
- ▶ دلیل استفاده از آغازگرها حائز اهمیت است. بخشی از منطق کل سیستم می‌تواند با استفاد از آغازگرها باشد. به طور مثال بررسی موجودی کاربران پس از هر فعالیتی و یا ایجاد جدولی به منظور لاگ گرفتن.
- ▶ پایگاه داده نیازمند حداقل ۳ تابع به منظور تسهیل پرس و جوها می‌باشد ( برای گروه‌های ۳ نفره، دستکم ۴ آغازگر داشته باشد)

## پروژه

- قسمت مهم پروژه ایجاد پرس و جوهای است که منجر به گزارش گیری می شود.
- به همین منظور تعداد حداقل ۶ پرس و جو با استفاده از مباحثی که در آزمایشگاه مطرح شده، برای پایگاه داده نیاز است.
- این پرس و جوها باید پیچیدگی لازم را داشته باشند. بنابراین در هنگام انتخاب موضوع دقت شود.
- مباحث مربوط به گزارش گیری شامل `grouping`، `rollup`، `having`، `cube`، `aggregation functions`، `window function` و ... می باشد.
- بطور مثال پرس و جویی که منجر به نشان دادن فروش فصلی، ماهانه و روزانه برای یک فروشگاه خاص و یا کالای خاص.
- زبان بک اند
  - اجباری است
  - هر زبانی می تواند باشد
  - زبان (فریم ورک) فرانت اند
  - اختیاری
  - هر زبان یا چارچوبی می تواند باشد (فلاتر، ری اکت، html/css و ...)
  - نمره ی اضاف
- هدف از انجام این پروژه در مرحله اول انتخاب و طراحی ساختار جداول، ایجاد پرس و جوهای مرتبط با گزارش گیری بصورت منطقی و هوشمند و در ادامه پیاده سازی می باشد. مستندات باید به صورت خلاصه توضیحی از موضوع پروژه و پرس و جوهای استفاده شده باشد که پرس و جوهای مربوط به گزارش گیری حائز اهمیت است.
- پایگاه داده
  - PostgreSQL و یا MongoDB

## بخشی از پروژه

تمرین زیر به عنوان بخشی از پروژه قلمداد می‌شود و ۱ نمره از ۱۱ نمره‌ی پروژه را به خود اختصاص می‌دهد:

- مشخص کردن هم‌گروهی‌ها
- مشخص کردن زبان بک‌اند (اجباری)
- مشخص کردن پایگاه‌داده‌ی مورد استفاده (اجباری): PostgreSQL و یا MongoDB
- مشخص کردن زبان فرانت‌اند (اختیاری): برای نمونه فلاتر، HTML/CSS، ری‌اکت و ...
- ارائه‌ی یک توضیح کوتاه (حدود دو خط) در مورد ماهیت پروژه

## تمرین

- ۱- از ترکیب دستورات GROUP BY و HAVING سه پرسوجوی متفاوت بنویسید و خروجی آن در جداول خودتان را بفرستید و آن را توضیح دهید.
- ۲- یک دستور مرتب سازی شده براساس دو ستون خاص با ترکیب intersection بنویسید و خروجی آن را بفرستید.
- ۳- یک عبارت CTE تعریف کنید و با یک دستور پرسوجوی دیگر ترکیب کنید و خروجی آن را بفرستید.
- ۴- با دو دستور مختلف دو تا از join های مختلف را استفاده کنید که خروجی هایشان متفاوت باشد و توضیح دهید.
- ۵- دو دستور تو در تو بنویسید که ۳ دستور را باهم ترکیب کند و حتما در خروجی از تک تک دستورات زیر استفاده کنید:
  - All➤
  - any➤
  - <➤
  - is null➤
  - like➤
  - Exist➤

➤ نام پرونده‌ی ارسالی: HW03\_99876543.pdf