



دانشگاه صنعتی امیر کبیر  
(پلی تکنیک تهران)

به نام خدا

## آزمایشگاه پایگاه داده جلسه چهارم

Window functions, Crosstab, CUBE, ROLLUP

# ROLLUP

- ▶ دستوری است که به کمک آن میتوان پرس وجوهایی را طراحی و اجرا کرد که ماهیت گزارشی دارند
- ▶ به عنوان گزارش های مدیریتی در نظر گرفته می شوند که قرار است مبدأ تصمیم سازی های مدیریتی باشند

## نحوه استفاده

- ▶ عبارت ROLLUP را بعد از عبارت BY GROUP می آوریم
- ▶ فیلدهایی که می خواهیم بر اساس آن ها گزارش تهیه شود را به عنوان مؤلفه های ROLLUP در نظر می گیریم
- ▶ با اینکار ROLLUP موجب اضافه شدن یک سطر بصورت مجموع مقادیر محاسبه شده برای تابع AGGREGATE (براساس مؤلفه تعریف شده در ROLLUP) می شود.

► فرض کنید جدول فروشی داریم که حاوی نام برند، بخش و تعداد است و می خواهیم بدانیم که علاوه بر این که هر برند در هر بخش چه تعدادی فرخته است، می خواهیم این مورد را نیز اضافه کنیم که یک برند در همه بخش ها، و همه برندها در همه بخش ها چه فروشی داشته اند.

**SELECT**

```
case grouping(brand)
when 0 then brand
when 1 then 'All brands'
end as brand,
case grouping(segment)
when 0 then segment
when 1 then 'All segment'
end as segment,
SUM (quantity)
FROM
    sales
GROUP BY
    ROLLUP (brand, segment)
ORDER BY
    brand,
    segment;
```

► پرس و جوی مقابل با استفاده از ROLLUP این گزارش را می دهد.

	brand character varying	segment character varying	sum bigint
1	ABC	All segment	300
2	ABC	Basic	200
3	ABC	Premium	100
4	All brands	All segment	700
5	XYZ	All segment	400
6	XYZ	Basic	300
7	XYZ	Premium	100

عبارت **GROUPING(brand)** به منظور بررسی گروه بندی شدن یا نشدن هر رکورد استفاده شده است. در صورتی که نتیجه یک رکورد حاصل گروه بندی بر اساس فیلد **brand** باشد مقدار یک و در غیر این صورت مقدار صفر بر می گرداند.

# CUBE

- ▶ این دستور در واقع مشابه **ROLLUP** است اما بصورت پیشرفته‌تر، گزارش های دقیق تری را تولید می‌کند.
- ▶ این دستور در واقع به ازای هر گروه‌بندی ممکن از داده‌ها، یک مقدار متناسب با تابع **AGGREGATE** استفاده شده در پرس و جو را به مجموعه نتایج اضافه می‌کند.

► به عنوان مثال، ROLLUP پرس و جوی مثال قبل را با CUBE جایگزین می‌کنیم.

```
SELECT
case grouping(brand)
when 0 then brand
when 1 then 'All brands'
end as brand,
  case grouping(segment)
when 0 then segment
when 1 then 'All segment'
end as segment,
  SUM (quantity)
FROM
  sales
GROUP BY
  cube (brand, segment)
ORDER BY
  brand,
  segment;
```

	brand character varying	segment character varying	sum bigint
1	ABC	All segment	300
2	ABC	Basic	200
3	ABC	Premium	100
4	All brands	All segment	700
5	All brands	Basic	500
6	All brands	Premium	200
7	XYZ	All segment	400
8	XYZ	Basic	300
9	XYZ	Premium	100

همانطور که مشخص است دو ردیف به جدول اضافه شده است که شامل ردیف ۵ و ۶ می‌شود. این دو ردیف گروه بندی brand با تک تک segment ها می‌باشد.

# تفاوت CUBE و ROLLUP

► تفاوت این دو دستور در نوع انتخاب گروه بندی است.

► CUBE همه زیرمجموعه های ممکن را در نظر می گیرد ولی در ROLLUP ترتیب و تقدم مهم است.

**مثال:** مجموعه (c1,c2,c3) را در نظر بگیرید. ► گروه بندی با استفاده از CUBE:

- (c1, c2, c3)
- (c1, c2)
- (c2, c3)
- (c1,c3)
- (c1)
- (c2)
- (c3)
- ()

► گروه بندی با استفاده از ROLLUP:

- (c1, c2, c3)
- (c1, c2)
- (c1)
- ()

**سوال:** اگر بخواهیم گروه بندی انجام دهیم که خروجی آن گزارشی از میزان فروش سالیانه، ماه و روزانه در سال ها و ماه ها و روز های خاص باشد باید از کدام دستور استفاده کرد.

# Window Functions

- معمولا از این نوع توابع روی مجموعه ای از سطرهای یک جدول، در جهت اعمال عملیاتهای محاسباتی، ارزیابی داده‌ها، رتبه بندی و غیره ... استفاده میگردد.
- به بیان ساده‌تر به وسیله Window Function ها میتوان، سطرهای یک جدول را گروه بندی نمود. و روی گروه‌ها از توابعی مثل توابع تجمعی (Functions Aggregate) استفاده کرد.
- این نوع توابع از قابلیت و انعطاف پذیری زیادی برخوردار هستند و به وسیله آنها میتوان نتایج (خروجی) بسیار مفیدی را از پرس و جو ها بدست آورد.
- معمولا از این نوع توابع در داده کاوی و گزارش گیری ها استفاده میگردد
- کلمه "Window" در Window Function، به مجموعه سطرهایی اشاره میکند، که محاسبات و ارزیابی و ... روی آنها اعمال میگردد.

## مثال WINDOW FUNCTION

► فرض کنید جدول کالایی به شکل زیر وجود دارد، که حاوی نام کالا، قیمت و گروه کالا می باشد.

	product_id [PK] integer	product_name character varying (255)	price numeric (11,2)	group_id integer
1	1	Microsoft Lumia	200.00	1
2	2	HTC One	400.00	1
3	3	Nexus	500.00	1
4	4	iPhone	900.00	1
5	5	HP Elite	1200.00	2
6	6	Lenovo Thinkpad	700.00	2
7	7	Sony VAIO	700.00	2
8	8	Dell Vostro	800.00	2
9	9	iPad	700.00	3
10	10	Kindle Fire	150.00	3
11	11	Samsung Galaxy Tab	200.00	3

► مطابق با توضیحات داده شده می خواهیم پرس و جویی را اجرا کنیم که در نهایت به ازای هر کالا مشخص کند که میانگین قیمت گروهی که کالا در آن است چه مقدار است.

► برای این کار نیاز است محدوده window را به اندازه ای داشته باشیم که در آن شناسه گروه (group\_id) یکسان باشد و در این محدوده تابع میانگین را اجرا کنیم.



## مثال

## SELECT

	product_name character varying (255)	price numeric (11,2)	group_name character varying (255)	avg numeric
1	HP Elite	1200.00	Laptop	850.0000000000000000
2	Lenovo Thinkpad	700.00	Laptop	850.0000000000000000
3	Sony VAIO	700.00	Laptop	850.0000000000000000
4	Dell Vostro	800.00	Laptop	850.0000000000000000
5	Microsoft Lumia	200.00	Smartphone	500.0000000000000000
6	HTC One	400.00	Smartphone	500.0000000000000000
7	Nexus	500.00	Smartphone	500.0000000000000000
8	iPhone	900.00	Smartphone	500.0000000000000000
9	iPad	700.00	Tablet	350.0000000000000000
10	Kindle Fire	150.00	Tablet	350.0000000000000000
11	Samsung Galaxy Tab	200.00	Tablet	350.0000000000000000

همانطور که مشخص است مقدار میانگین صرفاً در محدوده تعیین شده محاسبه شده است. تابع `avg` بر روی محدوده ای اجرا شده است که با کلیدواژه `over` بیان شده و به هر مجموعه ای که توسط `over` انتخاب می‌شود، یک محدوده یا پنجره گفته می‌شود. کلید واژه `partition` بیان گر گروه بندی یا پارتیشن بندی بوسیله یک ستون مشخص است.

# Window Functions

▶ window function ها همیشه بعد از Join ، where ، Group by و Having انجام می شوند و پس از آن order by اجرا می شود.

▶ حال این نکته حائز اهمیت است که می توان این محدوده را هوشمندانه تر انتخاب کرد. برای این کار می توان ساختار window function ها را بررسی کرد، که به صورت زیر است.

- ▶ window\_function(arg1, arg2,..) OVER (
- ▶ [PARTITION BY partition\_expression]
- ▶ [ORDER BY sort\_expression [ASC | DESC] [NULLS {FIRST | LAST }]])

▶ تابع window\_function یک تابع تجمعی است که می تواند ورودی های مختلفی داشته باشد. و همانطور که در قبل گفته شد کلید واژه over محدوده مورد نظر را انتخاب می کند.

▶ بعد از پارتیشن بندی می توان مشخص کرد که در هر پارتیشنی ردیف ها با چه ترتیبی در خروجی ظاهر شوند.

▶ در خصوص مقادیری که مقدار نداشته باشند یعنی NULLS، می توان مشخص کرد که در ابتدا یا در انتها ظاهر شوند، این کار با order by و FIRST یا LAST مشخص می شود.

# Window Functions

► توابعی که می توان استفاده کرد انواع مختلف دیگری نیز دارد که در لینک زیر برخی از آنها نشان داده شده است.

► <https://www.postgresql.org/docs/9.1/functions-window.html>

► بطور مثال توابع ROW\_NUMBER() ، RANK() ، DENSE\_RANK() عددی را برای هر ردیف از یک محدوده یا پنجره متناسب با ترتیب آنها انتخاب می کند. با این تفاوت که ROW\_NUMBER() رتبه بندی را بصورت پی در پی انجام می دهد.

پرس و جوی زیر مثالی از تابع ROW\_NUMBER() است. ▶

```
SELECT
  product_name,
  group_name,
  price,
  ROW_NUMBER () OVER (
    PARTITION BY group_name
    ORDER BY
      price
  )
FROM
  products
INNER JOIN product_groups
  USING (group_id);
```

	product_name character varying (255)	group_name character varying (255)	price numeric (11,2)	row_number bigint
1	Sony VAIO	Laptop	700.00	1
2	Lenovo Thinkpad	Laptop	700.00	2
3	Dell Vostro	Laptop	800.00	3
4	HP Elite	Laptop	1200.00	4
5	Microsoft Lumia	Smartphone	200.00	1
6	HTC One	Smartphone	400.00	2
7	Nexus	Smartphone	500.00	3
8	iPhone	Smartphone	900.00	4
9	Kindle Fire	Tablet	150.00	1
10	Samsung Galaxy Tab	Tablet	200.00	2
11	iPad	Tablet	700.00	3

► تابع RANK() مقدار یکسانی برای مقادیر یکسان در نظر می‌گیرد. پرس وجوی زیر مثالی از آن است.

```
SELECT
  product_name,
  group_name,
  price,
  RANK () OVER (
    PARTITION BY group_name
    ORDER BY
      price
  )
FROM
  products
INNER JOIN product_groups
  USING (group_id);
```

	product_name character varying (255)	group_name character varying (255)	price numeric (11,2)	rank bigint
1	Sony VAIO	Laptop	700.00	1
2	Lenovo Thinkpad	Laptop	700.00	1
3	Dell Vostro	Laptop	800.00	3
4	HP Elite	Laptop	1200.00	4
5	Microsoft Lumia	Smartphone	200.00	1
6	HTC One	Smartphone	400.00	2
7	Nexus	Smartphone	500.00	3
8	iPhone	Smartphone	900.00	4
9	Kindle Fire	Tablet	150.00	1
10	Samsung Galaxy Tab	Tablet	200.00	2
11	iPad	Tablet	700.00	3

سوال: فرق تابع DENSE\_RANK() با تابع RANK() چیست؟

▶ تابع FIRST\_VALUE و LAST\_VALUE اولین و آخرین مقدار در هر محدوده را انتخاب می‌کند. در پرس‌وجوی زیر کمترین قیمت در هر گروه برای هر ردیف مشخص شده است.

```
SELECT
    product_name,
    group_name,
    price,
    FIRST_VALUE (price) OVER (
        PARTITION BY group_name
        ORDER BY
            price
    ) AS lowest_price_per_group
FROM
    products
INNER JOIN product_groups
    USING (group_id);
```

	product_name character varying (255)	group_name character varying (255)	price numeric (11,2)	lowest_price_per_group numeric
1	Sony VAIO	Laptop	700.00	700.00
2	Lenovo Thinkpad	Laptop	700.00	700.00
3	Dell Vostro	Laptop	800.00	700.00
4	HP Elite	Laptop	1200.00	700.00
5	Microsoft Lumia	Smartphone	200.00	200.00
6	HTC One	Smartphone	400.00	200.00
7	Nexus	Smartphone	500.00	200.00
8	iPhone	Smartphone	900.00	200.00
9	Kindle Fire	Tablet	150.00	150.00
10	Samsung Galaxy Tab	Tablet	200.00	150.00
11	iPad	Tablet	700.00	150.00

# ORDER BY

- ▶ با استفاده از ORDER BY می‌توان محدوده ای را درون پنجره یا محدوده پارتیشن بندی شده ایجاد کرد.
- ▶ به طور مثال می‌خواهیم در هر گروه قیمت کالا را با قیمت کالای قبلی خود جمع کنیم. در این حالت علاوه بر این که پارتیشن بندی باید متناسب با گروه باشد، باید مشخص کرد که در زمان محاسبه هر ردیف فقط کالای قبلی و کالای فعلی را در نظر بگیرد. پرس‌وجوی زیر نشان دهنده همین حالت است.

## SELECT

```
product_name,  
group_name,  
price,  
sum (price) OVER (  
    PARTITION BY group_name  
    ORDER BY  
        price ROWS  
        BETWEEN 1 PRECEDING  
        AND CURRENT ROW  
    ) AS highest_price_per_group  
FROM  
    products  
INNER JOIN product_groups  
    USING (group_id);
```

	Data Output	Explain	Messages	Notifications
	product_name character varying (255)	group_name character varying (255)	price numeric (11,2)	highest_price_per_group numeric
1	Sony VAIO	Laptop	700.00	700.00
2	Lenovo Thinkpad	Laptop	700.00	1400.00
3	Dell Vostro	Laptop	800.00	1500.00
4	HP Elite	Laptop	1200.00	2000.00
5	Microsoft Lumia	Smartphone	200.00	200.00
6	HTC One	Smartphone	400.00	600.00
7	Nexus	Smartphone	500.00	900.00
8	iPhone	Smartphone	900.00	1400.00
9	Kindle Fire	Tablet	150.00	150.00
10	Samsung Galaxy Tab	Tablet	200.00	350.00
11	iPad	Tablet	700.00	900.00

## توابع

## LEAD و LAG

- ▶ کاربرد این توابع هنگامی است که نیاز به ردیف های بعدی یا قبلی در یک پارتیشن وجود دارد.
- ▶ بطور مثال پرس وجوی زیر در هر گروه نشان می دهد که قیمت کالای قبلی چه مقدار بوده است.
- ▶ ورودی های این دو تابع به ترتیب، عملیات محاسباتی که بر روی یک ستون انجام می شود، تعداد ردیفی که قبل تر یا بعدتر باید به آن دسترسی داشته باشد و مقدار پیش فرض در صورت نبود ردیف قبلی یا بعدی. پرس وجوی زیر مثالی از این مورد است.

### SELECT

```
product_name,  
group_name,  
price,  
LAG (price, 1) OVER (  
    PARTITION BY group_name  
    ORDER BY  
        price  
) AS prev_price,  
price - LAG (price, 1) OVER (  
    PARTITION BY group_name  
    ORDER BY  
        price  
) AS cur_prev_diff  
FROM  
    products  
INNER JOIN product_groups USING (group_id);
```

	product_name character varying (255)	group_name character varying (255)	price numeric (11,2)	prev_price numeric	cur_prev_diff numeric
1	Sony VAIO	Laptop	700.00	[null]	[null]
2	Lenovo Thinkpad	Laptop	700.00	700.00	0.00
3	Dell Vostro	Laptop	800.00	700.00	100.00
4	HP Elite	Laptop	1200.00	800.00	400.00
5	Microsoft Lumia	Smartphone	200.00	[null]	[null]
6	HTC One	Smartphone	400.00	200.00	200.00
7	Nexus	Smartphone	500.00	400.00	100.00
8	iPhone	Smartphone	900.00	500.00	400.00
9	Kindle Fire	Tablet	150.00	[null]	[null]
10	Samsung Galaxy Tab	Tablet	200.00	150.00	50.00
11	iPad	Tablet	700.00	200.00	500.00



# Crosstab

- یکی افزونه های پایگاه داده PostgresDB توابعی هستند که به آن ها tablefunc گفته می شود.
- ویژگی این توابع این است که در خروجی آن ها یک جدول است.
- یکی از این توابع CROSSTAB نام دارد. این تابع در واقع جایگزین PIVOT TABLE ها در این پایگاه داده می باشد.
- برای استفاده از این تابع باید افزونه tablefunc را بصورت زیر فعال کرد.
- **CREATE EXTENSION IF NOT EXISTS tablefunc;**
- به کمک این دستور میتوان جای سطر و ستون را در گزارش عوض کرد به عبارت بهتر به جای آنکه چندین سطر داشته باشیم، جهت راحت شدن تجزیه و تحلیل گزارش، تعدادی ستون به گزارش اضافه می کنیم.

# CROSSTAB

► ساختار کلی این تابع بصورت زیر است.

► `crosstab(text source_sql, text category_sql)`

► ورودی اول یعنی `source_sql` پرس و جویی است که اطلاعات اصلی از آن نشئت می‌گیرد، این پرس‌وجو باید یک ستون `row_name`، یک ستون `category` و یک ستون `value` داشته باشد. (نام ستون‌ها در اینجا با کاربریشان تناسب دارد). ستون `row_name` حتما باید اولین ستون باشد. `category`، `value` باید حتما آخرین ستون‌ها باشند. هر تعداد ستونی می‌تواند بعد از `row_name` باشد.

► ورودی دوم یعنی `category_sql` باید پرس و جویی باشد که تنها یک ستون، حداقل یک ردیف داشته باشد و بدون ردیف تکراری.

► در نهایت خروجی این تابع به این صورت است که به ازای هر مقدار یکتا در `row_name` یک ردیف ایجاد می‌کند و مقدار `row_name` به همراه ستون‌های اضافی (به غیر از دو ستون آخر) را بدون تغییر در آن ردیف می‌گذارد.

► مقدار `value` نیز با توجه به این که در ستون `category` چه مقداری وجود دارد، در ستونی که متعلق به آن `category` است قرار می‌گیرد. اگر مقدار `category` با هیچکدام از مقادیر خروجی پرس و جوی `category_sql` یکسان نباشد آن ردیف نادیده گرفته می‌شود.

فرض شود جدول فروشی مثل جدول زیر وجود دارد و پرس وجوی مقابل را روی آن اجرا می کنیم.

```
select * from crosstab(
  'select year, month, qty from sales order by
  1,
  'select m from generate_series(1,12) m'
) as (
  year int,
  "Jan" int,
  "Feb" int,
  "Mar" int,
  "Apr" int,
  "May" int,
  "Jun" int,
  "Jul" int,
  "Aug" int,
  "Sep" int,
  "Oct" int,
  "Nov" int,
  "Dec" int
);
```

	year integer	month integer	qty integer
1	2007	1	1000
2	2007	2	1500
3	2007	7	500
4	2007	11	1500
5	2007	12	2000
6	2008	1	1000

	year integer	Jan integer	Feb integer	Mar integer	Apr integer	May integer	Jun integer	Jul integer	Aug integer	Sep integer	Oct integer	Nov integer	Dec integer
1	2007	1000	1500	[null]	[null]	[null]	[null]	500	[null]	[null]	[null]	1500	2000
2	2008	1000	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]

همانطور که مشخص است، خروجی به شکلی است که انگار جای سطر و ستون category عوض شده است. این نوع پرس وجو در گزارش گیری بسیار مورد استفاده است.

## تمرین

۱. با استفاده از `crosstab` پرس‌وجویی بنویسید که مشخص کند تعداد فیلم اجاره ای مربوط به هر `store` در هر `category` چه تعداد بوده است.
۲. پرس‌وجویی بنویسید که مجموع `amount` اجاره فیلم هارا بر اساس روز، ماه و سال را با استفاده از `ROLLUP` گزارش دهد. بخشی از خروجی به شکل زیر است.

	year double precision	mmonth double precision	dday double precision	sum numeric
1	2005	6	14	41.89
2	2005	6	15	1179.97
3	2005	6	16	1191.11
4	2005	6	17	1158.19
5	2005	6	18	1284.99
6	2005	6	19	1283.92
7	2005	6	20	1223.09
8	2005	6	21	986.69
9	2005	6	[null]	8349.85
10	2005	7	5	128.73

## تمرین

۳. در سوال قبل بجای ROLLUP از CUBE استفاده کنید و توضیح دهید چه چیزی را نشان می‌دهد.
۴. با استفاده از window functions پرسو جویی بنویسید که در جدول film، فیلم‌ها را بر اساس length رتبه‌بندی کند. این رتبه‌بندی باید در هر category باشد.

**توجه:** در تمرین، از جداولی استفاده شده است که در پایگاه داده DVD rental وجود دارد. این پایگاه داده یک نوع پایگاه داده آموزشی است. برای دسترسی به آن می‌توانید کل پایگاه داده را دانلود کرده و با استفاده از فایل tar، آن را در پایگاه داده خود بازیابی کنید. مشخصات کامل این پایگاه داده نمونه به همراه model ER آن در لینک زیر وجود دارد.

<https://www.postgresqltutorial.com/postgresql-sample-database>

## بخش دوم پروژه

- ▶ یک پروژه در گیت‌لب بسازید و تمامی اعضای گروه را به آن اضافه کنید
- ▶ بررسی کامیتهای اعضای گروه
- ▶ افزودن مدرس آزمایشگاه به پروژه
- ▶ شناسه‌ی مدرس آزمایشگاه در گیت‌لب: msalimi