به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس آزمایشگاه پایگاه‌داده

دستورکار هفتم

نام و نام خانوادگی	**نام نام خانوادگی**
شماره دانشجویی	**۸۱۰۱۹۹۴۶۱**
تاریخ ارسال گزارش	**۱۴۰۲.۰۹.۳۰**

فهرست

پاسخ ۱. توابع پنجره ای ۱

۱-۱. **اضافه کردن جدول** student_score ۱

۱-۲. **محاسبه کمترین و بیشترین** score ۲

۱-۳. **اضافه کردن** PARTITION BY ۳

۱-۴. ROW_NUMBER ۴

۱-۵. RANK ۴

۱-۶. Dense RANK ۵

۱-۷. LAG ۶

۱-۸. Frame ۷

پاسخ ۲ - تریگرها ۸

۲-۱. اضافه کردن جدول users ۸

۲-۱. اضافه کردن password_hasher trigger ۹

۲-۱. حذف کردن trigger ۱۰

پاسخ ۱. توابع پنجره ای

۱-۱. اضافه کردن جدول student_score

با استفاده از دستورات زیر شمای جدول را تعیین کرده و مقادیر آن را مشخص می کنیم.

```sql
DROP TABLE IF EXISTS student_score;
CREATE TABLE student_score (
student_id SERIAL PRIMARY KEY,
student_name VARCHAR(30),
dep_name VARCHAR(40),
score INT
);
INSERT INTO student_score VALUES (11, 'Ibrahim', 'Computer Science', 80);
INSERT INTO student_score VALUES (7, 'Taiwo', 'Microbiology', 76);
INSERT INTO student_score VALUES (9, 'Nurain', 'Biochemistry', 80);
INSERT INTO student_score VALUES (8, 'Joel', 'Computer Science', 90);
INSERT INTO student_score VALUES (10, 'Mustapha', 'Industrial Chemistry', 78);
INSERT INTO student_score VALUES (5, 'Muritadoh', 'Biochemistry', 85);
INSERT INTO student_score VALUES (2, 'Yusuf', 'Biochemistry', 70);
INSERT INTO student_score VALUES (3, 'Habeebah', 'Microbiology', 80);
INSERT INTO student_score VALUES (1, 'Tomiwa', 'Microbiology', 65);
INSERT INTO student_score VALUES (4, 'Gbadebo', 'Computer Science', 80);
INSERT INTO student_score VALUES (12, 'Tolu', 'Computer Science', 67);
```

student_score

	student_id	student_name	dep_name	score
1	11	Ibrahim	Computer Science	80
2	7	Taiwo	Microbiology	76
3	9	Nurain	Biochemistry	80
4	8	Joel	Computer Science	90
5	10	Mustapha	Industrial Chemistry	78
6	5	Muritadoh	Biochemistry	85
7	2	Yusuf	Biochemistry	70
8	3	Habeebah	Microbiology	80
9	1	Tomiwa	Microbiology	65
10	4	Gbadebo	Computer Science	80
11	12	Tolu	Computer Science	67

۱-۲. محاسبه کمترین و بیشترین score

همان طور که از دستور زیر مشخص است، می خواهیم بیشترین و کمترین مقدار score را مشخص کنیم.

```
*<postgres> Script-11
SELECT
    *,
    MAX(score) OVER() AS maximum_score,
    MIN(score) OVER() AS minimum_score

FROM student_score;
```

student_score 1 | Output

SELECT *, MAX(score) OVER() AS maximum_sc

	student_id	student_name	dep_name	score	maximum_score	minimum_score
1	11	Ibrahim	Computer Science	80	90	
2	7	Taiwo	Microbiology	76	90	
3	9	Nurain	Biochemistry	80	90	
4	8	Joel	Computer Science	90	90	
5	10	Mustapha	Industrial Chemistry	78	90	
6	5	Muritadoh	Biochemistry	85	90	

در اینجا بدون استفاده از توابع پنجره ای و (همان طور که ذکر شده) با استفاده از subquery ها می توانیم خروجی را نشان دهیم:

```
*<postgres> Script-11
SELECT
    *,
    MAX(score) OVER() AS maximum_score,
    MIN(score) OVER() AS minimum_score

FROM student_score;
```

student_score 1 | Output

SELECT *, MAX(score) OVER() AS maximum_sc

	student_id	student_name	dep_name	score	maximum_score	minimum_score
1	11	Ibrahim	Computer Science	80	90	
2	7	Taiwo	Microbiology	76	90	
3	9	Nurain	Biochemistry	80	90	
4	8	Joel	Computer Science	90	90	
5	10	Mustapha	Industrial Chemistry	78	90	

ore	maximum_score	minimum_score
80	90	65
76	90	65
80	90	65
90	90	65
78	90	65
85	90	65

در اینجا مشخصا کمترین امتیاز ۶۵ و بیشترین ۹۰ است.

در بخش اول با استفاده از OVER() کوئری را نوشته ایم که مشخصا ساده تر است و چون برای OVER مقداری در نظر گرفته نشده است (داخل پرانتز چیزی نیست) از پنجره ی تمام سطور استفاده میشود.

۱-۳. اضافه کردن PARTITION BY

*<postgres> Script-11

```
SELECT
    *,
    MAX(score)OVER(PARTITION BY dep_name) AS dep_maximum_score,
    ROUND(AVG(score)OVER(PARTITION BY dep_name), 2) AS dep_average_score
FROM student_score;
```

student_score 1 | Output

SELECT *, MAX(score)OVER(PARTITION BY dep

	student_id	student_name	dep_name	score	dep_maximum_score	dep_average
1	2	Yusuf	Biochemistry	70	85	
2	5	Muritadoh	Biochemistry	85	85	
3	9	Nurain	Biochemistry	80	85	
4	12	Tolu	Computer Science	67	90	
5	8	Joel	Computer Science	90	90	
6	4	Gbadebo	Computer Science	80	90	

در این جا با PARTITION BY که مانند GROUP BY عمل می کند و با OVER()، سطر ها را می توانیم گروه بندی کنیم و سپس aggregate را انجام بدهیم.

مثلا در کوئری ذکر شده، بر اساس dep_name سطر ها را دسته بندی می کنیم و سپس بیشترین score را بدست میاوریم. از آنجا که ۴ دپارتمان داریم، خروجی ما به ۴ گروه تقسیم شده است و در هر کدام بیشترین و میانگین score نمایش داده شده است.

۱-۴. ROW_NUMBER

*<postgres> Script-11

```
SELECT
    *,
    ROW_NUMBER() OVER(ORDER BY student_name) AS name_serial_number
FROM student_score;
```

student_score 1

SELECT *, ROW_NUMBER() OVER(ORDER BY st | Enter a SQL expression to filter results (use Ctrl+Space)

	student_id	student_name	dep_name	score	name_serial_number
1	4	Gbadebo	Computer Science	80	1
2	3	Habeebah	Microbiology	80	2
3	11	Ibrahim	Computer Science	80	3
4	8	Joel	Computer Science	90	4
5	5	Muritadoh	Biochemistry	85	5
6	10	Mustapha	Industrial Chemistry	78	6
7	9	Nurain	Biochemistry	80	7

با استفاده از این تابع، می‌توانیم به ردیف‌های پنجره عددی اختصاص دهیم. در کوئری بالا، ردیف‌ها برای پنجره مشخص شده، بر اساس نام دانش‌آموز مرتب شده و سپس ستون name_serial_number را اضافه می‌کنیم، که به ترتیب اعداد از ۱ به ردیف‌ها نسبت می‌دهد. که نهایتا سطرها از ۱ تا ۱۱ شماره گذاری شده اند.

۱-۵. RANK

```
SELECT
    *,
    RANK()OVER(PARTITION BY dep_name ORDER BY score DESC)
FROM student_score;
```

SELECT *, RANK()OVER(PARTITION BY dep_nar | Enter a SQL expression to filter results (use Ctrl+Spac

	student_id	student_name	dep_name	score	rank
1	5	Muritadoh	Biochemistry	85	1
2	9	Nurain	Biochemistry	80	2
3	2	Yusuf	Biochemistry	70	3
4	8	Joel	Computer Science	90	1
5	11	Ibrahim	Computer Science	80	2
6	4	Gbadebo	Computer Science	80	2
7	12	Tolu	Computer Science	67	4

استفاده از این تابع به ما امکان می‌دهد رکوردهای پنجره را مرتب کنیم. در کوئری بالا، پنجره‌ها بر اساس dep_nameتقسیم شده و در هر یک، بر اساس score مرتب می‌شوند و سپس به ترتیب آنها رتبه‌بندی می‌شود. همانطور که مشاهده می‌شود، در هر پنجره که dep_name یکسانی دارند، ستون rank بر اساس score رتبه‌بندی می‌شود. در پنجره Computer Science دو دانشجویی با امتیاز یکسان دیده می‌شود و به همین دلیل، رتبه آنها نیز یکسان در نظر گرفته شده است و رتبه نفر بعدی فاصله و gap عددی دارد.

۶-۱. DENSE RANK

```
SELECT
    *,
    DENSE_RANK()OVER(PARTITION BY dep_name ORDER BY score DESC)
FROM student_score;
```

SELECT *, DENSE_RANK()OVER(PARTITION BY ... Enter a SQL expression to filter results (use Ctrl+Space)

	student_id	student_name	dep_name	score	dense_rank
1	5	Muritadoh	Biochemistry	85	1
2	9	Nurain	Biochemistry	80	2
3	2	Yusuf	Biochemistry	70	3
4	8	Joel	Computer Science	90	1
5	11	Ibrahim	Computer Science	80	2
6	4	Gbadebo	Computer Science	80	2

تابعی مشابه تابع RANK است، با این تفاوت که در صورت برابری رتبه دو رکورد در پنجره، رکورد بعدی رتبه‌ای پیوسته می‌گیرد و هیچ فاصله‌ای بین اعداد وجود ندارد .کوئری بالا تقریباً مشابه کوئری قبلی است، با این تفاوت که این‌بار از تابع DENSE_RANK استفاده شده است: در پنجرهComputer Science ، رتبه‌ها به ترتیب آمده‌اند و بعد از دو رتبه ۲، رتبه ۳ ادامه پیدا کرده است. در قسمت قبلی که از RANK استفاده شده بود، بعد از دو رتبه ۲، رتبه ۴ ظاهر شده بود.

۱-۷. LAG

*<postgres> Script-11

```
SELECT
    *,
    LAG(score) OVER(PARTITION BY dep_name ORDER BY score)
FROM student_score;
```

SELECT *, LAG(score) OVER(PARTITION BY dep... Enter a SQL expression to filter results (use Ctrl+Spac

	student_id	student_name	dep_name	score	lag
1	2	Yusuf	Biochemistry	70	[NULL]
2	9	Nurain	Biochemistry	80	70
3	5	Muritadoh	Biochemistry	85	80
4	12	Tolu	Computer Science	67	[NULL]
5	11	Ibrahim	Computer Science	80	67
6	4	Gbadebo	Computer Science	80	80

با استفاده از این تابع، می‌توانیم بخشی از اطلاعات رکورد قبلی را در رکورد فعلی نگه داریم. در کوئری بالا، پنجره‌ها بر اساس dep_name تقسیم شده و بر اساس score مرتب می‌شوند. حال با استفاده از تابع LAG بر رویscore ، مقدار score ردیف قبلی را در هر پنجره به دست می‌آوریم. در این مثال، در هر پنجره،

ردیف اول مقداری در lag ندارد؛ این به دلیل عدم وجود رکورد قبلی است. امّا برای سایر رکوردها، مقدار lagهمان مقدار score در رکورد قبلی است.

۸-۱. Frame

```
*<postgres> Script-11
SELECT
    *,
    SUM(score)OVER(ORDER BY student_id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cummulative_sum
FROM student_score
```

SELECT *, SUM(score)OVER(ORDER BY student | Enter a SQL expression to filter results (use Ctrl+Space)

	student_id	student_name	dep_name	score	cummulative_sum
1	1	Tomiwa	Microbiology	65	65
2	2	Yusuf	Biochemistry	70	135
3	3	Habeebah	Microbiology	80	215
4	4	Gbadebo	Computer Science	80	295
5	5	Muritadoh	Biochemistry	85	380
6	7	Taiwo	Microbiology	76	456

با استفاده از frame ها، می‌توانیم برای هر ردیف، منطقه‌ای اطراف آن را برای محاسبه مجموع تعیین کنیم. این کار با استفاده از توابع RANK انجام نمی‌شود. برای تعیین یکframe ، از کلیدواژه ROWS استفاده می‌کنیم. سپس محدوده قبل و بعد از ردیف کنونی برای frame را با استفاده از کلمات کلیدی N PRECEDINGو N FOLLOWING (که N می‌تواند یک عدد یا UNBOUNDED باشد) تعیین می‌کنیم. همچنین CURRENT ROW به ردیف کنونی اشاره دارد.

در کوئری بالا، پنجره تمامی ردیف‌ها را شامل می‌شود و بر اساس student_id مرتب شده‌اند. سپس برای هر ردیف، یک frame برای محدوده‌ی کلی از ردیف‌های قبل تا ردیف کنونی در نظر گرفته شده و مجموع scoreدر این frame محاسبه می‌شود. این کار باعث می‌شود به مجموع تجمعی score برسیم.

در نتیجه، هر ردیف از cummulative_sum ، معادل با جمع score خود ردیف و تمامیscore های قبل از آن است.

پاسخ ۲ – تریگرها

۲-۱. اضافه کردن جدول users

```
*<postgres> Script-11
CREATE TABLE
    users (
        fullname VARCHAR(120),
        email VARCHAR(120),
        username VARCHAR(30),
        password VARCHAR(60)
    );
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE users (fullname VARCHAR(120), email VARCHAR(120), username VARCHAR(30), password VARCHAR(60))
Start time	Fri Dec 22 15:28:43 IRST 2023

در کل، trigger ها به جداول متصل می‌شوند و در صورتی که رویدادINSERT ، UPDATE یا DELETE بر روی یک جدول رخ دهد، کد trigger قبل یا بعد از انجام آن رویداد اجرا می‌شود. با استفاده از دستور CREATE TRIGGER، یک trigger جدید تعریف می‌کنیم و سپس مشخص می‌کنیم که این trigger باید قبل یا بعد از انجام هرکدام از عملیات‌هایINSERT ، UPDATE یا DELETE بر روی هر سطر اجرا شود.

۲-۲. اضافه کردن password_hasher trigger

```
CREATE TRIGGER password_hasher BEFORE INSERT ON users FOR EACH ROW
SET
    NEW.password = MD5 (NEW.password);
```

توضیح: یک trigger ساخته می‌شود که قبل از درج رکورد جدید در جدول، فیلد رمز عبور را به شکل هش شده تغییر می‌دهد. کلمه کلیدی NEW در زمان استفاده از BEFORE INSERT/UPDATE وجود دارد و به رکورد جدید اشاره می‌کند. به همین ترتیب، کلمه کلیدی OLD در زمان استفاده از AFTER DELETE/UPDATE وجود دارد و به رکورد مورد نظر قبل از تغییر یا حذف اشاره می‌کند.

سینتکس داده شده برای MySQL است و در PostgreSQL کار نمیکند. باید به این شکل تغییر پیدا کند:

```
*<postgres> Script-11   users   *<postgres> Script-10   *<postgres>

CREATE OR REPLACE FUNCTION password_hashing()
RETURNS TRIGGER AS $$
BEGIN
NEW.password = MD5(NEW.password);
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER password_hasher
BEFORE INSERT ON users
FOR EACH ROW
EXECUTE FUNCTION password_hashing();
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE OR REPLACE FUNCTION password_hashing()
	RETURNS TRIGGER AS $$
	BEGIN
	NEW.password = MD5(NEW.password);
	RETURN NEW;
	END;
	$$ LANGUAGE plpgsql;
	CREATE TRIGGER password_hasher

توضیح: به عنوان یک روش درPostgreSQL ، باید ابتدا یک تابع تعریف کرده و سپس کد مربوط به trigger را داخل آن قرار دهیم. در نهایت، با استفاده از EXECUTE FUNCTION درtrigger ، این تابع را صدا می‌زنیم. حال تریگر ساخته شده را امتحان می کنیم:

```
INSERT INTO
    users
VALUES
    (
        'idris babu',
        'zubs@test.com',
        'zubby1',
        'password'
    );
```

users 1

select * from users u where u.fullname = 'idris b

	fullname	email	username	password
1	idris babu	zubs@test.com	zubby1	5f4dcc3b5aa765d61d8327deb882cf99

در اینجا مشاهده می شود که بعد insert، مقدار password به صورت MD5 ذخیره شده است.

۲-۳. حذف کردن trigger

برای حذف شدن دستور پایین زده می شود:

```
DROP TRIGGER password_hasher ON users;
```

Statistics 1

Name	Value
Updated Rows	0
Query	DROP TRIGGER password_hasher ON users
Start time	Fri Dec 22 15:52:22 IRST 2023
Finish time	Fri Dec 22 15:52:22 IRST 2023

حال آزمایش می کنیم که آیا حذف شده است تریگر یا نه:

```
INSERT INTO
    users
VALUES
    (
        'ahmad',
        'zubs@test.com',
        'zubby1',
        'password'
    );

select *
from users u
where u.fullname = 'ahmad';
```

users 1 | Statistics 1

INSERT INTO users VALUES ('ahmad', 'zubs@te | Enter a SQL expression to filter resul

	fullname	email	username	password
1	ahmad	zubs@test.com	zubby1	password

که می بینیم هش نشده است.