مریم سعیدمهر
ش.د. :۹۶۲۹۳۷۳
استاد مربوطه : دکتر قدیری

تکلیف سری سوم پایگاه داده ها ۱
دانشگاه صنعتی اصفهان
ترم اول سال تحصیلی ۹۸ – ۹۹

بخش اول

۱. در مورد transaction ها تحقیق کنید و به سوالات زیر پاسخ دهید:

I. خواص ACID را برای transaction نام برده و مختصراً توضیح دهید.

- **Atomicity : Either all operations of the transaction are reflected properly in the database, or none are.**
- **Consistency : Execution of a transaction in isolation (that is, with no other transaction executing concurrently) preserves the consistency of the database.**
- **Isolation : Even though multiple transactions may execute concurrently, the system guarantees that, for every pair of transactions T_i and T_j, it appears to T_i that either T_j finished execution before T_i started or T_j started execution after T_i finished. Thus, each transaction is unaware of other transactions executing concurrently in the system.**
- **Durability : After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.**

II. چرا اجرای همزمان transaction های طولانی یا transaction هایی که با I/O یا storage کار میکنند ، مهم تر از بقیه است ؟

Multiple transactions are allowed to run concurrently in the system.

Advantages are:

- **Increased processor and disk utilization, leading to better transaction throughput**

  e.g., one transaction can be using the CPU while another is reading from or writing to the disk

- **Reduced average response time for transactions**

  short transactions need not wait behind long ones.

به دو دلیل فوق اجرای همزمان transaction های طولانی یا transaction هایی که با I/O یا storage کار میکنند ، مهم تر از بقیه است

III. تفاوت اجرای serial و serializable را برای زمان بندی اجرای transaction در DBMS توضیح دهید.

Serial Schedules : Schedules in which the transactions are executed non-interleaved. a serial schedule is one in which no transaction starts until a running transaction has ended are called serial schedules.

Serializable : This is used to maintain the consistency of the database. It is mainly used in the Non-Serial scheduling to verify whether the scheduling will lead to any inconsistency or not.

Note : a serial schedule does not need the serializability because it follows a transaction only when the previous transaction is complete.

IV. حالت اولیه A=0 , B=0 است و شرط Consistency پایگاه $A = 0 \vee B = 0$ باشد :

آیا اجرای سریال این دو شرط Consistency را حفظ میکند؟

```
T2: read(B);
    read(A);
    if B = 0 then A := A + 1;
    write(A).
```

```
T1: read(A);
    read(B);
    if A = 0 then B := B + 1;
    write(B).
```

Yes : if first T1 then T2 : (T1) B==0→A=1 (T2) A!=0 → B=0

if first T2 then T1 : (T2) A==0→B=1 (T1) B!=0 → A=0

as you can see in both of them Consistency condition is true.

۲. برای گزینه های A و B در مورد Materialized View تحقیق کنید.

A) تفاوت آن با Simple View چیست ؟

Materialized Views: Certain database systems allow view relations to be physically stored. Physical copy created when the view is defined. Such views are called Materialized view. If relations used in the query are updated, the materialized view result becomes out of date. Need to maintain the view, by updating the view whenever the underlying relations are updated.

B) در چه کاربردهایی از Materialized Views استفاده میشود ؟

access will be faster so if you have a fairly large set of data that you often will need access to but doesn't often change, it may be a good place for a mat view.

C) آیا میتوان View قابل به روزرسانی از اطلاعات ساخت ؟ اگر بله شرط آن چیست ؟

View Updates in SQL : Most SQL implementations allow updates only on simple views.
1-The from clause has only one database relation.
2-The select clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
3-Any attribute not listed in the select clause can be set to null.
4-The query does not have a group by or having clause.

D) یکی از معایب استفاده از View ؟

Performance:Views create the appearance of a table, but the DBMS must still translate queries against the view into queries against the underlying source tables. If the view is defined by a complex, multi-table query then simple queries on the views may take considerable time.
Update restrictions:When a user tries to update rows of a view, the DBMS must translate the request into an update on rows of the underlying source tables. This is possible for simple views, but more complex views are often restricted to read-only. Note that if we have materializes view , integrity will be also a problem.

۳. چهار تفاوت function و stored procedure ؟

1-The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values.

2-Functions can have only input parameters for it whereas Procedures can have input or output parameters.

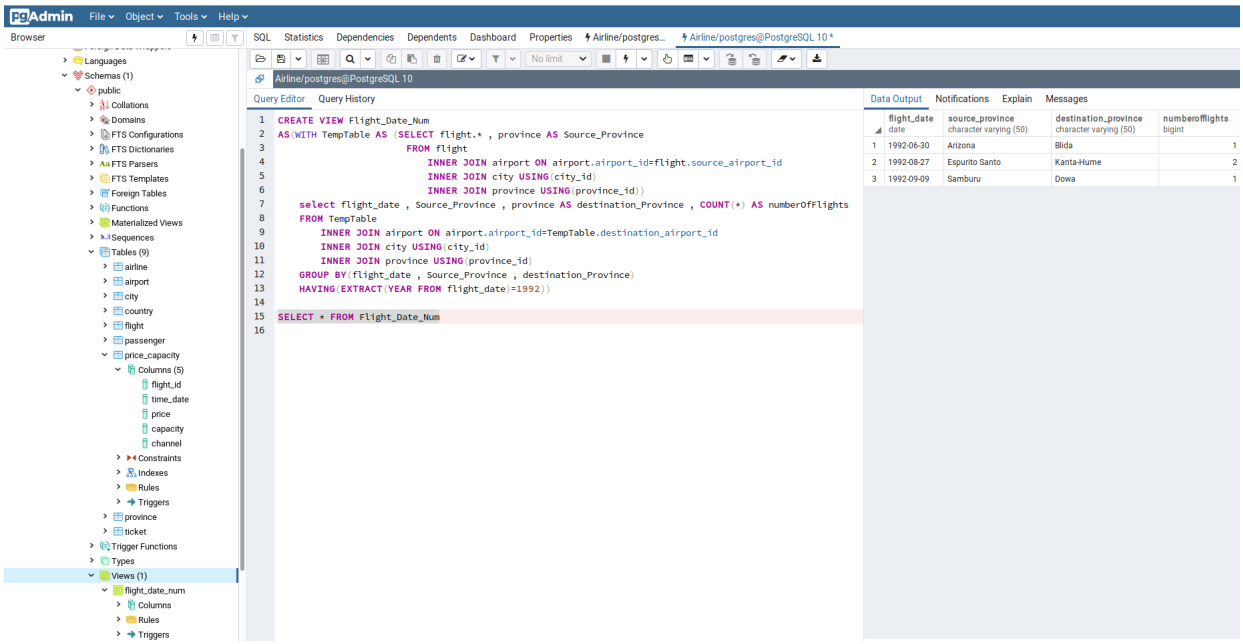
3-Functions can be called from Procedure whereas Procedures cannot be called from a Function.

4-The procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.

۴. سامانه نرم افزاری فروش بلیط هواپیما:

A) اسکریپت ها در فایل قرار گرفته است

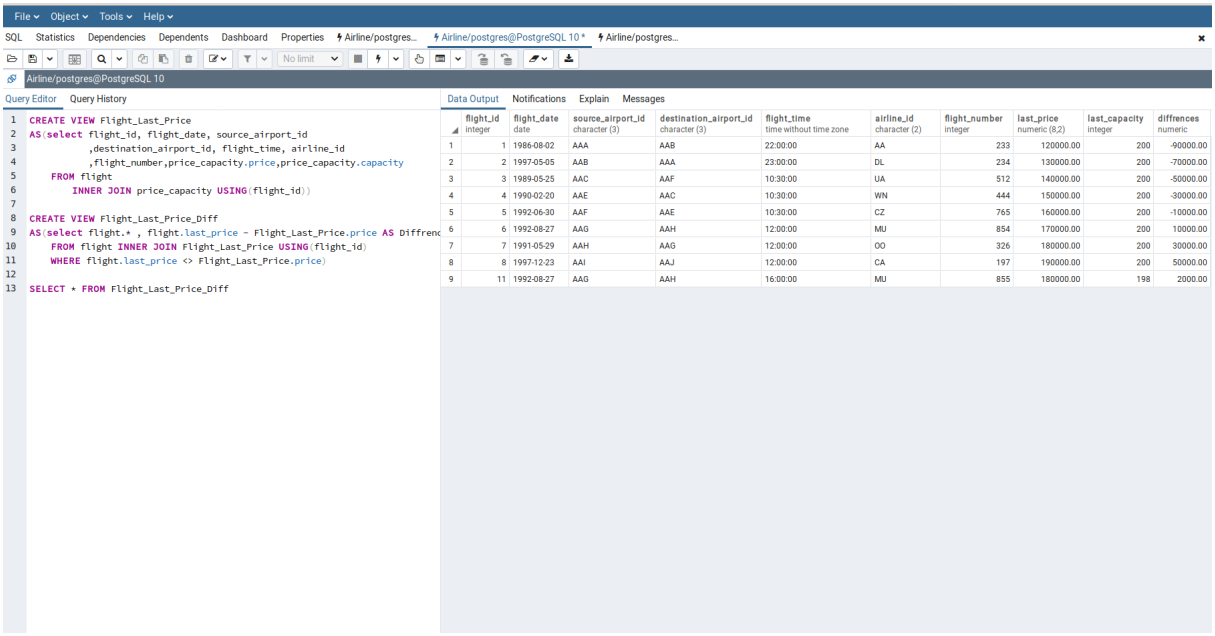
(B



این ویو نباید قابل آپدیت شدن باشد زیرا :

چون در این صورت برای هر آپدیت جدول ظرفیت قیمت اینم باید تریگر براش بخوره و اینم آپدیت شه و سربار خیلی زیادی پیش میاد

(C

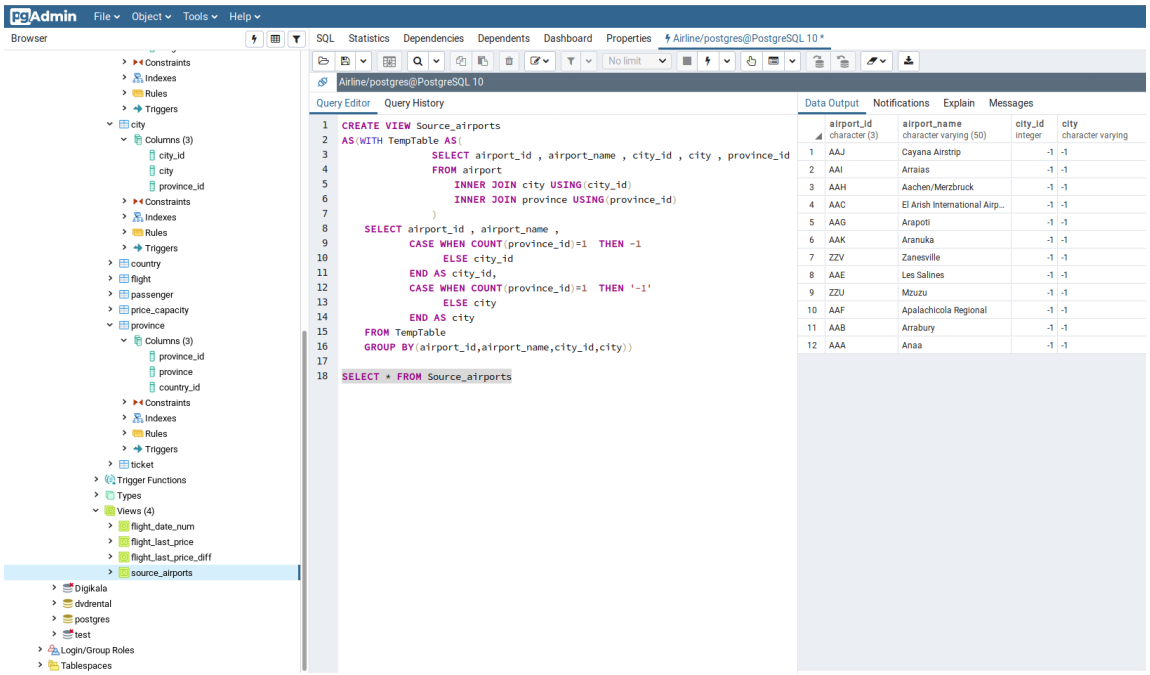


```
CREATE VIEW Flight_Last_Price
AS(select flight_id, flight_date, source_airport_id
          ,destination_airport_id, flight_time, airline_id
          ,flight_number,price_capacity.price,price_capacity.capacity
    FROM flight
        INNER JOIN price_capacity USING(flight_id))

CREATE VIEW Flight_Last_Price_Diff
AS(select flight.* , flight.last_price - Flight_Last_Price.price AS Diffrenc
    FROM flight INNER JOIN Flight_Last_Price USING(flight_id)
    WHERE flight.last_price <> Flight_Last_Price.price)

SELECT * FROM Flight_Last_Price_Diff
```

flight_id integer	flight_date date	source_airport_id character (3)	destination_airport_id character (3)	flight_time time without time zone	airline_id character (2)	flight_number integer	last_price numeric (8,2)	last_capacity integer	diffrences numeric
1	1986-08-02	AAA	AAB	22:00:00	AA	233	120000.00	200	-90000.00
2	1997-05-05	AAB	AAA	23:00:00	DL	234	130000.00	200	-70000.00
3	1989-05-25	AAC	AAF	10:30:00	UA	512	140000.00	200	-50000.00
4	1990-02-20	AAE	AAC	10:30:00	WN	444	150000.00	200	-30000.00
5	1992-06-30	AAF	AAE	10:30:00	CZ	765	160000.00	200	-10000.00
6	1992-08-27	AAG	AAH	12:00:00	MU	854	170000.00	200	10000.00
7	1991-05-29	AAH	AAG	12:00:00	OO	326	180000.00	200	30000.00
8	1997-12-23	AAI	AAJ	12:00:00	CA	197	190000.00	200	50000.00
11	1992-08-27	AAG	AAH	16:00:00	MU	855	180000.00	198	2000.00

چون شاید بعدا ازش خواسته میشه که بگه کی این تغییرات رو انجام داده

(D



```
CREATE VIEW Source_airports
AS(WITH TempTable AS(
                SELECT airport_id , airport_name , city_id , city , province_id
                FROM airport
                    INNER JOIN city USING(city_id)
                    INNER JOIN province USING(province_id)
                )
    SELECT airport_id , airport_name ,
            CASE WHEN COUNT(province_id)=1  THEN -1
                 ELSE city_id
            END AS city_id,
            CASE WHEN COUNT(province_id)=1  THEN '-1'
                 ELSE city
            END AS city
    FROM TempTable
    GROUP BY(airport_id,airport_name,city_id,city))

SELECT * FROM Source_airports
```

airport_id character (3)	airport_name character varying (50)	city_id integer	city character varying
AAJ	Cayana Airstrip	-1	-1
AAI	Arraias	-1	-1
AAH	Aachen/Merzbruck	-1	-1
AAC	El Arish International Airp...	-1	-1
AAG	Arapoti	-1	-1
AAK	Aranuka	-1	-1
ZZV	Zanesville	-1	-1
AAE	Les Salines	-1	-1
ZZU	Mzuzu	-1	-1
AAF	Apalachicola Regional	-1	-1
AAB	Arrabury	-1	-1
AAA	Anaa	-1	-1

توجه شود که در این حالت که دیتایی که در تیبل ها موجود بوده ، شرایطی که در یک استان بیش از یک فرودگاه باشد ، رخ نداده از همین جهت برای هیچ یک در خروجی ، نام شهر VALID نبوده

E) اسکریپت ها در فایل قرار گرفته است

F) اسکریپت ها در فایل قرار گرفته است

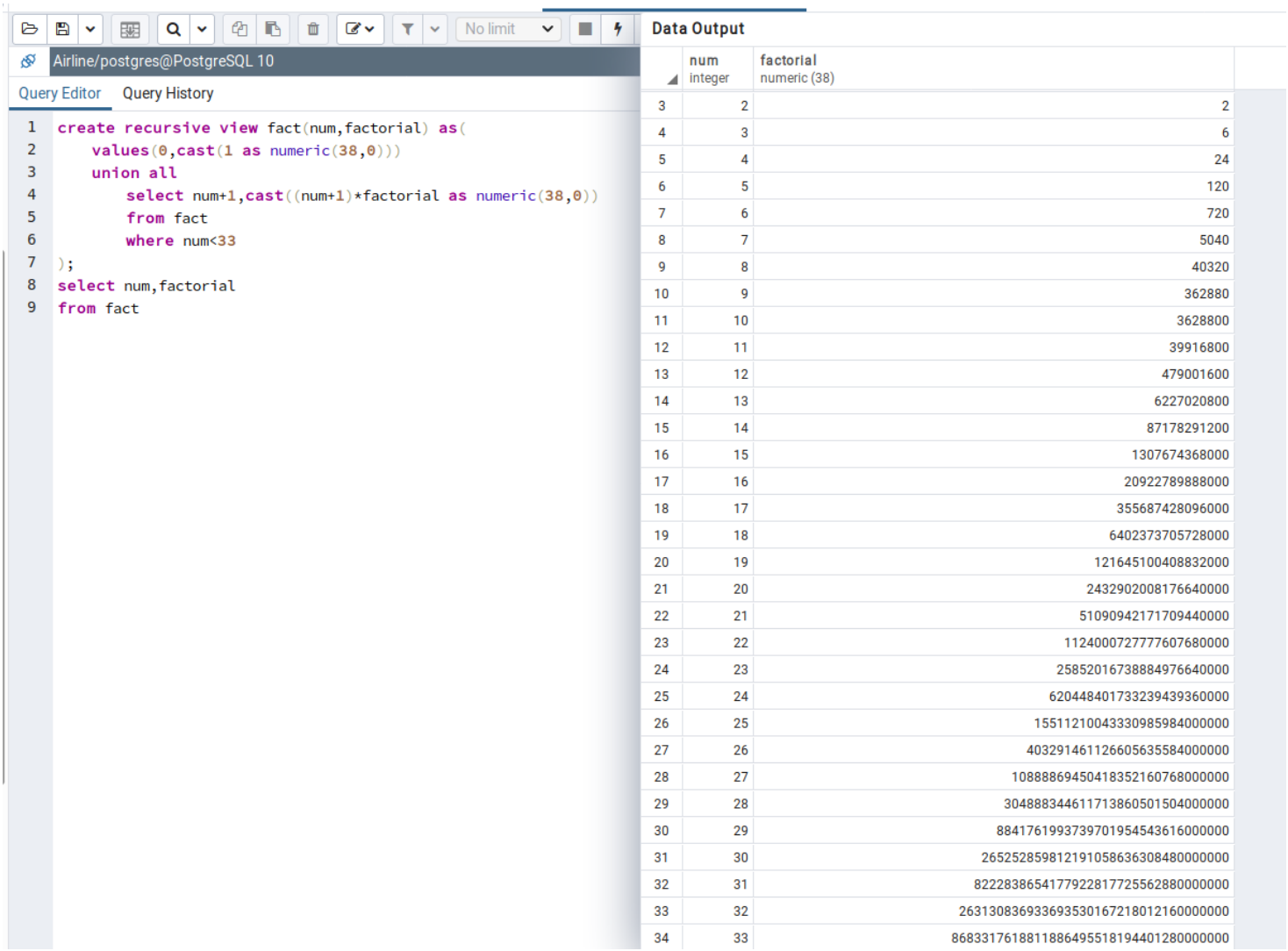
G) سوال ۳-الف چیه :(، صورت سوال خیلی نا واضحه ؟!!!!!!!!!!!!!!!!!!!!!!!!! اگر منظور سوال این باشه که یک تریگر رو این طور تعریف کنیم که BEFORE انجام عملیات مورد نظر باشد ، و در ابتدا ما لاگ اینکه دیتایی تغییر کرده را ثبت کنیم و در ادامه ی بدنه ی تریگر ، دیتاهای جدید را در تیبل مورد نظر اعمال کنیم

H) اسکریپت ها در فایل قرار گرفته است و همچنین

چون قرار نیست که این کاربر این کار رو انجام بده دلیلی نداره بهش دسترسی بدیم چون ممکنه خراب کاری کنه

۵. دستورات مربوط به recursive view را در Postgres مطالعه کنید و یک recursive view بنویسید که فاکتوریل اعداد کوچکتر از ۳۴ را در هر رکورد نمایش دهد. این مقدار برای !۳۴ چقدر است ؟چرا ؟

For bigger numbers , their factorials has more than 38 digits and SQL data types can only hold up to 38 .digits so it overflows a NUMERIC(38,0) field



Airline/postgres@PostgreSQL 10

Query Editor | Query History

```
1  create recursive view fact(num,factorial) as(
2      values(0,cast(1 as numeric(38,0)))
3      union all
4          select num+1,cast((num+1)*factorial as numeric(38,0))
5          from fact
6          where num<33
7  );
8  select num,factorial
9  from fact
```

Data Output

	num integer	factorial numeric (38)
3	2	2
4	3	6
5	4	24
6	5	120
7	6	720
8	7	5040
9	8	40320
10	9	362880
11	10	3628800
12	11	39916800
13	12	479001600
14	13	6227020800
15	14	87178291200
16	15	1307674368000
17	16	20922789888000
18	17	355687428096000
19	18	6402373705728000
20	19	121645100408832000
21	20	2432902008176640000
22	21	51090942171709440000
23	22	1124000727777607680000
24	23	25852016738884976640000
25	24	620448401733239439360000
26	25	15511210043330985984000000
27	26	403291461126605635584000000
28	27	10888869450418352160768000000
29	28	304888344611713860501504000000
30	29	8841761993739701954543616000000
31	30	265252859812191058636308480000000
32	31	8222838654177922817725562880000000
33	32	263130836933693530167218012160000000
34	33	8683317618811886495518194401280000000

۶. برای اینکه حجم پایگاه داده همیشه در بهینه ترین حالت ممکن قرار گیرد ، متخصصین پایگاه داده معمولاً روش هایی را مورد استفاده قرار میدهندکه یکی از آنها حذف داده های منقضی شده و نگهداری اطلاعات آنها در جداول دیگر است. در این مورد تحقیق کنید و دو روند انجام این کار را مختصراً شرح دهید.

1-using Event scheduler :

It's good when we want to expire records that have some condition with a period like every day/month/... .

```
CREATE EVENT archive_claims
        ON SCHEDULE EVERY 1 DAY //or the time we want
        COMMENT 'expired'
DO
BEGIN
        INSERT INTO old SELECT .... FROM now WHERE ....
        DELETE FROM now WHERE ....
END
```

2-using Expiry Date :

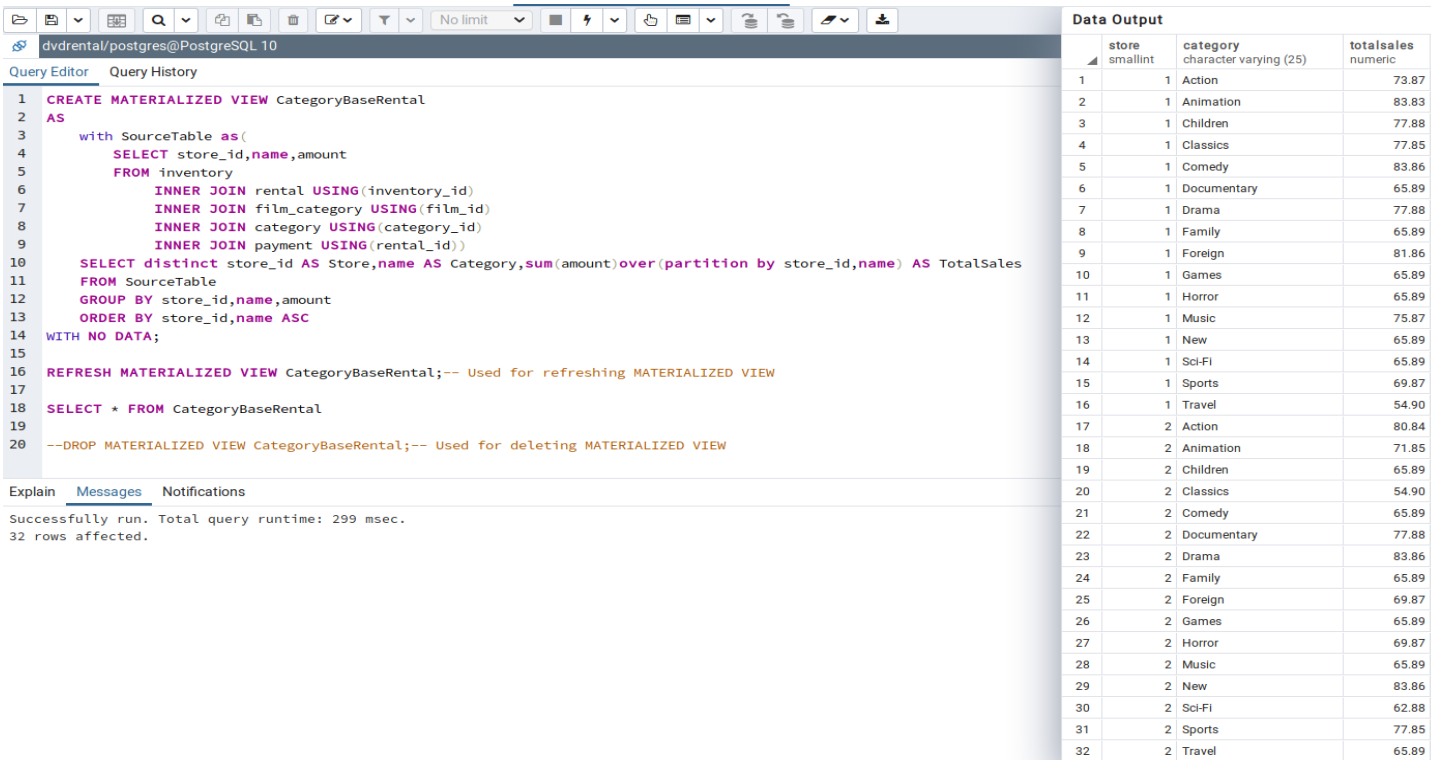
It's good for when sometime and once we want to expire records with some condition and older than some expiry date .

```
INSERT INTO old SELECT .... FROM now
WHERE .... and ExpiryDate>GETDATE()
DELETE FROM now WHERE .... and ExpiryDate<GETDATE()
```

بخش دوم

۷. اسکریپت بنویسید :

A) یک materialized view ایجاد کنید که میزان فروش هر فروشگاه را به تفکیک category های مختلف نشان دهد. دستورات لازم برای به روز رسانی و حذف این materialized view را بنویسید.



dvdrental/postgres@PostgreSQL 10

Query Editor | Query History

```
1  CREATE MATERIALIZED VIEW CategoryBaseRental
2  AS
3      with SourceTable as(
4          SELECT store_id,name,amount
5          FROM inventory
6               INNER JOIN rental USING(inventory_id)
7               INNER JOIN film_category USING(film_id)
8               INNER JOIN category USING(category_id)
9               INNER JOIN payment USING(rental_id))
10     SELECT distinct store_id AS Store,name AS Category,sum(amount)over(partition by store_id,name) AS TotalSales
11     FROM SourceTable
12     GROUP BY store_id,name,amount
13     ORDER BY store_id,name ASC
14 WITH NO DATA;
15
16 REFRESH MATERIALIZED VIEW CategoryBaseRental;-- Used for refreshing MATERIALIZED VIEW
17
18 SELECT * FROM CategoryBaseRental
19
20 --DROP MATERIALIZED VIEW CategoryBaseRental;-- Used for deleting MATERIALIZED VIEW
```

Explain | Messages | Notifications

```
Successfully run. Total query runtime: 299 msec.
32 rows affected.
```

Data Output

	store smallint	category character varying (25)	totalsales numeric
1	1	Action	73.87
2	1	Animation	83.83
3	1	Children	77.88
4	1	Classics	77.85
5	1	Comedy	83.86
6	1	Documentary	65.89
7	1	Drama	77.88
8	1	Family	65.89
9	1	Foreign	81.86
10	1	Games	65.89
11	1	Horror	65.89
12	1	Music	75.87
13	1	New	65.89
14	1	Sci-Fi	65.89
15	1	Sports	69.87
16	1	Travel	54.90
17	2	Action	80.84
18	2	Animation	71.85
19	2	Children	65.89
20	2	Classics	54.90
21	2	Comedy	65.89
22	2	Documentary	77.88
23	2	Drama	83.86
24	2	Family	65.89
25	2	Foreign	69.87
26	2	Games	65.89
27	2	Horror	69.87
28	2	Music	65.89
29	2	New	83.86
30	2	Sci-Fi	62.88
31	2	Sports	77.85
32	2	Travel	65.89

B) مدیریت فروشگاه تصمیم میگیرد با توجه به افزایش علاقه مندی به فیلم های انگلیسی زبان ، طول مدت اجاره ی فیلم را یک روز کم کند تا فیلم به دست همه مشتریان برسد. برای انجام این خواسته یک روش با استفاده از view ها و یک روش بدون استفاده از آنها قابل انجام است . نحوه انجام بدون استفاده از view ها را در پاسخنامه ی تشریحی وارد نموده و اسکریپت مربوط به ساخت یک view را در HW3.sql وارد کنید.

*** روش عدم استفاده از view**

```
CREATE TABLE NewFilmTable(
        film_id integer NOT NULL DEFAULT nextval('film_film_id_seq'::regclass),
    title character varying(255) NOT NULL,
    description text,
    release_year year,
    language_id smallint NOT NULL,
    rental_duration smallint NOT NULL DEFAULT 3,
    rental_rate numeric(4,2) NOT NULL DEFAULT 4.99,
    length smallint,
    replacement_cost numeric(5,2) NOT NULL DEFAULT 19.99,
    rating mpaa_rating DEFAULT 'G'::mpaa_rating,
    special_features text[],
    fulltext tsvector NOT NULL,
        name character(20) NOT NULL
);
INSERT INTO NewFilmTable
        SELECT film_id, title, description,
                release_year, language_id,
                rental_duration -1 as rental_duration,
                rental_rate,length, replacement_cost,
                rating,special_features, fulltext,name
        FROM film INNER JOIN language USING(language_id)
        WHERE name = 'English';
```

*** خروجی روش استفاده از view**

Messages

```
Successfully run. Total query runtime: 463 msec.
1000 rows affected.
```

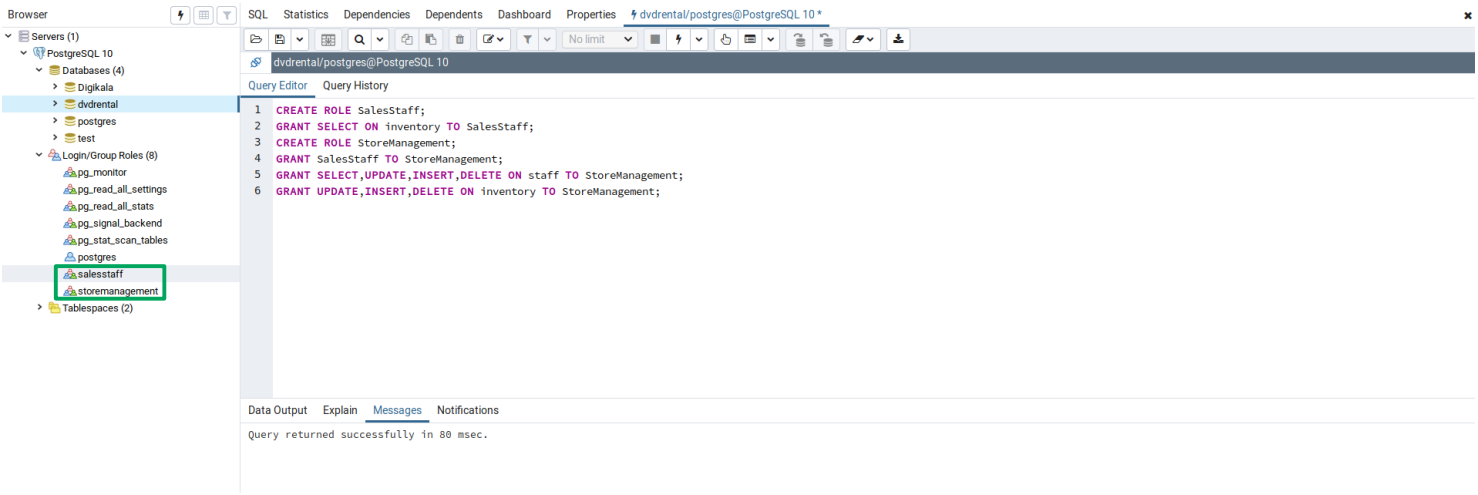
Query Editor | Query History

```
CREATE VIEW NewFilm
AS
    SELECT film_id, title, description,
            release_year, language_id,
            rental_duration -1 as rental_duration,
            rental_rate,length, replacement_cost,
            rating,special_features, fulltext,name
    FROM film INNER JOIN language USING(language_id)
    WHERE name='English';

SELECT * FROM NewFilm
```

Data Output | Explain | Notifications

	film_id integer	title character varying (255)	description text	release_year integer	language_id integer	rental_duration integer	rental_rate numeric (4,2)	length smallint	replacement_cost numeric (5,2)	rating mpaa_rating	special_features text[]	fulltext tsvector	name character (20)
1	133	Chamber Italian	A Fateful Refl...	2006	1	6	4.99	117	14.99	NC-17	{Trailers}	'chamber':...	English
2	384	Grosse Wonderful	A Epic Drama...	2006	1	4	4.99	49	19.99	R	{"Behind the Scenes"}	'australia':...	English
3	8	Airport Pollock	A Epic Tale of...	2006	1	5	4.99	54	15.99	R	{Trailers}	'airport':1 '...	English
4	98	Bright Encounters	A Fateful Yar...	2006	1	3	4.99	73	12.99	PG-13	{Trailers}	'boat':20 '...	English
5	1	Academy Dinosaur	A Epic Drama...	2006	1	5	0.99	86	20.99	PG	{"Deleted Scenes","B...	'academi':...	English
6	2	Ace Goldfinger	A Astounding...	2006	1	2	4.99	48	12.99	G	{Trailers,"Deleted Sc...	'ace':1 'ad...	English
7	3	Adaptation Holes	A Astounding...	2006	1	6	2.99	50	18.99	NC-17	{Trailers,"Deleted Sc...	'adapt':1 '...	English
8	4	Affair Prejudice	A Fanciful Do...	2006	1	4	2.99	117	26.99	G	{Commentaries,"Be...	'affair':1 'c...	English
9	5	African Egg	A Fast-Paced...	2006	1	5	2.99	130	22.99	G	{"Deleted Scenes"}	'african':1 '...	English
10	6	Agent Truman	A Intrepid Pa...	2006	1	2	2.99	169	17.99	PG	{"Deleted Scenes"}	'agent':1 '...	English
11	7	Airplane Sierra	A Touching S...	2006	1	5	4.99	62	28.99	PG-13	{Trailers,"Deleted Sc...	'airplan':1 '...	English
12	9	Alabama Devil	A Thoughtful ...	2006	1	2	2.99	114	21.99	PG-13	{Trailers,"Deleted Sc...	'administr'...	English
13	10	Aladdin Calendar	A Action-Pac...	2006	1	5	4.99	63	24.99	NC-17	{Trailers,"Deleted Sc...	'action':5 '...	English
14	11	Alamo Videotape	A Boring Epis...	2006	1	5	0.99	126	16.99	G	{Commentaries,"Be...	'alamo':1 '...	English
15	12	Alaska Phantom	A Fanciful Sa...	2006	1	5	0.99	136	22.99	PG	{Commentaries,"Del...	'alaska':1 '...	English
16	213	Date Speed	A Touching S...	2006	1	3	0.99	104	19.99	R	{Commentaries}	'compos':...	English
17	13	Ali Forever	A Action-Pac...	2006	1	3	4.99	150	21.99	PG	{"Deleted Scenes","B...	'action':5 '...	English
18	14	Alice Fantasia	A Emotional ...	2006	1	5	0.99	94	23.99	NC-17	{Trailers,"Deleted Sc...	'administr'...	English
19	15	Alien Center	A Brilliant Dra...	2006	1	4	2.99	46	10.99	NC-17	{Trailers,Comment...	'alien':1 'b...	English
20	16	Alley Evolution	A Fast-Paced...	2006	1	5	2.99	180	23.99	NC-17	{Trailers,Comment...	'alley':1 'a...	English
21	17	Alone Trip	A Fast-Paced...	2006	1	2	0.99	82	14.99	R	{Trailers,"Behind the...	'abandon':...	English
22	18	Alter Victory	A Thoughtful ...	2006	1	5	0.99	57	27.99	PG-13	{Trailers,"Behind the...	'agent':17 ...	English
23	19	Amadeus Holy	A Emotional ...	2006	1	5	0.99	113	20.99	PG	{Commentaries,"Del...	'amadeus'...	English
24	20	Amelie Hellfighters	A Boring Dra...	2006	1	3	4.99	79	23.99	R	{Commentaries,"Del...	'ameli':1 'b...	English
25	21	American Circus	A Insightful D...	2006	1	2	4.99	129	17.99	R	{Commentaries,"Be...	'administr'...	English
26	22	Amistad Midsummer	A Emotional ...	2006	1	5	2.99	85	10.99	G	{Commentaries,"Be...	'amistad':...	English
27	23	Anaconda Confessions	A Lacklustre...	2006	1	2	0.99	92	9.99	R	{Trailers,"Deleted Sc...	'anacond...	English
28	24	Analyze Hoosiers	A Thoughtful ...	2006	1	5	2.99	181	19.99	R	{Trailers,"Behind the...	'analyz':1 '...	English
29	25	Angels Life	A Thoughtful ...	2006	1	2	2.99	74	15.99	G	{Trailers}	'angel':1 'a...	English

C) دو نقش در پایگاه داده ی dvdrental اضافه کنید که یکی از آنها به عنوان کارمند بخش فروش فقط قادر به مشاهده ی اطلاعات موجودی فروشگاه باشد و دیگری به عنوان مسئول فروشگاه علاوه بر قابلیت های یک کارمند بخش فروش قادر به مشاهده ، حذف ،افزودن و به روزرسانی در اطلاعات کارمندان و موجودی فروشگاه باشد. اگر بخواهیم هر فرد در نقش های مذکورتنها قادر به مشاهده ی اطلاعات مربوط به فروشگاه محل کار خود باشد، به چه ترتیبی باید عمل کنیم؟ مختصرا در پاسخنامه تشریحی پاسخ دهید.



در ادامه ، برای اینکه هر فرد فقط به اطلاعات فروشگاه محل کار خودش دسترسی داشته باشد باید از **Row Security Policies** استفاده کنیم به صورت زیر :

```
ALTER TABLE staff ENABLE ROW LEVEL SECURITY;
ALTER TABLE inventory ENABLE ROW LEVEL SECURITY;

-- a row should be visible to/updatable by users whose store_id is equal to the row's store_id

CREATE POLICY fp_s ON staff FOR SELECT
  USING (store_id = (SELECT store_id FROM staff WHERE store_id = current_user_store_id));
CREATE POLICY fp_u ON staff FOR UPDATE
  USING (group_id = (SELECT store_id FROM staff WHERE store_id = current_user_store_id));
CREATE POLICY fp_s ON staff FOR INSERT
  USING (store_id = (SELECT store_id FROM staff WHERE store_id = current_user_store_id));
CREATE POLICY fp_s ON staff FOR DELETE
  USING (store_id = (SELECT store_id FROM staff WHERE store_id = current_user_store_id));

CREATE POLICY fp_s ON inventory FOR SELECT
  USING (store_id = (SELECT store_id FROM inventory WHERE store_id = current_user_store_id));
CREATE POLICY fp_u ON inventory FOR UPDATE
  USING (group_id = (SELECT store_id FROM inventory WHERE store_id = current_user_store_id));
CREATE POLICY fp_s ON inventory FOR INSERT
  USING (store_id = (SELECT store_id FROM inventory WHERE store_id = current_user_store_id));
CREATE POLICY fp_s ON inventory FOR DELETE
  USING (store_id = (SELECT store_id FROM inventory WHERE store_id = current_user_store_id));
```

۸. یکی از امکاناتی که postgres در اختیار توسعه دهندگان قرار میدهد دستورات EXPLAIN و EXPLAIN ANALYZE است . در مورد کاربرد این امکان و نحوه استفاده از آن تحقیق کنید و به سوالات زیر پاسخ دهید.

A) پرس و جوی زیر را عیناً در اجرا کنید و خروجی را تفسیر کنید.

```
EXPLAIN ANALYZE
SELECT * FROM payment WHERE staff_id = 2
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Seq Scan on payment (cost=0.00..290.45 rows=7304 width=26) (actual time=0.333..1.624 rows=7304 loops=1)
2	Filter: (staff_id = 2)
3	Rows Removed by Filter: 7292
4	Planning time: 12.315 ms
5	Execution time: 1.854 ms

یعنی Postgre انتظار داشته که برای پیدا کردن این مقادیر ۵.۰۴ از یک اوردر محاسباتی دلخواه باید هزینه شود .

rows : تعداد سطرهای خروجی تخمین زده شده توسط Postgre

width : تعداد بایت موجود در هر سطر

loop=1 : یعنی index scan فقط یک بار بررسی شده

filter : در حقیقت همان شرطی است که خودمان در قسمت WHERE کوئری گذاشتیم.

Rows removed by filter : هم به تعداد سطوری که از جدول پایه ای مان (payment) حذف شده اند تا سطور باقیمانده ، همان پاسخ درست کوئری بشود.

Planning time : زمان اجرای کوئری که تخمین زده شده توسط Postgre

Execution time : زمان اجرای واقعی کوئری

B) دستور زیر را یک بار عیناً و بار دیگر با استفاده از Nested Query بدون استفاده از عبارت WITH بازنویسی و سپس توسط EXPLAIN ANALYZE روی dvdrental اجرا کنید و خروجی ها را از نظر Query Plan و میانگین زمان اجرا بررسی کنید. چه نتیجه ای میگیرید؟

```
WITH italian_customers AS (
SELECT cu.customer_id cid, cu.first_name, Cu.last_name ,ci.city , co.country
FROM customer cu
INNER JOIN address USING (address_id)
INNER JOIN city ci USING (city_id)
INNER JOIN country co ON co.country_id = ci.country_id
WHERE co.country = 'Italy'
)
SELECT st.staff_id, ic.first_name, ic.last_name
FROM staff st
INNER JOIN rental rt USING (staff_id)
INNER JOIN italian_customers ic ON ic.cid = rt.customer_id;
```

*** خروجی برنامه با استفاده از WITH**

Query Editor | Query History

```
1  EXPLAIN ANALYSE
2  WITH italian_customers AS (
3  SELECT cu.customer_id cid, cu.first_name, Cu.last_name ,ci.city , co.country
4  FROM customer cu
5  INNER JOIN address USING (address_id)
6  INNER JOIN city ci USING (city_id)
7  INNER JOIN country co ON co.country_id = ci.country_id
8  WHERE co.country = 'Italy'
9  )
10 SELECT st.staff_id, ic.first_name, ic.last_name
11 FROM staff st
12 INNER JOIN rental rt USING (staff_id)
13 INNER JOIN italian_customers ic ON ic.cid = rt.customer_id;
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Nested Loop (cost=16.88..393.20 rows=134 width=220) (actual time=0.260..3.419 rows=189 loops=1)
2	Join Filter: (rt.staff_id = st.staff_id)
3	Rows Removed by Join Filter: 95
4	CTE italian_customers
5	-> Nested Loop (cost=4.87..16.71 rows=5 width=35) (actual time=0.077..0.163 rows=7 loops=1)
6	-> Nested Loop (cost=4.60..14.38 rows=6 width=22) (actual time=0.066..0.122 rows=7 loops=1)
7	-> Nested Loop (cost=4.32..12.09 rows=6 width=22) (actual time=0.053..0.078 rows=7 loops=1)
8	-> Seq Scan on country co (cost=0.00..2.36 rows=1 width=13) (actual time=0.022..0.038 rows=1 loops=1)
9	Filter: ((country)::text = 'Italy'::text)
10	Rows Removed by Filter: 108
11	-> Bitmap Heap Scan on city ci (cost=4.32..9.66 rows=6 width=15) (actual time=0.026..0.033 rows=7 loops=1)
12	Recheck Cond: (country_id = co.country_id)
13	Heap Blocks: exact=3
14	-> Bitmap Index Scan on idx_fk_country_id (cost=0.00..4.32 rows=6 width=0) (actual time=0.019..0.019 rows=7 loops=1)
15	Index Cond: (country_id = co.country_id)
16	-> Index Scan using idx_fk_city_id on address (cost=0.28..0.37 rows=1 width=6) (actual time=0.005..0.005 rows=1 loops=7)
17	Index Cond: (city_id = ci.city_id)
18	-> Index Scan using idx_fk_address_id on customer cu (cost=0.28..0.38 rows=1 width=19) (actual time=0.004..0.005 rows=1 loops=7)
19	Index Cond: (address_id = address.address_id)
20	-> Hash Join (cost=0.16..372.11 rows=134 width=218) (actual time=0.244..3.329 rows=189 loops=1)
21	Hash Cond: (rt.customer_id = ic.cid)
22	-> Seq Scan on rental rt (cost=0.00..310.44 rows=16044 width=4) (actual time=0.009..1.434 rows=16044 loops=1)
23	-> Hash (cost=0.10..0.10 rows=5 width=220) (actual time=0.190..0.190 rows=7 loops=1)
24	Buckets: 1024 Batches: 1 Memory Usage: 9kB
25	-> CTE Scan on italian_customers ic (cost=0.00..0.10 rows=5 width=220) (actual time=0.081..0.177 rows=7 loops=1)
26	-> Materialize (cost=0.00..1.03 rows=2 width=4) (actual time=0.000..0.000 rows=2 loops=189)
27	-> Seq Scan on staff st (cost=0.00..1.02 rows=2 width=4) (actual time=0.004..0.004 rows=2 loops=1)
28	Planning time: 1.557 ms
29	Execution time: 3.531 ms

* خروجی برنامه با استفاده از Nested Query

Query Editor | Query History

```
1  EXPLAIN ANALYSE
2  SELECT st.staff_id, ic.first_name, ic.last_name
3  FROM staff st
4  INNER JOIN rental rt USING (staff_id)
5  INNER JOIN (SELECT cu.customer_id cid,
6              cu.first_name, Cu.last_name
7              ,ci.city , co.country
8              FROM customer cu
9              INNER JOIN address USING (address_id)
10             INNER JOIN city ci USING (city_id)
11             INNER JOIN country co ON co.country_id = ci.country_id
12             WHERE co.country = 'Italy') ic ON ic.cid = rt.customer_id;
```

Data Output | Explain | Notifications | Messages

	QUERY PLAN text
1	Nested Loop (cost=16.78..393.42 rows=147 width=17) (actual time=0.310..7.935 rows=189 loops=1)
2	Join Filter: (rt.staff_id = st.staff_id)
3	Rows Removed by Join Filter: 95
4	-> Hash Join (cost=16.78..388.72 rows=147 width=15) (actual time=0.295..7.694 rows=189 loops=1)
5	Hash Cond: (rt.customer_id = cu.customer_id)
6	-> Seq Scan on rental rt (cost=0.00..310.44 rows=16044 width=4) (actual time=0.012..3.128 rows=16044 loops=1)
7	-> Hash (cost=16.71..16.71 rows=5 width=17) (actual time=0.216..0.216 rows=7 loops=1)
8	Buckets: 1024 Batches: 1 Memory Usage: 9kB
9	-> Nested Loop (cost=4.87..16.71 rows=5 width=17) (actual time=0.096..0.199 rows=7 loops=1)
10	-> Nested Loop (cost=4.60..14.38 rows=6 width=4) (actual time=0.083..0.149 rows=7 loops=1)
11	-> Nested Loop (cost=4.32..12.09 rows=6 width=4) (actual time=0.067..0.098 rows=7 loops=1)
12	-> Seq Scan on country co (cost=0.00..2.36 rows=1 width=4) (actual time=0.027..0.048 rows=1 loops=1)
13	Filter: ((country)::text = 'Italy'::text)
14	Rows Removed by Filter: 108
15	-> Bitmap Heap Scan on city ci (cost=4.32..9.66 rows=6 width=6) (actual time=0.035..0.043 rows=7 loops=1)
16	Recheck Cond: (country_id = co.country_id)
17	Heap Blocks: exact=3
18	-> Bitmap Index Scan on idx_fk_country_id (cost=0.00..4.32 rows=6 width=0) (actual time=0.027..0.027 rows=7 loops=1)
19	Index Cond: (country_id = co.country_id)
20	-> Index Scan using idx_fk_city_id on address (cost=0.28..0.37 rows=1 width=6) (actual time=0.006..0.006 rows=1 loops=7)
21	Index Cond: (city_id = ci.city_id)
22	-> Index Scan using idx_fk_address_id on customer cu (cost=0.28..0.38 rows=1 width=19) (actual time=0.005..0.006 rows=1 loops=7)
23	Index Cond: (address_id = address.address_id)
24	-> Materialize (cost=0.00..1.03 rows=2 width=4) (actual time=0.000..0.000 rows=2 loops=189)
25	-> Seq Scan on staff st (cost=0.00..1.02 rows=2 width=4) (actual time=0.005..0.006 rows=2 loops=1)
26	Planning time: 2.217 ms
27	Execution time: 8.086 ms

نتیجه اینکه :

Execution time برای زمانی که کوئری را با WITH نوشته بودیم کمتر از وقتی شد که از Nested Query استفاده کرده بودیم. در نهایت ، استفاده از WITH به جای Nested Query توصیه میشود چراکه بهینه تر است هم از لحاظ سرعت اجرا هم از لحاظ خوانا بودن کدها :)

C) با توجه به Query Plan درخروجی سوال (۸-ب) عملگر Hash به چه منظور توسط DBMS مورد استفاده قرار میگیرد؟

In DBMS, hashing is a technique to directly search the location of desired data on the disk without using index structure. Data is stored in the form of data blocks whose address is generated by applying a hash function in the memory location where these records are stored known as a data block or data bucket.