به نام خدا

پاسخنامه تکلیف چهارم درس پایگاه داده ها ۱

ترم اول ۱۳۹۹ — ۱۴۰۰

---

1 روی پایگاه AW و با استفاده از دستورات authorization و view و trigger و سایر امکانات لازم:

(a.1 به مشتریان عادی فروشگاه (normal_customer)این امکان را بدهید که مشخصات سفارشهای خود را که تاریخ آنها مربوط به امروز یا دیروز باشد و همچنین جزئیات اقلام مربوط به همین سفارشها را بتوانند تغییر دهند. امکان ایجاد شده را در هر دوحالت مجاز و تلاش برای دسترسی غیرمجاز تست کنید.

پاسخ :

```
CREATE OR REPLACE VIEW Header_Customer29825 AS
      SELECT *
      FROM Sales.SalesOrderHeader
      WHERE CustomerID = '29825'
      AND date_trunc('day', orderdate) = date_trunc('day', NOW())
      OR date_trunc('day', orderdate) = date_trunc('day', NOW() - INTERVAL '1 DAY');

CREATE OR REPLACE VIEW Detail_Customer30100 AS
      SELECT Sales.SalesOrderDetail.*
      FROM Header_Customer29825 INNER JOIN Sales.SalesOrderDetail USING(SalesOrderID);

CREATE USER Customer29825;
GRANT UPDATE ON Header_Customer29825, Detail_Customer29825 TO Customer29825;

SET SESSION AUTHORIZATION Customer29825;

UPDATE Sales.SalesOrderHeader
SET CustomerID = '14239'
WHERE CustomerID = '29825';

UPDATE Header_Customer29825
SET OrderDate = '2020-11-02';
```

```
17 UPDATE Sales.SalesOrderHeader
18 SET CustomerID = '14239'
19 WHERE CustomerID = '29825';
20
21 UPDATE Header_Customer29825
22 SET OrderDate = '2020-11-02';
Data Output  Explain  Messages  Notifications
ERROR:  permission denied for schema sales
LINE 1: UPDATE Sales.SalesOrderHeader
                ^
SQL state: 42501
Character: 8
```

دسترسی غیرمجاز

```
17 UPDATE Sales.SalesOrderHeader
18 SET CustomerID = '14239'
19 WHERE CustomerID = '29825';
20
21 UPDATE Header_Customer29825
22 SET OrderDate = '2020-11-02';
Data Output  Explain  Messages  Notifications
UPDATE 0

Query returned successfully in 270 msec.
```

دسترسی مجاز

b.1) برای بخش انبار کالا (product inventory) یک stored procedure بنویسید که طی یک تراکنش، تمام یا بخشی از موجودی یک کالا را از یک انبار (مبدا) به انبار دیگر (مقصد) منتقل کند. برای تعیین محل قرار دادن کالا در انبار (shelf, bin) در صورتی که از همین کالا قبلا موجود داریم، از همان shelf قبلی و با افزایش شماره bin استفاده شود. در صورتی که این نوع کالا را از قبل در انبار نداریم، از شماره shelf, bin پیش‌فرض خاص این انبار استفاده شود. شماره‌های پیش فرض برای انبارها، باید از قبل در جدول جدیدی به نام InventoryDefaults بطور دستی وارد شده باشد و در متن دستور استفاده شود.

پاسخ :

*** کدها در صفحه بعد آمده است .**

قبل از اجرای پراسیجر:

```
54 SELECT * FROM production.productinventory ORDER BY productid;
55
56 CALL transfer(1, 1, 6, 10);
57
58 SELECT * FROM production.productinventory ORDER BY productid;
```

Data Output | Explain | Messages | Notifications

	productid [PK] integer	locationid [PK] smallint	shelf character varying (10)	bin smallint	quantity smallint	rowguid uuid	modifieddate timestamp without time zone
1	1	1	A	3	410	47a24246-6...	2014-08-08 00:00:00
2	1	6	B	6	352	d4544d7d-c...	2014-08-08 00:00:00
3	1	50	A	5	353	bff7dc60-96...	2014-08-08 00:00:00
4	2	1	A	2	427	f407c07a-ca...	2014-08-08 00:00:00
5	2	6	B	1	318	ca1ff2f4-48f...	2014-08-08 00:00:00
6	2	50	A	6	364	d38cfbee-63...	2014-08-08 00:00:00
7	3	1	A	7	585	e18a519b-f...	2008-03-31 00:00:00
8	3	6	B	9	443	3c860c96-1...	2008-03-31 00:00:00
9	3	50	A	10	324	1339e5e3-1...	2008-03-31 00:00:00
10	4	1	A	6	512	6beaf0a0-9...	2014-08-08 00:00:00
11	4	6	B	10	422	2c82427a-6...	2014-08-08 00:00:00
12	4	50	A	11	388	fd912e69-ef...	2014-08-08 00:00:00
13	316	5	A	11	532	1ee3dbd3-2...	2008-03-31 00:00:00
14	316	10	B	1	388	cb2a24d7-9...	2008-03-31 00:00:00

بعد از اجرای پراسیجر:

```
54 SELECT * FROM production.productinventory ORDER BY productid;
55
56 CALL transfer(1, 1, 6, 10);
57
58 SELECT * FROM production.productinventory ORDER BY productid;
```

Data Output | Explain | Messages | Notifications

	productid [PK] integer	locationid [PK] smallint	shelf character varying (10)	bin smallint	quantity smallint	rowguid uuid	modifieddate timestamp without time zone
1	1	1	A	3	400	47a24246-6...	2014-08-08 00:00:00
2	1	6	B	7	362	d4544d7d-c...	2014-08-08 00:00:00
3	1	50	A	5	353	bff7dc60-96...	2014-08-08 00:00:00
4	2	1	A	2	427	f407c07a-ca...	2014-08-08 00:00:00
5	2	6	B	1	318	ca1ff2f4-48f...	2014-08-08 00:00:00
6	2	50	A	6	364	d38cfbee-63...	2014-08-08 00:00:00
7	3	1	A	7	585	e18a519b-f...	2008-03-31 00:00:00
8	3	6	B	9	443	3c860c96-1...	2008-03-31 00:00:00
9	3	50	A	10	324	1339e5e3-1...	2008-03-31 00:00:00
10	4	1	A	6	512	6beaf0a0-9...	2014-08-08 00:00:00
11	4	6	B	10	422	2c82427a-6...	2014-08-08 00:00:00
12	4	50	A	11	388	fd912e69-ef...	2014-08-08 00:00:00
13	316	5	A	11	532	1ee3dbd3-2...	2008-03-31 00:00:00
14	316	10	B	1	388	cb2a24d7-9...	2008-03-31 00:00:00

```sql
CREATE TABLE InventoryDefaults(
    productId  int,
    locationId smallint,
    shelf      varchar(10),
    bin        smallint,
    primary key (productId, locationId),
    foreign key (productId) references Production.product (productId),
    foreign key (locationId) references Production.location (locationid)
);

INSERT INTO InventoryDefaults
      SELECT productid,
             locationid,
            'A' AS shelf,
            (productid + locationid) % 60 AS bin
      FROM production.product, production.location;

CREATE OR REPLACE PROCEDURE transfer(
    prodID INT,
    srcLoc INT,
    desLoc INT,
    amount INT)
LANGUAGE PLPGSQL
AS
$$
BEGIN
    UPDATE production.productinventory
    SET quantity = quantity - amount
    WHERE locationID = srcLoc and productID = prodID;

      DELETE FROM production.productinventory
      WHERE locationID = srcLoc and productID = prodID and quantity = 0;

    IF EXISTS(SELECT *
              FROM production.productinventory
              WHERE locationID = desLoc AND productID = prodID) THEN
        UPDATE production.productinventory
        SET quantity = quantity + amount, bin = bin + 1
        WHERE locationID = desLoc AND productID = prodID;
      ELSE
      INSERT INTO production.productinventory(productid,locationid,shelf,
bin,quantity)
      VALUES (prodID,
              desLoc,
              (SELECT shelf FROM InventoryDefaults WHERE productid = prodID
AND locationid = desLoc),
              (SELECT bin FROM InventoryDefaults WHERE productid = prodID
AND locationid = desLoc),
              amount);
      END IF;
    COMMIT;
END;
$$;

SELECT * FROM production.productinventory ORDER BY productid;

CALL transfer(1, 1, 6, 10);

SELECT * FROM production.productinventory ORDER BY productid;
```

2 روی پایگاه دانشگاه، قصد داریم کنترل بیشتری روی ثبت‌نام انجام دهیم.

a.2) دستور ایجاد جدولی با نام RegistrationLog با ساختار زیر را بنویسید، طوری که شناسه دانشجو کلید خارجی به جدول Student باشد، کلید اصلی مناسب برای این جدول تعریف شود، و فیلد Status نیز فقط یکی از مقادیر Normal, CheckOver, CheckUnder, NotReg را بتواند بگیرد.

StudentID,

Semester,

Year,

Status,

OverAttempts

پاسخ :

```
CREATE TABLE registrationLog(
      studentid VARCHAR(5),
      semester VARCHAR(6),
      year NUMERIC(4),
      status VARCHAR(10),
      overattempts INT,
      PRIMARY KEY(studentid,semester,year),
      FOREIGN KEY (studentid) REFERENCES student(id),
      CHECK (semester IN ('Fall','Winter','Spring','Summer')),
      CHECK (status IN ('Normal','NotReg','CheckUnder','CheckOver'))
);
```

b.2) یک پراسیجر بنویسید که ثبت‌نام یک درس را برای یک دانشجو انجام دهد. این کار باید در قالب یک تراکنش باشد که اگر تعداد واحد دانشجو درست قبل از insert از 20 بالاتر بود اجازه درج ندهد، و ضمنا فیلد OverAttempts برای این دانشجو را در جدول فوق یکی افزایش دهد. همچنین در صورت ثبت موفق درس، فیلد Status در جدول فوق به Normal تغییر یابد (اگر رکوردی برای این دانشجو در جدول فوق از قبل موجود نبود، insert انجام شود.)

پاسخ :

```
CREATE OR REPLACE PROCEDURE take_pro(
      sid VARCHAR(5),
      course VARCHAR(8),
      sec VARCHAR(8),
      sem VARCHAR(6),
      y NUMERIC(4)
)
LANGUAGE PLPGSQL
AS
$$
BEGIN
    IF EXISTS(SELECT *
              FROM registrationlog
              WHERE studentid = sid AND semester = sem AND year = y) THEN

      UPDATE registrationlog
      SET overattempts = overattempts + 1
      WHERE studentid = sid AND semester = sem AND year = y;
      ELSE
                 INSERT INTO registrationlog VALUES (sid,sem,y,'NotReg',1);
      END IF;

      IF (SELECT tot_cred FROM student WHERE id = sid) < 20 THEN

      INSERT INTO takes(id,course_id,sec_id,semester,year,grade)
      VALUES (sid,course,sec,sem,y,null);

      UPDATE registrationlog
      SET status = 'Normal'
      WHERE studentid = sid AND semester = sem AND year = y;
      END IF;
    COMMIT;
END;
$$;
```

```
16  CALL take_pro('24746', '313', '1', 'Fall', 2010);
17
18  SELECT * FROM registrationLog;
```

Data Output | Explain | Messages | Notifications

	studentid [PK] character varying (5)	semester [PK] character varying (6)	year [PK] numeric (4)	status character varying (10)	overattempts integer
1	24746	Fall	2010	Normal	1

c.2) یک Trigger بنویسید که در صورتی که تعداد واحد ثبت‌نام شده دانشجو در ترم جاری به زیر 12 واحد رسید، فیلد جدول فوق به CheckUnder، و اگر به بالای 18 رسید، به CheckOver تغییر یابد.

پاسخ :

```
CREATE OR REPLACE FUNCTION LOG_REGISTER()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
      UPDATE registrationlog
      SET status =
      CASE
            WHEN (SELECT SUM(credits) from takes JOIN course USING(course_id)
                    WHERE NEW.id = takes.id AND NEW.semester =  takes.semester
                    AND NEW.year =  takes.year) < 12 THEN 'CheckUnder'
            WHEN (SELECT SUM(credits) from takes JOIN course USING(course_id)
                    WHERE NEW.id = takes.id and NEW.semester =  takes.semester
                    and NEW.year =  takes.year) > 18 THEN 'CheckOver'
       END
      WHERE registrationlog.studentid = NEW.ID
       AND registrationlog.semester = NEW.semester
       AND registrationlog.year = NEW.year;
      RETURN NEW;
END;
$$;
CREATE TRIGGER REGISTER_CHANGES AFTER
INSERT OR UPDATE ON TAKES
FOR EACH ROW EXECUTE PROCEDURE LOG_REGISTER();

INSERT INTO TAKES
VALUES ('24746', '105', '2', 'Fall', 2002);

SELECT * FROM registrationlog;
```

```
27  INSERT INTO TAKES
28  VALUES ('24746', '105', '2', 'Fall', 2002);
29
30  SELECT * FROM registrationlog;
```

Data Output | Explain | Messages | Notifications

	studentid [PK] character varying (5)	semester [PK] character varying (6)	year [PK] numeric (4)	status character varying (10)	overattempts integer
1	24746	Fall	2010	Normal	1

3 **با استفاده از پایگاه** dvdrental **دستورات زیر را بنویسید:**

a.3) **پراسیجری بنویسید که تمامی ژانرهایی را که میانگین طول فیلم‌هایشان از میانگین طول فیلمهای سه ژانری که بیشترین میانگین امتیاز فیلم ها را دارند، بیشتر است، از جدول** category_rating **حذف کند. (ابتدا باید جدولی با نام** category_rating **ایجاد کرده و در آن لیست ژانرها را به همراه میانگین امتیاز مشتریان به فیلم های هر یک از ژانرها و طول بلندترین فیلم آن ژانر ذخیره کنید و سپس پراسیجر خواسته شده را با استفاده از آن بنویسید.)**

پاسخ :

*** کدها در صفحه بعد آمده است.**

قبل از اجرای پراسیجر:

```
SELECT * FROM category_rating;
CALL category_length_remove_proc();
SELECT * FROM category_rating;
```

Data Output | Explain | Messages | Notifications

	category_id [PK] smallint	category_name character varying (25)	avg_rate [PK] numeric (4,2)	max_length [PK] smallint
1	4	Classics	2.74	184
2	14	Sci-Fi	3.22	185
3	3	Children	2.89	178
4	10	Games	3.25	185
5	7	Drama	3.02	181
6	13	New	3.12	183
7	9	Foreign	3.10	184
8	1	Action	2.65	185
9	5	Comedy	3.16	185
10	2	Animation	2.81	185
11	16	Travel	3.24	185
12	15	Sports	3.13	184
13	6	Documentary	2.67	183
14	12	Music	2.95	185

بعد از اجرای پراسیجر :

```
SELECT * FROM category_rating;
CALL category_length_remove_proc();
SELECT * FROM category_rating;
```

Data Output | Explain | Messages | Notifications

	category_id [PK] smallint	category_name character varying (25)	avg_rate [PK] numeric (4,2)	max_length [PK] smallint
1	4	Classics	2.74	184
2	14	Sci-Fi	3.22	185
3	3	Children	2.89	178
4	13	New	3.12	183
5	1	Action	2.65	185
6	2	Animation	2.81	185
7	16	Travel	3.24	185
8	6	Documentary	2.67	183
9	12	Music	2.95	185
10	8	Family	2.76	184
11	11	Horror	3.03	181

```sql
CREATE TABLE category_rating(
      category_id SMALLINT,
      category_name VARCHAR(25),
      avg_rate NUMERIC(4,2),
      max_length SMALLINT,
      PRIMARY KEY (category_id, avg_rate, max_length),
      FOREIGN KEY (category_id) REFERENCES category(category_id)
);

INSERT INTO category_rating
      SELECT category.category_id, name, AVG(rental_rate), MAX(length)
      FROM film_category, film, category
      WHERE film_category.film_id = film.film_id
        AND film_category.category_id = category.category_id
      GROUP BY category.category_id, name;

CREATE OR REPLACE PROCEDURE category_length_remove_proc()
LANGUAGE PLPGSQL
AS
$$
BEGIN
      WITH category_avg_rate(category_id, avg_rate) AS(
            SELECT category_id, AVG(rental_rate)
            FROM film_category, film
            WHERE film_category.film_id = film.film_id
            GROUP BY category_id),
      three_max_rate(category_id, avg_rate) AS(
            SELECT category_id, avg_rate
            FROM category_avg_rate
            ORDER BY avg_rate DESC LIMIT 3),
      category_avg_length(category_id, avg_length) AS(
            SELECT category_id, AVG(length)
            FROM film_category, film
            WHERE film_category.film_id = film.film_id
            GROUP BY category_id),
      three_max_rate_length(category_id, avg_length) AS(
            SELECT category_avg_length.category_id,
                   category_avg_length.avg_length
            FROM category_avg_length, three_max_rate
            WHERE category_avg_length.category_id =
                   three_max_rate.category_id),
      final_three_avg_length (final_avg_length) AS(
            SELECT AVG(avg_length)
            FROM category_avg_length)

      DELETE FROM category_rating
      WHERE category_id IN (SELECT category_id
                            FROM category_avg_length
                            WHERE avg_length > (SELECT *
                                                FROM final_three_avg_length));
END;
$$;

SELECT * FROM category_rating;
CALL category_length_remove_proc();
SELECT * FROM category_rating;
```

b.3) تابعی بنویسید که با دریافت یک تاریخ (روز و ماه و سال) و نام فیلم، وضعیت فیلم را در آن تاریخ اعلام کند (تمام نسخه‌های آن در اجاره بوده یا همچنان تعدادی از آن در فروشگاه موجود بوده است)

پاسخ :

```
CREATE OR REPLACE FUNCTION getstatus(date_ date, filmname_ character varying)
RETURNS character varying
LANGUAGE PLPGSQL
AS
$$
BEGIN
     IF EXISTS(SELECT TITLE
             FROM INVENTORY INNER JOIN FILM USING(FILM_ID)
             INNER JOIN RENTAL USING(INVENTORY_ID)
             WHERE LOWER(TITLE) = LOWER(FilmName_)
                      AND Date_ BETWEEN RENTAL_DATE AND RETURN_DATE) THEN
     RETURN 'Borrowed';
     END IF;
     RETURN 'Available';
END;
$$;

SELECT getstatus('2005-04-01','Academy Dinosaur');
```

```
18  SELECT getstatus('2005-04-01','Academy Dinosaur');
```

Data Output | Explain | Messages | Notifications

	getstatus character varying
1	Available

c.3) **تریگری بنویسید تا به محض اینکه مشتری فیلمی را با بیش از ۷ واحد پولی اجاره کرد، به او یک فیلم که در همان فروشگاه موجود است، به مدت یک روز به صورت رایگان اجـــاره داده شود.**

پاسخ :

```
CREATE OR REPLACE FUNCTION freeFilmDonationFunc()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
      IF(NEW.amount > 7) THEN
      INSERT INTO rental
      VALUES (DEFAULT, NOW()::timestamp,
                 (SELECT inventory_id
                  FROM inventory
                  WHERE inventory.store_id = (SELECT inventory.store_id
                                              FROM inventory JOIN rental
                                              USING(inventory_id)
                                              WHERE NEW.rental_id =
                                                        rental.rental_id)
                      AND inventory_id NOT IN (SELECT inventory_id
                                               FROM rental
                                               WHERE rental_date IS NULL OR
date_part('minute',age(NOW()::timestamp, return_date)) < 0)
                 FETCH FIRST 1 ROWS ONLY),
                 NEW.customer_id,
                 NOW()::timestamp + interval '1 day',
                 NEW.staff_id, DEFAULT);

      INSERT INTO payment
            SELECT NEW.payment_id + 1, NEW.customer_id, NEW.staff_id,
                   NEW.rental_id + 1, 0, NOW()::TIMESTAMP;
      END IF;
      RETURN NEW;
END;
$$;

CREATE TRIGGER freeFilmDonation
AFTER INSERT
ON payment
FOR EACH ROW
EXECUTE PROCEDURE freeFilmDonationFunc();

INSERT INTO payment VALUES(DEFAULT, 459, 1, 2, 7.6, NOW()::TIMESTAMP);

SELECT * FROM rental ORDER BY rental_id DESC;
SELECT * FROM payment ORDER BY payment_id DESC;
```

```
14
15  INSERT INTO payment VALUES(DEFAULT, 459, 1, 2, 7.6, NOW()::TIMESTAMP);
16
17  SELECT * FROM rental ORDER BY rental_id DESC;
18  SELECT * FROM payment ORDER BY payment_id DESC;
```

Data Output | Explain | Messages | Notifications

	rental_id [PK] integer	rental_date timestamp without time zone	inventory_id integer	customer_id smallint	return_date timestamp without time zone	staff_id smallint	last_update timestamp without time zone
1	16054	2020-11-19 16:01:40.663114	5	459	2020-11-20 16:01:40.663114	1	2020-11-19 16:01:40.663114
2	16049	2005-08-23 22:50:12	2666	393	2005-08-30 01:01:12	2	2006-02-16 02:30:53
3	16048	2005-08-23 22:43:07	2019	103	2005-08-31 21:33:07	1	2006-02-16 02:30:53
4	16047	2005-08-23 22:42:48	2088	114	2005-08-25 02:48:48	2	2006-02-16 02:30:53
5	16046	2005-08-23 22:26:47	4364	74	2005-08-27 18:02:47	2	2006-02-16 02:30:53
6	16045	2005-08-23 22:25:26	772	14	2005-08-25 23:54:26	1	2006-02-16 02:30:53
7	16044	2005-08-23 22:24:39	1312	468	2005-08-25 04:08:39	1	2006-02-16 02:30:53
8	16043	2005-08-23 22:21:03	3869	526	2005-08-31 03:09:03	2	2006-02-16 02:30:53
9	16042	2005-08-23 22:20:40	629	131	2005-08-24 17:54:40	1	2006-02-16 02:30:53
10	16041	2005-08-23 22:20:26	4116	121	2005-08-25 20:14:26	2	2006-02-16 02:30:53
11	16040	2005-08-23 22:19:33	3524	195	2005-09-02 02:19:33	2	2006-02-16 02:30:53
12	16039	2005-08-23 22:18:51	545	78	2005-08-31 19:55:51	2	2006-02-16 02:30:53
13	16038	2005-08-23 22:14:31	2612	172	2005-08-30 03:28:31	1	2006-02-16 02:30:53
14	16037	2005-08-23 22:13:04	341	45	2005-09-01 02:48:04	2	2006-02-16 02:30:53


```
14
15  INSERT INTO payment VALUES(DEFAULT, 459, 1, 2, 7.6, NOW()::TIMESTAMP);
16
17  SELECT * FROM rental ORDER BY rental_id DESC;
18  SELECT * FROM payment ORDER BY payment_id DESC;
```


Data Output | Explain | Messages | Notifications

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)	payment_date timestamp without time zone
1	32106	459	1	3	0.00	2020-11-19 16:01:40.663114
2	32105	459	1	2	7.60	2020-11-19 16:01:40.663114
3	32098	264	2	14243	2.99	2007-05-14 13:44:29.996577
4	32097	263	1	15293	0.99	2007-05-14 13:44:29.996577
5	32096	252	2	13756	4.99	2007-05-14 13:44:29.996577
6	32095	251	1	14107	0.99	2007-05-14 13:44:29.996577
7	32094	245	2	12682	2.99	2007-05-14 13:44:29.996577
8	32093	244	2	12736	4.99	2007-05-14 13:44:29.996577
9	32092	236	1	12988	0.99	2007-05-14 13:44:29.996577
10	32091	234	1	15778	0.99	2007-05-14 13:44:29.996577
11	32090	229	2	13295	0.99	2007-05-14 13:44:29.996577
12	32089	228	1	15234	0.00	2007-05-14 13:44:29.996577
13	32088	228	2	12672	3.98	2007-05-14 13:44:29.996577
14	32087	227	2	13374	4.99	2007-05-14 13:44:29.996577

3.d) دو نقش اضافه کنید که یکی از آنها به عنوان کارمند بخش فروش تنها قادر به مشاهده اطلاعات موجودی فروشگاه باشد و دیگری به عنوان مسئول فروشگاه علاوه بر قابلیت‌های یک کارمند بخش فروش، قادر به مشاهده، حذف، افزودن و بروزرسانی در اطلاعات کارمندان و موجودی فروشگاه باشد.

پاسخ :

```
CREATE ROLE SalesEmployee;
GRANT SELECT ON inventory TO SalesEmployee;

CREATE ROLE StoreManager;
GRANT SalesEmployee TO StoreManager;
GRANT SELECT, DELETE, UPDATE, INSERT ON staff TO StoreManager;
GRANT DELETE, UPDATE, INSERT ON inventory TO StoreManager;
```

3.e) اگر بخواهیم هر فرد در نقش های مذکور تنها قادر به مشاهده اطلاعات مربوط به فروشگاه محل کار خود باشد به چه ترتیبی باید عمل کرد؟ مختصراً در پاسخنامه تشریحی خود توضیح دهید.

پاسخ :

کافیست view های زیر را تعریف کنیم :

```
CREATE OR REPLACE VIEW store1_Inventory AS
     SELECT * FROM inventory WHERE store_id = 1;

CREATE OR REPLACE VIEW store2_Inventory AS
     SELECT * FROM inventory WHERE store_id = 2;

CREATE OR REPLACE VIEW store1_Staff AS
     SELECT * FROM staff WHERE store_id = 1;

CREATE OR REPLACE VIEW store2_Staff AS
     SELECT * FROM staff WHERE store_id = 2;
```

همچنین رول های زیر را ایجاد کرده و کاربران را در یکی از این چهار رول ، دسته بندی کنیم.

```
CREATE ROLE SalesEmployee_1;
CREATE ROLE SalesEmployee_2;
CREATE ROLE StoreManager_1;
CREATE ROLE StoreManager_2;
```

سپس مجوزهای خواسته شده را مشابه قسمت قبل به این چهار رول اعطا کنیم :

```
GRANT SELECT ON store1_Inventory TO SalesEmployee_1;
GRANT SELECT ON store2_Inventory TO SalesEmployee_2;

GRANT SalesEmployee_1 TO StoreManager_1;
GRANT SalesEmployee_2 TO StoreManager_2;

GRANT SELECT, DELETE, UPDATE, INSERT ON store1_Staff TO StoreManager_1;
GRANT SELECT, DELETE, UPDATE, INSERT ON store2_Staff TO StoreManager_2;
GRANT DELETE, UPDATE, INSERT ON store1_Inventory TO StoreManager_1;
GRANT DELETE, UPDATE, INSERT ON store2_Inventory TO StoreManager_2;
```

موفق باشید.