سوال اول- الف)

رابطه فقط یک جدول است که به صورت فیزیکی وجود دارد و در پایگاه داده ذخیره می‌شود که این موضوع با دید در تضاد است به این معنی که دید در واقع از مشتقات رابطه است ولی خود به صورت فیزیکی در پایگاه داده وجود ندارد. دید تنها به صورت یک تعریف ساختاری است و داده‌های خود را می‌تواند از تعدادی رابطه استخراج کند.

رابطه در واقع جدولی است که دارای مقادیری است، این در صورتی است که دید زیرمجموعه‌ای از پایگاه داده است.

اما به نظر می‌رسد که تفاوت اساسی دید و رابطه این است که یک رابطه حاوی داده(Data) است اما این در حالی است که دید یک کوئری است که روی یک یا چند رابطه قرار می‌گیرد و خود حاوی هیچ داده‌ای(Data) نیست.

ب)

- هنگامی که سیستم تک کاربره باشد.
- هنگامی که به تشخیص ادمین(admin) برای افزایش کارایی سیستم، برخی برنامه‌ها را مستقیما روی شمای ادراکی(جدول مبنایی) بنویسیم.
- هنگامی که کاربر نیازمند انجام عملیات ذخیره‌سازی باشد و از طریق دید امکان آن وجود نداشته باشد.

ج)

- Super Key یک صفت (یا مجموعه‌ای از صفت‌ها) است که برای شناسایی منحصر به فرد همه صفت‌ها در یک رابطه استفاده می‌شود اما کلید اصلی یک مجموعه حداقلی از صفت‌ها (یا مجموعه ای از صفت‌ها) است که برای شناسایی منحصر به فرد همگی صفت‌ها در یک رابطه استفاده می شود.
- همه Super Key ها نمی‌توانند کلیدهای اصلی باشند ولی کلید اصلی یک Super Key حداقلی است.
- Super Key های مختلف با هم معیارهای انتخاب کلیدهای کاندید را ایجاد می‌کنند این در صورتی است که ما می‌توانیم کمترین کلید کاندید را به عنوان کلید اصلی انتخاب کنیم.
- در یک رابطه، تعداد Super Key ها بیشتر از تعداد کلیدهای اصلی است.
- صفت‌های Super key می‌تواند حاوی مقدار NULL باشد این در حالی است که صفت‌های کلید اصلی نمی‌توانند حاوی مقادیر NULL باشند.

وهمچنین برای Super key در SQL با استفاده از دستور UNIQUE می‌توانیم محدودیت یکتایی مقدار را اعمال کنیم.

د)

اگر در یک جدول n ستون داشته باشیم که k ستون آن کلید کاندید باشد، به صورتی که k <= n باشد تعداد Super key های این جدول $2^{(n-k)}$ است.

سوال دوم-

تمامی کُـــد های مربوط به این سوال به زبان SQL همراه با کامنت مشخص شده و همراه همین گزارش با نام question2.sql آپلود شده است.

سوال سوم- قسمت اول- الف)

Student (ST_ID, Name, National_ID, Semester, Major, Phone_Num)
CK CK CK

Course (CO_ID, Title, Credit, Type, Professor)
CK

ST_CO (ST_ID, CO_ID, Term, Year, Semester)
CK

ب) در این قسمت برای انتخاب کلیدهای کاندید به عنوان کلید اصلی برای جدول Student مناسب‌ترین گزینه ST_ID است و برای جدول Course تنها کلیدی که کاندید است کلید اصلی نیز می‌باشد و برای جدول ST_CO کلید اصلی‌ای نداریم.

سوال سوم- قسمت دوم- الف)

1.

```
CREATE VIEW V1 AS

SELECT ST.ST_ID, ST.Name, ST_CO.CO_ID, ST_CO. Semester FROM
Student AS ST JOIN ST_CO
```

این دید در عمل ناپذیرا است بدلیل اینکه عباراتی که انتخاب(SELECT) شده‌اند از جدول Student ساده نیستند و عملگر JOIN مشاهده می‌شود ولی در تئوری پذیرا می‌باشد.

2.

```
CREATE VIEW V2 AS

SELECT ST_ID, Name, Semester, Major FROM Student

WHERE Major = 'COMP'
```

این دید هم در عمل و هم در تئوری پذیرا است و مشکلی ندارد و جدولی که در کلاز FROM نیز قید شده‌است یک جدول مبنا است و دید قابل بهنگام سازی می‌باشد.

3.

```
CREATE VIEW V3 AS

SELECT Title, Credit

FROM Course
```

این دید هم در عمل و هم در تئوری ناپذیرا است چون در کلاز SELECT ستون کلید نداریم و به همین دلیل پذیرا نیست.

4.

```
CREATE VIEW V4 AS

SELECT ST_ID, AVG(Semester) FROM ST_CO GROUP BY ST_ID
```

این دید در عمل ناپذیرا است بدلیل اینکه در کلاز SELECT کلا GROUP BY نباید داشته باشیم و به همین دلیل پذیرا نیست ولی در تئوری پذیرا است.

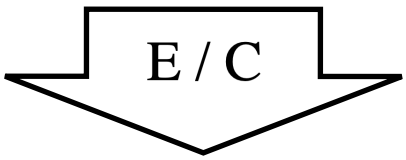
ب)

```
UPDATE V2
SET Major ='math'
WHERE ST_ID = 98543210
```

از نظر تئوریک به دلیل عدم رعایت محدودیت دید رد می‌شود اما در عمل اگر option چک نشود، [with check option] درخواست انجام می‌شود.

ج)

```
INSERT INTO V1
VALUES (98543210, 'Ali', 40638, 4)
```




```
INSERT INTO ST
VALUES (98543210, 'Ali', ?, ?)

INSERT INTO ST_CO
VALUES (98543210, 40438, 4)
```

سوال چهارم-

دو جدول نماد و سهامدار از سامانه بورس اوراق بهادار با نام های shareholder و symbol تعریف شده‌اند که برای هر دو جدول قوانین جامعیتی عام ناوابسته به داده‌های محیط هستند که دو نوع قاعده جامعیت موجودیتی و جامعیت ارجاعی را دارا می‌باشد و برای جدول سهامدار و نماد اگر قاعده جامعیت موجودی(Entity IR) را اعمال کنیم هیچ جزء تشکیل دهنده‌ی کلید های اصلی نباید هیچ‌مقدار(NULL) باشد دلیل آن هم این است که کلید اصلی عامل تمییز نمونه موجودیت‌ها و تاپل‌ها است و تضمین کننده دست‌یابی به تک‌موجودیت است که یه این معنی است که عامل تمییز خود نمیتواند ناشناخته باشد و همچنین مکانیزم اعمال قاعده جامعیت موجودی در DBMS به این صورت است که محدودیت یکتایی مقدار (با UNIQUE فقط این محدودیت کنترل میشود) که برای id جدول نماد تنها UNIQUE آمده که نشان می‌دهد محدودیت هیچ‌مقدارناپذیری را کنترل نمی‌کند این در صورتی است که باید محدودیت هیچ‌مقدارناپذیری کنترل شود.

برای قاعده دوم که قاعده جامعیت ارجاعی(Referential IR) است ناظر است برکلید خارجی که جدول سهامدار اصلا کلید خارجی ندارد و برای جدول نماد باید مقادیر id جدول سازمان در جدول نماد قابل انطباق باشد. دلیل نیاز به قاعده نیز هم این است که کلید خارجی عامل ارجاع است؛ ارجاع به نمونه موجودیت (ارجاع مقداری و نه ارجاع از طریق اشاره‌گر) و در واقعیت نمی‌توان به نمونه موجودیت ناموجود ارجاع داد و همچنین مکانیزم اعمال قاعده جامعیت ارجاعی در DBMS به این صورت است که در مرحله اول کلیدهای خارجی به سیستم معرفی می‌شوند و حتما بعد از تعریف کلیدخارجی، ارجاع آن نیز باید نوشته شود و در آخر هم برای بهنگام سازی مشخص کردن روش اعمال در عملیات حذف و بهنگام سازی مقدار کلید اصلی مشخص شود.(در درج روش خاصی لازم نیست و در صورت عدم وجود تاپل مرجع، درخواست رد می‌شود.)

دو مورد از قوانین جامعیتی خاص به عنوان مثال محدودیت پایگاهی و محدودیت صفتی است که محدودیت صفتی به این نحو بر دو جدول سهامدار و نماد اعمال شده است که مثلا id یک سهامدار، ۱۰ رقمی است و یک شماره تلفن حداکثر ۱۱ رقمی است و یا مثلا در جدول نماد، نام نماد می‌تواند تا ۴۰ کاراکتر داشته باشد و یا بیشینه و کمینه قیمت نماد باید یه صورت عدد نوشته شود و محدودیت پایگاهی نیز ناظر است به تاپل‌های بیش از یک رابطه که به نحوی با هم ارتباط معنایی[منطقی] دارند که در جدول نماد رابطه بین جداول سازمان و نماد به این صورت است.

سوال پنجم-

تمامی کـُـد های مربوط به این سوال به زبان SQL همراه با کامنت مشخص شده و همراه همین گزارش با نام question5.sql آپلود شده است.

الف)

```
CREATE ASSERTION TOT_TRANSACTION_CHECK
        CHECK (NOT EXISTS (SELECT COUNT(deposit.time)
        AS  number_deposit, deposit.depositor AS depositor
                FROM  deposit
                GROUP BY depositor
                HAVING time > time0)
                FULL OUTER JOIN
        SELECT COUNT(withdraw.time) AS number_withdraw,
                withdraw.withdrawer AS withdraw
                FROM  withdraw
                GROUP BY withdrawer
                HAVING time > time0
                ON withdrawer = depositor
        WHERE (number_deposit+number_withdraw>10)
```

ب)

```
CREATE ASSERTION TOTPLAYER-CHECK
        CHECK (NOT EXISTS (SELECT COUNT(PLAYER_ID)
                                AS PLAYER, TEAM_ID)
                FROM CONTRACT
                GROUP BY (TEAM_ID)
                HAVING (PLAYER>15)
```

سوال ششم-

تمامی کـُـد های مربوط به این سوال به زبان SQL همراه با کامنت مشخص شده و همراه همین گزارش با نام question6.sql آپلود شده است.

الف)

```
CREATE TRIGGER PRICE-TRIG
        BEFORE UPDATE OF price
        ON transaction
        REFERENCING NEW AS NPRICE
        FOR EACH ROW
        (UPADATE transaction
                SET transaction.price = NPRICE.price
                WHERE NPRICE.price < 10 and  1 <  NPRICE.price)
```

ب)

```
CREATE TRIGGER PLYR-PAY-TRIG
        AFTER UPDATE ON contract
        REFERENCING NEW AS NPLYR
        FOR EACH ROW
       (UPADATE contract
    SET contract.price = contract.price + 10 WHERE contract.id != NPLYR.id)

CREATE TRIGGER PLYR-PAY-TRIG
        AFTER INSERT ON contract
        REFERENCING NEW AS NPLYR
        FOR EACH ROWs
      (UPADATE contract
    SET contract.price = contract.price + 10 WHERE contract.id != NPLYR.id)
```