عنوان درس:

آزمایشگاه پایگاه داده

عنوان تمرین:

پروژه پایانی

دانشجویان:

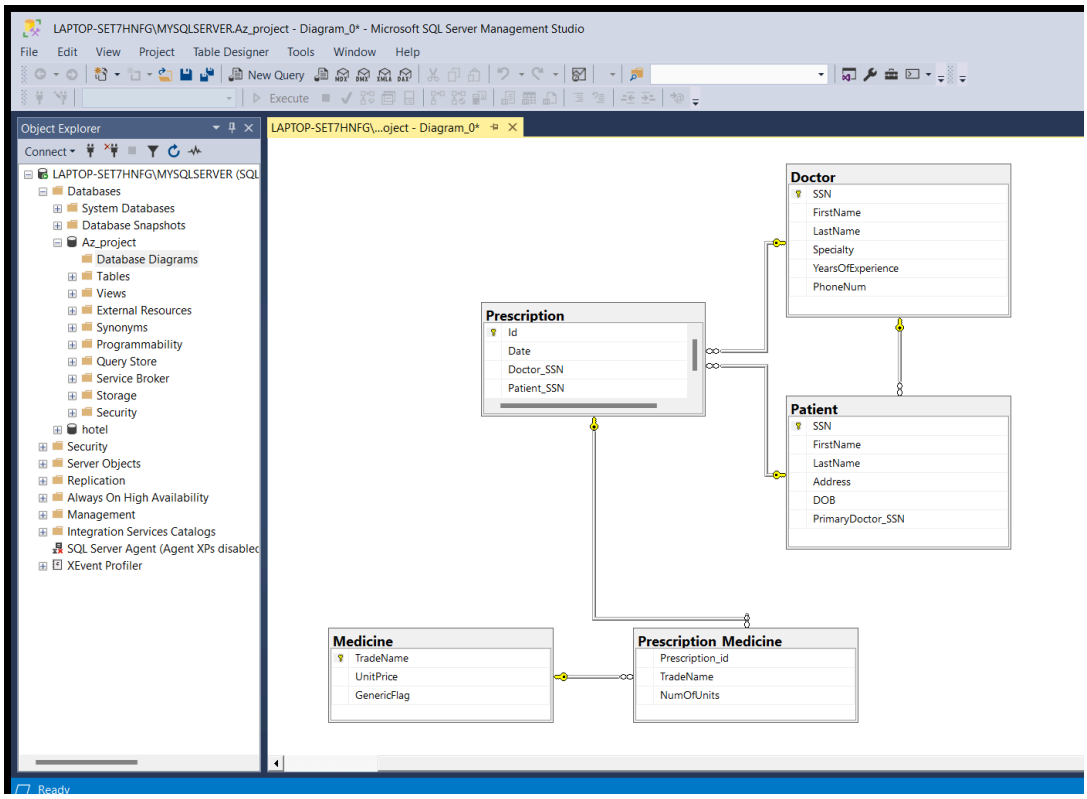
فائزه صالحی

عاطفه علی‌محمدی

پاییز 1403

سوال 2:

تصویر ERD دیتابیس ایجاد شده را نمایش دهید.



سوال 3:

مجموع هزینه پرداختی برای نسخه‌های تجویز شده را به تفکیک هر سال نشان دهید.

```
SELECT
    YEAR(p.Date) AS Year,
    SUM(pm.NumOfUnits * m.UnitPrice) AS TotalCost
FROM
    Prescription_Medicine pm
JOIN
    Prescription p ON pm.Prescription_id = p.Id
JOIN
    Medicine m ON pm.TradeName = m.TradeName
GROUP BY
    YEAR(p.Date)
ORDER BY
    Year;
```

توضیحات:

1. YEAR(p.Date):

سال تاریخ نسخه را از ستون Date در جدول Prescription استخراج می‌کند.

2. SUM(pm.NumOfUnits * m.UnitPrice):

تعداد واحدهای تجویز شده (NumOfUnits) را ضرب در قیمت هر واحد دارو (UnitPrice) می‌کند و مجموع هزینه‌ها را محاسبه می‌کند.

3. JOINها:

جدول Prescription_Medicine با جدول Prescription بر اساس Prescription_id مرتبط است.

جدول Prescription_Medicine با جدول Medicine بر اساس TradeName مرتبط است.

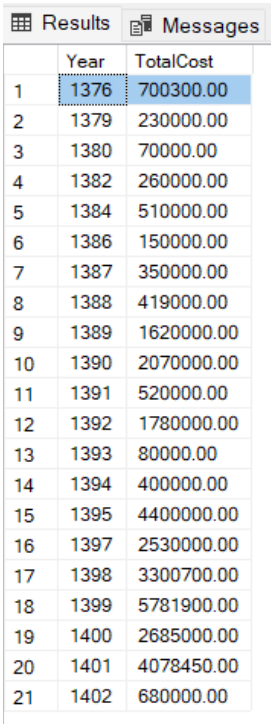
4. GROUP BY YEAR(p.Date):

داده‌ها را بر اساس سال گروه‌بندی می‌کند تا هزینه‌ها برای هر سال به صورت جداگانه محاسبه شود.

5. ORDER BY Year:

نتایج را بر اساس سال مرتب می‌کند.

خروجی:



Results | Messages

	Year	TotalCost
1	1376	700300.00
2	1379	230000.00
3	1380	70000.00
4	1382	260000.00
5	1384	510000.00
6	1386	150000.00
7	1387	350000.00
8	1388	419000.00
9	1389	1620000.00
10	1390	2070000.00
11	1391	520000.00
12	1392	1780000.00
13	1393	80000.00
14	1394	400000.00
15	1395	4400000.00
16	1397	2530000.00
17	1398	3300700.00
18	1399	5781900.00
19	1400	2685000.00
20	1401	4078450.00
21	1402	680000.00

سوال 4:

سه نفر از بیمارانی که بیشترین هزینه را بابت دارو در سال 1400 خرج کردهاند، معرفی کنید.

```
SELECT TOP 3
    p.SSN AS PatientSSN,
    p.FirstName + ' ' + p.LastName AS PatientName,
    SUM(pm.NumOfUnits * m.UnitPrice) AS TotalCost
FROM
    Prescription pr
JOIN
    Patient p ON pr.Patient_SSN = p.SSN
JOIN
    Prescription_Medicine pm ON pr.Id = pm.PrescriptionId
JOIN
    Medicine m ON pm.TradeName = m.TradeName
WHERE
    YEAR(pr.Date) = 1400
GROUP BY
    p.SSN, p.FirstName, p.LastName
ORDER BY
    TotalCost DESC;
```

توضیحات:

1. جداول JOIN:

Prescription_Medicine به Prescription متصل شده است تا ارتباط نسخه‌ها با بیماران مشخص شود.

Prescription_Medicine به Medicine متصل شده است تا قیمت هر دارو به دست آید.

2. محاسبه هزینه کل:

ستون NumOfUnits (تعداد واحدهای دارو) در PricePerUnit (قیمت هر واحد) ضرب شده تا هزینه هر دارو محاسبه شود.

هزینه‌های داروها با استفاده از SUM برای هر بیمار جمع شده است.

3. فیلتر بر اساس سال 1400:

شرط YEAR(p.PrescriptionDate) = 1400 فقط نسخه‌های مربوط به سال 1400 را انتخاب می‌کند.

4. مرتب‌سازی و محدود کردن نتایج:

نتایج بر اساس TotalCost (هزینه کل) به صورت نزولی مرتب شده‌اند.

با استفاده از TOP 3 فقط سه بیمار با بیشترین هزینه نمایش داده می‌شوند.

خروجی:

	PatientSSN	PatientName	TotalCost
1	4487521245	منصور شاهی	2516000.00
2	2145141212	آمنه محمدی	1252000.00
3	2315416320	سارا جمالی	960000.00

سوال 5:

نام و نام خانوادگی بیمارانی که دارای بیشترین تعداد نسخه هستند را گزارش کنید.

```
WITH PatientPrescriptionCount AS (
    SELECT
        p.Patient_SSN AS PatientSSN,
        pa.FirstName,
        pa.LastName,
        COUNT(*) AS PrescriptionCount
    FROM
        Prescription p
    JOIN
        Patient pa ON p.Patient_SSN = pa.SSN
    GROUP BY
        p.Patient_SSN, pa.FirstName, pa.LastName
)
SELECT
    FirstName,
    LastName,
    PrescriptionCount
FROM
    PatientPrescriptionCount
WHERE
    PrescriptionCount = (SELECT MAX(PrescriptionCount) FROM
PatientPrescriptionCount);
```

توضیحات:

1. CTE (Common Table Expression):

در قسمت WITH PatientPrescriptionCount، تعداد نسخه‌های هر بیمار (بر اساس SSN) را محاسبه می‌کنیم و نام و نام خانوادگی بیمار را نیز همراه با تعداد نسخه‌ها ذخیره می‌کنیم.

2. COUNT(*) AS PrescriptionCount:

تعداد نسخه‌های مربوط به هر بیمار را می‌شمارد.

3. GROUP BY p.SSN, pa.FirstName, pa.LastName:

گروه‌بندی بر اساس SSN بیمار (شناسه یکتا)، نام و نام خانوادگی.

4. WHERE PrescriptionCount = SELECT MAX(PrescriptionCount):

بیمارانی را فیلتر می‌کند که تعداد نسخه‌هایشان برابر با بیشترین تعداد نسخه باشد.

خروجی:

Results Messages

	FirstName	LastName	PrescriptionCount
1	آمنه	محمدی	10
2	منصور	شاهی	10

سوال 6:

نام و نام خانوادگی بیمارانی که دارای بیش از 5 نسخه هستند را گزارش کنید. خروجی را بر اساس نام خانوادگی و سپس نام، مرتب کنید.

```
SELECT
    p.FirstName,
    p.LastName,
    COUNT(pr.Id) AS PrescriptionCount
FROM
    Patient p
JOIN
    Prescription pr ON p.SSN = pr.Patient_SSN
GROUP BY
    p.FirstName, p.LastName
HAVING
    COUNT(pr.Id) > 5
ORDER BY
    p.LastName ASC,
    p.FirstName ASC;
```

توضیحات:

1. اتصال جداول (JOIN):

جدول Patient با جدول Prescription متصل شده است تا اطلاعات بیماران و نسخه‌هایشان ترکیب شود.

اتصال از طریق ستون SSN (در جدول Patient) و Patient_SSN (در جدول Prescription) انجام شده است.

2. محاسبه تعداد نسخه‌ها:

تعداد نسخه‌های هر بیمار با استفاده از COUNT(pr.Id) محاسبه شده است.

3. گروه‌بندی بر اساس بیمار:

داده‌ها بر اساس FirstName و LastName گروه‌بندی شده‌اند تا تعداد نسخه‌ها برای هر بیمار جداگانه محاسبه شود.

4. فیلتر بیماران با بیش از 5 نسخه:

شرط HAVING COUNT(pr.Id) > 5 فقط بیمارانی را انتخاب می‌کند که تعداد نسخه‌هایشان بیشتر از 5 است.

5. مرتب‌سازی خروجی:

خروجی بر اساس LastName به صورت صعودی (ASC) مرتب شده است. اگر دو بیمار نام خانوادگی یکسانی داشته باشند، بر اساس FirstName مرتب می‌شوند.

خروجی:

	FirstName	LastName	PrescriptionCount
1	سارا	جمالی	7
2	محمد	جمالی	6
3	منصور	شاهی	10
4	آمنه	محمدی	10

سوال 7:

تابعی بنویسید که با دریافت نام یک دارو، تعداد واحدهای تجویز شده از این دارو را محاسبه کند، سپس از این تابع در یک روال استفاده کرده و نام دو دارویی که بیشترین تعداد تجویز را دارند گزارش دهید.

1. تعریف تابع برای محاسبه تعداد واحدهای تجویز شده از یک دارو

```
CREATE FUNCTION GetTotalUnitsForMedicine (
    @TradeName NVARCHAR(100)
)
RETURNS INT
AS
BEGIN
    DECLARE @TotalUnits INT;

    SELECT
        @TotalUnits = SUM(NumOfUnits)
    FROM
        Prescription_Medicine
    WHERE
        TradeName = @TradeName;

    RETURN ISNULL(@TotalUnits, 0);
END;
```

توضیحات:

1. ورودی تابع: نام تجاری دارو (@TradeName).

2. خروجی تابع: مجموع تعداد واحدهای تجویز شده از این دارو.

3. ISNULL(@TotalUnits, 0): در صورتی که هیچ نسخه‌ای برای داروی مورد نظر وجود نداشته باشد، مقدار NULL را به صفر تبدیل می‌کند.

2. استفاده از تابع در یک روال برای یافتن دو داروی با بیشترین تعداد تجویز

```
CREATE PROCEDURE GetTopTwoMedicines
AS
BEGIN
    -- ایجاد یک جدول موقت برای ذخیره نام داروها و تعداد واحدهای تجویز شده
    CREATE TABLE #MedicineUsage (
        TradeName NVARCHAR(100),
        TotalUnits INT
    );

    -- محاسبه تعداد واحدهای تجویز شده برای هر دارو و درج آن در جدول موقت
    INSERT INTO #MedicineUsage (TradeName, TotalUnits)
    SELECT
        m.TradeName,
        dbo.GetTotalUnitsForMedicine(m.TradeName)
    FROM
        Medicine m;

    -- انتخاب دو داروی با بیشترین تعداد تجویز
    SELECT TOP 2
        TradeName,
        TotalUnits
    FROM
        #MedicineUsage
    ORDER BY
        TotalUnits DESC;

    -- حذف جدول موقت
    DROP TABLE #MedicineUsage;
END;

EXEC GetTopTwoMedicines;
```

توضیحات:

1. جدول موقت (#MedicineUsage):

برای ذخیره نام داروها و تعداد واحدهای تجویز شده استفاده می‌شود.

2. تابع GetTotalUnitsForMedicine:

برای هر دارو در جدول Medicine فراخوانی می‌شود تا تعداد واحدهای تجویز شده را محاسبه کند.

3. TOP 2:

دو دارویی که بیشترین تعداد واحدهای تجویز شده را دارند، انتخاب می‌شود.

4. ORDER BY TotalUnits DESC:

داروها بر اساس تعداد واحدهای تجویز شده به ترتیب نزولی مرتب می‌شوند.

خروجی:

Results Messages

	TradeName	TotalUnits
1	آسپرین	44
2	استامینوفن	27

سوال 8:

نام و نام خانوادگی بیمارانی که یک داروی خاص را از حداقل دو پزشک دریافت کرده‌اند، به همراه نام دارو گزارش کنید.

```
SELECT
    p.FirstName,
    p.LastName,
    pm.TradeName AS MedicineName
FROM
    Patient p
JOIN
    Prescription pr ON p.SSN = pr.Patient_SSN
JOIN
    Prescription_Medicine pm ON pr.Id = pm.PrescriptionId
GROUP BY
    p.FirstName, p.LastName, pm.TradeName, p.SSN
HAVING
    COUNT(DISTINCT pr.Doctor_SSN) >= 2
ORDER BY
    p.LastName ASC, p.FirstName ASC, pm.TradeName ASC;
```

توضیحات:

1. اتصال جداول (JOINs):

جدول Patient با جدول Prescription متصل شده است تا اطلاعات بیماران و نسخه‌هایشان ترکیب شود.

جدول Prescription با جدول Prescription_Medicine متصل شده است تا اطلاعات مربوط به داروهای تجویز شده برای هر نسخه مشخص شود.

2. گروه‌بندی (GROUP BY):

داده‌ها بر اساس نام بیمار (FirstName و LastName)، شماره ملی بیمار (SSN)، و نام دارو (TradeName) گروه‌بندی شده‌اند.

3. فیلتر بیماران با حداقل دو پزشک مختلف:

شرط HAVING COUNT(DISTINCT pr.Doctor_SSN) >= 2 بررسی می‌کند که تعداد پزشکان متفاوتی که یک داروی خاص را برای بیمار تجویز کرده‌اند، حداقل 2 باشد.

4. مرتب‌سازی خروجی:

خروجی بر اساس FirstName، LastName و MedicineName به صورت صعودی مرتب شده است.

خروجی:

	FirstName	LastName	MedicineName
1	مهدی	آقادادی	آسپرین
2	سارا	جمالی	آسپرین
3	محمد	جمالی	آسپرین
4	محمد	جمالی	استامینوفن
5	منصور	شاهی	استامینوفن
6	منصور	شاهی	پنی‌سیلین
7	منصور	شاهی	مفناهیک اسید
8	آمنه	محمدی	آسپرین
9	آمنه	محمدی	دیازپام
10	عباس	مرتضوی	آسپرین

سوال 9:

برای هر سال، تعداد نسخه‌هایی که شامل فقط یک واحد داروی آسپرین هستند را گزارش کنید. (درمورد داروهای دیگر محدودیتی وجود ندارد)

```
SELECT
    YEAR(p.Date) AS Year,
    COUNT(DISTINCT p.Id) AS SingleUnitAspirinPrescriptions
FROM
    Prescription p
JOIN
    Prescription_Medicine pm ON p.Id = pm.Prescription_id
WHERE
    pm.TradeName = N'آسپرین' AND pm.NumOfUnits = 1
GROUP BY
    YEAR(p.Date)
ORDER BY
    Year;
```

توضیحات:

1. SELECT YEAR(p.Date) AS Year:

ستون تاریخ نسخه‌ها (p.Date) را به سال تبدیل می‌کند و آن را به عنوان Year در خروجی نمایش می‌دهد.

2. COUNT(DISTINCT p.Id):

تعداد نسخه‌های یکتایی (p.Id) را که شرایط مشخص‌شده در بخش WHERE را دارند، می‌شمارد. این بخش تضمین می‌کند که هر نسخه فقط یک بار شمارش شود.

3. FROM Prescription p JOIN Prescription_Medicine pm ON p.Id = pm.Prescription_id:

جدول Prescription که اطلاعات نسخه‌ها (مانند تاریخ و شناسه نسخه) را نگهداری می‌کند، با جدول Prescription_Medicine که اطلاعات داروهای تجویز شده در هر نسخه را دارد، بر اساس شناسه نسخه (p.Id و pm.Prescription_id) به هم متصل می‌شود.

4. Where pm.TradeName=N "آسپرین" and pm.NumOfUnits=1:

فقط نسخه‌هایی انتخاب می‌شوند که:

داروی تجویز شده در آن‌ها "آسپرین" باشد (با استفاده از N'آسپرین' برای پشتیبانی از رشته‌های یونیکد).

تعداد واحدهای تجویز شده از داروی "آسپرین" دقیقاً برابر با 1 باشد.

5. GROUP BY YEAR(p.Date):

نتایج را بر اساس سال گروه‌بندی می‌کند. به این ترتیب، تعداد نسخه‌های مربوط به هر سال به صورت جداگانه محاسبه می‌شود.

6. ORDER BY Year:

نتایج را بر اساس سال به ترتیب صعودی مرتب می‌کند.

خروجی:

Results | Messages

	Year	SingleUnitAspirinPrescriptions
1	1390	1
2	1392	1
3	1397	1
4	1402	2

سوال 10:

به تفکیک هر سال، درصد نسخه‌هایی که شامل حداقل یک واحد داروی آسپرین هستند را به همراه تعداد کل نسخه‌های آن سال گزارش کنید.

```
WITH AspirinPrescriptions AS (
    SELECT DISTINCT P.Id, YEAR(P.Date) AS PrescriptionYear
    FROM Prescription P
    JOIN Prescription_Medicine PM ON P.Id = PM.Prescription_id
    WHERE PM.TradeName = N'آسپرین'
),
YearlyPrescriptionCount AS (
    SELECT YEAR(Date) AS PrescriptionYear, COUNT(*) AS TotalPrescriptions
    FROM Prescription
    GROUP BY YEAR(Date)
),
YearlyAspirinCount AS (
    SELECT PrescriptionYear, COUNT(*) AS AspirinPrescriptions
    FROM AspirinPrescriptions
    GROUP BY PrescriptionYear
)
SELECT
    YPC.PrescriptionYear,
    COALESCE(YAC.AspirinPrescriptions, 0) AS AspirinPrescriptions,
    YPC.TotalPrescriptions,
              COALESCE(ROUND((CAST(YAC.AspirinPrescriptions   AS   FLOAT)   /
YPC.TotalPrescriptions) * 100, 2), 0) AS AspirinPercentage
FROM
    YearlyPrescriptionCount YPC
LEFT JOIN
    YearlyAspirinCount YAC
ON
    YPC.PrescriptionYear = YAC.PrescriptionYear
WHERE
    COALESCE(YAC.AspirinPrescriptions, 0) > 0
ORDER BY
    YPC.PrescriptionYear;
```

توضیحات:

1. انتخاب ستون‌ها:

YPC.PrescriptionYear: سال نسخه‌ها.

COALESCE(YAC.AspirinPrescriptions, 0): تعداد نسخه‌های حاوی آسپرین. اگر برای سالی داده‌ای وجود نداشته باشد، مقدار 0 برگردانده می‌شود.

YPC.TotalPrescriptions: تعداد کل نسخه‌ها در سال.

COALESCE(ROUND(CAST(YAC.AspirinPrescriptions AS FLOAT) / YPC.TotalPrescriptions) * 100, 2), 0):

درصد نسخه‌های حاوی آسپرین را محاسبه می‌کند.

CAST(... AS FLOAT): تبدیل تعداد نسخه‌ها به نوع اعشاری برای محاسبه دقیق درصد.

ROUND(..., 2): نتیجه را تا دو رقم اعشار گرد می‌کند.

COALESCE(..., 0): اگر تعداد نسخه‌های حاوی آسپرین صفر باشد، مقدار 0 بازگردانده می‌شود.

2. اتصال جداول:

LEFT JOIN: جدول YearlyPrescriptionCount (که تعداد کل نسخه‌ها را دارد) با جدول YearlyAspirinCount (که تعداد نسخه‌های حاوی آسپرین را دارد) بر اساس سال متصل می‌شود.

این اتصال از نوع LEFT JOIN است تا مطمئن شویم که تمامی سال‌ها در خروجی حضور دارند، حتی اگر در آن سال هیچ نسخه‌ای حاوی آسپرین نباشد.

3. فیلتر:

WHERE COALESCE(YAC.AspirinPrescriptions, 0) > 0: فقط سال‌هایی که حداقل یک نسخه حاوی آسپرین دارند در خروجی نمایش داده می‌شوند.

4. مرتب‌سازی:

ORDER BY YPC.PrescriptionYear: خروجی بر اساس سال مرتب می‌شود.

خروجی:

	PrescriptionYear	AspirinPrescriptions	TotalPrescriptions	AspirinPercentage
1	1376	0	1	0
2	1379	0	1	0
3	1380	0	1	0
4	1382	0	1	0
5	1384	0	1	0
6	1386	1	1	100
7	1387	1	1	100
8	1388	0	1	0
9	1389	1	5	20
10	1390	2	4	50
11	1391	0	1	0
12	1392	2	2	100
13	1393	0	1	0
14	1394	0	1	0
15	1395	3	5	60
16	1397	1	4	25
17	1398	0	5	0
18	1399	0	5	0
19	1400	1	7	14.29
20	1401	1	3	33.33
21	1402	2	2	100

سوال 11:

نسخه یا نسخه‌هایی که دارای بیشترین تنوع دارویی هستند را گزارش کنید (تعداد واحدهای هر دارو اهمیت ندارد). خروجی باید شامل یک یا چند رکورد از جدول Prescription به همراه تعداد نوع داروهای تجویز شده باشد.

```
WITH PrescriptionDrugCounts AS (
    SELECT
        p.Id AS PrescriptionId,
        p.Date,
        COUNT(DISTINCT pm.TradeName) AS DrugCount
    FROM
        Prescription p
    JOIN
        Prescription_Medicine pm ON p.Id = pm.Prescription_id
    GROUP BY
        p.Id, p.Date
),
MaxDrugCount AS (
    SELECT
        MAX(DrugCount) AS MaxCount
    FROM
        PrescriptionDrugCounts
)
SELECT
    p.PrescriptionId,
    p.Date,
    p.DrugCount
FROM
    PrescriptionDrugCounts p
JOIN
    MaxDrugCount m ON p.DrugCount = m.MaxCount;
```

توضیحات:

1. COUNT(DISTINCT pm.TradeName) AS DrugCount:

تعداد انواع مختلف داروها (بدون توجه به تعداد واحدها) در هر نسخه را محاسبه می‌کند.

2. MAX(DrugCount) AS MaxCount:

بیشترین مقدار تنوع دارویی (بیشترین تعداد انواع داروها) را از بین تمام نسخه‌ها پیدا می‌کند.

3. JOIN MaxDrugCount m ON p.DrugCount = m.MaxCount:

نسخه‌هایی را انتخاب می‌کند که تعداد انواع داروهای آن‌ها برابر با بیشترین مقدار محاسبه‌شده باشد.

4. WITH PrescriptionDrugCounts و WITH MaxDrugCount:

ابتدا تعداد انواع داروها برای هر نسخه محاسبه شود (PrescriptionDrugCounts).

سپس بیشترین مقدار این تنوع دارویی پیدا شود (MaxDrugCount).

خروجی:

Results Messages

	PrescriptionId	Date	DrugCount
1	8	1392-05-08	3
2	16	1399-03-02	3
3	28	1388-05-06	3
4	46	1402-10-04	3

سوال 12:

دو نسخه‌ای که دارای بیشترین شباهت از نظر نوع دارو تجویز شده هستند را گزارش کنید. خروجی باید شامل دو رکورد از جدول Prescription باشد. (اگر دو نسخه کامل مشابه هستند، در نظر گرفته نشوند)

```
WITH Prescription_Medicine_List AS (
    SELECT
        Prescription_id,
            STRING_AGG(TradeName, ',') WITHIN GROUP (ORDER BY TradeName) AS
MedicineList
    FROM Prescription_Medicine
    GROUP BY Prescription_id
),
Pairwise_Comparison AS (
    SELECT
        p1.Prescription_id AS PrescriptionId1,
        p2.Prescription_id AS PrescriptionId2,
        COUNT(DISTINCT pm1.TradeName) AS CommonMedicines
    FROM Prescription_Medicine pm1
    JOIN Prescription_Medicine pm2
        ON pm1.TradeName = pm2.TradeName
       AND pm1.Prescription_id < pm2.Prescription_id
    JOIN Prescription_Medicine_List p1
        ON pm1.Prescription_id = p1.Prescription_id
    JOIN Prescription_Medicine_List p2
        ON pm2.Prescription_id = p2.Prescription_id
    WHERE p1.MedicineList != p2.MedicineList
    GROUP BY p1.Prescription_id, p2.Prescription_id
),
Max_CommonMedicines AS (
    SELECT
        MAX(CommonMedicines) AS MaxCommon
    FROM Pairwise_Comparison
)
SELECT
    pc.PrescriptionId1,
    pc.PrescriptionId2,
    pc.CommonMedicines
FROM Pairwise_Comparison pc
JOIN Max_CommonMedicines mc
    ON pc.CommonMedicines = mc.MaxCommon;
```

توضیحات:

1. انتخاب ستون‌ها:

pc.PrescriptionId1: شناسه نسخه اول از جفت نسخه‌هایی که بیشترین شباهت را دارند.

pc.PrescriptionId2: شناسه نسخه دوم از جفت نسخه‌هایی که بیشترین شباهت را دارند.

pc.CommonMedicines: تعداد داروهای مشترک بین این دو نسخه.

2. ساخت لیست داروها برای هر نسخه (Prescription_Medicine_List):

Prescription_id: شناسه نسخه.

3. STRING_AGG(TradeName, ',') WITHIN GROUP (ORDER BY TradeName):

تمام داروهای مربوط به یک نسخه را به صورت یک لیست متنی ترکیب می‌کند.

داروها بر اساس نام مرتب می‌شوند تا ترتیب لیست ثابت باشد.

GROUP BY Prescription_id: داده‌ها را بر اساس شناسه نسخه گروه‌بندی می‌کند.

4. مقایسه جفت نسخه‌ها (Pairwise_Comparison):

p1.Prescription_id AS PrescriptionId1: شناسه نسخه اول در جفت نسخه‌ها.

p2.Prescription_id AS PrescriptionId2: شناسه نسخه دوم در جفت نسخه‌ها.

COUNT(DISTINCT pm1.TradeName) AS CommonMedicines: تعداد داروهای مشترک بین نسخه اول و نسخه دوم.

ON pm1.TradeName = pm2.TradeName: شرط اتصال برای یافتن داروهای مشترک بین دو نسخه.

AND pm1.Prescription_id < pm2.Prescription_id: تضمین می‌کند که هر جفت نسخه فقط یک بار مقایسه شود.

WHERE p1.MedicineList != p2.MedicineList: نسخه‌هایی که لیست داروهایشان دقیقاً یکسان است، حذف می‌شوند.

GROUP BY p1.Prescription_id, p2.Prescription_id: جفت نسخه‌ها را گروه‌بندی می‌کند.

یافتن بیشترین تعداد داروی مشترک (Max_CommonMedicines):

MAX(CommonMedicines) AS MaxCommon: بیشترین تعداد داروهای مشترک بین دو نسخه را محاسبه می‌کند.

5. اتصال جداول:

JOIN Max_CommonMedicines: اتصال بین جدول مقایسه نسخه‌ها (Pairwise_Comparison) و جدول شامل بیشترین تعداد داروهای مشترک (Max_CommonMedicines) انجام می‌شود.

ON pc.CommonMedicines = mc.MaxCommon: فقط جفت نسخه‌هایی که تعداد داروهای مشترکشان برابر با حداکثر مقدار هستند، انتخاب می‌شوند.

6. مرتب‌سازی:

ORDER BY pc.CommonMedicines DESC: جفت نسخه‌ها بر اساس تعداد داروهای مشترک به صورت نزولی مرتب می‌شوند (اگر بخواهید).

خروجی:

	PrescriptionId1	PrescriptionId2	CommonMedicines
1	17	27	2
2	8	46	2

سوال 13:

فرض کنید داروی آسپرین با کلونازپام در تضاد است. بیمارانی را پیدا کنید که قبل از دریافت کلونازپام، داروی آسپرین برایشان تجویز شده بود. سپس داروی کلونازپام صادر شده برای آن بیماران را با دیازپام جایگزین کنید.

1. شناسایی بیماران:

```
WITH PatientsWithConflict AS (
    SELECT
        pm1.Prescription_id AS AspirinPrescriptionId,
        p1.Patient_SSN,
        p1.Date AS AspirinDate,
        pm2.Prescription_id AS ClonazepamPrescriptionId,
        p2.Date AS ClonazepamDate
    FROM
        Prescription p1
    JOIN
        Prescription_Medicine pm1 ON p1.Id = pm1.Prescription_id
    JOIN
        Prescription p2 ON p1.Patient_SSN = p2.Patient_SSN
    JOIN
        Prescription_Medicine pm2 ON p2.Id = pm2.Prescription_id
    WHERE
        pm1.TradeName = N'آسپرین'
        AND pm2.TradeName = N'کلونازپام'
        AND p1.Date < p2.Date -- تاریخ آسپرین قبل از کلونازپام باشد
)
SELECT * FROM PatientsWithConflict;
```

توضیحات:

1. pm1.TradeName = N"آسپرین" and pm2.TradeName = N"کلونازپام"

بررسی می‌کند که نسخه اول شامل داروی "آسپرین" و نسخه دوم شامل داروی "کلونازپام" باشد.

2. p1.Date < p2.Date:

اطمینان می‌دهد که نسخه حاوی "آسپرین" زودتر از نسخه حاوی "کلونازپام" صادر شده باشد.

3. p1.Patient_SSN = p2.Patient_SSN:

اطمینان می‌دهد که هر دو نسخه متعلق به یک بیمار باشند.

4. WITH PatientsWithConflict:

لیست بیماران دارای تضاد دارویی را استخراج می‌کند.

خروجی:

Results Messages

	AspirinPrescriptionId	Patient_SSN	AspirinDate	ClonazepamPrescriptionId	ClonazepamDate
1	7	2315416320	1392-01-02	26	1398-08-18
2	21	2315416320	1395-06-26	26	1398-08-18
3	31	2315416320	1387-08-06	26	1398-08-18

2. جایگزینی "کلونازپام" با "دیازپام":

```
UPDATE Prescription_Medicine
SET TradeName = N'دیازپام'
WHERE Prescription_id IN (
    SELECT ClonazepamPrescriptionId
    FROM (
        SELECT
            pm1.Prescription_id AS AspirinPrescriptionId,
            p1.Patient_SSN,
            p1.Date AS AspirinDate,
            pm2.Prescription_id AS ClonazepamPrescriptionId,
            p2.Date AS ClonazepamDate
        FROM
            Prescription p1
        JOIN
            Prescription_Medicine pm1 ON p1.Id = pm1.Prescription_id
        JOIN
            Prescription p2 ON p1.Patient_SSN = p2.Patient_SSN
        JOIN
```

```
            Prescription_Medicine pm2 ON p2.Id = pm2.Prescription_id
        WHERE
            pm1.TradeName = N'آسپرین'
            AND pm2.TradeName = N'کلونازپام'
            AND p1.Date < p2.Date
    ) AS ConflictedPatients
);
```

توضیحات:

1. SET TradeName = N دیازپام:

داروی "کلونازپام" را به "دیازپام" تغییر می‌دهد.

2. RE Prescription_id IN(...) :

فقط نسخه‌هایی را که در زیرکوئری مشخص شده‌اند (حاوی "کلونازپام") به‌روزرسانی می‌کند.

3. زیرکوئری داخلی:

همان CTE مرحله قبل است که نسخه‌های حاوی "کلونازپام" را برای بیماران دارای تضاد دارویی پیدا می‌کند.

4. Prescription_Medicine:

جدول نسخه‌ها را به‌روزرسانی می‌کند.

سوال 14:

جدولی به نام Appointment ساخته که اطلاعات ملاقاتهای بیماران با پزشکان را نگهداری میکند. مشخص کنید که این جدول به چه ویژگی‌هایی نیاز دارد و کلیدهای لازم آن را نیز مشخص کنید.

```
CREATE TABLE Appointment (
    AppointmentId INT PRIMARY KEY,
    Patient_SSN NvarCHAR(10) NOT NULL,
    Doctor_SSN  NvarCHAR(10) NOT NULL,
    AppointmentDateTime DATETIME NOT NULL,
    Status NVARCHAR(50) NOT NULL,
    FOREIGN KEY(Patient_SSN) REFERENCES Patient(SSN),
    FOREIGN KEY (Doctor_SSN) REFERENCES Doctor(SSN)
);
```

ویژگی‌های موردنیاز جدول Appointment:

شناسه یکتا برای هر ملاقات:

هر ملاقات باید یک شناسه یکتا داشته باشد تا بتوان آن را به صورت منحصربه‌فرد شناسایی کرد. این شناسه به عنوان کلید اصلی (Primary Key) عمل می‌کند.

ارتباط با بیمار:

هر ملاقات باید مربوط به یک بیمار باشد. بنابراین، باید ستونی برای نگهداری شناسه بیمار (PatientId) وجود داشته باشد که به جدول Patient متصل شود.

ارتباط با پزشک:

هر ملاقات باید مربوط به یک پزشک باشد. بنابراین، باید ستونی برای نگهداری شناسه پزشک (DoctorId) وجود داشته باشد که به جدول Doctor متصل شود.

تاریخ و زمان ملاقات:

نیاز است که تاریخ و زمان دقیق ملاقات ثبت شود (مثلاً ستونی به نام AppointmentDateTime).

وضعیت ملاقات:

وضعیت ملاقات (مانند برنامه‌ریزی‌شده، لغوشده، انجام‌شده) باید ثبت شود. این اطلاعات می‌تواند در قالب یک ستون Status ذخیره شود.

کلید اصلی (Primary Key):

ستون AppointmentId به‌عنوان کلید اصلی برای شناسایی یکتای هر ملاقات.

کلید خارجی (Foreign Key):

PatientId: به ستون PatientId در جدول Patient متصل می‌شود.

DoctorId: به ستون DoctorId در جدول Doctor متصل می‌شود.

سوال 15:

برای سوال قبل تعدادی داده وارد کنید. دو سوال طراحی کنید که ارتباط بین جدول Appointment با دیگر جداول موجود را شامل میشود. دادههای واردشده را نیز ارسال کنید.

وارد کردن داده ها:

```
INSERT INTO Appointment (PatientId, DoctorId, AppointmentDateTime, Status,
Notes)
VALUES
('1250519854', '0012542010', '2024-01-05 10:00:00', 'Scheduled'),
('2315416320', '1250514920', '2024-01-06 15:00:00', 'Completed'),
('6521258456', '9250523651', '2024-01-07 09:30:00', 'Cancelled'),
('8541200145', '8412506541', '2024-01-08 11:00:00', 'Scheduled');
```

1. نمایش اطلاعات ملاقات‌های انجام‌شده (Completed) به همراه نام بیمار و تخصص پزشک:

```
SELECT
    A.AppointmentId,
    P.FirstName AS PatientFirstName,
    P.LastName AS PatientLastName,
    D.FirstName AS DoctorFirstName,
    D.LastName AS DoctorLastName,
    D.Specialty,
    A.AppointmentDateTime,
    A.Status
FROM
    Appointment A
JOIN
    Patient P ON A.PatientId = P.PatientId
JOIN
    Doctor D ON A.DoctorId = D.DoctorId
WHERE
    A.Status = 'Completed';
```

خروجی:

	AppointmentId	PatientFirstName	PatientLastName	DoctorFirstName	DoctorLastName	Specialty	AppointmentDateTime	Status
1	2	سارا	جعالی	فرشاد	اخوتیان	فیزیوتراپ	2024-01-06 15:00:00.000	Completed

2. نمایش تعداد ملاقات‌های برنامه‌ریزی‌شده (Scheduled) برای هر پزشک:

```
SELECT
    D.FirstName AS DoctorFirstName,
    D.LastName AS DoctorLastName,
    D.Specialty,
    COUNT(A.AppointmentId) AS ScheduledAppointments
FROM
    Appointment A
JOIN
    Doctor D ON A.DoctorId = D.DoctorId
WHERE
    A.Status = 'Scheduled'
GROUP BY
    D.FirstName, D.LastName, D.Specialty;
```

خروجی:

	DoctorFirstName	DoctorLastName	Specialty	ScheduledAppointments
1	عباسعلی	کریمی	جراح قلب و عروق	1
2	علی	مطهری	جراحی غدد	1

سوال 16:

اتصالی بین زبان برنامه‌نویسی C# و دیتابیس تعریف شده در server sql ایجاد کرده و یک کوئری select و یک کوئری insert از طریق یک فرم اجرا کنید.)ورودی موردنیاز از طریق UI به برنامه داده شود.(در یک فیلم کوتاه، برنامه نوشته شده و همچنین دیتابیس قبل و بعد از انجام عملیات را نشان دهید.

لینک ویدیو:

https://drive.google.com/file/d/1U6ivxXLNxEx1ys_f9-9_pGUqXHhdYlZL/view?usp=sharing

سوال نمره اضافه. به کمک یک زبان برنامه نویسی غیر از C#، یک UI مانند اپلیکیشن، صفحه وب یا غیره ایجاد کنید؛ سپس از طریق آن UI، از کاربر ورودی دریافت کرده و به یکی از جداول insert کنید. همچنین عملیات update و delete را برای چند رکورد انجام دهید)مشخصات رکورد باید از طریق UI دریافت شود(. در یک فیلم کوتاه، برنامه نوشته شده و همچنین دیتابیس قبل و بعد از انجام عملیات را نشان دهید.

لینک ویدیو:

https://drive.google.com/file/d/1O9e_A0um9KzZB9tkQ09RcdVluH9-F1wq/view?usp=sharing

لینک دسترسی به کدهای C# و django:

https://drive.google.com/drive/folders/1K8dewPXdC3Ps7xZH3XWalOSBxpS2QVsB?usp=sharing