Trigger -

Updates AvgRiskScore for Patients when a new Diagnosis of Moderate or Severe severity is added to the database.

```
CREATE TRIGGER trg_after_diagnosis_insert
AFTER INSERT ON Diagnosis
FOR EACH ROW
BEGIN
  IF NEW.Severity IN ('Moderate','Severe') THEN
    UPDATE Patient p
      SET p.AvgRiskScore = (
        SELECT ROUND(AVG(d.RiskScore),2)
        FROM Visit v
        JOIN Diagnosis d
          ON v.VisitID = d.VisitID
        WHERE v.PatientID = (
          SELECT PatientID
          FROM Visit
          WHERE VisitID = NEW.VisitID
        )
      )
    WHERE p.PatientID = (
      SELECT PatientID
      FROM Visit
      WHERE VisitID = NEW.VisitID
    );
  END IF;
```

Transaction -

Generate two related “combined” reports in one go, and guarantee that the user’s UI either sees both complete result-sets or none at all (we don't want “top patients” list based on one version of the data and your “hospital counts” based on another -> REPEATABLE READ has frozen snapshot, so no dirty reads)

```
//combined-reports transaction
exports.combinedReports = async (req, res) => {
  const conn = await db.getConnection();
  try {
    await conn.query('SET TRANSACTION ISOLATION LEVEL REPEATABLE READ');
    await conn.query('BEGIN');

    //AQ #1: top diabetes patients (most diagnoses of diabetes -> highest diabetes risk)
```

```
    const [diabetesRows] = await conn.query(`
      SELECT
        p.PatientID,
        p.FirstName,
        p.LastName,
        COUNT(d.DiseaseID) AS DiabetesCount
      FROM Patient p
      JOIN Visit v ON p.PatientID = v.PatientID
      JOIN Diagnosis d ON v.VisitID   = d.VisitID
      WHERE d.DiseaseID = 1
        AND d.RiskScore > 5
        AND d.Severity  IN ('Moderate','Severe')
        AND v.VisitDate BETWEEN '2004-01-01' AND '2024-12-31'
      GROUP BY p.PatientID, p.FirstName, p.LastName
      HAVING COUNT(d.DiseaseID) >= 2
      ORDER BY DiabetesCount DESC
      LIMIT 15
    `);

    // AQ #2: diseased patient count for each hospital
    const [diseasedRows] = await conn.query(`
      SELECT
        h.Name,
        COUNT(DISTINCT p.PatientID) AS DiseasedPatientCount
      FROM Hospital h
      JOIN Patient p ON h.HospitalID = p.HospitalID
      JOIN Visit v ON p.PatientID = v.PatientID
      JOIN Diagnosis d ON v.VisitID = d.VisitID
      WHERE h.Rating > 2
        AND d.RiskScore > 3
        AND v.VisitDate > '2000-01-01'
      GROUP BY h.HospitalID, h.Name
      ORDER BY DiseasedPatientCount DESC
      LIMIT 15
    `);
    await conn.query('COMMIT');
    res.render('admin/combined_reports', {diabetesRows, diseasedRows});
  } catch (err) {
    await conn.query('ROLLBACK');
    handleError(res, err);
  } finally {
    conn.release();
  }
};
```

Constraints -

Attribute level

Diagnosis can only be the following severity (mild, moderate, or severe)
Diagnosis risk score has to be between 0 and 10

```
ALTER TABLE Diagnosis
  ADD CONSTRAINT severity_checker
    CHECK (Severity IN ('Mild','Moderate','Severe')),
  ADD CONSTRAINT riskscore_checker
    CHECK (RiskScore BETWEEN 0 AND 10);
```

Hospital rating must be between 1 and 5

```
ALTER TABLE Hospital
  ADD CONSTRAINT rating_checker
    CHECK (Rating BETWEEN 1 AND 5);
```

Tuple level

Each patient has at most 1 visit per date

```
ALTER TABLE Visit
  ADD CONSTRAINT visitdate_checker
    UNIQUE (PatientID, VisitDate);
```

Stored Procedure -

Generates the top 15 patients by average Moderate/Severe diagnosis risk (>= 2 diseases since Jan 1, 2020) and, when hospital_check = 1, returns up to 15 hospitals' patient counts, visit counts, and average risk for hospitals rated > 2 since Jan 1, 2020.

```
DROP PROCEDURE IF EXISTS GetOverviewReport;
CREATE PROCEDURE GetOverviewReport(
  IN hospital_check TINYINT
)
BEGIN
  -- AQ #3: patients with highest risk score since 2020
  SELECT
    p.PatientID,
    p.FirstName,
    p.LastName,
    ROUND(AVG(d.RiskScore),2) AS AvgRiskScore,
```

```
  COUNT(d.DiseaseID) AS DiagnosisCount
FROM Patient p
JOIN Visit v ON p.PatientID = v.PatientID
JOIN Diagnosis d ON v.VisitID = d.VisitID
WHERE d.Severity IN ('Moderate','Severe')
  AND v.VisitDate >= '2020-01-01'
GROUP BY p.PatientID, p.FirstName, p.LastName
HAVING COUNT(d.DiseaseID) >= 2
ORDER BY AvgRiskScore DESC
LIMIT 15;

-- AQ #4: hospital visit count since 2020
IF hospital_check = 1 THEN
  SELECT
    h.Name,
    COUNT(DISTINCT p.PatientID) AS PatientCount,
    COUNT(DISTINCT v.VisitID) AS VisitCount,
    ROUND(AVG(d.RiskScore),2) AS AvgRiskScore
  FROM Hospital h
  LEFT JOIN Patient p ON h.HospitalID = p.HospitalID
  LEFT JOIN Visit v ON p.PatientID = v.PatientID
  LEFT JOIN Diagnosis d ON v.VisitID = d.VisitID
  WHERE v.VisitDate >= '2020-01-01'
    AND h.Rating > 2
  GROUP BY h.HospitalID, h.Name
  ORDER BY AvgRiskScore DESC
  LIMIT 15;
END IF;
```