SQL queries

(i)
```
SELECT avg(s.Semester)
FROM Studenten s
WHERE 1
```

(ii)
```
SELECT *
FROM Studenten s
WHERE s.Semester > (
    SELECT avg(s.Semester)
    FROM Studenten s
    WHERE 1)
```

(iii)
```
(SELECT PersNr as Id, Name FROM Professoren)
UNION
(SELECT PersNr as Id, Name FROM Assistenten)
UNION
(SELECT Legi as Id, Name FROM Studenten)
ORDER BY Id DESC
```

(iv)
```
SELECT DISTINCT s.Name
FROM Studenten s
JOIN hören h ON s.Legi = h.Legi
JOIN Vorlesungen v ON h.VorlNr = v.VorlNr
JOIN Professoren p ON v.gelesenVon = p.PersNr
WHERE p.Name = 'Sokrates'
```

(v)
```
SELECT v.gelesenVon, p.Name, sum(v.KP)
FROM Vorlesungen v, Professoren p
WHERE v.gelesenVon = p.PersNr
GROUP BY v.gelesenVon, p.Name
```

(vi)
```
SELECT v2.Titel
FROM Vorlesungen v1
JOIN voraussetzen vor ON v1.VorlNr = vor.Vorgänger
JOIN Vorlesungen v2 ON vor.Nachfolger = v2.VorlNr
WHERE v1.Titel = 'Grundzuege'
```

Bonus: No SQL cannot handle transitivity. However, a lot of vendors offer custome extentions. Here the Oracle CONNECT BY clause:

(vii)
```
SELECT Titel
FROM Vorlesungen
WHERE VorlNr in (
    SELECT Vorgänger
    FROM voraussetzen
    CONNECTED BY Nachfolger = PRIOR Vorgänger
    START WITH Nachfolger = (
            SELECT VorlNr
            from Vorlesungen
            where Titel = '...' ));
```

(viii)
```
SELECT s.Legi, s.Name, count(*) as NumBekannter
FROM Studenten s, (
    SELECT DISTINCT h1.Legi as Student, h2.Legi as Bekannter
    FROM hören h1, hören h2
    WHERE h1.VorlNr = h2.VorlNr AND h2.Legi <> h1.Legi
    ) b
WHERE s.Legi = b.Student
GROUP BY s.Legi, s.Name
ORDER BY NumBekannter DESC;
```