1 The Tables

This is similar to the student DB used for the simple SQL examples, with added tables and references.

1.1 Degree

```
CREATE TABLE Degree (
  deg_code  CHAR(2)  PRIMARY KEY,
  d_name  VARCHAR(20)  UNIQUE NOT NULL
);
```

deg_code	d_name
BS	Bachelor of Science
BA	Bachelor of Arts
MS	Master of Science

1.2 Major

```
CREATE TABLE Major (
  code CHAR(3) PRIMARY KEY,
  m_name VARCHAR(30) UNIQUE NOT NULL
);
```

code	m_name
CS	Computer Science
IT	Information Technology
SWE	Software Engineering

1.3 Standing

```
CREATE TABLE Standing (
  deg_code      char(2) REFERENCES Degree(deg_code),
  min_cr        INTEGER DEFAULT 0 NOT NULL,
  max_cr        INTEGER NOT NULL,
  num           INTEGER NOT NULL,
  designation   VARCHAR(20) NOT NULL,
  CONSTRAINT Standing_PK
        PRIMARY KEY (deg_code, num),
  CONSTRAINT Standing_Unique_Designation
        UNIQUE (deg_code, designation),
  CONSTRAINT Standing_min_max
        CHECK (min_cr <= max_cr)
);
```

deg_code	min_cr	max_cr	num	designation
BS	0	30	1	Freshman
BS	31	60	2	Sophomore
BS	61	90	3	Junior
BS	91	10000	4	Senior
BA	0	30	1	Freshman
BA	31	60	2	Sophomore
BA	61	90	3	Junior
BA	91	10000	4	Senior
MS	0	1000	1	Graduate Student

1.4 Student

```
CREATE TABLE Student (
  Id CHAR(3) PRIMARY KEY,
  Name VARCHAR(20) NOT NULL,
  Age INT DEFAULT 20 CHECK(Age>0 AND AGE<100),
  Sex CHAR NOT NULL,
  Deg_code CHAR(2) NOT NULL REFERENCES Degree(deg_code),
  Major CHAR(3) REFERENCES Major(code),
  creds  INTEGER
);
```

id	name	age	gender	deg_code	major	credits
111	Lina Colli	22	F	BS	CS	20
222	Juan Perez	21	M	BA		30
333	John Perez	18	M	BS		40
444	Jane Perez	21	F	BA	CS	50
555	John Doe	20	M	MS	IT	60
666	Jane Doe	22	F	MS	SWE	27
777	David Perez	22	M	BS	IT	35
888	David Smith	18	M	BS	SWE	45
999	Eva Smith	18	F	MS	CS	95

2 Individual Examples

2.1 Implicit Joins

```
SELECT *
FROM Student s, Major m
WHERE s.major=m.code
```

2.2 JOIN ON

```
SELECT *
FROM Student JOIN Major ON major=code
```

2.3 NATURAL JOIN

```
SELECT *
FROM Student NATURAL JOIN Degree
```

Or equivalently

```
SELECT *
FROM Student JOIN Degree USING(deg_code)
```

Compare with:

```
SELECT *
FROM Student JOIN Degree ON
        Student.deg_code=Degree.deg_code
```

2.4 OUTER JOIN

```
SELECT *
FROM Student LEFT OUTER JOIN Major ON major=code
```

compare with:

```
SELECT *
FROM Student JOIN Major ON major=code

SELECT *
FROM Student RIGHT OUTER JOIN Major ON major=code

SELECT *
FROM Student FULL OUTER JOIN Major ON major=code
```

2.5 Nested Select (IN/NOT IN)

Majors who have no BA students

```
SELECT *
FROM Major
WHERE code NOT IN (
      SELECT major
      FROM student
      WHERE deg_code='BA'
   )
```

2.6 Nested Select (EXISTS)

```
SELECT *
FROM Major
WHERE EXISTS (
      SELECT *
      FROM Student
      WHERE major=code AND
          deg_code='BA'

   )
```

2.7 INTERSECT

Majors with both male and female students. (Notice intersect eliminates duplicates)

```
( SELECT major
  FROM student
  WHERE sex='F'
)
INTERSECT
( SELECT major
  FROM student
  WHERE sex='M'
)
```

3 Query Examples

3.1 Show the names of all students who are majoring in Computer Science

Note: You can't cheat and use the code 'CS', since the fact that CS corresponds to 'Computer Science' is NOT guaranteed to be true (although it is for this example).

3.1.1 Implicit Joins

```
SELECT s.name
FROM Student s, Major m
WHERE s.major=m.code AND
      m.name='Computer Science'
```

3.1.2 Explicit Join

```
SELECT name
FROM (Student JOIN Major ON major=code)
WHERE m_name='Computer Science'
```

3.1.3 Nested Select (IN)

```
SELECT name
FROM Student
WHERE major IN (
      SELECT code
      FROM Major
      WHERE m_name='Computer Science'
  );
```

3.2 Show the names and standings of all students

At last, a non-equijoin :)

3.2.1 Implicit Joins

```
SELECT s.name, g.designation
FROM Student s, Standing g
WHERE s.deg_code=g.deg_code AND
      s.creds BETWEEN g.min_cr AND g.max_cr
```

3.2.2 Explicit Join

```
SELECT name, designation
FROM Student JOIN Standing ON
    Student.deg_code=Standing.deg_code AND
    creds BETWEEN min_cr AND max_cr
```

3.2.3 Nested Queries

Here we cannot do it just with IN or EXISTS, since we are getting information from both tables.

3.3 Show the names of all majors with how many students they have

Note: Include the count for students that don't have any major.

3.3.1 OUTER JOIN

```
SELECT m_name, COUNT(*)
FROM Student LEFT OUTER JOIN Major ON
    major=code
GROUP BY m_name
```

4 Exercises

4.1 Show the names of all students, with the names of their major and the names of their degree

```
SELECT

FROM
```

4.2 Show the names of all degrees, with the number of students in the degree, and the average number of credit hours per student. Degrees with no students should appear with 0 on those columns

```
SELECT

FROM
```

4.3 Do query as above, but show only degrees with 3 or more students enrolled

```
SELECT

FROM
```

4.4 Provide a list of all degrees, with the number of first year students on each degreer

```
SELECT

FROM
```

4.5 Provide SQL code to create a view (called student_standing) containing all info from student table, plus the numeric standing and designation for thir standing

```
SELECT

FROM
```