Interactive Session 1

This Jupyter notebook is provided as a companion to Interactive Session 1, in order to practice what you have learned, and to present some new class material. It will help you to answer quiz questions included in this Interactive Session, and will give you an opportunity to experiment before answering the questions.

Please note that not all queries in the cells of this notebook are supposed to run properly. Some of them will fail and you are expected to find the reason for that.

Initial Steps

Please run the following few cells before we start. They create the required tables and insert some tuples so that we can start experimenting with them.

In [4]:
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [5]:
```
%sql sqlite:///is1.db
%config SqlMagic.displaylimit = None
```

In [6]:
```
%%sql

DROP TABLE IF EXISTS Students;
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
);

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Aadm', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'UBC', 19, 3.5),
        ('1003', 'Alice', 'SFU', 18, 3.7),
        ('1004', 'Bob', 'UBC', 22, 3.1),
        ('1005', 'David', 'SFU', 20, 3.2),
        ('1006', 'John', 'SFU', 21, 3.1),
        ('1007', 'Mary', 'UBC', 21, 3.4),
        ('1008', 'Mike', 'SFU', 24, 3.1),
        ('1009', 'Sarah', 'UBC', 18, 3.0);

SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
Done.
9 rows affected.
Done.
```

Out[6]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0

In [7]:

```sql
%%sql

DROP TABLE IF EXISTS Enrolled;
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
);

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '130','A'),
        ('1002', '125','A+'),
        ('1003', '120','A'),
        ('1003', '125','B'),
        ('1003', '150','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
Done.
8 rows affected.
Done.
```

Out[7]:

stid	cid	grade
1001	200	A
1001	295	A
1001	250	B+
1002	130	A
1002	125	A+
1003	120	A
1003	125	B
1003	150	A

Null Values

Before running each cell, try to analyze if it will succeed or fail, then confirm your answer with running the cell. Try to answer why for each of the scenarios.

In [8]:
```
%%sql
INSERT INTO Students(sid, name, school, age, gpa)
VALUES  ('1010', '', 'SFU', 23, 3.2);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
1 rows affected.
Done.
```

Out[8]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0
1010		SFU	23	3.2

In [9]:
```
%%sql
INSERT INTO Students(sid, name, school, age)
VALUES  ('1011', 'Caleb', 'SFU', 23);
```

```
SELECT * FROM Students;
```

```
 * sqlite:///is1.db
1 rows affected.
Done.
```

Out[9]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0
1010		SFU	23	3.2
1011	Caleb	SFU	23	None

In [10]:
```
%%sql
INSERT INTO Students(sid, school, age, gpa)
VALUES  ('1012', 'SFU', 23, 3.4);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) NOT NULL constraint failed: Students.name
[SQL: INSERT INTO Students(sid, school, age, gpa)
VALUES  ('1012', 'SFU', 23, 3.4);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [11]:
```
%%sql
INSERT INTO Students(sid, name, school, age, gpa)
VALUES  ('1013', 0, 'SFU', 23, 3.4);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
1 rows affected.
Done.
```

Out[11]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0
1010		SFU	23	3.2
1011	Caleb	SFU	23	None
1013	0	SFU	23	3.4

In [12]:
```
%%sql
INSERT INTO Students(sid, name, school, age, gpa)
VALUES  ('1014', NULL, 'SFU', 23, 3.2);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) NOT NULL constraint failed: Students.name
[SQL: INSERT INTO Students(sid, name, school, age, gpa)
VALUES  ('1014', NULL, 'SFU', 23, 3.2);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

In [13]:
```
%%sql
UPDATE Students SET name='Nikita' WHERE sid=1013;

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
1 rows affected.
Done.
```

Out[13]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0
1010		SFU	23	3.2
1011	Caleb	SFU	23	None
1013	Nikita	SFU	23	3.4

In [14]:
```
%%sql
UPDATE Students SET name=NULL WHERE sid=1013;

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) NOT NULL constraint failed: Students.name
[SQL: UPDATE Students SET name=NULL WHERE sid=1013;]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

Key Constraints

What do you think about the next query?

In [15]:
```
%%sql
INSERT INTO Students(sid, name, school, age, gpa)
VALUES  (NULL, 'Adam', 'SFU', 23, 3.2);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
1 rows affected.
Done.
```

Out[15]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0
1010		SFU	23	3.2
1011	Caleb	SFU	23	None
1013	Nikita	SFU	23	3.4
None	Adam	SFU	23	3.2

Did you expect the outcome? Can you explain what happened?

Now, using the queries below, drop the Students and Enrolled tables and create them again using the "WITHOUT ROWID" Option, and try the same command again (given for you again below).

In [16]:

```
%%sql
DROP TABLE Students;
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
) WITHOUT ROWID;

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Aadm', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'UBC', 19, 3.5),
        ('1003', 'Alice', 'SFU', 18, 3.7),
        ('1004', 'Bob', 'UBC', 22, 3.1),
        ('1005', 'David', 'SFU', 20, 3.2),
        ('1006', 'John', 'SFU', 21, 3.1),
        ('1007', 'Mary', 'UBC', 21, 3.4),
        ('1008', 'Mike', 'SFU', 24, 3.1),
        ('1009', 'Sarah', 'UBC', 18, 3.0);
```

```
SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
Done.
9 rows affected.
Done.
```

Out[16]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	3.0

In [17]:

```
%%sql
DROP TABLE Enrolled;

CREATE TABLE Enrolled (
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '130','A'),
        ('1002', '125','A+'),
        ('1003', '120','A'),
        ('1003', '125','B'),
        ('1003', '150','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
Done.
8 rows affected.
Done.
```

Out[17]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	125	A+
1002	130	A
1003	120	A
1003	125	B
1003	150	A

In [18]:
```
%%sql
INSERT INTO Students(sid, name, school, age, gpa)
VALUES  (NULL, 'Adam', 'SFU', 23, 3.2);

SELECT * FROM Students;
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) NOT NULL constraint failed: Students.sid
[SQL: INSERT INTO Students(sid, name, school, age, gpa)
VALUES  (NULL, 'Adam', 'SFU', 23, 3.2);]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

The reason for this observation is that SQLite (for backward compatibility reasons) creates a rowid for each entry in the table and as the rowid is unique, it allows NULL values for members of the primary key. Therefore, if you need to enforce the implied NOT NULL for your primary key, you need to use WITHOUT ROWID when creating your tables.

Using IS

We learned in class that you can use 'IS' to check if a value is 'NULL' or 'NOT NULL'. We can use IS in a few other ways as well, some are only possible in SQLite.

In [19]:
```
%%sql
UPDATE Students SET gpa=NULL WHERE sid=1009;
```

```
 * sqlite:///is1.db
1 rows affected.
```

Out[19]:
```
[]
```

In [20]:
```
%%sql
SELECT * FROM Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[20]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1
1009	Sarah	UBC	18	None

In [21]:
```
%%sql
SELECT * FROM Students WHERE gpa=NULL;
```

```
 * sqlite:///is1.db
Done.
```

Out[21]:

sid	name	school	age	gpa

In [22]:
```
%%sql
SELECT * FROM Students WHERE gpa IS NULL;
```

```
 * sqlite:///is1.db
Done.
```

Out[22]:

sid	name	school	age	gpa
1009	Sarah	UBC	18	None

In [23]:
```
%%sql
SELECT * FROM Students WHERE gpa=NOT NULL;
```

```
 * sqlite:///is1.db
Done.
```

Out[23]:

sid	name	school	age	gpa

In [24]:
```
%%sql
SELECT * FROM Students WHERE gpa IS NOT NULL;
```

```
 * sqlite:///is1.db
Done.
```

Out[24]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2
1002	Aiden	UBC	19	3.5
1003	Alice	SFU	18	3.7
1004	Bob	UBC	22	3.1
1005	David	SFU	20	3.2
1006	John	SFU	21	3.1
1007	Mary	UBC	21	3.4
1008	Mike	SFU	24	3.1

In standard SQL, 'IS' can also be used to compare BOOLEAN values. The same situation holds for SQLight. This is demonstrated below.

In [25]:
```
%%sql
CREATE TABLE WorksByDefaultInSQLite(
    command CHAR(20),
    flag BOOLEAN
);
INSERT INTO WorksByDefaultInSQLite(command, flag)
VALUES  ('IS', TRUE),
        ('RefIntegrity', FALSE);

SELECT * FROM WorksByDefaultInSQLite;
```

```
 * sqlite:///is1.db
Done.
2 rows affected.
Done.
```

Out[25]:

command	flag
IS	1
RefIntegrity	0

In [26]:
```
%%sql
SELECT * FROM WorksByDefaultInSQLite WHERE flag IS TRUE;
```

```
 * sqlite:///is1.db
Done.
```

Out[26]:

command	flag
IS	1

In [27]:
```
%%sql
SELECT * FROM WorksByDefaultInSQLite WHERE flag IS FALSE;
```

```
 * sqlite:///is1.db
Done.
```

Out[27]:

command	flag
RefIntegrity	0

In SQLite, 'IS' can also be used instead of '=' for some other data types (This behavior is not standard SQL).

In [28]:
```
%%sql
SELECT * FROM Students WHERE name IS 'Aadm';
```

```
 * sqlite:///is1.db
Done.
```

Out[28]:

sid	name	school	age	gpa
1001	Aadm	SFU	23	3.2

In [29]:
```
%%sql
SELECT * FROM Students WHERE gpa IS 3.1;
```

```
 * sqlite:///is1.db
Done.
```

Out[29]:

sid	name	school	age	gpa
1004	Bob	UBC	22	3.1
1006	John	SFU	21	3.1
1008	Mike	SFU	24	3.1

Referential Integrity

Remembring our discussions about Referential Integrity and Foriegn Key Constraints, let's try the following experiments.

What is Referential Integrity?

Referential Integrity is a concept in relational databases that ensures the relationships between tables remain consistent. It guarantees that a foreign key in one table (child table) always references a valid, existing primary key in another table (parent table).

In simpler terms:

- It makes sure that the data in related tables stays accurate and reliable.
- If there's a link between two tables, such as a foreign key referencing a primary key, referential integrity ensures that this link is always valid.

but in the following example there is no Referential Integrity as its not turned on

In [30]:
```
%%sql
```

```
DROP TABLE Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[30]: []

In [31]:
```
%%sql
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
) WITHOUT ROWID;

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Adam', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'SFU', 19, 3.1);

SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
2 rows affected.
Done.
```

Out[31]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2
1002	Aiden	SFU	19	3.1

In [32]:
```
%%sql
DROP TABLE Enrolled;
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '250','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
Done.
4 rows affected.
Done.
```

Out[32]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

In [33]:
```
%%sql
DELETE FROM Students WHERE sid='1002';
```

```
 * sqlite:///is1.db
1 rows affected.
```

Out[33]: []

In [34]:
```
%%sql
SELECT * FROM Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[34]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2

In [35]:
```
%%sql
SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[35]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

Did that surprise you? We still have stid 1002 in our Enrolled table! You can even drop the whole table that is referenced (as shown below)!

In [36]:
```
%%sql
DROP TABLE Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[36]: []

In [37]:
```
%%sql
SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[37]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

(The reason you can delete a key that is referenced by table Enrolled from table Students, and even drop table Students while it is referenced by Enrolled is that SQLite does not enforce referential integrity by default. You need to enable enforcing referential integrity using 'PRAGMA foreign_keys=1;'.)

EXPLANATION

Here's a simpler explanation of what is happening in your SQL script:

---

1. What Happened?

- **Students Table**: You created a `Students` table with student information and added two rows.
- **Enrolled Table**: You created an `Enrolled` table, which has a `FOREIGN KEY` linking the `stid` column to the `Students` table's `sid` column.

---

2. Foreign Keys in SQLite

- A **foreign key** ensures that values in one table (like `stid` in the `Enrolled` table) must correspond to values in another table (like `sid` in the `Students` table).
- Normally, **referential integrity** would prevent you from:
  1. Deleting a `sid` from the `Students` table if it is still being referenced by the `Enrolled` table.
  2. Dropping the `Students` table if it is referenced by the `Enrolled` table.

---

3. Why Were You Able to Delete `sid` and Drop `Students`?

- **SQLite does not enforce foreign key rules by default.**
- This means you can delete rows or drop the `Students` table even if it is referenced in the `Enrolled` table, and SQLite won't stop you.

---

4. How to Enable Referential Integrity?

To enforce foreign key rules in SQLite, you need to explicitly enable it. Use this command before creating or modifying tables:

```
PRAGMA foreign_keys = 1;
```

- Once enabled:
  - You won't be able to delete a `sid` from `Students` if it is referenced in `Enrolled`.
  - Dropping the `Students` table will also be blocked if it is still referenced.

What's the Takeaway?

- By default, SQLite allows you to break foreign key rules.
- If you want to enforce these rules, you must enable them using `PRAGMA foreign_keys = 1`.

In [42]:
```
%%sql
PRAGMA foreign_keys=1;
```

```
 * sqlite:///is1.db
Done.
```

Out[42]: []

In [43]:
```
%%sql
DROP TABLE Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[43]: []

In [ ]:
```
%%sql
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
) WITHOUT ROWID;

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Adam', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'SFU', 19, 3.1);

SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
Done.
2 rows affected.
Done.
```

Out[ ]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2
1002	Aiden	SFU	19	3.1

In [46]:
```
%%sql
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '250','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
4 rows affected.
Done.
```

Out[46]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

In [47]:
```
%%sql
DELETE FROM Students WHERE sid='1002';
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) FOREIGN KEY constraint failed
[SQL: DELETE FROM Students WHERE sid='1002';]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

Now, let's see what happens if we use 'ON DELETE CASCADE'?

ON DELETE CASCADE

In [48]:
```
%%sql
DROP TABLE Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[48]: []

In [49]:
```
%%sql
DROP TABLE Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[49]: []

Note that there will be no change in creation of table Students:

In [50]:
```
%%sql
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
) WITHOUT ROWID;

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Adam', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'SFU', 19, 3.1);

SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
2 rows affected.
Done.
```

Out[50]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2
1002	Aiden	SFU	19	3.1

But we include 'ON DELETE CASCADE' in creation of table Enrolled:

Because if i delete anything from the Student table,

it should also be get deleted from the Enrolled table as well

What is `ON DELETE CASCADE`?

`ON DELETE CASCADE` is an instruction used in a **foreign key constraint** that ensures when a referenced row in the parent table (e.g., `Students`) is deleted, all corresponding rows in the child table (e.g., `Enrolled`) are also automatically deleted.

This ensures **referential integrity** between tables and prevents orphaned rows in the child table.

How Does `ON DELETE CASCADE` Work?

- **Parent Table**: The table that contains the primary key (e.g., `Students` with `sid`).
- **Child Table**: The table with the foreign key that references the parent table (e.g., `Enrolled` with `stid`).

When you delete a row in the parent table:

- If `ON DELETE CASCADE` is specified, all rows in the child table that reference the deleted row will also be deleted.
- This happens automatically and keeps the database consistent.

In [51]:
```
%%sql
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
    ON DELETE CASCADE
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '250','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
4 rows affected.
Done.
```

Out[51]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

Now let's see what happens when we delete a row from Students that is referenced by Enrolled:

In [52]:
```
%%sql
SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
```

Out[52]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2
1002	Aiden	SFU	19	3.1

In [53]:
```
%%sql
SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[53]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

In [54]:
```
%%sql
DELETE FROM Students WHERE sid='1002';
```

```
 * sqlite:///is1.db
1 rows affected.
```

Out[54]: []

In [55]:
```
%%sql
SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[55]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A

In [56]:
```
%%sql
SELECT * FROM Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[56]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2

Was it what you expected?

Now let's try and see what happens when we use 'ON DELETE SET NULL'.

Now if i delete anything from the Student table,

instead of getting deleted from the Enrolled table as well,

we are gonna put Null in the Enrolled table(but null will only be in the ID)

When you use `ON DELETE SET NULL`, it **only nullifies the value of the foreign key column (`stid` in the `Enrolled` table)** for the rows where the parent key (`sid` in the `Students` table) matches the condition in your `DELETE` statement.

ON DELETE SET NULL

In [57]:
```
%%sql
DROP TABLE Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[57]:
```
[]
```

In [58]:
```
%%sql
DROP TABLE Students;
```

```
 * sqlite:///is1.db
Done.
```

Out[58]: []

In [59]:
```
%%sql
CREATE TABLE Students (
    sid CHAR(11),
    name CHAR(20) NOT NULL,
    school CHAR(10),
    age INTEGER,
    gpa REAL,
    PRIMARY KEY (sid)
) WITHOUT ROWID;

INSERT INTO students(sid, name, school, age, gpa)
VALUES  ('1001', 'Adam', 'SFU', 23, 3.2),
        ('1002', 'Aiden', 'SFU', 19, 3.1);

SELECT * FROM students;
```

```
 * sqlite:///is1.db
Done.
2 rows affected.
Done.
```

Out[59]:

sid	name	school	age	gpa
1001	Adam	SFU	23	3.2
1002	Aiden	SFU	19	3.1

In [60]:
```
%%sql
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (stid, cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
    ON DELETE SET NULL
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '250','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
4 rows affected.
Done.
```

Out[60]:

stid	cid	grade
1001	200	A
1001	250	B+
1001	295	A
1002	250	A

In [61]:
```
%%sql
DELETE FROM Students WHERE sid='1002';
```

```
 * sqlite:///is1.db
(sqlite3.IntegrityError) NOT NULL constraint failed: Enrolled.stid
[SQL: DELETE FROM Students WHERE sid='1002';]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
```

Now let's see what would have happened if the stid was not part of the 'PARIMARY KEY' of table Enrolled?

In [62]:
```
%%sql
DROP TABLE Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[62]: []

In [63]:
```
%%sql
CREATE TABLE Enrolled(
    stid CHAR(20),
    cid CHAR(20),
    grade CHAR(5),
    PRIMARY KEY (cid),
    FOREIGN KEY (stid) REFERENCES Students(sid)
    ON DELETE SET NULL
)WITHOUT ROWID;

INSERT INTO Enrolled(stid, cid, grade)
VALUES  ('1001', '200','A'),
        ('1001', '295','A'),
        ('1001', '250','B+'),
        ('1002', '260','A');

SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
4 rows affected.
Done.
```

Out[63]:

stid	cid	grade
1001	200	A
1001	250	B+
1002	260	A
1001	295	A

In [64]:
```
%%sql
DELETE FROM Students WHERE sid='1002';
```

```
 * sqlite:///is1.db
1 rows affected.
```

Out[64]: []

In [65]:
```
%%sql
SELECT * FROM Enrolled;
```

```
 * sqlite:///is1.db
Done.
```

Out[65]:

stid	cid	grade
1001	200	A
1001	250	B+
None	260	A
1001	295	A

The reason the deletion in the previous 'ON DELETE SET NULL' example failed was that stid which would be SET NULL as a result of query "DELETE FROM Students WHERE sid='1002';" is part of the 'PRIMARY KEY' in Enrolled table. 'PRIMARY KEY' constraints imply 'NOT NULL' on all attributes that are member of the key.

Clean up Steps

In [ ]:
```
%%sql
DROP TABLE Enrolled;
DROP TABLE Students;
DROP TABLE WorksByDefaultInSQLite;
```

In [ ]:

In [ ]: