# JSON to 3NF

1. Ingest students.json into Snowflake using snowsql


```
{"student":{"ID":1,"Name":"Jill","Age":25},"location":[{"name":"MI","active
":"Y","region":"N"},{"name":"TX","active":"N","region":"S"}],"courses":["Ma
th","Science","English","History"]}
{"student":{"ID":2,"Name":"Mark","Age":21},"location":[{"name":"TX","active
":"Y","region":"S"}],"courses":["English","History"]}
{"student":{"ID":3,"Name":"Tim","Age":18},"location":[{"name":"TX","active"
:"Y","region":"S"},{"name":"FL","active":"N","region":"S"},{"name":"AR","ac
tive":"N","region":"S"}],"courses":["Math"]}
{"student":{"ID":4,"Name":"Jane","Age":20},"location":[{"name":"CA","active
":"N","region":"W"},{"name":"FL","active":"Y","region":"S"}],"courses":["En
glish"]}
{"student":{"ID":7,"Name":"Anne","Age":21},"location":[{"name":"CA","active
":"Y","region":"W"}],"courses":["Math","English","History"]}
```


```
USE ROLE ACCOUNTADMIN;

CREATE OR REPLACE DATABASE SNOWTEST;
CREATE OR REPLACE SCHEMA SNOWTEST.PUBLIC;

CREATE OR REPLACE TABLE students (
  json_column variant
  );

CREATE OR REPLACE FILE FORMAT students_json_format
  TYPE = JSON;

CREATE OR REPLACE TEMPORARY STAGE students_stage
 FILE_FORMAT = students_json_format;

PUT 'file://<file path>/students.json' @students_stage AUTO_COMPRESS=TRUE;

COPY INTO students(json_column)
   FROM (SELECT *
         FROM @students_stage/students.json.gz t)
   ON_ERROR = 'continue';

SELECT * FROM students;
```

**2. Let's create a table with just students and course**

```
SELECT json_column
, json_column:student.Age
, json_column:student.ID
, json_column:student.Name
, json_column:courses
FROM students;
```

This is not in 3NF format due to the array of courses listed in the course column

**3. Create student table**

```
CREATE OR REPLACE TABLE student_roster AS
SELECT json_column:student.ID as ID
, json_column:student.Name as Name
, json_column:student.Age as Age
FROM students;

SELECT * FROM student_roster;
```

**4. Create a table for the course arrays for each student**

```
CREATE OR REPLACE TABLE course AS
SELECT json_column:student.ID as student_id
, json_column:courses as courses
FROM students
;

SELECT * FROM course;
```

**5. Use flatten lateral to unnest the array until individual rows**

```
CREATE OR REPLACE TABLE student_courses AS
SELECT Student_ID --reference ID
, X.VALUE AS course -- flatten value
FROM course,
LATERAL FLATTEN(courses) X
;

SELECT * FROM student_courses;
```

**6. We can also do this directly from the JSON column**

```
CREATE OR REPLACE TABLE student_courses AS
SELECT json_column:student.ID as student_ID
, X.VALUE as course
FROM students,
LATERAL FLATTEN(json_column:courses) X;

SELECT * FROM student_courses;
```

This is still not 3NF since the courses can repeat. We need to create a seperate table for course and general course_ids

**7. Create a dimension table for course**

```
CREATE OR REPLACE TEMPORARY TABLE course AS
SELECT UUID_STRING() as course_ID
, course
FROM (SELECT course FROM student_courses GROUP BY course) z
;
```

**8. Create new mapping table with course ID and student ID**

```
CREATE OR REPLACE TABLE student_course_mapping AS
SELECT a.student_id
, b.course_id
FROM (SELECT json_column:student.ID as student_ID
, X.VALUE as course
FROM students,
LATERAL FLATTEN(json_column:courses) X) a
INNER JOIN course b
ON a.course = b.course;
```

**9. Now we have a 3NF data model for students and courses**

**Test Your Skills**

Create a 3NF data model for the customer.json

Create a data vault model for the animal.json