Databases.

→ permanent storage of memory

Types
- ↳ SQL DB (stores data as table)
- → NoSQL DB.
  - ↳ uses key-value pair
  - → uses DOM like JSON
    - → flexible
    - → scalable
      - horizontal: add more fields.
      - vertical: add more parameters
    - → not very good for large scale projects.

→ SQL → CRUD
- C → create
- R → read
- U → update
- D → delete

→ PostgreSQL

```
import Client from "pg";
const db = new Client({
    User: "username",
    host: "localhost",
    database: "mydatabase",
    password: "password",
    port: 5432,
});

db.connect();
db.query("SELECT * FROM users", (err, res) => {
    if(err) console.log(err.stack);
    else console.log(res.rows);
});
db.end();
});
```

→ syntax:

```
CREATE TABLE <Name of tables>(
    <field1> <DATATYPE>
    <field2> <DATATYPE>
    <field3> <DATATYPE>
);
```

eg CREATE TABLE friends (
    id SERIAL PRIMARY_KEY, ← auto increment after each entry.
    name VARCHAR(50),
    age INT,
    is_cool BOOLEAN
); ← all entries must be unique.

→ 1) select entire table
SELECT * FROM world_food

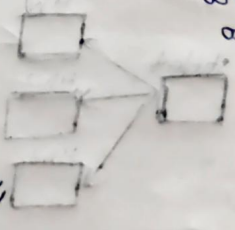
2) select column
SELECT <column_name> FROM world_food

SELECT <column1>, <column2> FROM world_food

3) WHERE keyword
SELECT <column> FROM <table> WHERE <condition>
↳ operators that can be used >, <, >=, <=, ==, !=

4) WHERE LIKE keyword
SELECT <column> FROM <table> WHERE <column> LIKE <pattern>

eg SELECT country FROM where_food WHERE country LIKE 'u' || '%';
↳ starts with 'u'
'%' || 'a';
ends with 'a'



→ Relationship types
1) one to one
2) one to many
3) many to one
4) many to many

eg one to one

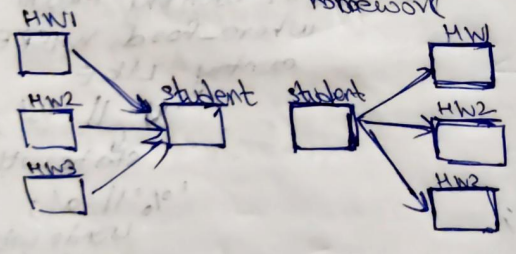
```
CREATE TABLE student(
  id SERIAL PRIMARY KEY,
  first TEXT,
  last TEXT
);
CREATE TABLE contact_detail(
  id INTEGER REFERENCES
        student(id) UNIQUE,
  tel TEXT,
  address TEXT,
);
INSERT INTO student(
        first, last)
  VALUES ('Angelw', 'Yu');

INSERT INTO contact_detail
      (id, tel, address)
  VALUES (1, '+9903836974',
      '123 App Brewery Road');

SELECT * FROM student
  JOIN contact_detail
  ON student.id = contact_detail.
                  id
```

eg one to many.

school → many → each student does many homework




```
CREATE TABLE homework_submission(
  id SERIAL PRIMARY KEY,
  mark INTEGER,
  student_id INTEGER REFERENCES
        student(id)
);
```

eg many to many

```
CREATE TABLE courses (
  id SERIAL PRIMARY KEY,
  title VARCHAR(50)
);
CREATE TABLE enrollment(
  student_id INTEGER
      REFERENCES student(id),
  course_id INTEGER
      REFERENCES courses(id)
);
SELECT * FROM enrollment
  JOIN student ON student.id
      = enrollment.student_id
  JOIN course ON class.id =
      enrollment.course_id
```

→ `ALTER TABLE student RENAME TO user.`

`ALTER TABLE user ALTER COLUMN first TYPE varchar(30);`

`ALTER TABLE contact_detail ADD email TEXT`

→ `DROP TABLE IF EXISTS <table_name>, <table_name> etc`

→ `UPDATE <table_name> SET <column_to_update> = value WHERE condition>`

→ `DELETE FROM <table_name> WHERE <condition> AND condition>`

→ `SELECT * FROM users ORDER BY color ASC;` (ascending order)