- Transaction
- ACID
- Basic SQL
    - filter even
    - none duplecate
    - count
    - order by, asc, desc, limit
    - SQL
        - structured query language
        - connect to database
        - change database
        - create database
        - drop database
        - create table
        - reset query buffer
        - check table schema
        - create table with constraint
        - check settings
        - insert
        - add column to schema
        - remove column from schema
        - change coloumn type
        - change column constraint
        - remove column constraint
        - only show empty table
        - all columns
        - insert data from .sql file
        - select
        - order by
        - distinct
        - where
        - operator
        - offset/limit
        - fetch
        - in
        - between
        - like and ilike
        - having

        - sum max min avg count
        - `+ - * / ^ %`
        - round
        - as
        - coalesce
        - nullif(if equal return null, else return the first value)
        - date
        - interval
        - extract
        - temporary table and age()
        - primary key
        - drop constraint
        - add primary key
        - add unique constraint
        - primary key vs unique
        - check constraint
        - delete
        - update
        - on conflict do nothing
        - upsert (update or insert)
    - Relationships
        - foreign key
        - add relationship
        - inner join on
        - left join on
        - delete with constaint(`cascade` is bad practice)
        - output to .csv
        - bigserial
        - extension
        - list all functions
        - uuid(universal unique identifier)

Transaction

- begin
- commit
    - after begin a transaction, any query failed, when we commit, it will actually do a roll back instead of commit.

- rollback

ACID

- atomicity
- All or nothing
- consistency
  - No constraint violation. Database contains consistent data.
- isolation
  - Users (sessions) don't affect each other.
- durability
  - s Once data is committed, it is permanent.

Basic SQL

filter even

```
select * from tbname
where mod(id, 2) = 2;
```

none duplecate

```
select DISTINCT name from dbname;
```

count

```
select count(city) - count(distinct(city))
from station;
```

order by, asc, desc, limit

```
select city, length(city)
from station
order by length(city) asc, city asc
limit 1;
```

SQL

structured query language

connect to database

```
psql -h localhost -p 5432 -U postgres -d dbname -c "select * from tbname;"
```

change database

```
\c database_name
```

create database

```
create database dbname;
```

drop database

```
drop database dbname;
```

create table

```
`create table tbname (
    id int,
    name varchar(20)
    );
```

reset query buffer

```
\r
```

check table schema

```
\d table_name
```

create table with constraint

```
create table person(
    id bigserial not null primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    age int not null,
    gender varchar(7) not null,
    date_of_birth date not null
)
```

check settings

```
\set
```

insert

```sql
insert into person (
    first_name,
    last_name,
    gender,
    date_of_birth)
values ('Anne', 'Smith', 'FEMALE', date '1988-01-09');
```

add column to schema

```sql
alter table person
add column email varchar(50) not null;
```

remove column from schema

```sql
alter table person
drop column email;
```

change coloumn type

```sql
alter table person
alter column email set data type varchar(100);
```

change column constraint

```sql
alter table person
alter column email set not null;
```

remove column constraint

```sql
alter table person
alter column email drop not null;
```

only show empty table

```
\dt
```

all columns

```
\?
```

insert data from .sql file

1. comannd line to the file location
2. pwd

3. copy the file path
4. `\i /Users/qianggao/Desktop/SQL/filename.sql`

```
\i filename.sql
```

select

```
select * from person;


select first_name, last_name from person;


select email from person;
```

order by

```
select * from person
order by first_name asc;


select * from person
order by first_name desc;


select * from person
order by first_name asc, last_name desc;
```

distinct

```
select distinct country_of_birth from person
order by country_of_birth asc;
```

where

```
select * from person
where country_of_birth = 'China' or country_of_birth = 'USA';
```

operator

```
select * from person
where country_of_birth != 'China';
```

offset/limit

```
select * from person limit 10 offset 10;
```

fetch

```
select * from person
offset 10
fetach 10 rows only;
```

in

```
select * from person
where country_of_birth in ('China', 'USA');
```

between

```
select * from person
where date_of_birth
between date '1988-01-09' and date '1988-01-10';
```

like and ilike

```
select * from person
where email like '%@gmail.com';
```

```
select * from person
where email ilike '_______@%';
```

```
select * from person
where email ilike '%@google%';
```

```
select * from person
where country_of_birth ilike 'china';
```

having

```
select country_of_birth, count(*) from person
group by country_of_birth
having count(*) > 5
order by country_of_birth asc;
```

sum max min avg count

```sql
select max(price) from car;
```

```sql
select sum(price) from car;
```

```sql
select round(avg(price)) from car;
```

```sql
select make, model, max(price) from car
group by make
order by max(price) desc;
```

```sql
select sum(price) from car
where make = 'Ford' and model = 'Fiesta';
```

```sql
select sum(price) from car
group by make;
```

+ - * / ^ %

```sql
select 10 % 3;
```

round

```sql
select round(avg(price), 2);
```

as

```sql
select make, model, max(price) as max_price from car
group by make;
```

coalesce

```sql
select coalesce(email, 'no email');
```

nullif(if equal return null, else return the first value)

```sql
select max(price) / nullif(count(*), 0), make
from car
group by make;
```

date

```
select now()::date;
select now()::time;
select now()::timestamp;
select now()::timestamp with time zone;
select now()::timestamp with time zone at time zone 'UTC';
```

interval

```
select now() - interval '1 day';
select now() - interval '1 day' + interval '1 hour';
select now() - interval '1 day' + interval '1 hour' + interval '1 minute';
select now() - interval '1 day' + interval '1 hour' + interval '1 minute' + interval '1
```

extract

```
select extract(year from now());
select extract(month from now());
select extract(day from now());
select extract(hour from now());
select extract(minute from now());
select extract(second from now());
```

temporary table and age()

```
create temporary table person_temp as select * from person;
alter table person_temp drop column email;
select *, age(now(), date_of_birth) from person_temp;
```

primary key

```
create table person(
    id bigserial not null primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    age int not null,
```

drop constraint

```
alter table person
drop constraint person_pkey;
insert into person (id,first_name, last_name, age)
values (1, 'Anne', 'Smith', 25);
insert into person (id,first_name, last_name, age)
values (1, 'John', 'Smith', 25);
--  without constraint, table will have duplicate rows with the same id
```

add primary key

```
alter table person
add primary key (id);
```

add unique constraint

```
alter table person
add constraint person_unique_email unique (email);


-- equivalent to the above, only different syntax and let sql name the constraint
alter table person
add unique (email);
```

primary key vs unique

- Primary Key is used to uniquely identify a row
- a unique key is used to prevent duplicate values in a column

check constraint

```
alter table person
add constraint person_check_gender
check (gender = 'Female' or gender = 'Male');
```

delete

- delete all

```
delete from person;
```

update

```
update person set email='Edsel@gmail.com' where first_name='Edsel';


update person set
first_name = 'Edsel',
last_name = 'Smith',
where first_name = 'Edsel';
```

on conflict do nothing

```
insert into person (id,first_name, last_name)
values (1, 'Edsel', 'Smith')
on conflict do nothing;
```

upsert (update or insert)

```
insert into person (id,first_name, last_name)
values (1, 'Edsel', 'Smith')
on conflict (id) do update set
first_name = 'Edsel',
last_name = 'Smith';
```

Relationships

foreign key

```
create table person(
    id bigserial not null primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    age int not null,
    country_of_birth varchar(50) not null,
    email varchar(50) not null,
    car_id bigint references car(id) unique(car_id);
);
```

add relationship

```
update person set car_id = 1 where id = 1;
```

inner join on

```
select * from person inner join car on person.car_id = car.id;
```

left join on

```
select * from person left join car on person.car_id = car.id;
```

delete with constaint(`cascade` is bad practice)

```
delete from person where car_id = 1;
delete from car where id = 1;
```

output to .csv

```
\copy
(select * from person left join car on car.id=person.car_id)
to '/Users/qianggao/Desktop/result.csv' delimiter ',' csv header
```

bigserial

```
create table person(
    id bigserial not null primary key,
    first_name varchar(50) not null,
    last_name varchar(50) not null,
    age int not null,
    country_of_birth varchar(50) not null,
    email varchar(50) not null,
    car_id bigserial references car(id) unique(car_id)
);


alter sequence person_id_seq restart with 1;
```

extension

```
select * from pg_available_extensions;


create extension if not exists "uuid-ossp";
```

list all functions

```
\df
```

uuid(universal unique identifier)

```
select uuid_generate_v4();
```