- SQL is a programming language designed to manipulate and manage data stored in relational databases.
- A relational database is a database that organizes information into one or more tables.
- A table is a collection of data organized into rows and columns.
- A statement is a string of characters that the database recognizes as a valid command.

```
-- (选择全部)
SELECT * from celebs;
-- (创建表格)
CREATE TABLE celebs (id INTEGER, name TEXT, age INTEGER);
-- (插入行数据)
INSERT INTO celebs (id, name, age) VALUES (1, 'Justin Bieber', 21);
-- (选择某列)
SELECT name FROM celebs;
-- (更新信息)
UPDATE celebs SET age = 22 WHERE id =1;
-- (添加表列)
ALTER TABLE celebs ADD COLUMN twitter_handle TEXT;
-- (处理缺失值)
DELETE FROM celebs WHERE twitter_handle IS NULL;
```

- **CREATE TABLE** creates a new table.
- **INSERT INTO** adds a new row to a table.
- **SELECT** queries data from a table.
- **UPDATE** edits a row in a table.
- **ALTER TABLE** changes an existing table.
- **DELETE FROM** deletes rows from a table.

```sql
-- (多变量筛选)
SELECT name ,imdb_rating FROM movies;
-- (唯一筛选)
SELECT DISTINCT genre FROM movies;
-- (条件匹配)
SELECT * FROM movies WHERE imdb_rating > 8;
-- (文本部分匹配_单)
SELECT * FROM movies WHERE name LIKE 'Se_en';
-- (文本部分匹配_多)
SELECT * FROM movies WHERE name LIKE 'a%' ;
-- (范围内的数据)
SELECT *FROM movies WHERE name BETWEEN 'A' AND 'J';
-- (与条件选择)
SELECT * FROM movies WHERE year BETWEEN 1990 AND 2000 AND genre = 'comedy';
-- (或条件选择)
SELECT * FROM movies WHERE genre = 'comedy' OR year < 1980;
-- (排序)
SELECT * FROM movies ORDER BY imdb_rating DESC;
-- (选取前几个)
SELECT * FROM movies ORDER BY imdb_rating ASC LIMIT 3;
```

- **SELECT** is the clause you use every time you want to query information from a database.
- **WHERE** is a popular command that lets you filter the results of the query based on conditions that you specify.
- **LIKE and BETWEEN** are special operators that can be used in a WHERE clause
- **AND and OR** are special operators that you can use with WHERE to filter the query on two or more conditions.
- **ORDER BY** lets you sort the results of the query in either ascending or descending order.
- **LIMIT** lets you specify the maximum number of rows that the query will return. This is especially important in large tables that have thousands or even millions of rows.

```sql
-- (计数)
SELECT COUNT(*) FROM fake_apps;
-- (条件计数)
SELECT COUNT(*) FROM fake_apps WHERE price = 0;
-- (分组计数)
SELECT price ,COUNT(*) FROM fake_apps GROUP BY price;
-- (求和)
SELECT SUM(downloads) FROM fake_apps;
-- (最大值)
SELECT MAX(downloads) FROM fake_apps;
-- (最小值)
SELECT MIN(downloads) FROM fake_apps;
-- (平均值)
SELECT AVG(downloads) FROM fake_apps ;
-- (控制小数点)
SELECT price , ROUND(AVG(downloads),2) FROM fake_apps GROUP BY price;
```

- **Aggregate** functions combine multiple rows together to form a single value of more meaningful information.
- **COUNT** takes the name of a column(s) as an argument and counts the number of rows where the value(s) is not NULL.
- **GROUP BY** is a clause used with aggregate functions to combine data from one or more columns.
- **SUM()** takes the column name as an argument and returns the sum of all the values in that column.
- **MAX()** takes the column name as an argument and returns the largest value in that column.
- **MIN()** takes the column name as an argument and returns the smallest value in that column.
- **AVG()** takes a column name as an argument and returns the average value for that column.
- **ROUND()** takes two arguments, a column name and the number of decimal places to round the values in that column.

```
--（创建主键表格）
CREATE TABLE artists(id INTEGER  PRIMARY KEY ,name TEXT);
--（外部健）
SELECT * FROM albums WHERE artist_id =3;
--（合并表格）
SELECT albums.name , albums.year , artists.name  FROM albums , artists ;
--（JOIN合并）
SELECT * FROM albums JOIN artists ON albums.artist_id = artists.id;
--（左合并）
SELECT * FROM albums LEFT JOIN artists ON albums.artist_id = artists.id ;
--（重命名）
SELECT albums.name AS 'Album' , albums.year , artists.name AS 'Artist'  FROM
albums  JOIN artists ON albums.artist_id = artists.id  WHERE albums.year >
1980;
```

- **Primary Key** is a column that serves a unique identifier for row in the table. Values in this column must be unique and cannot be NULL.
- **Foreign Key** is a column that contains the primary key to another table in the database. It is used to identify a particular row in the referenced table.
- **Joins** are used in SQL to combine data from multiple tables.
- **INNER JOIN** will combine rows from different tables if the join condition is true.
- **LEFT OUTER JOIN** will return every row in theleft table, and if the join condition is not met,NULL values are used to fill in the columns from the right table.
- **AS** is a keyword in SQL that allows you to rename a column or table in the result set using an alias.**