1. More complex queries with AND/OR

1.1 Insert data within specific columns

We can specify the column names in parentheses after the table name. After that, we don't have to specify ever column value, what we only have to specify the ones that we give in the list of column names.

With **AUTOINCREMENT** keyword of **PRIMARY KEY**, we automatically increment values of primary key column, not fill with NULL.

```
CREATE TABLE exercise_logs
    (id INTEGER PRIMARY KEY AUTOINCREMENT,
    type TEXT,
    minutes INTEGER,
    calories INTEGER,
    heart_rate INTEGER);
```

So we can leave out the id let the database fill the values for me.

1.2 Retrieve the records in multiple conditions

We can retrieve records fitted in multiple conditions using **AND** and **OR** operator. The AND operator returns only rows in multiple conditions, while the OR operator returns any rows in multiple conditions.

```
/* AND */
SELECT * FROM exercise_logs WHERE calories > 50 AND minutes < 30;

/* OR */
SELECT * FROM exercise_logs WHERE calories > 50 OR heart_rate > 100;
```

2. Querying IN subqueries

2.1 Retrieve the records without using OR

The **IN** operator will check to see if a particular value is in a list of values. The workflow of using IN operator is same as follows :

1. Put targets in a parenthesis
2. Seperate each of these strinigs with a commans

```
/* IN */
SELECT * FROM exercise_logs WHERE type IN ("biking", "hiking", "tree climbing", "rowing");
```

We can retrieve inverse query using **NOT IN** operators.

```
/* NOT IN */
SELECT * FROM exercise_logs WHERE type NOT IN ("biking", "hiking", "tree climbing", "rowing");
```

2.2 Retreive the records from subquery

If we want retreive the records in multiple conditions from other table, we can make query to be up to date using IN operator directly with the results of a SQL query, SELECT query. We call this **Inner Query** or **Subquery**.

```
/* Subquery */
SELECT * FROM exercise_logs WHERE type IN (
    SELECT type FROM drs_favorites
);
```

2.3 Retreive the records with inexact match

So there are times when we want to do an inexact match. And we can do that with **LIKE** operator, which is a pretty neat operator.

```
/* LIKE */

SELECT * FROM exercise_logs WHERE type IN (
    SELECT type FROM drs_favorites WHERE reason LIKE "cardiovascular"
);
```

3. Restricting grouped results with HAVING

3.1 Giving a column name a new name

When we use aggregat function, the column shows up as 'FUNC(column)' in the results. If we want, we can actually tell SQL to give that column a new name, just by writing **AS** and then giving it the new name.

```
SELECT type, SUM(calories) AS total_calories FROM exercise_logs GROUP BY type;
```

3.2 HAVING cluase

When we use **HAVING**, we're applying the conditions to the grouped values, not the individual values in the individual rows.

```
SELECT type, AVG(calories) AS avg_calories FROM exercise_logs
 GROUP BY type
HAVING avg_calories > 70;
```

4. Calculating results with CASE

The **CASE** expression goes through conditions and returns a value when the first condition is met. So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE cluase.

```
/* CASE */
SELECT type, heart_rate,
    CASE
        WHEN heart_rate > 220-30 THEN "above max"
        WHEN heart_rate > ROUND(0.90 * (220-30)) THEN "above target"
        WHEN heart_rate > ROUND(0.50 * (220-30)) THEN "within target"
        ELSE "below target"
    END as "hr_zone"
FROM exercise_logs;
```

5. Project : Data dig

```
/* Put your data in here and query it! */

/* Get max, average, min value of Space_Flight_hr */
SELECT MAX(Space_Flight_hr) AS max_flights,
       AVG(Space_Flight_hr) AS avg_flights,
       MIN(Space_Flight_hr) AS min_flights
  FROM astronauts;

SELECT COUNT(*), status FROM astronauts
 GROUP BY status
HAVING status='Active';

/* Categorize value of Space_Flight_hr into ('high', 'semi-high', 'average', 'semi-low') */
/* Extract its count where Gender is Male or Year is bigger than 2000 */
SELECT CASE
            WHEN Space_Flight_hr > 5000 THEN 'high'
            WHEN Space_Flight_hr > 2500 THEN 'semi-high'
            WHEN Space_Flight_hr > 900 THEN 'average'
            WHEN Space_Flight_hr > 100 THEN 'semi-low'
            ELSE 'low'
        END AS flight_level, COUNT(*)
    FROM astronauts
 WHERE Gender='Male' OR Year > 2000
 GROUP BY flight_level;
```