# The Basics

## Simple SELECTs

1) Query all the data in the `pets` table.

SELECT \*

FROM pets

2) Query only the first 5 rows of the `pets` table.

SELECT \*

FROM pets

LIMIT 5

3) Query only the names and ages of the pets in the `pets` table.

SELECT name, age FROM pets;

4) Query the pets in the `pets` table, sorted youngest to oldest.

SELECT \* FROM pets

ORDER BY age;

5) Query the pets in the `pets` table alphabetically.

SELECT name FROM pets

ORDER BY name;

6) Query all the male pets in the `pets` table.

SELECT name, sex FROM pets

WHERE sex = 'M';

7) Query all the cats in the `pets` table.

SELECT \* FROM pets

WHERE species = 'cat';

8) Query all the pets in the `pets` table that are at least 5 years old.

SELECT \* FROM pets

WHERE age >= 5;

9) Query all the male dogs in the `pets` table. Do not include the sex or species column, since you already know them.

SELECT name, age FROM pets

WHERE sex = 'M' AND species = 'dog';

10) Get all the names of the dogs in the `pets` table that are younger than 5 years old.

SELECT name FROM pets

WHERE species = 'dog' and AGE < 5;

11) Query all the pets in the `pets` table that are either male dogs or female cats.

SELECT \* FROM pets

WHERE species = 'dog' AND sex = 'M'

OR species = 'cat' AND sex = 'F';

12) Query the five oldest pets in the `pets` table.

SELECT \* FROM pets

WHERE age > 6;

13) Get the names and ages of all the female cats in the `pets` table sorted by age, descending.

SELECT name, age FROM pets

WHERE sex = 'F'

ORDER BY age DESC;

14) Get all pets from `pets` whose names start with P.

SELECT \* FROM pets

WHERE name LIKE 'P%';

15) Select all employees from `employees\_null` where the salary is missing.

SELECT \* FROM employees\_null

WHERE salary is NULL;

16) Select all employees from `employees\_null` where the salary is below $35,000 or missing.

SELECT \* FROM employees\_null

WHERE salary IS NULL OR salary < 35000;

17) Select all employees from `employees\_null` where the job title is missing. What do you see?

SELECT \* FROM employees\_null

WHERE job is NULL;

--Just one entry that was likely a mistake as the only information is the id.

18) Who is the newest employee in `employees`? The most senior?

Newest = Roger Conner, Sales, id 19

SELECT \* FROM employees

ORDER BY startdate DESC;

Most senior = Mary Nash, sales, id 32

SELECT \* FROM employees

--ORDER BY startdate;

19) Select all employees from `employees` named Thomas.

SELECT \* FROM employees

WHERE firstname = 'Thomas';

20) Select all employees from `employees` named Thomas or Shannon.

SELECT \* FROM employees

WHERE firstname = 'Thomas' OR firstname = 'Shannon';

21) Select all employees from `employees` named Robert, Lisa, or any name that begins with a J. In addition, only show employees who are \_not\_ in sales. This will be a little bit of a longer query.

\* \_Hint:\_ There will only be 6 rows in the result.

SELECT \* FROM employees

WHERE job != 'Sales' AND (firstname LIKE 'J%' OR firstname = 'Robert' OR firstname = 'Lisa');

## Column Operations

22) Query the top 5 rows of the `employees` table to get a glimpse of these new data.

SELECT \* FROM employees

LIMIT 5;

23) Query the `employees` table, but convert their salaries to Euros.

\* \_Hint:\_ 1 Euro = 1.1 USD.

\* \_Hint2:\_ If you think the output is ugly, try out the `ROUND()` function.

SELECT \*, ROUND(salary / 1.1 )FROM employees;

24) Repeat the previous problem, but rename the column `salary\_eu`.

SELECT \*, ROUND(salary / 1.1 ) AS salary\_eu FROM employees;

25) Query the `employees` table, but combine the `firstname` and `lastname` columns to be "Firstname, Lastname" format. Call this column `fullname`. For example, the first row should contain `Thompson, Christine` as `fullname`. Also, display the rounded `salary\_eu` instead of `salary`.

\* \_Hint:\_ The string concatenation operator is `||`

SELECT id, job, startdate, salary CONCAT(firstname, ' ', lastname) AS fullname,

ROUND(salary / 1.1 ) AS salary\_eu FROM employees;

26) Query the `employees` table, but replace `startdate` with `startyear` using the `SUBSTR()` function. Also include `fullname` and `salary\_eu`.

SELECT CONCAT(firstname, ' ', lastname) AS fullname,

ROUND(salary / 1.1 ) AS salary\_eu, SUBSTR(CAST(startdate AS text), 1, 4) AS start\_year

FROM employees;

27) Repeat the above problem, but instead of using `SUBSTR()`, use `STRFTIME()`.

SELECT CONCAT(firstname, ' ', lastname) AS fullname,

ROUND(salary / 1.1 ) AS salary\_eu, EXTRACT(year FROM startdate) AS start\_year

FROM employees;

28) Query the `employees` table, replacing `firstname`/`lastname` with `fullname` and `startdate` with `startyear`. Print out the salary in USD again, except format it with a dollar sign, comma separators, and no decimal. For example, the first row should read `$123,696`. This column should still be named `salary`.

\* \_Hint:\_ Check out SQLite's `printf` function.

\* \_Hint2:\_ The format string you'll need is `$%,.2d`. You should read more about such formatting strings as they're useful in Python, too!

SELECT CONCAT(firstname, ' ', lastname) AS fullname,

EXTRACT(year FROM startdate) AS start\_year, to\_char(salary, 'fm$999,999,999') AS salary

FROM employees;

\*\*Note:\*\* For the next few problems, you'll probably want to use `CASE`/`WHEN` statements.

29) Last year, only salespeople were eligible for bonuses. Create a column `bonus` that is "Yes" if you're eligible for a bonus, otherwise "No".

SELECT \*, CASE WHEN job = 'sales' THEN 'Yes' ELSE 'No' END AS Bonus

FROM employees;

30) This year, only sales people with a salary of $100,000 or higher are eligible for bonuses. Create a `bonus` column like in the last problem for salespeople with salaries at least $100,000.

SELECT \*, CASE WHEN job = 'sales' AND salary > 100000 THEN 'Yes' ELSE 'No' END AS Bonus

FROM employees;

31) Next year, the bonus structure will be a little more complicated. You'll create a `target\_comp` column which represents an employee's target total compensation after their bonus. Here is the company's bonus structure:

\* Salespeople who make more than $100,000 will be eligible for a 10% bonus.

\* Salespeople who make less than $100,000 will be eligible for a 5% bonus.

\* Administrators will also be eligible for a 5% bonus.

\* Anyone who does not meet any of the above descriptions is not eligible for a bonus.

Create this `target\_comp` column, making sure to format \_both\_ the `salary` and `target\_comp` columns nicely (ie, with dollar signs and comma separators).

SELECT \*, to\_char(salary, 'fm$999,999,999') AS salary,

to\_char (CASE WHEN job = 'Sales' THEN CASE WHEN salary >= 100000

THEN salary \* 1.1 ELSE salary \* 1.05 END

WHEN job = 'Administrator' THEN salary \* 1.05 ELSE 0 END, 'fm$999,999,999') AS target\_comp

FROM employees;