# Week 07

## SQL

Johnny Chan

# Previously ...

- Normalisation
  - Eliminate and control data redundancy
  - 1NF, 2NF and 3NF
  - Practice with business documents
- Physical data modelling

# Agenda

- Revisit and continue on the basics of SQL
  - CREATE, ALTER and DROP TABLE
  - INSERT, SELECT, UPDATE and DELETE (CRUD)
  - Operator
  - Sorting
  - Single-row function
- Get familiar with SQLite

# SQL

- What is structured query language (SQL)?
  - Industrial standard (ANSI and ISO)
  - 4GL
  - 🤔 How does SQL relate to Codd's relational model?
- SQL statement
  - DDL vs DML
  - Case insensitive
  - White space and carriage return are ignored
  - Single quote is used for string and date literal
  - The semi-colon

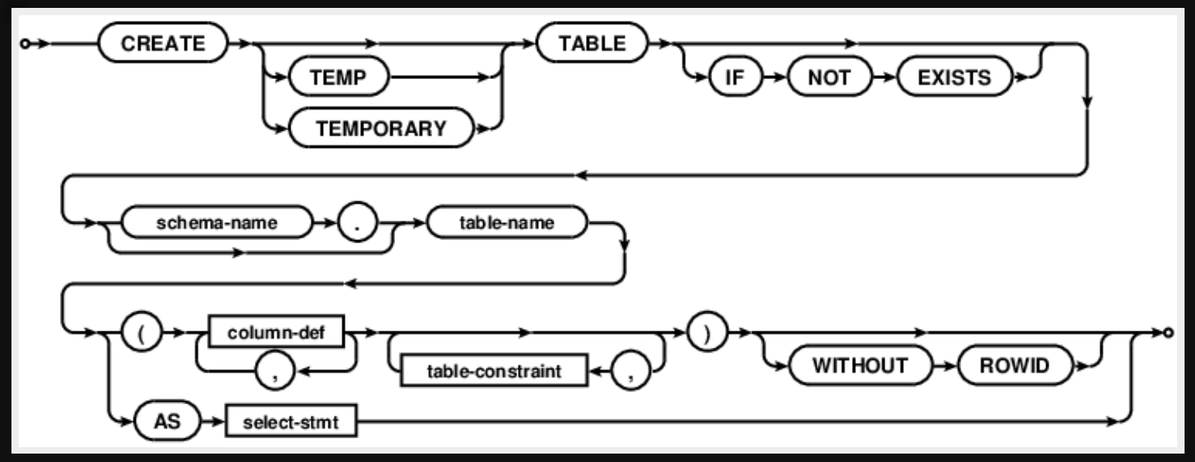
# SQLite

- SQLite is an open source cross-platform embedded relational database management system
  - SQLite command vs SQL statement in SQLite
- All SQL related content and assessment in this course are expected to be run with SQLite
- 🤔 What is the major difference between SQLite and other RDBMS product?

# Demo

## The book database

# CREATE TABLE




```
CREATE TABLE Author
(authorNo INTEGER PRIMARY KEY NOT NULL,
 authorFirstName TEXT,
 authorLastName TEXT,
 authorStreet TEXT,
 authorSuburb TEXT
);
```

- Further: ROWID and AUTOINCREMENT in SQLite

# Naming convention

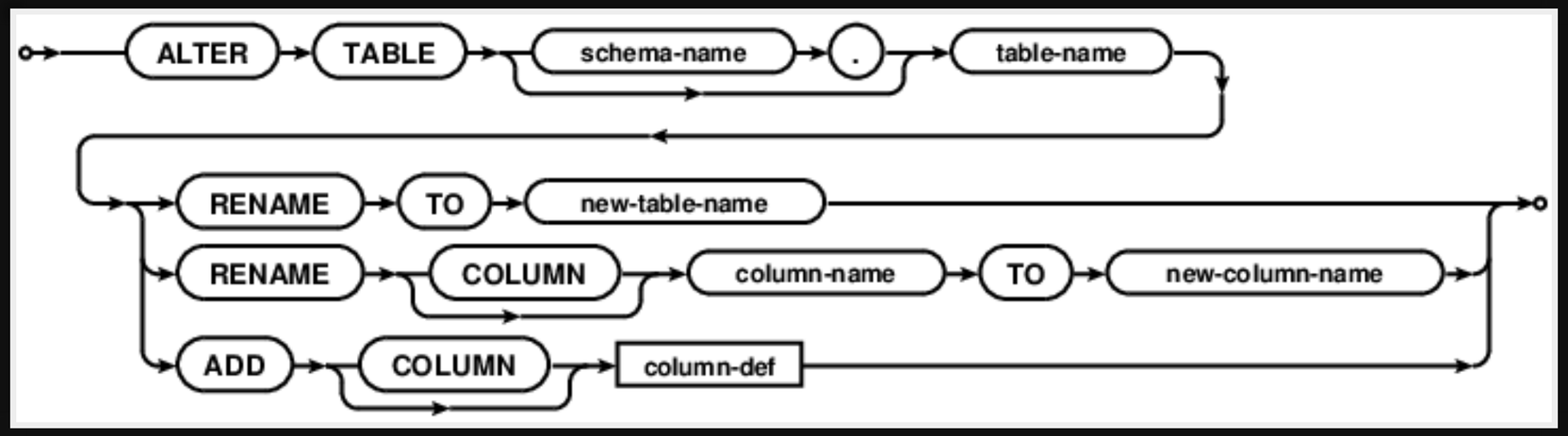
- Must begin with a letter
- Can be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another database object of the same user
- Must not be a reserved word

# CREATE TABLE AS SELECT

- The following statement creates and populates a table based on the result of a SELECT statement: the table has no defined PRIMARY KEY, no defined constraint, and the default value of each column is NULL
- Use this command only for temporary or testing purpose

```
CREATE TABLE AuthorTemp AS
SELECT * FROM Author
WHERE authorSuburb = 'Meadowbank';
```

# ALTER TABLE

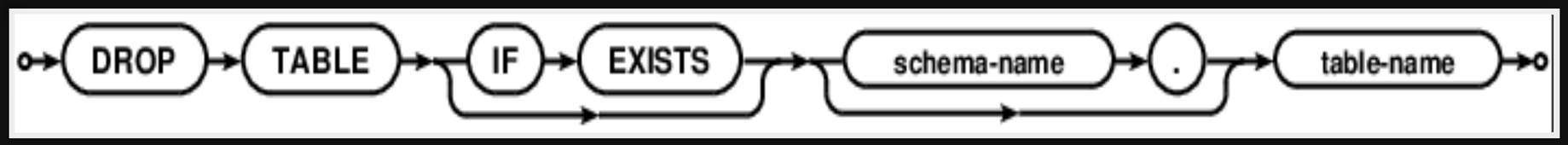



- SQLite only supports two forms of alteration to an existing table in the database: to rename the table or to add a new column to the table

```
ALTER TABLE Author ADD authorCity TEXT;
ALTER TABLE AuthorTemp RENAME TO MeadowbankAuthor;
```

- 🤔 Why is it not a good idea to support removable of a column too?

# DROP TABLE

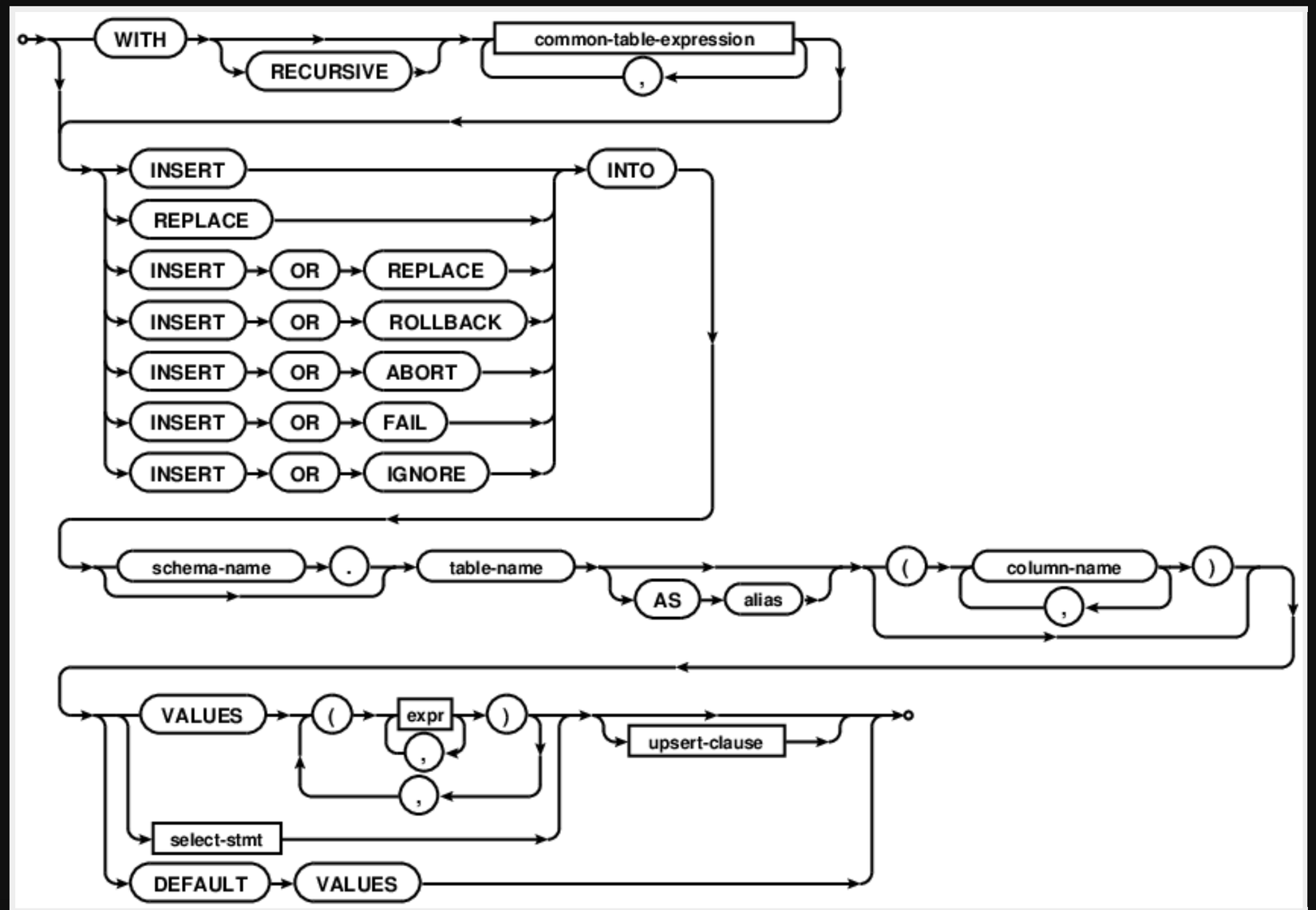



- All the data, structure, constraint and index associated with the table are deleted completely and permanently

```
DROP TABLE MeadowbankAuthor;
```

- 🤔 What would happen if foreign key constraint is involved in a table drop?

# INSERT


WITH
RECURSIVE
common-table-expression
,
INSERT
REPLACE
INSERT
OR
REPLACE
INSERT
OR
ROLLBACK
INSERT
OR
ABORT
INSERT
OR
FAIL
INSERT
OR
IGNORE
INTO
schema-name
.
table-name
AS
alias
(
column-name
,
)
VALUES
(
expr
,
)
,
upsert-clause
select-stmt
DEFAULT
VALUES

# INSERT INTO VALUES

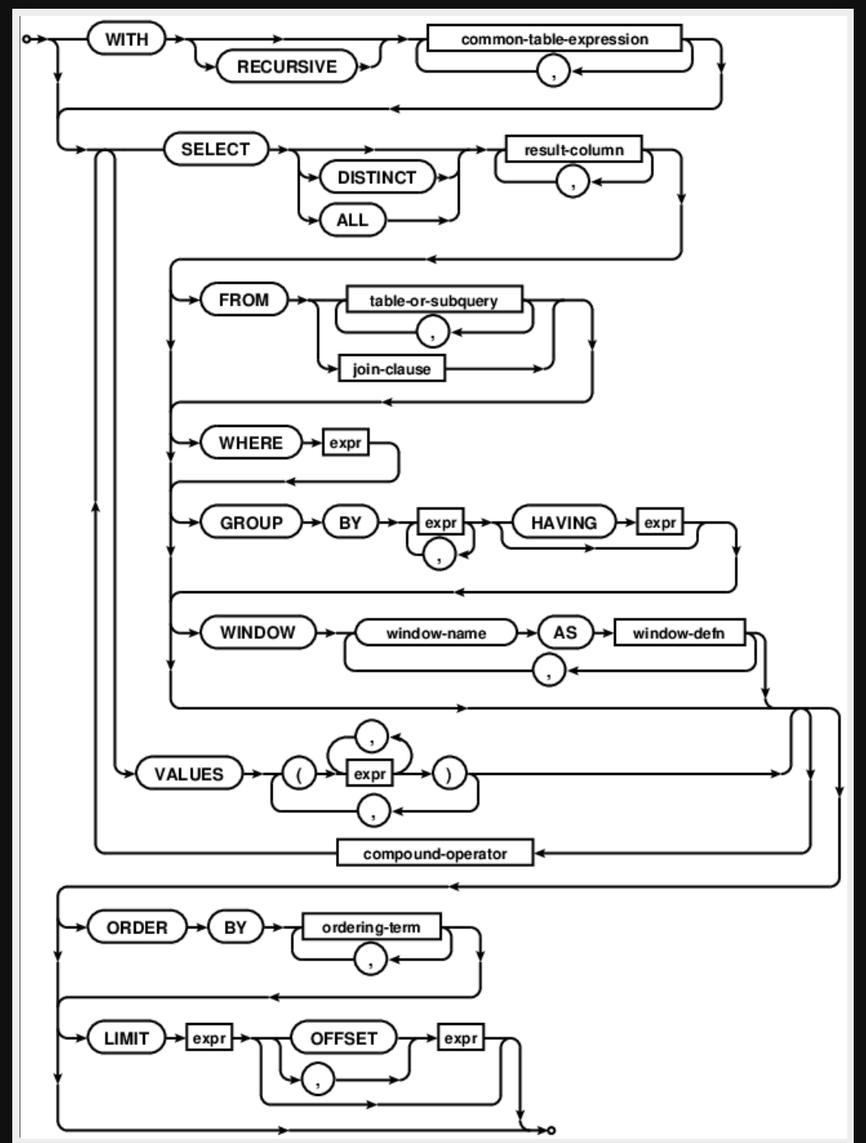
```
INSERT INTO Author (authorFirstName, authorLastName,
authorStreet, authorSuburb, authorCity) VALUES
('De Silva','Clarice', '21 Park View Street',
'Park View', 'Auckland');

INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
INSERT INTO Author (authorFirstName, authorLastName, authorSt
```

- 🤔 What is the difference between including and not including named columns in the INSERT statement?
- 🤔 What does such difference imply?

# INSERT INTO SELECT

```
INSERT INTO AuthorTemp
SELECT authorFirstName, authorLastName, authorStreet,
authorSuburb, authorCity
FROM Author;
```

# SELECT


WITH
RECURSIVE
common-table-expression
,
SELECT
DISTINCT
ALL
result-column
,
FROM
table-or-subquery
,
join-clause
WHERE
expr
GROUP
BY
expr
,
HAVING
expr
WINDOW
window-name
AS
window-defn
,
VALUES
(
expr
,
)
,
compound-operator
ORDER
BY
ordering-term
,
LIMIT
expr
OFFSET
,
expr

# Quiz 01

- Compose a simplest and valid SELECT statement in SQLite

```
SELECT ...
```

# SELECT all columns

```
SELECT *
FROM Book;

bookCode    bookTitle           bookType    paperback
----------  ------------------  ----------  ----------
110         Far from the Crowd  PSY         Y
111         A Loud Game         FIC         Y
112         The Artist          FIC         Y
113         Passage to Freedom  HOR         N
114         Tornado             MYS         Y
115         Knockdown           MYS         Y
116         Judo                HOR         Y
117         Sneaky Stories                  Y
999         Pilgrim's Progress              N
```

- The columns selected are collectively known as a projection

# SELECT specific columns

```
SELECT bookCode 'Book Code', bookTitle Title
FROM Book;

Book Code   Title
----------  ------------------
110         Far from the Crowd
111         A Loud Game
112         The Artist
113         Passage to Freedom
114         Tornado
115         Knockdown
116         Judo
117         Sneaky Stories
999         Pilgrim's Progress
```

- Column alias renames a column heading; useful for derived column

# SELECT DISTINCT

```
SELECT DISTINCT staffCity City
FROM Staff;

City
----------
Hamilton
Auckland
Nelson
Dunedin
Westland
Manukau
```

- Duplicate rows are detected and removed from the set of result rows
- In SQLIte, two NULL values are considered to be equal (but also unique)

# LIMIT and OFFSET clauses

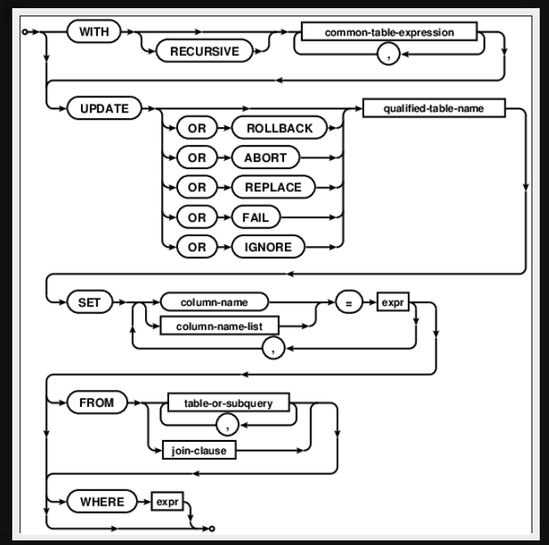
```
SELECT * FROM Book LIMIT 3;

bookCode    bookTitle           bookType    paperback
----------  ------------------  ----------  ----------
110         Far from the Crowd  PSY         Y
111         A Loud Game         FIC         Y
112         The Artist          FIC         Y

SELECT * FROM Book LIMIT 3 OFFSET 2;

bookCode    bookTitle   bookType    paperback
----------  ----------  ----------  ----------
112         The Artist  FIC         Y
113         Passage to  HOR         N
114         Tornado     MYS         Y
```

# WHERE clause

```
SELECT roleID, branchNo, staffCode
FROM StaffAssignment
WHERE roleID = 1;

roleID      branchNo    staffCode
----------  ----------  ----------
1           1           1
1           2           2
1           3           3
1           4           4
```

- When a WHERE clause is specified, each row from the input data is evaluated as a boolean expression
  - Rows evaluated as true are included
  - Rows evaluated as false (or NULL) are excluded
- Commonly used in SELECT, UPDATE and DELETE statements

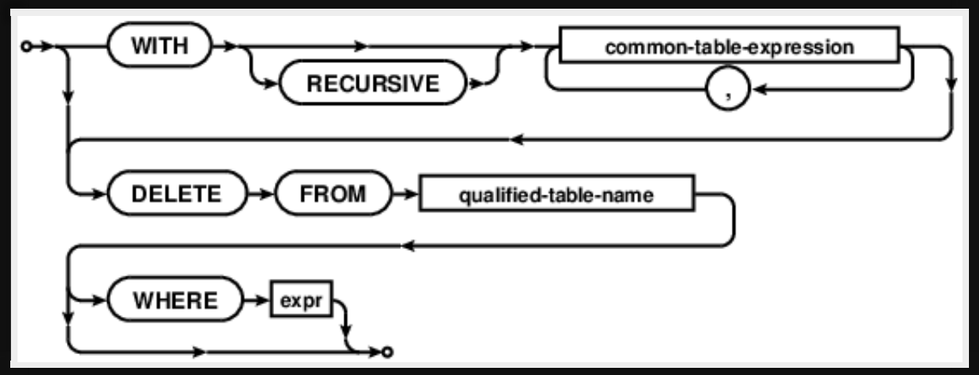
# UPDATE




```
UPDATE Author
SET authorStreet = '22 Park View Street'
WHERE authorFirstName = 'De Silva';
```

- 📢 Always use a WHERE clause in an UPDATE statement!

# DELETE




```
DELETE
FROM AuthorTemp
WHERE authorFirstName = 'De Silva';
```

- 📢 Always use a WHERE clause in a DELETE statement!

# Literal

| Type | Example |
|---|---|
| Integer | 1000 |
| Real | 1000.0 |
| String | '1000', 'Johnny is the king' |
| Date | '2016-09-12' |
| NULL | NULL |

- A literal represents a constant value, which could be an integer, real number, string, date, BLOB or NULL
  - String literal is case sensitive
  - Date literal is format sensitive (YYYY-MM-DD)
  - BLOB literal is outside the scope of this course
- There are specific functions designed to work with literal

# Operator

- SQLite supports the following operator, in order of precedence:

```
1. ||
2. * / %
3. + -
4. < <= > >= BETWEEN
5. = == != <> IS IN LIKE
6. NOT
7. AND
8. OR
```

# Concatenation

```
SELECT authorFirstName || ' ' || authorLastName Name
FROM Author;

Name
----------------
De Silva Clarice
Stevens Rob
Peiris Louis
Parker Clive
Mendis Theodora
Bingly Lisa
St.Louis Gabriel
Coorey Beatrice
Koorey Beatrice
```

- 🤔 How many string literals are being concatenated per row?

# Quiz 02

- Rewrite the previous SELECT statement so that the last name goes before the first name, separated with a comma in between

# Arithmetic

```
SELECT staffCode, salary, salary/12 monthly
FROM StaffAssignment;

staffCode   salary      monthly
----------  ----------  ----------
1           72000.0     6000.0
2           64000.0     5333.33333
3           45000.0     3750.0
4           54000.0     4500.0
5           48000.0     4000.0
6           35000.0     2916.66666
7           40000.0     3333.33333
8           40000.0     3333.33333
9           45000.0     3750.0
...
```

# Quiz 03

- Rewrite the previous SELECT statement with an additional column named tax which shows the annual income tax of staff

# Logical

| Operator | Description |
|---|---|
| AND | Returns TRUE if both conditions are TRUE |
| OR | Returns TRUE if either condition is TRUE |
| NOT | Returns TRUE if the condition is FALSE |

```
SELECT staffCode, salary FROM StaffAssignment
WHERE salary IS NOT 72000
AND salary > 50000
OR salary = 45000;

staffCode   salary
----------  ----------
2           64000.0
3           45000.0
4           54000.0
9           45000.0
11          45000.0
```

# Quiz 04

- Explain how the logical operators work in the previous SELECT statement; and what may happen if the order of AND and OR is swapped

# Comparison

```
SELECT staffCode, salary
FROM StaffAssignment
WHERE salary BETWEEN 54000 AND 72000;

SELECT staffCode, salary
FROM StaffAssignment
WHERE salary >= 54000
AND salary <= 72000;

staffCode   salary
----------  ----------
1           72000.0
2           64000.0
4           54000.0
```

# IN

```
SELECT staffCode, salary
FROM StaffAssignment
WHERE salary IN (54000, 72000);

SELECT staffCode, salary
FROM StaffAssignment
WHERE salary NOT IN (35000, 40000, 45000,
48000, 50000, 64000);

staffCode   salary
----------  ----------
1           72000.0
4           54000.0
```

# IS

```
SELECT *
FROM Book
WHERE bookType IS NULL;

bookCode    bookTitle       bookType    paperback
----------  --------------  ----------  ----------
117         Sneaky Stories              Y
999         Pilgrim's Prog              N
```

- In SQLite, IS operator is equivalent to = and ==
- Similarly, IS NOT is equivalent to <> and =!

# LIKE

```
SELECT authorFirstName
FROM Author
WHERE authorFirstName LIKE 'P%';

authorFirstName
---------------
Peiris
Parker
```

- The LIKE operator performs pattern matching comparison with wildcard symbols:
  - % denotes zero or many characters
  - _ denotes one character

# Quiz 05

- Rewrite the previous SELECT statement to show all authors with a first name that does not start with letter P
- Modify the SELECT statement to show all authors with a first name that is exactly 6 characters long

# Sorting

```
SELECT authorLastName, authorFirstName FROM Author
ORDER BY authorLastName DESC, authorFirstName;

authorLastName  authorFirstName
--------------  ---------------
Theodora        Mendis
Rob             Stevens
Louis           Peiris
Lisa            Bingly
Gabriella       St.Louis
Clive           Parker
Clarice         De Silva
Beatrice        Coorey
Beatrice        Koorey
```

- In SQL, the ORDER BY clause is used to sort rows selected
  - ASC stands for ascending order, the default mode
  - DESC stands for descending order
- The ORDER BY clause comes last in a SELECT statement

# ORDER BY clause

```
SELECT staffCode, salary/12 monthly
FROM StaffAssignment
ORDER BY monthly DESC, roleID DESC;

staffCode   monthly
----------  ----------
1           6000.0
2           5333.33333
4           4500.0
10          4166.66666
12          4166.66666
5           4000.0
9           3750.0
11          3750.0
...
```

- It is allowed to sort by a non-projected column
- It is allowed to sort by a derived column

# Single-row function

- Manipulate data item
- Accept argument and return one value
- Act on each row returned
- Return one result per row
- Can be nested

# String function

| Function | Description |
|---|---|
| LOWER(X) | It returns a copy of string X with all ASCII characters converted to lower case |
| UPPER(X) | It returns a copy of string X with all ASCII characters converted to upper case |
| SUBSTR(X,Y) or SUBSTR(X,Y,Z) | It returns a substring of string X that begins with the Y-th character and which is Z characters long |
| LENGTH(X) | It returns the number of characters in string X |
| INSTR(X,Y) | It finds the first occurrence of string Y within string X and returns the number of prior characters plus 1, or 0 if Y is nowhere found within X |
| REPLACE(X,Y,Z) | It returns a string formed by substituting string Z for every occurrence of string Y in string X |
| RTRIM(X) or RTRIM(X,Y) | It returns a string formed by removing any and all characters that appear in Y (or white spaces if Y is not provided) from the right side of X |
| LTRIM(X) or LTRIM(X,Y) | It returns a string formed by removing any and all characters that appear in Y (or white spaces if Y is not provided) from the left side of X |
| TRIM(X) or TRIM(X,Y) | It returns a string formed by removing any and all characters that appear in Y (or white spaces if Y is not provided) from both ends of X |

# Quiz 06

- What would be returned from the following statements?

```
SELECT LOWER('University of Auckland');
SELECT UPPER('University of Auckland');
SELECT SUBSTR('University of Auckland', 4, 7);
SELECT INSTR('University of Auckland', ' ');
SELECT LENGTH('University of Auckland');
SELECT REPLACE('University of Auckland','Auckland','Otago');
SELECT RTRIM('University of Auckland', 'U');
SELECT LTRIM('University of Auckland', 'U');
SELECT TRIM('University of Auckland', 'U');
```

# Quiz 07

- String literal is case sensitive in SQL. If your task is to look up the address of a particular staff member (e.g. James McDonald) without knowing for sure how names are stored (i.e. whether they are in upper, lower or mixed cases), what would be the optimised way to compose your SELECT statement?

```sql
SELECT staffStreet, staffSuburb, staffCity
FROM Staff
WHERE staffFirstName = 'James'
AND staffLastName = 'McDonald';
```

# Numeric function

| Function | Description |
|---|---|
| ABS(X) | It returns the absolute value of number X |
| MAX(X,Y,...) | It returns the maximum value among the list of numeric arguments |
| MIN(X,Y,...) | It returns the minimum value among the list of numeric arguments |
| ROUND(X) or ROUND(X,Y) | It returns a floating-point value X rounded to Y digits to the right of the decimal point. If the Y argument is omitted, it is assumed to be 0 |
| RANDOM() | It returns a pseudo-random integer between -9223372036854775808 and +9223372036854775807 |

# Date function

| Function | Description |
|---|---|
| DATE(X,Y,...) | It returns the date in this format: YYYY-MM-DD |
| TIME(X,Y,...) | It returns the date in this format: HH:MM:SS |
| DATETIME(X,Y,...) | It returns the date and time in this format: YYYY-MM-DD HH:MM:SS |
| JULIANDAY(X,Y,...) | It returns the Julian day - the number of days since noon in Greenwich on November 24, 4714 B.C. |
| STRFTIME(F,X,Y,...) | It returns the date formatted according to the format string F specified as the first argument. The DATE(), TIME() and DATETIME() functions are equivalent to STRFTIME() with certain formats |

Further: Date function in SQLite

# Example

- How many weeks have all the staff been employed?

```
SELECT staffCode, ROUND((JULIANDAY('now', 'localtime') -
JULIANDAY(startDate, 'localtime'))/7) weeks
FROM StaffAssignment;

staffCode   weeks
----------  ----------
1           480.0
2           475.0
3           480.0
4           475.0
5           474.0
6           479.0
7           457.0
8           474.0
...
```

# Quiz 08

- Modify the SELECT statement in the previous example, so that it calculates the total weeks of employment correctly for staff that have left the company as well

# More single-row function

| Function | Description |
|---|---|
| IFNULL(X,Y) | It returns a copy of its first non-NULL argument; it could be used to replace a NULL value from a column with an alternative value |
| TYPEOF(X) | It returns a string that indicates the datatype of the expression X: null, integer, real, text, or blob |
| UNICODE(X) | It returns the numeric unicode code point corresponding to the first character of the string X |

# Summary

- By now you should:
  - be familiar with the basics of SQL including
    - CREATE, ALTER and DROP TABLE
    - INSERT, SELECT, UPDATE and DELETE
    - operator
    - sorting
    - single-row function

# Reading

- Essential
  - Queries from SQL for Web Nerds
- Further
  - SQL supported in SQLite
  - Expression and operator in SQLite
  - Datatype in SQLite
  - Core function in SQLite
  - Date and time function in SQLite

# Schedule

| Week | Lecture |
|---|---|
| 01 | Introduction |
| 02 | Relational model |
| 03 | ER modelling |
| 04 | Data modelling |
| 05 | Data modelling |
| 06 | Normalisation |
| 07 | SQL |
| 08 | SQL |
| 09 | SQL |
| 10 | DBMS fundamentals |
| 11 | Data warehouse |
| 12 | Review |

# THE END

Database is awesome in everywhere!