SQL

Query Structure

Basic clauses for most queries

* capitalization DOES NOT matter

* line breaks are optional, but help readability

SELECT	columns to show
...	
FROM	data sources and join criteria
...	
WHERE	filter conditions; rows to include
...	
GROUP BY	list of un-aggregated columns groups = unique combinations of values
...	
HAVING	filter conditions based on aggregate values
...	
ORDER BY	sort order for rows
...	

SELECT clause

$$\underbrace{\text{people}}_{\text{source}}\underset{\text{possessive}}{\underset{\uparrow}{.}}\underbrace{\text{people\_id}}_{\text{column}}\ \underset{\text{optional}}{\underset{\uparrow}{\text{AS}}}\ \underbrace{\text{client\_people\_id}}_{\text{alias for column name}}$$

- column aliases only apply in SELECT
- helps clarify what the column represents
- use to label a new calculated column

* table name is required if more than one table has a column of that name
- Which version of people_id do you want?
- The one from people, not from staff!

+ − * / } Standard math operators

+ also joins strings:
'base' + 'ball' ⇒ 'baseball'

* calculations apply to each row

SQL has logical tests, but no Boolean data type, so use CASE

Each WHEN... THEN pair is an input test and output

Evaluated in order; first true condition is the result

ELSE is used if all tests false

```
CASE
   WHEN age < 13 THEN 'Child'
   WHEN age BETWEEN 13 AND 19
      THEN 'Teen'
   WHEN age >= 20 THEN 'Adult'
   ELSE NULL
END
```

Substitutes a default value if the column is null

```
isnull (end_date, '9999-12-31')
```
year-month-day

FROM clause

TABLE_1

ID1	VALUE1
1	A
2	B
3	C
4	NULL

TABLE_2

ID2	VALUE2
5	A
6	B
7	NULL
8	D

Join types:

inner	- null values from both tables removed
left outer	- null values from left table kept
right outer	- null values from right table kept
full outer	- null values from both tables kept

```
SELECT *
FROM TABLE_1 - left table
_____ JOIN TABLE_2 ON TABLE_1.VALUE = TABLE_2.VALUE
                ↳ right table
```

(optional keywords)

ID1	VALUE1	ID2	VALUE2
1	A	5	A
2	B	6	B
3	C	NULL	NULL
4	NULL	NULL	NULL
NULL	NULL	7	NULL
NULL	NULL	8	D

} inner join

} left (outer) join

} right (outer) join

} (full) outer join

* the same table can be used multiple times - and can even join to itself - but needs a different alias for each use.

* multiple matches produce one row for every match.

WHERE clause

logical operators

=	equals
<>	not equal
>	greater than
>=	" or equal to
<	less than
<=	" or equal to

* strings are compared lexically (in dictionary order)
'dogs' > 'cats' } both are true!
'3' > '10'

* comparisons to NULL always yield NULL
-use x is null or x is not null

NOT negates a result, e.g. NOT (1=2) ⇒ true
AND both tests are true, e.g. (1=1) AND (2=2)
OR one or both tests are true, e.g. (1=2 OR 3=3)

* use parentheses to group tests, otherwise all tests evaluate left to right
1=2 AND 3=4 OR 5=5 ⇒ true