SQL Tutorials - wk 1

SELECT col1, col2 ... FROM Table name ;

* : all columns

WHERE condition ;

Numerical conditions : | =, <, >, ≤, ≥ | BETWEEN 10 AND 20 | IN (A, B, C)

WHERE cond1 [AND / OR] cond2 ; * multiple condition checking.

'strings'

=

↳ are case-sensitive (though sql queries aren't case-sensitive)

LIKE % : any # of chars
_ : 1 position char
ILIKE (case-insensitive LIKE)
substring, position
~* (for regex)

} Not covered in wk1 !!
(self-research)

SQL Tutorials - 2 - Aggregates & Dates

Aggregates

`COUNT(*)` : count all rows

`COUNT(DISTINCT colname)`

↳ count # of unique values in colname.

`MIN(col)` , `MAX(col)`, `AVG(col)` , `SUM(col)`

minimum · maximum · average · total sum

Multiple Aggregates

```
SELECT MIN(temp), MAX(temp) FROM observations;
SELECT AVG(temp), AVG(speed) FROM observations;
```

Ordering - Order By

↳ by highest marks
most recent date (chronological)
lowest price
alphabetical

DESC : highest to lowest

ASC : lowest to highest (default)

order by multiple colums.

```
ORDER BY obsdate, city;
```

1. order by obsdate
2. among observations with same obsdate, order by city name (alphabetical)

ASC (oldest → newest) · DESC (Z → A)

Dates [YYYY-MM-DD] ← ISO Date format

range: `BETWEEN '2020-01-01' AND '2020-12-31';`

today = `CURRENT_DATE`;

Common SQL ERRORS

↳ AND/OR 적절히 사용

↳ 날짜 valid 한지 체크하기!

↳ convention 따르기

SQL Tutorials - Week 3 - Summarising

LIMIT 10; : retrieve only the top 10 rows

Random sampling

=> ORDER BY random() LIMIT 30;

Top n-queries

ex) top 3 windiest days in sydney.

```
SELECT * FROM observations
WHERE city = 'Sydney'
ORDER BY wind_speed DESC LIMIT 3;
```

windiest → less wind

Renaming columns

SELECT col1 AS "renamed col1" FROM ...

adding constants:

SELECT 'Celsius' AS "unit" => Celsius, Celsius, Celsius

Converting data values

celsius → farenheit:

Select air_temp * 9/5 + 32 AS "temperature (°F)"

ROUND(num, precision)

i = %0.1f

|| : string concatenation ← might need to cast numerical values first to string then concatenate

TO_CHAR(date, "format")

↳ "DD Mon YYYY" = 23 Feb 2003

Categorising

```
CASE
    WHEN <cond 1> THEN 'category 1'
    WHEN <cond 2> THEN '  cat 2'
    ELSE 'cat 3'
END AS ("category")
```

ordering by category

ORDER BY "category", "renamed col 2"

SQL Tutorials - Week 4 - Joins

Joins

From Stations, Organisations
WHERE Stations.Stnowner = Organisations.code

PK
FK

Table alias

From Stations S, Measurements M

Natural Join (⋈)

- unlike equality join, common attribute appears only once
  must have same attr name

Inner Join

From Stations Join Organisations on (Stnowner = code);
attr. with different names

or

From Stations Join Measurements Using (Station)
explicit natural join

Self-Join

ex) Stations commenced before 'Murray River at Penuquin'

```
Select B.site, B.commence, B.Stnowner
From Stations A, Stations B
(x) cartesian product
where A.commence > B.commence;
AND A.site = '          ';
```

technically these are join conditions

SQL Tutorials - wk 5 - Data Cleaning

Dealing with ~~inconsistes~~ inconsistencies

- spelling errors & capitalisation (NSW vs nsw vs New South Wales)
- missing values (NULL, n/a)
- special values (NaN, infinity)
- inconsistent data format (numbers written in string)

UPPER / LOWER() : solves case-sensitivity issues

IN (A, B, C) : lists all possible values

combine : LOWER(___) IN (a, b, c);

LIKE ' ' : pattern matching

- %: any string, _ : placeholder for one character
- note that LIKE is case-sensitive

REGEXP with..

- SIMILAR TO (in psql)] not examinable, look into it later :)

NULL (unknown / missing values) — Simply missing? placeholder intended?

≠ '' ≠ 0 ∴ '' IS NULL = ~~false~~ NULL

0 is NULL = ~~false~~ NULL

3-valued logic

Interesting 3-value logic exprs

- NULL OR TRUE = T
- NULL AND FALSE = F
- NOT (NULL) = unknown

* Aggregate functions simply ignore NULL values.

Displaying null : ~~xpset null (NULL)~~ (Don't use) ; ← prevents confusion with '', empty string.

Coalesce(val, 'unknown')

↳ if num, needs to be casted to string by ::text.

Placeholders (-, --, '.', '', n/a)

↳ replace to NULL by case_when.

↳ If placeholders have been removed, cast them to appropriate type.

& inconsistencies — CAST (col AS FLOAT)

SQL Tutorials wk 6 (+wk 5) Database Design

Outer Joins

- includes all rows from one table,
  - matching rows of other table. (nulls if no matching row)
- Unlike inner join, outer join keeps non-matching rows in retrieval.

Data Manipulation - Insert / Update / Delete

- Insert Into <table> (attr order) — optional
  VALUES (col1 , col2 ...)
- UPDATE <table> SET <expr> attr = (val) ← returns # of rows affected
  WHERE <condition>
- DELETE FROM <table> WHERE <cond>

Tables / constraints

- Attributes: SMALLINT, INT, FLOAT, CHAR(n), VARCHAR(n)
- consistency constraints: NOT NULL, PRIMARY KEY, UNIQUE, CHECK (expr), DEFAULT
- foreign key: REFERENCES <table> (attr)
  - value must also exist in ↑ parent table

DROP TABLE <table> IF EXISTS;

SET OPERATORS (not Joins)

UNION
INTERSECT
EXCEPT

Select colA FROM TABLEA
→ (set operator)
Select colA FROM TABLEB

(union all) keeps duplicates
intersect all
except all

Views

CREATE VIEW <view name> AS <query>;
- nothing stored, dynamically fetched

DROP VIEW <view name>;

SQL Tutorials - Wk 7 Subqueries

- Subqueries in WHERE clause

WHERE value (=) (SELECT aggregate From..)
WHERE value (IN) (SELECT column From...)

EXISTS / NOT EXISTS

SELECT * FROM TABLES S
WHERE EXISTS (S의 attr을 참조한 query)

Co-related vs Non-co-related

alias로 같은지 아닌 query에서 사용하면 co-related

SQL Tutorials - Wk 8 Grouping

GROUP BY

Only those attributes that defines the group and the aggregates can be displayed (fetched)

HAVING : checks condition for groups.