Lecture 6 & Lab 6

More about NULL

expressions with NULL values(we should treat is as **unknown**)

- Arithmetic operations:

```
col + null -> null
col - null -> null
col * null -> null
col / null -> null
...
```

- Comparison operations

```
col > null -> null
col = null -> null
...
```

- Logical operators

```
true and null -> null
false and null -> false
true or null -> true
false or null -> null
```

Ordering in SQL

1. `Order by`

```
select * from {tab1}
where {conditions}
order by {col1} [asc|desc]
```

2. Advance ordering
    - Multiple columns

```
select * from {tab1}
where {conditions}
order by {col1}, {col2} [asc|desc]
```

    - with `join`

```
select tab1.col1, tab2.col1
from {tab1} join {tab2} on tab1.col2 = tab2.col2
where {conditions}
order by {tab1.col1}, {tab2.col2} [asc|desc]
```

- create a new column by `case...when`

```
select * from {tab1}
order by
case col1
    when {case1} then val1
    when {case2} then val2
end new_col [desc|asc]
```

3. Data Types in Ordering
   - Ordering depends on the data type
     - Strings: **alphabetically**,
     - Numbers: **numerically**
     - Dates and times: **chronologically**
   - what about **NULL**?
     - It is implementation-dependent
     - **SQL Server, MySQL and SQLite**:

       "nothing" is **smaller** than everything
     - **Oracle and PostgreSQL**:

       "nothing" is **greater** than anything
   - ****more about **alphabetically**(Not cover in the test)

     They are a lot of **charset** for character encoding. We may need different charset according to the dataset.

     > chatGPT

     1. Modify the encoding of the PostgreSQL server and client, which means that after that, **newly created databases and clients will use the new encoding**.
        - Stop the PostgreSQL database service.
        - Locate and edit PostgreSQL's main configuration file `postgresql.conf`, which is usually located in the `data` subdirectory of the database installation directory.
        - Find the two parameters `client_encoding` and `server_encoding` in the `postgresql.conf` file, and modify their values to the encoding method to be used, for example:

          ```
          client_encoding = 'UTF8'
          server_encoding = 'UTF8'
          ```

Save the modified `postgresql.conf` file.

- Start the PostgreSQL database service.

2. For existing databases, you need to manually modify their **encoding** to be consistent with **server_encoding**.
   - back up the data
   - use the following command to modify the encoding of the database

```
alter database {database_name} set encoding {encoding_name};
```

   - Restart the postgreSQL to make all modifications effective.

4. `Limit` and `Offset`
   - syntax:

```
select * from {tab1}
where {cond}
order by {col1}
limit {k} offset {p}
```

   - function

     Return (at most) `top-k` rows in the result set **after** skipping the first (at most) `p` row

Window Function

1. Scalar Functions
   - Functions that operate on values **in the current row**
   - In short, **operate a column and return a column**
2. Issues with Aggregate Functions
   - details of the rows are **vanished**
   - Some problems that aggregate is not easy to solve:
     - How can we **rank** the movies **in each country separately** based on the released year?
     - we may be know the min, max... but difficult to know the other detail of the min/max element
   - A solution is Window Function

   Different between aggregation function and window function
   - aggregation function can only return a value for a group, but the window funtion can (1)return gradual values according to the partition in some order, and this gradual value always contain the characteristic value for a partition, or (2)grouping and map each row to a new value respectively without reducing the row count.
3. Window Function syntax

```
select
<function> over
(partition by <col_p> order by <col_o1, col_o2, …>)
{alias1}
from {tab1}
```

- `<function>`: we can apply

  (1) ranking window functions(apply in each row)

  (2) aggregation functions

- `over()`: define the function, if it's empty in the paratheness, the window is the whole table
- `partition by`: specify the column for grouping , same value of the specify column consist a **Window**
- `order by`: specify the column(s) for ordering in each **window**

4. Ranking Window Function

   - To rank the rows base on a column

```
select
rank() over (
    partition by {col1} order by {col2}
) ranking
from {tab1}
```

   - It is just a new column:

     Grouping first, then order in each group, then give a rank according to the order in each group.

   - Why window function, not group by?

     "Group by" reduces the rows in a group (partition) into **one result**.

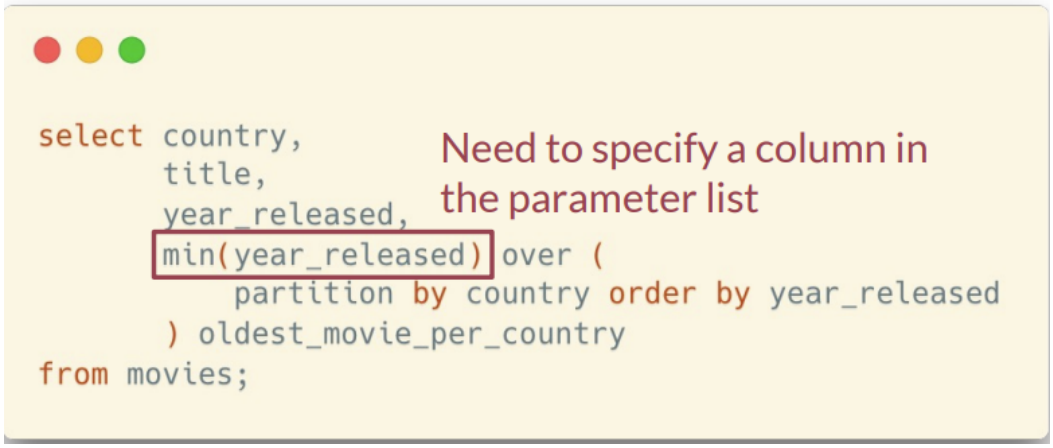
   - Some other ranking window function

     `dense_rank()` and `row_number()`

country	title	year_released	rank_result	dense_rank_result	row_number_result
cn	some title	1948	1	1	1
cn	some title	1959	2	2	2
cn	some title	1959	2	2	3
cn	some title	1987	4	3	4
cn	some title	2002	5	4	5
uk	some title	1985	1	1	1
uk	some title	1992	2	2	2
uk	some title	2010	3	3	3

     `row_number() over()` can be used to label each row with a row number.

5. Aggregation Functions as Window Functions

- `max(col)` and `min(col)`
  - 

  - The `min/max` value for each partition is assigned for **all the rows** inside this partition

	cou...	title	year_released	oldest_movie_per_country
1	am	Sayat Nova	1969	1969
2	ar	Pampa bárbara	1945	1945
3	ar	Albéniz	1947	1945
4	ar	Madame Bovary	1947	1945
5	ar	La bestia debe morir	1952	1945
6	ar	Las aguas bajan turbias	1952	1945
7	ar	Intermezzo criminal	1953	1945
8	ar	La casa del ángel	1957	1945
9	ar	Bajo un mismo rostro	1962	1945
10	ar	Las aventuras del Capitán Piluso	1963	1945
11	ar	Savage Pampas	1966	1945
12	ar	La hora de los hornos	1968	1945
13	ar	Waiting for the Hearse	1985	1945
14	ar	La historia oficial	1985	1945
15	ar	Hombre mirando al sudeste	1986	1945

- `sum(col)`, `count(col)`, `avg(col)`, `stddev(col)`
  - When order by is specified

    These windows function will be execute accumulately, which means **the aggregation value of [firstRowInPartition, LastRowOfSameOrder] in its partition**(it will be the same in the same level order)
  - When order by is not specified

    These windows function will be execute for all rows in the partition, same as aggregate function in `grouping by`
- Specially, we want to get a column of same characteristic value
  - `sum() over()`, `avg() over()`, `stddev() over()`, `count() over()`
  - The partition is the whole table, and no `order by` means they're of the same order

6. Useful window function

- `lead(col, offset) over()`: create a column that **lead** `col` with a `offset` in the window
- `lag(col, offset) over()`: create a column that **lag** `col` with a `offset` in the window