SQL DML. Part 5.

Groups

SELECT Statement – Grouping

CITY UNIVERSITY LONDON

- Use **GROUP BY** clause to get sub-totals.
- **SELECT** and **GROUP BY** closely integrated: each item in **SELECT** list must be single-valued per group, and **SELECT** clause may only contain:
  - column names appeared in **GROUP BY**
  - aggregate functions
  - constants
  - expression involving combinations of the above.

SELECT Statement – Grouping

CITY UNIVERSITY LONDON

- All column names in **SELECT** list must appear in GROUP BY clause unless name is used only in an aggregate function.
- If **WHERE** is used with **GROUP BY**, **WHERE** is applied first, then groups are formed from remaining rows satisfying predicate.
- ISO considers two NULLs to be equal for purposes of **GROUP BY**.

Example 1.

CITY UNIVERSITY LONDON

Calculate the number of matches on each ground.

```
SELECT ground_name, COUNT(*)
FROM mccMatches
GROUP BY ground_name;
```

But what if we need match_date?..

```
SELECT ground_name, match_date, COUNT(*)
FROM mccMatches
GROUP BY ground_name;
```

Same thing with *ONLY_FULL_GROUP_BY* SQL_MODE as with aggregation.

Example 2.

CITY UNIVERSITY LONDON

How many matches have been played at each of the different grounds since 2012?

```
SELECT ground_name, COUNT(*)
FROM mccMatches
WHERE match_date > 120100
GROUP BY ground_name;
```

Example 3.

CITY UNIVERSITY LONDON
EST 1894

The HAVING clause restricts the rows produced by the GROUP BY clause using comparison operations (just like a WHERE clause does for SELECT)

List the grounds where MCC have played at least 4 times since 2005

```
SELECT ground_name, COUNT(*)
FROM mccMatches
WHERE match_date > 050100
GROUP BY ground_name
HAVING COUNT(*) >= 4;
```