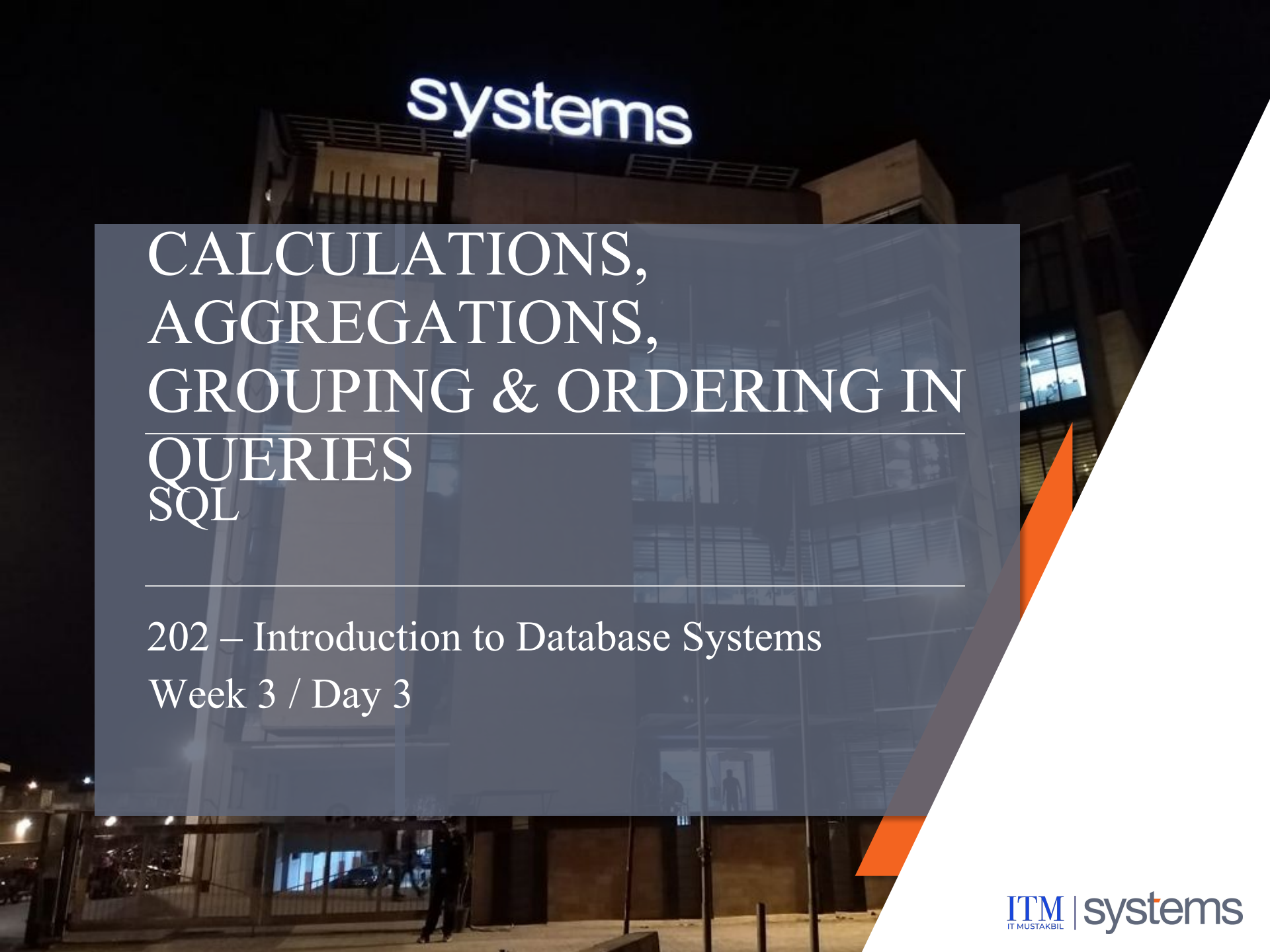

CALCULATIONS, AGGREGATIONS, GROUPING & ORDERING IN QUERIES

SQL

202 – Introduction to Database Systems

Week 3 / Day 3

ITM | systems
IT MUSTAKBIL

LEARNING OBJECTIVES & AGENDA

Learning Objectives:

- Calculating Derived Attributes
- Use aggregate functions.
- Use GROUP BY clause to group data.
- Use ORDER BY clause to arrange data in ascending / descending order.

Agenda / Sub-Topics:

- Derived Attributes.
- Aggregate Functions.
- GROUP BY Clause.
- ORDER BY Clause.

DERIVED ATTRIBUTES

DERIVED ATTRIBUTES

- Attributes that can readily be computed using other attributes and need not be stored.
  - E.g. Percentage marks can be calculated using obtained marks.
  - E.g. Sales tax amount can be calculated using the sales amount.
- Attributes that are computed on the run are known as ‘derived attributes’.

DERIVED ATTRIBUTES

- `SELECT mids + finals AS 'total_marks' FROM student_marks;`

- `SELECT 0.175 * price * quantity AS 'sales_tax' FROM sales;`

AGGREGATE FUNCTIONS

AGGREGATE FUNCTIONS

- Aggregate functions allow us to obtain single statistical metrics against a given collection of tuples for a given field.
- List of Aggregate Functions:
  - `COUNT()`
  - `SUM()`
  - `AVG()`
  - `MIN()`
  - `MAX()`

AGGREGATE FUNCTIONS

```
SELECT
   COUNT(roll_no) AS ‘total_students’,
   MIN(marks) AS ‘minimum_marks’,
   MAX(marks) AS ‘maximum_marks’,
   AVG(marks) AS ‘average_marks’
 FROM student_marks;
```

GROUP BY CLAUSE

GROUP BY CLAUSE

- `GROUP BY` used to obtain category-wise aggregates.
- Always accompanied by an aggregate function.
  - If we are viewing grouped data, how do we decide which tuple to show?
  - We need an aggregate function to determine a finite quantity to represent against each category collection.
- `HAVING` used to apply conditions on the category-wise aggregates.

GROUP BY CLAUSE – EXAMPLE

```
SELECT
   COUNT(roll_no) AS ‘total_students’,
   MIN(marks) AS ‘minimum_marks’,
   MAX(marks) AS ‘maximum_marks’,
   AVG(marks) AS ‘average_marks’
 FROM student_marks
 WHERE batch = 2022
 GROUP BY department;
 HAVING total_students < 50;
```

GROUP BY CLAUSE – EXAMPLE EXPLANATION

- `WHERE` ensures only students of batch 2022 are filtered and included before grouping / categorization takes place.
- `GROUP BY` collects students against their departments. This means there will be a department on one side, and a collection of student tuples with matching departments on the other side.
  - In order to select a finite value to show against each department, we will use an aggregator.

GROUP BY CLAUSE – EXAMPLE EXPLANATION

- `HAVING` used to filter out groups whose total students are greater than 50.
  - This is different from the `WHERE` clause because the where clause applied filters on the student tuples to choose which tuples should be included in the grouping.
  - `HAVING` applies filters on the aggregates that are computed against each group / category, and not on the individual tuples themselves.

ORDER BY CLAUSE

ORDERING RESULTS

- Data is stored on a first come first serve basis.
- Data is appended to the bottom of the table as it is comes.
- Tuples are not ordered in any particular way.
- We can order tuples using the `ORDER BY` clause.
  - The default ordering of the `ORDER BY` clause sorts the field in `ASC` order.
  - This can be changed to descending order too by using the keyword `DESC`.

ORDER BY CLAUSE

```
SELECT
   COUNT(roll_no) AS 'total_students',
   MIN(marks) AS 'minimum_marks',
   MAX(marks) AS 'maximum_marks',
   AVG(marks) AS 'average_marks'
FROM student_marks
WHERE batch = 2022
GROUP BY department
ORDER BY total_students DESC;
```