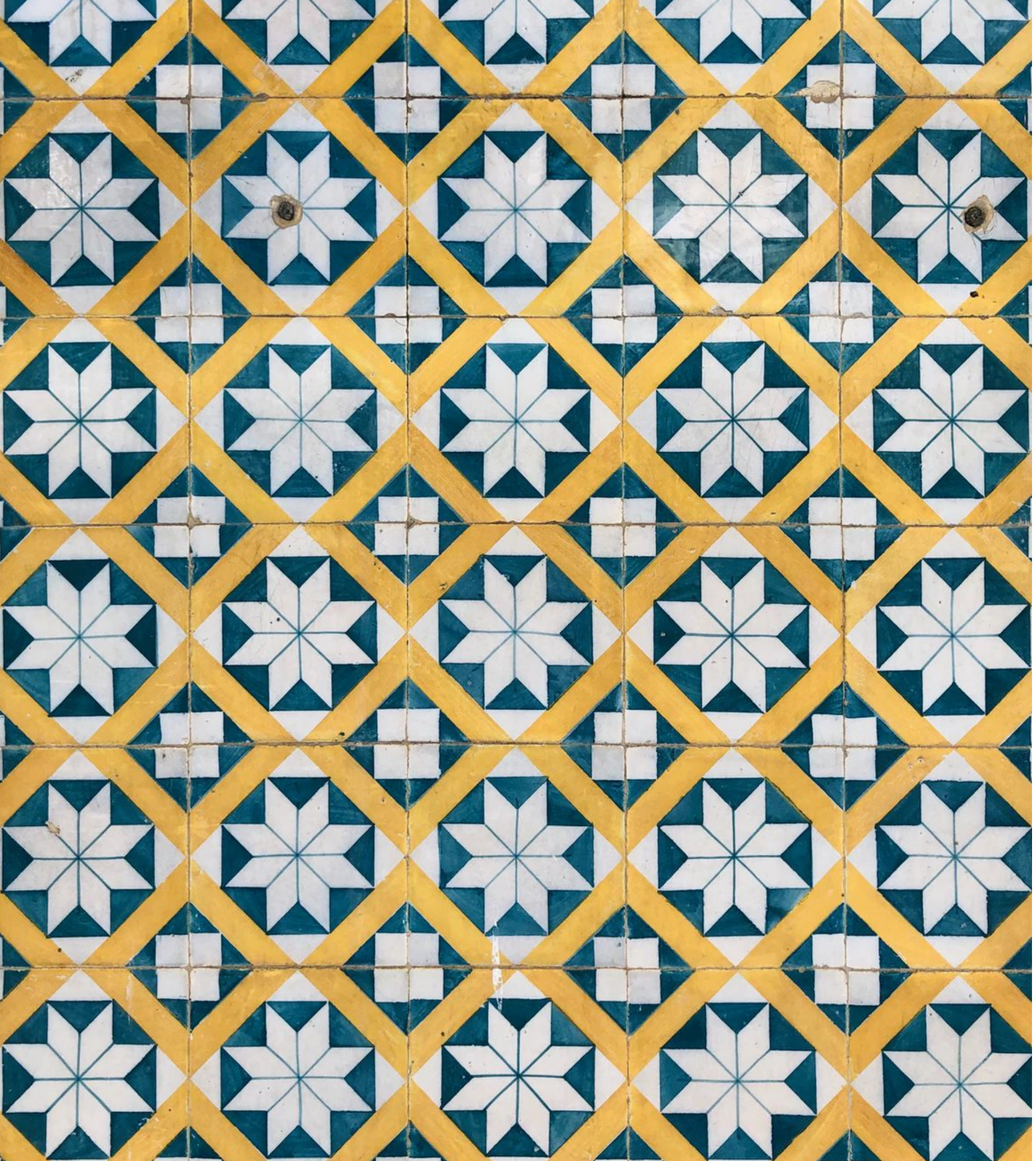


SQL 3
Intermediate SQL Queries & SQL Review

METIS®

Objectives

Intermediate SQL Queries

- Prerequisite: Query basics including the SELECT statement and JOINs
- This lesson: CASE statements, subqueries and window functions

Chapter 1

CASE

METIS®

Basic SQL queries

```
SELECT *
FROM doctors;
```

Name	Day	Location	Details	Rate
Arthur	Monday	Chicago	Check ups	90
Arthur	Sunday	Chicago	On call	20
Dora	Monday	Evanston	Surgery	210
Dora	Wednesday	Evanston	Surgery	210
Dora	Sunday	Chicago	On call	20
Peppa	Wednesday	Evanston	Check ups	90

```
SELECT Name, Details
FROM doctors;
```

Name	Details
Arthur	Check ups
Arthur	On call
Dora	Surgery
Dora	Surgery
Dora	On call
Peppa	Check ups

The SELECT Clause

Display existing columns

- Show all columns with an *
- Show specific columns by listing them out

Display additional columns

1. Calculations
2. Aggregations
3. The CASE statement

Doctors table

Name	Day	Location	Details	Rate
Arthur	Monday	Chicago	Check ups	90
Arthur	Sunday	Chicago	On call	20
Dora	Monday	Evanston	Surgery	210
Dora	Wednesday	Evanston	Surgery	210
Dora	Sunday	Chicago	On call	20
Peppa	Wednesday	Evanston	Check ups	90

1. Calculations

```
SELECT Name, Rate
FROM doctors;
```

Name	Rate
Arthur	90
Arthur	20
Dora	210
Dora	210
Dora	20
Peppa	90

```
SELECT Name, Rate + 5
FROM doctors;
```

Name	Rate + 5
Arthur	95
Arthur	25
Dora	215
Dora	215
Dora	25
Peppa	95

2. Aggregations

```
SELECT Name, AVG(Rate)
FROM doctors
GROUP BY Name;
```

Name	AVG(Rate)
Arthur	55
Dora	170
Peppa	90

```
SELECT Name, AVG(Rate) as Average
FROM doctors
GROUP BY Name;
```

Name	Average
Arthur	55
Dora	170
Peppa	90

3. CASE

```
SELECT Name, Day,
        CASE WHEN Day = ‘Saturday’
        OR Day = ‘Sunday’
        THEN ‘Weekend’
        ELSE ‘Weekday’ END
        AS Day_Type
FROM doctors;
```

Name	Day	Day_Type
Arthur	Monday	Weekday
Arthur	Sunday	Weekend
Dora	Monday	Weekday
Dora	Wednesday	Weekday
Dora	Sunday	Weekend
Peppa	Wednesday	Weekday

3. CASE with an aggregation (step 1)

How many weekend days does each doctor work?

```
SELECT Name, Day,
       CASE WHEN Day = 'Saturday'
       OR Day = 'Sunday'
       THEN 1
       ELSE 0 END
       AS Day_Type
FROM doctors;
```

Name	Day	Day_Type
Arthur	Monday	0
Arthur	Sunday	1
Dora	Monday	0
Dora	Wednesday	0
Dora	Sunday	1
Peppa	Wednesday	0

3. CASE with an aggregation (step 2)

How many weekend days does each doctor work?

```
SELECT Name, SUM(
        CASE WHEN Day = 'Saturday'
        OR Day = 'Sunday'
        THEN 1
        ELSE 0 END)
        AS Num_Weekend_Days
FROM doctors
GROUP BY Name;
```

Name	Num_Weekend_Days
Arthur	1
Dora	1
Peppa	0

Summary

Display existing columns

- Show all columns with an *
- Show specific columns by listing them out

Display additional columns

1. Calculations
2. Aggregations
3. The CASE statement

Chapter 2

Subqueries & Common Table Expressions (CTE)

METIS®

Example query

```
SELECT Name, Rate + 5 AS Rate_Plus_5
FROM doctors;
```

Name	Rate_Plus_ 5
Arthur	95
Arthur	25
Dora	215
Dora	215
Dora	25
Peppa	95

New_Rate_Table

What if?

```
SELECT *
FROM New_Rate_Table;
```

Name	Rate_Plus_ 5
Arthur	95
Arthur	25
Dora	215
Dora	215
Dora	25
Peppa	95

What if?

```
SELECT Name, AVG(Rate_Plus_5) AS Average
FROM New_Rate_Table
GROUP BY Name;
```

Name	Rate_Plus_ 5
Arthur	60
Dora	152
Peppa	95

Two approaches

Subqueries

- A query within a query

Common Table Expressions (CTE)

- Also called a WITH statement
- Creates a temporary data set only available during execution

What we want to do in theory

```
SELECT Name, AVG(Rate_Plus_5) AS Average
FROM New_Rate_Table
GROUP BY Name;
```

Name	Rate_Plus_ 5
Arthur	60
Dora	152
Peppa	95

Subquery example in practice

```
SELECT Name, AVG(Rate_Plus_5) AS Average
FROM
(SELECT Name, Rate + 5 AS Rate_Plus_5
FROM doctors) AS New_Rate_Table
GROUP BY Name;
```

Name	Rate_Plus_ 5
Arthur	60
Dora	152
Peppa	95

WITH example in practice

```
WITH New_Rate_Table AS
(SELECT Name, Rate + 5 AS Rate_Plus_5
FROM doctors)
SELECT Name, AVG(Rate_Plus_5) AS Average
FROM New_Rate_Table
GROUP BY Name;
```

Name	Rate_Plus_ 5
Arthur	60
Dora	152
Peppa	95

Comparison of approaches

Subqueries

- The simpler approach that will work with most SQL software
- Can occur within multiple clauses like SELECT, FROM and WHERE

Common Table Expressions (CTE) aka WITH Statement

- Cleaner code
- Can be used for more complex concepts like recursion
- Temporary data set can be used for multiple queries
- Not supported within some very old SQL software

Chapter 3

Window Functions

METIS®

Window functions

- Perform operations on a “window” of the data, or rows of data

Doctors table

Name	Day	Location	Details	Rate
Arthur	Monday	Chicago	Check ups	90
Arthur	Sunday	Chicago	On call	20
Dora	Monday	Evanston	Surgery	210
Dora	Wednesday	Evanston	Surgery	210
Dora	Sunday	Chicago	On call	20
Peppa	Wednesday	Evanston	Check ups	90

Example query

```
SELECT COUNT(*) AS Total
FROM doctors;
```

Total
6

OVER()

```
SELECT COUNT(*) OVER() AS Total
FROM doctors;
```

Total
6
6
6
6
6
6

OVER()

```
SELECT COUNT(*) OVER() AS Total, Name
FROM doctors;
```

Total	Name
6	Arthur
6	Arthur
6	Dora
6	Dora
6	Dora
6	Peppa

PARTITION BY()

```
SELECT COUNT(*) OVER(PARTITION BY Name) AS Total, Name
FROM doctors;
```

Total	Name
2	Arthur
2	Arthur
3	Dora
3	Dora
3	Dora
1	Peppa

Window function syntax

```
COUNT(*) OVER(PARTITION BY Name)
```

What operation you'd like to perform.

The clause to specify a window function.

What window you'd like to perform the operation on. If left blank, then all rows.

The output is the calculation for a window of data + each original row of data

PARTITION BY()

```
SELECT COUNT(*) OVER(PARTITION BY Name) AS Total, Name
FROM doctors;
```

Total	Name
2	Arthur
2	Arthur
3	Dora
3	Dora
3	Dora
1	Peppa

ROW_NUMBER()

```
SELECT ROW_NUMBER() OVER(PARTITION BY Name) AS Num,
Name, Rate
FROM doctors;
```

Num	Name	Rate
1	Arthur	90
2	Arthur	20
1	Dora	210
2	Dora	20
3	Dora	210
1	Peppa	90

ROW_NUMBER()

```
SELECT ROW_NUMBER() OVER(PARTITION BY Name) AS Num,
Name, Rate
FROM doctors;
```

Num	Name	Rate
1	Arthur	90
2	Arthur	20
1	Dora	210
2	Dora	20
3	Dora	210
1	Peppa	90

ROW_NUMBER()

```
SELECT ROW_NUMBER()
OVER(PARTITION BY Name ORDER BY Rate DESC) AS Num,
Name, Rate
FROM doctors;
```

Num	Name	Rate
1	Arthur	90
2	Arthur	20
1	Dora	210
2	Dora	210
3	Dora	20
1	Peppa	90

ROW_NUMBER()

```
SELECT ROW_NUMBER()
OVER(PARTITION BY Name, Location) AS Num,
Name, Location
FROM doctors;
```

Num	Name	Location
1	Arthur	Chicago
2	Arthur	Chicago
1	Dora	Chicago
1	Dora	Evanston
2	Dora	Evanston
1	Peppa	Evanston

RANK()

```
SELECT RANK()
OVER(PARTITION BY Name ORDER BY Rate DESC) AS Num,
Name, Rate
FROM doctors_over;
```

Num	Name	Rate
1	Arthur	90
2	Arthur	20
1	Dora	210
1	Dora	210
3	Dora	20
1	Peppa	90

vs ROW_NUMBER()

Num	Name	Rate
1	Arthur	90
2	Arthur	20
1	Dora	210
2	Dora	210
3	Dora	20
1	Peppa	90

Window functions

- Perform operations on a “window” of the data, or rows of data
- Combines aggregations along with individual rows of data

Syntax

- OVER() - using a window function
- PARTITION BY() - the window

Calculations

- Aggregations: COUNT(), SUM(), etc.
- ROW_NUMBER()
- RANK()

Chapter 4

SQL Summary

METIS®

SQL Summary

- Summarize concepts covered in SQL 1 / 2 / 3
- Tips for using SQL on the job

SQL 1 Summary

Key Concepts

- Databases: store data in an organized way
- Data models: how tables are connected in a database

Tools

- RDBMS: SQLite
- GUI: DB Browser for SQLite

SQL 1 Summary

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

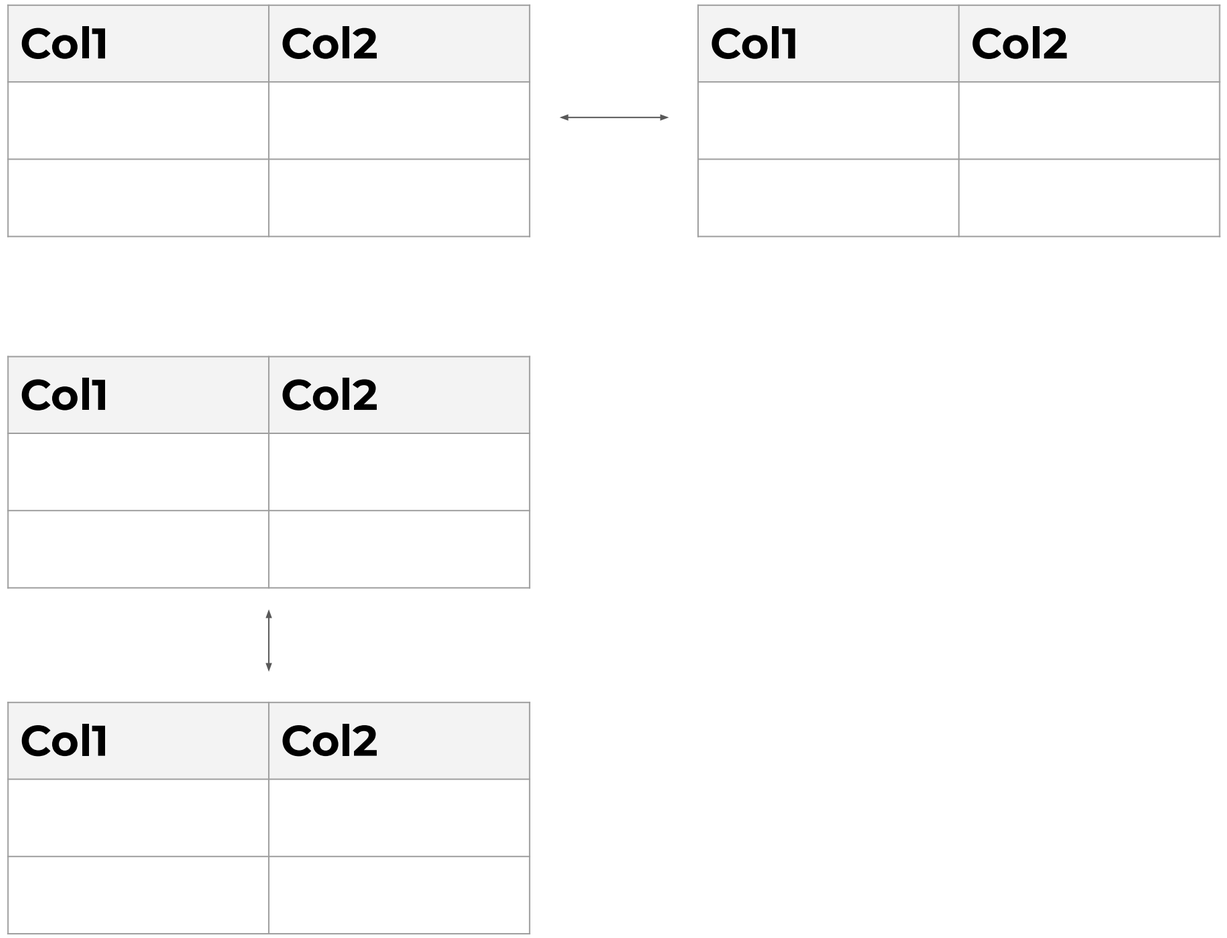
SQL 2 Summary

Create Tables

- CREATE
- INSERT

Work with Multiple Tables

- JOIN
- UNION



SQL 2 Summary

JOIN Types

- INNER JOIN - only rows in common
- LEFT JOIN - only rows in left table
- RIGHT JOIN - only rows in right table
- OUTER JOIN - all rows

SQL 3 Summary

Intermediate Queries

- CASE
- Subqueries and CTEs (WITH)
- Window Functions (OVER and PARTITION BY)

Next Step: Practice

Steps

1. Find a data model and understand it
2. Come up with interesting questions
3. Practice writing SQL queries to answer those questions

Next Step: Practice

Steps

1. Find a data model and understand it
2. Come up with interesting questions
3. Practice writing SQL queries to answer those questions

Example

1. European Soccer Dataset (open source) with 25k+ matches
2. Which team scored the most points when playing at home?
3. `SELECT ...`

Practical Tips

When you join a company

1. Download the RDBMS that they are using
2. Get familiar with the syntax and tools
3. Understand the company's database schemas
4. Talk to more experienced SQL people

Relational Database and SQL Closing Thoughts

Pros

- Stable
- Popular
- SQL is incorporated into other more modern tools and languages

Cons

- Rigid data structure
- Cumbersome queries

Suggestion

- Use SQL along with other programming languages