Data Analysis with SQL

Idan Gabrieli

Level 1 ⟶ Level 2

- Follow the order of the lectures
- Try to replicate the exact steps
- Quizzes
- Your final project
- Explore something you like….
  - Work? Side-project?
  - THINK → FUSE (new skills)
- Good luck!

Learning Objectives

- **Loading the Datasets**
  - Two groups of datasets
  - PostgreSQL
- **Combining Data from Multiple Tables**
  - Union, Intersect, Except
  - Join
- **Subqueries**
  - Nest queries within one another
  - Break a complex query
- **Conditional Logic (CASE)**
  - Data transformation and classification
  - Mimic the “if-then-else.”

Learning Objectives

- **Window Functions**
  - Complex data analysis
  - Transform a function to a widow function
- **Simplify Queries**
  - Views
  - CTEs
  - Stored procedures

Loading the Datasets

- **PostgreSQL**
  - DBMS for the training
- **Datasets**
  - eCommerce store
  - Books rating
- **Level 2 - SQL Code**
  - File to download!
  - Follow to steps – cut & paste the required code

Loading the Datasets

- **Books Rating Datasets**
  - books1.csv
    - List of books
      - ISBN, year, author, publisher
    - Published until 1999
  - books2.csv
    - Published from 1999
  - users.csv
    - List of users
      - User-id, age, city, state, country

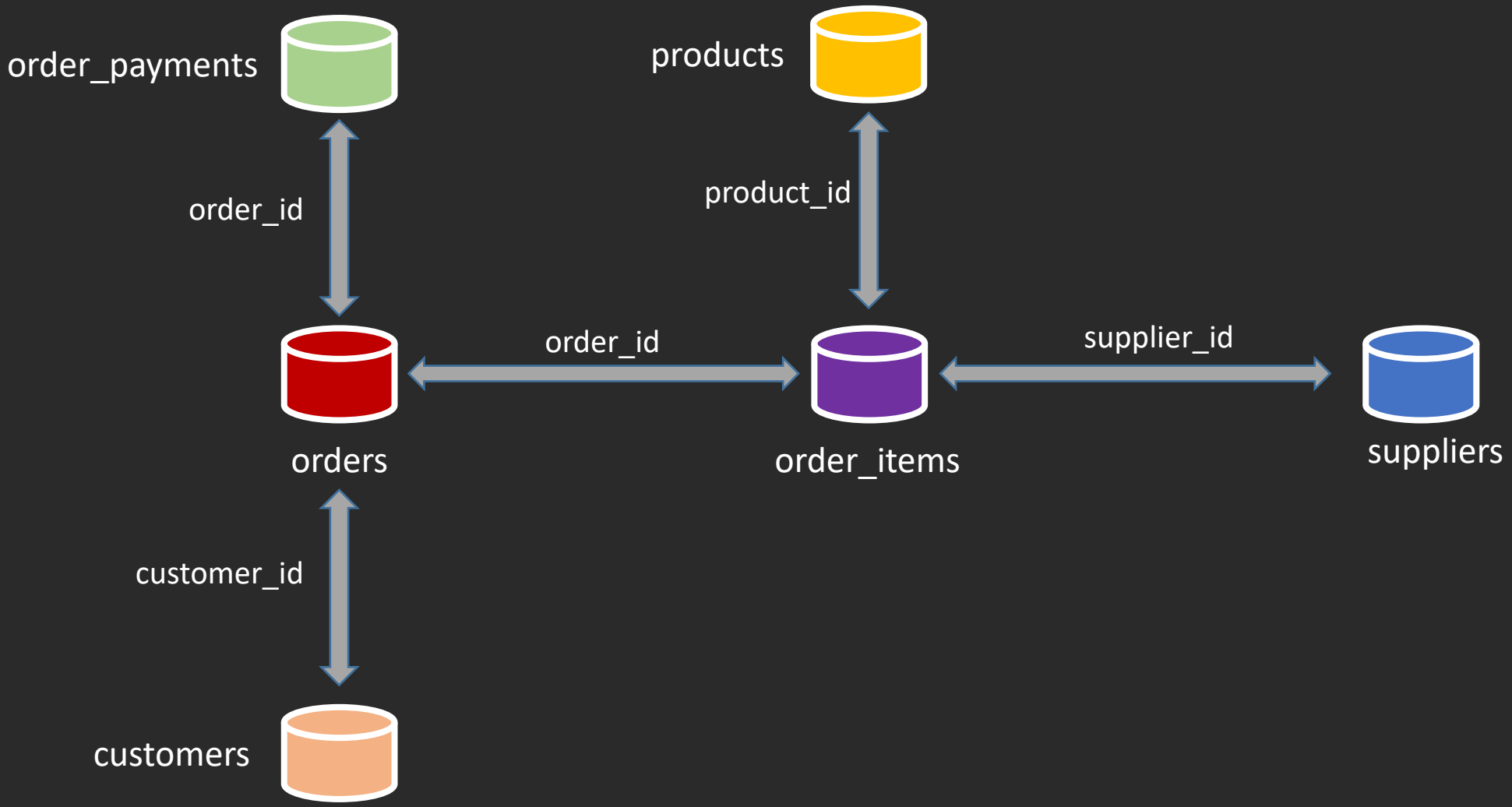
Loading the Datasets

- **Books Rating Datasets**
  - rating.csv
    - A rating score on a book by a user
      - user-id, ISBN, book rating
- **Steps:**
  - Database instance
  - Schema
  - Tables

Loading the Datasets

- **eCommerce Store Datasets**
  - customers.csv
  - products.csv
  - suppliers
  - orders.csv
  - order_reviews.csv
  - order_payment.csv

eCommerce Store



SQL - Combining Data from Multiple Tables

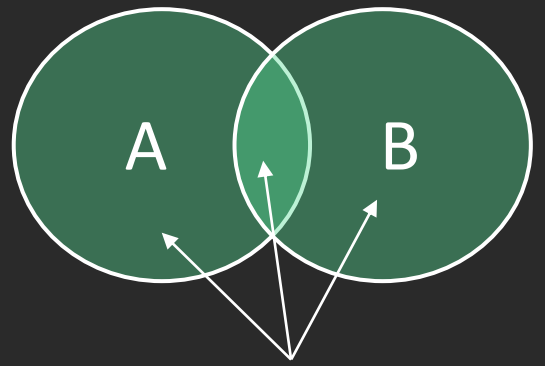
Union, Intersect, Except, Join

UNION

- **Combine outputs of SELECT statements**
  - Dropping duplicates
- **Syntax:**

```
SELECT column_names FROM table1
UNION
SELECT column_names FROM table2;
```

- **Queries must have the same structure**
  - Number of columns
  - Columns data types
  - Order

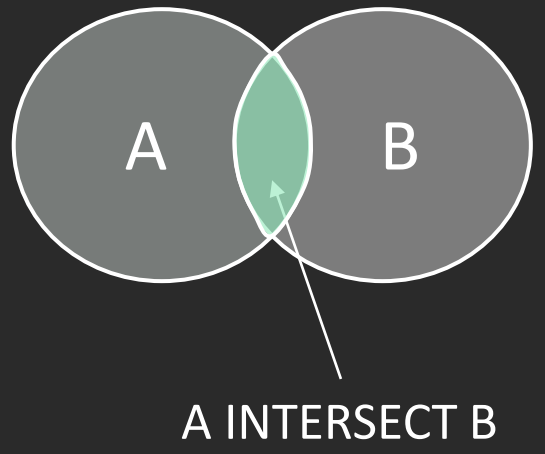


A UNION B

INTERSECT

- **Overlapping rows of SELECT statements**
- **Syntax:**

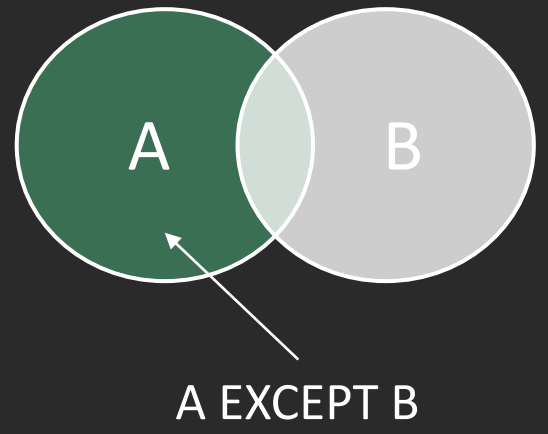
```
SELECT column_names FROM table1
INTERSECT
SELECT column_names FROM table2;
```



EXCEPT

- **Distinct rows from the first table**
  - Like: A – B
- **Syntax:**

```
SELECT column_names FROM table1
EXCEPT
SELECT column_names FROM table2;
```



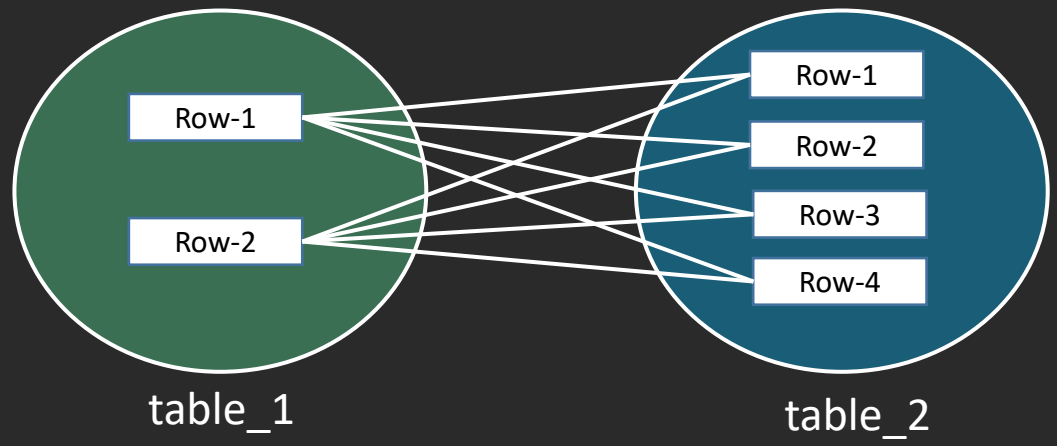
JOIN

- **Combine data from multiple unidentical tables**
- **Join Types: INNER, OUTER, CROSS-JOIN**
- **A Joined Table**
  - Derived from two tables based on the specific join type
- **Syntax**
  - FROM table_1 join_type table_2 [join_condition]
  - FROM table_1 join_type table_2 [join_condition] table_3 join_type [join condition]

CROSS JOIN

- **Also called cartesian join**
- **joining tables without specifying any join condition**
- **Syntax**
  - table_1 CROSS JOIN table_2
- **Joined table**

Each Row | Columns from table 1 | Columns from table 2

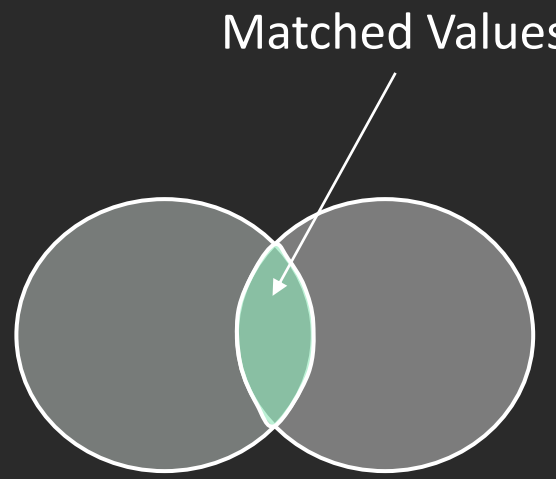


X = 10K, Y = 100K records → X * Y → joined table will reach 1M records!!!

INNER JOIN

- **Most frequently used join-type**
- **A join operation that is based on a condition**
  - Select records with matching values in both tables
  - Using relational links between tables
- **Syntax**

  SELECT [list of columns from both tables]
  FROM table1 INNER JOIN table2 ON [join condition]
- **Better to qualify the table names**
  - SELECT table1.column_x, table2.column_y



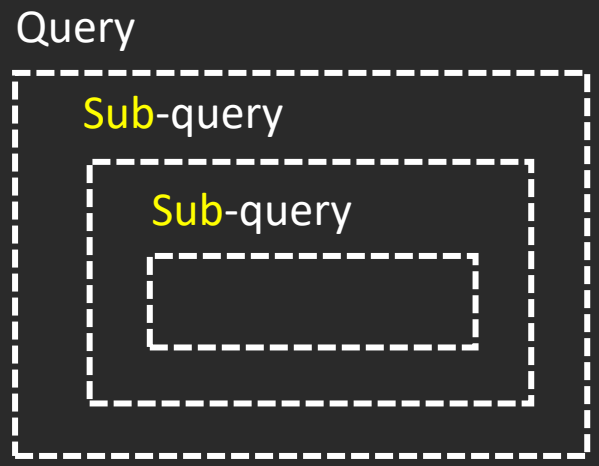
OUTER JOIN

- **Preserves unmatched rows from one of the tables**
- **Types:**
  - LEFT, RIGHT, FULL
- **LEFT OUTER JOIN**
  - All records from the left table (table1) and only the matched records from the right table (table2)
  - Table 1 = 10K, Table 2 = 5K → Output 10K
  - Matching rows will have columns values from table 2

OUTER JOIN

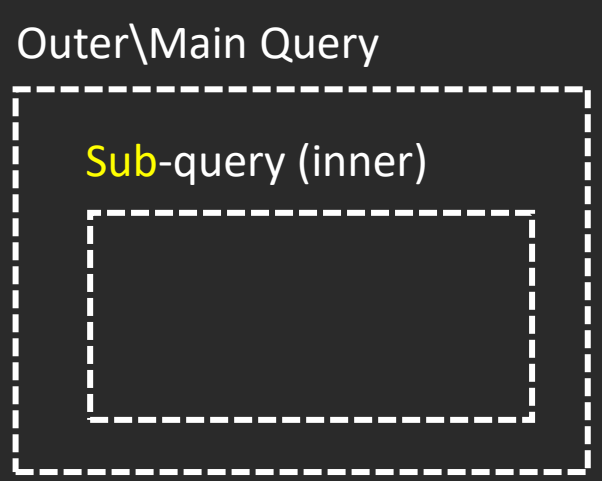
- **FULL OUTER JOIN**
  - All records when there is a match in left OR right table records
  - Three steps:
    1. An inner join is performed, matching rows based on a condition
    2. For each row in table 1 that does not satisfy the join condition with any row in table 2, a joined row is added with null values in columns of table 2
    3. And the third step, for each row of table 2 that does not satisfy the join condition with any row in table 1, a joined row with null values in the columns of table 1 is added.

SQL - Subqueries



What is a Subquery?

- **A subquery is also called an inner query**
  - A query within another query
- **The main statement - outer query**
- **For what?**
  - A tool for performing operations in multiple steps
  - Extract information from multiple tables
- **Where can it be used?**
  - SELECT
  - UPDATE, DELETE, INSERT



What is a Subquery?

- **What are the main types of sub-queries?**
  1. A single row with a single column
  2. Multiple rows with a single column
  3. Multiple rows having multiple columns – like a table
- **Output of the subquery**
  - HOW may it be used?
  - Which operators the containing statement may use?

SQL - Conditional Logic (CASE)

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

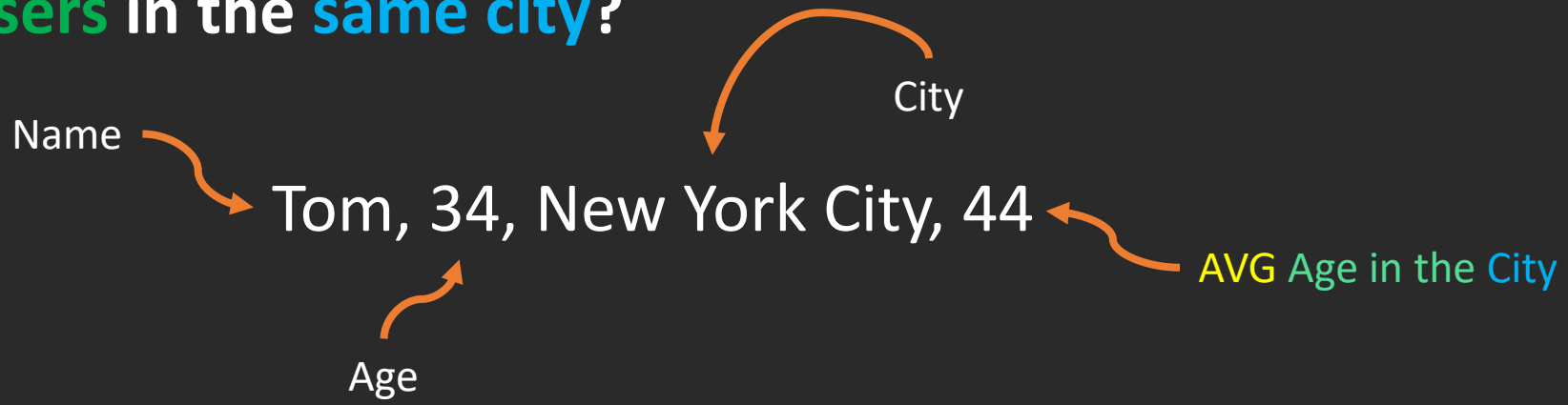
SQL - Window Functions

GROUP BY

The average user age from the users table, grouped by city location

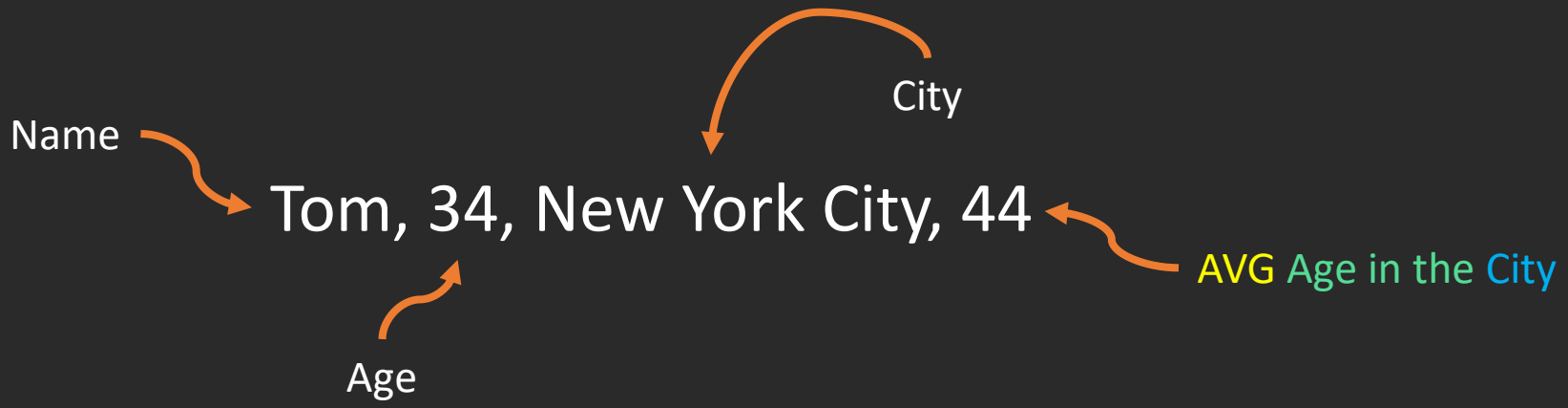
```
SELECT city, avg(age)
FROM books_schema.users
GROUP BY city
HAVING city IS NOT NULL
```

- **But what about the list of users, including their age and the average age of users in the same city?**



Window Functions

- **Function → Window Function**
  - E.g. AVG, COUNT, MAX
  - Apply calculation to a group of rows (window) BUT keep the rows in the final query



SQL - Simplify Queries (Views, CTEs)