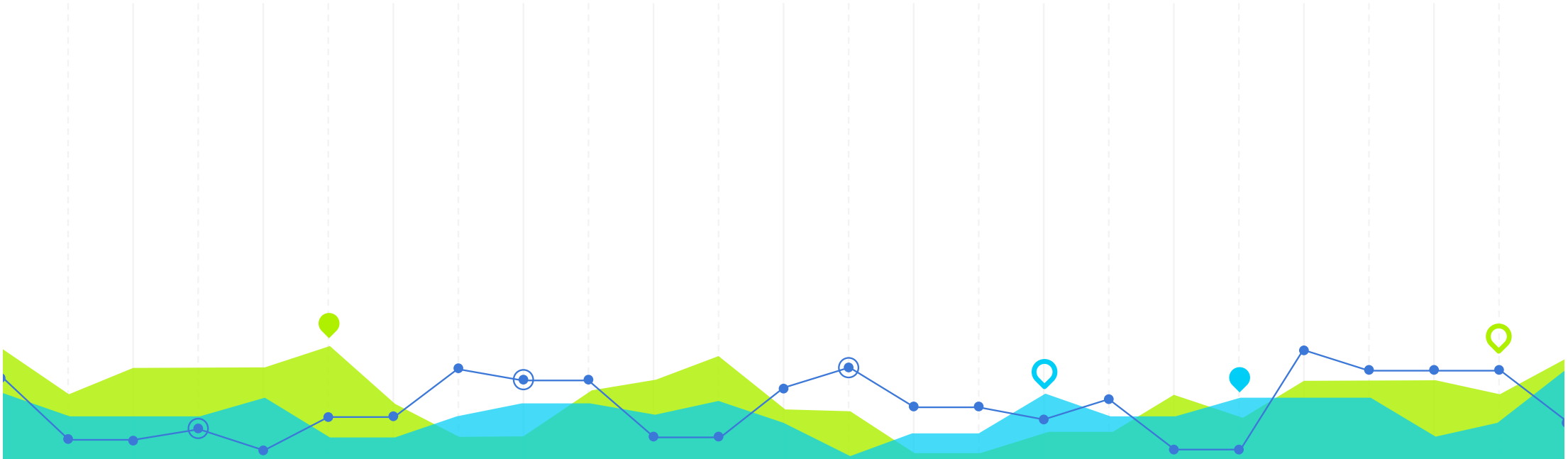
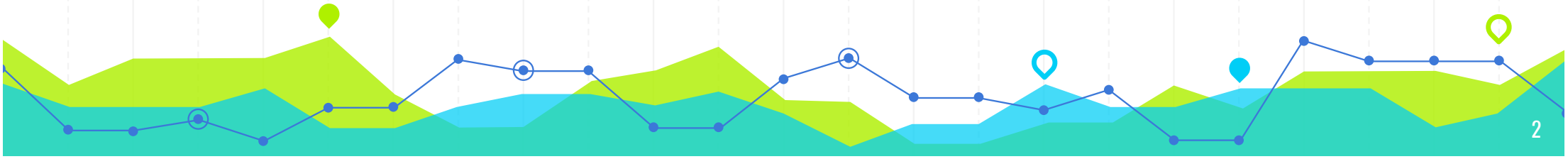


Lab 3: Intermediate SQL

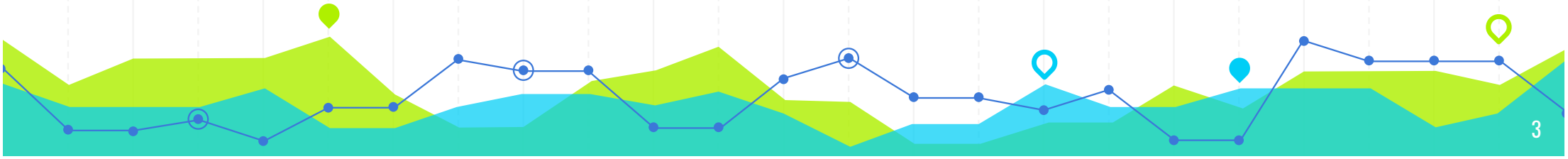
Queries on Multiple Relations

- **SELECT** * **FROM** instructor;
- **SELECT** * **FROM** department;
- **SELECT** * **FROM** instructor, department;
- **SELECT** name, course_id **FROM** instructor, teaches **WHERE** instructor.ID= teaches.ID;
- **SELECT** name, instructor.dept_name, building **FROM** instructor, department **WHERE** instructor.dept_name= department.dept_name;



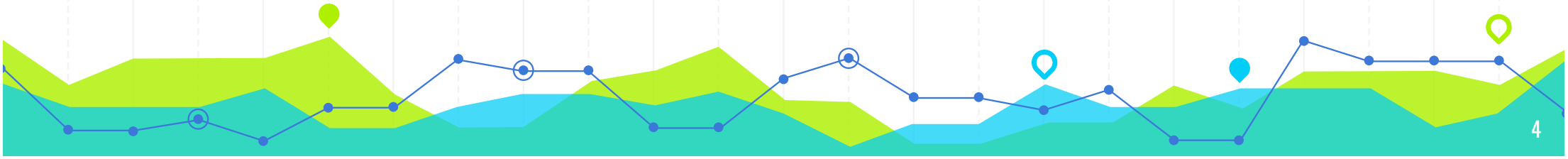
Natural join

- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught



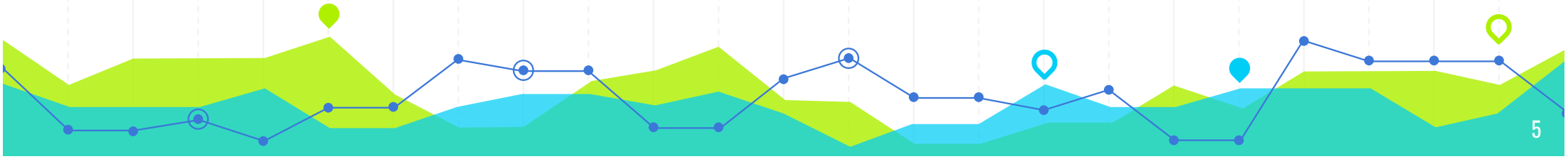
Natural join

- List the names of instructors along with the the titles of courses that they teach.”



The Rename Operation

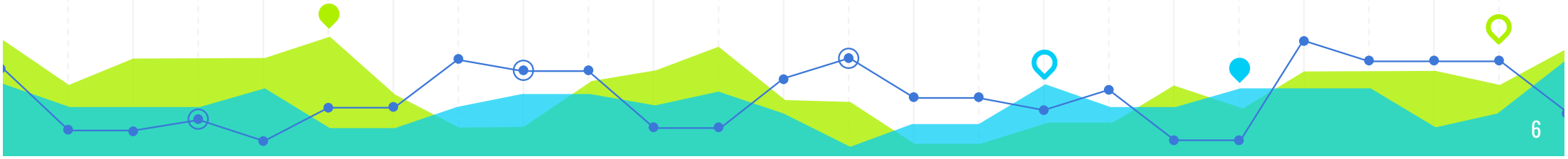
- **SELECT** name, course_id **FROM** instructor, teaches **WHERE** instructor.ID= teaches.ID;
- **SELECT** name as instructor_name, course_id **FROM** instructor, teaches **WHERE** instructor.ID= teaches.ID;



The Rename Operation

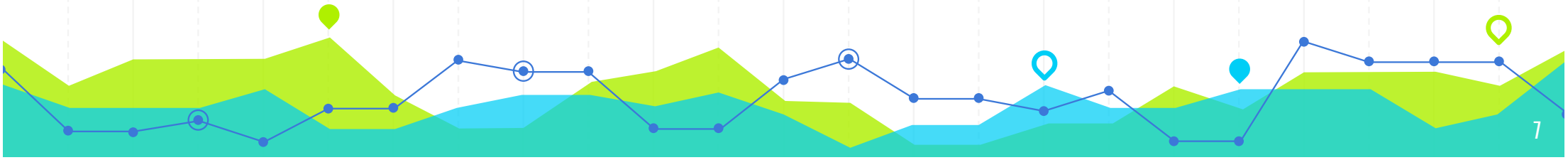
- For all instructors in the university who have taught some course, find their names and the course ID of all courses they taught.

- Find the names of all instructors whose salary is greater than at least one instructor in the Biology department.



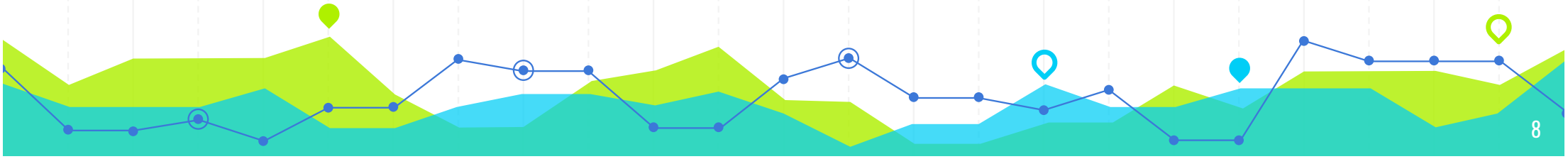
Join Conditions

- **SELECT** * **FROM** student, takes **WHERE** student.ID= takes.ID;
- **SELECT** * **FROM** student **JOIN** takes **ON** student.ID= takes.ID;
- **SELECT** * **FROM** student **NATURAL JOIN** takes **WHERE** student.ID= takes.ID;



The Outer Join

- Three types of outer joins: left, right, and full.
- The left and right designations reflect the order in which the tables are processed by the DBMS.
- The first table named in the FROM clause will be the left side, and the second table named will be the right side.
- If three or more tables are being joined, the result of joining the first two tables becomes the left side; the third table becomes the right side.

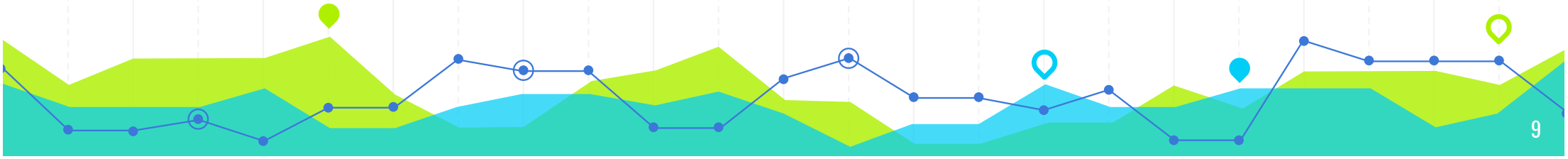


Outer Joins

- **SELECT** * **FROM** student **NATURAL JOIN** takes;
- **SELECT** * **FROM** student **NATURAL LEFT OUTER JOIN** takes;
- **SELECT** * **FROM** takes **NATURAL RIGHT OUTER JOIN** student;

Find all students who have not taken a course

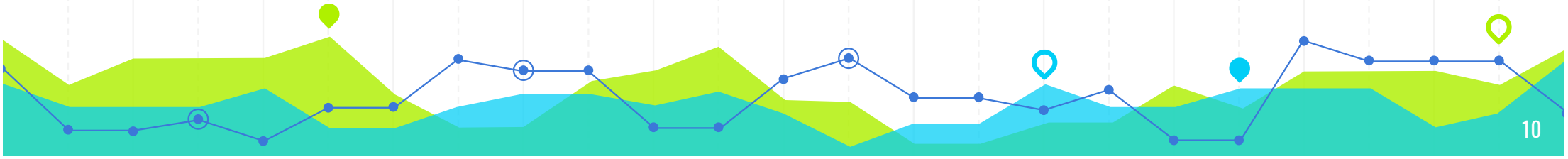
SELECT ID **FROM** student **NATURAL LEFT OUTER JOIN** takes
WHERE course_id is null;



Join Types and Conditions

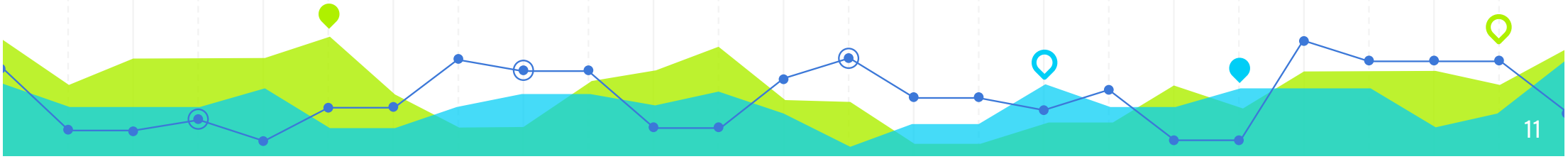
The query returns only the rows with matching values in the column indicated in the USING clause—and that column must exist in both tables.

- **SELECT * FROM** student **JOIN** takes **USING** (ID);
- **SELECT * FROM** student **INNER JOIN** takes **USING** (ID);



Homework

- Rewrite these statements without outer join
  - **SELECT * FROM** student **NATURAL LEFT OUTER JOIN** takes
  - **SELECT * FROM** student **NATURAL RIGHT OUTER JOIN** takes



Homework

- Display a list of all instructors, showing their ID, name, and the number of sections that they have taught.
- Display the list of all course sections offered in Spring 2010, along with the names of the instructors teaching the section. If a section has more than one instructor, it should appear as many times in the result as it has instructors.
- Display the list of all departments, with the total number of instructors in each department, without using scalar subqueries. Make sure to correctly handle departments with no instructors.