Programming Project 1 — Heuristic Query Optimization

CS 5300 – Database Systems (Fall 2025)

Instructor: Prof. San Yeung

Input Requirements

Your program must accept a **single text file (.txt)** containing both the **schema definitions** and the **SQL query**.

Input Rules and Assumptions

- Lines beginning with `--` are **comments** and should be ignored.
- Exactly **one query block** appears per file.
- Queries are assumed to be **syntactically valid SQL**.
- Schema definitions **precede** the SQL query.
- Attribute and table names are **case-insensitive**.
- Each relation lists its attributes in parentheses, separated by commas.
- Primary keys and unique keys are specified using `PRIMARY KEY(...)` and `UNIQUE(...)`.
    - Composite keys are supported by listing multiple attributes inside the parentheses, separated by commas.
- A terminating semicolon `;` is allowed at the end of each schema definition but not required.

Required Heuristic Optimization Rules

Your optimizer must implement and demonstrate, at minimum, the following steps:

Rule #	Heuristic Optimization Step	Description
1	**Cascade of Selections**	Break conjunctive selection conditions into a sequence of single-condition selections.
2	**Push Selections Down**	Move selections as close as possible to the base relations to reduce intermediate results.
3	**Apply Selections with Smallest Selectivity First**	Reorder leaf nodes and attached selections so that the most restrictive (smallest selectivity) filters are applied earliest.
4	**Replace Cartesian Product + Selection → Join**	Combine cross-products followed by join conditions into a single `⋈` operator.
5	**Push Projections Down**	Apply projections early to eliminate unnecessary attributes before joins.

Note: Heuristic Rule #3 (When No Selectivity Provided)

For this project, you may **assume that input files will not include numeric selectivity scores** for each selection condition.

Instead, your program should use **qualitative reasoning** to decide which selections are more restrictive and should be applied first.

This reasoning should be based on:

- The **predicate type** (equality vs. range), and
- The **schema information** provided in the input file (e.g., primary keys and unique keys).

Your program is **required to support** comparison operators (`=`, `<>`, `<`, `>`, `<=`, `>=`) combined using logical connectors (`AND`, `OR`). Other SQL predicate forms (such as `IN`, `EXISTS`, `BETWEEN`, `LIKE`, `IS NULL`, or `NOT`) are **not required** for this project except for the extra credit bonus.