# Lab 3
# Advanced Data Manipulation

CSE 4508

RELATIONAL DATABASE MANAGEMENT SYSTEM LAB

OCTOBER 7, 2024

# Contents

1 Join 2

2 String SQL Functions 4

3 Tasks 6

# 1 Join

Joins are used to combine rows from two or more tables based on related columns.

### INNER JOIN

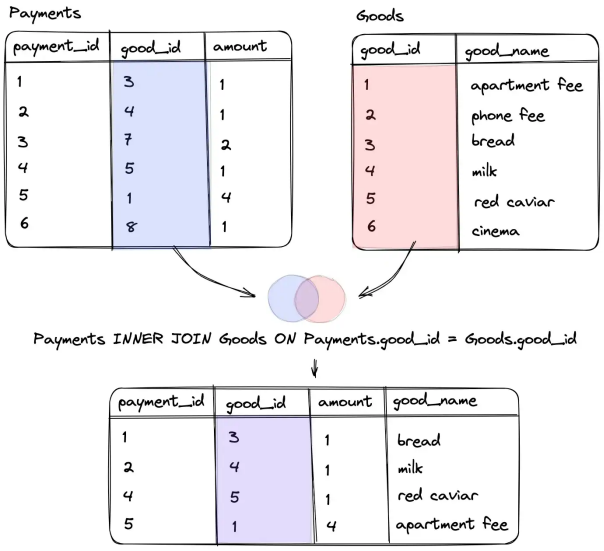


Figure 1: Sample of How Inner Join works

Another example:

```
SELECT S.STUDENT_NAME, C.COURSE_NAME
FROM STUDENT S
INNER JOIN ENROLLMENT E ON S.STUDENT_ID = E.STUDENT_ID
INNER JOIN COURSE C ON E.COURSE_ID = C.COURSE_ID;
```

This query aims to retrieve the names of students along with the courses they are enrolled in by joining three tables: **STUDENT**, **ENROLLMENT**, and **COURSE**.

| S_ID | S_NAME | DEPT |
|---|---|---|
| 1 | Alice | CSE |
| 2 | Bob | EEE |
| 3 | Carol | MCE |

Table 1: Student Table

| S_ID | C_ID |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 1 | 103 |
| 3 | 101 |

Table 2: ENROLLMENT Table

| C_ID | C_NAME | DEPT |
|---|---|---|
| 101 | Database Systems | CSE |
| 102 | Circuit Analysis | EEE |
| 103 | Operating Systems | CSE |

Table 3: COURSE Table

Table 4: Before Tables

| S_ID | S_NAME | DEPT | C_ID |
|---|---|---|---|
| 1 | Alice | CSE | 101 |
| 1 | Alice | CSE | 103 |
| 2 | Bob | EEE | 102 |
| 3 | Carol | MCE | 101 |

Table 5: Intermediate Result (Table A)

| S_ID | S_NAME | DEPT | C_ID | C_NAME |
|---|---|---|---|---|
| 1 | Alice | CSE | 101 | Database Systems |
| 1 | Alice | CSE | 103 | Operating Systems |
| 2 | Bob | EEE | 102 | Circuit Analysis |
| 3 | Carol | MCE | 101 | Database Systems |

Table 6: Intermediate Result (Table B)

| S_NAME | C_NAME |
|---|---|
| Alice | Database Systems |
| Alice | Operating Systems |
| Bob | Circuit Analysis |
| Carol | Database Systems |

Table 7: Final Result: After INNER JOIN with COURSE

## NATURAL JOIN

Natural join is an SQL join operation that creates a join on the base of the common columns in the tables. To perform natural join there must be one common attribute(Column) between two tables. Natural join will retrieve from multiple relations. It works in three steps.

```
SELECT *
FROM TABLE1
NATURAL JOIN TABLE2;
```

- A natural join matches columns with the same names in both tables and only includes rows where the values in those columns are equal (consistent tuples).
- It excludes rows where the values don't match (inconsistent tuples).
- After joining, it removes one of the duplicate columns from the result set.

```
SELECT *
FROM STUDENT
NATURAL JOIN ENROLLMENT;
```

| S_ID | S_NAME | DEPT |
|---|---|---|
| 1 | Alice | CSE |
| 2 | Bob | EEE |
| 3 | Carol | MCE |

Table 8: Student Table

| S_ID | C_ID |
|---|---|
| 1 | 101 |
| 2 | 102 |
| 1 | 103 |
| 3 | 101 |

Table 9: ENROLLMENT Table

| C_ID | C_NAME | DEPT |
|---|---|---|
| 101 | Database Systems | CSE |
| 102 | Circuit Analysis | EEE |
| 103 | Operating Systems | CSE |

Table 10: COURSE Table

Table 11: Before Tables

| S_ID | S_NAME | DEPT | C_ID |
|---|---|---|---|
| 1 | Alice | CSE | 101 |
| 1 | Alice | CSE | 103 |
| 2 | Bob | EEE | 102 |
| 3 | Carol | MCE | 101 |

Table 12: After Natural Join

# 2 String SQL Functions

1. **CONCAT**: Used to concatenate two or more strings.

```
SELECT CONCAT('Hello, ', 'World!') AS Greeting
FROM dual;
```

2. **INITCAP**: Capitalizes the first letter of each word in a string.

```
SELECT INITCAP('hello world') AS Proper_Case
FROM dual;
```

3. **LENGTH**: Returns the length of a string.

```
SELECT LENGTH('Hello, World!') AS String_Length
FROM dual;
```

4. **UPPER and LOWER**: Converts text to uppercase or lowercase.

```
SELECT UPPER('hello world') AS Uppercase_Text
FROM dual;

SELECT LOWER('HELLO WORLD') AS Lowercase_Text
FROM dual;
```

5. **SUBSTR**: Extracts a portion of a string.

```
SELECT SUBSTR('Hello, World!', 1, 5) AS Substring
FROM dual;
```

6. **INSTR**: Returns the position of a substring in a string.

```
SELECT INSTR('Hello, World!', 'World') AS Position
FROM dual;
```

7. **LPAD and RPAD**: Pads a string to the left or right with a specified character.

```
SELECT LPAD('Hello', 10, '*') AS Left_Padded
FROM dual;

SELECT RPAD('Hello', 10, '*') AS Right_Padded
FROM dual;
```

8. **LTRIM and RTRIM**: Removes leading or trailing spaces from a string.

```
SELECT LTRIM('   Hello') AS Trimmed_Left
FROM dual;

SELECT RTRIM('Hello   ') AS Trimmed_Right
FROM dual;
```

9. **COUNT**: Counts the number of rows returned.

```
SELECT COUNT(*) AS Total_Rows
FROM dual;
```

# 3 Tasks

1. Run the .sql file to create the tables and insert data.
2. Display the Name, Customer ID, and Lifetime Usage of the top 5 highest users.
3. Display the name, date of birth, district, and division of all customers whose lifetime usage exceeds the average usage of Prepaid users with Silver status.
4. Count how many of the customers from the previous query are from Dhaka.
5. Write a query to display the full name of each customer, with the prefix "Mr./Ms." concatenated.
6. Write a query to display customer names with the first letter of each word capitalized.
7. Write a query to find customers whose names contain the substring "an". Return the position where this substring appears.
8. Write a query to display all customer names in lowercase.
9. Write a query to display all customer names in uppercase.
10. Write a query to display the length of each customer's name.
11. Write a query to display the customer names padded to the left with asterisks (*) to make the total length 15 characters.
12. Write a query to display the customer names after trimming any leading whitespace.
13. Write a query to display the first 5 characters of each customer's name.
14. Write a query to count the number of customers from each division.

# Submission Guidelines

Submit a `.sql` file containing all your code.