Writing SQL Queries - II

CO226 : Database Systems

Lab - 06

Kalani Udayanthi
kalaniu@eng.pdn.ac.lk

Outline

- Relating Multiple Tables
- Sub-Queries
- GROUP BY Clause
- HAVING Clause
- Numeric Expressions
- SQL Functions
  - ➔ Arithmetic functions
  - ➔ String functions
  - ➔ Date functions
  - ➔ Aggregate functions
- SQL Joins

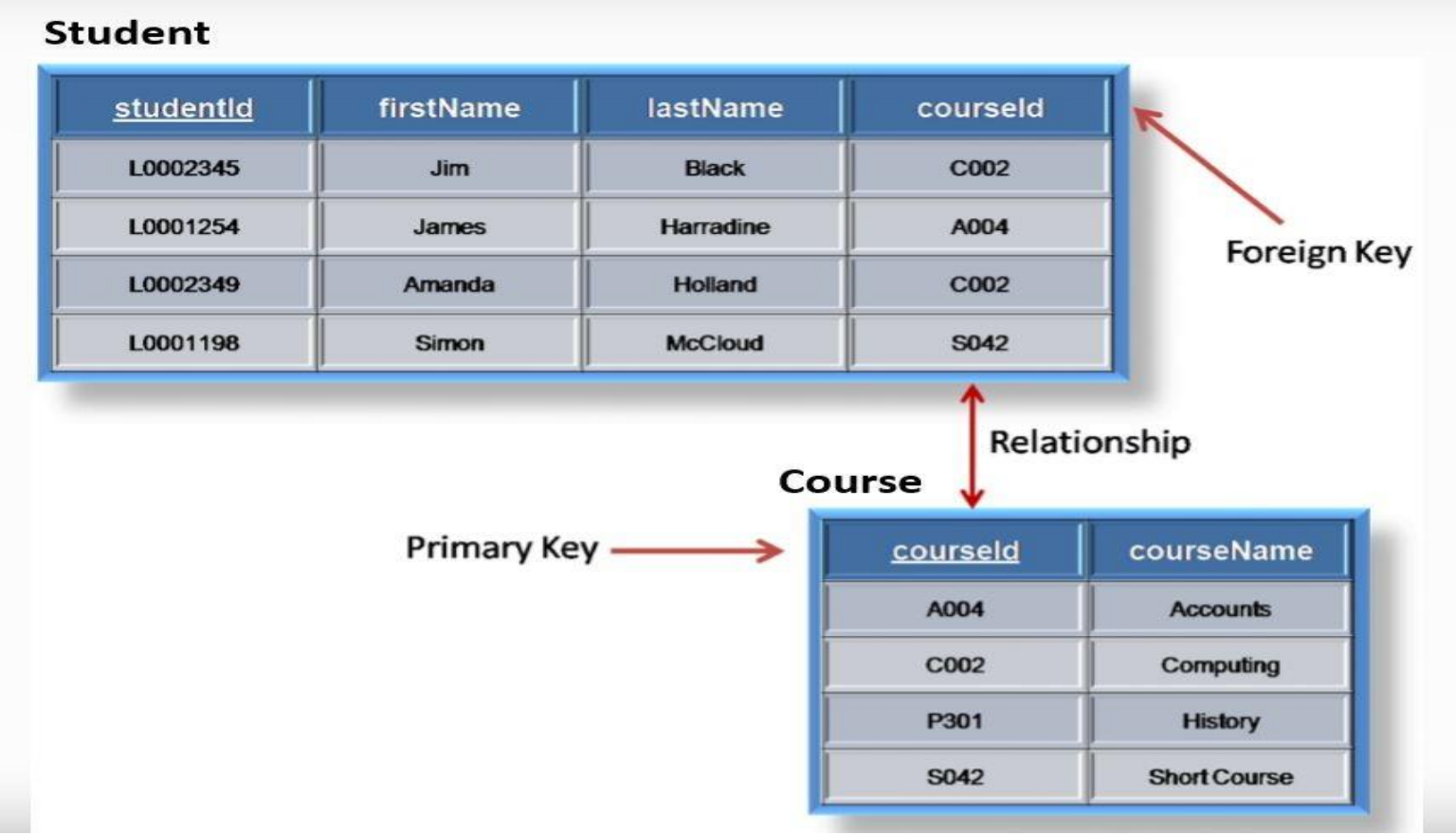
Relating Multiple Tables

- How do you write the query to display the details in Student and Course tables???
  - ➢ Check the primary keys and foreign keys
  - ➢ Relate the tables using those keys

Syntax :

```
SELECT < FieldNames >
FROM < Table1>,<Table2 >
WHERE < Table1>.< KeyField > = < Table2>.< KeyField >;
```

Example Database …



Example :

```
SELECT *
FROM Student, Course
WHERE Student.CourseID = Course .CourseID;
```

or

```
SELECT StudentID , FirstName , LastName , CourseID ,
CourseName  FROM Student , Course
WHERE Student.CourseID = Course.CourseID ;
```

Relating Multiple Tables cont.

- How do you select specific rows from multiple tables???
  - Combine the condition using AND in the WHERE clause

Syntax:

```
SELECT < FieldNames >
FROM < Table1>,<Table2>
WHERE < Table1>.< KeyField >= < Table2>.< KeyField >AND <Condition> ;
```

Example :

1. Select StudentID , FirstName , LastName , CourseID , CourseName from Student and Course tables where CourseName = “Computing”.

```
SELECT StudentID , FirstName , LastName , CourseID ,
CourseName FROM Student , Course
WHERE Student.CourseID = Course.CourseID AND CourseName = ‘Computing’ ;
```

Activity 01

BookShop Database

Book

BookID	Title	Author	Category	Price (Rs.)	PubID
B1	Access	KV Peter	2	1750.00	P02
B2	Python for Data Science	D Alex	1	3450.00	P01
B3	Algorithms	John Allen	1	1280.00	P03
B4	Excel	Alexander	3	2700.00	P01

Publisher

PubID	PublisherName	RegisteredDate	Country
P01	Atlas	2012-05-07	India
P02	B.Press	2009-08-17	USA
P03	PBP	2015-10-14	UK

Questions ??????

1. Display all the details in Book and Publisher tables.
2. Display titles of the books with the relevant publisher names.
3. View titles of books where publishers are from India.
4. Display book titles and prices where the relevant publisher registered before 2013-01-01.
5. Display Book ID, Title, Author, Category, Price and Publisher ID of the book published by ‘B.Press’.
6. Display publisher details of books which are not belong category 1.

1. Display all the details in Book and Publisher tables.

   Answer :

```
SELECT *
FROM   Book, Publisher
WHERE  Book.PubID = Publisher.PubID ;
```

2. Display titles of the books with the relevant publisher names.

   Answer :

```
SELECT   Title, PublisherName
FROM     Book, Publisher
WHERE  Book.PubID = Publisher.PubID ;
```

3. View titles of books where publishers are from India.

   Answer :

```
SELECT Title, Country
FROM   Book,
Publisher
WHERE  Book.PubID = Publisher.PubID AND
                    Publisher.Country='India' ;
```

4. Display book titles and prices where the relevant publisher registered before 2013-01-01.

   Answer :

```
SELECT Title, Price(Rs.)
FROM   Book, Publisher
WHERE  Book.PubID = Publisher.PubID AND
                    Publisher.RegisteredDate<'2013-01-01';
```

5. Display Book ID, Title, Author, Category, Price and Publisher ID of the book published by ‘B.Press’.

Answer :

```
SELECT   BookId, Title, Author, Category, Price(Rs.), PubID
FROM     Book, Publisher
WHERE    Book.PubID = Publisher.PubID AND
                      Publisher.PublisherName=’B.Press’
                      ;
```

6. Display publisher details of books which are not belong category 1.

Answer :

```
SELECT   *
FROM     Publisher p
WHERE    Book.PubID = Publisher.PubID AND NOT
         Book.Category=1;
```

Sub-Queries

A subquery is a SQL query nested inside a larger query.

- A subquery may occur in :
  - A SELECT clause
  - A FROM clause
  - A WHERE clause
- The subquery can be nested inside a SELECT, INSERT, UPDATE, or DELETE statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.
- A subquery is also called an inner query or inner select, while the statement containing a subquery is also called an outer query or outer select.

SubQuery - Syntax

```
SELECT    select_list
FROM      table
WHERE     expr operator
                (SELECT    select_list
                FROM       table);
```

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

SQL SubQueries example:

1. Write a query to identify all students who get better marks than that of the student who's StudentID is 'V002'.

Student

StudentID	Name
V001	Abe
V002	Abhay
V003	Acelin
V004	Adelphos

Mark

StudentID	Total_marks
V001	95
V002	80
V003	74
V004	81

To solve the problem, we require two queries.

- One query returns the marks (stored in Total_marks field) of 'V002' and
- Second query identifies the students who get better marks than the result of the first query.

First query:

```
1  SELECT *
2  FROM `marks`
3  WHERE studentid = 'V002';
```

Query result:

StudentID	Total_marks
V002	80

Second query:

```
1  SELECT a.studentid, a.name, b.total_marks
2  FROM student a, marks b
3  WHERE a.studentid = b.studentid
4  AND b.total_marks >80;
```

Query result:

studentid	name	total_marks
V001	Abe	95
V004	Adelphos	81

Above two queries identified students who get the better number than the student who's StudentID is 'V002' (Abhay).

You can combine the above two queries by placing one query inside the other. The subquery (also called the 'inner query') is the query inside the parentheses. See the following code and query result :

SQL Code:

```
1  SELECT a.studentid, a.name, b.total_marks
2  FROM student a, marks b
3  WHERE a.studentid = b.studentid AND b.total_marks >
4  (SELECT total_marks
5  FROM marks
6  WHERE studentid =  'V002');
```

Query result:

studentid	name	total_marks
V001	Abe	95
V004	Adelphos	81

GROUP BY Clause

- The **GROUP BY** statement is used in conjunction with the aggregate functions to group the result-set by one or more columns
  - ➔ An **aggregate function** performs a calculation on a set of values, and returns a single value. Except for COUNT, aggregate functions ignore null values. Aggregate functions are often used with the GROUP BY clause of the SELECT statement.
  - ➔ All aggregate functions are deterministic. In other words, aggregate functions return the same value each time that they are called, when called with a specific set of input values.

Syntax:

```
SELECT <FieldName>, <AggregateFunction>
FROM <TableName>
GROUP BY <GroupingField>;
```

Example :

1. Display the group number and the total of the marks obtained by each group in the class

```
SELECT GroupNo, SUM(Marks)
FROM Student
GROUP BY GroupNo;
```

HAVING Clause

- The MySQL **HAVING** clause is used in the SELECT statement to specify filter conditions for group of rows or aggregates

Syntax:

```
SELECT <FieldName >, <AggregateFunction >
FROM <TableName >
GROUP BY <GroupingField>
HAVING <Condition>;
```

Example :

1. Display the group number and the total of the marks obtained by each group in the class where the total mark is greater than 50

```
SELECT GroupNo , SUM(Marks)
FROM Student
GROUP BY GroupNo
HAVING SUM(Marks)> 50 ;
```

Numeric Expressions

- A book seller needs to increase the book prices by $1. How do you display the new price of all books along with their titles and existing price?

- Eg:

```
mysql > SELECT Title, Price, Price + 1
        FROM Book ;
```

Functions - Introduction

- Functions uses this format:
  - function(argument)

- Can have multiple arguments:
  - function(argument1, argument1, ……. argN)

Functions - Introduction cont.

- Functions can be categorized into several ways:
  - Arithmetic functions
  - String functions
  - Date functions
  - Aggregate functions

- Functions can be categorized by number of rows affected:
  - Individual functions
  - Group functions

String Functions

- **UPPER(FieldName)**
  - Returns the strings in upper case in the given field

    Eg: *UPPER (‘functions’)* => *FUNCTIONS*

- **LOWER(FieldName)**
  - Returns the strings in upper case in the given field

    Eg: *LOWER (‘FUNCTIONS’)* => *functions*

String Functions cont.

- **LENGTH(FieldName)**
  - Computes the number of characters in the string of the given field

    Eg: *LENGTH ('Hello')* => *5*

- **SUBSTR(FieldName, Position of starting character, Number of characters to be displayed (optional))**
  - List characters starting from the given character up to the given number

    Eg: *SUBSTR ('GoodMorning', 2, 4)* => *oodM*

    (2 →Starting character, 4 →No. of characters need to display)

String Functions cont.

- **CONCAT(FieldName1, FieldName2)**
  - Concatenate the strings in the given fields

    Eg: *CONCAT (‘Good', ‘Morning’)* => *GoodMorning*

- **LOCATE(‘String’, FieldName)**
  - Returns the position of the first string argument in the second string argument

    Eg: *LOCATE (‘ce’, ‘Science’)* => *6*

Arithmetic Functions

- **ABS(NumericFieldName(s))**
  - Returns the numerical value(absolute value) of a number given in the argument without regard of its sign

    Eg: *ABS (- 32)* => *32*

- **MOD(NumericFieldName1, NumericArgument)**
  - Returns the remainder of two values

    Eg: *MOD (25, 4)* => *1*

Arithmetic Functions cont.

- **CEIL(NumericFieldName)**
  - Returns the smallest integer that is greater than or equal to the value of the argument

    Eg: *CEIL (34.2)* *=> 35*

- **FlOOR(NumericFieldName)**
  - Returns the largest integer that is smaller than or equal to the value of the argument

    Eg: *FLOOR (34.2)* *=> 34*

Arithmetic Functions cont.

- **POW(NumericFieldName, Power)**
  - Return the argument raised to the specified power

    Eg: *POW (3,2)* *=> 9*

- **ROUND(NumericFieldName, DecimalPlaces)**
  - Returns the rounded value of the given decimal number

    Eg: *ROUND (34.2345, 3)* *=> 34.235*

Arithmetic Functions cont.

- **SQRT(NumericFieldName)**
  - Returns the square root of the given number

Eg: *SQRT (4)* *=> 2*

Date Functions

- **CURDATE()**
  - Returns the current date

    Eg: *CURDATE()* *=>2015 - 06 - 11*

- **CURTIME()**
  - Returns the current time

    Eg: *CURTIME ()* *=>10:00:00*

- **NOW()**
  - Returns the current date time value

    Eg: *NOW ()* *=>2015 - 06 - 1110:00:00*

Date Functions cont.

- **ADDDATE(DateField, NumericValue)**
  - Returns the new date after adding the numeric value to the date argument

    Eg: *ADDDATE ('2015 - 05 - 10', 10)* *=> 2015 - 05 - 20*

- **DATEDIFF(DateField1, DateField2)**
  - Returns the difference between the 2 dates given in the arguments

    Eg: *DATEDIFF ('2015 - 07 - 10', '2015 - 06 - 10')* *=> 30*

Date Functions cont.

- **DAYNAME(DateField)**
  - Returns the day of the week of the given date

    Eg : *DAYNAME (‘2015 - 07 - 10’)* => *Friday*

- **DAYOFYEAR(DateField)**
  - Returns the day of the year

    Eg : *DAYOFYEAR(‘2015 - 01 - 20’)* => *20*

Aggregate Functions

Aggregate functions retrieve a single value after performing a calculation on a set of values

- **COUNT(FieldName)**
  - Counts the rows in the given field

    Eg: COUNT(firstName) or COUNT(*)

- **MAX(FieldName)**
  - Finds the maximum value in the given field

    Eg: MAX(Age)

Aggregate Functions cont.

- **MIN(FieldName)**
  - Finds the minimum value in the given field

    Eg: MIN(Age)

- **SUM(FieldName)**
  - Calculate the sum of the given field

    Eg: SUM(Marks)

- **AVG(FieldName)**
  - Calculate the average of the given field

    Eg: AVG(Marks)

Activity 2

Filling the blanks with appropriate functions.

1. _____(‘Hello', ‘World’) => HelloWorld
2. _____(‘se’, ‘Database’) => 7
3. _____(‘functions’) => FUNCTIONS
4. _____(‘Aggregate’) => 9
5. SUBSTR (‘SriLanka’, 3, 5) => _____

Activity 3

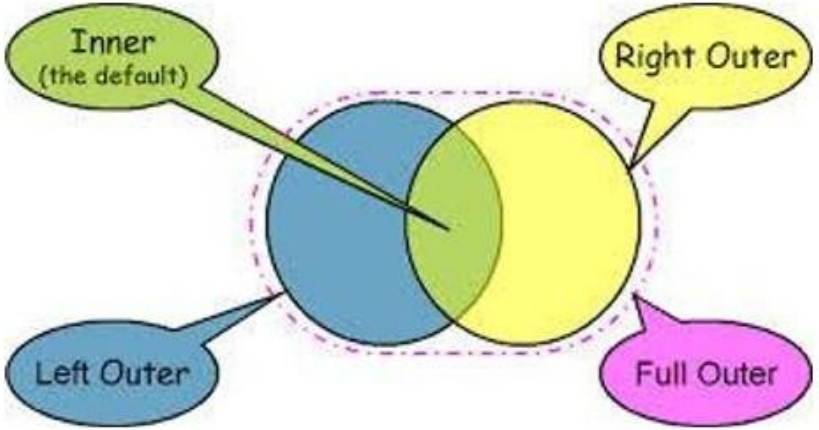
Write the functions,

1. To find absolute value of (-56)?
2. To return the remainder of (86 mod 6)?
3. To return the rounded value of 57.456985 to 4 decimal points?
4. To find the square root of 25?
5. To return the current date?
6. To return the current date and time?
7. To return the day of ‘2020-05-13’?

SQL Joins

SQL Joins

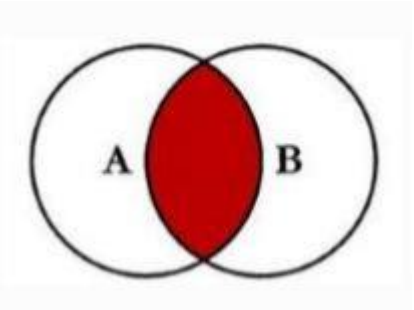
- Used to combine rows from two or more tables, based on a common field between the tables
- Types of Joins:
  - Inner Join
  - Left Join
  - Right Join
  - Full Outer Join



INNER Join

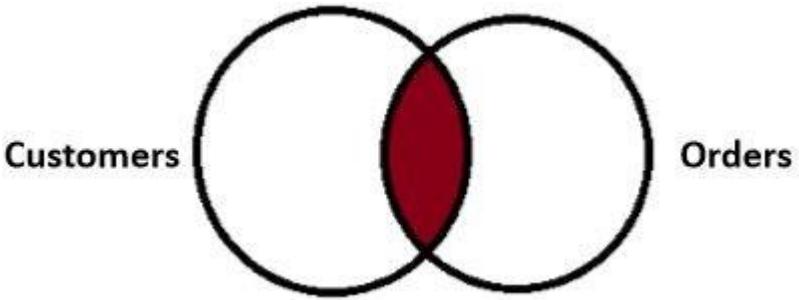
- Returns all rows when there is at least one match in BOTH tables

```
SELECT <select_list >
FROM TableA
INNER JOIN TableB
ON TableA.Key = TableB.Key ;
```



Question 1

Produce a set of records which match in both the Customer table and Orders tables(All customers who are placed and order)



Example :

Customer_ID	Customer_Name	Email	Mobile_Number	Address
C0001	A.W.Perera	awperera@gmail.com	115678978	Colombo
C0002	R.J.A.Rathnayake	ratnayakerj@gmail.com	315672976	Negombo
C0003	U.P.Upul	upul@gmail.com	115678978	Colombo
C0004	K.L.E.Karunarathna	Karukle@gmail.com	916767908	Galle
C0010	K.Rajan	krajan@hotmail.com	260345908	Trincomalee
C0031	S.R.Hettige	srhettige@gmail.com	556456478	Badulla

Order_ID	Order_Date	Amount	Customer_ID
O0001	2010-01-05	2500	C0001
O0002	2015-01-15	24000	C0005
O0003	2014-12-15	34000	C0003
O0004	2014-06-17	3600	C0010
O0005	2013-06-27	54000	C0011
O0006	2016-03-15	4000	C0006

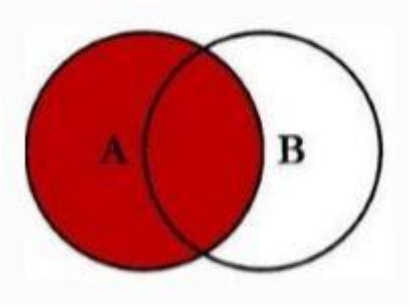
SELECT Customer_Name ,Address,Order_Date,Amount FROM Customer **INNER JOIN** Orders **ON** Customer.Customer_ID=Orders.Customer_ID

Customer_Name	Address	Order_Date	Amount
A.W.Perera	Colombo	2010-01-05	2500
U.P.Upul	Colombo	2014-12-15	34000
K.Rajan	Trincomalee	2014-06-17	3600

LEFT Join

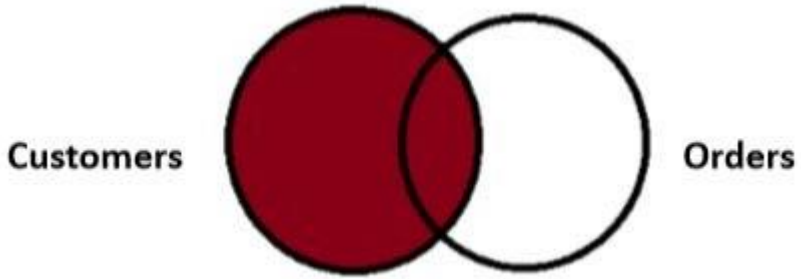
- Return all rows from the left table, and the matched rows from the right table

```
SELECT <select_list >
FROM TableA
LEFT JOIN TableB
ON TableA.Key = TableB.Key ;
```



Question 2

Produce a set of records which matches every entry in the Customer table regardless of any matching entry in the Orders table



Output :

Customer_ID	Customer_Name	Email	Mobile_Number	Address
C0001	A.W.Perera	awperera@gmail.com	115678978	Colombo
C0002	R.J.A.Rathnayake	ratnayakerj@gmail.com	315672976	Negombo
C0003	U.P.Upul	upul@gmail.com	115678978	Colombo
C0004	K.L.E.Karunarathna	Karukle@gmail.com	916767908	Galle
C0010	K.Rajan	krajan@hotmail.com	260345908	Trincomalee
C0031	S.R.Hettige	srhettige@gmail.com	556456478	Badulla

Order_ID	Order_Date	Amount	Customer_ID
O0001	2010-01-05	2500	C0001
O0002	2015-01-15	24000	C0005
O0003	2014-12-15	34000	C0003
O0004	2014-06-17	3600	C0010
O0005	2013-06-27	54000	C0011
O0006	2016-03-15	4000	C0006

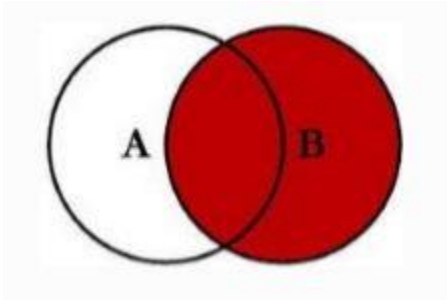
SELECT Customer_Name ,Address,Order_Date,Amount FROM Customer **LEFT JOIN** Orders **ON** Customer.Customer_ID=Orders.Customer_ID

Customer_Name	Address	Order_Date	Amount
A.W.Perera	Colombo	2010-01-05	2500
R.J.A.Rathnayake	Negombo	NULL	NULL
U.P.Upul	Colombo	2014-12-15	34000
K.L.E.Karunarathna	Galle	NULL	NULL
K.Rajan	Trincomalee	2014-06-17	3600
S.R.Hettige	Badulla	NULL	NULL

RIGHT Join

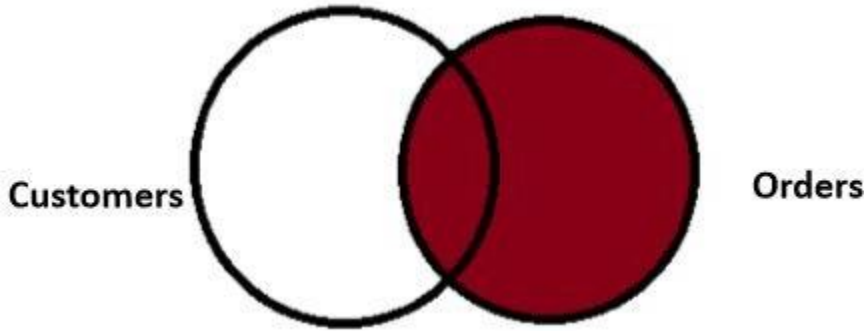
- Return all rows from the right table, and the matched rows from the left table

```
SELECT <select_list >
FROM TableA
RIGHT JOIN TableB
ON TableA.Key = TableB.Key ;
```



Question 3

Produces a set of records which matches every entry in the Orders regardless of any matching entry in the Customers.



Output :

Customer_ID	Customer_Name	Email	Mobile_Number	Address
C0001	A.W.Perera	awperera@gmail.com	115678978	Colombo
C0002	R.J.A.Rathnayake	ratnayakerj@gmail.com	315672976	Negombo
C0003	U.P.Upul	upul@gmail.com	115678978	Colombo
C0004	K.L.E.Karunarathna	Karukle@gmail.com	916767908	Galle
C0010	K.Rajan	krajan@hotmail.com	260345908	Trincomalee
C0031	S.R.Hettige	srhettige@gmail.com	556456478	Badulla

Order_ID	Order_Date	Amount	Customer_ID
O0001	2010-01-05	2500	C0001
O0002	2015-01-15	24000	C0005
O0003	2014-12-15	34000	C0003
O0004	2014-06-17	3600	C0010
O0005	2013-06-27	54000	C0011
O0006	2016-03-15	4000	C0006

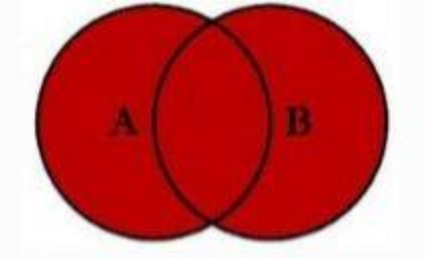
```
SELECT  Customer_Name ,Address,Order_Date,Amount FROM Customer RIGHT JOIN
Orders ON Customer.Customer_ID=Orders.Customer_ID
```

Customer_Name	Address	Order_Date	Amount
A.W.Perera	Colombo	2010-01-05	2500
NULL	NULL	2015-01-15	24000
U.P.Upul	Colombo	2014-12-15	34000
K.Rajan	Trincomalee	2014-06-17	3600
NULL	NULL	2013-06-27	54000
NULL	NULL	2016-03-15	4000

FULL OUTER Join

- Return all rows from the right table, and the matched rows from the left table

```
SELECT <select_list >
FROM TableA
FULL OUTER JOIN TableB
ON TableA.Key = TableB.Key ;
```



Output :

Customer_ID	Customer_Name	Email	Mobile_Number	Address
C0001	A.W.Perera	awperera@gmail.com	115678978	Colombo
C0002	R.J.A.Rathnayake	ratnayakerj@gmail.com	315672976	Negombo
C0003	U.P.Upul	upul@gmail.com	115678978	Colombo
C0004	K.L.E.Karunarathna	Karukle@gmail.com	916767908	Galle
C0010	K.Rajan	krajan@hotmail.com	260345908	Trincomalee
C0031	S.R.Hettige	srhettige@gmail.com	556456478	Badulla

Order_ID	Order_Date	Amount	Customer_ID
O0001	2010-01-05	2500	C0001
O0002	2015-01-15	24000	C0005
O0003	2014-12-15	34000	C0003
O0004	2014-06-17	3600	C0010
O0005	2013-06-27	54000	C0011
O0006	2016-03-15	4000	C0006

```
SELECT Customer_Name ,Address,Order_Date,Amount FROM Customer LEFT JOIN Orders ON
Customer.Customer_ID=Orders.Customer_ID UNION SELECT Customer_Name ,Address,Order_Date,Amount FROM Customer
RIGHT JOIN Orders ON Customer.Customer_ID=Orders.Customer_ID
```

Customer_Name	Address	Order_Date	Amount
A.W.Perera	Colombo	2010-01-05	2500
R.J.A.Rathnayake	Negombo	NULL	NULL
U.P.Upul	Colombo	2014-12-15	34000
K.L.E.Karunarathna	Galle	NULL	NULL
K.Rajan	Trincomalee	2014-06-17	3600
S.R.Hettige	Badulla	NULL	NULL
NULL	NULL	2015-01-15	24000
NULL	NULL	2013-06-27	54000
NULL	NULL	2016-03-15	4000

Summary

- Relating Multiple Tables
- Sub-Queries
- GROUP BY Clause
- HAVING Clause
- Numeric Expressions
- SQL Functions
  - ➔ Arithmetic functions
  - ➔ String functions
  - ➔ Date functions
  - ➔ Aggregate functions
- SQL Joins

- Reference for SQL Functions:

  https://www.geeksforgeeks.org/sql-numeric-functions/?ref=lbp

  https://www.geeksforgeeks.org/sql-string-functions/?ref=lbp

- Reference for SQL Joins(Q&A):

  https://www.w3resource.com/sql-exercises/sql-joins-exercises.php

- Reference for SQL Subqueries(Q&A):

  https://www.w3resource.com/sql-exercises/subqueries/index.php