Joins - I

Relevel
by Unacademy

What is JOIN?

Join is the most commonly used clause in SQL Server, and it is used to combine and retrieve data from two or more tables.

Data in a real-world relational database is structured in many tables, which necessitates the constant need to join these multiple tables based on logical relationships. The different types of Joins are:

- INNER JOIN;
- LEFT JOIN;
- RIGHT JOIN;
- FULL JOIN.

Why do we need JOIN?

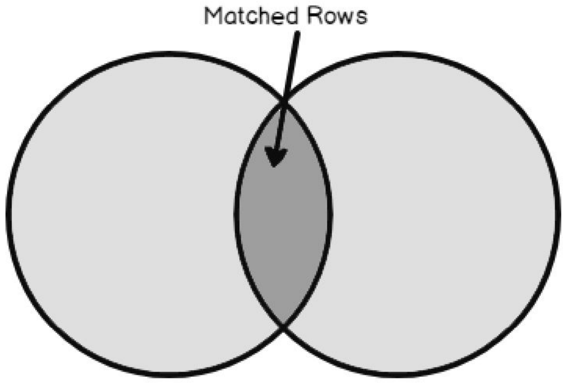
- Data in a real-world relational database is organized into many tables (star/snowflake schema), which is why there is a constant need to join these multiple tables based on logical relationships between them.

- Storing all the data in one table will make the query slow, and we keep limited information in a table and can retrieve it from another table, per need, using join.

INNER JOIN

In SQL Server, the Inner Join clause creates a new table (not physical) by combining rows with matching values from two or more tables.

Assume we have two tables, A and B, that we want to join using SQL Inner Join. This join will produce a new result set with matching rows from both tables.



SQL Inner Join

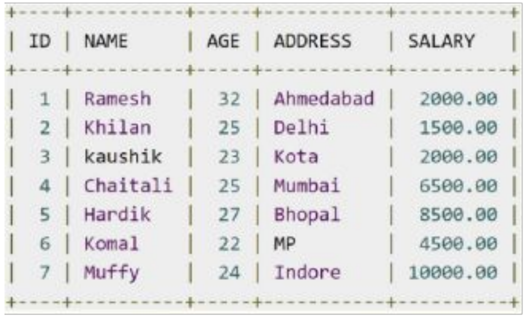
Basic Syntax of Inner Join

```
SELECT
          Column_list
FROM
          TABLE1
INNER JOIN
          TABLE2
ON Table1.ColName = Table2.ColName
```

Example of Inner Join

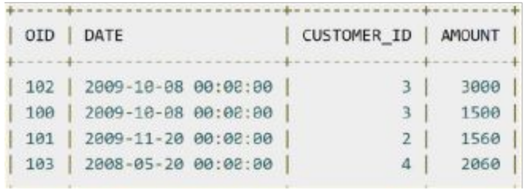
Question - Write a query to find amount spend by each customer

CUSTOMERS Table



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

ORDERS Table

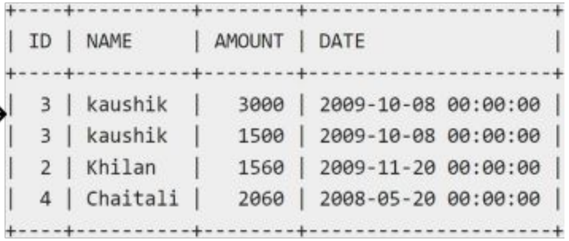


```
+-----+---------------------+-------------+--------+
| OID | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
```

Query

```
SELECT ID, NAME, AMOUNT, DATE
FROM CUSTOMERS
INNER JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

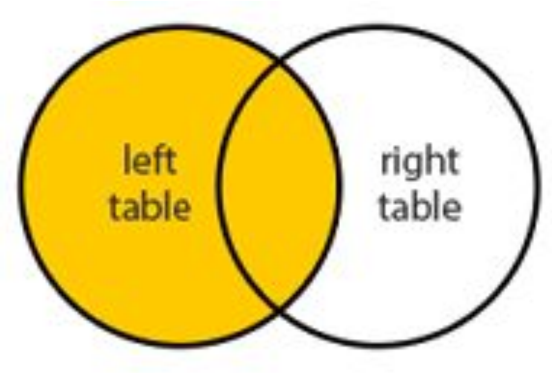
Output



```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+----+----------+--------+---------------------+
```

LEFT JOIN

SQL Left Join returns all records from the left table in the join clause, regardless of whether there are any matching records in the right table. The left SQL outer join includes all rows from the table on the left where the condition is met and all rows from the table on the left where the condition is not met. Fields from the correct table that do not match will have null values.



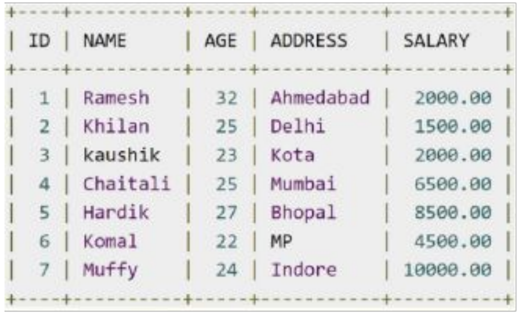
Basic Syntax of Left Join

```
SELECT
        Column_list
FROM
        TABLE1
LEFT JOIN
        TABLE2
ON Table1.ColName = Table2.ColName
```

Example of Left Join

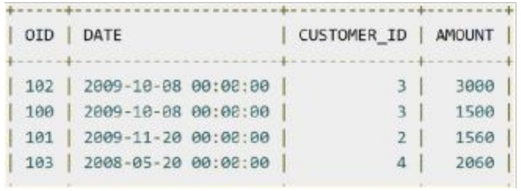
Question - Write a query to fetch customer name amount spent and date of order of each customer.

CUSTOMERS(Left) Table



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

ORDERS(Right) Table)

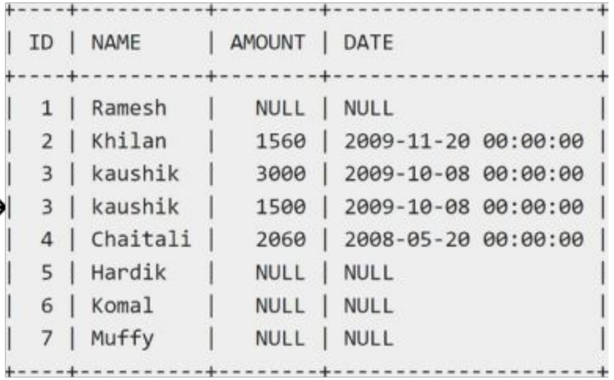


```
+-----+---------------------+-------------+--------+
| OID | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
```

Query

```
SELECT ID, NAME, AMOUNT, DATE  FROM CUSTOMERS
LEFT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

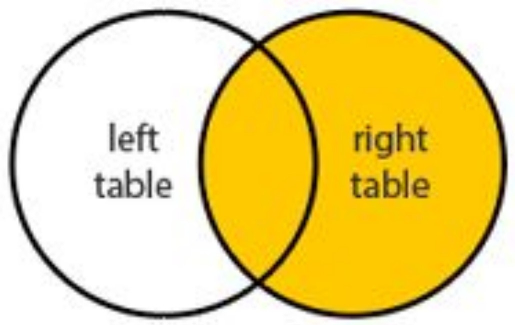
Output



```
+----+----------+--------+---------------------+
| ID | NAME     | AMOUNT | DATE                |
+----+----------+--------+---------------------+
|  1 | Ramesh   |   NULL | NULL                |
|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|  5 | Hardik   |   NULL | NULL                |
|  6 | Komal    |   NULL | NULL                |
|  7 | Muffy    |   NULL | NULL                |
+----+----------+--------+---------------------+
```

RIGHT JOIN

A **right outer join** will return all records in the join clauses' right table, regardless of matching records in the left table. The correct SQL outer join includes all of the rows from the right-hand table. The right SQL outer join is a special case, and many databases do not support right joins. A SQL right join can usually be rewritten as a SQL left join by simply changing the order of the tables in the query. Fields from the left table that do not match will display null values in this case.



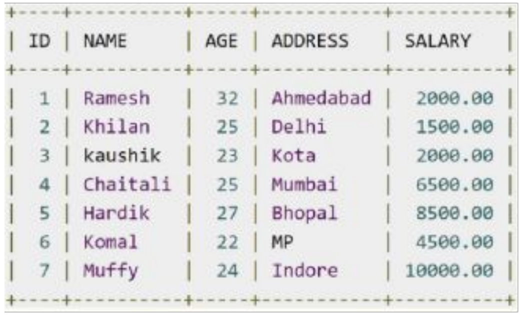
Basic Syntax of Right Join

```
SELECT
        Column_list
FROM
        TABLE1
RIGHT JOIN
        TABLE2
ON Table1.ColName = Table2.ColName
```

Example of Right Join

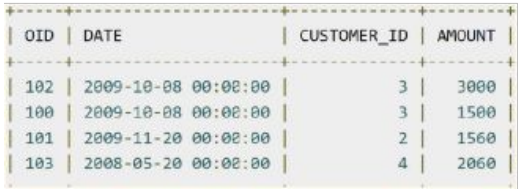
Question - Write a query to find total amount spent by each customer and dates on which they placed order.

CUSTOMERS(Left)



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

ORDERS(Right) Table

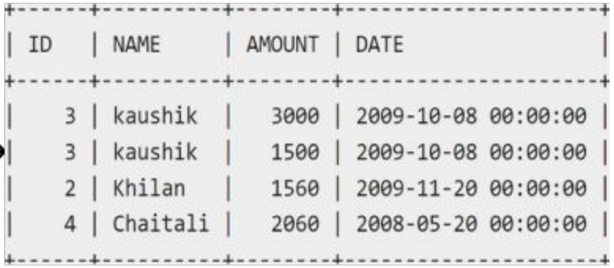


```
+-----+---------------------+-------------+--------+
| OID | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
```

Query

```
SELECT ID, NAME, AMOUNT, DATE  FROM CUSTOMERS
RIGHT JOIN ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

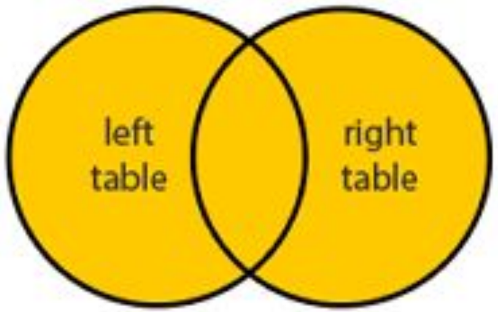
Output



```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

FULL JOIN

A **full join** will return all the rows in both tables. When rows don't match in one of the tables, the field will display a null value. A complete SQL outer join combines the effects of the SQL left joins and SQL right joins. Many databases do not support the implementation of full SQL outer joins.



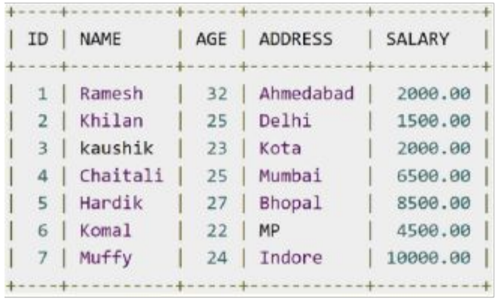
Basic Syntax of Full Join

```
SELECT
        Column_list
FROM
        TABLE1
FULL JOIN
        TABLE2
ON Table1.ColName = Table2.ColName
```

Example of Full Join

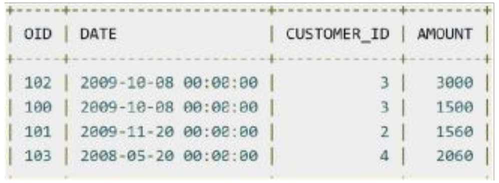
Question - Write a query to fetch entire database from customers and orders table

CUSTOMERS(Left)



```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

ORDERS(Right) Table



```
+-----+---------------------+-------------+--------+
| OID | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
```

Query

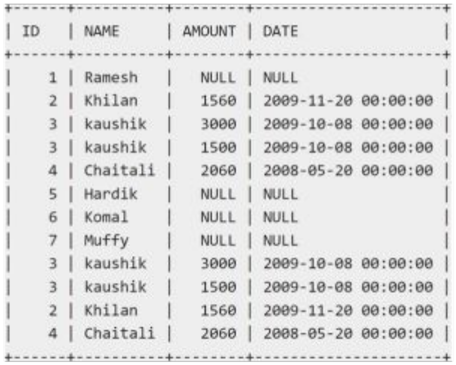
SELECT ID, NAME, AMOUNT, DATE FROM CUSTOMERS

FULL JOIN ORDERS

ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;

$\longrightarrow$

Output



```
+------+----------+--------+---------------------+
| ID   | NAME     | AMOUNT | DATE                |
+------+----------+--------+---------------------+
|    1 | Ramesh   |   NULL | NULL                |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|    5 | Hardik   |   NULL | NULL                |
|    6 | Komal    |   NULL | NULL                |
|    7 | Muffy    |   NULL | NULL                |
|    3 | kaushik  |   3000 | 2009-10-08 00:00:00 |
|    3 | kaushik  |   1500 | 2009-10-08 00:00:00 |
|    2 | Khilan   |   1560 | 2009-11-20 00:00:00 |
|    4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
+------+----------+--------+---------------------+
```

Join for more than two tables

- **How to prioritize tables** :

This can be done by determining which tables contain the data we need and include them

Thus start by writing the query for that table and also include all the tables that come along the way between this table that doesn’t contain data but serve as a relation between tables that do

Join for more than two tables

Question - Write a query to find course opted by each student

Student

id	first_name	last_name
1	Shreya	Bain
2	Rianna	Foster
3	Yosef	Naylor

Student_course

student_id	course_id
1	2
1	3
2	1
2	2
2	3
3	1

course

id	name	teacher_id
1	Database design	1
2	English literature	2
3	Python programming	1

Query

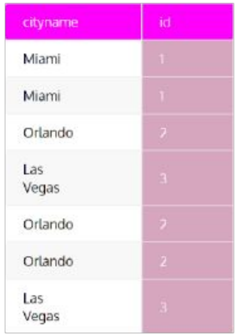
```
SELECT
  student.first_name,
  student.last_name,
  course.name
FROM student
JOIN student_course
  ON student.id = student_course.student_id
JOIN course
  ON course.id = student_course.course_id;
```

first_name	last_name	name
Shreya	Bain	English literature
Shreya	Bain	Python programming
Rianna	Foster	Database design
Rianna	Foster	English literature
Rianna	Foster	Python programming
Yosef	Naylor	Database design

Join + Aggregate + Group BY

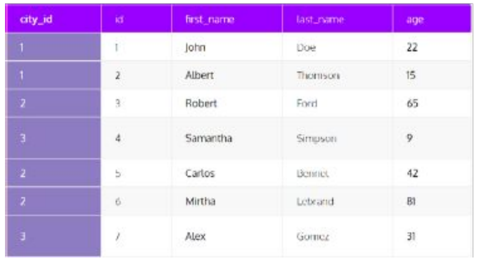
Question - Write a query to fetch city's name and minimum age of user from that city

cities



cityname	id
Miami	1
Miami	1
Orlando	2
Las Vegas	3
Orlando	2
Orlando	2
Las Vegas	3

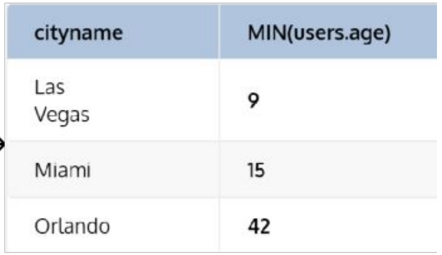
users



city_id	id	first_name	last_name	age
1	1	John	Doe	22
1	2	Albert	Thomson	15
2	3	Robert	Ford	65
3	4	Samantha	Simpson	9
2	5	Carlos	Bennet	42
2	6	Mirtha	Lebrand	81
3	7	Alex	Gomez	31

```
SELECT cities.cityname, MIN(users.age)
FROM cities
JOIN users
  ON cities.id = users.city_id
GROUP BY cities.cityname
```

Output

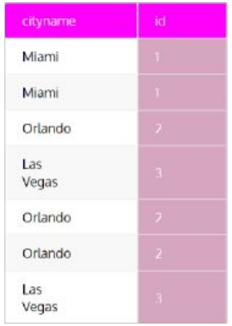


cityname	MIN(users.age)
Las Vegas	9
Miami	15
Orlando	42

Join + Aggregate + Group BY

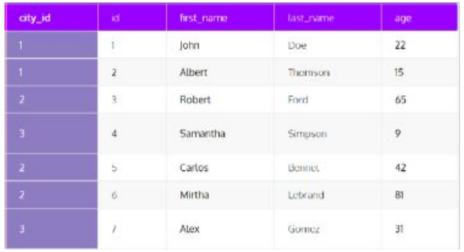
Question - Write a query to fetch city's name and maximum age of user from that city

cities



cityname	id
Miami	1
Miami	1
Orlando	2
Las Vegas	3
Orlando	2
Orlando	2
Las Vegas	3

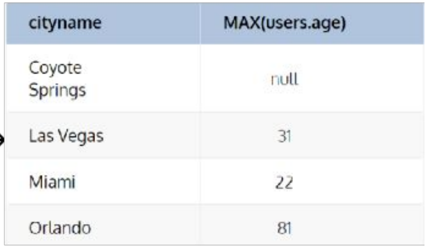
users



city_id	id	first_name	last_name	age
1	1	John	Doe	22
1	2	Albert	Thomson	15
2	3	Robert	Ford	65
3	4	Samantha	Simpson	9
2	5	Carlos	Bennet	42
2	6	Mirtha	Lebrand	81
3	7	Alex	Gomez	31

```
SELECT cities.cityname, MAX(users.age)
FROM cities
LEFT JOIN users
  ON cities.id = users.city_id
GROUP BY cities.cityname
```

Output

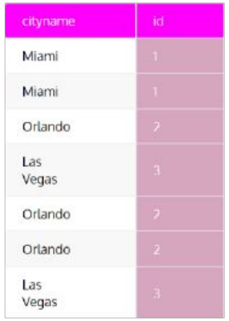


cityname	MAX(users.age)
Coyote Springs	null
Las Vegas	31
Miami	22
Orlando	81

Join + Aggregate + Group BY

Question - Write a query to fetch city's name and perform aggregate function on user's age and user's id

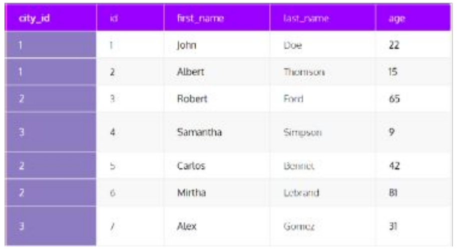
cities



cityname	id
Miami	1
Miami	1
Orlando	2
Las Vegas	3
Orlando	2
Orlando	2
Las Vegas	3

```
SELECT
   cities.cityname,
   SUM(users.age) AS sum,
   COUNT(users.id) AS count,
   SUM(users.age) / COUNT(users.id) AS
average
FROM cities
LEFT JOIN users
 ON cities.id = users.city_id
GROUP BY cities.cityname
```

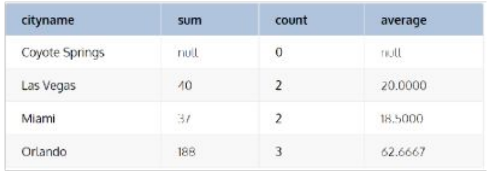
users



city_id	id	first_name	last_name	age
1	1	John	Doe	22
1	2	Albert	Thomson	15
2	3	Robert	Ford	65
3	4	Samantha	Simpson	9
2	5	Carlos	Bennet	42
2	6	Mirtha	Lebrand	81
3	7	Alex	Gomez	31

$\longrightarrow$

Output



cityname	sum	count	average
Coyote Springs	null	0	null
Las Vegas	40	2	20.0000
Miami	37	2	18.5000
Orlando	188	3	62.6667

Filtering the data in queries with Join

For filtering data in the queries containing joins, we have two options:

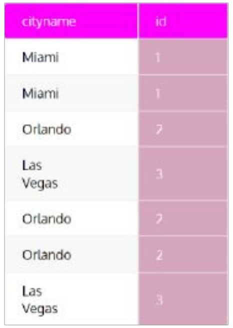
1. Where Clause
2. On condition in join

Depending on the situation, each of these options can have a different outcome. It's important to understand which to use when we want a specific result.

Filtering using ‘ON’ condition

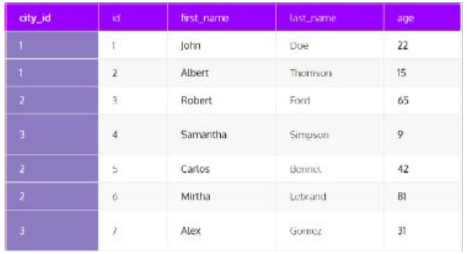
Question - Write a query to find number of users with ages lower than 30

cities



cityname	id
Miami	1
Miami	1
Orlando	2
Las Vegas	3
Orlando	2
Orlando	2
Las Vegas	3

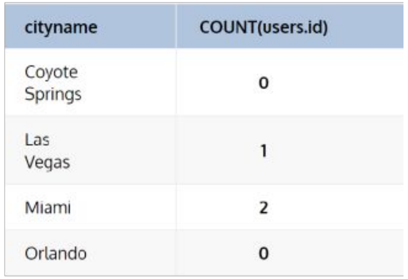
users



city_id	id	first_name	last_name	age
1	1	John	Doe	22
1	2	Albert	Thomson	15
2	3	Robert	Ford	65
3	4	Samantha	Simpson	9
2	5	Carlos	Bennet	42
2	6	Mirtha	Lebrand	81
3	7	Alex	Gomez	31

```
SELECT
 cityname,
 COUNT(users.id)
FROM cities LEFT JOIN users
ON cities.id = users.city_id
AND users.age < 30
GROUP BY cities.cityname
ORDER BY cities.cityname;
```

Output



cityname	COUNT(users.id)
Coyote Springs	0
Las Vegas	1
Miami	2
Orlando	0

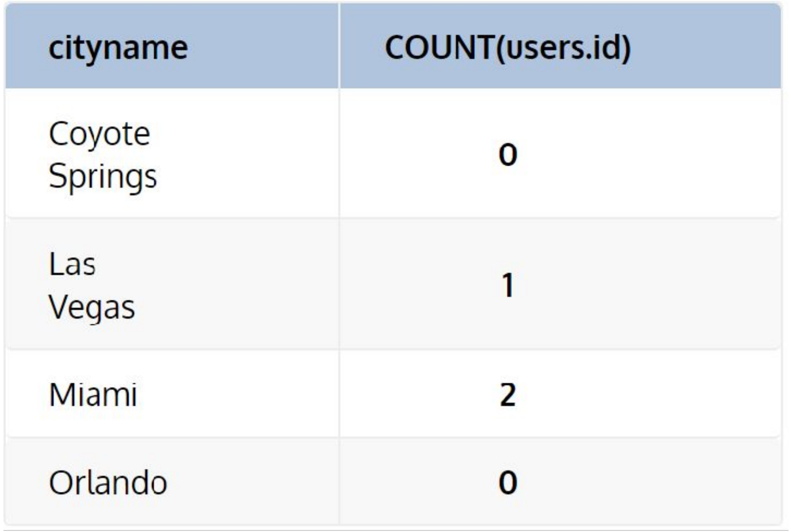
#270DaysofPurpose

Relevel
by Unacademy

Filtering using ‘ON’ condition - Understanding

The condition to include only users with ages lower than 30 is set in the JOIN predicate.

All cities are listed in the output, and only those users with ages within range return a non-zero number. Cities without any users matching our criteria return a zero.



cityname	COUNT(users.id)
Coyote Springs	0
Las Vegas	1
Miami	2
Orlando	0

Filtering using ‘WHERE’ condition

Question - Write a query to find number of users with ages lower than 30 using where condition

cities

cityname	id
Miami	1
Miami	1
Orlando	2
Las Vegas	3
Orlando	2
Orlando	2
Las Vegas	3

users

city_id	id	first_name	last_name	age
1	1	John	Doe	22
1	2	Albert	Thomson	15
2	3	Robert	Ford	65
3	4	Samantha	Simpson	9
2	5	Carlos	Bennet	42
2	6	Mirtha	Lebrand	81
3	7	Alex	Gomez	31

```
SELECT cityname, COUNT(users.id)
FROM cities
LEFT JOIN users
  ON cities.id = users.city_id
WHERE users.age < 30
GROUP BY cities.cityname
ORDER BY cities.cityname;
```

Output

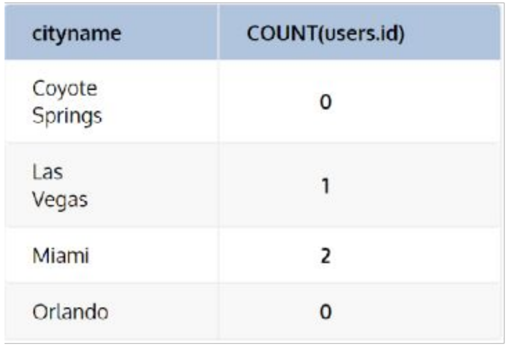
cityname	COUNT(users.id)
Las Vegas	1
Miami	2

Filtering using ‘Where’ condition - Understanding

The expected output is different from the actual output. We wanted to get ALL cities and count their respective users aged less than 30. Even if a city had no users, it should have been listed zero, as returned by the JOIN predicate example.

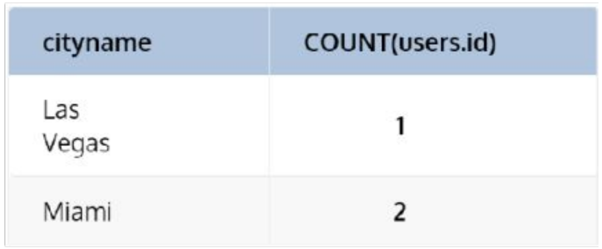
This didn't return those records because WHERE conditions are applied after the JOIN. Since the condition users.age < 30 removes all "Coyote Springs" and "Orlando" records; the summarized calculation can't include these values. Only "Las Vegas" and "Miami" meet the WHERE conditions, so only "Las Vegas" and "Miami" are returned.

Expected Output



cityname	COUNT(users.id)
Coyote Springs	0
Las Vegas	1
Miami	2
Orlando	0

Actual Output



cityname	COUNT(users.id)
Las Vegas	1
Miami	2

Practice Question

Instructions for practice questions

- Create account on
    - https://www.hackerrank.com/
    - https://leetcode.com/
    - https://www.stratascratch.com/
- Refer to the url provided in the practice questions

Practice Question

1. https://www.hackerrank.com/challenges/african-cities/problem?isFullScreen=true

Practice Question

Solution:

```
SELECT

city.name  FROM

city  INNER JOIN

country

ON city.countrycode = country.code  WHERE

country.continent = 'Africa'
```

Practice Question

2. https://www.hackerrank.com/challenges/average-population-of-each-continent/problem?isFullScreen=true

Practice Question

Solution:

```
SELECT

COUNTRY.Continent,

FLOOR(AVG(CITY.Population)) AS avg_population

FROM

city INNER JOIN

country

ON city.countrycode = country.code  GROUP BY

COUNTRY.Continent
```

Practice Question

3. https://leetcode.com/problems/combine-two-tables/submissions/

Practice Question

Solution:

```
SELECT

FirstName,  LastName,  City,

State  FROM

Person  LEFT JOIN

address

ON Person.PersonId =Address.PersonId
```

Practice Question

4. https://platform.stratascratch.com/coding/9891-customer-details?python=https://platform.stratascratch.com/coding/10353-workers-with-the-highest-salaries?python=

Practice Question

Solution:

```
SELECT
first_name,
last_name,
city,
order_details
FROM
customers
LEFT JOIN
orders
ON customers.id = orders.cust_id
ORDER BY
first_name,  order_details
```

Practice Question

5. https://platform.stratascratch.com/coding/10061-popularity-of-hack?python=

- Instruction:

If table_name seems long to include you can rename the table using
From table_name **x**
And use this **x** for referring the table in rest of query.

Practice Question

Solution:

```
SELECT
location,
AVG(popularity)
FROM
facebook_employees a
JOIN
facebook_hack_survey b
ON a.id = b.employee_id
GROUP BY
location
```

Practice Question

6. https://platform.stratascratch.com/coding/9913-order-details?code_type=1

Practice Question

Solution:

```
SELECT
          first_name,
          order_date,
          order_details,
          total_order_cost
FROM
          customers
JOIN
          orders
ON customers.id = orders.cust_id
WHERE    first_name IN ('Jill', 'Eva’)
ORDER BYcust_id
```

Practice Question

7. https://platform.stratascratch.com/coding/9915-highest-cost-orders?python=

Practice Question

Solution:

```
SELECT
first_name,
SUM(total_order_cost) AS total_cost,
order_date
FROM
            customers
JOIN
            orders
ON customers.id = orders.cust_id
WHERE order_date BETWEEN '2019-02-01' AND '2019-05-01’
GROUP BY
            first_name,
 order_date
ORDER BY    total_cost DESC
LIMIT 1
```

THANK YOU

In the next class we will study:

- Joins - II and Union