SQL Database Manipulations: SELECT statements

Thomas Schwarz, SJ

SELECT

- SELECT is the most frequent command
  - Basic use:
    - SELECT attribute1, attribute2, … FROM databasetable
    - SELECT * FROM databasetable

SELECT

- SELECT — WHERE clause:
    - Imposes a condition on the results

SELECT

- =   equals (comparison operator)
- AND, OR
- IN, NOT IN
- LIKE, NOT LIKE
- BETWEEN … AND
- EXISTS, NOT EXISTS
- IS NULL, IS NOT NULL
- comparison operators

SELECT

- AND operator
  - Combines two statements (concerning one or more tables)

```
SELECT
   *
FROM
   employees
WHERE
   first_name = 'Denis' and gender = 'M';
```

SELECT

- OR is the Boolean or
- Trick Question: How many records will this query return?

```
SELECT
   *
FROM
   employees
WHERE
   last_name = 'Denis' AND gender = 'M' OR gender = 'F'
```

SELECT

- Operator precedence:
    - AND < OR

```
SELECT
   *
FROM
   employees
WHERE
   last_name = 'Denis' AND (gender = 'M' OR gender = 'F')
```

SELECT

- Quiz:
    - Retrieve all female employees with first name 'Aruna' or 'Kelly'

```
SELECT
   *
FROM
   employees
WHERE
   gender = 'F' AND
   (first_name = 'Aruna' OR first_name = 'Kelly');
```

SELECT

- IN, NOT IN
  - Checks for membership in lists
  - MySQL: faster than equivalent OR formulation

```
SELECT
   *
FROM
   employees
WHERE
   first_name NOT IN ('Elvis','Kevin','Thomas');
```

SELECT

- LIKE
  - Pattern matching
    - Wild cards
      - % means zero or more characters
      - _ means a single letter
      - [ ] means any single character within the bracket
      - ^ means any character not in the bracket
      - - means a range of characters

Like Examples

- WHERE name LIKE 't%'
    - any values that start with 't'
- WHERE name LIKE '%t'
    - any values that end with 't'
- WHERE name LIKE '%t%'
    - any value with a 't' in it
- WHERE name LIKE '_t%'
    - any value with a 't' in second position

Like Examples

- WHERE name LIKE '[ts]%'
  - any values that start with 't' or 's'
- WHERE name LIKE '[t-z]'
  - any values that start with 't', 'u', 'v', 'w', 'x', 'y', 'z'
- WHERE name LIKE '[!ts]%'
  - any value that does not start with a 't' or a 's'
- WHERE name LIKE '_t%'
  - any value with a 't' in second position

SELECT

- BETWEEN … AND …
  - Selects records with a value in the range
    - endpoints included

```
SELECT
   *
FROM
   employees
WHERE
   hire_data between 1990-01-01 and 1999-12-31;
```

SELECT

- SELECT DISTINCT

```
SELECT DISTINCT
     gender
FROM
     employees
```

SELECT

- Aggregate Functions
  - Applied to a row of a result table
    - COUNT
    - SUM
    - MIN
    - MAX
    - AVG

SELECT

- SELECT COUNT

  -
    ```
    SELECT
        COUNT(emp_no)
    FROM
        employees
    ```

SELECT

- SELECT COUNT

```
SELECT COUNT(employees.emp_no)
FROM employees
WHERE
     first_name LIKE ('Tom%') or first_name
LIKE('Tho%');
```

SELECT

- Combine COUNT with DISTINCT

```
SELECT
   COUNT(DISTINCT first_name, last_name)
FROM
    employees
```

SELECT

- Combine COUNT with DISTINCT

```
SELECT
   COUNT(DISTINCT emp_no)
FROM
   salaries
WHERE
   salary >=100000;
```

SELECT

- ORDER BY
  - Orders result by default in ascending order
    - ASC ascending
    - DSC descending

```
SELECT
   *
FROM
   employees
WHERE
    hire_date > '2000-01-01'
ORDER BY first_name;
```

SELECT

- GROUP BY
  - Just before ORDER BY in a query
    - Needed with aggregate functions
      - Example: Getting all first names in order

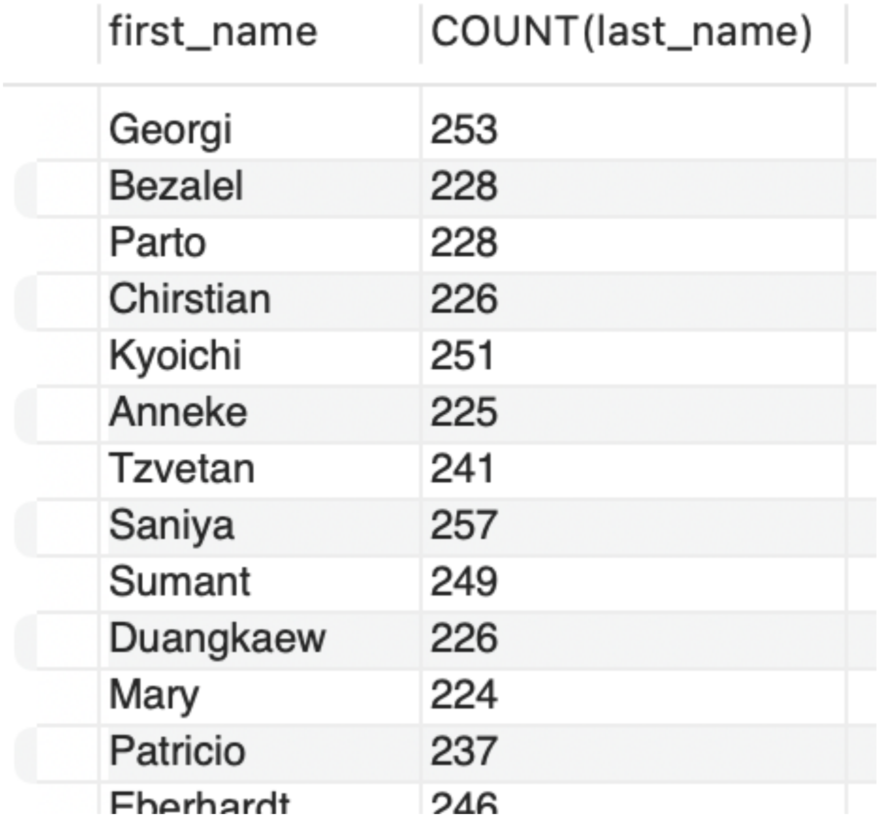
```
SELECT
   first_name
FROM
   employees
GROUP BY first_name;
```

SELECT

- GROUP BY is used with aggregate functions

-

```
SELECT
    first_name, COUNT(last_name)
FROM
    employees
GROUP BY first_name;
```

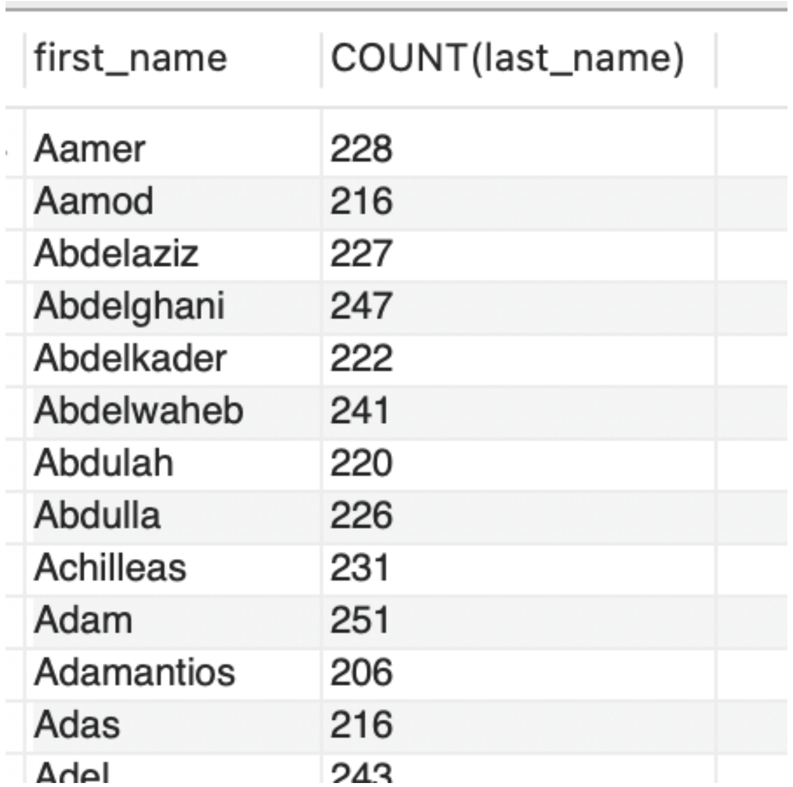


first_name	COUNT(last_name)
Georgi	253
Bezalel	228
Parto	228
Chirstian	226
Kyoichi	251
Anneke	225
Tzvetan	241
Saniya	257
Sumant	249
Duangkaew	226
Mary	224
Patricio	237
Eberhardt	246

SELECT

- GROUP BY is often combined with ORDER BY

```
SELECT
   first_name, COUNT(first_name)
FROM
   employees
GROUP BY first_name
ORDER BY first_name;
```

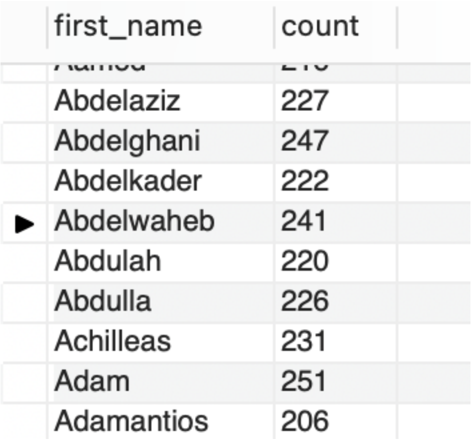


first_name	COUNT(last_name)
Aamer	228
Aamod	216
Abdelaziz	227
Abdelghani	247
Abdelkader	222
Abdelwaheb	241
Abdulah	220
Abdulla	226
Achilleas	231
Adam	251
Adamantios	206
Adas	216
Adel	243

SELECT

- GROUP BY
  - Example: Counting first names in the employee data base
    - To make it look better, add an AS clause

```
SELECT
    first_name, COUNT(last_name) AS count
FROM
    employees
GROUP BY first_name
ORDER BY first_name;
```



	first_name	count
	Abdelaziz	227
	Abdelghani	247
	Abdelkader	222
▶	Abdelwaheb	241
	Abdulah	220
	Abdulla	226
	Achilleas	231
	Adam	251
	Adamantios	206

In Class Exercises

- Using MySQL Workbench
  - Create a new database called TEST
  - Create a table R with attributes A and B of type INT
  - Insert these values into R using insert statements such as INSERT INTO R(A,B) VALUES(3,9);
  - Use a SELECT statement to insure that the table is correct (including the double values)

A	B
1	2
1	3
1	4
2	1
2	3
3	1
3	2
3	9
4	2
4	2

```
INSERT INTO R (A, B)
VALUES
(1,2),
(1,3),
(1,4),
(2,1),
(2,3),
(3,1),
(3,2),
(3,9),
(4,2),
(4,2);
```

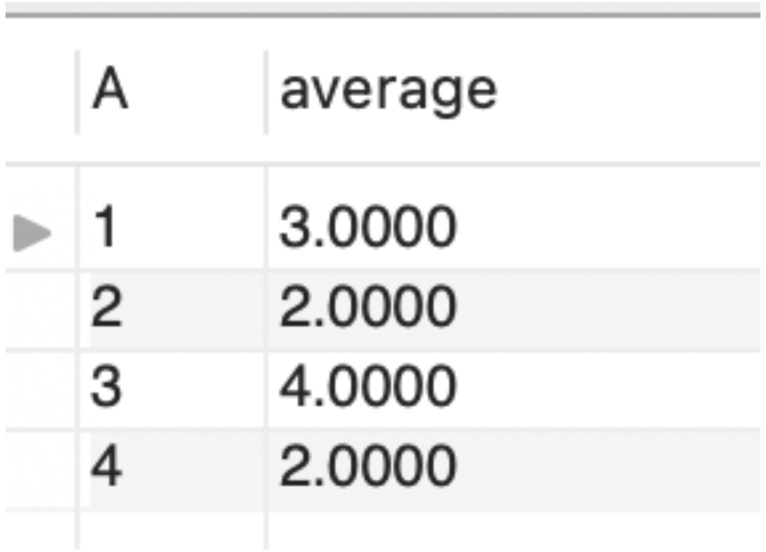
In Class Exercises

- Obtain a table that lists the average value of B (AVG) for all values of A

A	B
1	2
1	3
1	4
2	1
2	3
3	1
3	2
3	9
4	2
4	2

A	BAve
1	3.0
2	2.0
3	4.0
4	2.0

```
SELECT
    A, AVG(B) AS average
FROM
    R
GROUP BY A
ORDER BY A;
```

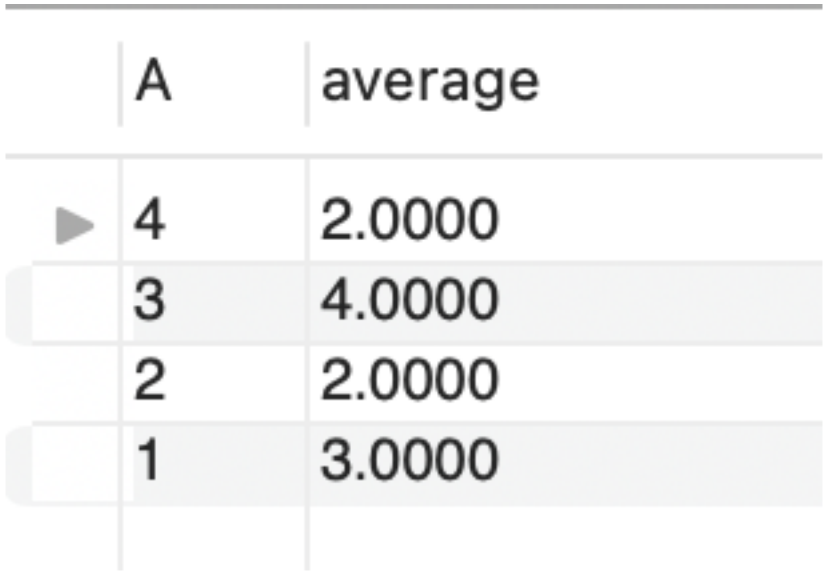


A	average
1	3.0000
2	2.0000
3	4.0000
4	2.0000

In Class Exercises

- Obtain the same table, but in descending order of A

```
SELECT
    A, AVG(B) AS average
FROM
    R
GROUP BY A
ORDER BY A DESC;
```



A	average
4	2.0000
3	4.0000
2	2.0000
1	3.0000

In Class Exercises

- Create a table that contains only the unique value pairs for A and B

In Class Exercises

```
SELECT DISTINCT
    *
FROM
    R;
```

In Class Exercises

- How many entries does the table have with and without uniqueness constraints?

In Class Exercises

```
SELECT
    COUNT(A,B) AS numberOfRecords
FROM
    R;
```

```
SELECT
    COUNT(DISTINCT A,B) AS numberOfRecords
FROM
    R;
```

In Class Exercises

- Find the average and the number of counts for all B-values depending on the A-value

A	countb	aveB
1	3	3.0000
2	2	2.0000
3	3	4.0000
4	2	2.0000

In Class Exercises

```
SELECT
    A, COUNT(B) AS countb, AVG(B) AS aveB
FROM
    R
```

A	countb	aveB
1	3	3.0000
2	2	2.0000
3	3	4.0000
4	2	2.0000

In Class Exercises

- Do the same, but make sure that we do not count double rows twice

In Class Exercises

```
SELECT
   A, COUNT(B) AS countb, AVG(B) AS aveB
FROM (
   SELECT DISTINCT
     A,B
   FROM
     R
        ) AS AUnique
GROUP BY A;
```

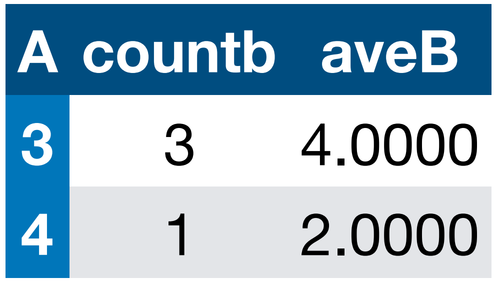
A	countb	aveB
1	3	3.0000
2	2	2.0000
3	3	4.0000
4	1	2.0000

In Class Exercises

- Select the count of B-values and average of B-values where the A value is at least 3
  - We modify this with a WHERE clause
  - The WHERE is applied to all tuples first, then the grouping and the calculation of the aggregate function happens

In Class Exercises

```
SELECT
    A, COUNT(B) AS countb, AVG(B) AS aveB
FROM
    (SELECT DISTINCT
        A, B
    FROM
        R) AS AUnique
WHERE
    A > 2
GROUP BY A;
```



A	countb	aveB
3	3	4.0000
4	1	2.0000

Having

- A WHERE clause applies to all the rows, but it cannot apply to the groups created by the GROUP BY
    - For this, SQL introduces the HAVING clause
    - Just like a WHERE clause, but refers to aggregated data

Having

- Syntax of Having

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Having

- Difference between WHERE and HAVING
  - WHERE is only for selecting tuples
  - HAVING can only refer to the group-by-ed attribute

In Class Exercises

- Insert another double tuple 1, 1
- Get count and average of the B-values in dependence on A where the count is 2 or less

Table 1

A	B
1	2
1	3
1	4
2	1
2	3
3	1
3	2
3	9
4	2
4	2
1	1
1	1

In Class Exercises

```
SELECT
    A, COUNT(B), AVG(B)
FROM
    R
GROUP BY A
HAVING COUNT(B) <= 2;
```

Table

A	B
1	2
1	3
1	4
2	1
2	3
3	1
3	2
3	9
4	2
4	2
1	1
1	1

Table 1

A	COUNT(B)	AVG(B)
2	2	2.0000
4	2	2.0000

In Class Exercises

- Get count and average of the B-values in dependence on A where A is less than or equal to 2

In Class Exercises

```
SELECT
    A, COUNT(B), AVG(B)
FROM
    R
WHERE
    A <= 2
GROUP BY A;
```

Table 1

A	B
1	2
1	3
1	4
2	1
2	3
3	1
3	2
3	9
4	2
4	2
1	1
1	1

Table 1-1

A	COUNT(B)	AVG(B)
1	5	2.2000
2	2	2.0000

SELECT

- LIMIT gives the maximum number of rows returned
  - Can be used for a sample
  - Can be used with ORDER BY ASC

SELECT

- Use the employees database
  - Find the five employees that have made the most money
    - Hint: The Salary table has the information but employees have different salaries over time

SELECT

```
SELECT
    first_name, last_name, MAX(salary)
FROM
    salaries,
    employees
WHERE
    employees.emp_no = salaries.emp_no
GROUP BY salaries.emp_no
ORDER BY MAX(salary) DESC
LIMIT 5;
```

Table 1

first_name	last_name	MAX(salary)
Tokuyasu	Pesch	158220
Xiahua	Whitcomb	155709
Tsutomu	Alameldin	155377
Willard	Baca	154459
Ibibia	Junet	150345

JOINS

- Create and populate another table

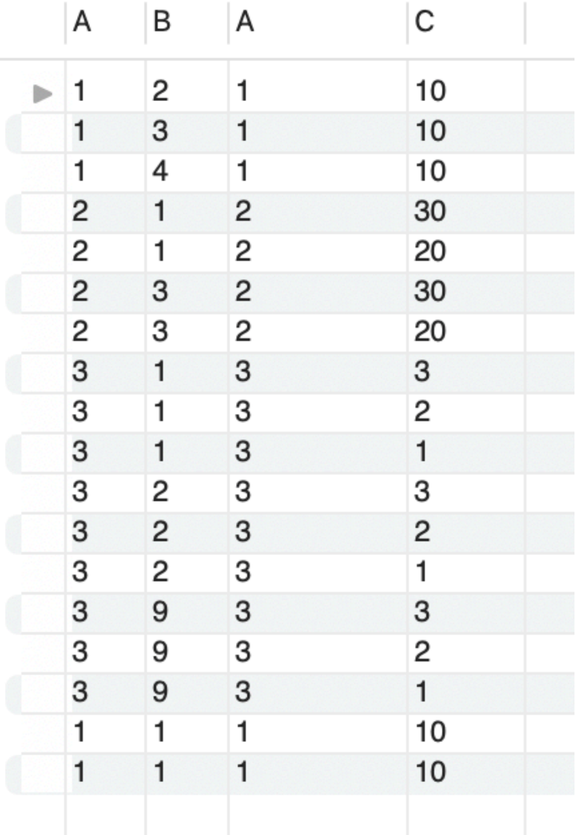
```
CREATE TABLE S (
    A INT,
    C INT
);
```

```
INSERT INTO S
   (A, C)
    VALUES
   (1,10),
    (2,20),
    (2,30),
    (3,1),
    (3,2),
    (3,3);
```

Joins

- Inner Join

```
SELECT
    *
FROM
    R
        INNER JOIN
    S ON R.A = S.A;
```

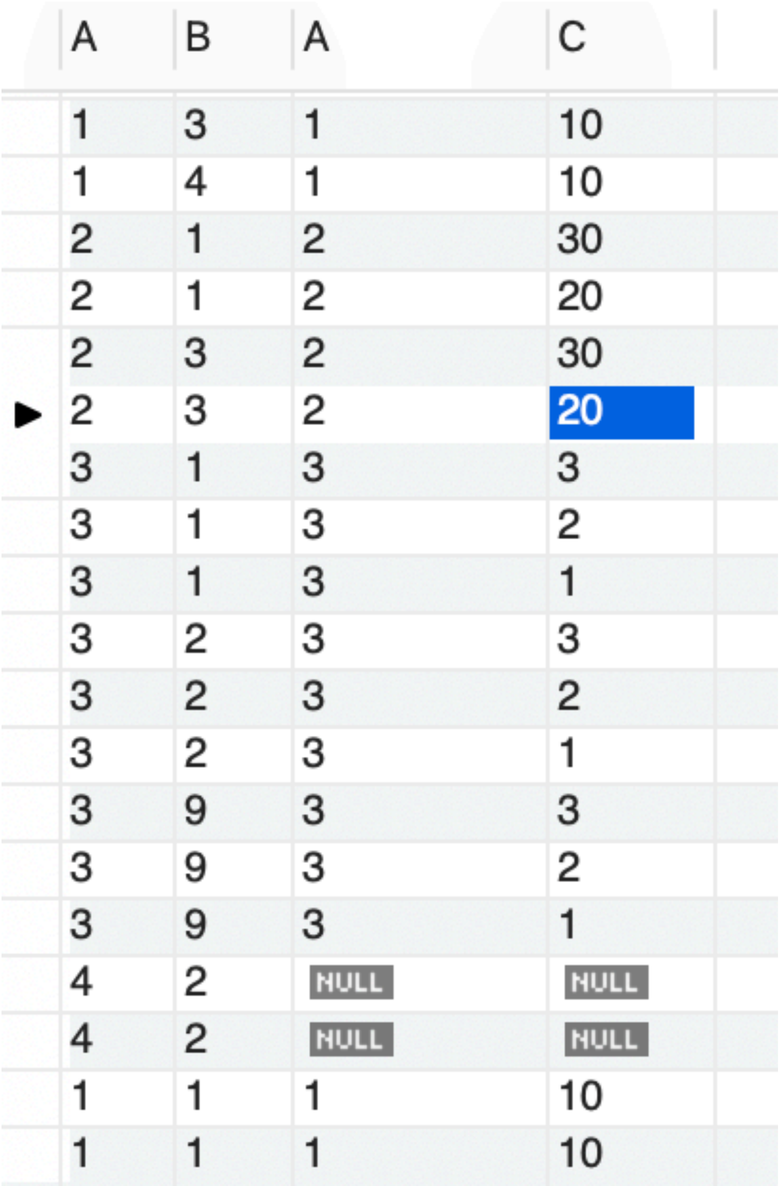


A	B	A	C
1	2	1	10
1	3	1	10
1	4	1	10
2	1	2	30
2	1	2	20
2	3	2	30
2	3	2	20
3	1	3	3
3	1	3	2
3	1	3	1
3	2	3	3
3	2	3	2
3	2	3	1
3	9	3	3
3	9	3	2
3	9	3	1
1	1	1	10
1	1	1	10

Joins

- Outer Join: MySQL only knows LEFT OUTER JOIN and RIGHT OUTER JOIN

```
SELECT
    *
FROM
    R LEFT OUTER JOIN S
    ON
     R.A = S.A;
```

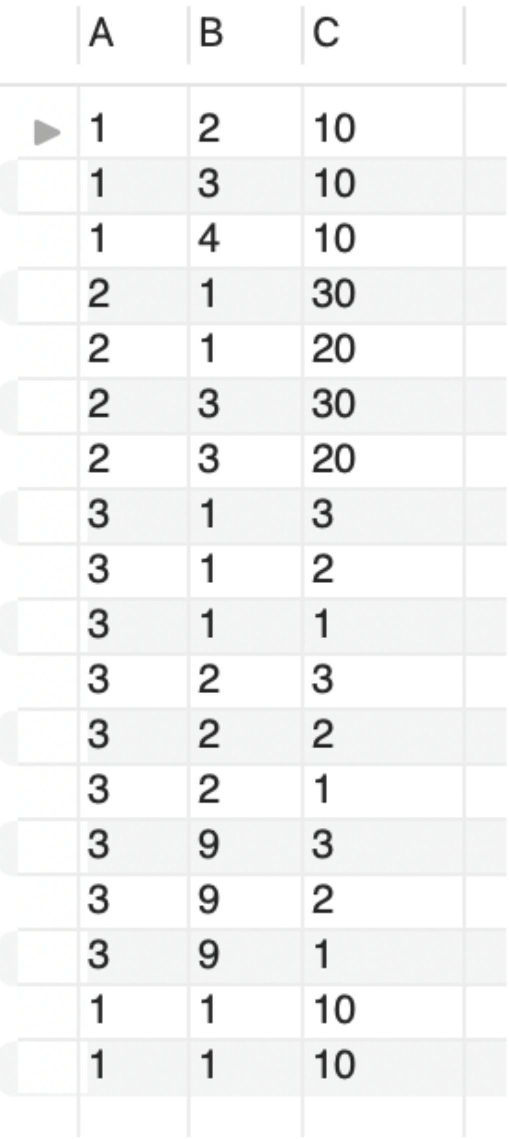


	A	B	A	C
	1	3	1	10
	1	4	1	10
	2	1	2	30
	2	1	2	20
	2	3	2	30
▶	2	3	2	20
	3	1	3	3
	3	1	3	2
	3	1	3	1
	3	2	3	3
	3	2	3	2
	3	2	3	1
	3	9	3	3
	3	9	3	2
	3	9	3	1
	4	2	NULL	NULL
	4	2	NULL	NULL
	1	1	1	10
	1	1	1	10

Joins

- The *“old”* SQL syntax uses the description of the join

```
SELECT
    R.A, R.B, S.C
FROM
    R,
    S
WHERE
    R.A = S.A;
```



A	B	C
1	2	10
1	3	10
1	4	10
2	1	30
2	1	20
2	3	30
2	3	20
3	1	3
3	1	2
3	1	1
3	2	3
3	2	2
3	2	1
3	9	3
3	9	2
3	9	1
1	1	10
1	1	10

Joins

- Download MySQL sample data base from
  - https://www.mysqltutorial.org/mysql-sample-database.aspx

Joins

- Contains order information for a fictitious model seller
  - Customers: stores customer’s data.
  - Products: stores a list of scale model cars.
  - ProductLines: stores a list of product line categories.
  - Orders: stores sales orders placed by customers.
  - OrderDetails: stores sales order line items for each sales order.
  - Payments: stores payments made by customers based on their accounts.
  - Employees: stores all employee information as well as the organization structure such as who reports to whom.
  - Offices: stores sales office data.

Joins

- Find the countries to which Ship models are being sent:

```
SELECT DISTINCT
    customers.country
FROM
    customers,
    orders,
    orderdetails,
    products,
    productlines
WHERE
    customers.customerNumber = orders.customerNumber
        AND orders.orderNumber = orderdetails.orderNumber
        AND orderdetails.productCode = products.productCode
        AND products.productLine = 'Ships'
ORDER BY customers.country;
```

SELECT

```
SELECT
    first_name, last_name, salary, employees.citizenship
FROM
    employees,
    salaries,
    (SELECT DISTINCT citizenship
     FROM
        employees
    ) chp
WHERE
    employees.emp_no = salaries.emp_no
        AND salaries.salary = (SELECT
            MAX(salaries.salary)
        FROM
            employees,
            salaries
        WHERE
            employees.emp_no = salaries.emp_no
                AND employees.citizenship = chp.citizenship)
    ORDER BY chp.citizenship;
```