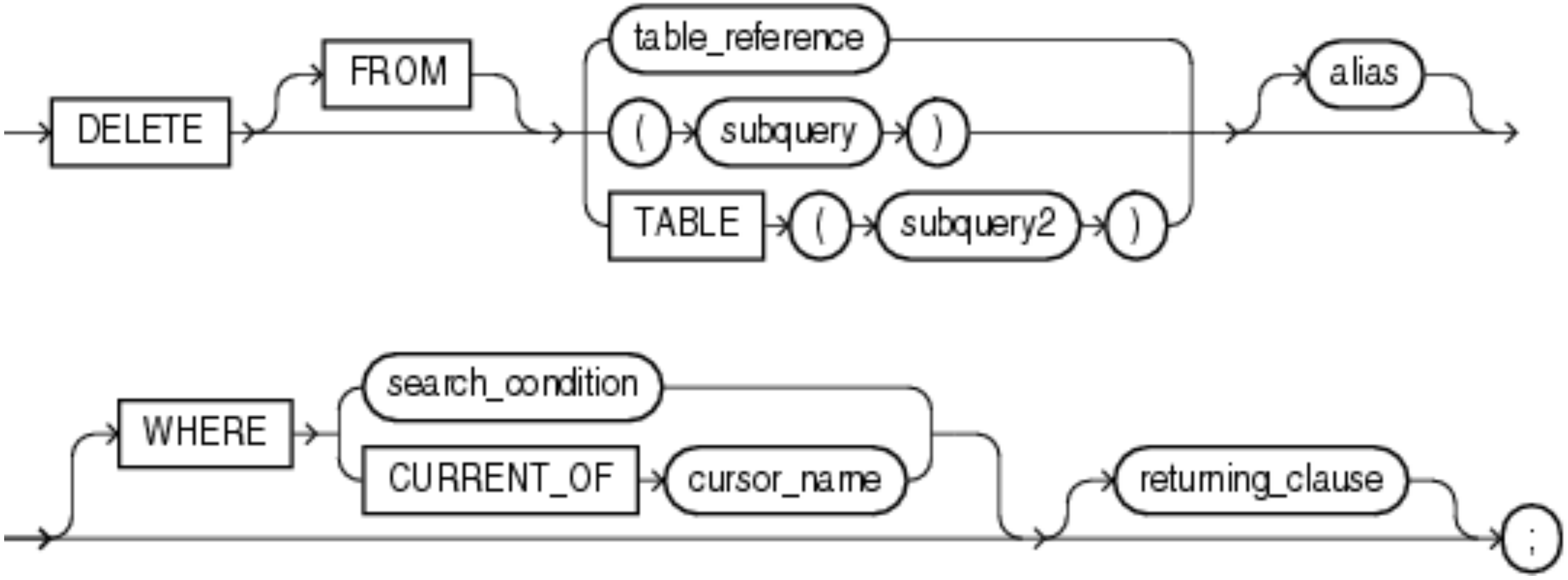
Agenda

- Keys & Constraints
- Aggregate functions – SUM, COUNT..
- Group By + Having
- Joins



Constraints

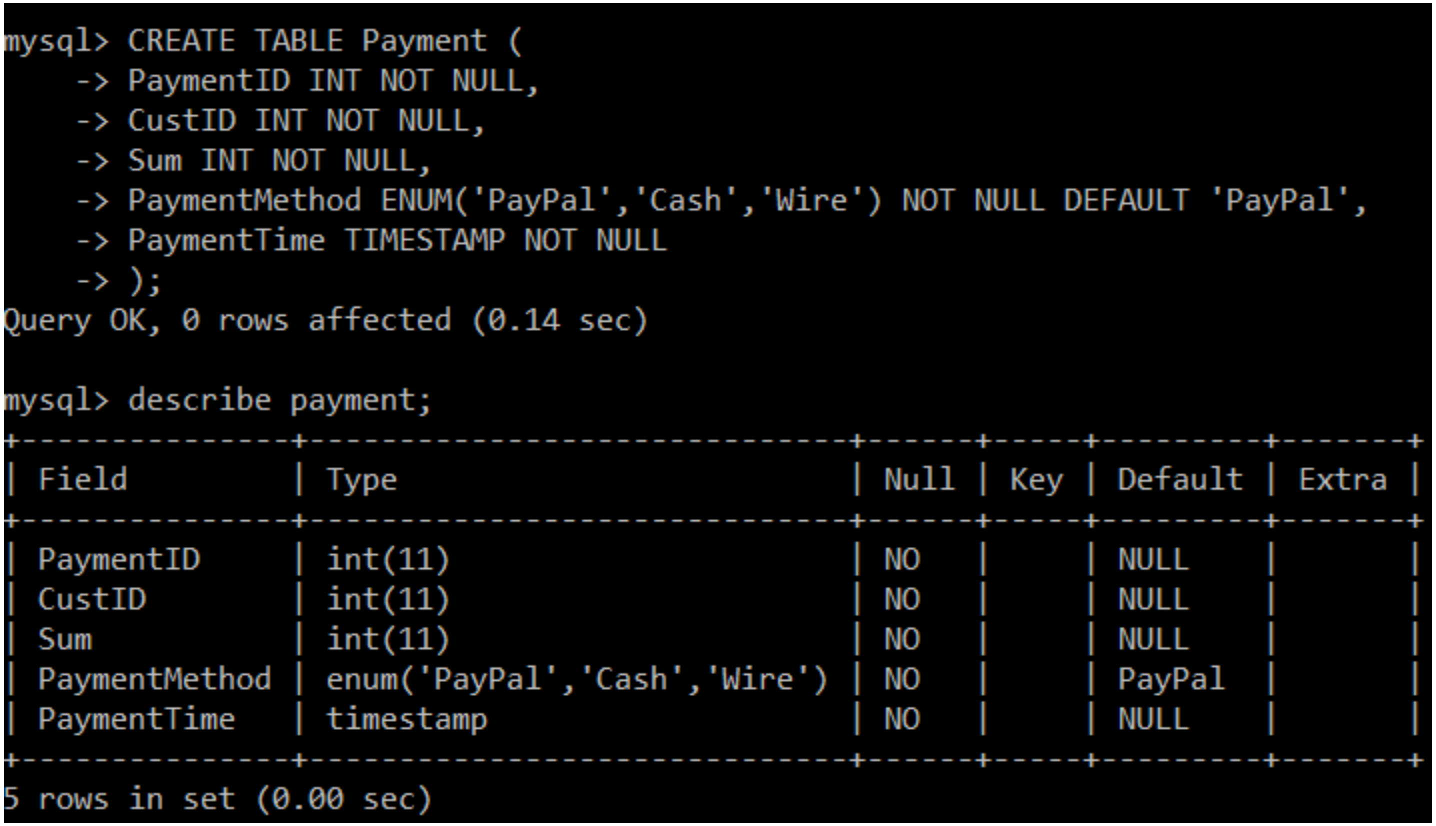
- ➢ In some cases, we’d like to restrict the values of a specific field to meet certain criteria. For example:
  - ○ Unique values: emails, usernames, IDs, etc.
  - ○ Cannot be NULL (i.e., has to have a value)
  - ○ Can only be a specific value from a list of options

Constraints

- Constraints are usually set when creating a table
- We can use `NOT NULL` and `ENUM` to restrict the valid values a column can take
- Let’s create a new payments per customer table:

```
CREATE TABLE Payment (
PaymentID INT NOT NULL,
CustID INT NOT NULL,
Sum INT NOT NULL,
PaymentMethod ENUM('PayPal','Cash','Wire') NOT
NULL DEFAULT 'PayPal’,
PaymentTime TIMESTAMP NOT NULL)
```

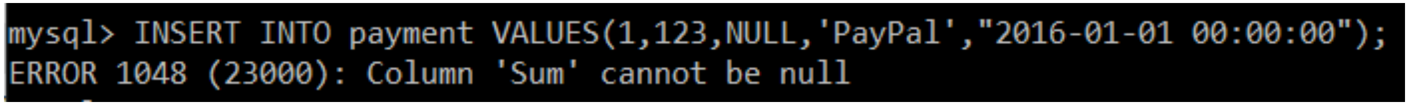
Constraints



```
mysql> CREATE TABLE Payment (
    -> PaymentID INT NOT NULL,
    -> CustID INT NOT NULL,
    -> Sum INT NOT NULL,
    -> PaymentMethod ENUM('PayPal','Cash','Wire') NOT NULL DEFAULT 'PayPal',
    -> PaymentTime TIMESTAMP NOT NULL
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql> describe payment;
+---------------+------------------------------+------+-----+---------+-------+
| Field         | Type                         | Null | Key | Default | Extra |
+---------------+------------------------------+------+-----+---------+-------+
| PaymentID     | int(11)                      | NO   |     | NULL    |       |
| CustID        | int(11)                      | NO   |     | NULL    |       |
| Sum           | int(11)                      | NO   |     | NULL    |       |
| PaymentMethod | enum('PayPal','Cash','Wire') | NO   |     | PayPal  |       |
| PaymentTime   | timestamp                    | NO   |     | NULL    |       |
+---------------+------------------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

- If we try inserting a NULL, or omit the value of any of the fields, we’ll get an error message:



```
mysql> INSERT INTO payment VALUES(1,123,NULL,'PayPal',"2016-01-01 00:00:00");
ERROR 1048 (23000): Column 'Sum' cannot be null
```

Enums

- Enums are commonly specified when creating a table
  - They are non-standard type, which is not supported by some RDBMS implementations

```
PaymentMethod ENUM('PayPal','Cash','Wire') NOT NULL DEFAULT 'PayPal',
```

Defining the ENUM and its values

Defining default value in case no value is given

- Enum values can be only strings
- When setting the value of an Enum field, we can use the index number of the desired value (0, 1, …)

```
INSERT INTO PAYMENT VALUES(2,124, 30, 2, '2016-02-01 00:00:00');
```

Inserting the ENUM value by index (2 = “Wire”)

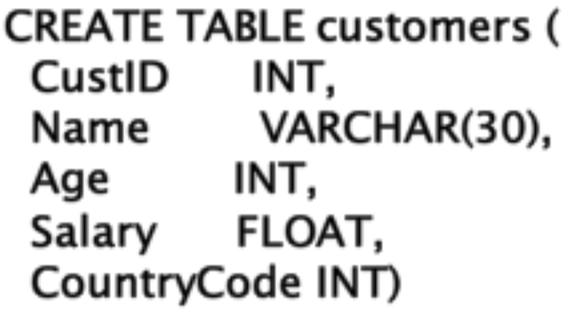
Primary Keys

- Every table should have a column or combination of columns whose values uniquely identify a row
- That column (or columns) is referred to as the **Primary Key**
- Defining a column as a primary key implicitly sets two constraints on it:
  1. It cannot be Null
  2. Its' value must be unique
- Using a primary key allows the RDBMS to perform queries faster and more efficiently.

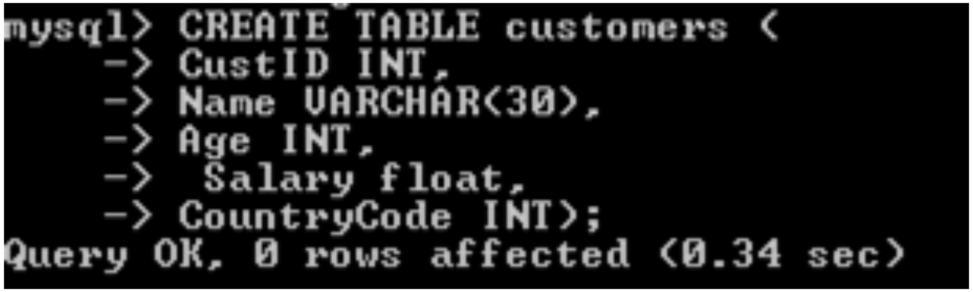
Read more in this blog post

Primary Key - Example

- Let’s revisit the table we created in the previous lecture:



```
CREATE TABLE customers (
  CustID      INT,
  Name         VARCHAR(30),
  Age          INT,
  Salary      FLOAT,
  CountryCode INT)
```

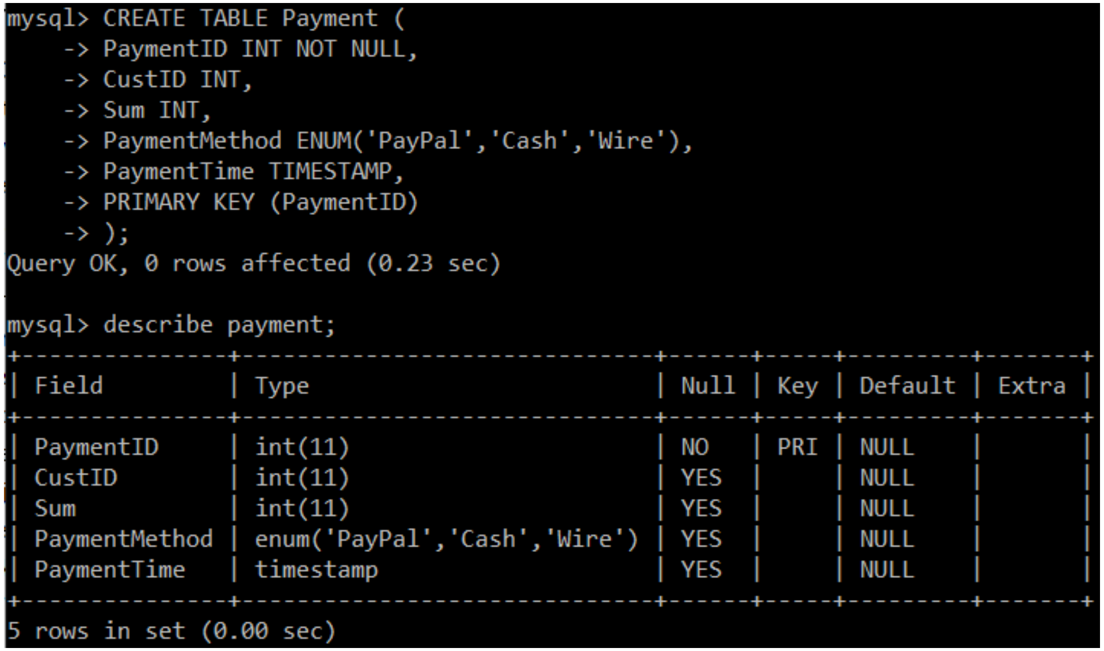


```
mysql> CREATE TABLE customers (
    -> CustID INT,
    -> Name VARCHAR(30),
    -> Age INT,
    ->  Salary float,
    -> CountryCode INT);
Query OK, 0 rows affected (0.34 sec)
```

If we wanted to define a Primary Key on the table, the right candidate would be the CustID column.

Setting a Primary Key When Creating a Table

```
CREATE TABLE Payment (
  PaymentID INT NOT NULL,
  CustID INT NOT NULL,
  Sum INT NOT NULL,
  PaymentMethod ENUM('PayPal','Cash','Wire') NOT NULL DEFAULT 'PayPal’,
  PaymentTime TIMESTAMP NOT NULL
  PRIMARY KEY (PaymentID)
);
```



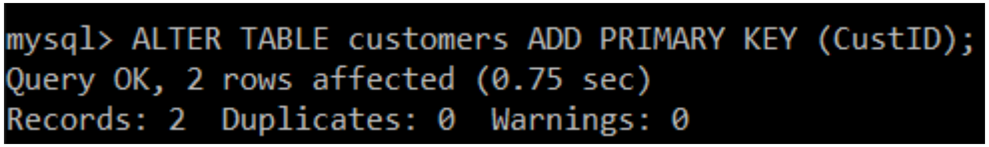
```
mysql> CREATE TABLE Payment (
    -> PaymentID INT NOT NULL,
    -> CustID INT,
    -> Sum INT,
    -> PaymentMethod ENUM('PayPal','Cash','Wire'),
    -> PaymentTime TIMESTAMP,
    -> PRIMARY KEY (PaymentID)
    -> );
Query OK, 0 rows affected (0.23 sec)

mysql> describe payment;
+---------------+-------------------------------+------+-----+---------+-------+
| Field         | Type                          | Null | Key | Default | Extra |
+---------------+-------------------------------+------+-----+---------+-------+
| PaymentID     | int(11)                       | NO   | PRI | NULL    |       |
| CustID        | int(11)                       | YES  |     | NULL    |       |
| Sum           | int(11)                       | YES  |     | NULL    |       |
| PaymentMethod | enum('PayPal','Cash','Wire')  | YES  |     | NULL    |       |
| PaymentTime   | timestamp                     | YES  |     | NULL    |       |
+---------------+-------------------------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Setting a Primary Key on an Existing Table

- Once we have ensured no duplicate field values exist, we can promote the CustID field to be our primary key.
- We'll use the ALTER TABLE command:

```
ALTER TABLE customers ADD PRIMARY KEY(CustID);
```

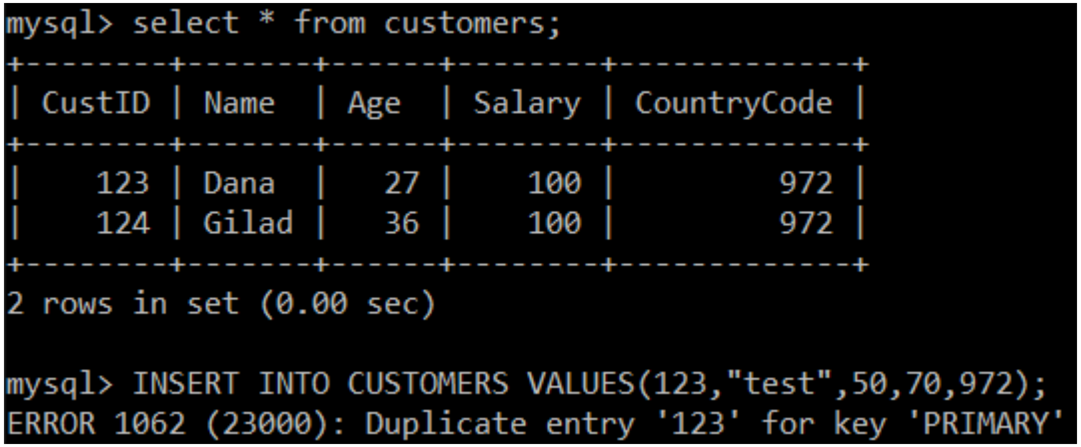


```
mysql> ALTER TABLE customers ADD PRIMARY KEY (CustID);
Query OK, 2 rows affected (0.75 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Testing the Constraint

- Now, let’s try to INSERT a duplicate ID:

```
INSERT INTO customers
VALUES(123,’test’,50,70,972);
```

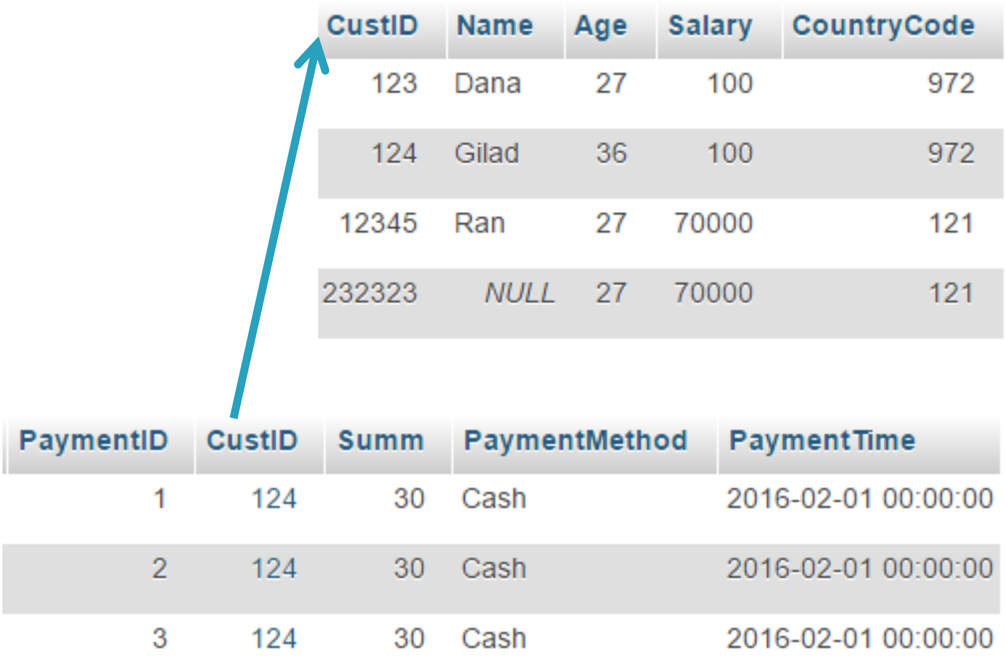


```
mysql> select * from customers;
+--------+-------+-----+--------+-------------+
| CustID | Name  | Age | Salary | CountryCode |
+--------+-------+-----+--------+-------------+
|    123 | Dana  |  27 |    100 |         972 |
|    124 | Gilad |  36 |    100 |         972 |
+--------+-------+-----+--------+-------------+
2 rows in set (0.00 sec)

mysql> INSERT INTO CUSTOMERS VALUES(123,"test",50,70,972);
ERROR 1062 (23000): Duplicate entry '123' for key 'PRIMARY'
```

Foreign Keys

- We can use a column of one table in another table.
- A column(s) in one table whose value matches a primary key's value in some other table, is called a foreign key



CustID	Name	Age	Salary	CountryCode
123	Dana	27	100	972
124	Gilad	36	100	972
12345	Ran	27	70000	121
232323	*NULL*	27	70000	121

PaymentID	CustID	Summ	PaymentMethod	PaymentTime
1	124	30	Cash	2016-02-01 00:00:00
2	124	30	Cash	2016-02-01 00:00:00
3	124	30	Cash	2016-02-01 00:00:00

Foreign Keys

Description

★ The foreign key identifies a column or set of columns in one (referencing) table which refer to a column or set of columns in another (referenced) table

★ The primary key and the foreign key form a **parent / child relationship** between the two tables

★ **Similar Types** - the types for the individual columns linked between the parent and child tables must be the same

Foreign Keys - Referential Actions

- There are 5 different referential actions:
  - **CASCADE:** If parent row is deleted then delete the child / children row(s) as well. If updated, the child is updated as well
  - **RESTRICT:** A parent row cannot be deleted if there are references to it from a child / children row(s)
  - **SET NULL:** Set the child's field value to NULL
  - **SET DEFAULT:** Set the child's field value to its default value

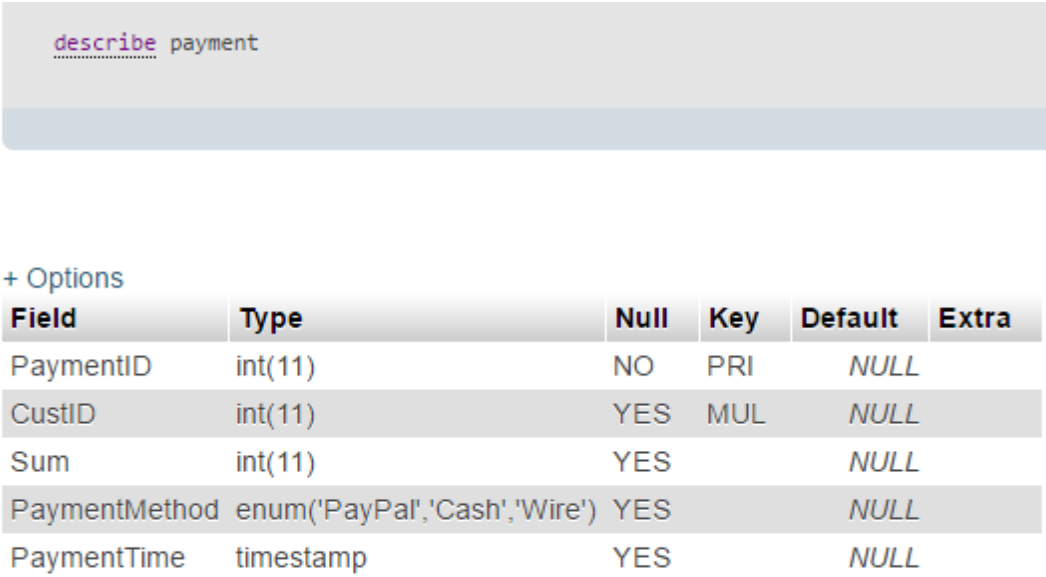
```
CREATE TABLE payments
...
FOREIGN KEY (CustID)
REFERENCES customers(CustID)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
...
```

Setting a Foreign Key When Creating a Table

```
CREATE TABLE payments(
PaymentID INT NOT NULL AUTO_INCREMENT,
CustID INT NOT NULL,
Sum INT NOT NULL,
PaymentMethod ENUM('PayPal','Cash','Wire') NOT NULL DEFAULT 'PayPal',
PaymentTime TIMESTAMP NOT NULL,
PRIMARY KEY (PaymentID),
FOREIGN KEY (CustID)
   REFERENCES customers(CustID)
      ON UPDATE CASCADE
      ON DELETE RESTRICT
);
```

Setting a Foreign Key on an Existing Table

```
ALTER TABLE payments
ADD CONSTRAINT FK_payments1
FOREIGN KEY (CustID)
REFERENCES customers(CustID)
     ON UPDATE CASCADE
     ON DELETE RESTRICT;
```

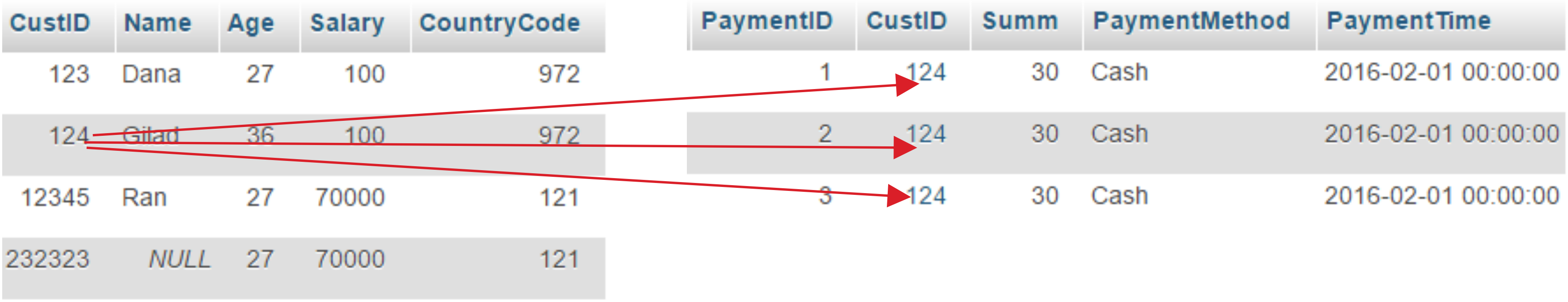


```
describe payment
```

+ Options

Field	Type	Null	Key	Default	Extra
PaymentID	int(11)	NO	PRI	*NULL*	
CustID	int(11)	YES	MUL	*NULL*	
Sum	int(11)	YES		*NULL*	
PaymentMethod	enum('PayPal','Cash','Wire')	YES		*NULL*	
PaymentTime	timestamp	YES		*NULL*	

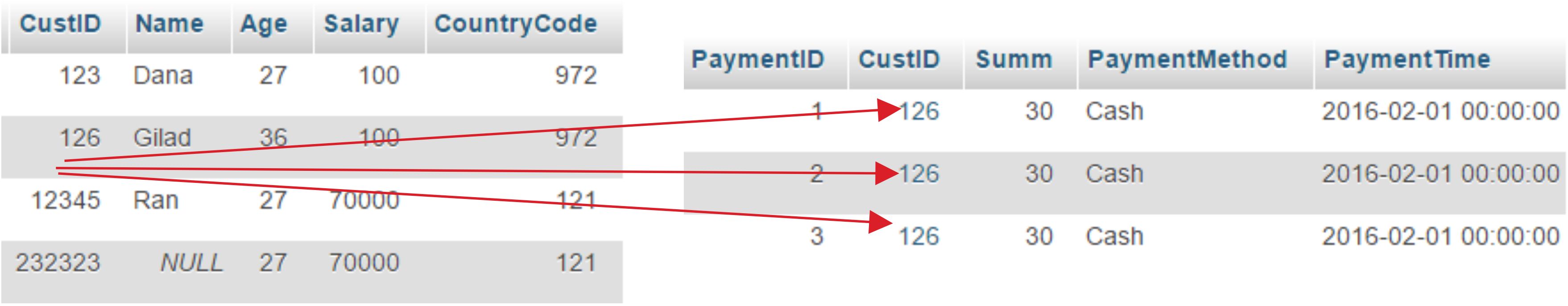
Update Customer -> Updates Payment



CustID	Name	Age	Salary	CountryCode
123	Dana	27	100	972
124	Gilad	36	100	972
12345	Ran	27	70000	121
232323	*NULL*	27	70000	121

PaymentID	CustID	Summ	PaymentMethod	PaymentTime
1	124	30	Cash	2016-02-01 00:00:00
2	124	30	Cash	2016-02-01 00:00:00
3	124	30	Cash	2016-02-01 00:00:00

```
UPDATE customers
SET CustID = 126
WHERE CustID=124;
```

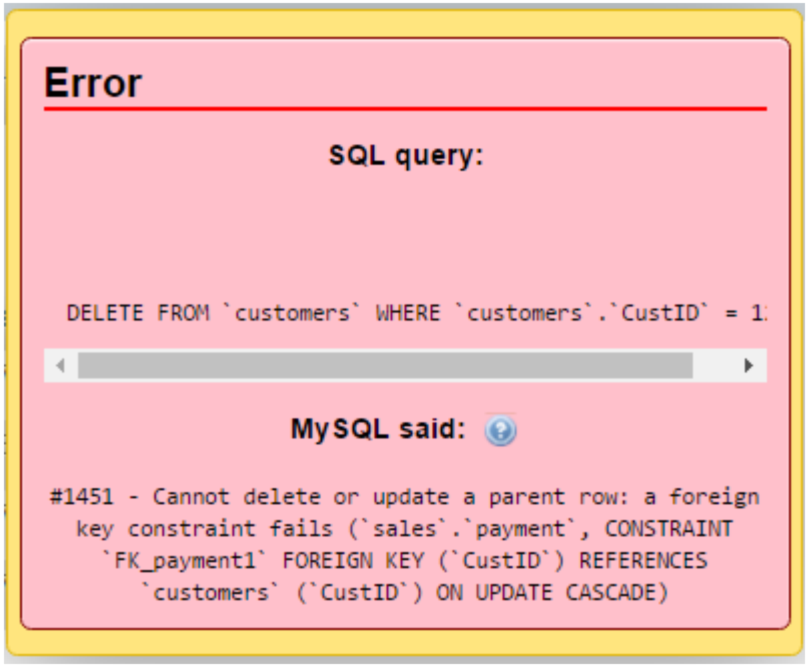


CustID	Name	Age	Salary	CountryCode
123	Dana	27	100	972
126	Gilad	36	100	972
12345	Ran	27	70000	121
232323	*NULL*	27	70000	121

PaymentID	CustID	Summ	PaymentMethod	PaymentTime
1	126	30	Cash	2016-02-01 00:00:00
2	126	30	Cash	2016-02-01 00:00:00
3	126	30	Cash	2016-02-01 00:00:00

Can’t Delete – RESTRICT

```
DELETE FROM customers WHERE CustID=126;
```



Error

SQL query:

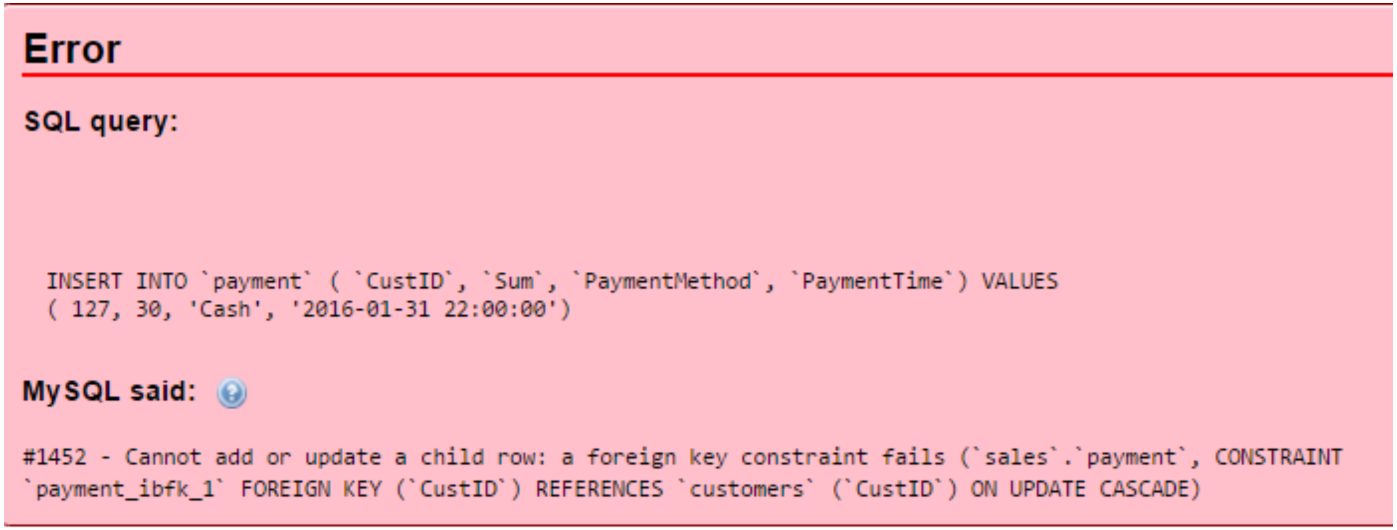
```
DELETE FROM `customers` WHERE `customers`.`CustID` = 1:
```

MySQL said:

```
#1451 - Cannot delete or update a parent row: a foreign
key constraint fails (`sales`.`payment`, CONSTRAINT
`FK_payment1` FOREIGN KEY (`CustID`) REFERENCES
`customers` (`CustID`) ON UPDATE CASCADE)
```

Inserting not existing CustID will fail

```
INSERT INTO `payment` ( `CustID`, `Sum`, `PaymentMethod`, `PaymentTime`)
VALUES( 127, 30, 'Cash', '2016-01-31 22:00:00');
```



Error

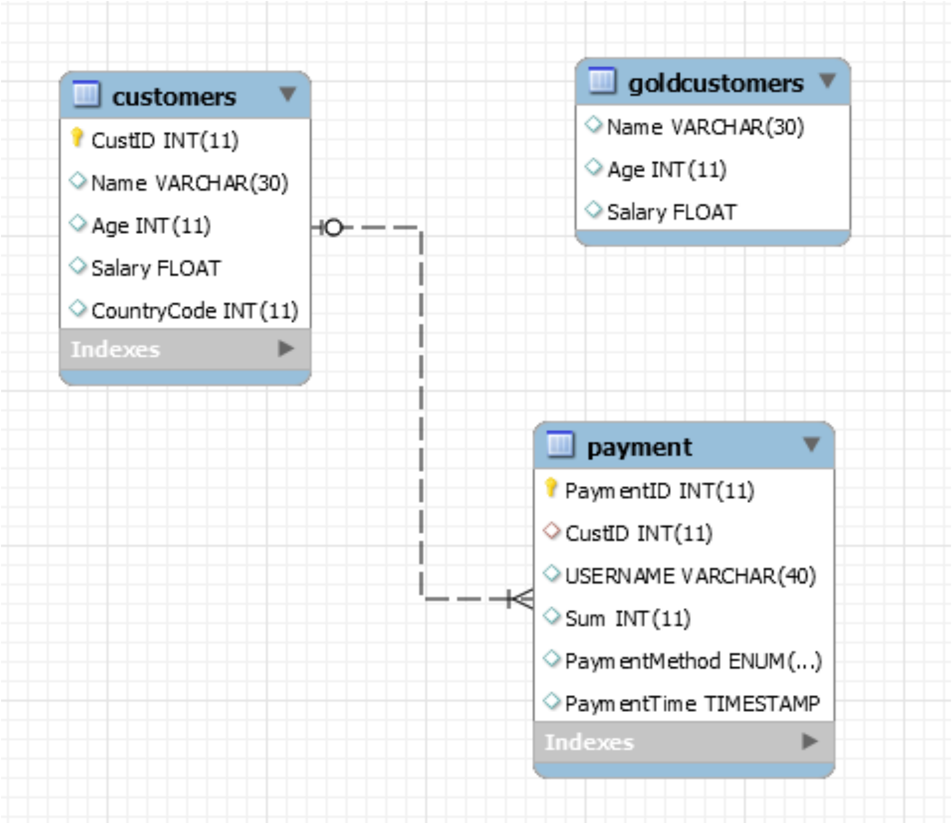
SQL query:

```
INSERT INTO `payment` ( `CustID`, `Sum`, `PaymentMethod`, `PaymentTime`) VALUES
( 127, 30, 'Cash', '2016-01-31 22:00:00')
```

MySQL said:

```
#1452 - Cannot add or update a child row: a foreign key constraint fails (`sales`.`payment`, CONSTRAINT
`payment_ibfk_1` FOREIGN KEY (`CustID`) REFERENCES `customers` (`CustID`) ON UPDATE CASCADE)
```

That’s how it looks



customers
- CustID INT(11)
- Name VARCHAR(30)
- Age INT(11)
- Salary FLOAT
- CountryCode INT(11)
- Indexes

goldcustomers
- Name VARCHAR(30)
- Age INT(11)
- Salary FLOAT

payment
- PaymentID INT(11)
- CustID INT(11)
- USERNAME VARCHAR(40)
- Sum INT(11)
- PaymentMethod ENUM(...)
- PaymentTime TIMESTAMP
- Indexes

UNIQUE

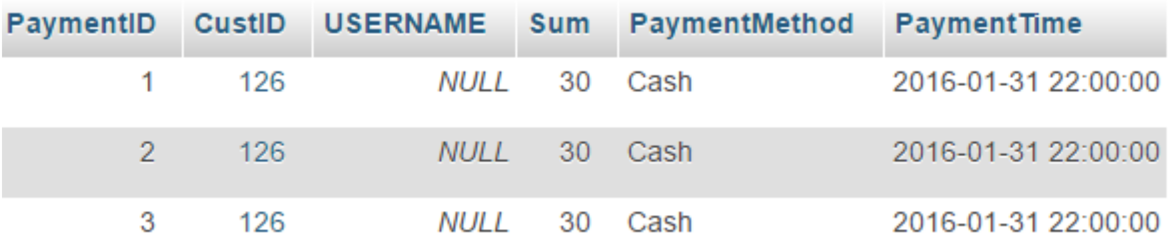
- What if we wanted to ensure a unique value in a field which is not a primary key?

```
CREATE TABLE payments(
   Sum INT NOT NULL,
   USERNAME VARCHAR(40) UNIQUE
);
```

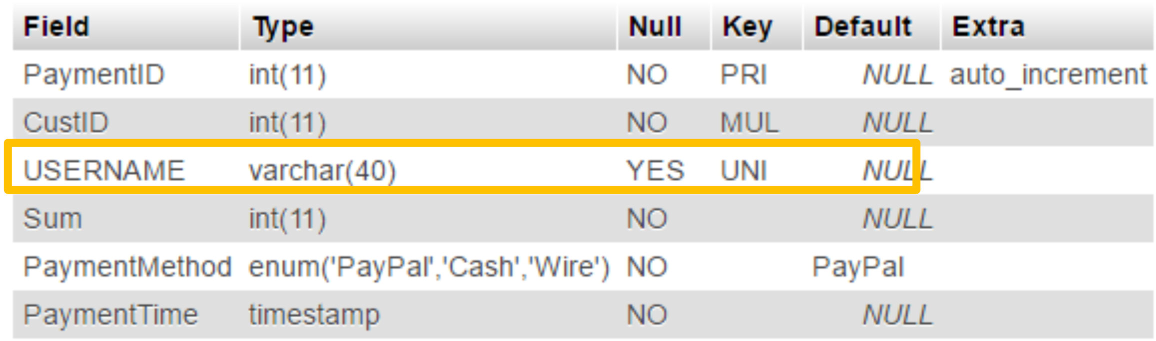
UNIQUE

- Or add a new unique column?

```
ALTER TABLE payment
ADD COLUMN USERNAME VARCHAR(40) UNIQUE AFTER CustID;
```



PaymentID	CustID	USERNAME	Sum	PaymentMethod	PaymentTime
1	126	*NULL*	30	Cash	2016-01-31 22:00:00
2	126	*NULL*	30	Cash	2016-01-31 22:00:00
3	126	*NULL*	30	Cash	2016-01-31 22:00:00

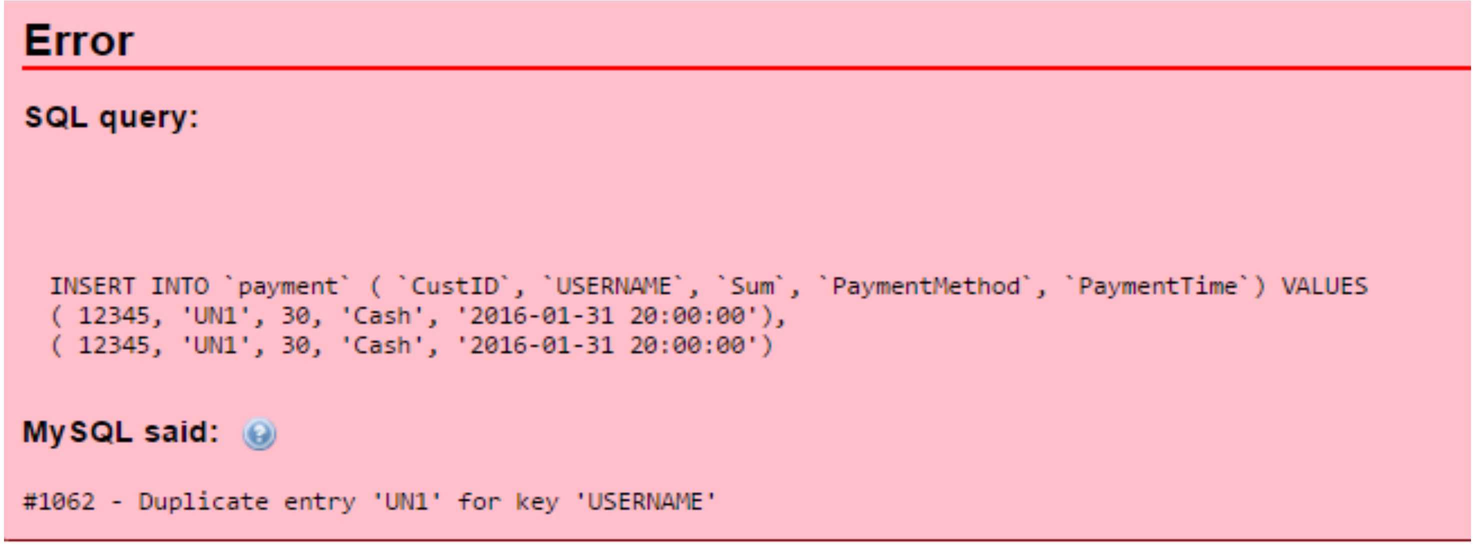


Field	Type	Null	Key	Default	Extra
PaymentID	int(11)	NO	PRI	*NULL*	auto_increment
CustID	int(11)	NO	MUL	*NULL*	
USERNAME	varchar(40)	YES	UNI	*NULL*	
Sum	int(11)	NO		*NULL*	
PaymentMethod	enum('PayPal','Cash','Wire')	NO		PayPal	
PaymentTime	timestamp	NO		*NULL*	

UNIQUE Means Unique

- It is possible to add any number of NULL values, but not repeat and non-NULL value twice

```
INSERT INTO payment
(CustID,USERNAME,Sum,PaymentMethod,PaymentTime)
VALUES
( 12345, 'UN1', 30, 'Cash', '2016-01-31 20:00:00'),
( 12345, 'UN1', 30, 'Cash', '2016-01-31 20:00:00')
```



Error

SQL query:

```
INSERT INTO `payment` ( `CustID`, `USERNAME`, `Sum`, `PaymentMethod`, `PaymentTime`) VALUES
( 12345, 'UN1', 30, 'Cash', '2016-01-31 20:00:00'),
( 12345, 'UN1', 30, 'Cash', '2016-01-31 20:00:00')
```

MySQL said:

```
#1062 - Duplicate entry 'UN1' for key 'USERNAME'
```

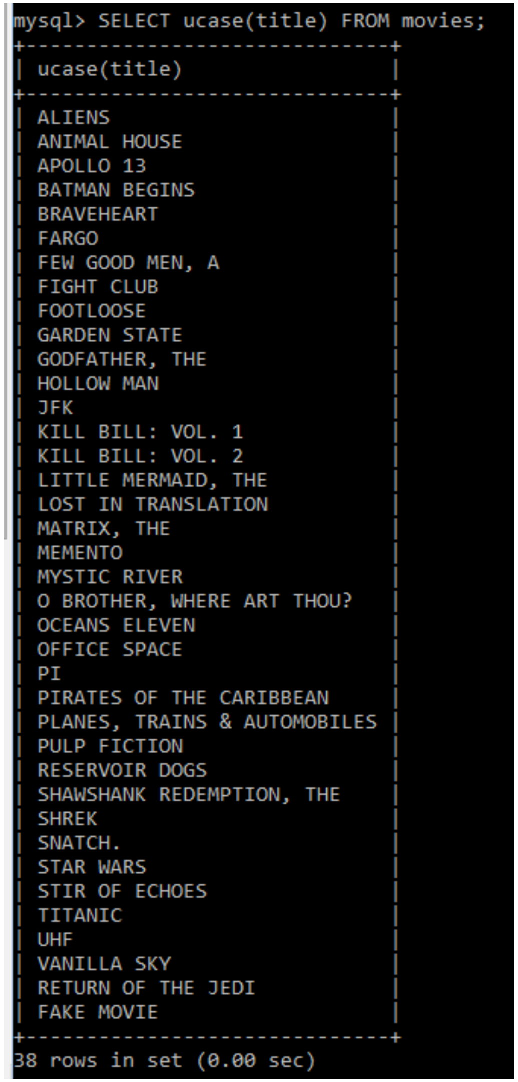
Questions?

SQL Scalar Functions - 1

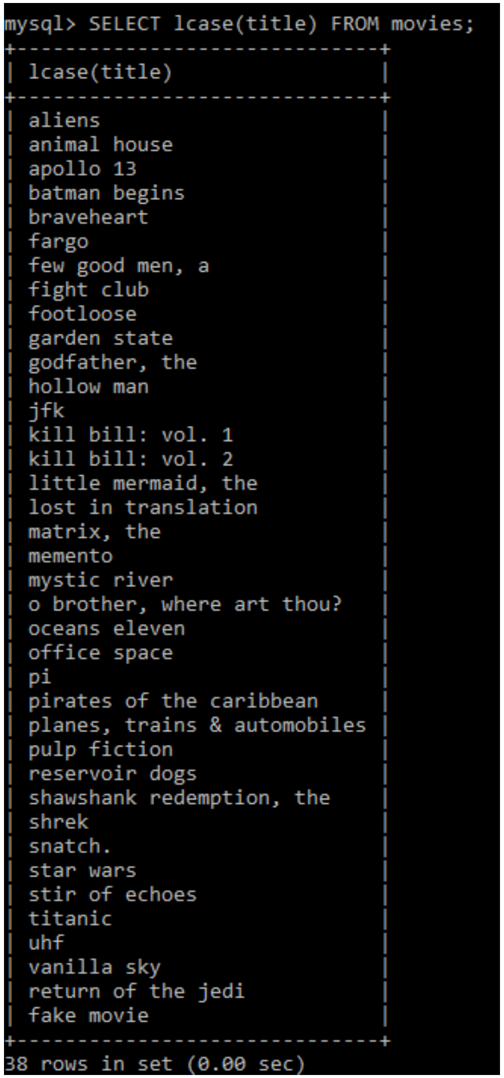
- These functions return a single value, based on the value of the field they take as parameter:

- `SELECT ucase(name) FROM customers;`
  - Equivalent to Python's str.upper() method
- `SELECT lcase(name) FROM customers;`
  - Equivalent to Python's str.lower() method
- `SELECT now();`
  - Returns the current system time
- More: **LEN, MID,ROUND,FORMAT**…

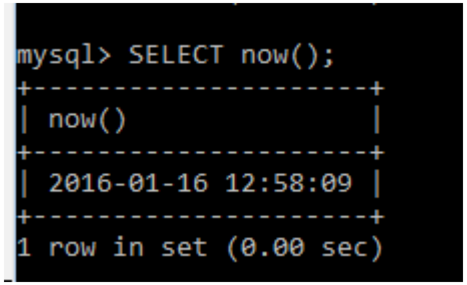
SQL Scalar functions - 2



```
mysql> SELECT ucase(title) FROM movies;
+------------------------------+
| ucase(title)                 |
+------------------------------+
| ALIENS                       |
| ANIMAL HOUSE                 |
| APOLLO 13                    |
| BATMAN BEGINS                |
| BRAVEHEART                   |
| FARGO                        |
| FEW GOOD MEN, A              |
| FIGHT CLUB                   |
| FOOTLOOSE                    |
| GARDEN STATE                 |
| GODFATHER, THE               |
| HOLLOW MAN                   |
| JFK                          |
| KILL BILL: VOL. 1            |
| KILL BILL: VOL. 2            |
| LITTLE MERMAID, THE          |
| LOST IN TRANSLATION          |
| MATRIX, THE                  |
| MEMENTO                      |
| MYSTIC RIVER                 |
| O BROTHER, WHERE ART THOU?   |
| OCEANS ELEVEN                |
| OFFICE SPACE                 |
| PI                           |
| PIRATES OF THE CARIBBEAN     |
| PLANES, TRAINS & AUTOMOBILES |
| PULP FICTION                 |
| RESERVOIR DOGS               |
| SHAWSHANK REDEMPTION, THE    |
| SHREK                        |
| SNATCH.                      |
| STAR WARS                    |
| STIR OF ECHOES               |
| TITANIC                      |
| UHF                          |
| VANILLA SKY                  |
| RETURN OF THE JEDI           |
| FAKE MOVIE                   |
+------------------------------+
38 rows in set (0.00 sec)
```



```
mysql> SELECT lcase(title) FROM movies;
+------------------------------+
| lcase(title)                 |
+------------------------------+
| aliens                       |
| animal house                 |
| apollo 13                    |
| batman begins                |
| braveheart                   |
| fargo                        |
| few good men, a              |
| fight club                   |
| footloose                    |
| garden state                 |
| godfather, the               |
| hollow man                   |
| jfk                          |
| kill bill: vol. 1            |
| kill bill: vol. 2            |
| little mermaid, the          |
| lost in translation          |
| matrix, the                  |
| memento                      |
| mystic river                 |
| o brother, where art thou?   |
| oceans eleven                |
| office space                 |
| pi                           |
| pirates of the caribbean     |
| planes, trains & automobiles |
| pulp fiction                 |
| reservoir dogs               |
| shawshank redemption, the    |
| shrek                        |
| snatch.                      |
| star wars                    |
| stir of echoes               |
| titanic                      |
| uhf                          |
| vanilla sky                  |
| return of the jedi           |
| fake movie                   |
+------------------------------+
38 rows in set (0.00 sec)
```

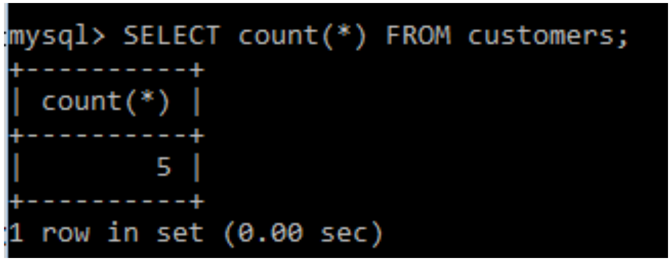


```
mysql> SELECT now();
+---------------------+
| now()               |
+---------------------+
| 2016-01-16 12:58:09 |
+---------------------+
1 row in set (0.00 sec)
```

SQL Aggregate Functions - 1

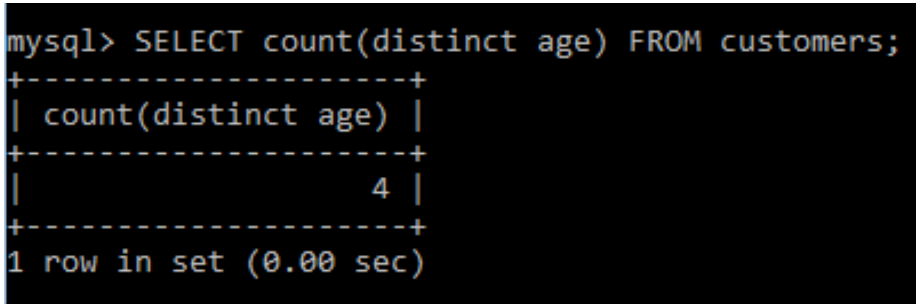
- These functions perform an operation on the values of all the fields matching the WHERE clause condition:

```
SELECT count(*) FROM customers;
```



```
mysql> SELECT count(*) FROM customers;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

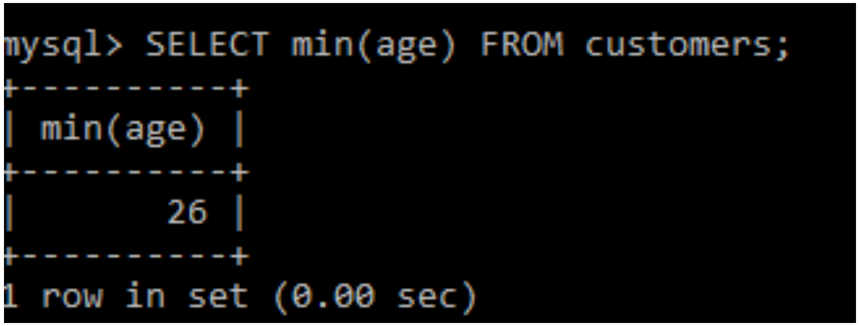
```
SELECT count(distinct age) FROM customers;
```



```
mysql> SELECT count(distinct age) FROM customers;
+---------------------+
| count(distinct age) |
+---------------------+
|                   4 |
+---------------------+
1 row in set (0.00 sec)
```

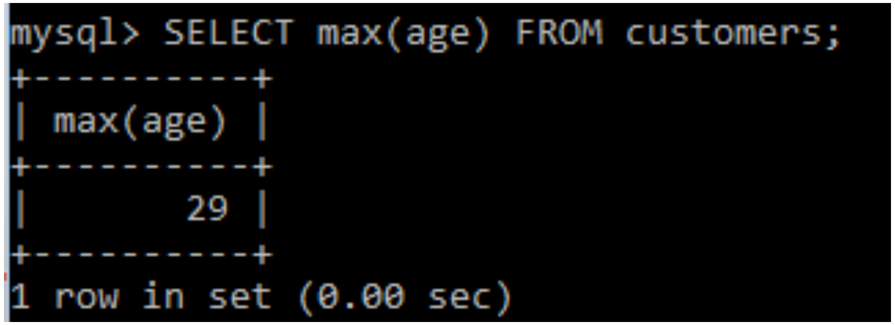
SQL Aggregate Functions - 2

```
SELECT min(age) FROM customers;
```



```
mysql> SELECT min(age) FROM customers;
+----------+
| min(age) |
+----------+
|       26 |
+----------+
1 row in set (0.00 sec)
```

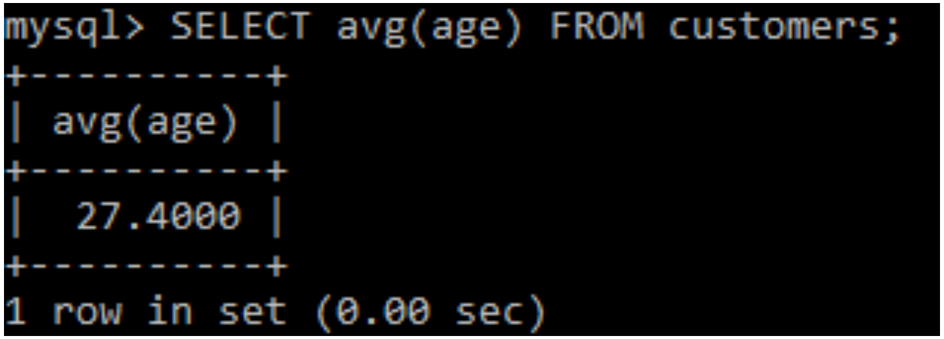
```
SELECT max(age) FROM customers;
```



```
mysql> SELECT max(age) FROM customers;
+----------+
| max(age) |
+----------+
|       29 |
+----------+
1 row in set (0.00 sec)
```

SQL Aggregate Functions - 2

```
SELECT avg(age) FROM customers;
```

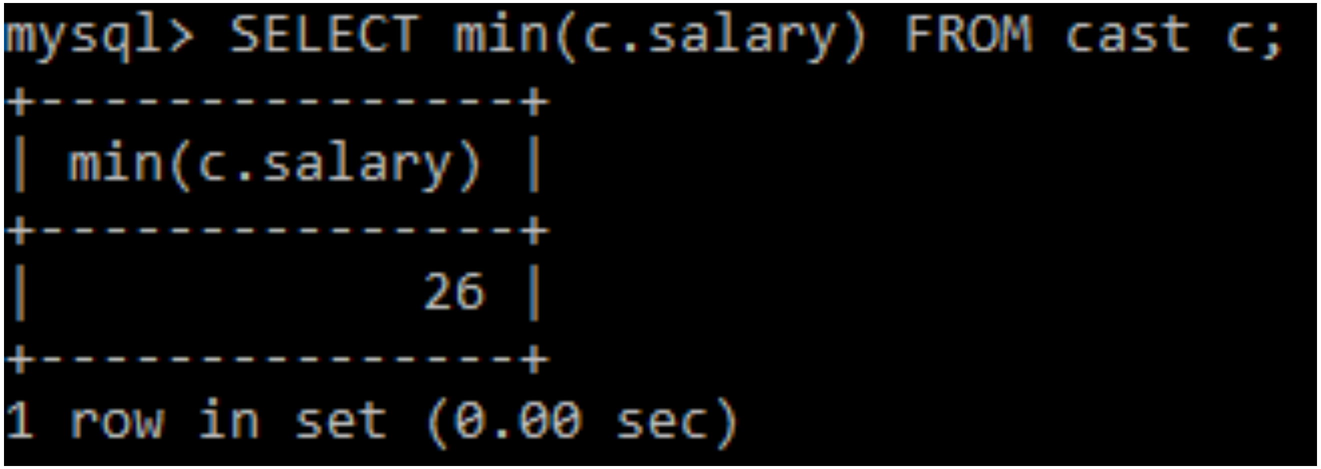


```
mysql> SELECT avg(age) FROM customers;
+----------+
| avg(age) |
+----------+
|  27.4000 |
+----------+
1 row in set (0.00 sec)
```

Aliasing

- Aliasing is a convenience feature allowing for assigning shorter names to fields and tables.
- Useful if the aliased names need to be referenced multiple times in the query.

```
SELECT min(c.salary)
FROM cast as c;
```

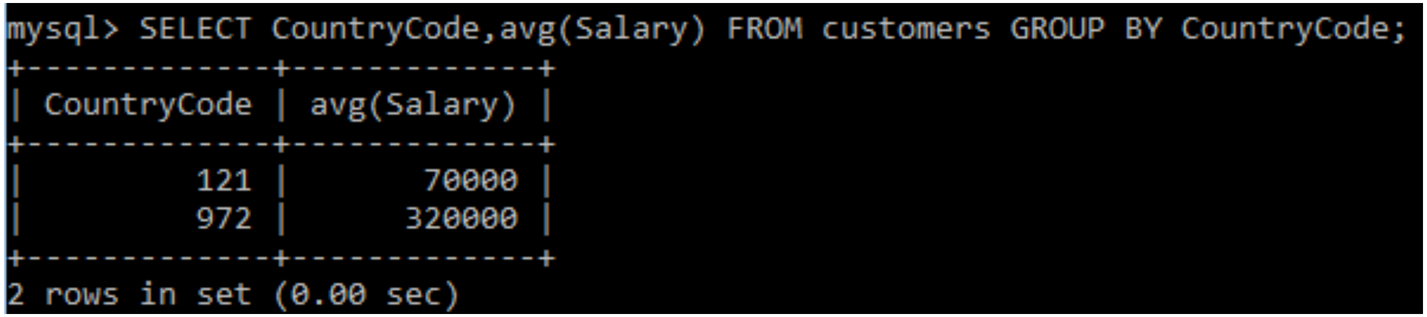


```
mysql> SELECT min(c.salary) FROM cast c;
+----------------+
| min(c.salary) |
+----------------+
|             26 |
+----------------+
1 row in set (0.00 sec)
```

The Group By Clause

- Aggregate functions can be made to work on distinct subsets of the result-set, also know as a “**group**”
- The grouping factor is the value of a given field
- In other words, the aggregate function will be separately applied to all the rows matching the grouping value

```
SELECT CountryCode, avg(Salary)
FROM customers
GROUP BY CountryCode;
```



```
mysql> SELECT CountryCode,avg(Salary) FROM customers GROUP BY CountryCode;
+-------------+-------------+
| CountryCode | avg(Salary) |
+-------------+-------------+
|         121 |       70000 |
|         972 |      320000 |
+-------------+-------------+
2 rows in set (0.00 sec)
```

The Group By Clause

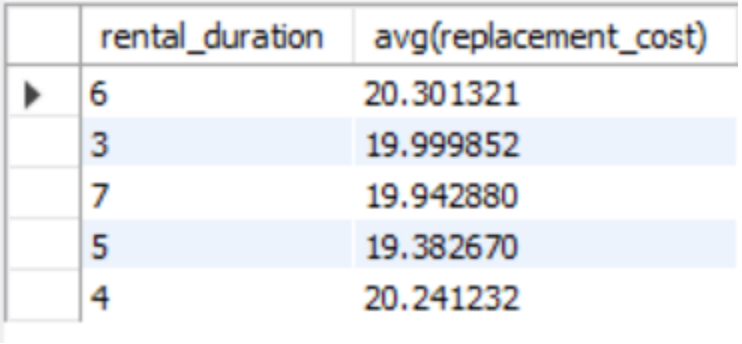
- It is possible to Group By more than one field
- Using a single field to Group By, we get a single row per unique value of the field
- Using two fields to Group By, we get a single row per combination of the possible unique values for the two fields
- For example, if we want to Group By *department* and within a given department Group By *gender*, then we will end-up with: number of departments * 2 rows

```
...
GROUP BY department, gender;
```

Group By Example – Film Rental DB

- Get the average replacement cost for each rental duration

```
SELECT rental_duration, AVG(replacement_cost)
FROM film
group by rental_duration
```

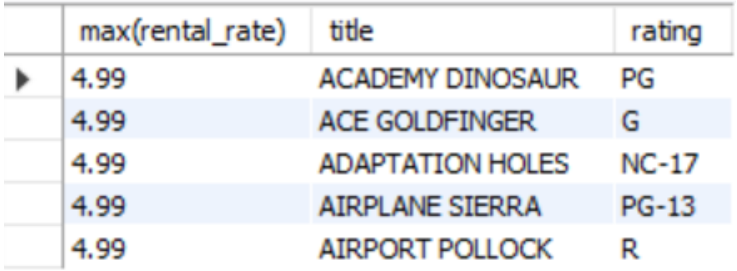


rental_duration	avg(replacement_cost)
6	20.301321
3	19.999852
7	19.942880
5	19.382670
4	20.241232

Group By Example – Film Rental DB

- Get the films with the maximum rental rate for each rating type

```
SELECT MAX(rental_rate), title, rating
FROM film
GROUP BY rating
```



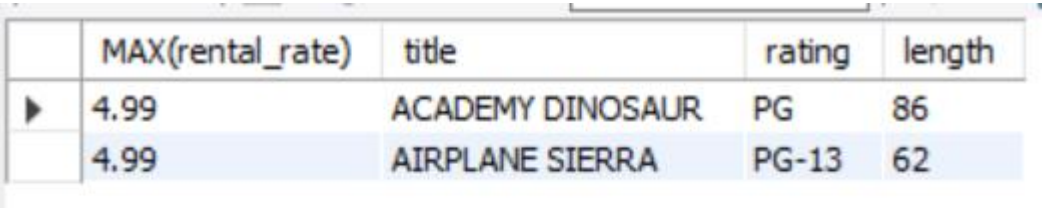
max(rental_rate)	title	rating
4.99	ACADEMY DINOSAUR	PG
4.99	ACE GOLDFINGER	G
4.99	ADAPTATION HOLES	NC-17
4.99	AIRPLANE SIERRA	PG-13
4.99	AIRPORT POLLOCK	R

The Having Clause

- Complements the GROUP BY clause
    - It can’t do without it…
- Allows for filtering on aggregate values
- It lets us narrow down the list of aggregated results returned by applying the GROUP BY clause

- Get the films with the maximum rental rate for each rating type which are longer than 60 minutes

```
SELECT MAX(rental_rate), title, rating, length
FROM film
GROUP BY rating
HAVING length > 60
```

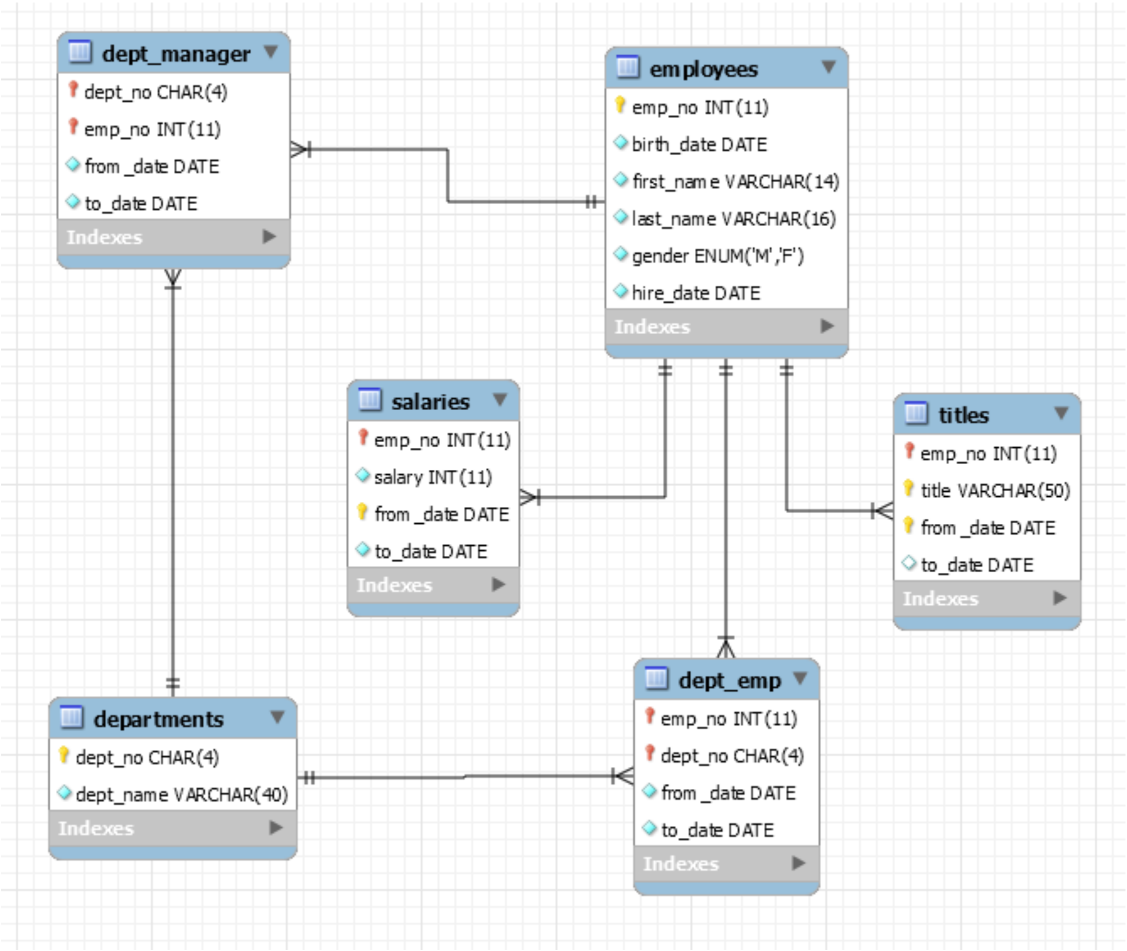


MAX(rental_rate)	title	rating	length
4.99	ACADEMY DINOSAUR	PG	86
4.99	AIRPLANE SIERRA	PG-13	62

ERD

- An **Entity-Relationship Diagram**, or **ERD**, is a chart which visually represents the relationship between database entities
- An ERD is comprised of three main components: entities, attributes, and relationships

Example



dept_manager
dept_no CHAR(4)
emp_no INT(11)
from _date DATE
to_date DATE

employees
emp_no INT(11)
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
gender ENUM('M','F')
hire_date DATE

salaries
emp_no INT(11)
salary INT(11)
from _date DATE
to_date DATE

titles
emp_no INT(11)
title VARCHAR(50)
from _date DATE
to_date DATE

departments
dept_no CHAR(4)
dept_name VARCHAR(40)

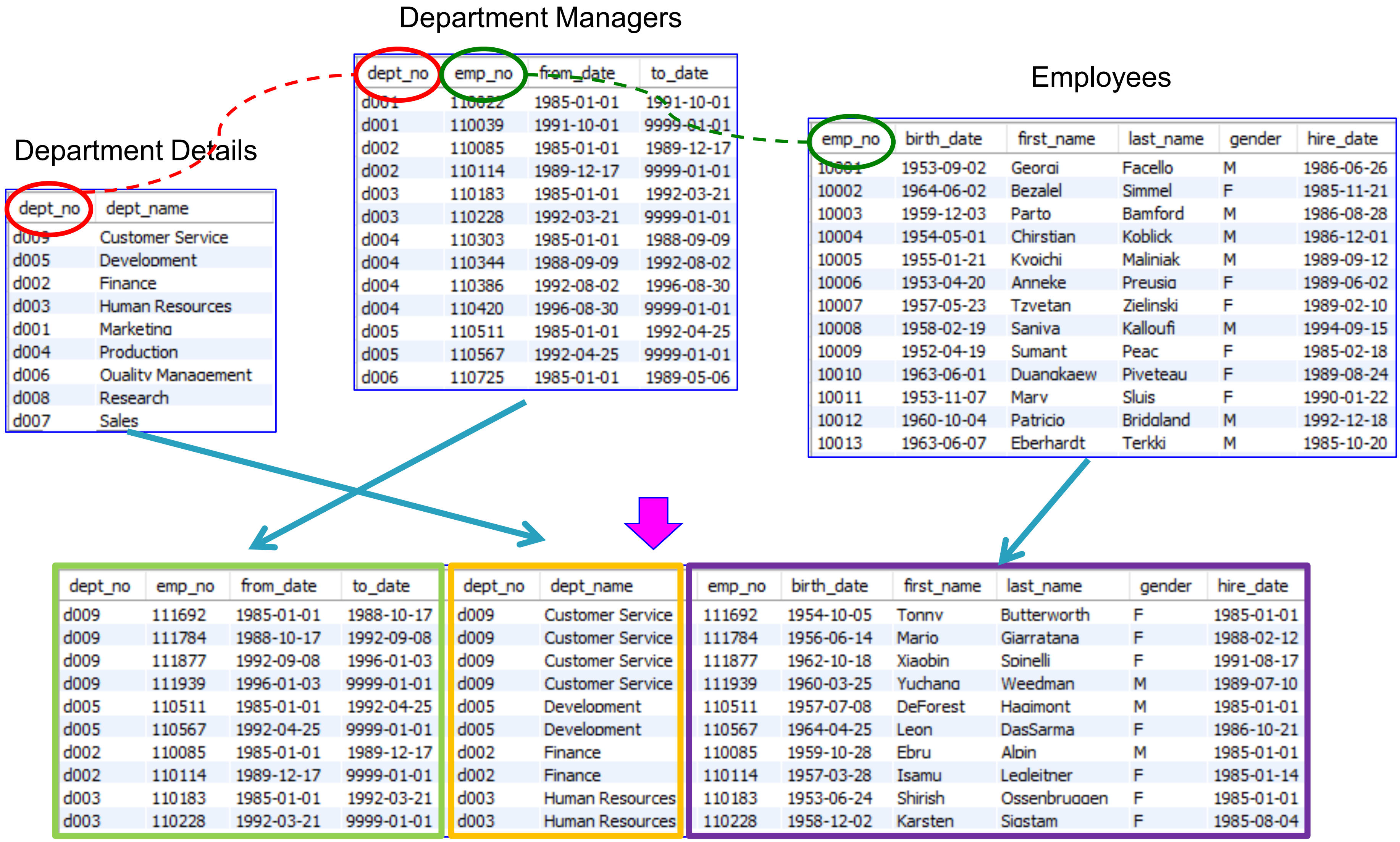
dept_emp
emp_no INT(11)
dept_no CHAR(4)
from _date DATE
to_date DATE

https://dev.mysql.com/doc/employee/en/employees-installation.html

Join

- Sometimes we have 2 or more tables with interrelated data, like actors, movies and cast (or customers and payments)
- When we have a mutual piece of data (i.e., same value with the same meaning), in 2 (or more) tables, we can **JOIN** the tables, creating a virtual table comprised of both (all) tables.
- A simple example:
  - We would like to display the customer’s name next to every payment made
  - The payments data is in the **`payment`** table, while the customer’s name is in the **`customers`** table
  - Both tables have a Customer ID column

Join



Department Details

dept_no	dept_name
d009	Customer Service
d005	Development
d002	Finance
d003	Human Resources
d001	Marketing
d004	Production
d006	Quality Management
d008	Research
d007	Sales

Department Managers

dept_no	emp_no	from_date	to_date
d001	110022	1985-01-01	1991-10-01
d001	110039	1991-10-01	9999-01-01
d002	110085	1985-01-01	1989-12-17
d002	110114	1989-12-17	9999-01-01
d003	110183	1985-01-01	1992-03-21
d003	110228	1992-03-21	9999-01-01
d004	110303	1985-01-01	1988-09-09
d004	110344	1988-09-09	1992-08-02
d004	110386	1992-08-02	1996-08-30
d004	110420	1996-08-30	9999-01-01
d005	110511	1985-01-01	1992-04-25
d005	110567	1992-04-25	9999-01-01
d006	110725	1985-01-01	1989-05-06

Employees

emp_no	birth_date	first_name	last_name	gender	hire_date
10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20

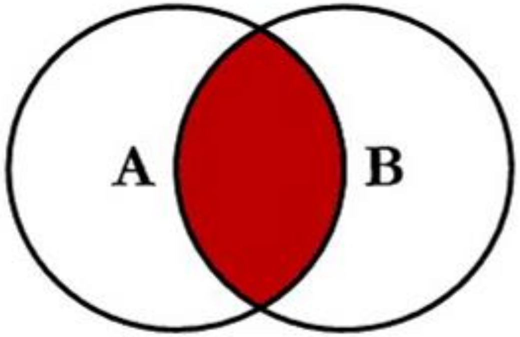
dept_no	emp_no	from_date	to_date
d009	111692	1985-01-01	1988-10-17
d009	111784	1988-10-17	1992-09-08
d009	111877	1992-09-08	1996-01-03
d009	111939	1996-01-03	9999-01-01
d005	110511	1985-01-01	1992-04-25
d005	110567	1992-04-25	9999-01-01
d002	110085	1985-01-01	1989-12-17
d002	110114	1989-12-17	9999-01-01
d003	110183	1985-01-01	1992-03-21
d003	110228	1992-03-21	9999-01-01

dept_no	dept_name
d009	Customer Service
d009	Customer Service
d009	Customer Service
d009	Customer Service
d005	Development
d005	Development
d002	Finance
d002	Finance
d003	Human Resources
d003	Human Resources

emp_no	birth_date	first_name	last_name	gender	hire_date
111692	1954-10-05	Tonny	Butterworth	F	1985-01-01
111784	1956-06-14	Mario	Giarratana	F	1988-02-12
111877	1962-10-18	Xiaobin	Spinelli	F	1991-08-17
111939	1960-03-25	Yuchang	Weedman	M	1989-07-10
110511	1957-07-08	DeForest	Hagimont	M	1985-01-01
110567	1964-04-25	Leon	DasSarma	F	1986-10-21
110085	1959-10-28	Ebru	Alpin	M	1985-01-01
110114	1957-03-28	Isamu	Legleitner	F	1985-01-14
110183	1953-06-24	Shirish	Ossenbruggen	F	1985-01-01
110228	1958-12-02	Karsten	Sigstam	F	1985-08-04

Join Types – Inner Join

- **INNER JOIN**: Returns only rows matching on the JOIN field from both left and right tables



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

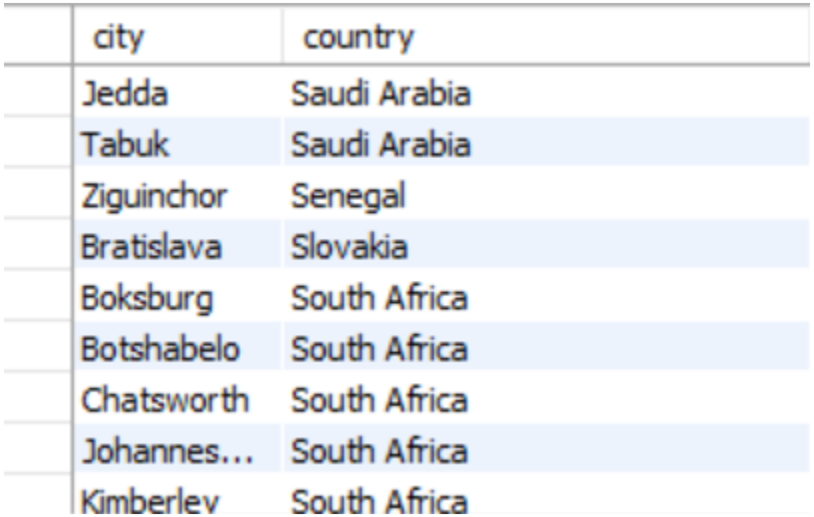
Join Types – Inner Join - Example

- **INNER JOIN**: Returns only rows matching on the JOIN field from both left and right tables

```
SELECT ci.city, co.country
FROM city AS ci
INNER JOIN country AS co
ON ci.country_id = co.country_id
WHERE co.country LIKE "S%"
```

Select cities and their countries from both “country” and “city” tables, based on the `country_id` key.

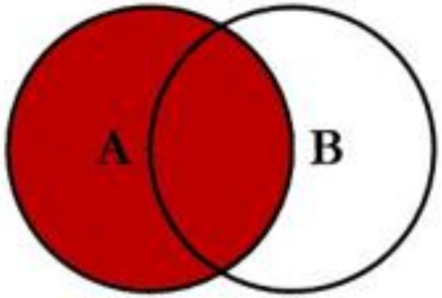
The countries must start with an “S”



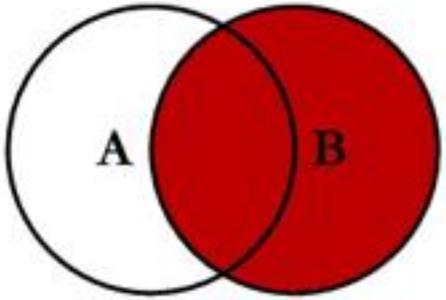
city	country
Jedda	Saudi Arabia
Tabuk	Saudi Arabia
Ziguinchor	Senegal
Bratislava	Slovakia
Boksburg	South Africa
Botshabelo	South Africa
Chatsworth	South Africa
Johannes...	South Africa
Kimberley	South Africa

Join Types – Left and Right Join

- **LEFT JOIN**: Returns all rows from the left table, and matched rows from the right table
- **RIGHT JOIN**: Returns all rows from the right table, and matched rows from the left table



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



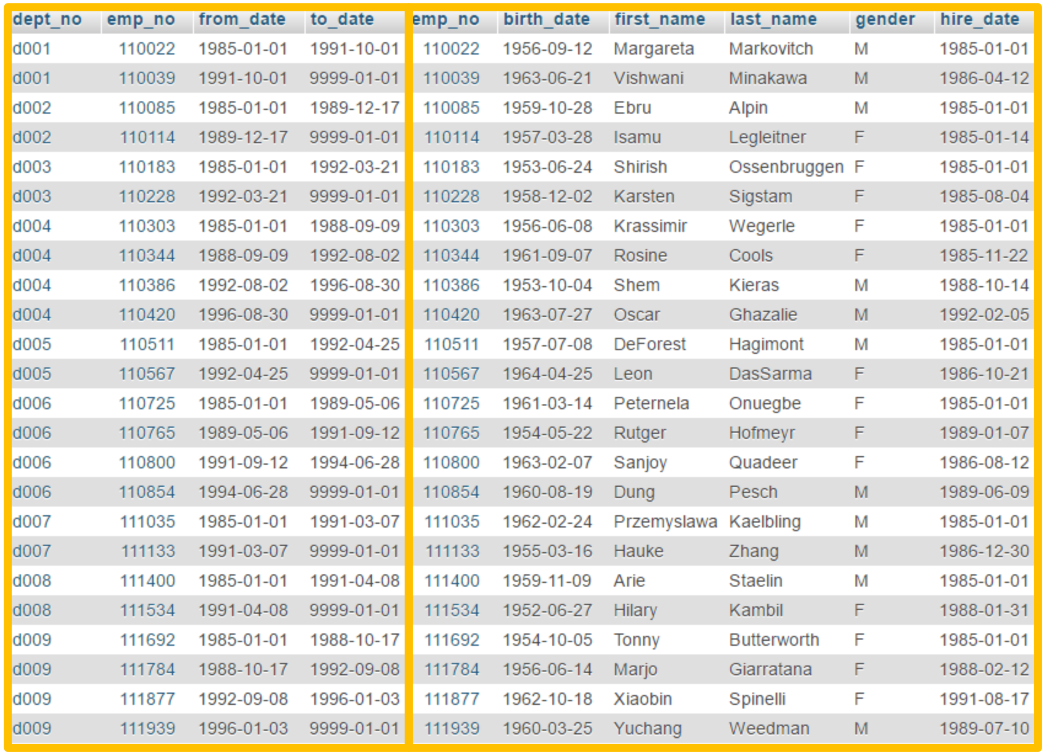
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

Left Join Example

- Fetch the details of employees who are **<u>also</u>** department managers

```
SELECT *
FROM dept_manager as dm
  LEFT JOIN employees as e
    ON dm.emp_no = e.emp_no
```

We do a left join because we need the details of each manager



dept_no	emp_no	from_date	to_date	emp_no	birth_date	first_name	last_name	gender	hire_date
d001	110022	1985-01-01	1991-10-01	110022	1956-09-12	Margareta	Markovitch	M	1985-01-01
d001	110039	1991-10-01	9999-01-01	110039	1963-06-21	Vishwani	Minakawa	M	1986-04-12
d002	110085	1985-01-01	1989-12-17	110085	1959-10-28	Ebru	Alpin	M	1985-01-01
d002	110114	1989-12-17	9999-01-01	110114	1957-03-28	Isamu	Legleitner	F	1985-01-14
d003	110183	1985-01-01	1992-03-21	110183	1953-06-24	Shirish	Ossenbruggen	F	1985-01-01
d003	110228	1992-03-21	9999-01-01	110228	1958-12-02	Karsten	Sigstam	F	1985-08-04
d004	110303	1985-01-01	1988-09-09	110303	1956-06-08	Krassimir	Wegerle	F	1985-01-01
d004	110344	1988-09-09	1992-08-02	110344	1961-09-07	Rosine	Cools	F	1985-11-22
d004	110386	1992-08-02	1996-08-30	110386	1953-10-04	Shem	Kieras	M	1988-10-14
d004	110420	1996-08-30	9999-01-01	110420	1963-07-27	Oscar	Ghazalie	M	1992-02-05
d005	110511	1985-01-01	1992-04-25	110511	1957-07-08	DeForest	Hagimont	M	1985-01-01
d005	110567	1992-04-25	9999-01-01	110567	1964-04-25	Leon	DasSarma	F	1986-10-21
d006	110725	1985-01-01	1989-05-06	110725	1961-03-14	Peternela	Onuegbe	F	1985-01-01
d006	110765	1989-05-06	1991-09-12	110765	1954-05-22	Rutger	Hofmeyr	F	1989-01-07
d006	110800	1991-09-12	1994-06-28	110800	1963-02-07	Sanjoy	Quadeer	F	1986-08-12
d006	110854	1994-06-28	9999-01-01	110854	1960-08-19	Dung	Pesch	M	1989-06-09
d007	111035	1985-01-01	1991-03-07	111035	1962-02-24	Przemyslawa	Kaelbling	M	1985-01-01
d007	111133	1991-03-07	9999-01-01	111133	1955-03-16	Hauke	Zhang	M	1986-12-30
d008	111400	1985-01-01	1991-04-08	111400	1959-11-09	Arie	Staelin	M	1985-01-01
d008	111534	1991-04-08	9999-01-01	111534	1952-06-27	Hilary	Kambil	F	1988-01-31
d009	111692	1985-01-01	1988-10-17	111692	1954-10-05	Tonny	Butterworth	F	1985-01-01
d009	111784	1988-10-17	1992-09-08	111784	1956-06-14	Marjo	Giarratana	F	1988-02-12
d009	111877	1992-09-08	1996-01-03	111877	1962-10-18	Xiaobin	Spinelli	F	1991-08-17
d009	111939	1996-01-03	9999-01-01	111939	1960-03-25	Yuchang	Weedman	M	1989-07-10

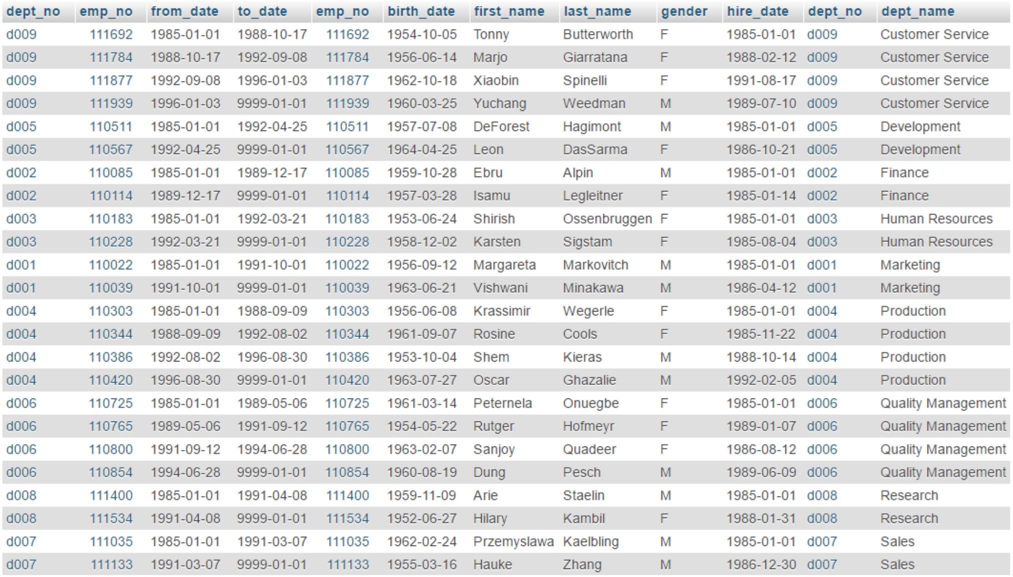
What differs this from an inner join? **The ‘also’.**

3-Way Left Join Example

- Same as the previous one, but now we also want the names of the departments (3 tables)

```
SELECT *
FROM dept_manager AS dm
  LEFT JOIN employees AS e
    ON dm.emp_no = e.emp_no
  LEFT JOIN departments AS de
    ON dm.dept_no = de.dept_no
```

We can opt to return any subset of columns we like.

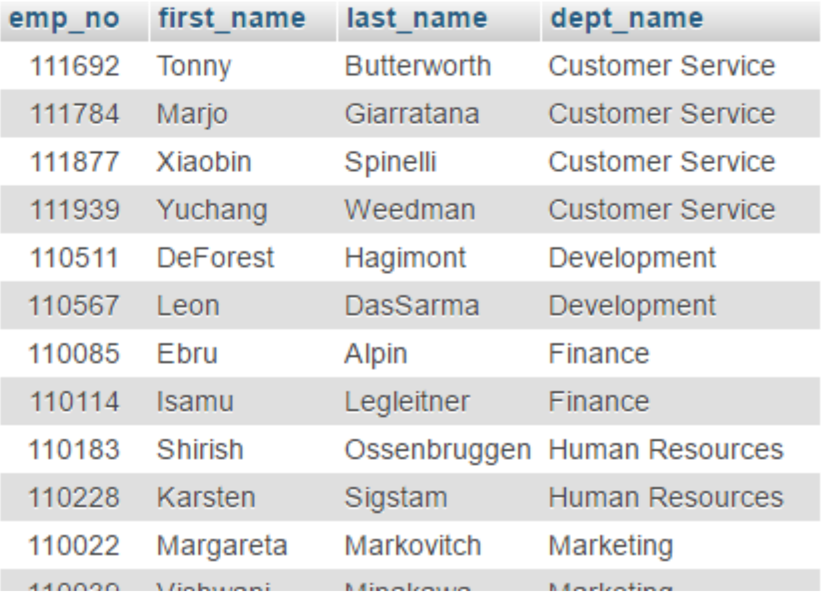


dept_no	emp_no	from_date	to_date	emp_no	birth_date	first_name	last_name	gender	hire_date	dept_no	dept_name
d009	111692	1985-01-01	1988-10-17	111692	1954-10-05	Tonny	Butterworth	F	1985-01-01	d009	Customer Service
d009	111784	1988-10-17	1992-09-08	111784	1956-06-14	Marjo	Giarratana	F	1988-02-12	d009	Customer Service
d009	111877	1992-09-08	1996-01-03	111877	1962-10-18	Xiaobin	Spinelli	F	1991-08-17	d009	Customer Service
d009	111939	1996-01-03	9999-01-01	111939	1960-03-25	Yuchang	Weedman	M	1989-07-10	d009	Customer Service
d005	110511	1985-01-01	1992-04-25	110511	1957-07-08	DeForest	Hagimont	M	1985-01-01	d005	Development
d005	110567	1992-04-25	9999-01-01	110567	1964-04-25	Leon	DasSarma	F	1986-10-21	d005	Development
d002	110085	1985-01-01	1989-12-17	110085	1959-10-28	Ebru	Alpin	M	1985-01-01	d002	Finance
d002	110114	1989-12-17	9999-01-01	110114	1957-03-28	Isamu	Legleitner	F	1985-01-14	d002	Finance
d003	110183	1985-01-01	1992-03-21	110183	1953-06-24	Shirish	Ossenbruggen	F	1985-01-01	d003	Human Resources
d003	110228	1992-03-21	9999-01-01	110228	1958-12-02	Karsten	Sigstam	F	1985-08-04	d003	Human Resources
d001	110022	1985-01-01	1991-10-01	110022	1956-09-12	Margareta	Markovitch	M	1985-01-01	d001	Marketing
d001	110039	1991-10-01	9999-01-01	110039	1963-06-21	Vishwani	Minakawa	M	1986-04-12	d001	Marketing
d004	110303	1985-01-01	1988-09-09	110303	1956-06-08	Krassimir	Wegerle	F	1985-01-01	d004	Production
d004	110344	1988-09-09	1992-08-02	110344	1961-09-07	Rosine	Cools	F	1985-11-22	d004	Production
d004	110386	1992-08-02	1996-08-30	110386	1953-10-04	Shem	Kieras	M	1988-10-14	d004	Production
d004	110420	1996-08-30	9999-01-01	110420	1963-07-27	Oscar	Ghazalie	M	1992-02-05	d004	Production
d006	110725	1985-01-01	1989-05-06	110725	1961-03-14	Peternela	Onuegbe	F	1985-01-01	d006	Quality Management
d006	110765	1989-05-06	1991-09-12	110765	1954-05-22	Rutger	Hofmeyr	F	1989-01-07	d006	Quality Management
d006	110800	1991-09-12	1994-06-28	110800	1963-02-07	Sanjoy	Quadeer	F	1986-08-12	d006	Quality Management
d006	110854	1994-06-28	9999-01-01	110854	1960-08-19	Dung	Pesch	M	1989-06-09	d006	Quality Management
d008	111400	1985-01-01	1991-04-08	111400	1959-11-09	Arie	Staelin	M	1985-01-01	d008	Research
d008	111534	1991-04-08	9999-01-01	111534	1952-06-27	Hilary	Kambil	F	1988-01-31	d008	Research
d007	111035	1985-01-01	1991-03-07	111035	1962-02-24	Przemyslawa	Kaelbling	M	1985-01-01	d007	Sales
d007	111133	1991-03-07	9999-01-01	111133	1955-03-16	Hauke	Zhang	M	1986-12-30	d007	Sales

Fetching Selected Columns Example

- Same as the previous one, but now we would like to fetch only a small subset of the columns

```
SELECT
  e.emp_no,
  e.first_name,
  e.last_name,
  de.dept_name
FROM dept_manager AS dm
  LEFT JOIN employees AS e
    ON dm.emp_no = e.emp_no
  LEFT JOIN departments AS de
    ON dm.dept_no = de.dept_no
```

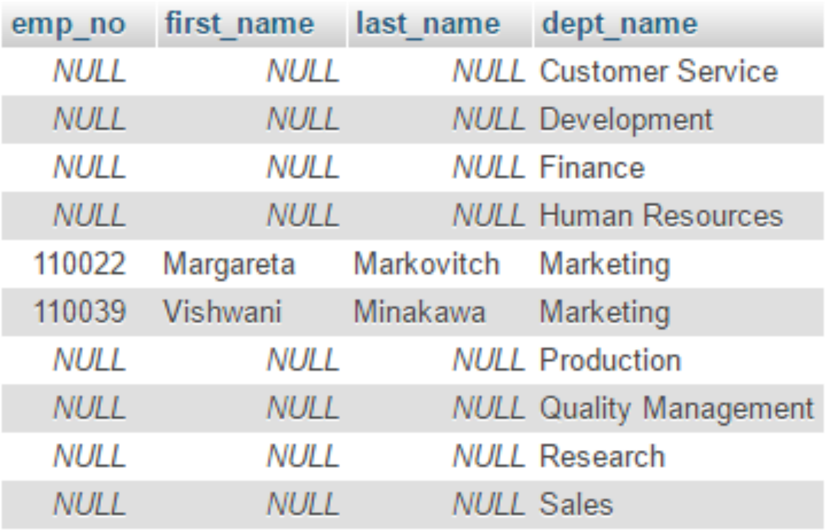


emp_no	first_name	last_name	dept_name
111692	Tonny	Butterworth	Customer Service
111784	Marjo	Giarratana	Customer Service
111877	Xiaobin	Spinelli	Customer Service
111939	Yuchang	Weedman	Customer Service
110511	DeForest	Hagimont	Development
110567	Leon	DasSarma	Development
110085	Ebru	Alpin	Finance
110114	Isamu	Legleitner	Finance
110183	Shirish	Ossenbruggen	Human Resources
110228	Karsten	Sigstam	Human Resources
110022	Margareta	Markovitch	Marketing
110039	Vishwani	Minakawa	Marketing

Left and Right Join Example

- Returns all the departments, including employee details for managers of the Marketing department

```
SELECT
  e.emp_no,
  e.first_name,
  e.last_name,
  de.dept_name
FROM dept_manager AS dm
  LEFT JOIN employees AS e
    ON dm.emp_no = e.emp_no
  RIGHT JOIN departments AS de
    ON dm.dept_no = de.dept_no
    and de.dept_name = 'Marketing';
```

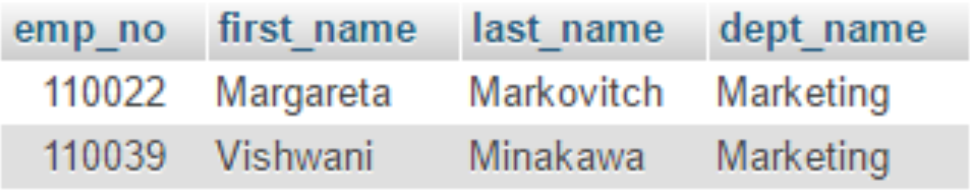


emp_no	first_name	last_name	dept_name
NULL	NULL	NULL	Customer Service
NULL	NULL	NULL	Development
NULL	NULL	NULL	Finance
NULL	NULL	NULL	Human Resources
110022	Margareta	Markovitch	Marketing
110039	Vishwani	Minakawa	Marketing
NULL	NULL	NULL	Production
NULL	NULL	NULL	Quality Management
NULL	NULL	NULL	Research
NULL	NULL	NULL	Sales

Another Join Example – inner join

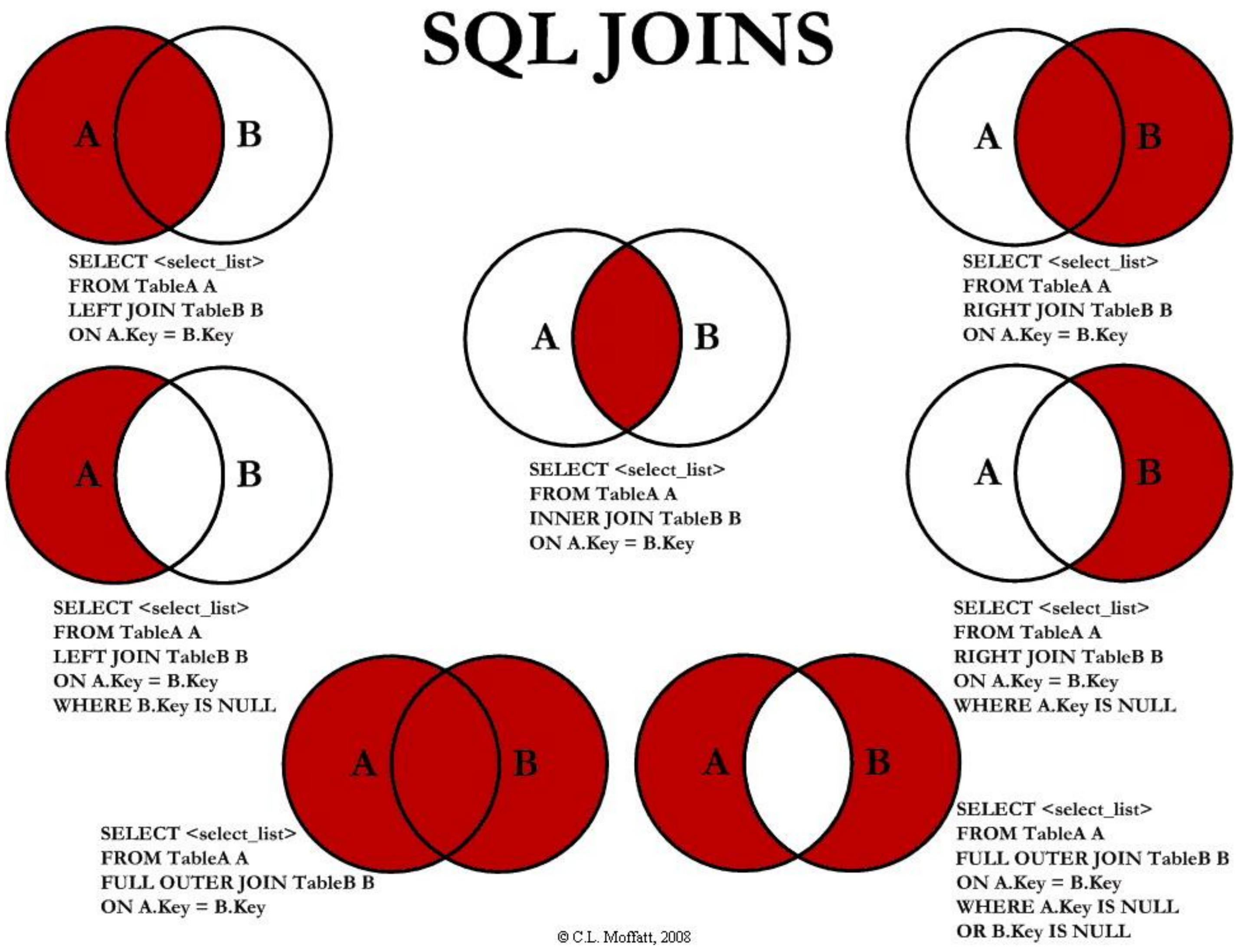
- Returns employee details only for managers of the Marketing department

```sql
SELECT
  e.emp_no,
  e.first_name,
  e.last_name,
  de.dept_name
FROM dept_manager AS dm
  LEFT JOIN employees AS e
    ON dm.emp_no = e.emp_no
  INNER JOIN departments AS de
    ON dm.dept_no = de.dept_no
    and de.dept_name = 'Marketing';
```



emp_no	first_name	last_name	dept_name
110022	Margareta	Markovitch	Marketing
110039	Vishwani	Minakawa	Marketing

Join Types



SQL JOINS

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

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

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

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

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

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Questions?