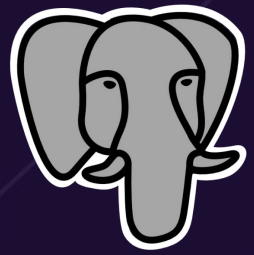
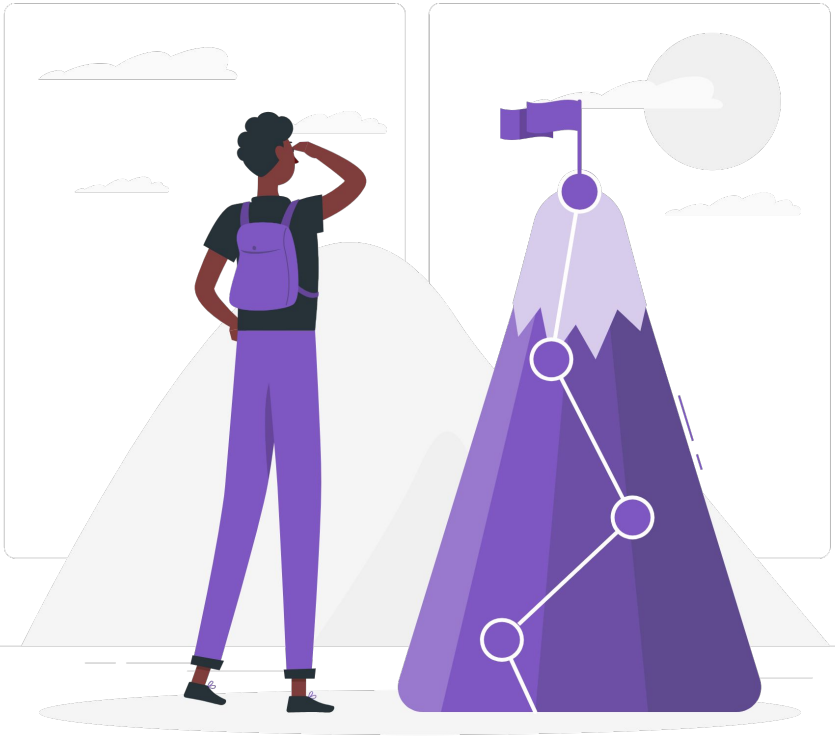


PostgreSQL

Uso de PostgreSQL Subqueries & Joins

DEV.F:
DESARROLLAMOS(PERSONAS);

Elaborado por: César Guerra



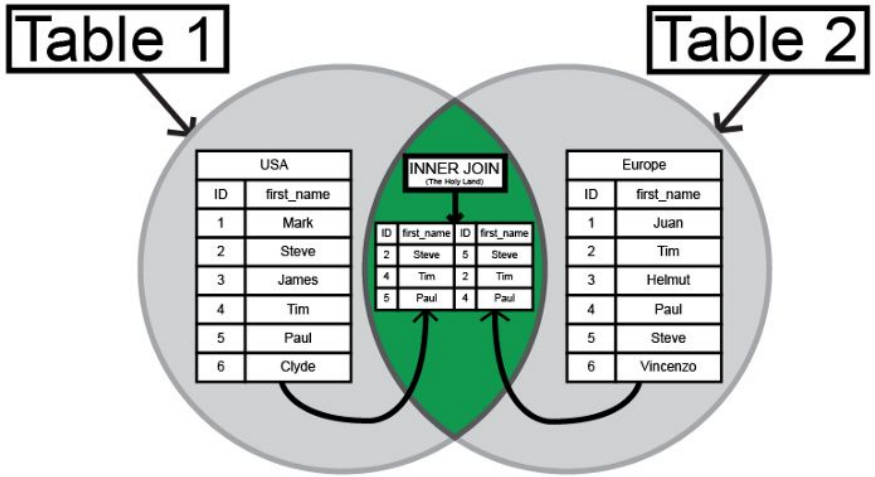
Objetivos de la Sesión

- Entender cómo hacer consultas entre dos o más tablas.
- Aprender a usar la sentencia JOIN para la creación de consultas con dos o más tablas.
- Aprender a realizar consultas anidadas (subqueries).

DEV.F::

Consultas a Base de Datos Con Join

DEV.F:
DESARROLLAMOS(PERSONAS);



JOIN

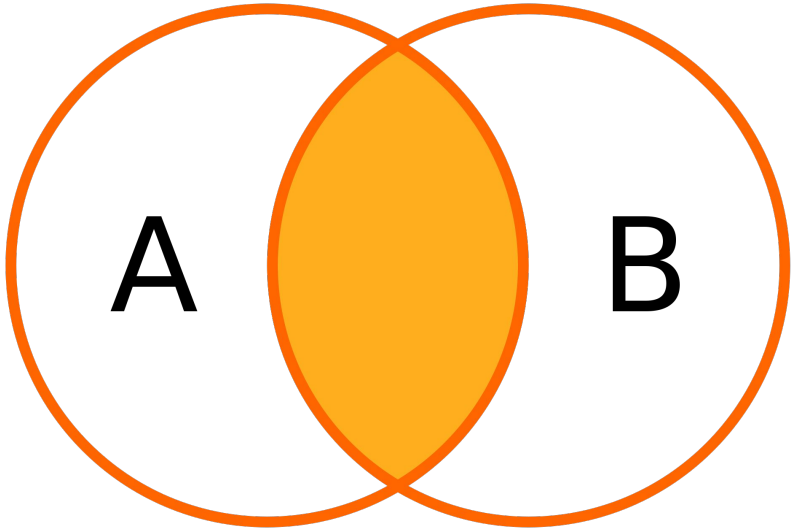
JOIN sirve para unir/cruzar la información de una o más tablas en una nueva tabla "temporal" de resultado.

Matemáticamente, JOIN es una composición relacional, la operación fundamental en el álgebra relacional.

Los tipos más comunes son:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

DEV.F::




```
SELECT fields
FROM TableA AS a
INNER JOIN TableB AS b
ON a.Key = b.Key;
WHERE condition(opcional)
```

INNER JOIN

Con esta operación cada registro en la tabla A es combinado con los correspondientes de la tabla B que satisfagan las condiciones que se especifiquen en el predicado del JOIN.

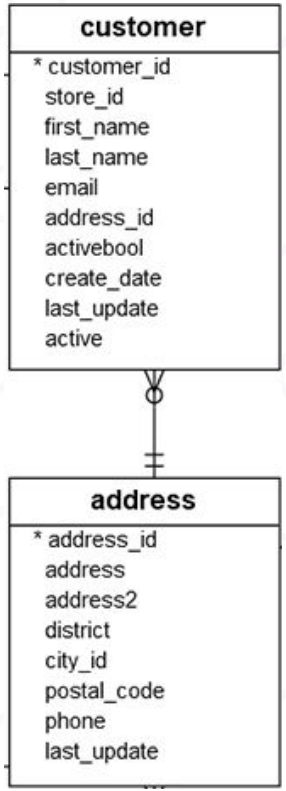
Para hacer un INNER JOIN, o generalmente abreviado como JOIN es necesario escoger tablas que TENGAN una relación.

Cualquier registro de la tabla A o de la tabla B que no tenga uno correspondiente en la otra tabla es excluido, y solo aparecerán los que tengan correspondencia en la otra tabla.

Este es el tipo de JOIN más utilizado, por lo que es considerado el tipo de combinación predeterminado.

DEV.F::

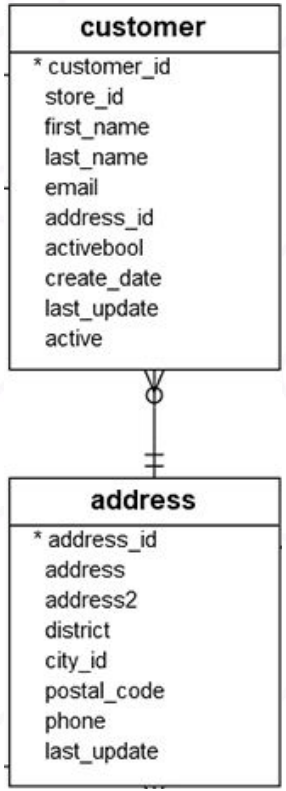
Ejemplo #1 Inner Join (en DVD Rentals)



Buscar dentro de **CUSTOMER** los que tengan **address_id** y los empata con la tabla **ADDRESS** que corresponda al mismo **address_id**

```
SELECT *
FROM customer
INNER JOIN address
ON customer.address_id = address.address_id;
```

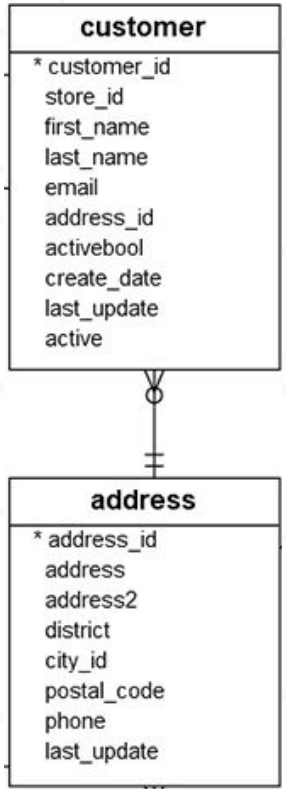
Ejemplo #2 Inner Join (en DVD Rentals)



Buscar dentro de **CUSTOMER** los que tengan **address_id** y los empata con la tabla **ADDRESS** que corresponda al mismo **address_id** pero solo si pertenecen a un determinado código postal.

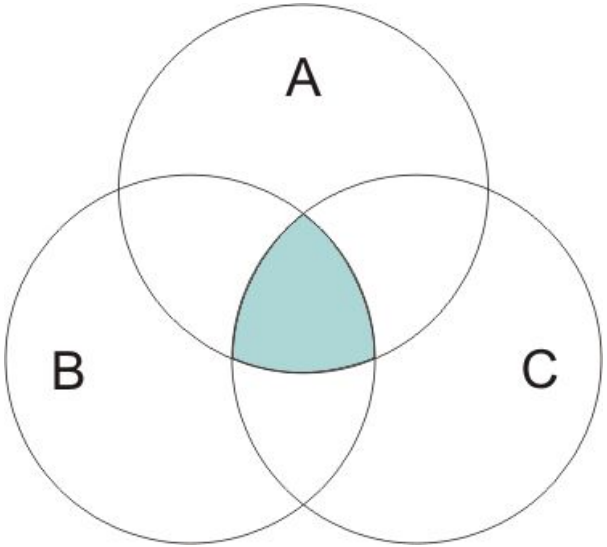
```
SELECT *
FROM customer
INNER JOIN address
ON customer.address_id = address.address_id
WHERE postal_code = '52137';
```

Ejemplo #3 Inner Join (en DVD Rentals)



Buscar dentro de todos los códigos postales de los clientes, contar cuantas veces se repite cada código postal y ordenarlos de mayor a menor. (Para saber dentro de que C.P. tengo más clientes)

```
SELECT postal_code, count(*)
FROM customer
INNER JOIN address
ON customer.address_id = address.address_id
GROUP BY postal_code
ORDER BY count(*) DESC
```




```
SELECT fields
FROM TableA AS a
INNER JOIN TableB AS b
ON a.Key = b.Key;
INNER JOIN TableC AS c
ON c.Key = b.Key;
WHERE condition(opcional)
```

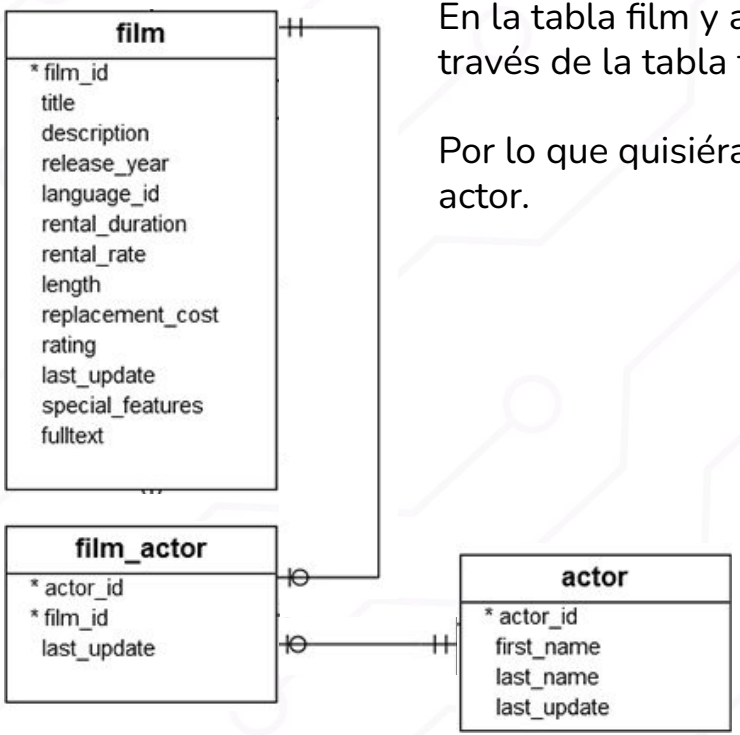
DOBLE INNER JOIN

INNER sirve para juntar tablas, lo más común es para juntar 2 tablas, pero es posible ejecutarlo en 3...4...5... tablas.

Aunque si se requiere hacer en un alto número de tablas puede indicar un mal diseño de la base de datos.

DEV.F::

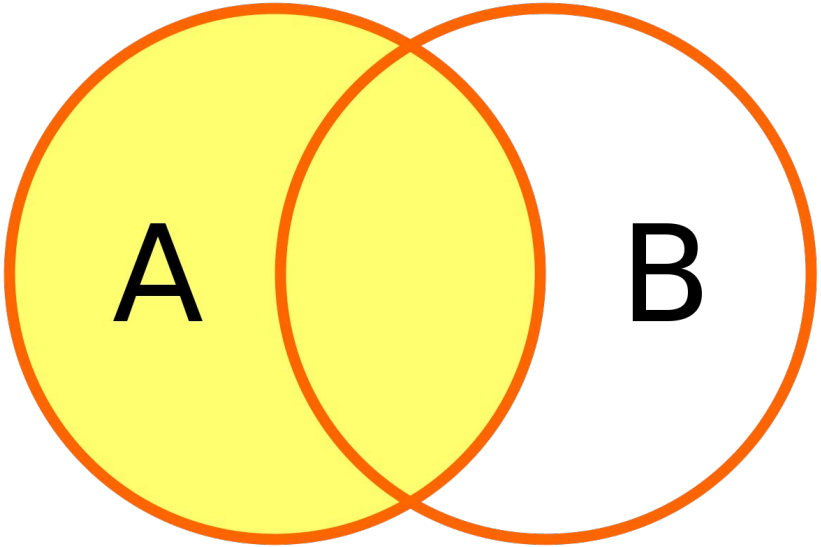
DOBLE INNER JOIN



En la tabla film y actor tienen un relación muchos a muchos a través de la tabla film_actor.

Por lo que quisiéramos saber, en que película ha trabajado qué actor.

```
SELECT title, first_name, last_name
FROM film AS F
INNER JOIN film_actor AS FA
ON F.film_id = FA.film_id
INNER JOIN actor AS AC
ON AC.actor_id = FA.actor_id
```




```
SELECT fields
FROM TableA AS a
LEFT JOIN TableB AS b
ON a.Key = b.Key;
WHERE condition(opcional)
```

LEFT JOIN

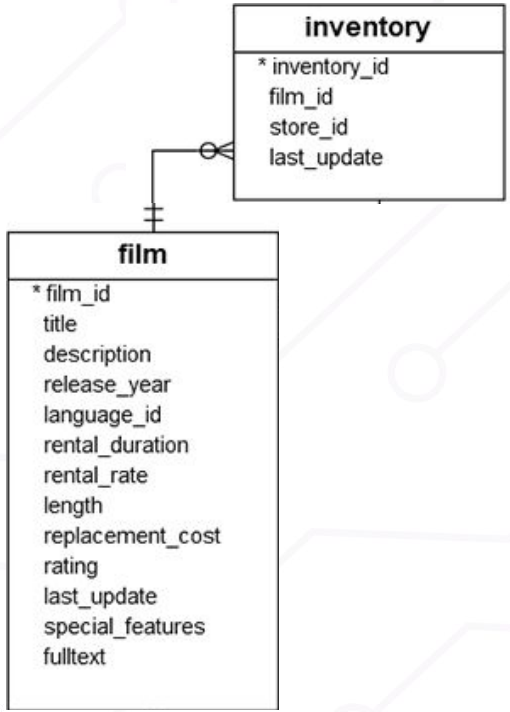
A diferencia de un **INNER JOIN**, donde se busca una intersección respetada por ambas tablas, con **LEFT JOIN** damos prioridad a la tabla de la izquierda, y buscamos en la tabla derecha.

La tabla en que usamos en el SELECT se considera la tabla de la IZQUIERDA (LEFT) y la que colocamos en LEFT JOIN se considera la de la DERECHA (RIGHT)

- La tabla LEFT (izquierda), y todas sus filas se mostrarán en los resultados.
- En la otra tabla (derecha) , si se encuentran coincidencias, se mostrarán los valores correspondientes, pero sino, aparecerá NULL en los resultados.

DEV.F::

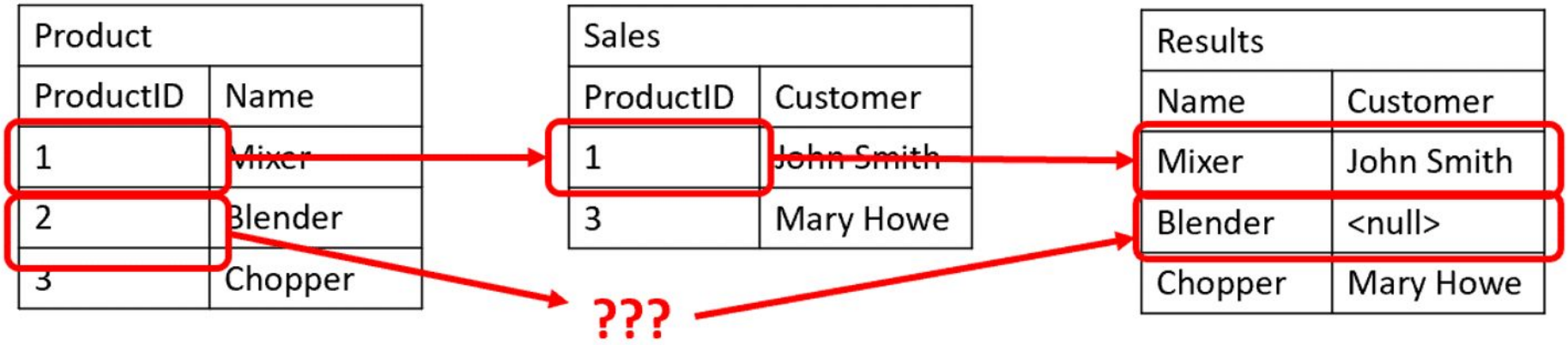
Ejemplo #1 Left Join (en DVD Rentals)



Traer las películas que NO estén en inventario

```
SELECT film.film_id, title, inventory_id, store_id
FROM film
LEFT JOIN inventory
ON film.film_id = inventory.film_id
WHERE inventory.film_id IS NULL
```

LEFT JOIN



Si existe un valor en la tabla IZQUIERDA pero no en la derecha, no puede existir una relación por lo que devolverá **null**

INNER JOIN VS LEFT JOIN

dvdrental/master@devf

Query Editor | Query History

```sql
-- Traer las peliculas que no están en inventario
SELECT film.film_id, title, inventory_id, store_id
FROM film
LEFT JOIN inventory
ON film.film_id = inventory.film_id
WHERE inventory.film_id IS NULL
```

Data Output | Explain | Messages | Notifications

	film_id integer	title character varying (255)	inventory_id integer	store_id smallint
1	14	Alice Fantasia	[null]	[null]
2	33	Apollo Teen	[null]	[null]
3	36	Argonauts Town	[null]	[null]
4	38	Ark Ridgemont	[null]	[null]
5	41	Arsenic Independence	[null]	[null]
6	87	Boondock Ballroom	[null]	[null]
7	108	Butch Panther	[null]	[null]

dvdrental/master@devf

Query Editor | Query History

```sql
-- Traer las peliculas que no están en inventario
SELECT film.film_id, title, inventory_id, store_id
FROM film
INNER JOIN inventory
ON film.film_id = inventory.film_id
WHERE inventory.film_id IS NULL
```

Data Output | Explain | Messages | Notifications

film_id integer	title character varying (255)	inventory_id integer	store_id smallint

Si intento ejecutar lo anterior solamente cambiando el **LEFT JOIN** por **INNER JOIN**, obtendré un resultado vacío, ya que al no existir el **film_id** (nulo) no lo puede relacionar.

DEV.F

Consultas a Base de Datos con Subqueries

DEV.F:
DESARROLLAMOS(PERSONAS);

```
SELECT campos
FROM Tabla
WHERE condición (SELECT
campos FROM Tabla)
```

SUB QUERIES

A diferencia de los JOINS, este no muestra directamente datos de una segunda tabla.

Simplemente crea una tabla intermedia temporal basada en el resultado de una consulta para obtener un resultado que se utilizará en la consulta y tabla original.

Para usarlo, realizamos otra consulta dentro de la condición qué forma parte del WHERE

DEV.F::

Ejemplo #1 Sub-queries (en DVD Rentals)

film
* film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
fulltext

Traer el promedio de renta de todas las películas

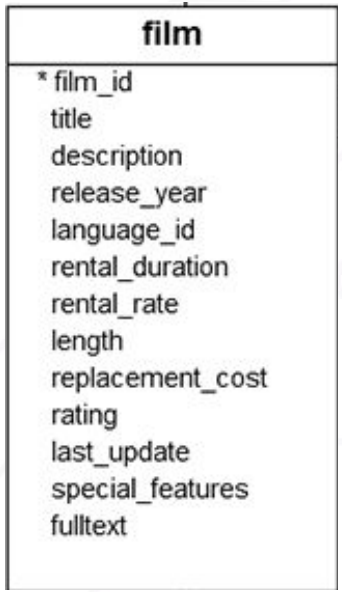
```
SELECT AVG (rental_rate)
FROM film;
```

Data Output | Explain

	avg numeric
1	2.9800000000000000

Esta primera aproximación no nos cuenta nada realmente interesante.

Ejemplo #1 Sub-queries (en DVD Rentals)



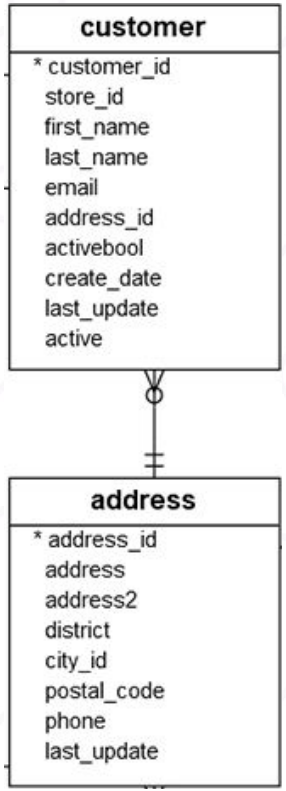
Traer las películas que menos se rentan y que estén por debajo del promedio:

```
SELECT title, rental_rate
FROM film
WHERE rental_rate < (SELECT AVG
(rental_rate) FROM film)
ORDER BY rental_rate
DESC;
```

Data Output | Explain | Messages | Notifications

	title character varying (255)	rental_rate numeric (4,2)
1	Academy Dinosaur	0.99
2	Alamo Videotape	0.99
3	Alaska Phantom	0.99
4	Date Speed	0.99
5	Alice Fantasia	0.99

Ejemplo #2 Sub-queries (en DVD Rentals)



Traer todos los customers que vivan en el código postal 52137, ya que no conocemos su address_id.

```
SELECT *
FROM customer
WHERE address_id
IN (SELECT address_id FROM address
WHERE postal_code = '52137');
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	store_id smallint	first_name character vary	last_name character vary	email characte	address_id smallint	activebo boolean	create_date date	last_update timestamp without time
1	299	2	James	Gannon	james....	304	true	2006-02-14	2013-05-26 14:49:45.73
2	597	1	Freddie	Duggan	freddi...	603	true	2006-02-14	2013-05-26 14:49:45.73