**Universidad de san buenaventura**

**Mario Juan Sebastian Reyes Casas**

**Bitácora SQL**

**Luis Barreto**

**Bogotá D.C**

**05/03/2020**

**Índice:**

[**Select**](#select)

[**Select DIstinct**](#select_distintc)

[**Where**](#where)

[**And or not**](#where_and_or_not)

[**Order by**](#order_by)

[**Insert into**](#insert)

[**Null values**](#null)

[**Update**](#update)

[**Delete**](#delete)

[**Select Top**](#limit)

[**Min and Max**](#minMax)

[**Count ,avg,sum**](#count)

[**Like**](#like)

[**In**](#in)

[**Between**](#between)

[**Aliases**](#alias)

**[Joins](#union)**

[**Inner join**](#join)

[**Left join**](#left)

[**Right join**](#right)

[**Full join**](#full)

[**Self join**](#self)

[**Unión**](#union)

[**Group by**](#group)

[**Having**](#having)

[**Exists**](#exist)

[**Any all**](#any)

**[Insert into select](#select_into)**

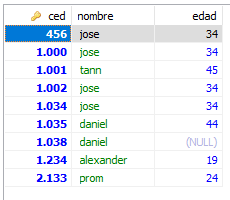
[**Case**](#case)

[**Null Functions**](#ifnull)

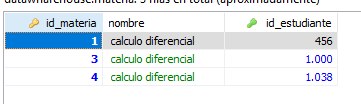
[**Procedures**](#procedures)

**Base de datos utilizada**

La base de datos utilizada es una de estudiantes la podemos observar a continuación



Y luego tenemos la tabla de materias



**SELECT**

select \* from datawharehouse.estudiantes ;

+--------+----------+------+

| ced | nombre | edad |

+--------+----------+------+

| 456 | jordan | 34 |

| 1000 | sec | 34 |

| 1001 | tann | 45 |

| 1002 | gonazalo | 34 |

| 2133 | prom | 24 |

| 4564 | mintu | 45 |

| 349055 | sdfnksad | 23 |

+--------+----------+------+

7 rows in set (0.00 sec)

Selecciona todos los registros de la base de datos

**Select distinct**

select distinct edad from datawharehouse.estudiantes;

+------+

| edad |

+------+

| 34 |

| 45 |

| 24 |

| 23 |

+------+

4 rows in set (0.02 sec)

nos devuelve la edad sin repetición .

**Select count**

SELECT COUNT(DISTINCT edad) FROM datawharehouse.estudiantes;

+----------------------+

| COUNT(DISTINCT edad) |

+----------------------+

| 4 |

+----------------------+

1 row in set (0.17 sec)

El query nos devuelve la suma de la edad pero sin que esta se repita

SELECT count(edad) FROM datawharehouse.estudiantes;

+-------------+

| count(edad) |

+-------------+

| 7 |

+-------------+

1 row in set (0.00 sec)

Acá podemos observar que pasa si no aplicamos la palabra distinct .

**Where**

select edad ,nombre from datawharehouse.estudiantes where edad=34 or edad=23;

+------+----------+

| edad | nombre |

+------+----------+

| 34 | jordan |

| 34 | sec |

| 34 | gonazalo |

| 23 | sdfnksad |

+------+----------+

4 rows in set (0.00 sec)

Este query nos devuelve la edad y los nombres con la clausula where y el operador or , y toma un rango de edades comprendidas entre los 34 y 23 años

**Where Not-And-or**

select nombre from datawharehouse.estudiantes where not nombre ='prom';

+----------+

| nombre |

+----------+

| jordan |

| sec |

| tann |

| gonazalo |

| mintu |

| sdfnksad |

+----------+

6 rows in set (0.00 sec)

Así como podemos consultar un registro también podemos consultar donde no hay un registro ,por ejemplo en esta tabla buscamos con la condición donde el nombre no sea ‘prom’ o no empiece con ‘prom’

select nombre from datawharehouse.estudiantes where not nombre ='prom' and not nombre='sec';

+----------+

| nombre |

+----------+

| jordan |

| tann |

| gonazalo |

| mintu |

| sdfnksad |

+----------+

5 rows in set (0.00 sec)6 rows in set (0.00 sec)

Lo mismo aplica en este ejemplo ,aquí simplemente excluimos los valores “prom “ y “sec”

Nótese como cuando se usa el or una de las dos condiciones puede cumplirse mientras que con el and deben ser ambas condiciones

**Order by**

select \* from datawharehouse.estudiantes order by edad asc , nombre desc ;

+--------+-----------+------+

| ced | nombre | edad |

+--------+-----------+------+

| 1234 | alexander | 19 |

| 349055 | sdfnksad | 23 |

| 2133 | prom | 24 |

| 1034 | sezilia | 34 |

| 1000 | sec | 34 |

| 456 | jordan | 34 |

| 1002 | gonazalo | 34 |

| 1001 | tann | 45 |

| 4564 | mintu | 45 |

+--------+-----------+------+

Este query devuelve todos los datos ordenando la edad ascendentemente y ordenando los nombres descendentemente ,notese que lo primero en cumplirse es la edad ascendentemente y luego si hay dos valores iguales usa los nombres descendientemente par organizarlo

**INSERT**

insert into datawharehouse.estudiantes values (1034 , "sezilia", 34);

Query OK, 1 row affected (0.15 sec)

+--------+-----------+------+

| ced | nombre | edad |

+--------+-----------+------+

| 1234 | alexander | 19 |

| 349055 | sdfnksad | 23 |

| 2133 | prom | 24 |

| 1034 | sezilia | 34 |

| 1000 | sec | 34 |

| 456 | jordan | 34 |

| 1002 | gonazalo | 34 |

| 1001 | tann | 45 |

| 4564 | mintu | 45 |

+--------+-----------+------+

El siguiente query nos inserta los tres valores de nuestra database simplemente se hace sin especificar los nombres de las columnas

insert into datawharehouse.estudiantes (ced,nombre,edad) values (1035 , "daniel", 44);

Query OK, 1 row affected (0.15 sec)

+--------+-----------+------+

| ced | nombre | edad |

+--------+-----------+------+

| 1234 | alexander | 19 |

| 349055 | sdfnksad | 23 |

| 2133 | prom | 24 |

| 1034 | sezilia | 34 |

| 1000 | sec | 34 |

| 456 | jordan | 34 |

| 1002 | gonazalo | 34 |

| 1035 | daniel | 44 |

| 1001 | tann | 45 |

| 4564 | mintu | 45 |

+--------+-----------+------+

Sin embargo se pueden especificar los nombres de las columnas en las que se depositaran los datos

insert into datawharehouse.estudiantes (ced,nombre) values (1035 , "daniel");

Query OK, 1 row affected (0.15 sec)

+--------+-----------+------+

| ced | nombre | edad |

+--------+-----------+------+

| 1038 | daniel | NULL |

| 1234 | alexander | 19 |

| 349055 | sdfnksad | 23 |

| 2133 | prom | 24 |

| 1034 | sezilia | 34 |

| 1000 | sec | 34 |

| 456 | jordan | 34 |

| 1002 | gonazalo | 34 |

| 1035 | daniel | 44 |

| 1001 | tann | 45 |

| 4564 | mintu | 45 |

+--------+-----------+------+

11 rows in set (0.00 sec)

También se aceptan valores nulos dentro de un insert ,siempre y cuando no haya una llave primaria

**NULL**

select nombre from datawharehouse.estudiantes where edad is null;

+--------+

| nombre |

+--------+

| daniel |

+--------+

1 row in set (0.00 sec)

Para los select también es posible consultar que valores están nulos por ejemplo el nombre de la persona que no tiene edad

**UPDATE**

update datawharehouse.estudiantes set nombre= 'jose ' where edad=34 ;

+--------+-----------+------+

| ced | nombre | edad |

+--------+-----------+------+

| 456 | jose | 34 |

| 1000 | jose | 34 |

| 1001 | tann | 45 |

| 1002 | jose | 34 |

| 1034 | jose | 34 |

| 1035 | daniel | 44 |

| 1234 | alexander | 19 |

| 2133 | prom | 24 |

| 4564 | mintu | 45 |

| 349055 | sdfnksad | 23 |

+--------+-----------+------+

10 rows in set (0.00 sec))

la sentencia update nos permite actualizar el nombre de un estudiante donde la edad sea de 34

**DELETE**

delete from datawharehouse.estudiantes where nombre = "sdfnksad";

Query OK, 1 row affected (0.14 sec)

En el siguiente query podremos eliminar con delete una fila ,no la tabla sino una fila nada mas podemos usar el where con sus respectivos operadores and ,not ,or

**LIMIT**

select \* from datawharehouse.estudiantes where edad = 34 limit 2 ;

+------+--------+------+

| ced | nombre | edad |

+------+--------+------+

| 456 | jose | 34 |

| 1000 | jose | 34 |

+------+--------+------+

2 rows in set (0.00 sec)

No hay que olvidar que podemos filtrar las consultas con un limite de consultas en el ejemplo que se presenta los resultados se limitaron a solo dos registros

select \* from estudiantes limit 3;

+------+--------+------+

| ced | nombre | edad |

+------+--------+------+

| 456 | jose | 34 |

| 1000 | jose | 34 |

| 1001 | tann | 45 |

+------+--------+------+

3 rows in set (0.00 sec)

Acá se limitan a solo 3 registros

**MIN y MAX**

select min(edad) from datawharehouse.estudiantes ;

+-----------+

| min(edad) |

+-----------+

| 19 |

+-----------+

1 row in set (0.00 sec)

select max(edad) from datawharehouse.estudiantes ;

+-----------+

| max(edad) |

+-----------+

| 45 |

+-----------+

1 row in set (0.00 sec))

Min devuelve el menor valor de los registros ,y max el mayor ,el parámetro filtrado por max y min es edad

**COUNT AVG Y SUM**

select count(nombre)from datawharehouse.estudiantes where edad is not null ;

+---------------+

| count(nombre) |

+---------------+

| 8 |

+---------------+

1 row in set (0.00 sec)

Count nos devuelve el valor de la cantidad de nombres que hay en la tabla donde la condición es que no haya edad vacía , nos cuenta todos los registros y nos muestra 8 resultados

select avg(edad)from datawharehouse.estudiantes ;

+-----------+

| avg(edad) |

+-----------+

| 33.5000 |

+-----------+

1 row in set (0.00 sec)

AVG nos devuelve el promedio ,en este caso nos devuelve el promedio de las edades

select avg(ced)from datawharehouse.estudiantes ;

+-----------+

| avg(ced) |

+-----------+

| 1103.6667 |

+-----------+

1 row in set (0.00 sec)

En este caso el promedio le corresponde a las cedulas

**LIKE**

select nombre from datawharehouse.estudiantes where nombre like 'a%' ;

+-----------+

| nombre |

+-----------+

| alexander |

+-----------+

1 row in set (0.00 sec)

Acá lo que nos devolvieron fue el nombre que empezara con *“a”*

select nombre from datawharehouse.estudiantes where nombre like '%a%' ;

+-----------+

| nombre |

+-----------+

| tann |

| daniel |

| daniel |

| alexander |

+-----------+

4 rows in set (0.00 sec)

Devuelve todos los nombres que contengan una “*a”*  en cualquier posición

select nombre from datawharehouse.estudiantes where nombre like 'a\_\_\_%' ;

+-----------+

| nombre |

+-----------+

| alexander |

+-----------+

1 row in set (0.00 sec)

Devuelve los nombres que empiezan con a y tienen al menos 3 espacios

select nombre from datawharehouse.estudiantes where nombre like 'a%o' ;

Empty set (0.00 sec)

Devuelve los nombres donde se empiece con “*a”* y termine en *“o”,*  no hay ninguno con esas características

**SELECT** \* **FROM** datawharehouse.estudiantes **WHERE** nombre **LIKE** '\_a%';

Empty set (0.00 sec)

+------+--------+------+

| ced | nombre | edad |

+------+--------+------+

| 1001 | tann | 45 |

| 1035 | daniel | 44 |

| 1038 | daniel | NULL |

+------+--------+------+

3 rows in set (0.00 sec)

Este selecciona los registros los cuales la *“a”*  este en la segunda posición

**IN**

**select** \* **from** datawharehouse.estudiantes **where** edad **in** (34,50,23,19);

Empty set (0.00 sec)

+------+-----------+------+

| ced | nombre | edad |

+------+-----------+------+

| 456 | jose | 34 |

| 1000 | jose | 34 |

| 1002 | jose | 34 |

| 1034 | jose | 34 |

| 1234 | alexander | 19 |

+------+-----------+------+

5 rows in set (0.00 sec)

Selecciona los registros en los que la edad se encuentren entre los valores 34,50,23,19

**select \* from estudiantes where edad in (select edad where edad >18);**

**+------+-----------+------+**

**| ced | nombre | edad |**

**+------+-----------+------+**

**| 456 | jose | 34 |**

**| 1000 | jose | 34 |**

**| 1001 | tann | 45 |**

**| 1002 | jose | 34 |**

**| 1034 | jose | 34 |**

**| 1035 | daniel | 44 |**

**| 1234 | alexander | 19 |**

**| 2133 | prom | 24 |**

**+------+-----------+------+**

**8 rows in set (0.00 sec)**

También se pueden realizar sub consultar dentro de las consultas en este caso consultamos donde la edad sea mayor a 18

**BETWEEN**

**select nombre from datawharehouse.estudiantes where edad BETWEEN 19 and 39;**

**+-----------+**

**| nombre |**

**+-----------+**

**| jose |**

**| jose |**

**| jose |**

**| jose |**

**| alexander |**

**| prom |**

**+-----------+**

**6 rows in set (0.00 sec)**

Este query nos permite obtener valores que oscilen entre rangos ,por ejemplo la edad que se encuentre entre los 19 y 39 años

**select nombre from datawharehouse.estudiantes where edad BETWEEN 19 and 39 and nombre BETWEEN 'a' and 'z' ;**

**+-----------+**

**| nombre |**

**+-----------+**

**| jose |**

**| jose |**

**| jose |**

**| jose |**

**| alexander |**

**| prom |**

**+-----------+**

**6 rows in set (0.00 sec)**

No obstante también se pueden especificar letras ,en este caso que los nombres estén entre la a y la z

**ALIAS**

**select ced as c ,nombre as nom , edad as ed from datawharehouse.estudiantes;**

**+------+-----------+------+**

**| c | nom | ed |**

**+------+-----------+------+**

**| 456 | jose | 34 |**

**| 1000 | jose | 34 |**

**| 1001 | tann | 45 |**

**| 1002 | jose | 34 |**

**| 1034 | jose | 34 |**

**| 1035 | daniel | 44 |**

**| 1038 | daniel | NULL |**

**| 1234 | alexander | 19 |**

**| 2133 | prom | 24 |**

**+------+-----------+------+**

**9 rows in set (0.00 sec)**

Acá establecemos los alias para las filas ced como c , nombre como nom y edad como ed

**INNER JOIN**

select materia.id\_materia, estudiantes.\* from materia INNER join estudiantes on materia.id\_estudiante = estudiantes.ced ;

+------------+------+--------+------+

| id\_materia | ced | nombre | edad |

+------------+------+--------+------+

| 1 | 456 | jose | 34 |

| 3 | 1000 | jose | 34 |

| 4 | 1038 | daniel | NULL |

+------------+------+--------+------+

3 rows in set (0.00 sec)

nos devuelve la intersección de ambas tablas ,donde el id del estudiante sea igual al id de estudiante en la tabla materia ,podemos seleccionar todos los datos de estudiante

**LEFT JOIN**

select materia.id\_materia, estudiantes.\* from materia left join estudiantes on materia.id\_estudiante = estudiantes.ced ;

+------------+------+--------+------+

| id\_materia | ced | nombre | edad |

+------------+------+--------+------+

| 1 | 456 | jose | 34 |

| 3 | 1000 | jose | 34 |

| 4 | 1038 | daniel | NULL |

+------------+------+--------+------+

3 rows in set (0.00 sec)

Devuelve los resultados de la izquierda (solo id,acá pueden traerse todos los resultados y aparecerán en null las materias que no tienen relación con estudiante ) y todos los de la derecha(todos los datos) únicamente cuando hacen match los id’s ,los estudiantes sin relación los devuelve en null

select materia.\*, estudiantes.\* from materia left join estudiantes on materia.id\_estudiante = estudiantes.ced ;

+------------+----------------------+---------------+------+--------+------+

| id\_materia | nombre | id\_estudiante | ced | nombre | edad |

+------------+----------------------+---------------+------+--------+------+

| 1 | calculo diferencial | 456 | 456 | jose | 34 |

| 3 | calculo diferencial | 1000 | 1000 | jose | 34 |

| 4 | calculo diferencial | 1038 | 1038 | daniel | NULL |

+------------+----------------------+---------------+------+--------+------+

**RIGHT JOIN**

select materia.\*, estudiantes.ced , estudiantes.nombre from materia RIGHT join estudiantes on materia.id\_estudiante= estudiantes.ced;

+------------+----------------------+---------------+------+-----------+

| id\_materia | nombre | id\_estudiante | ced | nombre |

+------------+----------------------+---------------+------+-----------+

| 1 | calculo diferencial | 456 | 456 | jose |

| 3 | calculo diferencial | 1000 | 1000 | jose |

| 4 | calculo diferencial | 1038 | 1038 | daniel |

| NULL | NULL | NULL | 1001 | tann |

| NULL | NULL | NULL | 1002 | jose |

| NULL | NULL | NULL | 1034 | jose |

| NULL | NULL | NULL | 1035 | daniel |

| NULL | NULL | NULL | 1234 | alexander |

| NULL | NULL | NULL | 2133 | prom |

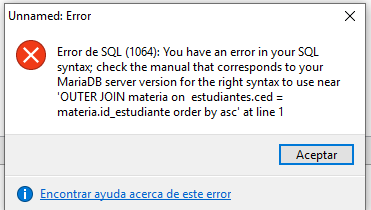
+------------+----------------------+---------------+------+-----------+

9 rows in set (0.00 sec)

Devuelve todos los valores de ambas tablas pero marcara en null los que no tienen relación estudiantes con materias, las materias sin relación no las mostrara

**select** estudiantes.ced , materia.id\_materia **from** estudiantes **FULL** **OUTER** **JOIN** materia **on** estudiantes.ced = materia.id\_estudiante **order** **by** **asc** ;

Este se supone que hace un full join , no obstante persiste enviando mensajes de error



Un self join es una consulta normal a ella misma .

**UNION**

select \* from datawharehouse.estudiantes union select \* from datawharehouse.materia ;

+------+----------------------+------+

| ced | nombre | edad |

+------+----------------------+------+

| 456 | jose | 34 |

| 1000 | jose | 34 |

| 1001 | tann | 45 |

| 1002 | jose | 34 |

| 1034 | jose | 34 |

| 1035 | daniel | 44 |

| 1038 | daniel | NULL |

| 1234 | alexander | 19 |

| 2133 | prom | 24 |

| 1 | calculo diferencial | 456 |

| 3 | calculo diferencial | 1000 |

| 4 | calculo diferencial | 1038 |

+------+----------------------+------+

12 rows in set (0.00 sec)

El unión nos permite juntar dos columnas en este caso tengo dos consultas de todos pero de diferente tabla y esto lo devuelve en un solo query simplificando las ejecuciones

**GROUP BY**

select count(edad) , nombre from datawharehouse.estudiantes group by nombre desc;

+-------------+-----------+

| count(edad) | nombre |

+-------------+-----------+

| 1 | tann |

| 1 | prom |

| 4 | jose |

| 1 | daniel |

| 1 | alexander |

+-------------+-----------+

5 rows in set (0.00 sec)

Con el siguiente query podemos agrupar los resultados según la edad por ejemplo ,se puede también por ciudad , en este caso agrupamos las edades de las personas y sus nombres ,es decir nos dice cuantos valores tiene ese nombre

**select** **count**(materia.id\_estudiante), materia.nombre ,estudiantes.nombre **from** datawharehouse.materia **inner** **join** estudiantes **on** materia.id\_estudiante = estudiantes.ced **group** **by** materia.nombre **asc** ;

+------------------------------+----------------------+---------+

| count(materia.id\_estudiante) | nombre | nombre |

+------------------------------+----------------------+---------+

| 1 | calculo diferencial | rodrigo |

| 1 | calculo vectorial | jose |

| 1 | metodos numericos | daniel |

+------------------------------+----------------------+---------+

3 rows in set (0.00 sec)

Este corto query lo que devuelve es la cantidad de materias que hay el nombre de esas materias ,el nombre de los estudiantes luego se realiza un inner join para ver cuales son las que están relacionadas de id ,tanto como materia como estudiante y las organiza por el nombre de la materia ascendentemente ,cabe aclarar que si hubieran dos personas en alguna materia solo contaría la ultima en los resultados y no mostraría a ambos

**HAVING**

**select nombre from estudiantes where edad >16 having nombre = "jose" ;**

**+--------+**

**| nombre |**

**+--------+**

**| jose |**

**| jose |**

**+--------+**

**2 rows in set (0.00 sec)**

}

El having nos ayuda a especificar otra condicion , por ejemplo aquí solo se escogen los estudiantes donde su edad sea superior a 16 pero también donde su nombre sea jose

**EXISTS**

**select** nombre **from** estudiantes **where** **EXISTS** (**select** \* **from** materia **where** materia.nombre **is** **not** **null** );

**+-----------+**

**| nombre |**

**+-----------+**

**| rodrigo |**

**| jose |**

**| tann |**

**| anuel |**

**| jose |**

**| daniel |**

**| daniel |**

**| alexander |**

**| prom |**

**+-----------+**

**9 rows in set (0.00 sec)**

Acá garantizamos que si existe algo dentro de otra tabla podemos mostrar un nombre ,es como hacer una consulta indirectamente para obtener otra es decir una concatenación

**ANY ALL**

**select** **count**(nombre)**from** datawharehouse.materia **where** materia.id\_materia **is** **not** **null** = **all** ( **select** **count**(nombre)**from** datawharehouse.estudiantes **where** estudiantes.edad =34 );

**+---------------+**

**| count(nombre) |**

**+---------------+**

**| 0 |**

**+---------------+**

**1 row in set (0.00 sec)**

Con all es mas restrictiva la busqueda sin embargo con any se puede seleccionar cualquiera de los valores que se le den

**SELECT INTO**

**insert into estudentbackup select \* from estudiantes;**

**Query OK, 9 rows affected (0.21 sec)**

**Records: 9 Duplicates: 0 Warnings: 0**

Este comando nos permite insertar con un select de una tabla , pero primero se debe crear la tabla con los campos que tiene la tabla vieja sino no funcionara el resultado fueron los 9 registros copiados de un lado a otro

**select \* from estudentbackup ;**

**+------+-----------+------+**

**| ced | nombre | edad |**

**+------+-----------+------+**

**| 456 | rodrigo | 34 |**

**| 1000 | jose | 34 |**

**| 1001 | tann | 45 |**

**| 1002 | anuel | 34 |**

**| 1034 | jose | 34 |**

**| 1035 | daniel | 44 |**

**| 1038 | daniel | NULL |**

**| 1234 | alexander | 19 |**

**| 2133 | prom | 24 |**

**+------+-----------+------+**

**9 rows in set (0.00 sec)**

**CASE**

**SELECT** estudiantes.edad,

**case**

**when** estudiantes.edad > 39 **then** "es mayor"

**ELSE** "es menor que 39"

**END**

**FROM** estudiantes;

**+------+-------------------------------------------------------------------------------+**

**| 34 | es menor que 39 |**

**| 34 | es menor que 39 |**

**| 45 | es mayor |**

**| 34 | es menor que 39 |**

**| 34 | es menor que 39 |**

**| 44 | es mayor |**

**| NULL | es menor que 39 |**

**| 19 | es menor que 39 |**

**| 24 | es menor que 39 |**

**+------+-------------------------------------------------------------------------------+**

**9 rows in set (0.00 sec)**

En el case podemos hacer un intento de if en donde si una condición se cumple entonces se realiza una acción en este caso hemos hecho una simple comparación

**IFNULL**

**select** nombre\*(ced+**IFNULL**(edad,0)) **from** estudiantes;

+-----------------------------+

| nombre\*(ced+IFNULL(edad,0)) |

+-----------------------------+

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

+-----------------------------+

9 rows in set, 9 warnings (0.00 sec)

No permite que se devuelvan resultados null

**PROCEDURES**

**CREATE** **PROCEDURE** `consultarTodo`()

**LANGUAGE** **SQL**

**NOT** **DETERMINISTIC**

**CONTAINS** **SQL**

**SQL** **SECURITY** **DEFINER**

**COMMENT** ''

**BEGIN**

**select** \* **from** estudiantes;

**END**

CALL `consultarTodo`();

+------+-----------+------+

| ced | nombre | edad |

+------+-----------+------+

| 456 | rodrigo | 34 |

| 1000 | jose | 34 |

| 1001 | tann | 45 |

| 1002 | anuel | 34 |

| 1034 | jose | 34 |

| 1035 | daniel | 44 |

| 1038 | daniel | NULL |

| 1234 | alexander | 19 |

| 2133 | prom | 24 |

+------+-----------+------+

9 rows in set (0.00 sec)

Query OK, 0 rows affected (0.13 sec)

Heidysql crea el procedimiento de la manera en la que observamos arriba ,en este caso solo hemos consultado