

Universidad de San Carlos de Guatemala
Ingeniería en Ciencias Y Sistemas
Sistemas de Bases de datos 2

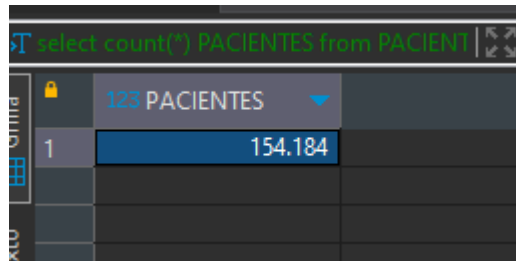
Práctica #2

José Abraham Solórzano Herrera	201800937
Francisco Magdiel Asicono Mateo	201801449
Keila Avril Vilchez Suarez	201700569

Guatemala, 25 de Marzo de 2023

- **CARGA DE DATOS
MYSQL**

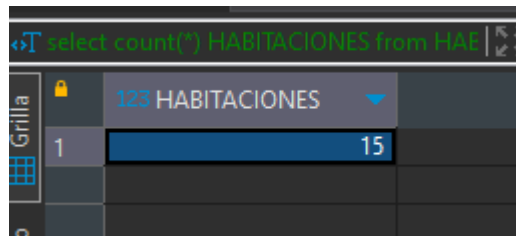
→ PACIENTES



A screenshot of a MySQL query window showing the command `select count(*) PACIENTES from PACIENTE`. The results table has a dropdown menu set to '123 PACIENTES'. The first row shows the count as 154,184.

	123 PACIENTES
1	154.184

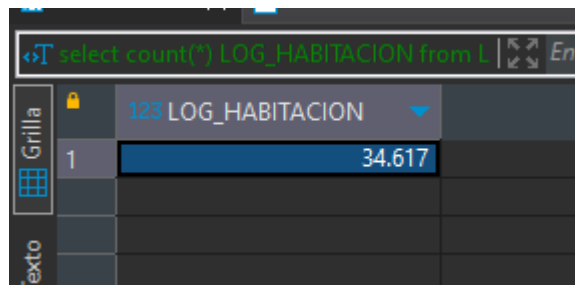
→ HABITACIONES



A screenshot of a MySQL query window showing the command `select count(*) HABITACIONES from HAE`. The results table has a dropdown menu set to '123 HABITACIONES'. The first row shows the count as 15.

	123 HABITACIONES
1	15

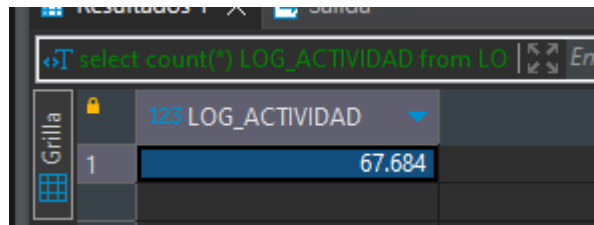
→ LOG_HABITACION



A screenshot of a MySQL query window showing the command `select count(*) LOG_HABITACION from L`. The results table has a dropdown menu set to '123 LOG_HABITACION'. The first row shows the count as 34,617.

	123 LOG_HABITACION
1	34.617

→ LOG_ACTIVIDAD

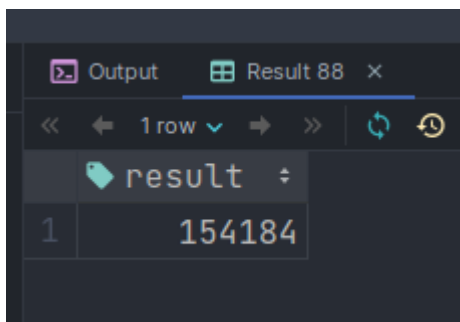


A screenshot of a MySQL query window showing the command `select count(*) LOG_ACTIVIDAD from LO`. The results table has a dropdown menu set to '123 LOG_ACTIVIDAD'. The first row shows the count as 67,684.

	123 LOG_ACTIVIDAD
1	67.684

MONGODB

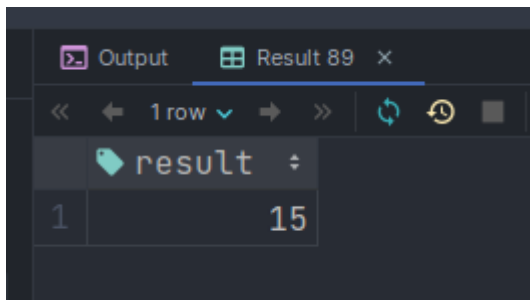
→ PACIENTES



A screenshot of a MongoDB query window showing the command `select count(*) LOG_ACTIVIDAD from LO`. The results table has a dropdown menu set to '123 LOG_ACTIVIDAD'. The first row shows the count as 67,684.

	123 LOG_ACTIVIDAD
1	67.684

→ HABITACIONES

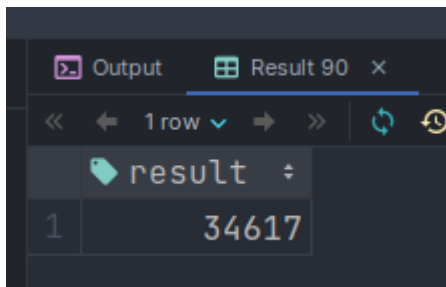


Output Result 89

1 row

1	15
---	----

→ LOG_HABITACION

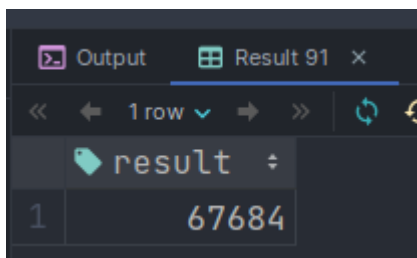


Output Result 90

1 row

1	34617
---	-------

→ LOG_ACTIVIDAD



Output Result 91

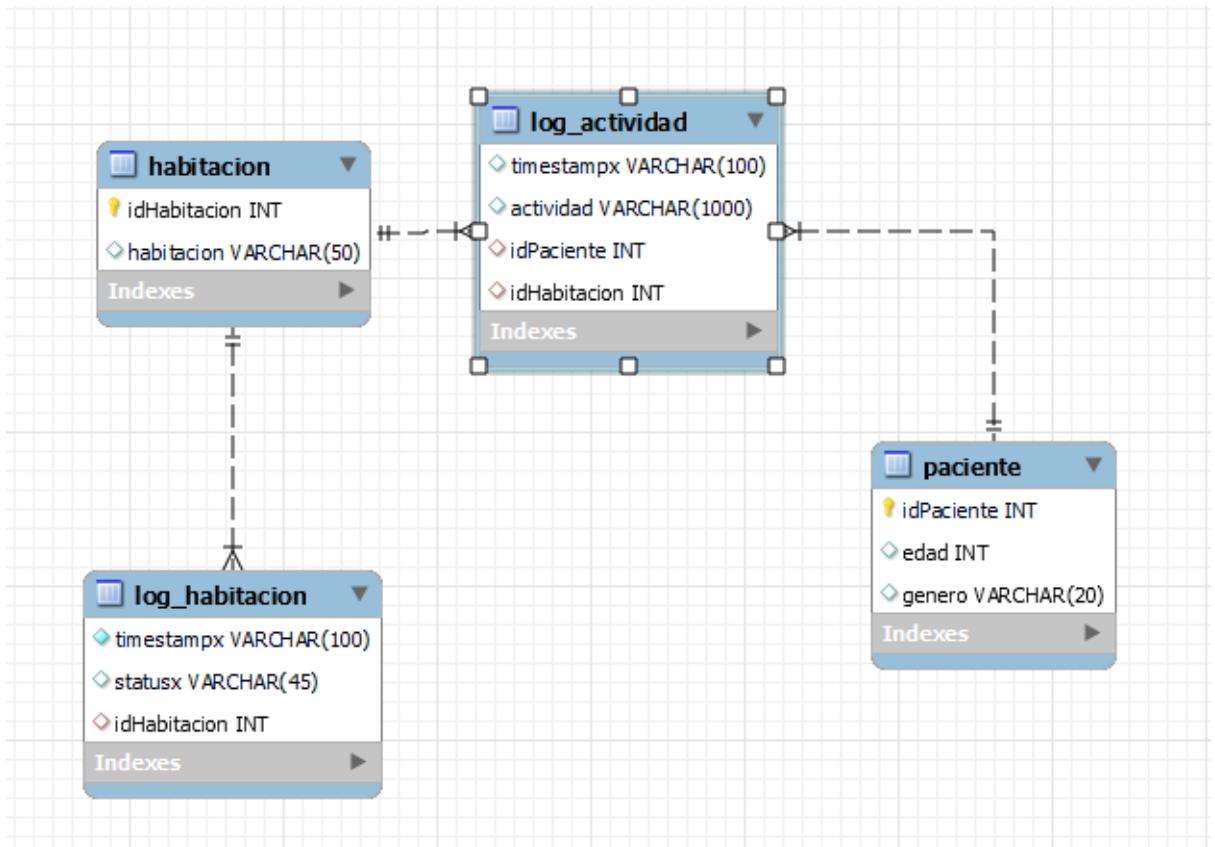
1 row

1	67684
---	-------

- Explicación con los modelos utilizados en cada base de datos.

GESTOR MYSQL

En nuestro modelo podemos observar 4 tablas, paciente contiene los datos de los pacientes y habitación que contiene las habitaciones de la clínica, con log_actividad la tabla sirve para llevar un monitoreo de los pacientes hospedados en cierta habitación, y log_habitacion para poder llevar un status de las habitaciones, una habitación llevará el monitoreo de varios estatus y en una habitación se hospedaron en su momento varios pacientes, y a un paciente se le puede asignar varias habitaciones.



➤ CONSULTA 1

Se hizo uso de la expresión case when para definir condiciones de cada categoría donde pediátrico es menor a 18, mediana edad entre 18 y 60 y el resto geriátrico, el count se utilizó para el conteo de los pacientes por cada categoría.

```

SELECT CASE
  WHEN p.edad < 18 THEN 'PEDIATRICO'
  WHEN p.edad BETWEEN 18 AND 60 THEN 'MEDIANA EDAD'
  ELSE 'GERIATRICO'
  END AS CATEGORIA
, count(p.idPaciente) TOTAL_PACIENTES
from PACIENTE p
GROUP BY CATEGORIA;
  
```

➤ CONSULTA 2

Para esta consulta se hizo la agrupación de los pacientes según la habitación de la clínica usando el group by, se hizo el uso de tres tablas paciente, log_actividad y habitación lo cual se hizo uso del left join para unirlos, se hizo uso de la función count para el conteo de los pacientes, having para mostrar resultados que sean mayores a 0 y se ordenó la cantidad de pacientes en forma descendente.

```

select
    h.habitacion
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by h.habitacion
having count(la.idPaciente) > 0
order by CANTIDAD desc, h.habitacion ;

```

➤ CONSULTA 3

Se utilizó left join para poder asociar el paciente con las actividades que realizó, tanto como con la habitación, además, se utilizó un group by para poder agrupar a todos los géneros, obteniendo la cantidad de pacientes que atendieron el género, y agregando un having para limitar que el paciente si llego por lo menos 1 vez a la habitación.

```

select
    p.genero
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by p.genero
having count(la.idPaciente) > 0
order by p.genero ASC, CANTIDAD asc;

```

➤ CONSULTA 4

Se utilizó left join para poder asociar el paciente con las actividades que realizó, tanto como con la habitación, además, se utilizó un group by para poder agrupar las edades, así obtener las edades más atendidas por el paciente, agregando un having para poder limitar que el paciente si llego por lo menos 1 vez a la habitación, ordenado de tal forma que sean descendente para obtener la mayor cantidad de edades atendidas y limitando a 5 edades.

```

select
    p.edad
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by p.edad
having count(la.idPaciente) > 0
order by CANTIDAD DESC, p.edad
limit 5;

```

➤ CONSULTA 5

Se utilizó left join para poder asociar el paciente con las actividades que realizó, tanto como con la habitación, además, se utilizó un group by para poder agrupar las edades, así obtener las edades más atendidas por el paciente, agregando un having para poder limitar que el paciente si llego por lo menos 1 vez a la habitación, ordenado de tal forma que sean ascendente para obtener la menor cantidad de edades atendidas y limitando a 5 edades.

```

-- CONSULTA 5
select
    p.edad
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by p.edad
having count(la.idPaciente) > 0
order by CANTIDAD asc, p.edad asc
limit 5;

```

➤ CONSULTA 6

Para esta consulta se hizo la agrupación de los pacientes según la habitación de la clínica usando el group by, se hizo el uso de tres tablas paciente, log_actividad y habitación lo cual se hizo uso del left join para unir las, se hizo uso de la función count para el conteo de los pacientes, having para mostrar resultados que sean mayores a 0 y se ordenó la cantidad de pacientes en forma descendente, con un límite de 5, con el objetivo de obtener las 5 habitaciones más utilizadas.

```
-- CONSULTA 6
select
    h.habitacion
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by h.habitacion
having count(la.idPaciente) > 0
order by CANTIDAD DESC, h.habitacion asc
limit 5;
```

➤ CONSULTA 7

Para esta consulta se hizo la agrupación de los pacientes según la habitación de la clínica usando el group by, se hizo el uso de tres tablas paciente, log_actividad y habitación lo cual se hizo uso del left join para unirlos, se hizo uso de la función count para el conteo de los pacientes, having para mostrar resultados que sean mayores a 0 y se ordenó la cantidad de pacientes en forma ascendente, con un límite de 5, con el objetivo de obtener las 5 habitaciones menos utilizadas.

```
-- CONSULTA 7
select
    h.habitacion
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by h.habitacion
having count(la.idPaciente) > 0
order by CANTIDAD ASC, h.habitacion asc
limit 5;
```

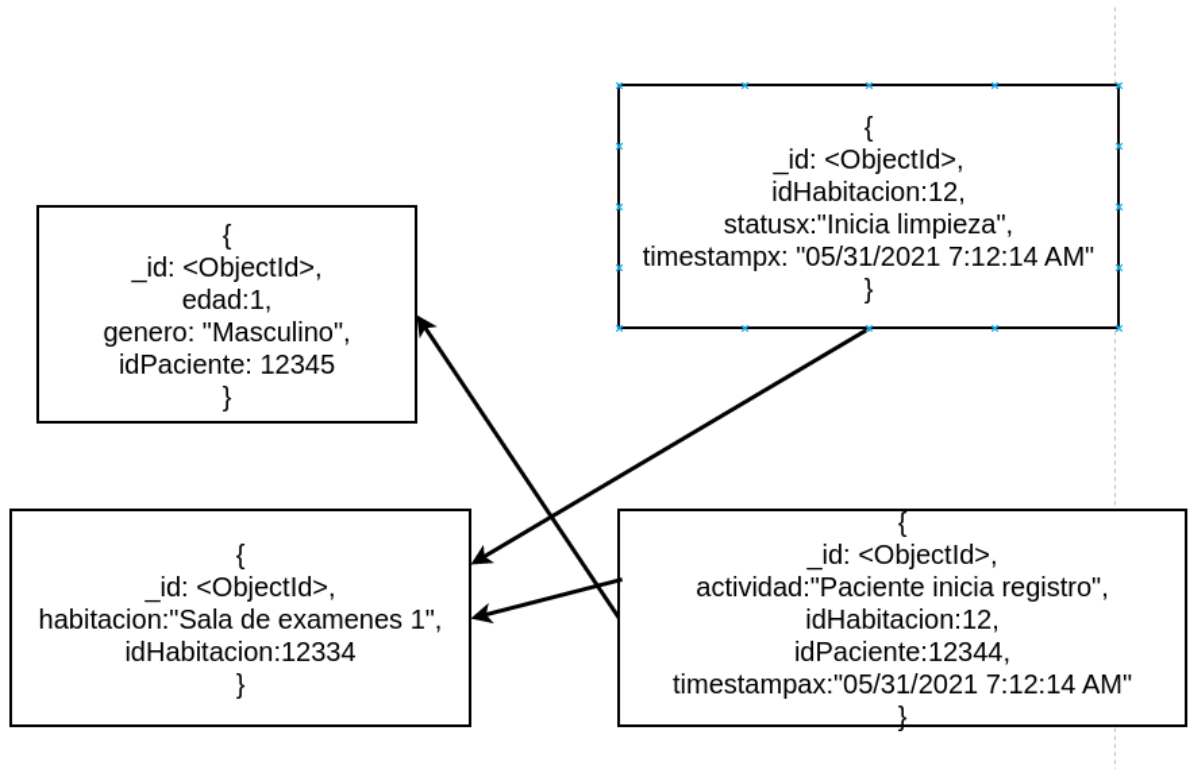
➤ CONSULTA 8

Se utilizó left join para poder asociar el paciente con las actividades que realizó, además, se convierte la cadena a tipo fecha, dónde se vuelve a cambiar el formato a únicamente día; se utilizó un group by para poder agrupar el día, así obtener el día con más pacientes, agregando un having para poder limitar que el paciente si llegó por lo menos 1 vez a la habitación, ordenado de tal forma que sean ascendente para obtener la menor cantidad de edades atendidas y

limitando a 5 edades.

```
-- CONSULTA 8
select
    date_format(STR_TO_DATE(la.timestampx, '%m/%d/%Y %h:%i:%s %p'), '%m/%d/%Y') dia
    ,count(la.idPaciente) as CANTIDAD
from DB_G8.PACIENTE p
left join DB_G8.LOG_ACTIVIDAD la on p.idPaciente = la.idPaciente
left join DB_G8.HABITACION h on la.idHabitacion = h.idHabitacion
group by dia
having count(la.idPaciente) > 0
order by CANTIDAD desc, dia asc
limit 1;
```

MONGODB MODELO



○ CONSULTA 1

La consulta es una operación de agregación en MongoDB que calcula la cantidad de pacientes en diferentes categorías de edad (pediátrico, mediana edad y geriátrico). Utiliza los operadores `$project` y `$group` para seleccionar y agrupar los datos según las categorías de edad definidas. Al final, proyecta solo los campos necesarios y renombra los campos de salida para una mejor legibilidad.

```
DB_G8.Paciente.aggregate([
  {
```



```

    $project: {
      _id: 0,
      CATEGORIA: {
        $switch: {
          branches: [
            { case: { $lt: ["$edad", 18] }, then: "PEDIATRICO" },
            { case: { $and: [{ $gte: ["$edad", 18] }, { $lte: ["$edad",
60] }] }, then: "MEDIANA EDAD" }
          ],
          default: "GENIATRICO"
        }
      },
      idPaciente: 1
    }
  },
  {
    $group: {
      _id: "$CATEGORIA",
      TOTAL_PACIENTES: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      CATEGORIA: "$_id",
      TOTAL_PACIENTES: 1
    }
  }
] );

```

○ CONSULTA 2

Esta consulta realiza una agregación en la colección "LogActividad" de la base de datos "DB_G8". La agregación agrupa los registros por "idHabitacion" y cuenta la cantidad de registros por cada grupo. Luego, los resultados son ordenados por cantidad descendente y se toman los primeros 15 registros.

```

DB_G8.LogActividad.aggregate([
  {
    $group: {
      _id: "$idHabitacion",
      CANTIDAD: { $sum: 1 }
    }
  },
  {
    $sort: { CANTIDAD: -1 }
  },
  {
    $limit: 15
  },
  {

```

```

    $lookup: {
      from: "Habitacion",
      localField: "_id",
      foreignField: "idHabitacion",
      as: "habitacion"
    }
  },
  {
    $project: {
      _id: 0,
      HABITACION: { $arrayElemAt: ["$habitacion.habitacion", 0] },
      CANTIDAD: 1
    }
  }
] );

```

- CONSULTA 3
- CONSULTA 4
- CONSULTA 5
- CONSULTA 6

Primero, se realiza un "lookup" en la colección "Habitacion" para obtener información adicional de cada habitación y se unen los resultados con los registros de "LogActividad" utilizando el campo "idHabitacion" de la primera colección y de la segunda. Luego, se deshace la agrupación creada por el "lookup" utilizando "\$unwind" para poder volver a agrupar los registros por "idHabitacion". En esta nueva agrupación, se cuenta la cantidad de registros por cada grupo y se utiliza "\$first" para obtener el nombre de la habitación a partir del campo "habitacion.habitacion" del primer documento encontrado en el arreglo "habitacion" de la colección "Habitacion".

Los resultados son ordenados por cantidad descendente y se toman los primeros 5 registros. Finalmente, se proyectan los resultados para mostrar sólo el nombre de la habitación y la cantidad de registros por cada grupo. El campo "_id" es excluido de los resultados.

```

DB_G8.LogActividad.aggregate([
  {
    $lookup: {
      from: "Habitacion",
      localField: "idHabitacion",
      foreignField: "idHabitacion",
      as: "habitacion"
    }
  },
  {
    $unwind: "$habitacion"
  },

```

```
{
  $group: {
    _id: "$idHabitacion",
    CANTIDAD: { $sum: 1 },
    HABITACION: { $first: "$habitacion.habitacion" }
  },
  $sort: { CANTIDAD: -1 },
  $limit: 5,
  $project: {
    _id: 0,
    HABITACION: 1,
    CANTIDAD: 1
  }
}
1);
```

○ CONSULTA 7

La consulta también realiza una agregación en la colección "LogActividad" de la base de datos "DB_G8". Primero, se realiza un "lookup" en la colección "Habitacion" para obtener información adicional de cada habitación y se unen los resultados con los registros de "LogActividad" utilizando el campo "idHabitacion" de ambas colecciones.

Luego, se deshace la agrupación creada por el "lookup" utilizando "\$unwind" para poder volver a agrupar los registros por "idHabitacion". En esta nueva agrupación, se cuenta la cantidad de registros por cada grupo y se utiliza "\$first" para obtener el nombre de la habitación a partir del campo "habitacion.habitacion" del primer documento encontrado en el arreglo "habitacion" de la colección "Habitacion".

Los resultados son ordenados por cantidad ascendente y se toman los primeros 5 registros. Finalmente, se proyectan los resultados para mostrar sólo el nombre de la habitación y la cantidad de registros por cada grupo. El campo "_id" es excluido de los resultados.

```
DB_G8.LogActividad.aggregate([
  {
    $lookup: {
      from: "Habitacion",
      localField: "idHabitacion",
      foreignField: "idHabitacion",
      as: "habitacion"
    }
  }
])
```

```

},
{
  $unwind: "$habitacion"
},
{
  $group: {
    _id: "$idHabitacion",
    CANTIDAD: { $sum: 1 },
    HABITACION: { $first: "$habitacion.habitacion" }
  }
},
{
  $sort: { CANTIDAD: 1 }
},
{
  $limit: 5
},
{
  $project: {
    _id: 0,
    HABITACION: 1,
    CANTIDAD: 1
  }
}
];

```

- CONSULTA 8
- Capturas de resultados de las consultas realizadas en ambas bases de datos.

GESTOR MYSQL

➤ CONSULTA 1

	CATEGORIA	TOTAL_PACIENTES
1	GENIATRICO	34089
2	MEDIANA EDAD	82254
3	PEDIATRICO	37841

➤ CONSULTA 2

	habitacion	CANTIDAD
1	Recepcion	37546
2	Estación de revisión 1	5286
3	Sala de exámenes 1	3778
4	Sala de exámenes 2	2968
5	Estación de revisión 2	2790
6	Sala de procedimientos 1	2154
7	Sala de exámenes 3	2139
8	Laboratorio	1876
9	Sala de procedimientos 2	1816
10	Sala de imagenes 1	1786
11	Sala de exámenes 4	1387
12	Sala de procedimientos 3	1352
13	Estación de revisión 3	1288
14	Sala de procedimientos 4	980
15	Estación de revisión 4	538

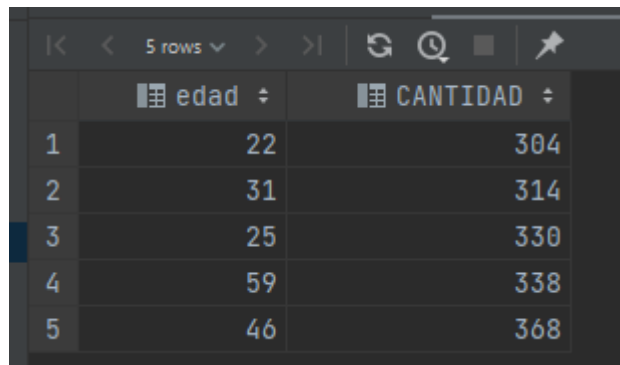
➤ CONSULTA 3

	genero	CANTIDAD
1	Femenino	33754
2	Masculino	31732
3	Otro	2198

➤ CONSULTA 4

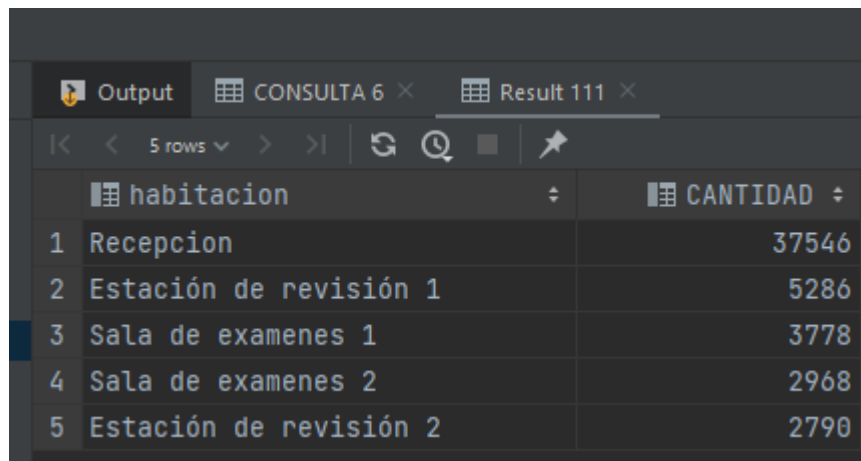
	edad	CANTIDAD
1	6	1872
2	2	1698
3	15	1664
4	11	1606
5	10	1592

➤ CONSULTA 5



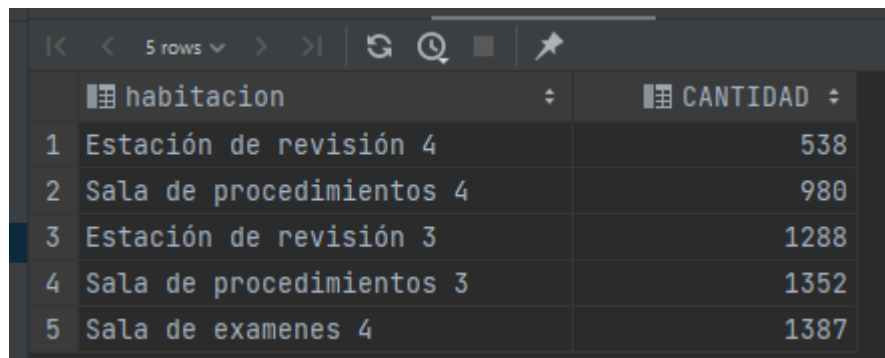
	edad	CANTIDAD
1	22	304
2	31	314
3	25	330
4	59	338
5	46	368

➤ CONSULTA 6



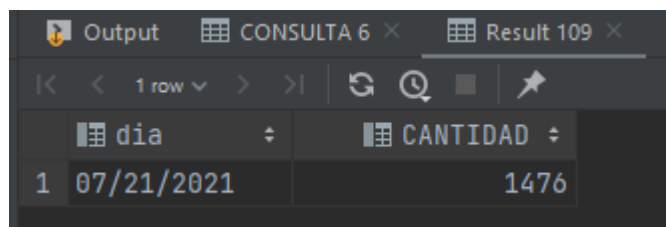
	habitacion	CANTIDAD
1	Recepcion	37546
2	Estación de revisión 1	5286
3	Sala de exámenes 1	3778
4	Sala de exámenes 2	2968
5	Estación de revisión 2	2790

➤ CONSULTA 7



	habitacion	CANTIDAD
1	Estación de revisión 4	538
2	Sala de procedimientos 4	980
3	Estación de revisión 3	1288
4	Sala de procedimientos 3	1352
5	Sala de exámenes 4	1387

➤ CONSULTA 8



	dia	CANTIDAD
1	07/21/2021	1476

MONGODB

➤ CONSULTA 1

Output Result 83		
3 rows		
	CATEGORIA	TOTAL_PACIENTES
1	GENIATRICO	34089
2	PEDIATRICO	37841
3	MEDIANA EDAD	82254

➤ CONSULTA 2

	CANTIDAD	HABITACION
1	37546	Recepcion
2	5286	Estación de revisión 1
3	3778	Sala de exámenes 1
4	2968	Sala de exámenes 2
5	2790	Estación de revisión 2
6	2154	Sala de procedimientos 1
7	2139	Sala de exámenes 3
8	1876	Laboratorio
9	1816	Sala de procedimientos 2
10	1786	Sala de imagenes 1
11	1387	Sala de exámenes 4
12	1352	Sala de procedimientos 3
13	1288	Estación de revisión 3
14	980	Sala de procedimientos 4
15	538	Estación de revisión 4

➤ CONSULTA 3

➤ CONSULTA 4

➤ CONSULTA 5

➤ CONSULTA 6

Output Result 85		
	CANTIDAD	HABITACION
1	37546	Recepcion
2	5286	Estación de revisión 1
3	3778	Sala de exámenes 1
4	2968	Sala de exámenes 2
5	2790	Estación de revisión 2

➤ CONSULTA 7

Output Result 86		
	CANTIDAD	HABITACION
1	538	Estación de revisión 4
2	980	Sala de procedimientos 4
3	1288	Estación de revisión 3
4	1352	Sala de procedimientos 3
5	1387	Sala de exámenes 4

➤ CONSULTA 8

● Conclusión y justificación del trabajo realizado

- El uso de la cláusula HAVING puede ser más eficiente que filtrar los resultados con una cláusula WHERE, especialmente cuando se trabaja con grandes conjuntos de datos. Esto se debe a que la cláusula HAVING filtra los resultados después de que se hayan agregado y agrupado los datos.
- Entre MySQL y MongoDB depende del tipo de aplicación y del uso que se les vaya a dar. MySQL es mejor para aplicaciones con transacciones ACID y un esquema de datos bien definido, mientras que MongoDB es mejor para aplicaciones web modernas con un alto rendimiento y datos no estructurados. La velocidad depende de

factores como la arquitectura de la aplicación, la complejidad de la consulta y el hardware subyacente.

- MongoDB es utilizado para datos no estructurados y con alto volumen de datos, sobre todo cuando se desea insertar datos de diferentes tipos de datos, en esta ocasión los datos no son variados y tampoco es un volumen alto por lo que la mejor opción sería seguir manejando base de datos relacionales.
- La agregación en MongoDB es una característica muy poderosa que permite a los usuarios procesar grandes conjuntos de datos y obtener resultados más específicos y personalizados que los que se pueden obtener con una simple consulta de MongoDB. La capacidad de crear canalizaciones de agregación personalizadas y combinar múltiples etapas de agregación hace que MongoDB sea una excelente opción para aplicaciones que requieren un procesamiento de datos avanzado.
- Las consultas en MongoDB que utilizan las etapas de agregación "project" y "group" permiten manipular y transformar los datos para presentarlos de una forma más útil y significativa. Estas dos etapas de agregación permiten transformar y presentar los datos de manera más útil y efectiva. Esto es especialmente importante cuando se trabaja con grandes cantidades de datos, ya que permite reducir el volumen de información a un conjunto manejable de resultados que se pueden utilizar para la toma de decisiones o para la generación de informes.

- **Link del repositorio.**

- https://github.com/bram814/BD2S12023_Grupo_8.git