**Documentación requerimientos funcionales**

**Requerimiento 1:**

Para este requerimiento se obtiene el identificador de cada habitación (habitacion\_id), el nombre de cada servicio (servicio) y calcula la suma total del precio de los servicios (dinero\_recolectado). Luego decidimos unir las tablas ‘habitaciones’ y ‘reservas’ por medio de los campos ‘id’ y ‘habitaciones\_id’. También se unieron las tablas ‘reservas\_servicios’ con ‘reservas’, a través de sus id’s. De igual manera sucedió con las tablas ‘reservas\_servicios’ y ‘servicios’. Para después agrupar los valores por el identificador de la habitación y el nombre del servicio para sumar correctamente los precios de servicios por habitación, y luego ordenarlos por el nombre del servicio.

@Query(value = "SELECT\r\n" +

" h.id AS habitacion\_id,\r\n" +

" s.nombre AS servicio,\r\n" +

" SUM(s.precio) AS dinero\_recolectado\r\n" +

"FROM\r\n" +

" habitaciones h\r\n" +

"JOIN\r\n" +

" reservas r ON h.id = r.habitaciones\_id\r\n" +

"JOIN\r\n" +

" reservas\_servicios rs ON r.id = rs.reservas\_id\r\n" +

"JOIN\r\n" +

" servicios s ON rs.servicios\_id = s.id\r\n" +

"WHERE\r\n" +

" r.fecha\_entrada >= (SELECT CURRENT\_DATE-365 from dual)\r\n" +

"GROUP BY\r\n" +

" h.id, s.nombre\r\n" +

"ORDER BY\r\n" +

" h.id, s.nombre"

, nativeQuery = true )

Collection<Object[]> darConsumos();

}

**Requerimiento 2:**

Para este requerimiento se seleccionan el nombre del servicio y la suma del consumo de los servicios en la tabla ‘reserva\_servicios’; Luego, se unen las tablas ‘reserva\_servicios’ con ‘servicios’ y de la misma manera se unen ‘reserva\_servicios’ con ‘reservas’. Después de esto, se aplica un filtrado por rango de fechas utilizando un where, para especificar fecha inicial y fecha final; Para luego agrupar los datos obtenidos por nombre, y ordenarlos en orden descendente por valor consumido, y mostrar únicamente los primeros 20 registros.

@Query(value = "SELECT s.nombre AS servicio, SUM(rs.cont\_servicios) AS total\_consumido " +

"FROM reservas\_servicios rs " +

"INNER JOIN servicios s ON rs.servicios\_id = s.id " +

"INNER JOIN reservas r ON rs.reservas\_id = r.id " +

"WHERE r.fecha\_entrada > TO\_DATE(:fechaI, 'YYYY-MM-DD') AND r.fecha\_salida < TO\_DATE(:fechaO,'YYYY-MM-DD')" +

"GROUP BY s.nombre " +

"ORDER BY total\_consumido DESC " +

"FETCH FIRST 20 ROWS ONLY", nativeQuery = true )

Collection<Object[]> darRta(@Param("fechaI") String fechaInicial, @Param("fechaO") String fechaFinal); }

**Requerimiento 3:**

Para este requerimiento se selecciona el id de la habitación y realizo una operación para sacar el índice de ocupación. Esto lo modelamos a través del atributo cont\_habiaciones, el cual se encarga de contabilizar la cantidad de veces que ha sido reservada una habitación, y este valor lo multiplico por 100, y lo divido entre la variable ‘Dia’, la cual corresponde al número del día dentro del año. Este valor lo sacamos por medio del controller, utilizando la función LocalDate.now() y getDayOfYear(). Así logramos obtener el índice de ocupación de cada una de las habitaciones del hotel

@Query(value = "select id, round(cont\_habitaciones\*100/ :Dia, 2) as resultado\_multiplicacion\r\n" +

"from habitaciones", nativeQuery = true )

Collection<Object[]> darRta(@Param("Dia")Integer diaActual);

}

**Requerimiento 5:**

Para este requerimiento se unen las tablas ‘usuarios’ y ‘reservas’, a través de los campos ‘id’ y ‘usuarios\_id’, de igual manera se unen las tablas ‘reservas’ y ‘reservas\_servicios’, ‘servicios’ y ‘reservas\_servicios’, ‘servicios’ y ‘servicios\_productos’, y ‘productos’ y ‘servicios\_productos’. Luego filtro los resultados por medio de las coincidencias con el login de usuario, y la fecha de inicio y finalización.

@Query(value = "SELECT\r\n" +

" u.login AS usuario,\r\n" +

" p.nombre AS producto\_nombre,\r\n" +

" rfs.fecha\_consumo AS fecha\_consumo,\r\n" +

" p.costo AS costo\_producto\r\n" +

"FROM\r\n" +

" usuarios u\r\n" +

"JOIN\r\n" +

" reservas r ON u.id = r.Usuarios\_id\r\n" +

"JOIN\r\n" +

" reservas\_servicios rfs ON r.id = rfs.reservas\_id\r\n" +

"JOIN\r\n" +

" servicios s ON rfs.servicios\_id = s.id\r\n" +

"JOIN\r\n" +

" servicios\_productos sp ON s.id = sp.servicios\_id\r\n" +

"JOIN\r\n" +

" productos p ON sp.productos\_id = p.id\r\n" +

"WHERE\r\n" +

" u.login = :login -- Reemplaza con el nombre de usuario que estás buscando\r\n" +

" AND r.fecha\_entrada >= TO\_DATE( :fechaI, 'YYYY-MM-DD')\r\n" +

" AND r.fecha\_salida <= TO\_DATE( :fechaO, 'YYYY-MM-DD')", nativeQuery = true )

Collection<Object[]> darRta(@Param("fechaI") String fechaInicial, @Param("fechaO") String fechaFinal, @Param(value = "login") String login);

}

**Requerimiento 6:**

Para este requerimiento se realiza inicialmente una consulta llamada OcupacionDiaria para calcular la ocupación diaria de las habitaciones basándose en las reservas con check-in confirmado. Filtra las reservas por fechas entre las fechas mínimas y máximas encontradas en las entradas y salidas de todas las reservas. Luego, se agrupan los resultados por fecha\_entrada y cuenta el número de habitaciones distintas ocupadas por día. Finalmente, selecciona las fechas y el número de habitaciones ocupadas, ordenando los resultados en orden descendente de habitaciones ocupadas.

Este proceso se realiza de la misma manera para el cálculo de mayores ingresos, pero con se ordena con un orden ascendente.

@Query(value = "WITH OcupacionDiaria AS (\r\n" +

"SELECT\r\n" +

"fecha\_entrada AS fecha\_reserva,\r\n" +

"COUNT(DISTINCT habitaciones\_id) AS habitaciones\_ocupadas\r\n" +

"FROM\r\n" +

"reservas\r\n" +

"WHERE\r\n" +

"check\_in = 1 -- Asegúrate de que solo estás considerando reservas con check-in confirmado\r\n" +

"AND fecha\_entrada BETWEEN (SELECT MIN(fecha\_entrada) FROM reservas) AND (SELECT MAX(fecha\_salida) FROM reservas)\r\n" +

"GROUP BY\r\n" +

"fecha\_entrada\r\n" +

")\r\n" +

"SELECT\r\n" +

"fecha\_reserva, habitaciones\_ocupadas\r\n" +

"FROM\r\n" +

"OcupacionDiaria\r\n" +

"ORDER BY\r\n" +

"habitaciones\_ocupadas DESC\r\n" +

"FETCH FIRST 50 ROWS ONLY"

, nativeQuery = true )

Collection<Object[]> darPrimeraParte();

@Query(value = "WITH IngresosDiarios AS (\r\n" +

"SELECT\r\n" +

"fecha\_consumo AS fecha,\r\n" +

"SUM(cont\_servicios) AS total\_ingresos\r\n" +

"FROM\r\n" +

"reservas\_servicios\r\n" +

"GROUP BY\r\n" +

"fecha\_consumo\r\n" +

")\r\n" +

"SELECT\r\n" +

"fecha, total\_ingresos\r\n" +

"FROM\r\n" +

"IngresosDiarios\r\n" +

"ORDER BY\r\n" +

"total\_ingresos DESC"

, nativeQuery = true )

Collection<Object[]> darSegundaParte();

@Query(value = "WITH OcupacionDiaria AS (\r\n" +

"SELECT\r\n" +

"fecha\_entrada AS fecha\_reserva,\r\n" +

"COUNT(DISTINCT habitaciones\_id) AS habitaciones\_ocupadas\r\n" +

"FROM\r\n" +

"reservas\r\n" +

"WHERE\r\n" +

"check\_in = 1 -- Asegúrate de que solo estás considerando reservas con check-in confirmado\r\n" +

"AND fecha\_entrada BETWEEN (SELECT MIN(fecha\_entrada) FROM reservas) AND (SELECT MAX(fecha\_salida) FROM reservas)\r\n" +

"GROUP BY\r\n" +

"fecha\_entrada\r\n" +

")\r\n" +

"SELECT\r\n" +

"fecha\_reserva, habitaciones\_ocupadas\r\n" +

"FROM\r\n" +

"OcupacionDiaria\r\n" +

"ORDER BY\r\n" +

"habitaciones\_ocupadas ASC\r\n" +

"FETCH FIRST 50 ROWS ONLY"

, nativeQuery = true )

Collection<Object[]> darTerceraParte();

}

**Requerimiento 7:**

Para este requerimiento, primeramente calculamos el consumo total de cada usuario (total\_consumido) en reservas durante el último año (hasta la fecha actual), y se revisa si el resultado es mayor a $15.000.000. Luego se hace otra subconsulta que calcula el total de días de estadía (total\_dias\_estadia) de cada usuario, donde el total de días es igual o superior a 14 días. Luego se filtran las reservas que tienen una fecha\_entrada dentro del último año y hasta la fecha actual y se agrupan los resultados por usuario\_id y usuario\_login.

Finalmente se realiza un join entre las dos subconsultas en el campo ‘usuario\_id’ y Devuelve una colección de objetos donde cada objeto contiene los campos usuario\_id, usuario\_login, total\_dias\_estadia, y total\_consumido para los usuarios que cumplen con ambas condiciones: un consumo total mayor a $15.000.000 y una estadía total igual o mayor a 14 días, en el último año.

@Query(value = "SELECT consumo.usuario\_id, consumo.usuario\_login, estadia.total\_dias\_estadia, total\_consumido\r\n" + //

"FROM (\r\n" +

" -- Resultado del cálculo de consumo\r\n" +

" SELECT u.id AS usuario\_id, u.login AS usuario\_login, SUM(r.costo\_total) AS total\_consumido\r\n" +

" FROM usuarios u\r\n" +

" JOIN reservas r ON u.id = r.usuarios\_id\r\n" +

" WHERE r.fecha\_entrada >= TO\_DATE(SYSDATE - 365, 'DD/MM/YYYY')\r\n" +

" GROUP BY u.id, u.login\r\n" +

" HAVING SUM(r.costo\_total) > 15000000\r\n" +

") consumo\r\n" +

"JOIN (\r\n" +

" -- Resultado del cálculo de duración de estadía\r\n" +

" SELECT u.id AS usuario\_id, u.login AS usuario\_login, SUM(ABS(r.fecha\_salida - r.fecha\_entrada)) AS total\_dias\_estadia\r\n" +

" FROM usuarios u\r\n" +

" JOIN reservas r ON u.id = r.usuarios\_id\r\n" +

" WHERE r.fecha\_entrada >= TO\_DATE(SYSDATE - 365, 'DD/MM/YYYY') and r.fecha\_entrada <= (SELECT CURRENT\_DATE from dual)\r\n" +

" GROUP BY u.id, u.login\r\n" +

" HAVING SUM(ABS(r.fecha\_salida - r.fecha\_entrada)) >= 14\r\n" +

") estadia\r\n" +

"ON consumo.usuario\_id = estadia.usuario\_id\r\n" +

"", nativeQuery = true )

Collection<Object[]> darRta();

}

**Requerimiento 8:**

Para este requerimiento se unen las tablas ‘reservas\_servicios’ con ‘servicios’ y ‘reservas’ con ‘reservas\_servicios’. Donde se tienen en cuenta los datos para el último año de operación, se agrupan los datos por nombre y filtra los datos para obtener aquellos servicios que han sido menos solicitados.

@Query(value = "SELECT s.nombre AS nombre\_del\_servicio, COUNT(rs.reservas\_id) AS cantidad\_de\_solicitudes\r\n" +

"FROM servicios s\r\n" +

"LEFT JOIN reservas\_servicios rs ON s.id = rs.servicios\_id\r\n" +

"LEFT JOIN reservas r ON rs.reservas\_id = r.id\r\n" +

"WHERE r.fecha\_entrada >= (SYSDATE - 365) -- Último año de operación\r\n" +

"GROUP BY s.nombre\r\n" +

"HAVING COUNT(rs.reservas\_id) < 3\r\n" +

"ORDER BY cantidad\_de\_solicitudes\r\n" +

"", nativeQuery = true )

Collection<Object[]> darRta();

}