Laura Calderón - 202122045

David Tobón

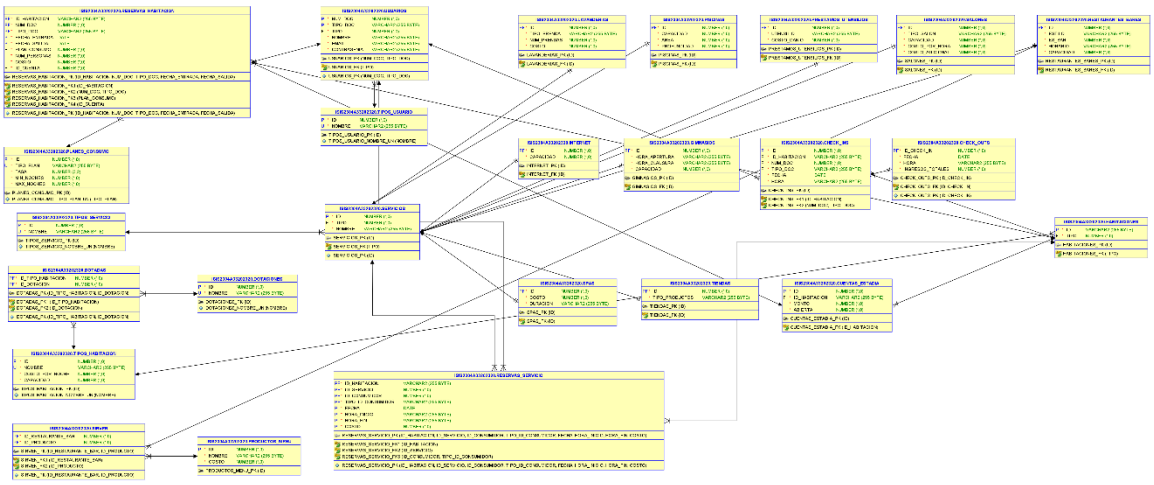
Esteban Orjuela - 202211227

ENTREGA 2 – PROYECTO

❖ Análisis:

La nueva versión del esquema de la base de datos para esta entrega introduce ciertos cambios en su estructura y diseño en comparación con la versión anterior. En primer lugar, se observa una diferencia notable en la nomenclatura de las tablas y columnas. Mientras que la versión vieja utiliza nombres en mayúsculas y guiones bajos, la nueva versión opta por nombres más descriptivos y legibles que combinan letras mayúsculas y minúsculas para mejorar el manejo de estas columnas con las consultas necesarias de la entrega 2. Además, la nueva versión incorpora el uso de secuencias, como "id_tipos_habitacion," para generar valores de clave primaria en algunas tablas, lo que automatiza la generación de identificadores únicos y simplifica la gestión de datos. En contraste, en la versión vieja, los valores de clave primaria se ingresan manualmente. Por ejemplo, "id_tipos_habitacion" se define como una secuencia en la nueva versión, mientras que, en la vieja, se establece simplemente como "id INTEGER.” Otro cambio importante se observa en la definición de restricciones de clave foránea. En la nueva versión, se utilizan restricciones de clave foránea explícitas, como "CONSTRAINT habitaciones_FK FOREIGN KEY (tipo) REFERENCES tipos_habitacion(id)," para definir relaciones entre tablas, lo que proporciona una mayor claridad y control sobre las relaciones entre los datos. En contraste, la versión vieja define estas relaciones directamente en la columna, como "tipo INTEGER NOT NULL," sin una restricción de clave foránea explícita. En cuanto a los tipos de datos y restricciones, la nueva versión utiliza tipos de datos más específicos, como "NUMERIC" con restricciones de verificación, como "CHECK (costo_por_noche > 0)," para garantizar la integridad de los datos. En la versión vieja, se utilizan "NUMERIC" o "VARCHAR2" sin restricciones de verificación en algunos campos, lo que puede llevar a problemas de integridad de datos en las consultas si no se realizaba el cambio. Finalmente, la nueva versión habilita la eliminación en cascada mediante la cláusula "ON DELETE CASCADE" en diversas tablas como la de "check_outs," lo que permite la eliminación automática de registros relacionados entre tablas. En resumen, la nueva versión del esquema de la base de datos representa una mejora significativa en términos de legibilidad, automatización de la generación de claves primarias, definición de relaciones de clave foránea, uso de tipos de datos más específicos y garantía de la integridad de los datos, lo que facilita la administración y el mantenimiento de la base de datos para la realización de los requerimientos de esta entrega.

Por lo anterior, a continuación, se adjunta el modelo relacional actualizado de la base de datos:



❖ Índices

- Para cada uno de los requerimientos funcionales:
    - Identifiquen si es necesario crear un índice.
    - Justifiquen la selección del índice creado desde el punto de vista de la selectividad (concepto visto en clase). En caso de que para algún requerimiento funcional Uds determinen que un índice no es necesario, justifíquenlo también desde el punto de vista de la selectividad.
    - Indiquen claramente cuál es el tipo de índice utilizado (B+, Hash, …, primario, secundario).

Para la ejecución de los requerimientos, consideramos pertinente crear índices para:

Requerimiento 1:

- La selectividad se refiere a la proporción de filas que cumplen con una condición específica en relación con el total de filas en la tabla. En el caso del requerimiento 1 en la tabla reservas_servicio, se puede crear un índice secundario en el campo fecha para acelerar la búsqueda de registros relacionados con reservas de servicios en un año específico, en este caso, el año 2023.En la tabla servicios, el campo id es utilizado para unirse con reservas_servicio, por lo para esta tabla también puede haber un índice.

Requerimiento 2:

- En la tabla reservas_servicio, se puede crear índice secundario en el campo fecha para acelerar la búsqueda de registros relacionados con reservas de servicios dentro del rango de fechas especificado. Además, en la tabla servicios, el campo id se utiliza para unirse con reservas_servicio, sin embargo, al ser una llave primaria no es necesario crear dicho índice.

Requerimiento 3:

- En la tabla reservas_habitacion, el campo id_habitacion se utiliza en la agrupación y en la condición de filtro. Por lo tanto, se puede crear un índice secundario en id_habitacion para acelerar la búsqueda de datos relacionados con cada habitación. Dado que el campo id_habitacion no es una clave primaria en la tabla reservas_habitacion, se trata de un índice secundario.

Requerimiento 4:

- Dado que la consulta realiza múltiples uniones y filtrados en diferentes tablas, la creación de índices adecuados en las columnas relevantes ayudará a mejorar el rendimiento de la consulta al acelerar la búsqueda y selección de datos. Teniendo en cuenta lo anterior, para este requerimiento, se podrían crear índices en las columnas involucradas en las condiciones de costo y capacidad. Los tipos de índice pueden ser índices secundarios en las columnas relevantes de las tablas lavanderías, prestamos_utensilios, salones y spas para las condiciones de costo, y también en las columnas de las tablas gimnasios, piscinas, salones y restaurantes_bares para la condición de capacidad.

Requerimiento 5:

- Se podría crear un índice secundario en el campo fecha de la tabla reservas_servicio. Esto acelerará la búsqueda de registros relacionados con las reservas de servicio que caen dentro del rango de fechas especificado en la consulta. Al indexar este campo, las búsquedas serán más eficientes. Por otro lado, se podrían poner Índices secundarios en la tabla usuarios en los campos num_doc y tipo_doc. Lo anterior permitiría acelerar la búsqueda de usuarios específicos. Esto mejorará la eficiencia de la consulta al filtrar usuarios basados en su número de documento y tipo de documento. También, sería útil crear un Índice en la tabla habitaciones en el campo id.

Este índice facilitará la operación de unión con la tabla reservas_servicio, mejorando el rendimiento de la consulta. Por útlimo, se podría crear un índice secundario en la tabla servicios en el campo id. Esto acelerará la operación de unión con la tabla reservas_servicio.

Requerimiento 6:

- Se podría crear un índice secundario en el campo fecha_entrada de la tabla reservas_habitacion, ya que se utiliza en las tres partes de la consulta para evaluar la ocupación en diferentes momentos. Además, un índice en el campo fecha de la tabla reservas_servicio sería útil, ya que se utiliza para calcular los ingresos. Estos índices mejorarían la velocidad de búsqueda y agregación de datos.

Requerimiento 7:

- En la tabla usuarios, un índice secundario en num_doc para acelerar la búsqueda de usuarios por número de documento y un índice secundario en nombre si es necesario realizar búsquedas por nombre de usuario. En la tabla reservas_habitacion, un índice secundario en num_doc para acelerar la búsqueda de reservas por número de documento del usuario y un índice secundario en fecha_salida si es común buscar reservas por fecha de salida. En la tabla reservas_servicio, un índice secundario en id_consumidor y fecha para acelerar la búsqueda de reservas de servicios por el consumidor y fecha.

Requerimiento 8:

- Se podría considerar la creación de un índice secundario en el campo fecha de la tabla reservas_servicio para acelerar la búsqueda de registros que cumplen con la condición de fecha. Además, en la tabla servicios, un índice secundario en el campo id podría mejorar la operación de unión. Asimismo, se podría pensar en la creación de un índice secundario en el campo nombre de la tabla servicios, ya que se utiliza en la cláusula GROUP BY. La combinación de estos índices permitiría una búsqueda más eficiente de datos y un procesamiento más rápido de la consulta en su totalidad.

Requerimiento 9:

- En primer lugar, se podría considerar la creación de un índice compuesto en los campos num_doc, tipo_doc y id_consumidor en la tabla reservas_servicio, ya que se utilizan para la unión y la filtración. Además, un índice en el campo fecha de la misma tabla agilizaría la búsqueda de registros dentro del rango de fechas especificado. Asimismo, en la tabla usuarios, la creación de un índice secundario en los campos num_doc y tipo_doc permitiría una búsqueda más eficiente de usuarios. La combinación de estos índices optimizaría el proceso de búsqueda y recuperación de datos, lo que a su vez mejoraría el rendimiento general de la consulta.

Requerimiento 10:

- Para mejorar el rendimiento del requerimiento 10, se pueden considerar los siguientes índices. En la tabla usuarios, el campo num_doc ya es una clave primaria (PK) y se utilizará en la condición de filtro. Debido a lo anterior, no se requiere un índice adicional. Sin embargo, en la tabla reservas_servicio, los campos id_consumidor, tipo_id_consumidor, id_servicio, y fecha se utilizan en la subconsulta. Teniendo esto en cuenta, se podría crear un índice secundario en estos campos para acelerar la búsqueda de reservas de servicio según el servicio, el consumidor y la fecha.

Requerimiento 11:

- Se podría hacer la creación de un índice secundario en el campo fecha de la tabla reservas_servicio, lo que permite agilizar las operaciones de filtrado por fecha, un componente crítico en todas las partes de la consulta. Asimismo, realizar la creación de un índice secundario

en el campo nombre de la tabla servicios sería útil, ya que se utiliza para agrupar y seleccionar los servicios más y menos consumidos. En el contexto de las habitaciones, se podría realizar la creación de un índice secundario en el campo fecha_entrada de la tabla reservas_habitacion, lo que mejora la eficiencia de las consultas relacionadas con las habitaciones más y menos solicitadas.

Requerimiento 12:

- Se puede crear un índice secundario en el campo fecha de la tabla check_ins, ya que este campo se utiliza en la primera parte de la consulta para identificar estancias trimestrales. Además, un índice en el campo costo de la tabla reservas_servicio sería beneficioso, ya que se utiliza para filtrar servicios costosos en la segunda parte de la consulta. Asimismo, en la tercera parte de la consulta, se podría considerar la creación de un índice secundario en los campos hora_inicio y hora_fin de la tabla reservas_servicio, lo que facilitaría la evaluación de la duración de los servicios de SPA o salones de reuniones.

Índices creados por Oracle:

Oracle creo los siguientes índices automáticamente:

	INDEX_NAME	STATUS	COLUMNS	COMMENTS	UNIQUENESS	DISTINCT_KEYS	NUM_ROWS	LAST_ANALYZED	LAST_DDL_TIME	CREATED
1	USUARIOS_PK	VALID	2	(null)	UNIQUE	1153	1153	05-NOV-23	05-NOV-23	05-NOV-23
2	RESERVAS_SERVICIO_PK	VALID	8	(null)	UNIQUE	5	5	07-NOV-23	05-NOV-23	05-NOV-23
3	SIRVEN_PK	VALID	2	(null)	UNIQUE	50	50	05-NOV-23	05-NOV-23	05-NOV-23
4	DOTADAS_PK	VALID	2	(null)	UNIQUE	30	30	05-NOV-23	05-NOV-23	05-NOV-23
5	RESERVAS_HABITACION_PK	VALID	5	(null)	UNIQUE	299	299	07-NOV-23	05-NOV-23	05-NOV-23
6	SERVICIOS_PK	VALID	1	(null)	UNIQUE	32	32	05-NOV-23	05-NOV-23	05-NOV-23
7	CHECK_OUTS_PK	VALID	1	(null)	UNIQUE	299	299	05-NOV-23	05-NOV-23	05-NOV-23

Los índices de clave primaria mostrados, USUARIOS_PK, RESERVAS_SERVICIO_PK, SIRVEN_PK, DOTADAS_PK, RESERVAS_HABITACION_PK, SERVICIOS_PK y CHECK_OUTS_PK, son componentes esenciales en la base de datos de Oracle, ya que garantizan la unicidad de los registros y mejoran el rendimiento de las operaciones de búsqueda y actualización. Por esta razón, estos índices son creados automáticamente por Oracle cuando se define una columna como clave primaria en una tabla.

❖ Diseño de las consultas

Teniendo en cuenta el análisis realizado anteriormente, los índices creados fueron los siguientes:

```
--INDICES:
-- Índice en la tabla `reservas_servicio` en el campo `fecha`
CREATE INDEX idx_reservas_servicio_fecha ON reservas_servicio (fecha);

-- Índice secundario en la tabla `servicios` en el campo `nombre`
CREATE INDEX idx_servicios_nombre ON servicios (nombre);

-- Índice en la tabla `reservas_habitacion` en el campo `fecha_entrada`
CREATE INDEX idx_reservas_habitacion_fecha_entrada ON reservas_habitacion (fecha_entrada);

-- Índice secundario en la tabla `habitaciones` en el campo `fecha_entrada`
CREATE INDEX idx_habitaciones_fecha_entrada ON habitaciones (fecha_entrada);

-- Índice secundario en el campo `costo` de la tabla `reservas_servicio`
CREATE INDEX idx_reservas_servicio_costo ON reservas_servicio (costo);

-- Índice secundario en los campos `hora_inicio` y `hora_fin` de la tabla `reservas_servicio`
CREATE INDEX idx_reservas_servicio_horas ON reservas_servicio (hora_inicio, hora_fin);

-- Índice en la tabla `reservas_habitacion` en el campo `fecha_entrada`
CREATE INDEX idx_reservas_habitacion_fecha_entrada ON reservas_habitacion (fecha_entrada);
```

RF1:

```
--CONSULTA PARA REQUERIMIENTO 1
SELECT reservas_servicio.id_habitacion, servicios.nombre,'$' || TO_CHAR(SUM(reservas_servicio.costo), 'FM999,999,999.99') AS total_ganancias
FROM reservas_servicio
INNER JOIN servicios ON servicios.id = reservas_servicio.id_servicio
WHERE EXTRACT(YEAR FROM reservas_servicio.fecha) = 2023
GROUP BY reservas_servicio.id_habitacion, servicios.nombre
ORDER BY reservas_servicio.id_habitacion;
```

RF2:

```
SELECT rs.id_servicio, s.nombre AS nombre_servicio, COUNT(rs.id_servicio) AS consumos
FROM reservas_servicio rs
JOIN servicios s ON rs.id_servicio = s.id
WHERE rs.fecha BETWEEN TO_DATE('01/01/2010', 'DD/MM/YYYY') AND TO_DATE('01/01/2024', 'DD/MM/YYYY')
GROUP BY rs.id_servicio, s.nombre
ORDER BY consumos DESC
FETCH FIRST 20 ROWS ONLY;
```

RF3:

```
-- Req3

SELECT
    r.id_habitacion AS ID_Habitacion,
    COUNT(DISTINCT r.fecha_entrada) AS Dias_Ocupados,
    (COUNT(DISTINCT r.fecha_entrada) / 365.0) * 100 AS Tasa_Ocupacion
FROM
    reservas_habitacion r
WHERE
    r.fecha_entrada BETWEEN SYSDATE - INTERVAL '1' YEAR AND SYSDATE
GROUP BY
    r.id_habitacion
ORDER BY
    Tasa_Ocupacion DESC;
```

RF4:

```sql
--CONSULTA PARA REQUERIMIENTO 4
SELECT servicios.nombre,
    COALESCE(gimnasios.capacidad, piscinas.capacidad, salones.capacidad, restaurantes_bares.capacidad) AS capacidad_total,
    CASE
        WHEN lavanderias.costo IS NOT NULL THEN lavanderias.costo
        WHEN prestamos_utensilios.costo_danio IS NOT NULL THEN prestamos_utensilios.costo_danio
        WHEN salones.costo_por_hora IS NOT NULL THEN salones.costo_por_hora
        WHEN spas.costo IS NOT NULL THEN spas.costo
        ELSE 0
    END AS costo_del_servicio
FROM servicios
LEFT JOIN gimnasios ON gimnasios.id = servicios.id
LEFT JOIN piscinas ON piscinas.id = servicios.id
LEFT JOIN salones ON salones.id = servicios.id
LEFT JOIN restaurantes_bares ON restaurantes_bares.id = servicios.id
LEFT JOIN lavanderias ON lavanderias.id = servicios.id
LEFT JOIN prestamos_utensilios ON prestamos_utensilios.id = servicios.id
LEFT JOIN spas ON spas.id = servicios.id

WHERE (CASE
        WHEN lavanderias.costo IS NOT NULL THEN lavanderias.costo
        WHEN prestamos_utensilios.costo_danio IS NOT NULL THEN prestamos_utensilios.costo_danio
        WHEN salones.costo_por_hora IS NOT NULL THEN salones.costo_por_hora
        WHEN spas.costo IS NOT NULL THEN spas.costo
        ELSE 0
    END >= :costo_minimo AND
    CASE
        WHEN lavanderias.costo IS NOT NULL THEN lavanderias.costo
        WHEN prestamos_utensilios.costo_danio IS NOT NULL THEN prestamos_utensilios.costo_danio
        WHEN salones.costo_por_hora IS NOT NULL THEN salones.costo_por_hora
        WHEN spas.costo IS NOT NULL THEN spas.costo
        ELSE 0
    END <= :costo_maximo)
    OR
    (COALESCE(gimnasios.capacidad, piscinas.capacidad, salones.capacidad, restaurantes_bares.capacidad) >= :capacidad_minima AND
    COALESCE(gimnasios.capacidad, piscinas.capacidad, salones.capacidad, restaurantes_bares.capacidad) <= :capacidad_maxima);
```

RF5:

```sql
SELECT u.nombre AS nombre_cliente, s.nombre AS nombre_servicio, rs.fecha, rs.hora_inicio, rs.hora_fin, rs.costo
FROM reservas_servicio rs
JOIN habitaciones h ON rs.id_habitacion = h.id
JOIN reservas_habitacion rh ON h.id = rh.id_habitacion
JOIN usuarios u ON rh.num_doc = u.num_doc AND rh.tipo_doc = u.tipo_doc
JOIN servicios s ON rs.id_servicio = s.id
WHERE u.num_doc = numero_de_doc
    AND u.tipo_doc = 'tipo_doc'
    AND rs.fecha BETWEEN 'fecha_inicial' AND 'fecha_final'
ORDER BY rs.fecha, rs.hora_inicio;
```

RF6:

```sql
-- Req6

SELECT fecha_entrada AS Fecha, COUNT(*) AS Ocupacion
FROM reservas_habitacion
GROUP BY fecha_entrada
ORDER BY Ocupacion DESC
FETCH FIRST 3 ROWS ONLY;

SELECT fecha as Fecha, SUM(costo) AS Ingresos
FROM reservas_servicio
GROUP BY fecha
ORDER BY Ingresos DESC
FETCH FIRST 3 ROWS ONLY;

SELECT fecha_entrada AS Fecha, COUNT(*) AS Ocupacion
FROM reservas_habitacion
GROUP BY fecha_entrada
ORDER BY Ocupacion ASC
FETCH FIRST 3 ROWS ONLY;
```

RF7:

```sql
--CONSULTA PARA REQUERIMIENTO 7
WITH DatosCliente AS (
    SELECT
        usuarios.num_doc AS num_documento,
        usuarios.nombre AS clientes_buenos,
        SUM(rh.fecha_salida - rh.fecha_entrada) AS dias_hospedado,
        SUM(COALESCE(rh.costo, 0) + COALESCE(rs.costo, 0)) AS costo_total
    FROM usuarios
    LEFT JOIN reservas_habitacion rh ON usuarios.num_doc = rh.num_doc
    LEFT JOIN reservas_servicio rs ON usuarios.num_doc = rs.id_consumidor
    WHERE EXTRACT(YEAR FROM rh.fecha_salida) = 2023 OR EXTRACT(YEAR FROM rs.fecha) = 2023
    GROUP BY usuarios.num_doc, usuarios.nombre
)
SELECT num_documento, clientes_buenos, dias_hospedado, costo_total
FROM DatosCliente
WHERE costo_total > 15000000 OR dias_hospedado > 14
ORDER BY costo_total DESC;
```

RF8:

```sql
SELECT s.nombre AS servicio, COUNT(rs.id_servicio) AS consumos, COUNT(rs.id_servicio) / 52 AS avg_semanal
FROM reservas_servicio rs
JOIN servicios s ON rs.id_servicio = s.id
WHERE rs.fecha >= SYSDATE - INTERVAL '1' YEAR -- Data from the last year
GROUP BY s.nombre
HAVING COUNT(rs.id_servicio) / 52 < 3
ORDER BY avg_semanal;
```

RF9:

```sql
-- Req9

SELECT u.num_doc, u.tipo_doc, u.nombre, r.fecha, COUNT(*) as cantidad
FROM usuarios u
JOIN reservas_servicio r ON u.num_doc = r.id_consumidor AND u.tipo_doc = r.tipo_id_consumidor
WHERE r.id_servicio = :service_id AND r.fecha BETWEEN :start_date AND :end_date
GROUP BY u.num_doc, u.tipo_doc, u.nombre, r.fecha
ORDER BY :order_by;
```

RF10:

```sql
--CONSULTA PARA REQUERIMIENTO 10
SELECT u.num_doc, u.tipo_doc, u.nombre
FROM usuarios u
WHERE (u.num_doc, u.tipo_doc) NOT IN (
    SELECT r.id_consumidor, r.tipo_id_consumidor
    FROM reservas_servicio r
    WHERE r.id_servicio = :service_id
    AND r.fecha BETWEEN :start_date AND :end_date
)
ORDER BY :order_by;
```

RF11:

```sql
/*
Servicios más consumidos
*/
SELECT semana,
    nombre AS servicio_mas_consumido,
    consumos
FROM (
    SELECT TO_CHAR(rs.fecha, 'IW') AS semana,
           s.nombre,
           COUNT(*) AS consumos,
           ROW_NUMBER() OVER (PARTITION BY TO_CHAR(rs.fecha, 'IW') ORDER BY COUNT(*) DESC) AS rn
    FROM reservas_servicio rs
    JOIN servicios s ON rs.id_servicio = s.id
    WHERE rs.fecha BETWEEN TRUNC(SYSDATE, 'YEAR') - INTERVAL '1' YEAR AND TRUNC(SYSDATE)
    GROUP BY TO_CHAR(rs.fecha, 'IW'), s.nombre
)
WHERE rn = 1;
```

```sql
/*
Servicios menos consumidos
*/
SELECT semana,
    nombre AS servicio_menos_consumido,
    consumos
FROM (
    SELECT TO_CHAR(rs.fecha, 'IW') AS semana,
           s.nombre,
           COUNT(*) AS consumos,
           ROW_NUMBER() OVER (PARTITION BY TO_CHAR(rs.fecha, 'IW') ORDER BY COUNT(*)) AS rn
    FROM reservas_servicio rs
    JOIN servicios s ON rs.id_servicio = s.id
    WHERE rs.fecha BETWEEN TRUNC(SYSDATE, 'YEAR') - INTERVAL '1' YEAR AND TRUNC(SYSDATE)
    GROUP BY TO_CHAR(rs.fecha, 'IW'), s.nombre
)
WHERE rn = 1;
```

```sql
/*
Habitaciones más solicitadas
*/
SELECT semana,
    id AS habitacion_mas_solicitada,
    reservaciones
FROM (
    SELECT TO_CHAR(rh.fecha_entrada, 'IW') AS semana,
           h.id,
           COUNT(*) AS reservaciones,
           ROW_NUMBER() OVER (PARTITION BY TO_CHAR(rh.fecha_entrada, 'IW') ORDER BY COUNT(*) DESC) AS rn
    FROM reservas_habitacion rh
    JOIN habitaciones h ON rh.id_habitacion = h.id
    WHERE rh.fecha_entrada BETWEEN TRUNC(SYSDATE, 'YEAR') - INTERVAL '1' YEAR AND TRUNC(SYSDATE)
    GROUP BY TO_CHAR(rh.fecha_entrada, 'IW'), h.id
)
WHERE rn = 1;
```

```sql
/*
Habitaciones menos solicitadas
*/
SELECT semana,
    id AS habitacion_menos_solicitada,
    reservaciones
FROM (
    SELECT TO_CHAR(rh.fecha_entrada, 'IW') AS semana,
           h.id,
           COUNT(*) AS reservaciones,
           ROW_NUMBER() OVER (PARTITION BY TO_CHAR(rh.fecha_entrada, 'IW') ORDER BY COUNT(*)) AS rn
    FROM reservas_habitacion rh
    JOIN habitaciones h ON rh.id_habitacion = h.id
    WHERE rh.fecha_entrada BETWEEN TRUNC(SYSDATE, 'YEAR') - INTERVAL '1' YEAR AND TRUNC(SYSDATE)
    GROUP BY TO_CHAR(rh.fecha_entrada, 'IW'), h.id
)
WHERE rn = 1;
```

RF12:

```sql
-- Req12

WITH
trimester_stays AS (
  SELECT num_doc, tipo_doc
  FROM check_ins
  GROUP BY num_doc, tipo_doc, EXTRACT(YEAR FROM fecha), TRUNC((EXTRACT(MONTH FROM fecha) - 1) / 3)
  HAVING COUNT(DISTINCT fecha) >= 1
),
expensive_services AS (
  SELECT id_consumidor, tipo_id_consumidor
  FROM reservas_servicio
  WHERE costo > 300000
  GROUP BY id_consumidor, tipo_id_consumidor
  HAVING COUNT(DISTINCT id_servicio) >= 1
),
```

```sql
long_services AS (
  SELECT id_consumidor, tipo_id_consumidor
  FROM reservas_servicio
  JOIN servicios ON reservas_servicio.id_servicio = servicios.id
  WHERE (LOWER(nombre) LIKE '%spa%' OR LOWER(nombre) LIKE '%salones%') AND (TO_DATE(hora_fin, 'HH24:MI:SS') - TO_DATE(hora_inicio, 'HH24:MI:SS')) * 24 > 4
  GROUP BY id_consumidor, tipo_id_consumidor
  HAVING COUNT(DISTINCT id_servicio) >= 1
)
SELECT num_doc, tipo_doc, 'Estancia Trimestral' AS categoria
FROM trimester_stays
UNION
SELECT id_consumidor, tipo_id_consumidor, 'Servicios Costosos'
FROM expensive_services
UNION
SELECT id_consumidor, tipo_id_consumidor, 'Servicios Largo'
FROM long_services;
```

❖ Diseño y cargue masivo de datos.

- Para lograr una carga masiva de datos en SQL se usaron herramientas como Excel y Mockaroo. Con la ayuda de Mockaroo, se generaron numeros de cédulas y nombres aleatorios. Estos fueron puestos en un Excel el cual está dividido por pestañas para los datos que le corresponden a cada una de las tablas. Para llenar cada tabla teniendo en cuenta las restricciones del código (tales como valores únicos o foreign keys correspondientes a otras tablas) se usaron funciones como ÍNDICE, BUSCARV, ALEATORIO, TEXTO y algunas otras que permitieron obtener todos los datos de cada tabla correspondiente. Luego, se usó la función de excel de CONCAT, para unir los datos de cada tabla por medio del insert que se debe poner en SQL.

Algunos de los insert creados fueron los siguientes:

Tabla habitaciones:

A	B	C
ID	TIPO	
H101	1	=CONCAT("Insert into habitaciones (id,tipo) values (";"'";A2;"'";",";B2;");")
H102	3	Insert into habitaciones (id,tipo) values ('H102',3);
H103	1	Insert into habitaciones (id,tipo) values ('H103',1);
H104	4	Insert into habitaciones (id,tipo) values ('H104',4);
H105	3	Insert into habitaciones (id,tipo) values ('H105',3);
H106	5	Insert into habitaciones (id,tipo) values ('H106',5);
H107	1	Insert into habitaciones (id,tipo) values ('H107',1);
H108	1	Insert into habitaciones (id,tipo) values ('H108',1);
H109	3	Insert into habitaciones (id,tipo) values ('H109',3);

Tabla usuarios:

num_doc	tipo_doc	tipo	nombre	email	contrasenia	
969400226	CC	111	Camilo Olaya	Cami303@gmail.com	cam1234	=CONCAT("Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (";A2;",'";B2;"','";C2;"','";D2;"','";E2;"','";F2;"');")
162396326	CC	111	Rocio Mendivelso	Roci476@gmail.com	MRo91829	Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (162396326,'CC',111,'Rocio Mendivelso','Roci476@gmail.com','MRo91829');
995690363	CC	222	Daniela Luque	Dani82@gmail.com	006dal	Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (995690363,'CC',222,'Daniela Luque','Dani82@gmail.com','006dal');
576996261	CC	333	Ivan Perez	Ivan527@gmail.com	para124	Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (576996261,'CC',333,'Ivan Perez','Ivan527@gmail.com','para124');
868102945	CC	333	Fernando Huertas	Fern162@gmail.com	fers34	Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (868102945,'CC',333,'Fernando Huertas','Fern162@gmail.com','fers34');
674428981	CC	333	Sandra Patricia	Sand207@gmail.com	Sandy18201	Insert into usuarios (num_doc,tipo_doc,tipo,nombre,email,contrasenia) values (674428981,'CC',333,'Sandra Patricia','Sand207@gmail.com','Sandy18201');

Tabla tipos_servicio:

ID	NOMBRE	
50	'Bares y Restaurantes'	=CONCAT("Insert into tipos_servicio (id,nombre) values (";A2;",";B2;");")
51	'Gimnasio'	Insert into tipos_servicio (id,nombre) values (51,'Gimnasio');
52	'Internet'	Insert into tipos_servicio (id,nombre) values (52,'Internet');
53	'Lavandería'	Insert into tipos_servicio (id,nombre) values (53,'Lavandería');
54	'Piscina'	Insert into tipos_servicio (id,nombre) values (54,'Piscina');
55	'Prestamo de Utensilios'	Insert into tipos_servicio (id,nombre) values (55,'Prestamo de Utensilios');
56	'Salones'	Insert into tipos_servicio (id,nombre) values (56,'Salones');
57	'Spa'	Insert into tipos_servicio (id,nombre) values (57,'Spa');
59	'Tiendas y Supermercados'	Insert into tipos_servicio (id,nombre) values (59,'Tiendas y Supermercados');

Todas las fórmulas, se realizaron de manera automatizada, con el fin de poderlas arrastrar y optimizar tiempo a la hora de generar la gran cantidad de datos solicitada.

Para agilizar el proceso de la generación aleatoria de datos que cumplan con las restricciones de las relaciones, se optó por la elaboración de un script de Python. El cual generó archivos *.sql* de población de tablas a partir de sentencias de INSERT.

Lo anterior permitió obtener resultados exitosos en la carga masiva realizada:

- Se crearon 150.000 usuarios

- 1.000 habitaciones

- 100.000 reservas

- 300.000 reservas de habitaciones

- 100.000 check-ins

- 100.000 check-outs

+ Las tuplas generadas en las tablas restantes correspoondientes a los servicios.

Obteniendo un resultado final de 751.000 registros.

❖ Construcción de la aplicación, ejecución de pruebas y análisis de resultados