Nombre de la práctica	Manual operadores set, joins y subconsultas			No.	1
Asignatura:	Taller de Base de datos	Carrera:	Ingeniería en Sistemas Computacionales	Duración de la práctica (Hrs)	

NOMBRE DEL ALUMNO: Vanesa Hernández Martínez
GRUPO: 3501

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro):
Actividades en aula de clases y en equipo personal

III. Material empleado:
- Laptop
- Navicat

SUBCONSULTAS

1. Calcular la Longitud Promedio de los Nombres de los Actores

```
SELECT AVG(LENGTH(first_name)) FROM actor;
```

Descripción: Esta consulta calcula la longitud promedio de los nombres de los actores en la tabla actor.
- **Construcción:** Se usa AVG para obtener el promedio y LENGTH para medir la longitud de first_name.
- **Función:** Calcula y devuelve un valor que representa el promedio de la longitud de los nombres.
- **Resultado:** Un número decimal que indica la longitud promedio de los nombres de los actores.

2. Bloque 2: Actores con Nombres de Longitud Mayor a 5.3050

```
SELECT * FROM actor WHERE LENGTH(first_name) > 5.3050;
```

Descripción: Selecciona todos los actores cuyo first_name tenga una longitud mayor a 5.3050 caracteres.
- **Construcción:** LENGTH mide la longitud de cada nombre y WHERE filtra por longitud.
- **Función:** Busca nombres de actores que superen la longitud especificada.
- **Resultado:** Una lista con toda la información de los actores que cumplen con el criterio de longitud.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

3. Bloque 3: Actores con Nombres de Longitud Mayor al Promedio

SELECT * FROM actor WHERE LENGTH(first_name) > (SELECT AVG(LENGTH(first_name)) FROM actor);

Descripción: Selecciona todos los actores cuyos nombres son más largos que el promedio de la longitud de nombres en la tabla actor.

- **Construcción:** Una subconsulta SELECT AVG(...) calcula el promedio y el WHERE filtra los resultados que lo superan.
- **Función:** Compara cada longitud de nombre contra el promedio para filtrar.
- **Resultado:** Una lista de actores con nombres más largos que el promedio general.

4. Bloque 4: Identificar el ID de Categoría para "Comedy"

SELECT category_id FROM category WHERE name="Comedy";

Descripción: Encuentra el category_id de la categoría "Comedy" en la tabla category.

- **Construcción:** WHERE name="Comedy" especifica que se busca solo esa categoría.
- **Función**: Obtiene el ID necesario para la categoría de comedia.
- **Resultado:** Devuelve el valor 5, que corresponde a la categoría "Comedy".

Bloque 5: Obtener Películas de la Categoría Comedy Usando su ID

SELECT film_id FROM film_category WHERE category_id=5;

Descripción: Selecciona los IDs de las películas que pertenecen a la categoría "Comedy" (con category_id = 5).

- **Construcción:** WHERE category_id=5 especifica la categoría de interés.
- **Función:** Filtra y muestra las películas pertenecientes a la categoría Comedia.
- **Resultado:** Una lista de film_id para todas las películas en la categoría Comedia.

SELECT film_id FROM film_category WHERE category_id=(SELECT category_id FROM category WHERE name="Comedy");

Descripción: Alternativa que hace lo mismo pero usando una subconsulta para obtener el category_id de "Comedy".

- **Construcción:** Usa una subconsulta para encontrar el category_id según el nombre de la categoría.
- **Función:** Simplifica la consulta, sin necesidad de saber directamente el category_id.
- **Resultado:** Similar al anterior, devuelve una lista de film_id para películas en la categoría Comedia.

Bloque 6: Actores que Participaron en Películas Específicas

SELECT actor_id FROM film_actor WHERE film_id IN (7,28,99);

Descripción: Busca actores que han participado en las películas con IDs 7, 28 y 99.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

- **Construcción:** Utiliza IN para filtrar los film_id de interés.
- **Función:** Extrae los actores que aparecen en las películas seleccionadas.
- **Resultado:** Lista de actor_id para los actores involucrados en esas películas específicas.

SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id=(SELECT category_id FROM category WHERE name="Comedy"));

Descripción: Alternativa que obtiene actores de películas en la categoría "Comedy" sin necesidad de escribir los film_id manualmente.

- **Construcción:** Usa varias subconsultas para obtener los film_id de comedia y luego filtrar los actores.
- **Función:** Automatiza el proceso de obtención de actores en la categoría de comedia.
- **Resultado:** Una lista de actor_id de actores que han trabajado en películas de la categoría Comedia.

Bloque 7: Nombres de Actores en Películas de la Categoría Comedy

SELECT first_name, last_name FROM actor WHERE actor_id IN (SELECT actor_id FROM film_actor WHERE film_id IN (SELECT film_id FROM film_category WHERE category_id=(SELECT category_id FROM category WHERE name="Comedy")));

Descripción: Selecciona los nombres y apellidos de actores que han participado en películas de la categoría "Comedy".

- **Construcción:** Usa una serie de subconsultas para identificar actores en películas de comedia y luego muestra sus nombres.
- **Función:** Extrae la información personal de actores relacionados con la categoría específica.
- **Resultado:** Lista de first_name y last_name de actores que han participado en películas de comedia.

Bloque 8: Seleccionar inventory_id de la Tabla rental

SELECT inventory_id FROM rental;

Descripción: Obtiene todos los inventory_id de la tabla rental, que representa los artículos de inventario (copias de películas) que han sido alquilados.

- **Construcción**: Una simple selección de la columna inventory_id.
- **Función**: Proporciona una lista de todas las identificaciones de inventario asociadas con algún alquiler.
- **Resultado**: Devuelve los inventory_id que han sido rentados.

Bloque 9: Seleccionar film_id de la Tabla inventory Usando una Subconsulta

SELECT film_id FROM inventory WHERE inventory_id IN (SELECT inventory_id FROM rental);

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

Descripción: Selecciona los film_id de las películas que han sido alquiladas, utilizando los inventory_id previamente obtenidos en el bloque anterior.

- **Construcción**: La subconsulta (SELECT inventory_id FROM rental) obtiene todos los inventarios alquilados, y el WHERE inventory_id IN (...) asegura que solo se seleccionen film_id asociados a esos inventarios.
- **Función**: Identifica las películas (film_id) que han sido alquiladas.
- **Resultado**: Lista de film_id de las películas que han sido alquiladas.

Bloque 10: Seleccionar los Títulos de Películas Que Nunca Han Sido Alquiladas

```
SELECT title FROM film WHERE film_id NOT IN (SELECT film_id FROM inventory WHERE inventory_id IN (SELECT inventory_id FROM rental));
```

Descripción: Encuentra los títulos de películas que nunca han sido alquiladas.

- **Construcción**: La subconsulta anidada (SELECT inventory_id FROM rental) selecciona inventarios rentados, y la consulta intermedia (SELECT film_id FROM inventory WHERE inventory_id IN (...)) obtiene los film_id de películas alquiladas. Finalmente, film_id NOT IN (...) selecciona solo las películas que no están en esa lista.
- **Función**: Identifica los títulos de películas que no están asociadas a ningún alquiler.
- **Resultado**: Una lista de títulos de películas que nunca han sido alquiladas.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

JOINS

Bloque 1: Seleccionar Título de Película y Nombre de la Categoría Usando INNER JOIN

```
SELECT f.title, c.name AS category_name
FROM film AS f
INNER JOIN film_category AS fc ON f.film_id = fc.film_id
INNER JOIN category AS c ON fc.category_id = c.category_id;
```

Descripción: Esta consulta obtiene los títulos de películas junto con sus categorías.

- **Construcción**: Utiliza INNER JOIN para unir tres tablas (film, film_category y category). Primero, une film y film_category en f.film_id = fc.film_id, y luego une film_category y category en fc.category_id = c.category_id.
- **Función**: Muestra el título de cada película y el nombre de su categoría.
- **Resultado**: Lista con el título de cada película junto con la categoría correspondiente.

Bloque 2: Unir Tablas film, film_category y category

```
SELECT * FROM film
INNER JOIN film_category ON film.film_id = film_category.film_id
JOIN category AS c ON film_category.category_id = c.category_id;
```

Descripción: Selecciona todas las columnas de la tabla film junto con las correspondientes de film_category y category.

- **Construcción**: Utiliza INNER JOIN entre film y film_category, y otro INNER JOIN entre film_category y category, similar al bloque anterior.
- **Función**: Proporciona todos los datos de cada película, junto con su categoría.
- **Resultado**: Todas las columnas de la tabla film, además de la información asociada de film_category y category.

Bloque 3: Seleccionar Título de Película y Nombre de Categoría Usando INNER JOIN

```
SELECT film.title, c.name FROM film
INNER JOIN film_category ON film.film_id = film_category.film_id
JOIN category AS c ON film_category.category_id = c.category_id;
```

Descripción: Esta consulta es una variación simplificada del Bloque 1, que solo selecciona el título de la película y el nombre de su categoría.

- **Construcción**: Utiliza INNER JOIN para unir las tablas film, film_category, y category por sus respectivos campos de identificación.
- **Función**: Muestra una lista simplificada con solo el título de cada película y su categoría.
- **Resultado**: Título y categoría de cada película.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

Bloque 4: Título de Película y Cantidad de Veces que ha sido Alquilada

```
SELECT f.title, COUNT(r.rental_id) AS cantidad_rentas
FROM film AS f
LEFT JOIN inventory AS i ON f.film_id = i.film_id
LEFT JOIN rental AS r ON i.inventory_id = r.inventory_id
GROUP BY f.title;
```

Descripción: Esta consulta muestra el título de cada película y la cantidad de veces que ha sido alquilada.

- **Construcción**: Usa LEFT JOIN para unir film con inventory y rental. La primera unión, f.film_id = i.film_id, enlaza películas con su inventario; la segunda, i.inventory_id = r.inventory_id, conecta inventarios con los alquileres. COUNT(r.rental_id) cuenta los alquileres por título de película, agrupados por f.title.
- **Función**: Cuenta las veces que cada película ha sido alquilada, incluso aquellas que no han tenido ningún alquiler (debido a LEFT JOIN).
- **Resultado**: Título de cada película y el número de veces que ha sido alquilada.

Bloque 5: Mostrar Nombres de Actores y Títulos de Películas Usando LEFT JOIN

```
SELECT a.first_name, a.last_name, f.title
FROM actor AS a
LEFT JOIN film_actor AS fa ON a.actor_id = fa.actor_id
LEFT JOIN film AS f ON f.film_id = fa.film_id;
```

Descripción: Esta consulta muestra los nombres de los actores y los títulos de las películas en las que han participado, permitiendo incluir actores que no tienen películas asociadas.

- **Construcción**: Utiliza LEFT JOIN para unir las tablas actor, film_actor, y film. Primero se unen actor y film_actor en a.actor_id = fa.actor_id, y luego film_actor con film en f.film_id = fa.film_id.
- **Función**: Lista el nombre de cada actor junto con el título de las películas en las que ha participado. Si un actor no ha participado en ninguna película, aún aparece en la lista con el campo title en blanco.
- **Resultado**: Lista de actores y sus películas, incluyendo a los actores que no tienen películas.

Bloque 6: Mostrar Nombres de Actores y Títulos de Películas Usando RIGHT JOIN

```
SELECT a.first_name, a.last_name, f.title
FROM actor AS a
RIGHT JOIN film_actor AS fa ON a.actor_id = fa.actor_id
RIGHT JOIN film AS f ON fa.film_id = f.film_id;
```

Descripción: Similar a la consulta anterior, pero en este caso se utiliza RIGHT JOIN para mostrar todos los registros de film y film_actor.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

- **Construcción**: Se emplea RIGHT JOIN entre actor y film_actor en a.actor_id = fa.actor_id, y otro RIGHT JOIN entre film_actor y film en fa.film_id = f.film_id.
- **Función**: Lista cada película y sus actores, incluyendo películas que no tienen actores asociados.
- **Resultado**: Lista de títulos de películas y sus actores, incluyendo las películas sin actores.

Bloque 7: Mostrar Nombre Completo de los Actores y Títulos de Películas Usando JOIN

```
SELECT CONCAT(a.first_name, " " ,a.last_name) AS full_name, f.title
FROM actor AS a
JOIN film_actor AS fa ON a.actor_id = fa.actor_id
JOIN film AS f ON fa.film_id = f.film_id;
```

Descripción: Esta consulta muestra el nombre completo de cada actor (unido en un solo campo) y los títulos de las películas en las que ha participado, pero solo incluye actores que tienen películas asociadas.

- **Construcción**: Utiliza JOIN (o INNER JOIN) para unir las tablas actor, film_actor, y film. Se unen actor y film_actor en a.actor_id = fa.actor_id, y film_actor y film en fa.film_id = f.film_id.
- **Función**: Crea una lista de los nombres completos de los actores y sus películas, excluyendo a los actores sin películas.
- **Resultado**: Lista de actores con sus nombres completos y las películas en las que han participado.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

OPERADORES SET

Bloque 1: Selección de Primeros Nombres en Tablas Diferentes

```
SELECT first_name FROM actor;
SELECT first_name FROM customer;
```

Descripción: Estas dos consultas seleccionan el primer nombre (first_name) de todas las filas en las tablas actor y customer, respectivamente.

- **Construcción**: Cada consulta usa SELECT para obtener la columna first_name de una tabla específica.
- **Función**: Extrae los nombres de las tablas actor y customer por separado.
- **Resultado**: Muestra dos listas de nombres, una con los actores y otra con los clientes.

Bloque 2: Uso de UNION para Unir Listas de Nombres

```
SELECT first_name FROM actor
UNION
SELECT first_name FROM customer;
```

Descripción: Esta consulta combina los resultados de las dos consultas anteriores en una sola lista de nombres únicos.

- **Construcción**: Utiliza el operador UNION para unir los valores de first_name de ambas tablas (actor y customer). UNION elimina duplicados, por lo que si un nombre está en ambas tablas, aparece solo una vez.
- **Función**: Muestra una lista única de nombres provenientes tanto de actor como de customer.
- **Resultado**: Lista de nombres únicos que incluye todos los primeros nombres de los actores y clientes, sin duplicados.

Bloque 3: Encontrar Películas que Nunca Han Sido Alquiladas Usando EXCEPT

```
SELECT title FROM film
EXCEPT
SELECT f.title FROM film AS f
JOIN inventory AS i ON f.film_id = i.film_id
JOIN rental AS r ON i.inventory_id = r.inventory_id;
```

Descripción: Esta consulta encuentra y muestra los títulos de películas que nunca han sido alquiladas.

- **Construcción**:
  - La primera parte SELECT title FROM film selecciona todos los títulos de películas en la tabla film.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

  - La segunda parte SELECT f.title FROM film AS f JOIN inventory AS i ON f.film_id = i.film_id JOIN rental AS r ON i.inventory_id = r.inventory_id selecciona los títulos de las películas que han sido alquiladas, utilizando JOIN para unir las tablas film, inventory, y rental.
  - El operador EXCEPT se utiliza para excluir del primer conjunto las películas que aparecen en el segundo, resultando en los títulos de películas no alquiladas.
- **Función**: Genera una lista de películas que están en la base de datos, pero que no han tenido ningún registro de alquiler.
- **Resultado**: Lista de títulos de películas que no tienen ningún registro en la tabla rental, indicando que nunca fueron alquiladas.

Bloque 4: Ciudades Donde Viven Clientes o Empleados sin Duplicados (Con UNION y JOIN)

```
SELECT DISTINCT c.city
FROM city c
JOIN address a ON c.city_id = a.city_id
JOIN customer cu ON cu.address_id = a.address_id
UNION
SELECT DISTINCT c.city
FROM city c
JOIN address a ON c.city_id = a.city_id
JOIN staff s ON s.address_id = a.address_id;
```

Descripción: Esta consulta obtiene las ciudades en las que viven clientes o empleados, eliminando duplicados.

- **Construcción**:
  - SELECT DISTINCT c.city en ambas partes selecciona las ciudades de forma única (DISTINCT).
  - El primer conjunto de JOIN une city y address para encontrar ciudades donde viven clientes (customer).
  - El segundo conjunto de JOIN une city y address para encontrar ciudades donde viven empleados (staff).
  - UNION combina ambas listas y elimina duplicados.
- **Función**: Obtiene una lista única de ciudades donde residen tanto los clientes como los empleados.
- **Resultado**: Una lista única de ciudades, sin duplicados, donde viven los clientes o empleados.

Bloque 5: Ciudades Donde Viven Clientes o Empleados sin Duplicados (Con Subconsultas)

```
SELECT DISTINCT c.city
FROM city c
JOIN address a ON c.city_id = a.city_id
```


*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

```
JOIN customer cu ON cu.address_id = a.address_id
UNION
SELECT city FROM city WHERE city_id IN (
SELECT city_id FROM address WHERE address_id IN (
SELECT address_id FROM staff));
```

Descripción: Similar al bloque anterior, este bloque también encuentra las ciudades donde viven los clientes o empleados, pero utiliza subconsultas para obtener la información de los empleados.

- **Construcción**:
  - La primera parte usa JOIN para encontrar las ciudades donde viven clientes.
  - La segunda parte utiliza una subconsulta anidada para encontrar ciudades donde viven empleados (staff).
  - UNION combina los resultados, eliminando duplicados.
- **Función**: Devuelve las ciudades donde residen clientes y empleados en una lista única.
- **Resultado**: Lista única de ciudades donde viven clientes o empleados, eliminando duplicados mediante UNION.

Bloque 6: Creación y Uso de una Vista (VIEW)

```
CREATE VIEW prueba AS
SELECT first_name, last_name FROM actor
WHERE YEAR(last_update) > 2020;
```

Descripción: Esta consulta crea una vista llamada prueba que contiene el nombre y apellido de los actores cuya última actualización ocurrió después de 2020.

- **Construcción**:
  - CREATE VIEW crea la vista prueba.
  - SELECT first_name, last_name FROM actor selecciona las columnas first_name y last_name.
  - WHERE YEAR(last_update) > 2020 filtra solo los actores con un valor de last_update posterior a 2020.
- **Función**: Almacena una consulta predefinida como una vista que facilita su uso posterior.
- **Resultado**: Vista prueba que almacena nombres y apellidos de actores actualizados después de 2020.

Bloque 7: Consultas y Modificaciones Usando la Vista

```
SELECT * FROM prueba;
```

Descripción: Esta consulta selecciona todos los registros de la vista prueba.

- **Construcción**: SELECT * FROM prueba recupera todos los datos almacenados en la vista prueba.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

- **Función**: Muestra la lista de nombres y apellidos de actores actualizados después de 2020.
- **Resultado**: Muestra los datos de la vista prueba.

Bloque 8: Inserción de Nuevo Registro en la Tabla actor

```
INSERT actor VALUES (DEFAULT, "Vanesa", "Hernandez", NOW());
```

Descripción: Inserta un nuevo actor en la tabla actor con el nombre "Vanesa Hernandez" y la fecha de actualización actual.

- **Construcción**:
  - INSERT INTO actor especifica la tabla actor.
  - VALUES (DEFAULT, "Vanesa", "Hernandez", NOW()) inserta valores en los campos, con DEFAULT generando un actor_id automáticamente, NOW() colocando la fecha actual en last_update.
- **Función**: Agrega un nuevo actor con el nombre y fecha actual a la tabla actor.
- **Resultado**: Se crea un nuevo registro en la tabla actor para "Vanesa Hernandez" con la fecha y hora actuales.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*

Conclusión

Los ejercicios que involucran **joins, subconsultas y operadores set** demuestran cómo SQL puede aprovechar su gran capacidad para manejar datos de manera eficiente y flexible. Cada tipo de operación permite obtener información específica y detallada al combinar, comparar o filtrar datos de varias tablas.

Mediante los **joins,** logramos combinar datos de diferentes tablas para obtener relaciones directas entre ellas, lo cual es esencial para consultas que requieren información integrada, como los nombres de actores y las películas en las que han participado, o los títulos de las películas y su categoría. Las **subconsultas**, por su parte, facilitan el uso de resultados de una consulta dentro de otra, permitiéndonos realizar análisis complejos sin necesidad de listar datos manualmente, como cuando buscamos títulos de películas que nunca han sido alquiladas. Finalmente, los **operadores set** como UNION y EXCEPT nos ayudan a combinar o diferenciar conjuntos de resultados, siendo especialmente útiles para obtener listas sin duplicados o comparar datos entre varias consultas.

*Cualquier documento no identificado como **Controlado** se considera **COPIA NO CONTROLADA** y no es auditable.*