1. Búsqueda por niveles de relación

Permite encontrar todas las personas que están entre 2 y 4 niveles de amistad con una persona llamada **Miguel**. Es decir, se buscan amigos de amigos, hasta cuatro niveles de profundidad.

1.1. Consulta en Cypher

```
MATCH p = (a:Persona {nombre: "Miguel"})-[:AMIGO_DE*2..4]-(b)
RETURN p
```

2. Contar cuántos nodos tienen una relación específica

Muestra cuántos productos hay por categoría.

2.1. Consulta en Cypher

```
MATCH (p:Producto)-[:CATEGORIZADO_COMO]->(c:Categoria)
RETURN c.nombre, COUNT(p) AS productos
ORDER BY productos DESC
```

3. Encontrar el camino más corto entre dos nodos

Encuentra la ruta más corta entre dos ciudades.

3.1. Consulta en Cypher

```
MATCH (a:Ciudad {nombre: "Bogota"}), (b:Ciudad {nombre: "Medellin"})
MATCH path = shortestPath((a)-[:RUTA*]-(b))
RETURN path
```

4. Crear relaciones condicionalmente (MERGE)

Crea la relación solo si no existe ya, evitando duplicados.

4.1. Consulta en Cypher

```
MATCH (a:Persona {nombre: "Ana"}), (b:Persona {nombre: "Luis"})
MERGE (a)-[:AMIGO_DE]->(b)
```

5. Agrupar y calcular promedios

Calcula el promedio de notas por persona.

5.1. Consulta en Cypher

```
MATCH (p:Persona)-[:REALIZO]->(e:Examen)
RETURN p.nombre, AVG(e.nota) AS promedio
ORDER BY promedio DESC
```

6. Filtrar relaciones con propiedades

Encuentra empleados que han trabajado en un departamento antes de cierto año.

6.1. Consulta en Cypher

```
MATCH (a:Empleado)-[r:TRABAJA_EN]->(d:Departamento)
WHERE r.fechaInicio < date("2022-01-01")
RETURN a.nombre, d.nombre, r.fechaInicio
```

7. Insertar datos condicionalmente

Si el producto no existe, lo crea. Si existe, lo actualiza.

7.1. Consulta en Cypher

```
MERGE (p:Producto {id: "P001"})
ON CREATE SET p.nombre = "Laptop", p.precio = 2500
ON MATCH SET p.ultimaActualizacion = date()
```

8. Consulta con etiquetas múltiples

Encuentra nodos que son tanto Persona como Cliente a la vez.

8.1. Consulta en Cypher

```
MATCH (n:Persona:Cliente)
RETURN n.nombre
```

9. Uso del CASE y condiciones

Devuelve el nombre y la categoría de edad según la edad de la persona

9.1. Consulta en Cypher

```
MATCH (p:Persona)
RETURN p.nombre,
       CASE
         WHEN p.edad < 18 THEN 'Menor de edad'
         WHEN p.edad >= 18 AND p.edad < 65 THEN 'Adulto'
         ELSE 'Adulto mayor'
       END AS categoria_edad
```

10. Agrupar resultados y hacer operaciones intermedias (WITH).

Encuentra personas con más de 3 amigos y devuelve su nombre y el número de amigos.

10.1. Consulta en Cypher

```
MATCH (p:Persona)-[:AMIGO_DE]->(amigo)
WITH p, COUNT(amigo) AS num_amigos
WHERE num_amigos > 3
RETURN p.nombre, num_amigos
```

11. Uso del CASE, COUNT y WITH en una sola consulta

Cuenta la cantidad de amigos de cada persona y categoriza su "nivel social".

11.1. Consulta en Cypher

```
MATCH (p:Persona)-[:AMIGO_DE]->(amigo)
WITH p, COUNT(amigo) AS num_amigos
RETURN p.nombre,
       num_amigos,
       CASE
         WHEN num_amigos = 0 THEN 'Sin amigos'
         WHEN num_amigos < 3 THEN 'Pocos amigos'
         ELSE 'Muchos amigos'
       END AS nivel_social
```

12. Buscar nodos con condición y devolver propiedades

Busca todos los nodos con etiqueta Persona cuya propiedad edad sea mayor a 30 y que vivan en la ciudad "Bogotá".

12.1. Consulta en Cypher

```
MATCH (p:Persona)
WHERE p.edad > 30 AND p.ciudad = 'Bogot '
RETURN p.nombre, p.edad, p.ciudad
```

13. Crear una relación entre nodos existentes

Busca un nodo con etiqueta Autor cuyo nombre sea "Gabriel García Márquezτ un nodo con etiqueta Libro cuyo título sea Çien años de soledad". Luego crea una relación ESCRIBIO desde el autor hacia el libro.

13.1. Consulta en Cypher

```
MATCH (a:Autor {nombre: 'Gabriel Garc a M rquez'}), (l:Libro {titulo: 'Cien a os de soledad'})
CREATE (a)-[:ESCRIBIO]->(l)
RETURN a, l
```

14. Actualizar propiedades de un nodo

Busca un nodo Usuario con el correo 'usuario@example.com' y actualiza dos propiedades: cambia estado a .^{a}ctivoτ pone la fecha actual en ultimaconexion.

14.1. Consulta en Cypher

```
MATCH (u:Usuario {email: 'usuario@example.com'})
SET u.estado = 'activo', u.ultima_conexion = date()
RETURN u
```

15. Eliminar una relación específica

Busca la relación AMIGO$_D$*Eentredospersonasllamadas*"*Ana*"*y*"*Luis*"*yeliminaesarelación*(*sinborrarlosnodos*).

15.1. Consulta en Cypher

```
MATCH (p1:Persona)-[r:AMIGO_DE]-(p2:Persona)
WHERE p1.nombre = 'Ana' AND p2.nombre = 'Luis'
DELETE r
```

16. OPTIONAL MATCH

Busca al usuario Çarlosτ opcionalmente los productos que ha comprado. Si no ha comprado ninguno, igual devuelve el usuario con p.nombre vacío.

16.1. Consulta en Cypher

```
MATCH (u:Usuario {nombre: 'Carlos'})
OPTIONAL MATCH (u)-[:COMPRA]->(p:Producto)
RETURN u.nombre, p.nombre
```

17. UNWIND

Descompone la lista frutas en filas individuales, devolviendo cada fruta por separado.

17.1. Consulta en Cypher

```
WITH ['manzana', 'banana', 'pera'] AS frutas
UNWIND frutas AS fruta
RETURN fruta
```

18. DISTINCT

Devuelve los nombres de amigos, sin repetir ninguno aunque aparezcan varias veces.

18.1. Consulta en Cypher

```
MATCH (p:Persona)-[:AMIGO_DE]->(amigo)
RETURN DISTINCT amigo.nombre
```

19. COUNT(DISTINCT variable)

Cuenta cuántas ciudades diferentes existen entre todos los nodos Persona, sin repetir.

19.1. Consulta en Cypher

```
MATCH (p:Persona)
RETURN COUNT(DISTINCT p.ciudad) AS total_ciudades_distintas
```

20. EJEMPLOS VARIOS:

1.) Buscar los nombres de las personas que han leído libros escritos por autores que se llaman "Mario Vargas Llosa".

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)<-[:ESCRIBIO]-(a:Autor {nombre: 'Mario Vargas Llosa'})
RETURN p.nombre
```

2.) Mostrar los títulos de los libros leídos por personas que viven en 'Medellín'.

```
MATCH (p:Persona {ciudad: 'Medell n '})-[:LEYO]->(l:Libro)
RETURN l.titulo
```

3.) Listar todas las personas y los libros que han leído, ordenado por el nombre de la persona.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)
RETURN p.nombre, l.titulo
ORDER BY p.nombre
```

4.) Encontrar cuántas personas han leído el libro titulado Rayuela".

```
MATCH (:Persona)-[:LEYO]->(l:Libro {titulo: 'Rayuela'})
RETURN COUNT(*) AS lectores_rayuela
```

5.) Devolver los autores que hayan escrito más de un libro.

```
MATCH (a:Autor)-[:ESCRIBIO]->(l:Libro)
WITH a, COUNT(l) AS cantidad
WHERE cantidad > 1
RETURN a.nombre, cantidad
```

6.) Mostrar todas las combinaciones únicas de persona y autor que están conectados a través de un libro leído.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)<-[:ESCRIBIO]-(a:Autor)
RETURN DISTINCT p.nombre, a.nombre
```

7.) Mostrar los libros que no han sido leídos por nadie.

```
MATCH (l:Libro)
WHERE NOT (l)<-[:LEYO]-(:Persona)
RETURN l.titulo
```

8.) Mostrar las ciudades con más de 3 lectores distintos.

```
MATCH (p:Persona)-[:LEYO]->(:Libro)
WITH p.ciudad AS ciudad, COUNT(DISTINCT p) AS lectores
WHERE lectores > 3
RETURN ciudad, lectores
```

9.) Se desea obtener el listado de autores cuyos libros han sido leídos por al menos 3 personas distintas que vivan en ciudades diferentes, y que todos esos libros hayan sido publicados después del año 2000.

```
MATCH (a:Autor)-[:ESCRIBIO]->(l:Libro)<-[:LEYO]-(p:Persona)
WHERE l.a o > 2000
WITH a, l, p
ORDER BY a.nombre
WITH a, l, COLLECT(DISTINCT p) AS lectores, COLLECT(DISTINCT p.ciudad) AS ciudades
WHERE SIZE(lectores) >= 3 AND SIZE(ciudades) >= 3
RETURN
  a.nombre AS autor,
  COUNT(DISTINCT l) AS libros_validos,
  SIZE(lectores) AS total_lectores
```

10.) Obtener las personas que han leído todos los libros escritos por "Gabriel García Márquez".

```
MATCH (a:Autor {nombre: "Gabriel Garc a M rquez"})-[:ESCRIBIO]->(l:Libro)
WITH COLLECT(l) AS librosGGMarquez

MATCH (p:Persona)-[:LEYO]->(l:Libro)
WHERE l IN librosGGMarquez
WITH p, COLLECT(DISTINCT l) AS librosLeidos, librosGGMarquez
WHERE SIZE(librosLeidos) = SIZE(librosGGMarquez)
RETURN DISTINCT p.nombre AS lector_completo
```

11.) Listar los libros leídos por personas de más de una ciudad distinta, y mostrar cuántas ciudades distintas lo han leído.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)
WITH l.titulo AS titulo, COLLECT(DISTINCT p.ciudad) AS ciudades
WHERE SIZE(ciudades) > 1
RETURN titulo, SIZE(ciudades) AS cantidad_ciudades
```

12.) Encontrar las combinaciones de pares de personas que han leído al menos 2 libros en común.

```
MATCH (p1:Persona)-[:LEYO]->(l:Libro)<-[:LEYO]-(p2:Persona)
WHERE p1 <> p2
WITH p1.nombre AS persona1, p2.nombre AS persona2, COLLECT(DISTINCT l.titulo) AS librosCompartidos
WHERE SIZE(librosCompartidos) >= 2
RETURN persona1, persona2, librosCompartidos
```

13.) Calcular el promedio de edad de las personas que han leído libros escritos después del 2010.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)
WHERE l.a o > 2010
RETURN AVG(p.edad) AS promedio_edad
```

14.) Mostrar el top 3 de libros más leídos junto con la cantidad de lectores únicos.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)
WITH l.titulo AS titulo, COUNT(DISTINCT p) AS lectores
ORDER BY lectores DESC
LIMIT 3
RETURN titulo, lectores
```

15.) Encontrar las personas que han leído al menos un libro de cada uno de los autores que han publicado desde 2015.

```
// Paso 1: obtener todos los autores activos desde 2015
MATCH (a:Autor)-[:ESCRIBIO]->(l:Libro)
WHERE l.a o >= 2015
WITH COLLECT(DISTINCT a) AS autoresRelevantes

// Paso 2: por cada persona, revisar cu ntos autores de esa lista ha le do
MATCH (p:Persona)-[:LEYO]->(l:Libro)<-[:ESCRIBIO]-(a:Autor)
WHERE a IN autoresRelevantes
WITH p, COLLECT(DISTINCT a) AS autoresLeidos, autoresRelevantes
WHERE SIZE(autoresLeidos) = SIZE(autoresRelevantes)
RETURN p.nombre AS lector_completo
```

16.) Mostrar todas las personas que hayan leído libros de al menos 5 autores diferentes, pero no más de un libro por autor.

```
MATCH (p:Persona)-[:LEYO]->(l:Libro)<-[:ESCRIBIO]-(a:Autor)
WITH p, a, COUNT(l) AS libros_por_autor
WHERE libros_por_autor = 1
WITH p, COUNT(DISTINCT a) AS autores_unicos
WHERE autores_unicos >= 5
RETURN DISTINCT p.nombre AS persona_exigente
```