BBDD2 - Práctica 4

Parte 1: Introducción

1. ¿Qué tipo de bd es Redis?
   Es una base de datos no relacional, de tipo clave-valor

2. ¿Dónde almacena los datos Redis?
   En la RAM, aunque se pueden persistir en disco

3. ¿Qué tipos de datos posee Redis?
   Strings: Texto (cadenas de caracteres). También acepta operaciones numéricas y de manejo de bits.
   Lists: Listas enlazadas de strings.
   Sets: Conjuntos de strings únicos.
   Sorted Sets: Sets ordenados por un valor asociado a cada string.
   Hashes: Registros de pares clave-valor.
   Geoespaciales: Coordenadas geográficas.
   Bitfields: Enteros de tamaño arbitrario.
   Tipos Probabilísticos: Permiten calcular estadísticas en conjuntos grandes.
   Streams: Estructuras para manejar flujos de datos tipo logs.
   Objetos JSON: JavaScript Object Notation

4. Enuncie las características de Redis
   Velocidad: Redis almacena datos en la memoria RAM, la cual tiene una velocidad de lectura/escritura notablemente mayor que el disco.
   Estructuras de datos: Además de texto, Redis soporta tipos de datos avanzados como listas, conjuntos, registros hash y bitmaps.
   Simplicidad y eficiencia: Tiene un diseño simple, comandos fáciles de aprender y una arquitectura que permite procesar millones de operaciones por segundo en una sola instancia.
   Comunicación Pub/Sub: Redis permite la comunicación entre procesos mediante un sistema de publicación y suscripción, lo que facilita tareas en tiempo real.

5. Comparar Redis con los RDBMS

Característica	**RDBMS**	**Redis**
Estructura	Tablas con filas y columnas basadas en el esquema de SQL	Almacen clave-valor
Transacciones	Soporte completo de transacciones y manejo de rollback.	Soporte de transacciones simples, pero no maneja rollback
Principios ACID	Están diseñados para cumplir los principios ACID.	Atomicidad: Se cumple Consistencia: Asegura la consistencia estructural, pero no

		impone integridad referencial. Aislamiento: No se cumple en las transacciones. Durabilidad: Incluye mecanismos de persistencia, pero no garantizan la durabilidad en datos recientes.
Tipos de datos	Admite tipos simples, como texto, números, fechas, booleanos, etc.	Admite tipos más complejos, como listas, conjuntos, hashes, streams, geoespaciales, etc.
Modo de ejecución	Ejecución multihilo, lo que permite realizar varias operaciones simultáneas	Ejecución monohilo, aunque acepta operaciones asíncronas
Escalabilidad horizontal	Es posible, pero más compleja	Es más óptima, admite mecanismos de sharding y clustering.
Escalabilidad vertical	Es posible, pero presenta limitaciones de costo	Es posible, pero presenta limitaciones por el modo de ejecución
Robustez	Están diseñados para alta consistencia, integridad y recuperación ante fallos	Es menos robusto para persistencia, es más propenso a pérdida de datos por fallos.

6. ¿Redis tiene transacciones?
   Sí, Redis soporta un modelo simple de transacciones, pero no tiene aislamiento total como los RDBMS. Las operaciones se ejecutan en orden, pero no hay rollback si alguna falla.

7. ¿Redis tiene persistencia?
   Sí, Redis permite almacenar los datos en el disco, asegurando la persistencia. Para eso, existen dos técnicas posibles:
   - RDB (Redis DataBase): Almacena una captura (snapshot) del contenido de la base de datos cada cierto periodo de tiempo.
   - AOF (Append Only File): Almacena un log de las escrituras realizadas sobre la base de datos. No se trata de una persistencia real, pero permite recuperar los datos siguiendo las escrituras realizadas.

   También admite el uso de ambas técnicas en la misma base de datos.

8. ¿Cuáles son los principales usos de Redis?
   - Almacenamiento de sesiones
   - Búsquedas por cercanía con la geolocalización
   - Caché
   - Queues (colas de prioridad)

Parte 2: Manejo Simple de Valores String

1. Agregue una clave package con el valor “Bariloche 3 days”

```
127.0.0.1:6379> set package "Bariloche 3 days"
OK
```

2. Agregue una clave user con el valor “Turismo BD2”. Obtenga el valor de la clave user

```
127.0.0.1:6379> set user "Turismo BD2"
OK
127.0.0.1:6379> get user
"Turismo BD2"
```

3. Obtenga todos los valores de claves almacenadas

```
127.0.0.1:6379> keys *
1) "user"
2) "package"
```

Nota: “KEYS” retorna todas las claves que coinciden con el patrón dado. El caracter “*” funciona como comodín.

4. Agregue una clave user con el valor “Cronos Turismo” ¿Cuál es el valor actual de la clave user?

```
127.0.0.1:6379> set user "Cronos Turismo"
OK
127.0.0.1:6379> get user
"Cronos Turismo"
```

5. Concatene “ S.A.” a la clave user. ¿Cuál es el valor actual de la clave user?

```
127.0.0.1:6379> append user " S.A."
(integer) 19
127.0.0.1:6379> get user
"Cronos Turismo S.A."
```

6. Elimine la clave user.

```
127.0.0.1:6379> del user
(integer) 1
```

7. ¿Qué valor retorna si queremos obtener la clave user?

```
127.0.0.1:6379> get user
(nil)
```

(nil) indica que la clave no referencia ningún valor.

Parte 3: Manejo Simple de Valores Numéricos

1. Verificar si existe la clave visits
   No existe

```
127.0.0.1:6379> exists visits
(integer) 0
127.0.0.1:6379> get visits
(nil)
```

2. Agregue una clave visits con el valor 0

```
127.0.0.1:6379> set visits 0
OK
```

3. Incremente en 1 visits ¿Cuál es el valor actual de la clave visits?

```
127.0.0.1:6379> get visits
"0"
127.0.0.1:6379> incr visits
(integer) 1
127.0.0.1:6379> get visits
"1"
```

4. Incremente en 5 visits. ¿Cuál es el valor actual de la clave visits?
   Ahora vale 6

```
127.0.0.1:6379> incrby visits 5
(integer) 6
127.0.0.1:6379> get visits
"6"
```

5. Decremente en 1 visits ¿Cuál es el valor actual de la clave visits?
   Ahora vale 5

```
127.0.0.1:6379> decr visits
(integer) 5
```

6. Incremente en 2 visits. ¿Cuál es el valor actual de la clave visits?

```
127.0.0.1:6379> incrby visits 2
(integer) 7
127.0.0.1:6379> get visits
"7"
```

7. Agregue una clave “value package” con el valor 539789.32

```
127.0.0.1:6379> set "value package" 539789.32
OK
127.0.0.1:6379> get "value package"
"539789.32"
```

8. Incremente en 20000 la clave “value package”. ¿Cuál es el valor actual de “value package”?
   Vale 559789.31999999999999318

```
127.0.0.1:6379> incrbyfloat "value package" 20000
"559789.31999999999999318"
127.0.0.1:6379> get "value package"
"559789.31999999999999318"
```

9. ¿Cuál es el tipo de datos de “value package”, visits y user?
   “value package” y visits son de tipo string y user no está seteado así que no tiene tipo

```
127.0.0.1:6379> type "value package"
string
127.0.0.1:6379> type visits
string
127.0.0.1:6379> type user
none
```

Parte 4: Manejo de Claves

1. Obtenga todas las claves que empiecen con v

```
> KEYS v*
1) "visits"
2) "value package"
```

2. Obtenga todas las claves que contengan la “t”

```
> KEYS *[t]*
1) "visits"
```

3. Obtenga todas las claves que terminen con “age”

```
> KEYS *age
1) "package"
2) "value package"
```

4. Renombre la clave “package” por “bariloche package”

```
> RENAME "package" "bariloche package"
"OK"
```

5. Si quisiera evitar renombrar una clave por otra existente, ¿qué comando utilizaría? Utilizaría el comando **RENAMENX** key newkey, key es la clave a cambiar y newkey la nueva clave por la que se quiere renombrar, esto ocurrirá solo si newkey no existe. En caso de que se pueda efectuar el cambio, lo realizará y devolverá 1, de lo contrario devolverá 0.

```
> RENAMENX visits "value package"
(integer) 0

> RENAMENX visits visitas
(integer) 1
```

6. Elimine todas las claves

```
> FLUSHDB
"OK"

> KEYS *
(empty list or set)
```

Parte 5: Expiración de Claves

1. Agregue una clave agency con el valor “Cronos Tours”

```
127.0.0.1:6379> set agency "Cronos Tours"
OK
```

2. ¿Cúal es el tiempo de vida de la clave agency?

```
127.0.0.1:6379> ttl agency
(integer) -1
```

“-1” indica que la clave no tiene un tiempo de vida asignado (dura indefinidamente).

3. Agregue una expiración de 30 segundos a la clave agency

```
127.0.0.1:6379> expire agency 30
(integer) 1
```

4. ¿Cúal es el tiempo de vida de la clave agency?

```
127.0.0.1:6379> ttl agency
(integer) 27
```

“TTL” retorna el tiempo de vida al momento de ejecutar el comando.

5. Pasados los 30 segundos, ¿Cúal es el tiempo de vida de la clave agency? ¿Qué retorna si pido el valor de agency?

```
127.0.0.1:6379> ttl agency
(integer) -2
```

“-2” indica que la clave no existe en la base de datos.

6. Agreguemos una clave agency con el valor “Cronos Tours” que expire en 20 segundos

```
127.0.0.1:6379> set agency "Cronos Tours" EX 20
OK
```

Parte 6: Listas

1. Inserte una lista llamada pets con el valor dog.

```
127.0.0.1:6379> LPUSH pets dogs
(integer) 1
```

2. ¿Qué sucede si ejecuto el comando get pets? ¿Cómo obtengo los valores de la lista?
No te deja, porque tenés que accederlo con métodos de lista (en este caso lindex)

```
127.0.0.1:6379> get pets
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> lindex pets 0
"dogs"
```

3. Agregue a la lista pets el valor cat a la izquierda.

```
127.0.0.1:6379> lpush pets cat
(integer) 2
127.0.0.1:6379> lindex pets 0
"cat"
```

4. Agregue a la lista pets el valor fish a la derecha.

```
127.0.0.1:6379> rpush pets fish
(integer) 3
127.0.0.1:6379> lindex pets -1
"fish"
```

5. ¿Qué tipo de datos es el valor de pets?
   list

```
127.0.0.1:6379> type pets
list
```

6. Elimine el valor a la izquierda de la lista.

```
127.0.0.1:6379> lpop pets
"cat"
```

```
127.0.0.1:6379> llen pets
(integer) 2
```

7. Elimine el valor a la fecha de la lista.
   (Entiendo que quisieron decir derecha y pusieron fecha)

```
127.0.0.1:6379> rpop pets
"fish"
127.0.0.1:6379> llen pets
(integer) 1
```

8. Agregue una clave “vuelo:ar389” los valores: aep, mdz, brc, nqn y mdq.

```
127.0.0.1:6379> rpush "vuelo:ar389" aep mdz brc nqn mdq
(integer) 5
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "nqn"
5) "mdq"
```

9. Ordene los valores de lista “vuelo:ar389”. ¿Qué sucede si solicito todos los valores de la lista?
   No se guarda la lista ordenada, solo la devuelve.
   Si quisiera guardarla ordenada tendría que usar la STORE option

```
127.0.0.1:6379> sort "vuelo:ar389"
(error) ERR One or more scores can't be converted into double
127.0.0.1:6379> sort "vuelo:ar389" alpha
1) "aep"
2) "brc"
3) "mdq"
4) "mdz"
5) "nqn"
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "nqn"
5) "mdq"
```

10. Inserte el valor “fte” luego de “brc”

```
127.0.0.1:6379> linsert "vuelo:ar389" after brc "fte"
(integer) 6
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "fte"
5) "nqn"
6) "mdq"
```

11. Inserte el valor “ush” antes de “fte”

```
127.0.0.1:6379> linsert "vuelo:ar389" before fte ush
(integer) 7
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "ush"
5) "fte"
6) "nqn"
7) "mdq"
```

12. Modifique el último elemento por “sla”

```
127.0.0.1:6379> lset "vuelo:ar389" -1 "sla"
OK
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "aep"
2) "mdz"
3) "brc"
4) "ush"
5) "fte"
6) "nqn"
7) "sla"
```

13. Obtenga la cantidad de elementos de “vuelo:ar389”

```
127.0.0.1:6379> llen "vuelo:ar389"
(integer) 7
```

14. Obtenga el 3 valor de “vuelo:ar389”
    Entiendo que quisieron decir el tercer valor:

```
127.0.0.1:6379> lindex "vuelo:ar389" 2
"brc"
```

15. Elimine el valor “aep” de “vuelo:ar389”

```
127.0.0.1:6379> lrem "vuelo:ar389" 1 "aep"
(integer) 1
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "mdz"
2) "brc"
3) "ush"
4) "fte"
5) "nqn"
6) "sla"
```

16. Quédese con los valores de las posiciones 3 a 5 de “vuelo:ar389”

```
127.0.0.1:6379> lrange "vuelo:ar389" 3 5
1) "fte"
2) "nqn"
3) "sla"
```

17. Agregue en “vuelo:ar389” el valor “fte”. ¿Cuántas veces aparece?
    Aparece 2 veces

```
127.0.0.1:6379> rpush "vuelo:ar389" "fte"
(integer) 7
127.0.0.1:6379> lrange "vuelo:ar389" 0 -1
1) "mdz"
2) "brc"
3) "ush"
4) "fte"
5) "nqn"
6) "sla"
7) "fte"
```

Parte 7: Conjuntos

1. Agregue un conjunto llamado airports los valores: eze aep nqn mdz mdq ush fte sla aep nqn brc cpc juj aep tuc eqs

```
> SADD airports eze aep nqn mdz mdq ush fte sla aep nqn brc cpc juj aep tuc eqs
(integer) 13
```

2. ¿Cuántos valores tiene el conjunto?
   Tiene 13 elementos.

```
> SCARD airports
(integer) 13
```

3. Liste los valores del conjunto airports.

```
> SMEMBERS airports
1) "eze"
2) "aep"
3) "nqn"
4) "mdz"
5) "mdq"
6) "ush"
7) "fte"
8) "sla"
9) "brc"
10) "cpc"
11) "juj"
12) "tuc"
13) "eqs"
```

4. Quite el valor cpc del conjunto airports.

```
> SREM airports "cpc"
(integer) 1
```

5. Quite un valor aleatorio del conjunto airports.

```
> SPOP airports
"fte"
```

6. ¿Qué cantidad de valores tiene ahora airports?

Tiene 11 elementos.

```
> SCARD airports
(integer) 11
```

7. Compruebe si cpc es miembro del conjunto airports.

```
> SISMEMBER airports "cpc"
(integer) 0
```

El 0 indica que no es miembro, si lo fuese devolveria 1.

8. Mueva los valores sla y juj a un conjunto denominado noa_airports.

```
> SMOVE airports noa_airports "sla"
(integer) 1

> SMOVE airports noa_airports "juj"
(integer) 1
```

9. Retorne la unión de los conjuntos airports y noa_airports. ¿Modifica los conjuntos base?

No, no modifica los conjuntos base.

```
> SUNION airports noa_airports
1) "eze"
2) "aep"
3) "nqn"
4) "mdz"
5) "mdq"
6) "ush"
7) "brc"
8) "tuc"
9) "eqs"
10) "sla"
11) "juj"
```

Los conjuntos después de la unión:

```
> SMEMBERS airports
1) "eze"
2) "aep"
3) "nqn"
4) "mdz"
5) "mdq"
6) "ush"
7) "brc"
8) "tuc"
9) "eqs"
```

```
> SMEMBERS noa_airports
1) "sla"
2) "juj"
```

10. Realice la unión de los conjuntos airports y noa_airports en un conjunto llamado total_airports.

```
> SUNIONSTORE total_airports airports noa_airports
(integer) 11
```

11. Realice la intersección entre los conjuntos total_airports y noa_airports.

```
> SINTER total_airports noa_airports
1) "sla"
2) "juj"
```

12. Realice la diferencia entre los conjuntos total_airports y noa_airports.

```
> SDIFF total_airports noa_airports
1) "eze"
2) "aep"
3) "nqn"
4) "mdz"
5) "mdq"
6) "ush"
7) "brc"
8) "tuc"
9) "eqs"
```

Parte 8: Conjuntos Ordenados

1. En un conjunto ordenado llamado passengers agregue los siguientes datos: 2.5 federico 4 alejandra 3 julian 1 ivan 2 andrea 2 luciana 2.4 natalia

```
127.0.0.1:6379> zadd passengers 2.5 federico 4 alejandra 3 julian 1 ivan 2 andrea 2 luciana 2.4 natalia
(integer) 7
```

2. Obtenga los valores del conjunto passengers

“ZRANGE” retorna un rango de elementos entre los rankings dados. Se utiliza 0 y -1 para indicar que retorne todo el conjunto.

```
127.0.0.1:6379> zrange passengers 0 -1
1) "ivan"
2) "andrea"
3) "luciana"
4) "natalia"
5) "federico"
6) "julian"
7) "alejandra"
```

3. Actualice el score de luciana a 2.7

```
127.0.0.1:6379> zadd passengers 2.7 luciana
(integer) 0
```

Como el valor ya existe, sobrescribe el score asociado.

4. Agregue al conjunto passengers a silvia con score 5.1

```
127.0.0.1:6379> zadd passengers 5.1 silvia
(integer) 1
```

5. Incremente en 2 el score de alejandra en el conjunto passengers.

```
127.0.0.1:6379> zincrby passengers 2 alejandra
"6"
```

6. Obtenga los valores del conjunto passengers con sus scores.

```
127.0.0.1:6379> zrange passengers 0 -1 WITHSCORES
 1) "ivan"
 2) "1"
 3) "andrea"
 4) "2"
 5) "natalia"
 6) "2.4"
 7) "federico"
 8) "2.5"
 9) "luciana"
10) "2.7"
11) "julian"
12) "3"
13) "silvia"
14) "5.1"
15) "alejandra"
16) "6"
```

7. Obtenga los valores del conjunto passengers con sus scores en orden inverso.

```
127.0.0.1:6379> zrange passengers 0 -1 WITHSCORES REV
 1) "alejandra"
 2) "6"
 3) "silvia"
 4) "5.1"
 5) "julian"
 6) "3"
 7) "luciana"
 8) "2.7"
 9) "federico"
10) "2.5"
11) "natalia"
12) "2.4"
13) "andrea"
14) "2"
15) "ivan"
16) "1"
```

8. Obtenga la cantidad de elementos del conjunto passengers.
   "ZCOUNT" retorna la cantidad de elementos en un rango de scores dado. Se utiliza -inf e inf para que retorne el total de elementos.

```
127.0.0.1:6379> zcount passengers -inf inf
(integer) 8
```

9. Obtenga la cantidad de elementos que tienen scores entre 2 y 3.

```
127.0.0.1:6379> zcount passengers 2 3
(integer) 5
```

10. Obtenga el ranking de Julian en el conjunto passengers.

```
127.0.0.1:6379> zrank passengers julian
(integer) 5
```

11. Obtenga el score de Andrea en el conjunto passengers.

```
127.0.0.1:6379> zscore passengers andrea
"2"
```

12. Extraiga el valor de menor score del conjunto passengers.

```
127.0.0.1:6379> zpopmin passengers
1) "ivan"
2) "1"
```

13. Extraiga el valor de mayor score del conjunto passengers.

```
127.0.0.1:6379> zpopmax passengers
1) "alejandra"
2) "6"
```

14. Elimine del conjunto passengers al valor silvia.

```
127.0.0.1:6379> zrem passengers silvia
(integer) 1
```

Parte 9: Hashes

1. Agregue a un hash llamado user:cronos los valores:
   “razon social” “cronos s.a”, domicilio “47 236 La Plata”, “teléfono” 2215556677

```
127.0.0.1:6379> hset user:cronos "razon social" "cronos s.a" domicilio "47 236 La Plata" "teléfono " 2215556677
(integer) 3
```

2. Agregue el mail info@cronos.com.ar a user:cronos

```
127.0.0.1:6379> hset "user:cronos" mail "info@cronos.com.ar"
(integer) 1
```

3. Obtenga todos los valores de user:cronos

```
127.0.0.1:6379> hvals "user:cronos"
1) "cronos s.a"
2) "47 236 La Plata"
3) "2215556677"
4) "info@cronos.com.ar"
```

4. Obtenga el mail de user:cronos

```
127.0.0.1:6379> hget "user:cronos" "mail"
"info@cronos.com.ar"
```

5. Elimine el teléfono de user:cronos

```
127.0.0.1:6379> hdel "user:cronos" "teléfono "
(integer) 1
```

6. Obtenga la cantidad de campos de user:cronos

```
127.0.0.1:6379> hlen "user:cronos"
(integer) 3
```

7. Obtenga las claves de los campos de user:cronos

```
127.0.0.1:6379> hkeys "user:cronos"
1) "razon social"
2) "domicilio"
3) "mail"
```

8. Determine si existe el campo cuil en user:cronos
   No existe

```
127.0.0.1:6379> hexists "user:cronos" cuil
(integer) 0
```

9. Obtenga todos los valores de los campos de user:cronos

```
127.0.0.1:6379> hvals "user:cronos"
1) "cronos s.a"
2) "47 236 La Plata"
3) "info@cronos.com.ar"
```

10. Obtenga la longitud del campo mail de user:cronos

```
127.0.0.1:6379> hstrlen "user:cronos" mail
(integer) 18
```

Parte 10: Geospatial

1. Agregue en un conjunto denominado cities las siguientes localidades:

Buenos Aires	-34.61315, -58.37723
Córdoba	-31.4135, -64.18105
Rosario	-32.94682, -60.63932
Mendoza	-32.89084, -68.82717
San Miguel de Tucumán	-26.82414, -65.2226
La Plata	-34.92145, -57.95453
Mar del Plata	-38.00042, -57.5562
Salta	-24.7859, -65.41166
Santa Fe	-31.64881, -60.70868
San Juan	-31.5375, -68.53639
Resistencia	-27.46056, -58.98389
Santiago del Estero	-27.79511, -64.26149
Posadas	-27.36708, -55.89608
San Salvador de Jujuy	-24.19457, -65.29712
Bahía Blanca	-38.71959, -62.27243
Paraná	-31.73271, -60.52897

```
> GEOADD cities -34.61315 -58.37723 "Buenos Aires" -31.4135 -64.18105 "Córdoba" -32.94682 -60.63932 "Rosario" -32.89084 -68.82717 "Mendoza" -26.82414 -65.2226
ucumán" -34.92145 -57.95453 "La Plata" -38.00042 -57.5562 "Mar del Plata" -24.7859 -65.41166 "Salta" -31.64881 -60.70868 "Santa Fe" -31.5375 -68.53639 "San Ju
8.98389 "Resistencia" -27.79511 -64.26149 "Santiago del Estero" -27.36708 -55.89608 "Posadas" -24.19457 -65.29712 "San Salvador de Jujuy" -38.71959 62.27243 "
1.73271 -60.52897 "Paraná"
```

2. Obtenga los valores del conjunto cities

```
> ZRANGE cities 0 -1
1) "Mendoza"
2) "San Juan"
3) "C\xc3\xb3rdoba"
4) "San Miguel de Tucum\xc3\xa1n"
5) "Santiago del Estero"
6) "Salta"
7) "San Salvador de Jujuy"
8) "Mar del Plata"
9) "Buenos Aires"
10) "La Plata"
11) "Rosario"
12) "Santa Fe"
13) "Paran\xc3\xa1"
14) "Resistencia"
15) "Posadas"
16) "Bah\xc3\xada Blanca"
```

3. Obtenga las coordenadas de Santa Fe

```
> GEOPOS cities "Santa Fe"
1) 1) "-31.648808419704437 26"
   2) "-60.70867938681186615"
```

4. Obtenga la distancia en km entre Buenos Aires y Córdoba

```
> GEODIST cities "Buenos Aires" "Córdoba" KM
"667.5963"
```

5. Obtenga las ciudades que están en un radio de 100 km de la coordenada -27.37 -55.9 con su distancia.

```
> GEOSEARCH cities FROMLONLAT -27.37 -55.9 BYRADIUS 200 KM WITHDIST ASC
1) 1) "Posadas"
   2) "0.4725"
```

6. Obtenga las ciudades que están a menos de 700 km de Córdoba.

```
> GEOSEARCH cities FROMMEMBER "Córdoba" BYRADIUS 700 KM
1) "Mendoza"
2) "San Juan"
3) "C\xc3\xb3rdoba"
4) "San Miguel de Tucum\xc3\xa1n"
5) "Santiago del Estero"
6) "Salta"
7) "San Salvador de Jujuy"
8) "Buenos Aires"
9) "Rosario"
10) "Santa Fe"
11) "Paran\xc3\xa1"
12) "Resistencia"
```

Para la entrega se solicita que cada grupo describa brevemente la instalación de Redis utilizada:

(Augusto) Usé la versión de linux que se instala de manera muy sencilla con sudo apt install redis (gracias linux) y se ejecuta con redis-cli. Es una interfaz de comandos similar a la de MySQL.
Me gusta mucho que te va tirando los parámetros que espera recibir el comando

```
127.0.0.1:6379> linsert key BEFORE|AFTER pivot element
```

(Tomás) Al no haber una versión de windows, usé docker para ejecutar redis-cli. Funciona igual que la versión de linux.

(Sofía) Como utilice la versión online Try Redis no tuve que realizar una instalación local. Funciona de forma muy parecida a redis-cli, solo que desde el navegador.