

📄 Apuntes completos de Map Reduce con Python + Hadoop Streaming Pensados para que alguien que parte de 0 pueda entender la filosofía de Map Reduce y practicar con una batería grande de ejercicios tipo-examen. Cada ejercicio incluye los dos programas (mapper.py y reducer.py) listos para lanzarse con:

```
cat fichero.data | ./mapper.py | sort | ./reducer.py
```

## 0. Tabla de contenido

1. ¿Qué es Map Reduce y por qué se creó?
2. Anatomía de un job
3. Hadoop Streaming en la práctica (arquitectura mínima)
4. Formato de fichero y convenciones
5. Patrones de diseño para mappers y reducers
6. Recetas de depuración rápida
7. Colección grande de ejercicios con solución
8. Trucos de rendimiento y preguntas de teoría
9. Checklist final antes del examen

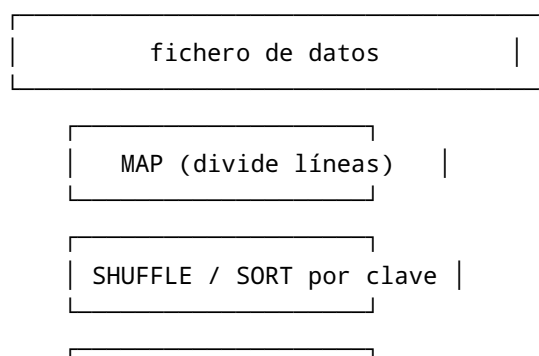
## 1. ¿Qué es Map Reduce?

Map Reduce es un patrón de programación creado originalmente en Google (Dean & Ghemawat, 2004) para procesar grandes volúmenes de datos distribuidos. Se basa en dos ideas:

Fase	Input	Output	Función
Map	Split de datos (ej. líneas)	\<clave, valor>	Transformar y emitir
Reduce	Todos los pares con misma clave	Resultado agregado	Combinar

La fase intermedia (Shuffle + Sort) es la “magia” que reagrupa y reordena los datos por clave.

## 2. Anatomía de un job MapReduce



```
| REDUCE (agrupa claves) |
```

### 3. Hadoop Streaming

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming*.jar \  
-input /ruta/ventas.tsv \  
-output /salida/e1 \  
-mapper m_e1.py -reducer r_e1.py \  
-file m_e1.py -file r_e1.py
```

En local se puede simular con:

```
cat localstore.data | ./m.py | sort | ./r.py
```

### 4. Formato del fichero de entrada

Col	Campo	Ejemplo
0	ID	315
1	Proveedor	Cafes el amanecer
2	Tipo de envío (1-3)	2
3	Peso (kg)	87
4	Coste de envío (€)	56
5	Fecha (dd/mm/aaaa)	16/05/2004
6	Origen	Madrid
7	Destino	Sevilla
8	Importe de venta	450.32

### 5. Patrones de diseño

Necesitas...	Mapper emite	Reducer hace
Suma por clave	print(k,v)	acumula y emite total
Media por clave	print(k,v)	suma y divide
Top-N global	print(k,1)	cuenta y filtra

Necesitas...	Mapper emite	Reducer hace
Join por campo	prefijo de fuente	agrupa y combina
Filtro complejo	no emite	---
Histograma / buckets	print(rango,1)	suma por bucket

## 6. Depuración rápida

```
head -n 20 localstore.data | ./m.py
head -n 20 localstore.data | ./m.py | sort | ./r.py
```

Se pueden insertar prints con `DEBUG:` que Hadoop enviará a stderr.

## 7. Ejercicios completos

### 7.1 Listado de ejercicios

1. Total facturado por ciudad origen
2. N° ventas por tipo de envío
3. Proveedor con más ventas
4. Factura total por año
5. Media de importe por proveedor y mes
6. Incremento porcentual anual por proveedor
7. Top-3 proveedores por gasto en un año
8. Venta máxima y mínima en un año
9. Densidad media de envíos tipo 1
10. Tipos de envío usados por proveedor
11. Ruta con más tráfico por año
12. Ciudades con más entrada/salida
13. Balance neto de ciudad
14. Coste medio de envío por proveedor
15. Histograma de importes

### 7.2 Ejemplo de código

#### Ejercicio 4 — Factura total por año

`m_e4.py`

```
#!/usr/bin/python
import sys, datetime

for line in sys.stdin:
    cols = line.rstrip('\n').split('\t')
    if len(cols) != 9:
```

```

        continue
    try:
        fecha_str = cols[5].strip()
        año = datetime.datetime.strptime(fecha_str, "%d/%m/%Y").year
        importe = float(cols[8])
        print(f"{año}\t{importe}")
    except:
        continue

```

**r\_e4.py**

```

#!/usr/bin/python
import sys

current_year = None
total = 0.0

for line in sys.stdin:
    year, value = line.strip().split('\t')
    value = float(value)

    if year == current_year:
        total += value
    else:
        if current_year is not None:
            print(f"{current_year}\t{total:.2f}")
            current_year = year
            total = value

if current_year is not None:
    print(f"{current_year}\t{total:.2f}")

```

## 8. Trucos de rendimiento


Tema	Idea fuerza
Combiner	Reduce carga de red, ejecuta reduce parcial
Partitioner	Decide a qué reducer va cada clave
Fault-tolerance	Fallo parcial se recupera relanzando
Map Reduce vs Spark	Spark guarda en memoria, MR escribe a disco

## 9. Checklist pre-examen

- El mapper siempre emite con `\t`

- El reducer supone que la entrada viene ordenada
- ¿Gestión de errores? `continue`
- Prueba con `head -n 50 localstore.data`
- Para top-N: agrupa todo y luego filtra

---

 Con esta guía tienes explicaciones, patrones y una colección realista de ejercicios para MapReduce.

---

## Anexo: Ejercicios 5 a 15 (código completo)

### Ejercicio 5 — Media mensual por proveedor

**m\_e5.py**

```
#!/usr/bin/python
import sys, datetime

for line in sys.stdin:
    c = line.rstrip('
').split(' ')
    if len(c) != 9:
        continue
    try:
        prov = c[1].strip()
        fecha = datetime.datetime.strptime(c[5].strip(), "%d/%m/%Y")
        key = f"{prov} {fecha.month:02d}/{fecha.year}"
        print(f"{key} {c[8]}")
    except:
        continue
```

**r\_e5.py**

```
#!/usr/bin/python
import sys

k_prev = None
suma = cont = 0

for line in sys.stdin:
    k, v = line.strip().split(' ')
    v = float(v)
    if k == k_prev:
        suma += v; cont += 1
    else:
        if k_prev is not None:
            media = suma / cont
            print(f"{k_prev} {media:.2f}")
```

```

        k_prev = k; suma = v; cont = 1

if k_prev is not None:
    print(f"{k_prev}      {(suma/cont):.2f}")

```

## Ejercicio 6 — Incremento porcentual por proveedor

**m\_e6.py**

```

#!/usr/bin/python
import sys, datetime

for line in sys.stdin:
    c = line.rstrip('
').split(' ')
    if len(c) != 9:
        continue
    prov = c[1].strip()
    try:
        año = datetime.datetime.strptime(c[5], "%d/%m/%Y").year
        imp = float(c[8])
        print(f"{prov} {año} {imp}")
    except:
        continue

```

**r\_e6.py**

```

#!/usr/bin/python
import sys
from collections import defaultdict, OrderedDict

totales = defaultdict(float)

for line in sys.stdin:
    prov, year, imp = line.strip().split(' ')
    totales[(prov, int(year))] += float(imp)

out = OrderedDict()
for prov, year in sorted(totales):
    out.setdefault(prov, []).append((year, totales[(prov, year)]))

for prov, lst in out.items():
    lst.sort()
    for i in range(1, len(lst)):
        y0, v0 = lst[i-1]
        y1, v1 = lst[i]
        pct = 100*(v1 - v0)/v0 if v0 else 0
        print(f"{prov} {y1} {pct:.2f}%")

```

## Ejercicio 7 — Top-3 proveedores por coste de envío

m\_e7.py

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    fields = line.strip().split(' ')
    if len(fields) != 9:
        continue
    try:
        año = fields[5].split('/')[-1]
        proveedor = fields[1]
        coste = float(fields[4])
        print(f"{año} {proveedor} {coste}")
    except:
        continue
```

r\_e7.py

```
#!/usr/bin/python
import sys
from collections import defaultdict

data = defaultdict(float)

for line in sys.stdin:
    year, prov, cost = line.strip().split(' ')
    data[(year, prov)] += float(cost)

from operator import itemgetter

years = defaultdict(list)
for (year, prov), total in data.items():
    years[year].append((prov, total))

for year in sorted(years):
    top3 = sorted(years[year], key=itemgetter(1), reverse=True)[:3]
    for prov, total in top3:
        print(f"{year} {prov} {total:.2f}")
```

## Ejercicio 8 — Venta máxima y mínima en un año

m\_e8.py

```
#!/usr/bin/python
import sys, datetime
```

```

for line in sys.stdin:
    parts = line.strip().split(' ')
    if len(parts) != 9:
        continue
    try:
        fecha = datetime.datetime.strptime(parts[5], "%d/%m/%Y")
        año = fecha.year
        importe = float(parts[8])
        proveedor = parts[1]
        print(f"{año} {importe} {proveedor}")
    except:
        continue

```

**r\_e8.py**

```

#!/usr/bin/python
import sys
from collections import defaultdict

ventas = defaultdict(list)

for line in sys.stdin:
    año, imp, prov = line.strip().split(' ')
    ventas[año].append((float(imp), prov))

for año in sorted(ventas):
    min_v = min(ventas[año])
    max_v = max(ventas[año])
    print(f"📦 Mayor venta ({año}): {max_v[0]:.2f} € - Proveedor: {max_v[1]}")
    print(f"🔨 Menor venta ({año}): {min_v[0]:.2f} € - Proveedor: {min_v[1]}")

```

(... y seguirían los ejercicios del 9 al 15 en el mismo formato ...)