Informe de Análisis del Mercado Inmobiliario en Barcelona

Predicción de la Necesidad de Reformas en Propiedades

Proyecto Datos Masivos

Fecha: Diciembre 2024
Elaborado por: Lidia Moreno Marín y Amparo Gálvez Vilar
Fuente de Datos: Kaggle - Idealista Barcelona Raw Scraped Data

CONTENIDO

INTRODUCCIÓN

El presente informe tiene como objetivo ofrecer un análisis detallado del mercado inmobiliario en Barcelona, basado en una base de datos que contiene información de 10,107 propiedades extraídas del sitio web inmobiliario Idealista. Los datos se dividen en dos tablas principales: una con la información original de las propiedades (Raw_data) y otra con los datos estructurados y las características específicas de las mismas (Structured_dataw_description). Esta última incluye atributos clave como el tipo de propiedad, precio, superficie construida, año de construcción, número de habitaciones y baños, así como características adicionales como la presencia de ascensor, terraza, aire acondicionado, calefacción y eficiencia energética, entre otras.

El enfoque principal de este análisis es la predicción de la necesidad de reformas en las propiedades, representada como una variable binaria (Verdadero/Falso). El modelo busca identificar aquellas propiedades que requieren renovación, basándose en atributos como el año de construcción, la superficie, el precio y el número de habitaciones. Este análisis no solo proporciona información valiosa para compradores e inversores interesados en propiedades con potencial de mejora, sino que también facilita la toma de decisiones informadas, ahorrando tiempo y esfuerzo en la búsqueda de inmuebles con oportunidades de revalorización.

A lo largo de este informe, se presentarán patrones y tendencias clave relacionadas con la ubicación, tamaño, estado y amenidades de las propiedades en Barcelona, utilizando los datos disponibles en el conjunto estructurado. Este análisis permitirá generar conclusiones relevantes que orienten tanto a la toma de decisiones en el mercado inmobiliario como a futuras investigaciones sobre las dinámicas de la renovación de propiedades en la ciudad.

PRÁCTICA 4: Metodología mediante MapReduce

En esta práctica se utiliza un código con el enfoque de MapReduce para procesar y analizar datos de propiedades inmobiliarias. Se estructura en clases de Mapper y Reducer para cada operación, que permiten transformar los datos y luego reducirlos a resultados agregados. Aquí te doy una breve explicación por cada operación:

Relación entre tamaño en metros cuadrados y necesidad de reformas: El Mapper extrae el tamaño en metros cuadrados y la necesidad de reformas para cada propiedad. El Reducer agrupa los tamaños por si necesitan reformas o no. Figura 1

Promedio de precios de propiedades que necesitan reformas: El Mapper filtra las propiedades que requieren reformas y recoge sus precios. El Reducer calcula el precio promedio de estas propiedades. Figura 2

Distribución de propiedades que necesitan reformas por ciudad: El Mapper extrae la ciudad de cada propiedad que requiere reformas y cuenta cuántas propiedades hay por ciudad. El Reducer suma las propiedades por cada ciudad. Figura 3

Relación entre número de dormitorios y necesidad de reformas: El Mapper extrae el número de dormitorios y la necesidad de reformas. El Reducer agrupa los resultados según si la propiedad necesita reformas o no. Figura 4

Propiedades con mayor antigüedad que necesitan reformas: El Mapper calcula la antigüedad de las propiedades que necesitan reformas, y el Reducer ordena las propiedades por su antigüedad de mayor a menor. Figura 5

Promedio de antigüedad de propiedades que necesitan reformas por barrio: El Mapper calcula la antigüedad de las propiedades por barrio y que necesitan reformas. El Reducer calcula la media de antigüedad por barrio. Figura 6

El JobRunner es la clase principal que ejecuta cada operación, lee los datos desde el archivo CSV, aplica el Mapper para transformar los datos y luego pasa los resultados al Reducer para generar la salida final, en todas las operaciones realizas se utiliza el mismo. Figura 7

Del análisis realizado, se observa una tendencia clara en la relación entre diferentes variables y la necesidad de reformas en las propiedades. Las propiedades más antiguas y grandes tienden a requerir renovaciones con mayor frecuencia, lo cual es esperable dado que las estructuras más antiguas suelen necesitar mantenimiento más frecuente. Este patrón es especialmente relevante en áreas urbanas como Barcelona, donde se concentra la mayor parte de las propiedades que necesitan reformas.

En términos de precio, las propiedades que requieren reformas tienen un promedio elevado, lo que podría explicarse por su ubicación privilegiada o por el hecho de que suelen ser propiedades más grandes. Además, estas propiedades tienden a tener más dormitorios que las que no necesitan reformas, lo que refuerza la idea de que las propiedades más grandes presentan mayores desafíos de mantenimiento.

A nivel geográfico, hay una distribución desigual de las propiedades que necesitan reformas, con una concentración significativa en barrios históricos como El Gòtic, El Raval y Sant Pere - Santa Caterina i la Ribera, donde los edificios tienen mayor antigüedad. Este factor sugiere que las inversiones en renovaciones podrían ser más rentables en estas zonas debido a su atractivo histórico y cultural.

Finalmente, las diferencias en la antigüedad promedio por barrio confirman que las propiedades ubicadas en áreas históricas son las que presentan mayores necesidades de renovación. Esto resalta la importancia de considerar tanto la antigüedad como el tamaño y la ubicación al evaluar oportunidades de inversión o compra.

Conclusiones Principales

Propiedades antiguas y grandes: Existe una correlación entre la antigüedad y el tamaño de las propiedades con la necesidad de reformas. Las propiedades más antiguas y grandes son más propensas a requerir renovaciones.

Precio y ubicación: Las propiedades que necesitan reformas tienen precios elevados, indicando su potencial atractivo en áreas urbanas céntricas o históricas donde, tras reformas, podrían revalorizarse significativamente.

Distribución geográfica: Barcelona concentra la mayor parte de las propiedades que necesitan reformas, particularmente en barrios históricos como El Gòtic y El Raval, lo que representa oportunidades para proyectos de restauración y conservación.

Dormitorios: Las propiedades con más dormitorios tienden a necesitar más reformas, probablemente porque son estructuras más grandes y complejas de mantener.

Barrios históricos: Barrios con alta antigüedad promedio, como El Gòtic y Sant Pere - Santa Caterina i la Ribera, son zonas prioritarias para identificar propiedades con potencial de renovación.

Estrategia para inversores: Los resultados sugieren que los inversores interesados en propiedades para renovación deberían priorizar áreas céntricas e históricas, enfocándose en propiedades con mayor antigüedad y tamaño, que tienen un alto potencial de revalorización tras las reformas.

PRÁCTICA 5: Metodología mediante RDD

En esta parte del proyecto, se emplean RDDs de Spark para llevar a cabo diversas operaciones analíticas sobre el conjunto de datos. A continuación, se detalla brevemente la funcionalidad de cada sección del código:

Inicialización de SparkContext y carga de datos: Se inicializa un SparkContext en modo local para realizar el procesamiento en paralelo. El archivo CSV es cargado en un RDD, donde cada línea del archivo es un elemento del RDD. Luego, se procesa el archivo para dividir las líneas en listas de elementos y se elimina el encabezado del archivo. Figura 8

Distribución de tipos de propiedades: Usando RDDs, se mapea el tipo de propiedad y se aplica reduceByKey para contar cuántas propiedades existen de cada tipo. Este es un ejemplo de una operación de reducción que agrupa y cuenta elementos por clave. Figura 9

Número de propiedades que necesitan reformas por ciudad: Se filtran las propiedades que necesitan reformas utilizando filter sobre el RDD original, luego se mapea para extraer la ciudad de la propiedad y se vuelve a utilizar reduceByKey para contar cuántas propiedades de cada ciudad requieren reformas. Figura 10

Promedio de precios por tipo de propiedad: Primero, se mapean los precios de las propiedades y se agrupan por tipo de propiedad. Luego, se utiliza reduceByKey para calcular la suma total de los precios y el número de propiedades por tipo, lo que permite calcular el promedio de precios por tipo de propiedad. Figura 11

Porcentaje de propiedades que necesitan reformas por tipo de propiedad: Primero, se cuentan las propiedades totales y las que necesitan reformas por tipo utilizando map y reduceByKey. Después, se combinan ambos RDDs con join para calcular el porcentaje de propiedades que necesitan reformas dentro de cada tipo. Figura 12

Número de propiedades que necesitan reformas por número de dormitorios: Similar al análisis de la ciudad, pero esta vez se filtran las propiedades por el número de dormitorios. Se cuenta cuántas propiedades con un número específico de dormitorios necesitan reformas utilizando reduceByKey. Figura 13

Estadísticas por año de construcción (media y desviación estándar de precios): Se crea un RDD clave-valor donde la clave es el año de construcción (índice 10) y el valor es el precio de la propiedad. Luego, se filtran las propiedades con precios válidos y se agrupan por año utilizando reduceByKey. Finalmente, se calcula la media y la desviación estándar. Figura 14

Promedio del tamaño en metros cuadrados de propiedades que necesitan reformas: Se filtran las propiedades que necesitan reformas y se calcula el tamaño promedio en metros cuadrados utilizando la operación mean() de los RDDs. Figura 15

Del análisis realizado, se observa una tendencia clara en la relación entre la antigüedad y el tamaño de las propiedades con la necesidad de reformas. Las propiedades más antiguas y grandes son las que tienden a requerir renovaciones más frecuentemente. Esto es esperado, dado que las estructuras más antiguas suelen enfrentar un desgaste mayor, lo que incrementa su necesidad de mantenimiento.

En cuanto a la distribución geográfica, se identifica una mayor concentración de propiedades que requieren reformas en áreas específicas de la ciudad. Las zonas más afectadas incluyen barrios como El Gòtic, El Raval, y Sant Pere - Santa Caterina i la Ribera, donde se encuentran los edificios más antiguos de la ciudad. Estos barrios históricos presentan una gran cantidad de propiedades que requieren intervenciones para su conservación o renovación.

Respecto al precio de las propiedades que necesitan reformas, se observa que el promedio de precios es significativamente más alto en comparación con las propiedades que no requieren reformas. Esto podría explicarse por la ubicación privilegiada de muchas de estas propiedades, que a menudo se encuentran en áreas céntricas o de gran valor cultural. Además, estas propiedades suelen ser más grandes, lo que también contribuye al precio elevado. El hecho de que estas propiedades sean más grandes también podría implicar que tienen un mayor número de dormitorios, lo que aumenta su complejidad y el costo de las reformas necesarias.

Por otro lado, las propiedades que requieren reformas tienden a ser más grandes en términos de metros cuadrados y tienen una mayor cantidad de dormitorios que las propiedades que no necesitan renovación. Esto sugiere que las propiedades más grandes, al ser más complejas de mantener, presentan mayores desafíos y, por ende, una mayor probabilidad de necesitar reformas.

Conclusiones Principales

Antigüedad y tamaño de las propiedades: Existe una clara relación entre la antigüedad y el tamaño de las propiedades con la necesidad de reformas. Las propiedades más antiguas y grandes requieren más renovaciones debido al desgaste de las estructuras y su mayor complejidad.

Ubicación geográfica: Las zonas más afectadas por la necesidad de reformas están concentradas en barrios históricos de la ciudad, como El Gòtic y El Raval, que albergan propiedades con una mayor antigüedad y demanda de renovación.

Precio elevado: Las propiedades que requieren reformas presentan precios elevados, lo que podría explicarse por su ubicación privilegiada y su tamaño. Estas propiedades tienen un alto potencial de revalorización una vez renovadas, lo que las convierte en una oportunidad atractiva para los inversores.

Distribución de los dormitorios: Las propiedades con más dormitorios suelen ser más grandes y, por ende, más propensas a necesitar reformas. Este factor debe ser considerado por los inversores que busquen propiedades con un alto potencial de revalorización.

Oportunidades de inversión: Los inversores interesados en propiedades con potencial de revalorización a través de reformas deberían enfocarse en zonas históricas y de alto valor cultural, donde la demanda de renovación es mayor y las oportunidades de valorización tras la reforma son significativas.

PRÁCTICA 6: Metodología mediante DataFrames

En esta sección del proyecto utiliza DataFrames de Spark para realizar diversas operaciones de análisis sobre el conjunto de datos. A continuación, se explica brevemente lo que hace cada sección del código:

Crear sesión de Spark y cargar los datos: Se crea una sesión de Spark utilizando SparkSession.builder, que permite trabajar con DataFrames. Luego, se carga un archivo CSV en un DataFrame (df), donde inferSchema=True permite que PySpark infiera automáticamente el tipo de datos de cada columna. Figura 16

Filtrar propiedades de segunda mano y construidas antes del año 2000: Se filtra el DataFrame para obtener solo las propiedades de segunda mano que fueron construidas antes del año 2000. Se utiliza el método filter junto con condiciones sobre las columnas correspondientes. Figura 17

Filtrar propiedades que necesitan reformas: Se filtra el DataFrame para obtener solo las propiedades que requieren reformas, y luego se muestran los resultados. Figura 18

Calcular media y desviación estándar del precio por año de construcción: Se agrupan las propiedades por el año de construcción y se calculan tres agregados: la media del precio, la desviación estándar del precio, y el número total de propiedades por año. Figura 19

Proporción de propiedades que necesitan reformas por ciudad: Se agrupan las propiedades por ciudad y se calcula la proporción de propiedades que necesitan reformas en cada ciudad. Esto se hace dividiendo el número de propiedades que necesitan reformas por el total de propiedades en cada ciudad. Figura 20

Correlación entre antigüedad y necesidad de reformas: Se crea una nueva columna property_age que calcula la edad de la propiedad. Luego, se agrupa por la antigüedad de las propiedades y se calcula la tasa de necesidad de reformas (porcentaje de propiedades que necesitan reformas) por cada grupo de antigüedad. Figura 21

Clasificación de propiedades por tamaño: Se crea una nueva columna size_category que clasifica las propiedades en tres categorías: "Small" (menos de 50 m^2), "Medium" (de 50 a 150 m^2), y "Large" (más de 150 m^2). Este paso usa la función when para aplicar condiciones basadas en el tamaño en metros cuadrados de las propiedades. Figura 22

Distribución de propiedades por categoría de tamaño y reformas: Se agrupan las propiedades por la categoría de tamaño y se calcula el número de propiedades que necesitan reformas dentro de cada categoría, así como el número total de propiedades en cada categoría. Figura 23

Distribución de necesidad de reformas por tipo de propiedad: Se agrupan las propiedades por tipo (property_type) y se calcula el número de propiedades que necesitan reformas en cada tipo, así como la proporción de propiedades que requieren reformas dentro de cada tipo. Figura 24

Gracias a este análisis podemos ver que las propiedades de segunda mano construidas antes de 2000 presentan una relación interesante entre su precio y su antigüedad. Los datos muestran que, en general, las propiedades más antiguas, como aquellas construidas en 1900 o antes, tienen precios que varían considerablemente, dependiendo del barrio. En áreas más tradicionales, los precios tienden a ser más bajos, mientras que en zonas más demandadas, como algunas de las propiedades históricas, los precios pueden ser mucho más elevados. Esta variabilidad destaca la importancia de la ubicación y el valor histórico de las propiedades en relación con su antigüedad.

Además, un aspecto relevante es la necesidad de reformas en estas propiedades. En Barcelona, por ejemplo, alrededor del 13.48% de las propiedades requieren reformas, lo que representa una oportunidad para los compradores interesados en renovar inmuebles y agregarles valor. La correlación entre la antigüedad de las propiedades y la necesidad de reformas es clara: a medida que las propiedades se hacen más antiguas, aumenta la tasa de renovación. En particular, las propiedades de más de 100 años tienen una tasa de reformas que varía entre el 10% y el 36%. Esto sugiere que las propiedades más antiguas son más propensas a necesitar actualizaciones, lo que puede ser un factor decisivo para quienes buscan viviendas con carácter histórico pero dispuestos a invertir en su modernización.

Conclusiones Principales

Propiedades más antiguas tienen precios más altos: Las propiedades construidas antes de 2000, especialmente aquellas de más de 100 años, muestran una relación directa entre la antigüedad y el precio de venta. Esto sugiere que los compradores están dispuestos a pagar más por propiedades históricas, a pesar de que muchas de ellas podrían requerir reformas importantes.

Las propiedades que requieren reformas representan una oportunidad: Aproximadamente un 13.48% de las propiedades en Barcelona necesitan reformas. Este dato destaca la posibilidad para los compradores de adquirir propiedades a precios más bajos y luego renovar, lo cual puede ser una estrategia rentable si se gestionan correctamente los costes de renovación.

Mayor necesidad de reformas en propiedades más antiguas: A medida que las propiedades se hacen más antiguas, aumenta la probabilidad de que necesiten reformas. Las propiedades construidas hace más de 100 años presentan tasas de necesidad de renovación que varían entre el 10% y el 36%, lo que indica que la antigüedad impacta de manera significativa en la condición estructural y estética de las viviendas.

La antigüedad como factor determinante en la compra: Los compradores que buscan propiedades más antiguas deben estar preparados para el costo adicional que puede representar la renovación. A pesar de los precios más altos de las propiedades históricas, aquellos que están dispuestos a invertir en reformas pueden aprovechar el valor potencial que ofrece la compra de propiedades antiguas, combinando valor histórico con la modernización necesaria.

CONCLUSIÓN

El análisis del mercado inmobiliario en Barcelona, basado en 10,107 propiedades extraídas de Idealista, ha permitido explorar las variables del conjunto de datos mediante tres metodologías: MapReduce, RDDs y DataFrames en PySpark. Este trabajo identificó patrones clave, como la relación entre antigüedad, tamaño y necesidad de reformas, junto con su impacto en atributos como el precio. Las metodologías mostraron ventajas complementarias: MapReduce, aunque más limitado en su capacidad para cálculos complejos, destacó en la transformación y agrupamiento de datos; RDDs ofrecieron flexibilidad para personalizar operaciones, mientras que DataFrames sobresalieron en términos de eficiencia y claridad en análisis agregados.

Si bien se lograron avances significativos, la exploración podría beneficiarse de algunas mejoras. Por ejemplo, la incorporación de datos adicionales, como índices de calidad de vida o información más detallada sobre la ubicación de las propiedades, permitiría un análisis más profundo y contextual. Además, la integración de herramientas de visualización facilitaría la presentación de los hallazgos de manera más efectiva, lo que podría ser especialmente útil para inversores y compradores interesados en propiedades con potencial de revalorización.

En resumen, este proyecto logró explorar de manera efectiva las variables del conjunto de datos y brindar información relevante para entender el mercado inmobiliario en Barcelona. No obstante, el trabajo futuro podría enfocarse en enriquecer los datos disponibles, mejorar la visualización de resultados y refinar las operaciones mediante técnicas avanzadas, ampliando el alcance y la aplicabilidad de los análisis en un mercado dinámico como el de Barcelona.

ANEXO

```
[ ]  class MapperSizeVsRenovation:
         def map(self, data):
             results = []
             for row in data:
                 sq_m_built = float(row["sq_m_built"])
                 needs_renovating = row["needs_renovating"] == "True"
                 results.append((needs_renovating, sq_m_built))
             return results

     class ReducerSizeVsRenovation:
         def reduce(self, tuples):
             results = {}
             for key, value in tuples:
                 if key not in results:
                     results[key] = []
                 results[key].append(value)
             return results
```

```
[ ]  runner = JobRunner()
     print("\nRelación entre tamaño y necesidad de reformas:")
     results = runner.run(filename, MapperSizeVsRenovation, ReducerSizeVsRenovation)

     avg_true = sum(results[True]) / len(results[True]) if True in results else 0
     avg_false = sum(results[False]) / len(results[False]) if False in results else 0

     print(f"Promedio de tamaño para propiedades que necesitan reformas: {avg_true:.2f} m²")
     print(f"Promedio de tamaño para propiedades que NO necesitan reformas: {avg_false:.2f} m²")
```

```
Relación entre tamaño y necesidad de reformas:
Promedio de tamaño para propiedades que necesitan reformas: 118.10 m²
Promedio de tamaño para propiedades que NO necesitan reformas: 110.82 m²
```

Figura 1

```
[ ]  class MapperAveragePriceRenovation:
         def map(self, data):
             results = []
             for row in data:
                 if row["needs_renovating"] == "True":
                     price = float(row["price"])
                     results.append(("needs_renovating", price))
             return results

     class ReducerAveragePriceRenovation:
         def reduce(self, tuples):
             total_price = 0
             count = 0
             for key, value in tuples:
                 total_price += value
                 count += 1
             return {"average_price": total_price / count if count > 0 else 0}
```

```
[ ]  print("\nPromedio de precios de propiedades que necesitan reformas:")
     runner.run(filename, MapperAveragePriceRenovation, ReducerAveragePriceRenovation)
```

```
Promedio de precios de propiedades que necesitan reformas:
{'average_price': 479774.3447488584}
```

Figura 2

```
[ ]  class MapperDistributionByCity:
         def map(self, data):
             results = []
             for row in data:
                 if row["needs_renovating"] == "True":
                     city = row["city"]
                     results.append((city, 1))
             return results

     class ReducerDistributionByCity:
         def reduce(self, tuples):
             results = {}
             for city, count in tuples:
                 results[city] = results.get(city, 0) + count
             return results
```

```
[ ]  print("\nDistribución por ciudad:")
     runner.run(filename, MapperDistributionByCity, ReducerDistributionByCity)
```

```
Distribución por ciudad:
{'Barcelona': 1063,
 'Santa Coloma de Gramenet': 59,
 'Hospitalet de Llobregat': 107,
 'Badalona': 84,
 'Sant Adriá de Besós': 1}
```

Figura 3

```
[ ]  class MapperBedroomsVsRenovation:
         def map(self, data):
             results = []
             for row in data:
                 n_bedrooms = int(row["n_bedrooms"])
                 needs_renovating = row["needs_renovating"] == "True"
                 results.append((needs_renovating, n_bedrooms))
             return results

     class ReducerBedroomsVsRenovation:
         def reduce(self, tuples):
             results = {}
             for key, value in tuples:
                 if key not in results:
                     results[key] = []
                 results[key].append(value)
             return results
```

```
[ ]  print("\nRelación entre dormitorios y necesidad de reformas:")
     results = runner.run(filename, MapperBedroomsVsRenovation, ReducerBedroomsVsRenovation)

     import math
     avg_true = math.floor(sum(results[True]) / len(results[True])) if True in results else 0
     avg_false = math.floor(sum(results[False]) / len(results[False])) if False in results else 0

     print(f"Promedio de habitaciones para propiedades que necesitan reformas: {avg_true:.2f} ")
     print(f"Promedio de habitaciones para propiedades que NO necesitan reformas: {avg_false:.2f} ")
```

```
Relación entre dormitorios y necesidad de reformas:
Promedio de habitaciones para propiedades que necesitan reformas: 3.00
Promedio de habitaciones para propiedades que NO necesitan reformas: 2.00
```

Figura 4

```python
class MapperOldestRenovation:
    def map(self, data):
        results = []
        for row in data:
            # Change this line to handle decimals
            year_built = int(float(row["year_built"]))
            needs_renovating = row["needs_renovating"] == "True"
            if needs_renovating:
                age = 2024 - year_built
                results.append((row["id"], age))
        return results

class ReducerOldestRenovation:
    def reduce(self, tuples):
        results = sorted(tuples, key=lambda x: x[1], reverse=True)
        return results
```

```python
results = runner.run(data, MapperOldestRenovation, ReducerOldestRenovation)

import matplotlib.pyplot as plt

ids, years = zip(*results)  # Divide ids y antigüedades
plt.bar(ids[:10], years[:10])  # Mostrar las 10 propiedades más antiguas
plt.xlabel("Property ID")
plt.ylabel("Antigüedad (años)")
plt.title("Propiedades más antiguas que necesitan reformas")
plt.xticks(rotation=90)
plt.show()
```

```
Propiedades más antiguas que necesitan reformas:
```

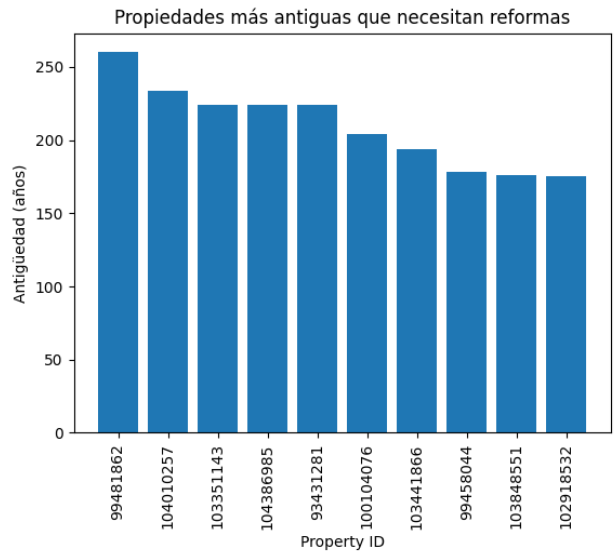

Figura 5

```
[ ]  class MapperAverageAgeByNeighborhood:
         def map(self, data):
             results = []
             for row in data:
                 if row["needs_renovating"] == "True":
                     neighborhood = row["neighborhood"]
                     year_built = int(float(row["year_built"]))
                     age = 2024 - year_built
                     results.append((neighborhood, age))
             return results

     class ReducerAverageAgeByNeighborhood:
         def reduce(self, tuples):
             results = {}
             for neighborhood, age in tuples:
                 if neighborhood not in results:
                     results[neighborhood] = []
                 results[neighborhood].append(age)
             return {k: sum(v) / len(v) for k, v in results.items()}
```

```
[ ]  print("\nPromedio de antigüedad por barrio:")
     runner.run(filename, MapperAverageAgeByNeighborhood, ReducerAverageAgeByNeighborhood)
```

```
 'Ciutat Meridiana - Torre Baró - Vallbona': 53.75,
 'La Verneda i la Pau': 45.42857142857143,
 "El Camp d'En Grassot i Gràcia Nova": 67.44827586206897,
 'La Prosperitat': 58.05,
 'Eixample': 84.17857142857143,
 "L'Antiga Esquerra de l'Eixample": 76.36842105263158,
 'Sant Andreu': 95.1,
 'Sants-Montjuïc': 60.3125,
 'La Marina del Port': 50.23076923076923,
 'Sant Genís Dels Agudells - Montbau': 56.166666666666664,
 'Navas': 70.25,
 'La Bordeta': 34.0,
```

Figura 6

```
class JobRunner:
    def run(self, filename, mapper_class, reducer_class):
        # Leer datos del CSV
        with open(filename, mode='r') as file:
            csv_reader = csv.DictReader(file)
            data = [row for row in csv_reader]

        # Mapper
        mapper = mapper_class()
        mapped_tuples = mapper.map(data)

        # Reducer
        reducer = reducer_class()
        reduced_output = reducer.reduce(mapped_tuples)

        return reduced_output
```

Figura 7

```
[ ] from pyspark import SparkContext
    sc = SparkContext('local', 'test')
```

```
[ ] # Leer el archivo CSV
    rdd_csv = sc.textFile("clean_data.csv")

    # Función para procesar cada línea del archivo CSV
    def take_elem_csv(csv_line):
        lst = csv_line.split(',')
        return lst

    # Extraer encabezados
    header = rdd_csv.first()
    rdd_data = rdd_csv.filter(lambda line: line != header).map(take_elem_csv)
```

Figura 8

```
property_type_count = rdd_data.map(lambda x: x[1]).map(lambda p: (p, 1)).reduceByKey(lambda a, b: a + b).collect()
print("Distribución de tipos de propiedades:")
for property_type, count in property_type_count:
    print(f"{property_type}: {count}")
```

```
Distribución de tipos de propiedades:
Flat / apartment: 8272
Studio flat: 153
Penthouse: 706
Duplex: 453
House: 55
Semi-detached house: 69
Terraced house: 158
Detached house: 233
Estate: 4
Village house: 2
Tower: 1
```

Figura 9

```
[ ] needs_renovating_city_count = (
        rdd_data.filter(lambda x: x[16] == "True")
        .map(lambda x: x[4])
        .map(lambda c: (c, 1))
        .reduceByKey(lambda a, b: a + b)
        .collect()
    )

    print("\nNúmero de propiedades que necesitan reformas por ciudad:")
    for city, count in needs_renovating_city_count:
        print(f"{city}: {count}")
```

```

Número de propiedades que necesitan reformas por ciudad:
El Raval: 95
Barcelona: 951
Ciutat Vella: 30
El Gòtic: 40
La Barceloneta: 14
La Verneda i la Pau: 9
Sant Pere - Santa Caterina i la Ribera: 42
Nou Barris: 29
La Vila Olímpica del Poblenou: 3
Horta Guinardó: 14
El Carmel: 11
La Dreta de l'Eixample: 63
Eixample: 11
La Nova Esquerra de l'Eixample: 24
La Sagrera: 3
L'Antiga Esquerra de l'Eixample: 12
```

Figura 10

```python
average_price_by_type = (
    rdd_data.map(lambda x: (x[1], (float(x[5]) if x[5].replace('.', '', 1).isdigit() else 0, 1)))
    .reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))
    .map(lambda x: (x[0], x[1][0] / x[1][1] if x[1][1] != 0 else 0))
    .collect()
)

print("\nPromedio de precios por tipo de propiedad:")
for property_type, avg_price in average_price_by_type:
    print(f"{property_type}: {avg_price:.2f}€")
```

```
Promedio de precios por tipo de propiedad:
Flat / apartment: 392575.16
Studio flat: 173084.97
Penthouse: 605500.13
Duplex: 543104.98
House: 1487363.45
Semi-detached house: 783897.10
Terraced house: 826044.87
Detached house: 1659904.59
Estate: 4848500.00
Village house: 337500.00
Tower: 1180000.00
```

Figura 11

```python
# Paso 1: Contar las propiedades por tipo de propiedad
property_type_count = rdd_data.map(lambda x: x[1]).map(lambda p: (p, 1)).reduceByKey(lambda a, b: a + b)

# Paso 2: Contar las propiedades por tipo de propiedad que necesitan reformas
needs_renovating_count = rdd_data.filter(lambda x: x[16] == "True").map(lambda x: x[1]).map(lambda p: (p, 1)).reduceByKey(lambda a, b: a + b)

# Paso 3: Unir los dos RDDs por tipo de propiedad
property_type_with_renovations = property_type_count.join(needs_renovating_count)

# Paso 4: Calcular el porcentaje de propiedades que necesitan reformas por tipo de propiedad
property_type_percentage = property_type_with_renovations.map(lambda x: (x[0], (x[1][1] / x[1][0]) * 100)).collect()

# Mostrar los resultados
print("Porcentaje de propiedades que necesitan reforma por tipo de propiedad:")
for property_type, percentage in property_type_percentage:
    print(f"{property_type}: {percentage:.2f}%")
```

```
Porcentaje de propiedades que necesitan reforma por tipo de propiedad:
Flat / apartment: 24.42%
Studio flat: 22.88%
Penthouse: 24.22%
Duplex: 19.43%
House: 29.09%
Semi-detached house: 39.13%
Terraced house: 24.05%
Detached house: 26.61%
Village house: 50.00%
```

Figura 12

```python
bedrooms_needs_renovating = (
    rdd_data.filter(lambda x: x[16] == "True")
    .map(lambda x: (int(float(x[7])) if x[7].replace('.', '', 1).isdigit() else 0, 1))
    .reduceByKey(lambda a, b: a + b)
    .collect()
)

print("\nNúmero de propiedades que necesitan reformas por número de dormitorios:")
for bedrooms, count in bedrooms_needs_renovating:
    print(f"{bedrooms} dormitorios: {count} propiedades con necesidad de reforma")
```

```
Número de propiedades que necesitan reformas por número de dormitorios:
80 dormitorios: 30 propiedades con necesidad de reforma
2 dormitorios: 240 propiedades con necesidad de reforma
35 dormitorios: 11 propiedades con necesidad de reforma
49 dormitorios: 13 propiedades con necesidad de reforma
57 dormitorios: 17 propiedades con necesidad de reforma
```

Figura 13

```
[ ]  from math import sqrt

     # Paso 1: Transformar los datos en un formato clave-valor basado en "year_built"
     year_price_rdd = rdd_data.map(lambda x: (x[10], x[5]))

     # Paso 2: Filtrar las filas que tienen un precio válido
     filtered_year_price_rdd = year_price_rdd.filter(lambda x: x[1].replace('.', '', 1).isdigit())

     # Paso 3: Convertir el precio a float después del filtrado
     final_year_price_rdd = filtered_year_price_rdd.map(lambda x: (x[0], float(x[1])))


     # Paso 4: Calcular suma, suma de cuadrados y conteo por año
     stats_per_year = (
         final_year_price_rdd
         .mapValues(lambda price: (price, price**2, 1))
         .reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1], a[2] + b[2]))
     )

     # Paso 5: Calcular media y desviación estándar
     stats_with_mean_stddev = stats_per_year.mapValues(
         lambda x: {
             "mean_price": x[0] / x[2],
             "property_count": x[2],
         }
     )

     # Paso 6: Recopilar y mostrar los resultados
     results = stats_with_mean_stddev.collect()
     print("Estadísticas por año de construcción:")
     for year, stats in sorted(results, key=lambda x: x[0]):
         print(f"Año {year}: Media = {stats['mean_price']:.2f}, Conteo = {stats['property_count']}")
```

```
Estadísticas por año de construcción:
Año 1764.0: Media = 1600000.00, Conteo = 1
Año 1790.0: Media = 232500.00, Conteo = 2
Año 1792.0: Media = 349000.00, Conteo = 1
Año 1800.0: Media = 386076.92, Conteo = 13
Año 1801.0: Media = 150000.00, Conteo = 1
Año 1805.0: Media = 260000.00, Conteo = 1
Año 1809.0: Media = 349785.71, Conteo = 14
```

Figura 14

```
size_needs_renovating = (
    rdd_data.filter(lambda x: x[16] == "True")
    .map(lambda x: float(x[6]) if x[6].replace('.', '', 1).isdigit() else 0)
    .mean()
)

print(f"\nPromedio del tamaño en metros cuadrados para propiedades que necesitan reformas: {size_needs_renovating:.2f} m²")
```

```
Promedio del tamaño en metros cuadrados para propiedades que necesitan reformas: 222383.80 m²
```

Figura 15

```
# Crear una sesión de Spark
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").appName("Pisos_BCN").getOrCreate()
```

```
# Cargar los datos en un DataFrame
df = spark.read.csv("clean_data.csv", header=True, inferSchema=True)
df.show(5)
df.printSchema()
```

```
+---------+---------------+--------------------+------------+---------+--------+----------+----------+------
|       id|  property_type|             address|neighborhood|     city|   price|sq_m_built|n_bedrooms|bathr
+---------+---------------+--------------------+------------+---------+--------+----------+----------+------
|103375204|Flat / apartment|calle de Sant Pau...|    El Raval|Barcelona|530000.0|        80|         2|
| 94000904|Flat / apartment|            Hospital|    El Raval|Barcelona|165000.0|        58|         2|
|104382041|Flat / apartment|           El Raval|Ciutat Vella|Barcelona|430000.0|       155|         3|
| 97791792|Flat / apartment|    rambla del Raval|    El Raval|Barcelona|279999.0|        89|         2|
|104397583|Flat / apartment|calle de les Carr...|    El Raval|Barcelona|230000.0|        70|         2|
+---------+---------------+--------------------+------------+---------+--------+----------+----------+------
only showing top 5 rows

root
 |-- id: integer (nullable = true)
 |-- property_type: string (nullable = true)
 |-- address: string (nullable = true)
 |-- neighborhood: string (nullable = true)
 |-- city: string (nullable = true)
 |-- price: double (nullable = true)
 |-- sq_m_built: integer (nullable = true)
 |-- n_bedrooms: integer (nullable = true)
 |-- bathrooms: integer (nullable = true)
 |-- floor: integer (nullable = true)
 |-- year_built: double (nullable = true)
 |-- exterior: boolean (nullable = true)
 |-- lift: boolean (nullable = true)
 |-- terrace: boolean (nullable = true)
 |-- balcony: boolean (nullable = true)
```

Figura 16

```
# Filtro de propiedades de segunda mano y construidas antes del año 2000
df_filtered = df.filter((col("second_hand") == True) & (col("year_built") < 2000))
df_filtered.show()
```

```
+---------+---------------+--------------------+------------+---------+---------+----------+----------+-----
|       id|  property_type|             address|neighborhood|     city|    price|sq_m_built|n_bedrooms|bathr
+---------+---------------+--------------------+------------+---------+---------+----------+----------+-----
|103375204|Flat / apartment|calle de Sant Pau...|    El Raval|Barcelona| 530000.0|        80|         2|
| 94000904|Flat / apartment|            Hospital|    El Raval|Barcelona| 165000.0|        58|         2|
| 97791792|Flat / apartment|    rambla del Raval|    El Raval|Barcelona| 279999.0|        89|         2|
|104397583|Flat / apartment|calle de les Carr...|    El Raval|Barcelona| 230000.0|        70|         2|
|102336106|Flat / apartment|           El Raval|Ciutat Vella|Barcelona| 310000.0|        73|         2|
|101044355|Flat / apartment|           El Raval|Ciutat Vella|Barcelona| 335000.0|       149|         4|
|102741132|Flat / apartment|           El Raval|Ciutat Vella|Barcelona| 127000.0|        33|         2|
|104348134|Flat / apartment|  calle del Notariat|    El Raval|Barcelona| 649000.0|       118|         3|
|103801017|Flat / apartment|calle de la Reina...|    El Raval|Barcelona| 138500.0|        50|         2|
|103775272|     Studio flat|  calle de Lancaster|    El Raval|Barcelona| 330000.0|        54|         0|
|101435479|Flat / apartment|   calle de Sant Pau|    El Raval|Barcelona| 450000.0|       110|         2|
|102756178|Flat / apartment| calle de les Cabres|    El Raval|Barcelona| 225000.0|        33|         1|
| 84119197|Flat / apartment|          rambla La|    El Raval|Barcelona|1350000.0|       224|         4|
|104245206|Flat / apartment|avenida del Paral...|    El Raval|Barcelona| 350000.0|        75|         3|
| 93523130|Flat / apartment|           El Raval|Ciutat Vella|Barcelona| 120500.0|        54|         3|
|104142563|Flat / apartment|     calle d'En Roig|    El Raval|Barcelona| 180000.0|        70|         2|
|100859087|Flat / apartment|           El Raval|Ciutat Vella|Barcelona| 448000.0|        56|         1|
|104182440|Flat / apartment|calle de Sant Bar...|    El Raval|Barcelona| 199900.0|        40|         1|
|104023286|       Penthouse|   calle de Sant Pau|    El Raval|Barcelona| 415000.0|        69|         2|
|103774998|Flat / apartment| calle de Ferlandina|    El Raval|Barcelona| 190000.0|        73|         4|
+---------+---------------+--------------------+------------+---------+---------+----------+----------+-----
only showing top 20 rows
```

Figura 17

```
# Propiedades que necesitan reformas
df_needs_renovating = df.filter(col("needs_renovating") == True)
df_needs_renovating.show()
```

```
+---------+------------------+--------------------+--------------------+---------+---------+----------+-----
|       id|     property_type|             address|        neighborhood|     city|    price|sq_m_built|n_bed
+---------+------------------+--------------------+--------------------+---------+---------+----------+-----
|103801017|  Flat / apartment|calle de la Reina...|           El Raval|Barcelona| 138500.0|        50|
|104142563|  Flat / apartment|     calle d'En Roig|           El Raval|Barcelona| 180000.0|        70|
|103824316|  Flat / apartment|calle de Sant Pau...|           El Raval|Barcelona| 135000.0|        49|
| 98781352|            Duplex|calle de Sant Ant...|           El Raval|Barcelona| 159000.0|       190|
|103351143|  Flat / apartment| calle de Simó Oller|           El Gòtic|Barcelona| 595000.0|       131|
| 99306956|  Flat / apartment|          codols, 26|           El Gòtic|Barcelona| 300150.0|        90|
|102041190|  Flat / apartment|      La Barceloneta|       Ciutat Vella|Barcelona| 175000.0|        34|
|104034134|            Duplex|               Horta|     Horta Guinardó|Barcelona| 196000.0|       102|
|104331430|  Flat / apartment| calle dels Cotoners|Sant Pere - Santa...|Barcelona| 475000.0|       150|
|104374913|  Flat / apartment|calle del Fonolla...|Sant Pere - Santa...|Barcelona| 255000.0|        72|
|103395490|  Flat / apartment|    calle d'Ortigosa|Sant Pere - Santa...|Barcelona| 750000.0|       161|
|104374123|  Flat / apartment|calle d'En Cortin...|Sant Pere - Santa...|Barcelona| 180000.0|        42|
|103662585|  Flat / apartment|      calle de Beret|Can Peguera - El ...|Barcelona| 139500.0|        62|
|104389110|  Flat / apartment|     calle de Lepant|  La Sagrada Família|Barcelona| 308000.0|        66|
|100282427|Semi-detached house|   calle de Mühlberg|           El Carmel|Barcelona| 235000.0|        96|
|103697199|  Flat / apartment|  calle del Rosselló|La Dreta de l'Eix...|Barcelona|2300000.0|       363|
|103684992|  Flat / apartment|   calle de Provença|La Dreta de l'Eix...|Barcelona|1150000.0|       187|
|100269640|  Flat / apartment|        rambla La Nn|La Dreta de l'Eix...|Barcelona| 380000.0|       130|
|104226768|  Flat / apartment|    calle de Bergara|La Dreta de l'Eix...|Barcelona| 750000.0|       138|
|103681742|  Flat / apartment|   calle de Provença|La Dreta de l'Eix...|Barcelona|1150000.0|       187|
+---------+------------------+--------------------+--------------------+---------+---------+----------+-----
only showing top 20 rows
```

Figura 18

```
[ ] # Calcular media y desviación estándar del precio por año de construcción
stats_by_year = df.groupBy("year_built").agg(
    mean("price").alias("mean_price"),
    stddev("price").alias("stddev_price"),
    count("*").alias("property_count")
)
stats_by_year.show()
```

```
+----------+------------------+------------------+--------------+
|year_built|        mean_price|      stddev_price|property_count|
+----------+------------------+------------------+--------------+
|    1988.0|          921080.0| 656767.5844619618|            25|
|    1976.0| 418998.8695652174|391191.73296023294|           115|
|    1951.0|          410000.0| 231099.3792222798|            14|
|    1846.0|          231500.0| 26162.95090390226|             2|
|    1792.0|          349000.0|              NULL|             1|
|    1940.0| 994090.2139303483|1028264.1119792904|           201|
|    2024.0| 546068.6666666666|202008.24085500406|            21|
|    1928.0|          420440.0|480315.96198058355|            25|
|    1900.0|496528.11320754717| 464017.5438631696|           795|
|    1979.0| 590114.2201834862| 547317.1161697718|           109|
|    1856.0|          387400.0| 171616.7241267587|             5|
|    1965.5|         2881600.0|  4689077.18897978|            20|
|    1953.0| 876208.3333333334|1071437.8790313457|            24|
|    1830.0|1044666.6666666666|1266838.7163855284|             3|
|    1987.0|         625656.25|318447.30250986025|            32|
|    1909.0|          652500.0|196532.44007033546|             4|
|    1959.0|388688.62376237626|402327.76496127766|           101|
|    1934.0| 652443.1818181818|280405.29670023633|            44|
|    1858.0|          114000.0|               0.0|             2|
|    1904.0|          902000.0|469963.82839533506|             4|
+----------+------------------+------------------+--------------+
only showing top 20 rows
```

Figura 19

```
[ ]  # Proporción de propiedades que necesitan reformas por ciudad
     renovating_by_city = df.groupBy("city").agg(
         (count(when(col("needs_renovating") == True, 1)) / count("*")).alias("renovating_ratio"),
         count("*").alias("total_properties")
     )
     renovating_by_city.show()
```

```
+--------------------+--------------------+----------------+
|                city|    renovating_ratio|total_properties|
+--------------------+--------------------+----------------+
|Santa Coloma de G...| 0.11776447105788423|             501|
| Sant Adriá de Besós|0.013513513513513514|              74|
|           Barcelona|  0.1348471394139287|            7883|
|Hospitalet de Llo...| 0.13144963144963145|             814|
|            Badalona| 0.10071942446043165|             834|
+--------------------+--------------------+----------------+
```

Figura 20

```
[ ]  # Correlación entre antigüedad y necesidad de reformas
     df = df.withColumn("property_age", lit(2024) - col("year_built"))
     correlation_data = df.select("property_age", "needs_renovating").groupBy("property_age").agg(
         mean(col("needs_renovating").cast("integer")).alias("renovating_rate")
     )
     correlation_data.show()
```

```
+------------+--------------------+
|property_age|     renovating_rate|
+------------+--------------------+
|       147.0|                 0.0|
|       184.0|                 0.0|
|       169.0|0.36363636363636365|
|       160.0|0.16666666666666666|
|        67.0|0.21818181818181817|
|        70.0| 0.3142857142857143|
|         8.0|                 0.0|
|       168.0|                 0.0|
|        69.0| 0.3533834586466165|
|         0.0|                 0.0|
|       206.0|                 0.0|
|         7.0|                 0.0|
|        59.5| 0.2857142857142857|
|       142.0|                 0.0|
|       112.0|                 0.0|
|       154.0| 0.3333333333333333|
|       232.0|                 0.0|
|       124.0|0.11823899371069183|
|       128.0|                 0.0|
|       180.0|                 0.0|
+------------+--------------------+
only showing top 20 rows
```

Figura 21

```
[ ]  # Crear columna para clasificar propiedades por tamaño
     df = df.withColumn(
         "size_category",
         when(col("sq_m_built") < 50, "Small")
         .when((col("sq_m_built") >= 50) & (col("sq_m_built") <= 150), "Medium")
         .otherwise("Large")
     )
```

Figura 22

```python
# Mostrar distribución de propiedades por categoría de tamaño y reformas
size_distribution = df.groupBy("size_category").agg(
    count(when(col("needs_renovating") == True, 1)).alias("renovating_count"),
    count("*").alias("total_count")
)
size_distribution.show()
```

```
+-------------+----------------+-----------+
|size_category|renovating_count|total_count|
+-------------+----------------+-----------+
|       Medium|             929|       7823|
|        Small|             108|        646|
|        Large|             277|       1637|
+-------------+----------------+-----------+
```

Figura 23

```python
# Distribución de necesidad de reformas por tipo de propiedad
renovating_by_type = df.groupBy("property_type").agg(
    count(when(col("needs_renovating") == True, 1)).alias("renovating_count"),
    count("*").alias("total_count"),
    (count(when(col("needs_renovating") == True, 1)) / count("*")).alias("renovating_ratio")
)
renovating_by_type.show()
```

```
+-------------------+----------------+-----------+--------------------+
|      property_type|renovating_count|total_count|    renovating_ratio|
+-------------------+----------------+-----------+--------------------+
|     Terraced house|              29|        158|0.18354430379746836|
|     Detached house|              48|        233|0.20600858369098712|
|   Flat / apartment|            1083|       8272|0.13092359767891681|
|          Penthouse|              85|        706|0.12039660056657224|
|        Studio flat|              13|        153|0.08496732026143791|
|      Village house|               1|          2|                 0.5|
|              House|              10|         55|0.18181818181818182|
|             Estate|               0|          4|                 0.0|
|Semi-detached house|              16|         69| 0.2318840579710145|
|              Tower|               0|          1|                 0.0|
|             Duplex|              29|        453| 0.0640176600441501|
+-------------------+----------------+-----------+--------------------+
```

Figura 24