

[illegible]

3. El archivo llamado archivo3.db debe ser abierto con SQLite.

```
# 3. El archivo llamado archivo3.db debe ser abierto con SQLite.

conexion = sqlite3.connect("archivo3.db")
✓ 0.0s

df = pd.read_sql_query("select * from archivo3", conexion)
df.head()
✓ 0.0s
```

	Año	Kilómetros	Precio	Lugar	Negociación	Categoría	Marca	Subtipo	Modelo	Publicación	...	Color	Último número de la placa	Dirección	Motor (cilindros)
0	2012	86000	16.0	Quito	Fijo	USED	Ford	Todoterreno	Escape	None	...	None	None	None	None
1	2019	48399	13.2	Cuenca	Negociable	USED	Chevrolet	Hatchback (5 Puertas)	Spark GT	None	...	None	None	None	None
2	2018	29000	17.0	Guayaquil	Negociable	USED	Suzuki	Todoterreno Ligero	S-Cross	None	...	None	None	None	None
3	2015	150000	16.8	Cuenca	Negociable	USED	Suzuki	Todoterreno	GRAN VITARA SZ NEXT	None	...	None	None	None	None
4	2013	171000	26.4	Quito	Negociable	USED	Kia	Todoterreno	Sportage	#1722523	...	None	None	None	None

5 rows × 16 columns

4. El archivo llamado archivo4.csv debe ser importado a mongoDB.

```
# 4. El archivo llamado archivo4.csv debe ser importado a mongoDB.

infocarros4 = "archivo4.csv"
df4 = pd.read_csv(infocarros4)
cars4 = db["carros4"]
cars4.insert_many(df4.to_dict(orient="records"))
✓ 0.2s

InsertManyResult([ObjectId('673cc5b54a1068af8d69641b'), ObjectId('673cc5b54a1068af8d69641c'), ObjectId('673cc5b54a1068af8d69641d'), ObjectId('673cc5b54a1068af8d69641e'), ObjectId('673cc5b54a1068af8d69641f')])

info4 = list(cars4.find())
datoscarros4 = pd.DataFrame(info4)
datoscarros4.head()
✓ 0.2s
```

	_id	Año	Kilómetros	Precio	Lugar	Negociación	Categoría	Marca	Subtipo	Modelo	...	Color	Último número de la placa	Dirección	Motor (cilindros)	Tapizado
0	673cb7a3886af74b2b0f67b5	2016	71000	40.9	Loja	Negociable	USED	Ford	Todoterreno	Explorer	...	NaN	NaN	NaN	NaN	NaN
1	673cb7a3886af74b2b0f67b6	2016	98000	23.9	Quito	Negociable	USED	Mitsubishi	Todoterreno	Outlander	...	NaN	NaN	NaN	NaN	NaN
2	673cb7a3886af74b2b0f67b7	2022	39000	37.9	Quito	Negociable	USED	Toyota	Todoterreno	RAV 4	...	NaN	NaN	NaN	NaN	NaN
3	673cb7a3886af74b2b0f67b8	2008	224000	10.9	Quito	Negociable	USED	Nissan	Todoterreno	X-Trail	...	NaN	NaN	NaN	NaN	NaN
4	673cb7a3886af74b2b0f67b9	2013	151000	26.5	Riobamba	Negociable	USED	Ford	Todoterreno	Explorer XLT	...	NaN	NaN	NaN	NaN	NaN

5 rows × 17 columns

5. El dataframe generado en el paso 1 se lo debe exportar a mongoDB

```
# 5. El dataframe generado en el paso 1 se lo debe exportar a mongoDB

infocarros1MDB = cars1
cars1MDB = db["carros1"]
cars1MDB.insert_many(infocarros1MDB.to_dict(orient="records"))
✓ 0.1s

InsertManyResult([ObjectId('673cc5b54a1068af8d6975ba'), ObjectId('673cc5b54a1068af8d6975bb'), ObjectId('673cc5b54a1068af8d6975bc')])
```

carros1				
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
761.86 kB	9 K	563.00 B	1	126.98 kB

6. El dataframe generado en el paso 3 se lo debe exportar a mongoDB

```
# 6. El dataframe generado en el paso 3 se lo debe exportar a mongoDB

infocarros2MDB = df
cars2MDB = db["carros3"]
cars2MDB.insert_many(infocarros2MDB.to_dict(orient= 'records'))

✓ 0.0s
```

InsertManyResult([ObjectId('673cc5b64a1068af8d698759'), ObjectId('673cc5b64a1068af8d698759')])

carros3

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
364.54 kB	4.5 K	438.00 B	1	81.92 kB

7. El dataframe generado en el paso 2 se lo debe exportar a SQLite

```
# 7. El dataframe generado en el paso 2 se lo debe exportar a SQLite

data = list(cars2.find())
for record in data:
    record['_id'] = str(record['_id'])
cars2SQL = pd.DataFrame(data)
cars2SQL.to_sql("carros2", conexion, if_exists="replace", index=False)

✓ 0.1s
```

4510

Name	Type	Schema
▼ Tables (4)		
> archivo3		CREATE TABLE "archivo3" ("A
> carros1		CREATE TABLE "carros1" ("Af
> carros2		CREATE TABLE "carros2" ("_i
> carros4		CREATE TABLE "carros4" ("_i

8. El dataframe generado en el paso 4 se lo debe exportar a SQLite

```
# 8. El dataframe generado en el paso 4 se lo debe exportar a SQLite
data2 = list(cars4.find())
cars4SQL = pd.DataFrame(data)
cars4SQL.to_sql('carros4', conexion, if_exists="replace", index=False)

✓ 0.2s
```

4510

Name	Type	Schema
▼ Tables (4)		
> archivo3		CREATE TABLE "archivo3" ("A
> carros1		CREATE TABLE "carros1" ("Af
> carros2		CREATE TABLE "carros2" ("_i
> carros4		CREATE TABLE "carros4" ("_i

9. Se debe exportar los datos a un dataframe.

```
# 9. Se debe exportar los datos a un dataframe.
data = pd.concat([cars2SQL, cars4SQL], ignore_index=True)
carsMDB = db["carros"]
data_dict = data.to_dict(orient="records")
for record in data_dict:
    if "_id" in record:
        del record["_id"]
carsMDB.insert_many(data_dict)
```

✓ 0.2s

InsertManyResult([ObjectId('673ccfd14a1068af8d699028'), ObjectId('673ccfd1

carros

Storage size: 397.31 KB	Documents: 14 K	Avg. document size: 589.00 B	Indexes: 1	Total index size: 69.63 KB
----------------------------	--------------------	---------------------------------	---------------	-------------------------------

10. Se debe exportar los datos a un dataframe.

```
# 10. Se debe exportar los datos a un dataframe.

data1 = list(cars1MDB.find())
data2 = list(cars2MDB.find())
for record in data1: record["_id"] = str(record["_id"])
for record in data2: record["_id"] = str(record["_id"])
df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)
carsSQL = pd.concat([df1, df2], ignore_index=True)
conexion = sqlite3.connect('database.db', timeout=30)
with conexion:
    carsSQL.to_sql("carros", conexion, if_exists="replace", index=False)
```

✓ 0.4s

Tables (3)

>  carros	CREATE TABLE "carros" ("_ic
>  carros1	CREATE TABLE "carros1" ("ir
>  carros2	CREATE TABLE "carros2" ("ir

Limpieza de datos. (4 puntos)

11. Debe unificar y limpiar los datos de tal manera que estén listos para analizarlos

```
# 11. Debe unificar y limpiar los datos de tal manera que estén listos para analizarlos.

df_Carros1 = carsSQL.drop_duplicates()
df_Carros1 = carsSQL.dropna()
df_Carros1.isnull().sum()
```

[45] ✓ 0.1s

...	_id	0
	Año	0
	Kilómetroraje	0
	Precio	0
	Lugar	0
	Negociación	0
	Categoría	0
	Marca	0
	Subtipo	0
	Modelo	0
	Publicación	0
	Recorrido	0
	Vidrios	0
	Transmisión	0
	Combustible	0
	Sistema de climatización	0
	Tracción	0
	Color	0
	Último número de la placa	0
	Dirección	0
	Motor(cilindraje)	0
	Tapizado	0
	Placa	0
	Tipo de Motor	0
	Entrada desde	0
	...	
	TipodeMotor	0
	Entradasdesde	0
	Cuotasdesde	0
	Númerodecuotas	0
	dtype: int64	

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

```
data = list(carsMDB.find())
carrosMDB = pd.DataFrame(data)

df_Carros2 = carrosMDB.drop_duplicates()
df_Carros2 = carrosMDB.dropna()

df_Carros2.isnull().sum()
```

[53] ✓ 0.3s

...	_id	0
	Año	0
	Kilómetroraje	0
	Precio	0
	Lugar	0
	Negociación	0
	Categoría	0
	Marca	0
	Subtipo	0
	Modelo	0
	Publicación	0
	Recorrido	0
	Vidrios	0
	Transmisión	0
	Combustible	0
	Sistema de climatización	0
	Tracción	0
	Color	0
	Último número de la placa	0
	Motor(cilindraje)	0
	Placa	0
	Tapizado	0
	Dirección	0
	Entrada desde	0
	Cuotas desde	0
	Número de cuotas	0
	Tipo de Motor	0
	dtype: int64	

Análisis de datos. (10 puntos)

Con pandas debe contestar a las siguientes preguntas:

- a. Los autos que tienen los años más recientes y los autos que tienen los autos más antiguos.
- b. El promedio de kilometraje de los autos de Quito o Guayaquil.
- c. El promedio de kilometraje de los autos de Cuenca.
- d. Los autos que son negociables y fijos en porcentaje, campo negociación.
- e. Porcentaje de autos nuevos y usados.
- f. Las marcas que más se repiten.
- g. Modelos que más se repiten.
- h. Porcentaje de subtipos.
- i. Auto más caro y más barato, con sus características.
- j. Auto de Guayaquil que menos kilometraje tiene y más caro es.

Parte teórica (6 puntos)

Contestar a las siguientes preguntas:

- a) Defina cada etapa de la pirámide del conocimiento.

La pirámide del conocimiento es un esquema que representa un paso a paso de como procesar y analizar la información, empezando por los datos que es la parte mas básica de la pirámide, pasando por la información que nos proporcionan esos datos. Llegando al conocimiento una vez analizado los datos y por ultimo la sabiduría una vez obtenido el conocimiento.

- b) Indique lo que es un archivo .json

Un archivo json es un formato de transmisión de información basado en javascript. Su sintaxis es parecida a un objeto en el mismo lenguaje. Es altamente usado en el desarrollo web y bases de datos no sql.

- c) Explique las diferencias entre Bases de Datos relacionales y NoSQL.

Las bases de datos sql o relacionales se escriben en con la sintaxis de SQL y están relacionadas por tablas, mientras que las no sql se basan en json para guardar sus datos

Entregables:

Archivo ipynb y el pdf con capturas de pantalla indicando el procedimiento.

En el pdf puede poner las respuestas de la parte teórica.

Importante:

Tomar en consideración:

- El uso de dataframes en cada etapa.
- MongoDB y SQLite tienen las opciones de importación y exportación.