

## Informe de resultados prueba técnica Quind

Aspirante: David Armando Ruiz Lugo

### Análisis exploratorio de datos

Se recibió la base de datos Films2.xlsx, con el fin de realizar el proceso de ETL con Python y PySpark.

#### 1. Identificar las dimensiones de las tablas

```
Dimensiones Store: (2, 4)
Dimensiones Customer: (1392, 11)
Dimensiones Rental: (16044, 7)
Dimensiones Inventory: (4581, 4)
Dimensiones Film: (1003, 14)
```

#### 2. Validación de los datos imputados

Al realizar el análisis exploratorio fue posible identificar inconsistencias en la imputación de datos y en los nombres de las columnas de las diferentes tablas del archivo Films 2.xlsx, como se puede apreciar a continuación.

```
print(df_customer.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1392 entries, 0 to 1391
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customer_id           1392 non-null  int64
1   store_id              1392 non-null  int64
2   first_name            1392 non-null  object
3   last_name             1392 non-null  object
4   email                 1392 non-null  object
5   address_id            1392 non-null  int64
6   active                1392 non-null  int64
7   create_date           1392 non-null  object
8   last_update           1392 non-null  object
9   customer_id_old       1392 non-null  object
10  segment               1392 non-null  object
dtypes: int64(4), object(7)

print(df_film.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1003 entries, 0 to 1002
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   film_id               1003 non-null  object
1   title                 1003 non-null  object
2   description            1003 non-null  object
3   release_year          1003 non-null  object
4   language_id           1003 non-null  int64
5   original_language_id  1003 non-null  object
6   rental_duration       1003 non-null  int64
7   rental_rate           1003 non-null  object
8   length                1003 non-null  object
9   replacement_cost      1003 non-null  object
10  num_voted_users        1003 non-null  object
11  rating                1003 non-null  object
12  special_features       1003 non-null  object
13  last_update           1003 non-null  object
dtypes: int64(2), object(12)
```

#### Inconsistencias encontradas

- Nombres de las columnas con espacios adicionales
- Valores de features con caracteres alfanuméricos y espacios adicionales
- Valores nulos no reconocidos por el intérprete

Debido a la irregularidad de las tablas se procede a limpiarlas todas en pro de eliminar cualquier posible error. Esto con ayuda de las funciones

```
print(df_film.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1003 entries, 0 to 1002
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   film_id               1003 non-null  int64
1   title                 1003 non-null  object
2   description            1003 non-null  object
3   release_year          1003 non-null  int64
4   language_id           1003 non-null  int64
5   original_language_id  0 non-null     float64
6   rental_duration       1003 non-null  int64
7   rental_rate           1003 non-null  float64
8   length                1003 non-null  int64
9   replacement_cost      1003 non-null  float64
10  num_voted_users        1003 non-null  int64
11  rating                1003 non-null  object
12  special_features       1003 non-null  object
13  last_update           1003 non-null  object
dtypes: float64(3), int64(6), object(5)

print(df_customer.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1392 entries, 0 to 1391
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   customer_id           1392 non-null  int64
1   store_id              1392 non-null  int64
2   first_name            1392 non-null  object
3   last_name             1391 non-null  object
4   email                 1392 non-null  object
5   address_id            1392 non-null  int64
6   active                1392 non-null  int64
7   create_date           1392 non-null  object
8   last_update           1392 non-null  object
9   customer_id_old       793 non-null   object
10  segment               793 non-null   object
dtypes: int64(4), object(7)
```

Como se puede observar las inconsistencias han sido eliminadas, esto se evidencia en el conteo de valores no nulos, el nombre de las columnas y el formato de las mismas.

**Nota:** Debido al tamaño del informe solo se muestra la info de 2 tablas, pero se filtraron las 5.

### 3. Análisis de datos:

#### a. Clientes que más rentan películas y películas más rentadas.

Frecuencia de clientes			Frecuencia películas rentadas		
	customer_id	frecuencia		inventory_id	frecuencia
0	148	46	0	4581	5
1	526	45	1	1083	5
2	144	42	2	1113	5
3	236	42	3	1111	5
4	75	41	4	2897	5
..	...	...	...	...	...
594	248	15	4575	3214	2
595	110	14	4576	2786	1
596	61	14	4577	2662	1
597	281	14	4578	1580	1
598	318	12	4579	3372	1

[599 rows x 2 columns]

[4580 rows x 2 columns]

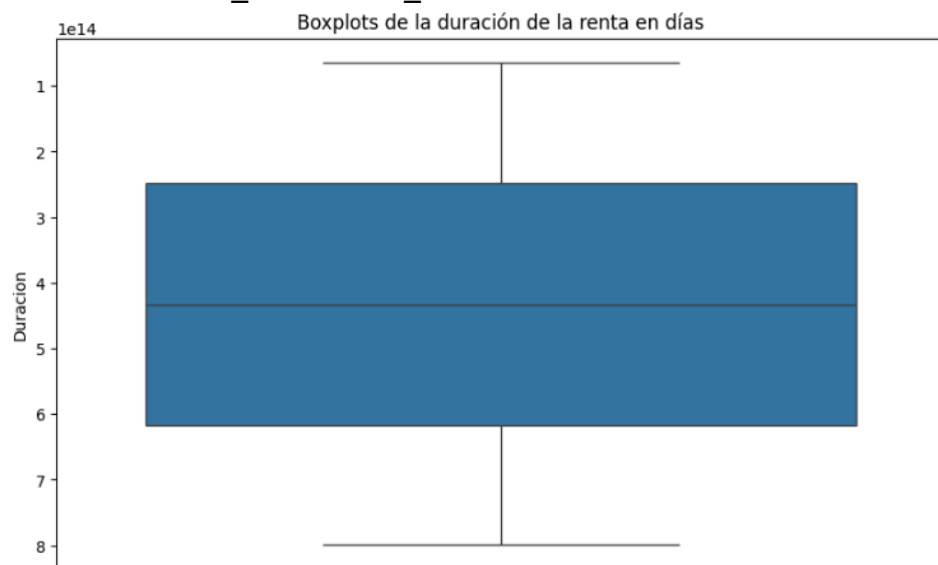
Se puede observar que todos los clientes han rentado películas en múltiples ocasiones y que ninguna película ha sido rentada más de 5 veces.

#### b. Películas rentadas por tienda

Frecuencia películas rentadas por tienda		
	staff_id	frecuencia
0	1	8040
1	2	8004

Se observa que el número de películas rentadas por tienda es muy similar.

#### c. Distribución de la duración de renta de las películas, para lo cual se restando la columna return\_date - rental\_date.



### Esquema de aplicación propuesto

etl\_pyspark\_project/

└─ etl\_project/

— __init__.py	# Archivo de inicialización del paquete principal
— config/	
— __init__.py	# Para convertir 'config' en un módulo
— config.py	# Configuración de rutas y variables globales
— extract/	
— __init__.py	# Para convertir 'extract' en un módulo
— extract_data.py	# Código de extracción de datos
— load/	
— __init__.py	# Para convertir 'load' en un módulo
— load_data.py	# Código de carga de datos
— transform/	
— __init__.py	# Para convertir 'transform' en un módulo
— transform_data.py	# Código de transformación de datos
— utils/	
— __init__.py	# Para convertir 'utils' en un módulo
— spark_session.py	# Código para crear la sesión de Spark
— observability/	
— __init__.py	# Para inicializar el módulo de observabilidad
— logging_config.py	# Configuración de logs para registrar eventos
— metrics.py	# Códigos para métricas de rendimiento y otros KPIs
— main.py	# Archivo principal para ejecutar el ETL

En el archivo readme del repositorio encontrará más detalles acerca de la construcción de la app propuesta.

### Preguntas de negocio

1. ¿Cuál es la distribución del tiempo promedio de renta por meses del año?
2. ¿Cuál es la distribución de special\_features por meses del año?
3. Correlación entre rental\_duration y replacement\_cost
4. Distribución de customer["address\_id"] y rental["rental\_duration"]
5. Cuales son las zonas de influencia de cada tienda, ( se deben cruzar las tablas: customer, rental e inventory.

Nota 1: Es conveniente enriquecer la data con la edad de los clientes

Nota 2: Es conveniente enriquecer la data con las categorías de las películas rentado

### Conclusiones:

- La base de datos presentada contiene muchas inconsistencias tanto en la imputación, como en el nombre de las columnas, por lo que es conveniente limpiar y transformar todas las columnas de todas las tablas para evitar datos corruptos.
- Para realizar un análisis que permita la toma de decisión, es necesario realizar varios cruces entre tablas con el fin de exponer patrones y comportamiento de los usuarios y películas, así como de las tiendas.
- Si bien la data suministrada nos da información sobre todos los actores involucrados en el modelo de negocio, es necesario enriquecer los datos con el fin de entender mejor el patrón de comportamiento de los usuarios.
- Debido a la gran cantidad de información lo más recomendable es trabajar estos datos con el framework de procesamiento de datos PySpark, dado que gracias a sus cualidades es posible distribuir las cargas y atender a las solicitudes de estas 2 o más tiendas.