

# Proyecto Integrador III

## Crea un notebook, carga los csv. responde y justifica.

El notebook se debe entregar y debe ser posible visualizar los resultados de cada pregunta:  
El campo TotalPrice en la tabla sales no tiene valores válidos. Utilizando la información de precios de la tabla products, calcula el valor real de la venta para cada registro y almacena en una nueva columna

Utiliza la siguiente fórmula:

$$\text{TotalPriceCalculated} = (\text{Quantity} \times \text{UnitPrice}) \times (1 - \text{Discount})$$

### Evidencia:

```
# Verifico los registros de cada dataset
print("Cantidad de registros del dataframe categories:", len(categories))
print("Cantidad de registros del dataframe cities:", len(cities))
print("Cantidad de registros del dataframe countries:", len(countries))
print("Cantidad de registros del dataframe customers:", len(customers))
print("Cantidad de registros del dataframe employees:", len(employees))
print("Cantidad de registros del dataframe products:", len(products))
print("Cantidad de registros del dataframe sales:", len(sales))
```

```
Cantidad de registros del dataframe categories: 11
Cantidad de registros del dataframe cities: 96
Cantidad de registros del dataframe countries: 206
Cantidad de registros del dataframe customers: 98759
Cantidad de registros del dataframe employees: 23
Cantidad de registros del dataframe products: 452
Cantidad de registros del dataframe sales: 6758125
```

Acá se puede ver la carga de los diferentes dataframes desde los archivos indicados.

```
# creo el campo TotalPriceCalculated
sales_df['TotalPriceCalculated'] = (sales_df['Quantity'] * sales_df['Price']) * (1 - sales_df['Discount'])
```

```
sales_df.head()
```

	SalesID	SalesPersonID	CustomerID	ProductID	Quantity	Discount	TotalPrice	SalesDate	TransactionNumber	Price	TotalPriceCalculated
0	1	6	27039	381	7	0.0	0.0	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.2337	309.63590
1	2	16	25011	61	7	0.0	0.0	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.5460	437.82200
2	3	13	94024	23	24	0.0	0.0	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.0184	1896.44160
3	4	8	73966	176	19	0.2	0.0	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.3167	1236.01384
4	5	10	32653	310	9	0.0	0.0	2018-02-12 15:37:03.940	4BGS0Z50MAZ8NDAFHHP3	79.9780	719.80200

En esta imagen se puede ver el nuevo campo calculado llamado `TotalPriceCalculated`

## Detecta los outliers en la columna de ventas totales (`TotalPriceCalculated`)

Utilizando el criterio del rango intercuartílico (IQR). Luego, crea una nueva columna llamada `IsOutlier` que tenga el valor 1 si el registro es un outlier y 0 en caso contrario. ¿Cuántos outliers se detectaron?

```
# Detecto los outliers
# Calculo los cuartiles e IQR

Q1 = sales_df['TotalPriceCalculated'].quantile(0.25)
Q3 = sales_df['TotalPriceCalculated'].quantile(0.75)
IQR = Q3 - Q1
```

```
# Calculo los límites
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
# Creo el campo IsOutlier para identificar si el dato TotalPriceCalculated es outlier

sales_df['IsOutlier'] = ((sales_df['TotalPriceCalculated'] < lower_bound) |
                        (sales_df['TotalPriceCalculated'] > upper_bound)).astype(int)
```

```
# Verifico los outliers
num_outliers = sales_df['IsOutlier'].sum()
print(f"Se detectaron {num_outliers} outliers.")
```

Se detectaron 48217 outliers.

Según el dataframe suministrado se han detectado 48217 datos outliers.

A partir de la columna `SalesDate`, crea una nueva columna que contenga únicamente la hora de la venta.

Luego, identifica en qué hora del día se concentran más ventas totales (`TotalPriceCalculated`).

	Hour	TotalPriceCalculated
0	0.00	178,313,404.12
1	1.00	177,823,245.94
2	2.00	178,420,846.95
3	3.00	177,443,825.95
4	4.00	177,932,710.89
5	5.00	177,506,190.92
6	6.00	178,381,199.01
7	7.00	177,817,815.98
8	8.00	177,882,244.32
9	9.00	178,166,580.43
10	10.00	177,978,639.80
11	11.00	178,143,011.00
12	12.00	177,662,635.35
13	13.00	177,899,938.82
14	14.00	177,744,367.94
15	15.00	178,021,788.94
16	16.00	179,014,421.24
17	17.00	178,290,351.38
18	18.00	177,932,537.66
19	19.00	178,346,115.33
20	20.00	178,949,163.88
21	21.00	177,885,339.28
22	22.00	177,690,265.66
23	23.00	177,968,147.55

En esta imagen se puede ver la cantidad de ventas por hora. La hora que más se ve es a las 16 horas con una suma total de 179,014,421.24.

¿La empresa vende más durante los días de semana o en el fin de semana? Utiliza la columna SalesDate para identificar el día de la semana de cada venta, clasifica los registros como Entre semana o Fin de semana, y compara el total de ventas (TotalPriceCalculated) entre ambos grupos.

```
# Obtengo el número del día (0=lunes, 6=domingo)
sales_df['Weekday'] = sales_df['SalesDate'].dt.weekday
```

```
sales_df.head()
```

ersonID	CustomerID	ProductID	Quantity	Discount	TotalPrice	SalesDate	TransactionNumber	Price	TotalPriceCalculated	IsOutlier	Hour	Weekday
6	27039	381	7	0.00	0.00	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.23	309.64	0	7.00	0.00
16	25011	61	7	0.00	0.00	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.55	437.82	0	16.00	4.00
13	94024	23	24	0.00	0.00	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.02	1,896.44	0	19.00	3.00
8	73966	176	19	0.20	0.00	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.32	1,236.01	0	14.00	5.00
10	32653	310	9	0.00	0.00	2018-02-12 15:37:03.940	4BGS0Z5OMAZ8NDAFHHP3	79.98	719.80	0	15.00	0.00

Aquí creo la columna Weekday del dataframe el número del día donde 0 es lunes y 6 es domingo. Una vez calculado el número del día calculo si es día de la semana o fin de semana como se puede ver en la siguiente imagen.

```
# Clasifico si es día de semana o fin de semana
sales_df['DayType'] = sales_df['Weekday'].apply(lambda x: 'Weekday' if x < 5 else 'Weekend')
```

```
sales_df.head()
```

CustomerID	ProductID	Quantity	Discount	TotalPrice	SalesDate	TransactionNumber	Price	TotalPriceCalculated	IsOutlier	Hour	Weekday	DayType
27039	381	7	0.00	0.00	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.23	309.64	0	7.00	0.00	Weekday
25011	61	7	0.00	0.00	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.55	437.82	0	16.00	4.00	Weekday
94024	23	24	0.00	0.00	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.02	1,896.44	0	19.00	3.00	Weekday
73966	176	19	0.20	0.00	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.32	1,236.01	0	14.00	5.00	Weekend
32653	310	9	0.00	0.00	2018-02-12 15:37:03.940	4BGS0Z5OMAZ8NDAFHHP3	79.98	719.80	0	15.00	0.00	Weekday

```
# Calculo las ventas por tipo de día
sales_daytype = sales_df.groupby('DayType')['TotalPriceCalculated'].sum().reset_index()
```

```
sales_daytype
```

	DayType	TotalPriceCalculated
0	Weekday	3,080,351,837.91
1	Weekend	1,235,915,841.01

En esta imagen se puede verificar que las mayores compras se realizan durante los días de la semana. El importe total es de 3,080,351,837.91.

Como parte del proceso de feature engineering, en el mismo df que vienes trabajando, calcula dos nuevas columnas en el dataset de ventas:

La edad del empleado al momento de su contratación y años de experiencia al momento de realizar cada venta.

Utiliza las columnas BirthDate, HireDate (de la tabla employees) y SalesDate (de la tabla sales). Asegúrate de trabajar con fechas en formato adecuado.

<pre># Cambio el tipo del campo de la tabla employee  employees['BirthDate'] = pd.to_datetime(employees['BirthDate'], errors='coerce') employees['HireDate'] = pd.to_datetime(employees['HireDate'], errors='coerce')  sales_df = sales_df.merge(     employees[['EmployeeID', 'BirthDate', 'Gender', 'CityID', 'HireDate']],     left_on='SalesPersonID',     right_on='EmployeeID',     how='left' )</pre>														
sales_df														
ice	SalesDate	TransactionNumber	Price	TotalPriceCalculated	IsOutlier	Hour	Weekday	DayType	EmployeeID	BirthDate	Gender	CityID	HireDate	
.00	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.23	309.64	0	7.00	0.00	Weekday	6	1987-01-13	M	65	2013-06-22 13:20:18.080	
.00	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.55	437.82	0	16.00	4.00	Weekday	16	1951-07-07	M	28	2017-02-10 11:21:26.650	
.00	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.02	1,896.44	0	19.00	3.00	Weekday	13	1963-04-18	M	68	2011-12-12 10:43:52.940	
.00	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.32	1,236.01	0	14.00	5.00	Weekend	8	1956-12-13	M	18	2014-10-14 23:12:53.420	
.00	2018-02-12 15:37:03.940	4BGS0Z5OMAZ8NDAFHHP3	79.98	719.80	0	15.00	0.00	Weekday	10	1963-12-30	M	9	2012-07-23 15:02:12.640	
<pre># Calculo la edad al momento de su contratación  sales_df['Age_hire'] = (sales_df['HireDate'] - sales_df['BirthDate']).dt.days // 365</pre>														
sales_df.head()														
TotalPrice	SalesDate	TransactionNumber	Price	...	IsOutlier	Hour	Weekday	DayType	EmployeeID	BirthDate	Gender	CityID	HireDate	Age_hire
0.00	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.23	...	0	7.00	0.00	Weekday	6	1987-01-13	M	65	2013-06-22 13:20:18.080	26
0.00	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.55	...	0	16.00	4.00	Weekday	16	1951-07-07	M	28	2017-02-10 11:21:26.650	65
0.00	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.02	...	0	19.00	3.00	Weekday	13	1963-04-18	M	68	2011-12-12 10:43:52.940	48
0.00	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.32	...	0	14.00	5.00	Weekend	8	1956-12-13	M	18	2014-10-14 23:12:53.420	57
0.00	2018-02-12 15:37:03.940	4BGS0Z5OMAZ8NDAFHHP3	79.98	...	0	15.00	0.00	Weekday	10	1963-12-30	M	9	2012-07-23 15:02:12.640	48
<pre># Calculo años de experiencia del empleado  sales_df['Years_Experience'] = (sales_df['SalesDate'] - sales_df['HireDate']).dt.days // 365</pre>														
sales_df.head()														
ice	SalesDate	TransactionNumber	Price	...	Hour	Weekday	DayType	EmployeeID	BirthDate	Gender	CityID	HireDate	Age_hire	Years_Experience
0.00	2018-02-05 07:38:25.430	FQL4S94E4ME1EZFTG42G	44.23	...	7.00	0.00	Weekday	6	1987-01-13	M	65	2013-06-22 13:20:18.080	26	4.00
0.00	2018-02-02 16:03:31.150	12UGLX40DJ1A5DTFBHB8	62.55	...	16.00	4.00	Weekday	16	1951-07-07	M	28	2017-02-10 11:21:26.650	65	0.00
0.00	2018-05-03 19:31:56.880	5DT8RCPL87KI5EORO7B0	79.02	...	19.00	3.00	Weekday	13	1963-04-18	M	68	2011-12-12 10:43:52.940	48	6.00
0.00	2018-04-07 14:43:55.420	R3DR9MLD5NR76VO17ULE	81.32	...	14.00	5.00	Weekend	8	1956-12-13	M	18	2014-10-14 23:12:53.420	57	3.00
0.00	2018-02-12 15:37:03.940	4BGS0Z5OMAZ8NDAFHHP3	79.98	...	15.00	0.00	Weekday	10	1963-12-30	M	9	2012-07-23 15:02:12.640	48	5.00

```
# Label Encoding para variables con pocos valores ordenables o binarias
label_cols = ['DayType', 'Gender', 'Class']
label_encoders = {}

for col in label_cols:
    le = LabelEncoder()
    sales_df[col] = le.fit_transform(sales_df[col])
    label_encoders[col] = le

# One-Hot Encoding para variables categóricas nominales (no ordenables)
one_hot_cols = ['Resistant', 'IsAllergic', 'CategoryName']
df = pd.get_dummies(sales_df, columns=one_hot_cols, drop_first=True)
```

```
sales_df.head()
```

msactionNumber	Price	...	HireDate	Age_hire	Years_Experience	CategoryID	Class	Resistant	IsAllergic	VitalityDays	CategoryName	CityID_Customer
IE4ME1EZFTG42G	44.23	...	2013-06-22 13:20:18.080	26	4.00	1.00	0	Unknown	Unknown	41.00	Confections	54
40DJ1A5DTFBH8	62.55	...	2017-02-10 11:21:26.650	65	0.00	8.00	1	Durable	FALSE	90.00	Grain	71
PL87KI5EORO7B0	79.02	...	2011-12-12 10:43:52.940	48	6.00	11.00	2	Durable	TRUE	0.00	Produce	2
D5NR76VO17ULE	81.32	...	2014-10-14 23:12:53.420	57	3.00	6.00	0	Durable	TRUE	90.00	Seafood	45
OMAZ8NDAFHHP3	79.98	...	2012-07-23 15:02:12.640	48	5.00	9.00	1	Weak	FALSE	0.00	Poultry	82

## IMPORTANTE

- Prepara un único dataset definitivo para modelado que combine información relevante de las tablas disponibles.
- Incluye las features que se han calculado previamente.
- Aplica transformaciones adecuadas a las variables categóricas y a las variables numéricas (si lo consideras necesario) para dejar los datos listos para ser utilizados por un modelo de machine learning.
- Justifica las transformaciones realizadas. La variable objetivo es TotalPriceCalculated, por lo que debe quedar sin transformaciones.

## Conocimientos necesarios

- Manipulación de datos
- Limpieza
- Transformación
- Feature engineering
- Clasificación temporal

## Tech Stack necesario

- Python
- Pandas
- Jupyter Notebook

# Respuestas