Contenido de los ficheros

1. `raw_data/sales_data.csv`

```
Date,Product,Quantity,Price
2024-01-01,ProductA,10,100
2024-01-02,ProductB,5,50
2024-01-03,ProductC,20,200
```

2. `raw_data/customer_info.csv`

```
CustomerID,Name,Country,Email
1,John Doe,USA,johndoe@example.com
2,Alice Smith,UK,alicesmith@example.com
3,Roberto García,Spain,robertogarcia@example.com
```

3. `raw_data/transactions_data.csv`

```
TransactionID,CustomerID,Date,Amount
1001,1,2024-01-01,100
1002,2,2024-01-02,50
1003,3,2024-01-03,200
```

4. `processed_data/cleaned_sales_data.csv`

```
Date,Product,Quantity,Total
2024-01-01,ProductA,10,1000
2024-01-02,ProductB,5,250
2024-01-03,ProductC,20,4000
```

5. `processed_data/normalized_customer_info.csv`

```
CustomerID,Name,Country,Email
1,John Doe,USA,johndoe@example.com
2,Alice Smith,UK,alicesmith@example.com
3,Roberto García,Spain,robertogarcia@example.com
```

6. `processed_data/aggregated_transactions_data.csv`

```
Month,Total_Amount
2024-01,1000
2024-02,500
```

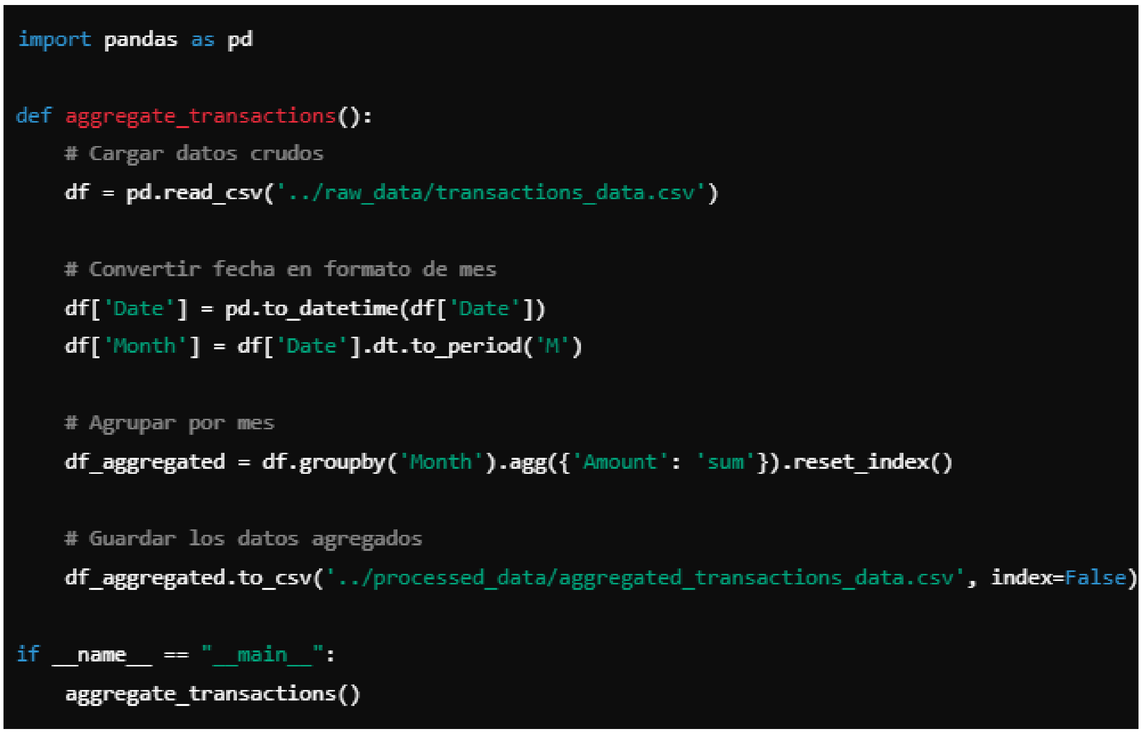
7. `scripts/data_cleaning.py`

```python
import pandas as pd

def clean_sales_data():
    # Cargar datos crudos
    df = pd.read_csv('../raw_data/sales_data.csv')

    # Eliminar filas con datos faltantes
    df_cleaned = df.dropna()

    # Guardar el archivo limpio
    df_cleaned.to_csv('../processed_data/cleaned_sales_data.csv', index=False)

if __name__ == "__main__":
    clean_sales_data()
```

8. `scripts/data_normalization.py`

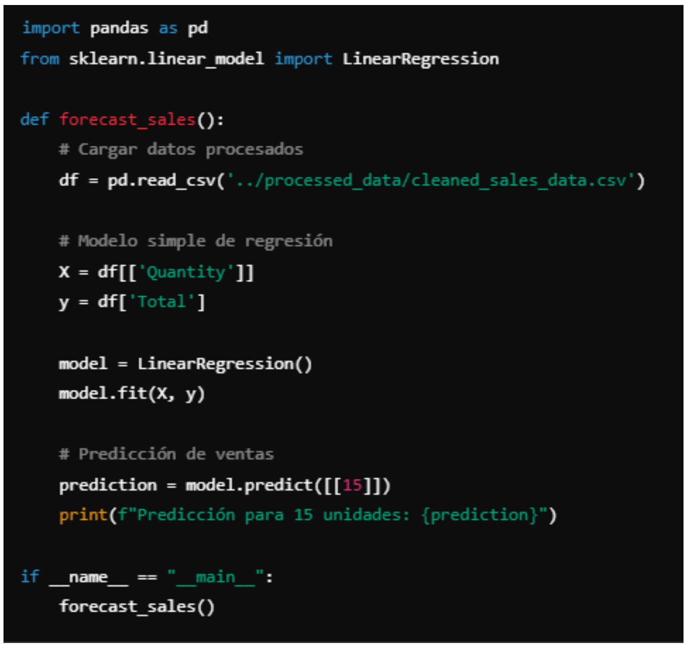
```python
import pandas as pd

def normalize_customer_info():
    # Cargar datos crudos
    df = pd.read_csv('../raw_data/customer_info.csv')

    # Normalizar nombres en mayúsculas
    df['Name'] = df['Name'].str.upper()

    # Guardar el archivo normalizado
    df.to_csv('../processed_data/normalized_customer_info.csv', index=False)

if __name__ == "__main__":
    normalize_customer_info()
```

9. `scripts/data_aggregation.py`



```python
import pandas as pd

def aggregate_transactions():
    # Cargar datos crudos
    df = pd.read_csv('../raw_data/transactions_data.csv')

    # Convertir fecha en formato de mes
    df['Date'] = pd.to_datetime(df['Date'])
    df['Month'] = df['Date'].dt.to_period('M')

    # Agrupar por mes
    df_aggregated = df.groupby('Month').agg({'Amount': 'sum'}).reset_index()

    # Guardar los datos agregados
    df_aggregated.to_csv('../processed_data/aggregated_transactions_data.csv', index=False)

if __name__ == "__main__":
    aggregate_transactions()
```

10. `models/sales_forecasting.py`



```python
import pandas as pd
from sklearn.linear_model import LinearRegression

def forecast_sales():
    # Cargar datos procesados
    df = pd.read_csv('../processed_data/cleaned_sales_data.csv')

    # Modelo simple de regresión
    X = df[['Quantity']]
    y = df['Total']

    model = LinearRegression()
    model.fit(X, y)

    # Predicción de ventas
    prediction = model.predict([[15]])
    print(f"Predicción para 15 unidades: {prediction}")

if __name__ == "__main__":
    forecast_sales()
```

11. `models/customer_segmentation.py`

```python
import pandas as pd
from sklearn.cluster import KMeans

def segment_customers():
    # Cargar datos procesados
    df = pd.read_csv('../processed_data/normalized_customer_info.csv')

    # Usar un modelo de clustering simple
    kmeans = KMeans(n_clusters=2)
    df['Cluster'] = kmeans.fit_predict(df[['CustomerID']])

    # Mostrar resultados de la segmentación
    print(df)

if __name__ == "__main__":
    segment_customers()
```

12. `models/transaction_analysis.py`

```python
import pandas as pd

def analyze_transactions():
    # Cargar datos procesados
    df = pd.read_csv('../processed_data/aggregated_transactions_data.csv')

    # Calcular estadísticas simples
    total = df['Total_Amount'].sum()
    print(f"Total de transacciones: {total}")

if __name__ == "__main__":
    analyze_transactions()
```

13. `config/database_config.json`

```json
{
    "host": "localhost",
    "port": 5432,
    "user": "admin",
    "password": "adminpass"
}
```

14. `config/pipeline_config.yaml`

```
ingestion:
  path: /data/raw
  format: csv
processing:
  steps:
    - clean
    - normalize
    - aggregate
```

15. `config/model_params.json`

```
{
    "learning_rate": 0.01,
    "n_estimators": 100
}
```

16. `.gitignore`

```
# Ignorar archivos de datos crudos
raw_data/*.csv

# Ignorar resultados procesados
processed_data/*.csv

# Ignorar configuraciones locales
config/database_config.json

# Ignorar archivos temporales y cache
*.log
*.tmp
__pycache__/

# Archivos generados por Python
*.pyc
```