Assignment 3 code

March 23, 2023

[ ]:
```
import pandas as pd # needed for most operation
import numpy as np # needed for some array operations
from sqlalchemy import create_engine, types # needed for DB connection
from datetime import datetime
from dateutil.relativedelta import relativedelta

# Load the data from CSV files
main_df = pd.read_csv('inputdata/main.csv', delimiter=';',␣
 ↪encoding="ISO-8859-1")
managers_df = pd.read_csv('inputdata/managers.csv', delimiter=';',␣
 ↪encoding="ISO-8859-1")
returns_df = pd.read_csv('inputdata/returns.csv', delimiter=';',␣
 ↪encoding="ISO-8859-1")
```

[ ]:
```
# Create the 'Late' column

# Convert date strings to datetime objects
main_df['Order Date'] = main_df['Order Date'].apply(lambda x: datetime.
 ↪strptime(x, '%d/%m/%y'))
main_df['Ship Date'] = main_df['Ship Date'].apply(lambda x: datetime.
 ↪strptime(x, '%d/%m/%y'))

# Calculate the number of days late and create the "Late" column
main_df['Late'] = main_df.apply(lambda x: 'Late' if (x['Ship Date'] - x['Order␣
 ↪Date']).days > 2 else 'NotLate', axis=1)
```

[ ]:
```
# Join the data from the three CSV files to create a single DataFrame
merged_df = main_df.merge(managers_df, on='Region')
merged_df = merged_df.merge(returns_df, on='Order ID')

# Replace commas with periods
merged_df['Sales'] = merged_df['Sales'].str.replace(',', '.').astype(float)
merged_df['Unit Price'] = merged_df['Unit Price'].str.replace(',', '.').
 ↪astype(float)
merged_df['Profit'] = merged_df['Profit'].str.replace(',', '.').astype(float)
merged_df['Shipping Cost'] = merged_df['Shipping Cost'].str.replace(',', '.').
 ↪astype(float)
```

[ ]:
```
# Create the "ReturnStatus" dimension table
return_status_df = pd.DataFrame({
    'returnstatusid': [0, 1],
    'returnvalue': ['NotReturned', 'Returned']
})
```

[ ]:
```
# Create the "product" dimension table
product_df = merged_df[['Product Name', 'Product Category', 'Product␣
 ↪Sub-Category']].drop_duplicates().reset_index(drop=True).reset_index().
 ↪rename(columns={'index': 'productid'})
product_df['productid'] += 1

# Create the "customers" dimension table
customers_df = merged_df[['Customer Name', 'Province', 'Region', 'Customer␣
 ↪Segment']].drop_duplicates().reset_index(drop=True).reset_index().
 ↪rename(columns={'index': 'customerid'})
customers_df['customerid'] += 1
```

[ ]:
```
# Create the "sales" fact table
sales_df = merged_df.merge(product_df, on=['Product Name', 'Product Category',␣
 ↪'Product Sub-Category'])
sales_df = sales_df.merge(customers_df, on=['Customer Name', 'Province',␣
 ↪'Region', 'Customer Segment'])
sales_df = sales_df.merge(return_status_df, left_on='Status',␣
 ↪right_on='returnvalue')
sales_df = sales_df.rename(columns={
    'Order Date': 'orderdate',
    'Order Quantity': 'orderquantity',
    'Sales': 'sales',
    'Unit Price': 'unitprice',
    'Profit': 'profit',
    'Shipping Cost': 'shippingcost',
    'Late': 'late'
}).drop(['Product Name', 'Product Category', 'Product Sub-Category', 'Customer␣
 ↪Name', 'Province', 'Region', 'Customer Segment', 'Status'], axis=1)
```

[ ]:
```
# Export the resulting tables to CSV files in an output directory (optional)
sales_df.to_csv('output/sales.csv', index=False)
product_df.to_csv('output/product.csv', index=False)
customers_df.to_csv('output/customers.csv', index=False)
return_status_df.to_csv('output/return_status.csv', index=False)
```

[ ]:
```
# Connect to database
driver='postgresql'
username='dab_ds22232a_46'
dbname=username # it is the same as the username
password='5wQ5aeeIp3Xaobd6'
```

```
server='bronto.ewi.utwente.nl'
port='5432'

# Creating the connection pool for SQLalchemy
engine = create_engine(f'{driver}://{username}:{password}@{server}:{port}/
 ↪{dbname}')
column_data_types = {
    'product_id': types.INTEGER,
    'customer_id': types.INTEGER,
    'orderdate': types.DATE,
    'returnstatusid': types.INTEGER,
    'late': types.TEXT,
    'sales': types.DOUBLE_PRECISION,
    'orderquantity': types.DOUBLE_PRECISION,
    'unitprice': types.DOUBLE_PRECISION,
    'profit': types.DOUBLE_PRECISION,
    'shippingcost': types.DOUBLE_PRECISION,
}
sales_df.to_sql('sales', engine,schema='ass3', index=False,␣
 ↪if_exists='replace', dtype=column_data_types)
product_df.to_sql('product', engine,schema='ass3', index=False,␣
 ↪if_exists='replace')
customers_df.to_sql('customers', engine,schema='ass3', index=False,␣
 ↪if_exists='replace')
return_status_df.to_sql('return_status', engine,schema='ass3', index=False,␣
 ↪if_exists='replace')
```

[ ]: 2