Creating Medallian architecture with Python and Cassandra

Pre-requisite

```
# Install the Cassandra python driver
!pip install cassandra-driver
```

```
Requirement already satisfied: cassandra-driver in /usr/local/lib/python3.10/dist-packages (3.29.2)
Requirement already satisfied: geomet<0.3,>=0.1 in /usr/local/lib/python3.10/dist-packages (from cassandra-driver) (0.2.1.post1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (8.1.7)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from geomet<0.3,>=0.1->cassandra-driver) (1.16.0)
```

```
# Import the necessary libraries
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
import json
```

Reference taken from youtube video link provided in assignment

1. Setting up the Connection

```
# This secure connect bundle is autogenerated when you download your SCB,
# if yours is different update the file name below
cloud_config= {
  'secure_connect_bundle': 'secure-connect-learner.zip'
}

# This token JSON file is autogenerated when you download your token,
# if yours is different update the file name below
with open("learner-token.json") as f:
    secrets = json.load(f)

CLIENT_ID = secrets["clientId"]
CLIENT_SECRET = secrets["secret"]

auth_provider = PlainTextAuthProvider(CLIENT_ID, CLIENT_SECRET)
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()

if session:
  print('Connected!')
else:
  print("An error occurred.")
```

```
WARNING:cassandra.cluster:Downgrading core protocol version from 66 to 65 for b320a3f4-4527-4d6d-8081-21626a7b1621-eu-west-1.db.astra.da
WARNING:cassandra.cluster:Downgrading core protocol version from 65 to 5 for b320a3f4-4527-4d6d-8081-21626a7b1621-eu-west-1.db.astra.dat
WARNING:cassandra.cluster:Downgrading core protocol version from 5 to 4 for b320a3f4-4527-4d6d-8081-21626a7b1621-eu-west-1.db.astra.data
Connected!
```

Loading and Reading Data from Cassandra

Load CSV data into Cassandra

Creating Bronze Table

```
def create_tables(session):
    # Bronze Table
    session.execute("""
    CREATE TABLE IF NOT EXISTS example.bronze_sales (
        id UUID PRIMARY KEY,
```

```
        region TEXT,
        country TEXT,
        item_type TEXT,
        sales_channel TEXT,
        order_priority TEXT,
        order_date TEXT,
        order_id TEXT,
        ship_date TEXT,
        units_sold INT,
        unit_price FLOAT,
        unit_cost FLOAT,
        total_revenue FLOAT,
        total_cost FLOAT,
        total_profit FLOAT
    );
    """)



    print("Tables created successfully.")

create_tables(session)
```

```
Tables created successfully.
```

Inserting data to created bronze table

```
# Load the CSV file
import uuid
import pandas as pd

file_path = 'sales_100.csv'
data = pd.read_csv(file_path)

def load_bronze_table(session, data):
    for _, row in data.iterrows():
        session.execute("""
        INSERT INTO example.bronze_sales (id, region, country, item_type, sales_channel, order_priority, order_date, order_id, ship_date, un
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);
        """, (uuid.uuid4(), row['Region'], row['Country'], row['Item Type'], row['Sales Channel'], row['Order Priority'], row['Order Date'],
    print("Bronze table loaded successfully.")

load_bronze_table(session, data)
```

```
Bronze table loaded successfully.
```

reading from bronze table

```
Start coding or generate with AI.
```

```
# Select the data from the users table
session = cluster.connect()
rowo = session.execute("SELECT * FROM example.bronze_sales;")

# Count the number of rows
row_count = sum(1 for _ in rowo)
print(f"Total Rows in example.bronze_sales: {row_count}")

rows = session.execute("SELECT * FROM example.bronze_sales")
for row in rows:
    print(row)

# Close the session
session.shutdown()
```

```
Total Rows in example.bronze_sales: 99
Row(id=UUID('f15c28a0-2bf4-433a-9315-1a0f06d5848f'), country='Montenegro', item_type='Clothes', order_date='9/4/2016', order_id='9025
Row(id=UUID('e76a83cb-faa0-4876-938e-a480089062be'), country='Turkmenistan', item_type='Vegetables', order_date='4/13/2015', order_id
Row(id=UUID('207e881f-1d87-44d1-a6bf-f5a26e1b0da3'), country='Egypt', item_type='Clothes', order_date='8/9/2016', order_id='882908809
Row(id=UUID('84d2a280-22a0-4db8-a7ac-d596251b3dcc'), country='Samoa ', item_type='Household', order_date='12/5/2016', order_id='93743
Row(id=UUID('73ea680e-9ce1-4eb9-896b-25d79a469953'), country='Singapore', item_type='Snacks', order_date='1/28/2013', order_id='17646
```

```
Row(id=UUID('fb55708a-67e7-429f-8cca-02ca8a530bb3'), country='Dominican Republic', item_type='Baby Food', order_date='8/25/2011', ord
Row(id=UUID('5106c5b9-1613-452a-a26a-17d3817dd910'), country='Solomon Islands', item_type='Household', order_date='12/17/2010', order
Row(id=UUID('5e38aeaa-e798-43c8-8023-7e169ad1fa2e'), country='Malawi', item_type='Meat', order_date='3/1/2016', order_id='450544869',
Row(id=UUID('bb475bfe-09c6-4830-9d5c-c6f08994e70b'), country='Slovakia', item_type='Beverages', order_date='10/26/2016', order_id='17
Row(id=UUID('6312b94e-7492-479b-919a-c9266b51ed4b'), country='Italy', item_type='Cereal', order_date='11/15/2011', order_id='29453085
Row(id=UUID('f550bbd5-6356-4d47-a2a5-782cb245c194'), country='United States of America', item_type='Personal Care', order_date='1/21/
Row(id=UUID('95d845d2-ef71-41e9-ade4-a028d5335c20'), country='Romania', item_type='Beverages', order_date='2/19/2012', order_id='7568
Row(id=UUID('bec6a35b-c350-4324-9795-188442b3963f'), country='Saint Lucia', item_type='Fruits', order_date='12/12/2015', order_id='73
Row(id=UUID('3ccd4ef4-daf1-47b3-aeb6-410d8547b081'), country='Tonga', item_type='Baby Food', order_date='5/11/2016', order_id='839094
Row(id=UUID('01e36895-2a43-4605-901b-41674caba117'), country='France', item_type='Cosmetics', order_date='12/7/2015', order_id='32466
Row(id=UUID('dc2c8266-cb3e-45b4-8dca-e495fd238664'), country='Sweden', item_type='Beverages', order_date='9/13/2012', order_id='26508
Row(id=UUID('c73205f3-5f9c-45b8-8996-d03a13b64cef'), country='Brunei', item_type='Cereal', order_date='5/28/2013', order_id='15384234
Row(id=UUID('bbf0671d-becb-4678-b8d7-713d508e2002'), country='Monaco', item_type='Beverages', order_date='5/29/2016', order_id='46739
Row(id=UUID('68e21a6f-a1aa-47b6-a2b4-572def450e2a'), country='Ghana', item_type='Fruits', order_date='5/29/2016', order_id='496523940
Row(id=UUID('c9879daf-ae1d-44a8-969d-f021e1f55e99'), country='Ghana', item_type='Office Supplies', order_date='3/23/2017', order_id='
Row(id=UUID('50cadb1a-9a8c-48dc-8857-e8e82d15741a'), country='The Bahamas', item_type='Fruits', order_date='6/18/2010', order_id='488
Row(id=UUID('aa46f649-c146-4adf-ac90-308d79905677'), country='Albania', item_type='Personal Care', order_date='2/6/2016', order_id='1
Row(id=UUID('15f8444b-aa0f-4871-a51d-78f4d8338e31'), country='Rwanda', item_type='Fruits', order_date='9/5/2012', order_id='699160754
Row(id=UUID('a3d6c8d6-663a-43f9-83d2-7d987d93f97e'), country='Papua New Guinea', item_type='Clothes', order_date='6/20/2011', order_i
Row(id=UUID('b040496d-8aee-4f86-9d6b-03403d6b6c1b'), country='Ethiopia', item_type='Cosmetics', order_date='7/7/2011', order_id='8077
Row(id=UUID('0a723873-1abc-4e08-ae00-6c6d564e03a5'), country='Uganda', item_type='Personal Care', order_date='6/19/2014', order_id='5
Row(id=UUID('9d34e578-8b33-458f-9c08-934a0ba50abb'), country='Serbia', item_type='Clothes', order_date='7/6/2016', order_id='92513664
Row(id=UUID('b6d8db5a-dc86-4a5c-ab66-c1ceb327909a'), country='Morocco', item_type='Clothes', order_date='9/14/2013', order_id='667593
Row(id=UUID('e62826e1-c16b-4f62-9ea4-544e254fde4a'), country='Togo', item_type='Cosmetics', order_date='9/8/2015', order_id='56368173
Row(id=UUID('c3a5400c-f3ef-432b-9d08-5345ee8afe5e'), country='Vietnam', item_type='Personal Care', order_date='4/4/2010', order_id='3
Row(id=UUID('57535c31-6059-4166-8634-6af03ee98ba8'), country='Vanuatu', item_type='Cereal', order_date='6/20/2014', order_id='5723356
Row(id=UUID('0086ec2b-cd23-4abb-84b4-d89f5ef0cb4d'), country='Romania', item_type='Cereal', order_date='4/16/2015', order_id='6331342
Row(id=UUID('b77bab25-932f-4857-9bf6-8e3b1ebf5076'), country='Tanzania', item_type='Cosmetics', order_date='5/23/2016', order_id='739
Row(id=UUID('7002d3b3-13b8-4991-b7c4-e148aee2ef0c'), country='Belgium', item_type='Personal Care', order_date='11/1/2011', order_id='
Row(id=UUID('65269f74-b23b-4064-9811-5fd1e29f3b32'), country='Dominica', item_type='Household', order_date='11/19/2011', order_id='27
Row(id=UUID('83c2736f-b95e-4cd4-a68e-76901f0fa149'), country='Moldova ', item_type='Fruits', order_date='1/27/2013', order_id='180908
Row(id=UUID('fbe8f4e5-84d9-4402-8e1e-ff23ca20543d'), country='Kuwait', item_type='Household', order_date='6/13/2011', order_id='45938
Row(id=UUID('52fc10e6-2b61-4f16-9cdb-618638eaf560'), country='Indonesia', item_type='Personal Care', order_date='10/22/2012', order_i
Row(id=UUID('4cba2954-6886-4fd6-8ea1-46374149246f'), country='South Korea', item_type='Meat', order_date='3/16/2016', order_id='29787
Row(id=UUID('94361aff-24ad-40fc-ba9a-7dec3ec7c010'), country='Belize', item_type='Personal Care', order_date='6/14/2015', order_id='3
Row(id=UUID('f2d08ce9-9090-4fc3-aee6-a5b218a99f94'), country='Sao Tome and Principe', item_type='Clothes', order_date='11/11/2015', o
Row(id=UUID('c4dfd6e4-adc5-4315-9a94-26460b00ee13'), country='Liberia', item_type='Baby Food', order_date='6/6/2015', order_id='14663
Row(id=UUID('20a6cb50-c695-4387-b29a-7b453de2cf7c'), country='Thailand', item_type='Fruits', order_date='9/30/2012', order_id='434008
Row(id=UUID('e11053e6-8a62-419d-86e6-e83e38b281be'), country='Iceland', item_type='Baby Food', order_date='10/2/2010', order_id='6782
Row(id=UUID('e109beb2-6a61-4074-9f6f-e56c0fbfd6b3'), country='Taiwan', item_type='Cereal', order_date='4/11/2010', order_id='49807189
Row(id=UUID('9a5797c9-f9cb-4e88-ae96-a6f814c48881'), country='Papua New Guinea', item_type='Meat', order_date='5/15/2015', order_id='
Row(id=UUID('b4d841fb-7afd-41d5-b793-065c1e372e94'), country='Greece', item_type='Cereal', order_date='8/22/2015', order_id='88712438
Row(id=UUID('88eb3c7d-767a-48dc-a7d0-d433a3fc40a9'), country='Taiwan', item_type='Fruits', order_date='2/9/2014', order_id='732588374
Row(id=UUID('d04e57ad-f114-4c01-aad4-46fcaa5f3303'), country='Lebanon', item_type='Meat', order_date='3/8/2017', order_id='704205024'
Row(id=UUID('21337629-8e76-4419-87b1-40d5da681685'), country='Switzerland', item_type='Office Supplies', order_date='7/29/2014', orde
Row(id=UUID('d64630c4-ea19-481e-a7e5-c7da980073ce'), country='Canada', item_type='Cosmetics', order_date='5/9/2011', order_id='368977
Row(id=UUID('a048c341-74a5-4155-931e-61e6dc4c6277'), country='South Africa', item_type='Fruits', order_date='7/27/2012', order_id='44
Row(id=UUID('03fc3e84-8e1d-43bf-9468-db7a13a14157'), country='Burundi', item_type='Beverages', order_date='3/9/2011', order_id='52927
Row(id=UUID('fc75fc90-9e0c-456d-8b60-0f06fb83de32'), country='Zimbabwe', item_type='Office Supplies', order_date='3/28/2011', order_i
Row(id=UUID('218ceea9-0c3f-44fa-a9a4-891d28f294b5'), country='United Kingdom', item_type='Cosmetics', order_date='5/1/2015', order_id
Row(id=UUID('1ba5920c-9c40-4243-b99f-2ee3253546df'), country='Israel', item_type='Beverages', order_date='9/8/2013', order_id='371502
```

Data Profiing

```
session = cluster.connect()
session.set_keyspace('example')  # Replace 'example' with your keyspace

# Query data from Bronze table
rows = session.execute("SELECT * FROM bronze_sales;")

# Convert the data to a Pandas DataFrame
data = []
for row in rows:
    data.append(row._asdict())  # Convert Cassandra rows to dictionary
df = pd.DataFrame(data)

# Display the first few rows for verification
print("Bronze Data:\n", df.head())
```

```
Bronze Data:
                                     id       country   item_type order_date  \
0  f15c28a0-2bf4-433a-9315-1a0f06d5848f    Montenegro     Clothes   9/4/2016
1  e76a83cb-faa0-4876-938e-a480089062be  Turkmenistan  Vegetables  4/13/2015
2  207e881f-1d87-44d1-a6bf-f5a26e1b0da3         Egypt     Clothes   8/9/2016
3  84d2a280-22a0-4db8-a7ac-d596251b3dcc        Samoa    Household  12/5/2016
4  73ea680e-9ce1-4eb9-896b-25d79a469953     Singapore      Snacks  1/28/2013

    order_id order_priority                      region sales_channel  \
0  902511680              M                      Europe       Offline
1  116205585              M                        Asia        Online
```

```
2  882908809              C  Middle East and North Africa        Online
3  937431466              L          Australia and Oceania        Online
4  176461303              C                           Asia        Online

   ship_date    total_cost   total_profit  total_revenue   unit_cost  \
0   9/8/2016  7.587328e+04  155472.484375   2.313458e+05   35.840000
1   6/2/2015  6.065031e+05  421077.093750   1.027580e+06   90.930000
2  8/24/2016  4.272128e+04   87540.476562   1.302618e+05   35.840000
3  12/8/2016  2.842869e+06  937534.625000   3.780404e+06  502.540009
4   2/7/2013  7.479494e+05  423254.625000   1.171204e+06   97.440002

   unit_price  units_sold
0  109.279999        2117
1  154.059998        6670
2  109.279999        1192
3  668.270020        5657
4  152.580002        7676
```

```
# Check for null values
print("\nNull Value Counts:")
print(df.isnull().sum())

# Check for zero values in numeric columns
numeric_cols = ['units_sold', 'unit_price', 'unit_cost', 'total_revenue', 'total_cost', 'total_profit']
zero_counts = (df[numeric_cols] == 0).sum()
print("\nZero Value Counts in Numeric Columns:")
print(zero_counts)

# Descriptive statistics
print("\nDescriptive Statistics:")
print(df.describe())
```

```
Null Value Counts:
id                0
country           0
item_type         0
order_date        0
order_id          0
order_priority    0
region            0
sales_channel     0
ship_date         0
total_cost        0
total_profit      0
total_revenue     0
unit_cost         0
unit_price        0
units_sold        0
dtype: int64

Zero Value Counts in Numeric Columns:
units_sold       0
unit_price       0
unit_cost        0
total_revenue    0
total_cost       0
total_profit     0
dtype: int64

Descriptive Statistics:
         total_cost  total_profit  total_revenue   unit_cost  unit_price  \
count  9.900000e+01  9.900000e+01   9.900000e+01   99.000000   99.000000
mean   1.044069e+06  4.174464e+05   1.461515e+06  191.126769  271.181016
std    1.337149e+06  4.503887e+05   1.724090e+06  185.365590  233.458916
min    9.155160e+03  3.188430e+03   1.234359e+04    6.920000    9.330000
25%    9.318011e+04  5.877198e+04   1.597606e+05   35.840000   81.730003
50%    3.924920e+05  2.517372e+05   6.525323e+05  117.110001  205.699997
75%    1.596572e+06  6.588507e+05   2.172734e+06  364.690002  437.200012
max    5.051690e+06  1.681149e+06   6.666662e+06  524.960022  668.270020

        units_sold
count    99.000000
mean   4998.080808
std    2992.013804
min     259.000000
25%    2473.500000
50%    4806.000000
75%    7722.000000
max    9976.000000
```

I do not see any nulls or zeros and any edge cases here so im moving to change the data types and creating silver tables.

Creating Silver table

```
Start coding or generate with AI.
```

```
def create_tables(session):
    session.execute("""
    CREATE TABLE IF NOT EXISTS example.silver_sales (
        id UUID PRIMARY KEY,
        region TEXT,
        country TEXT,
        item_type TEXT,
        sales_channel TEXT,
        order_priority TEXT,
        order_date DATE,
        order_id BIGINT,
        ship_date DATE,
        units_sold INT,
        unit_price FLOAT,
        unit_cost FLOAT,
        total_revenue FLOAT,
        total_cost FLOAT,
        total_profit FLOAT
    );
    """)



    print("Tables created successfully.")

create_tables(session)
```

```
Tables created successfully.
```

loading data to silver table

```
import uuid
from datetime import datetime

# Convert string to date (YYYY-MM-DD format)
def convert_to_date(date_string):
    try:
        return datetime.strptime(date_string, "%m/%d/%Y").date()  # Assuming format is MM/DD/YYYY
    except ValueError:
        return None  # Return None if the date format is incorrect

# Convert order_id to BIGINT (long integer)
def convert_to_bigint(order_id_string):
    try:
        return int(order_id_string)
    except ValueError:
        return None  # Return None if order_id is invalid

# Insert cleaned data with converted order_id and dates into Silver table
for _, row in df.iterrows():
    order_date = convert_to_date(row['order_date'])
    ship_date = convert_to_date(row['ship_date'])
    order_id = convert_to_bigint(row['order_id'])  # Convert order_id to BIGINT

    session.execute("""
    INSERT INTO example.silver_sales (
        id, region, country, item_type, sales_channel, order_priority, order_date,
        order_id, ship_date, units_sold, unit_price, unit_cost, total_revenue, total_cost, total_profit
    ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);
    """, (
        uuid.uuid4(),
        row['region'],
        row['country'],
        row['item_type'],
```

```
            row['sales_channel'],
            row['order_priority'],
            order_date,  # Store the converted date
            order_id,  # Store the converted BIGINT order_id
            ship_date,  # Store the converted date
            row['units_sold'],
            row['unit_price'],
            row['unit_cost'],
            row['total_revenue'],
            row['total_cost'],
            row['total_profit']
        ))

print("Data loaded to silver table")
```

```
Data loaded to silver table
```

Data in silver table

```
rows = session.execute("SELECT * FROM example.silver_sales")
for row in rows:
    print(row)
```

```
Row(id=UUID('eaa3e33c-5d1e-4e12-9601-92dcfef59149'), country='Thailand', item_type='Fruits', order_date=Date(15613), order_id=4340083
Row(id=UUID('fc8f3f64-7f1f-4f0a-bb7e-ae598a9e0a1d'), country='Pakistan', item_type='Meat', order_date=Date(16067), order_id=500371730
Row(id=UUID('df091210-e821-4d46-ab25-846b60c83e86'), country='Slovakia', item_type='Beverages', order_date=Date(17100), order_id=1745
Row(id=UUID('70a45218-f22e-4b9b-8fef-5a71d29399fc'), country='Taiwan', item_type='Fruits', order_date=Date(16110), order_id=732588374
Row(id=UUID('9396b3f8-b845-47de-91da-361c4ffe44c0'), country='Morocco', item_type='Clothes', order_date=Date(15962), order_id=6675935
Row(id=UUID('b3257394-ca8b-406e-a21a-fe727ade21a3'), country='Taiwan', item_type='Cereal', order_date=Date(14710), order_id=498071897
Row(id=UUID('c1ac2521-9dad-4eb4-a185-4f5da1c30c44'), country='Papua New Guinea', item_type='Clothes', order_date=Date(15145), order_i
Row(id=UUID('f8cfaa64-a57b-4562-b5a4-2a9afc893875'), country='Malaysia', item_type='Beverages', order_date=Date(16387), order_id=9558
Row(id=UUID('f5282ede-1a27-4367-a6ca-be24c5c7702e'), country='Vanuatu', item_type='Cereal', order_date=Date(16241), order_id=57233561
Row(id=UUID('4370f53a-1103-49c7-b7ca-87dc678e62dc'), country='Nepal', item_type='Meat', order_date=Date(16122), order_id=179137074, o
Row(id=UUID('8859980e-d2ec-4aca-b017-93ca331824cb'), country='Togo', item_type='Cosmetics', order_date=Date(16686), order_id=56368173
Row(id=UUID('f04cd705-767d-400f-92d0-f6e8cd8203fb'), country='Democratic Republic of the Congo', item_type='Cosmetics', order_date=Da
Row(id=UUID('6d1a8eb9-15f4-4e1c-869b-bc033e744a0d'), country='Monaco', item_type='Beverages', order_date=Date(16950), order_id=467399
Row(id=UUID('79094181-9552-4192-bc7b-8fbb6b4687ec'), country='Kuwait', item_type='Household', order_date=Date(15138), order_id=459386
Row(id=UUID('2ceeaee5-d105-48c9-8ae6-6407ca80b7be'), country='Dominican Republic', item_type='Baby Food', order_date=Date(15211), ord
Row(id=UUID('560e2c8e-ab7b-47bb-a868-2c5b053b832d'), country='Finland', item_type='Household', order_date=Date(16815), order_id=75725
Row(id=UUID('79ee4edf-a56b-47c5-bcdd-419662c6aade'), country='Belgium', item_type='Personal Care', order_date=Date(15279), order_id=2
Row(id=UUID('8affcd91-58a9-4c91-8ff1-1f839aa41d55'), country='China', item_type='Office Supplies', order_date=Date(16841), order_id=1
Row(id=UUID('567fa20c-1873-4b10-aadd-96337d4caf3d'), country='Serbia', item_type='Clothes', order_date=Date(16988), order_id=92513664
Row(id=UUID('8c9caf32-029f-4cfc-b61e-ef0b20169f75'), country='Belize', item_type='Personal Care', order_date=Date(16600), order_id=31
Row(id=UUID('c756e26e-54fd-43a7-a418-efe4de38e3c1'), country='Moldova ', item_type='Fruits', order_date=Date(15732), order_id=1809086
Row(id=UUID('b393690b-d549-4640-8478-d6ee54420519'), country='Malawi', item_type='Meat', order_date=Date(16861), order_id=450544869,
Row(id=UUID('41534b0a-d570-4184-97f2-8da0e04bd3be'), country='Tanzania', item_type='Beverages', order_date=Date(17135), order_id=6598
Row(id=UUID('a040fb6e-9254-44f2-9252-816d14f18586'), country='Haiti', item_type='Office Supplies', order_date=Date(14974), order_id=4
Row(id=UUID('1a5260cf-c330-4bc3-83fb-409f9b190455'), country='Italy', item_type='Cereal', order_date=Date(15293), order_id=294530856,
Row(id=UUID('bee61c6b-cd3c-4356-ae33-c46173bccbcf'), country='Ghana', item_type='Clothes', order_date=Date(15280), order_id=791778934
Row(id=UUID('9f7f6c51-2c10-44ed-b406-cb968299c05d'), country='Indonesia', item_type='Personal Care', order_date=Date(15635), order_id
Row(id=UUID('d8f690b9-99e2-4a42-99de-347f49566bfa'), country='New Zealand', item_type='Beverages', order_date=Date(15624), order_id=9
Row(id=UUID('a6d5a5a6-1a75-4e0f-99f2-6d72f1696e0d'), country='Burundi', item_type='Beverages', order_date=Date(15042), order_id=52927
Row(id=UUID('d2d04284-21af-47fd-8dbb-a62966616246'), country='Vietnam', item_type='Personal Care', order_date=Date(14703), order_id=3
Row(id=UUID('a63da243-ed55-4999-9456-bc8964ec9eb4'), country='Italy', item_type='Cereal', order_date=Date(16769), order_id=887409770,
Row(id=UUID('61db957b-b4fd-4241-91dc-f4315488a84e'), country='Papua New Guinea', item_type='Meat', order_date=Date(16570), order_id=9
Row(id=UUID('a000db98-3acc-4be2-b485-996990b1ba1f'), country='Dominica', item_type='Beverages', order_date=Date(15503), order_id=4380
Row(id=UUID('5ea600bb-4327-4127-8057-08e93107512e'), country='Montenegro', item_type='Clothes', order_date=Date(17048), order_id=9025
Row(id=UUID('db8588b0-aac9-4411-bd69-6276bd975dc8'), country='Dominica', item_type='Household', order_date=Date(15297), order_id=2749
Row(id=UUID('00071311-bf5b-4800-8785-ed14f1cac975'), country='Iceland', item_type='Baby Food', order_date=Date(14884), order_id=67823
Row(id=UUID('f6753f5d-ddd9-4194-98a0-e6f7ab1e40ff'), country='Zimbabwe', item_type='Office Supplies', order_date=Date(15061), order_i
Row(id=UUID('8df713d1-c232-4571-9ec9-a0df1830d8ab'), country='Romania', item_type='Beverages', order_date=Date(15389), order_id=75683
Row(id=UUID('70a57170-5b12-42ec-80a3-1b9d5ba4e3a2'), country='Vanuatu', item_type='Fruits', order_date=Date(16012), order_id=57199786
Row(id=UUID('b4a0504f-1acb-40b9-a7b6-4d3c201849e0'), country='United States of America', item_type='Personal Care', order_date=Date(1
Row(id=UUID('ffc0f4a1-9657-4553-bb18-fde009e93f30'), country='Oman', item_type='Cosmetics', order_date=Date(14942), order_id=35857084
Row(id=UUID('d31005a6-c8cf-4191-a3f0-0b0b63767642'), country='Japan', item_type='Household', order_date=Date(16267), order_id=7477962
Row(id=UUID('684c00ed-e75b-438e-80ef-405b1807a111'), country='Ghana', item_type='Office Supplies', order_date=Date(17248), order_id=6
Row(id=UUID('0f9d1f3c-8796-4d3e-baa7-ddfd252be353'), country='Seychelles ', item_type='Beverages', order_date=Date(15723), order_id=4
Row(id=UUID('8c20c282-d036-4297-ab33-0b54ed681a84'), country='Rwanda', item_type='Fruits', order_date=Date(15588), order_id=699160754
Row(id=UUID('86b55f7d-fd68-446b-9699-f75a73877329'), country='Switzerland', item_type='Office Supplies', order_date=Date(16280), orde
Row(id=UUID('2ee2a716-e5c9-4c19-868e-553b9b9a80d0'), country='Albania', item_type='Baby Food', order_date=Date(15540), order_id=75252
Row(id=UUID('d1cefa9f-3b70-4475-abcf-5a1836b7c9ff'), country='Singapore', item_type='Snacks', order_date=Date(15733), order_id=176461
Row(id=UUID('b6f87851-ded8-4635-ad62-89f4214537de'), country='Nicaragua', item_type='Household', order_date=Date(16736), order_id=573
Row(id=UUID('64ae4e34-a28a-45d0-80c6-88fca4a0dfaa'), country='Egypt', item_type='Clothes', order_date=Date(17022), order_id=882908809
Row(id=UUID('3130203a-b636-4a01-b07a-e2100c8c70c3'), country='Netherlands', item_type='Fruits', order_date=Date(17124), order_id=8450
Row(id=UUID('4e78a678-d7d6-40b0-91ed-9091de9ed6d2'), country='The Gambia', item_type='Fruits', order_date=Date(15298), order_id=86286
Row(id=UUID('e6905560-9a85-47c6-9f33-ad003dd67e0b'), country='Turkmenistan', item_type='Meat', order_date=Date(17286), order_id=75761
Row(id=UUID('5b9c3ed9-78c4-4912-8d80-6244b108dc34'), country='France', item_type='Cosmetics', order_date=Date(16776), order_id=324669
Row(id=UUID('caad5fbd-ceaf-464f-8ab2-74cbb8c02cdf'), country='Tonga', item_type='Baby Food', order_date=Date(16932), order_id=8390943
Row(id=UUID('924c473a-d6dc-421c-b27b-105a4b65ecef'), country='Italy', item_type='Office Supplies', order_date=Date(15062), order_id=7
Row(id=UUID('0cdfe83f-3af4-451d-adb0-74534a8a2e45'), country='Sao Tome and Principe', item_type='Clothes', order_date=Date(16750), or
```

Gold Tables

```
session.execute("""
CREATE TABLE IF NOT EXISTS example.gold_sales_by_region (
    region TEXT PRIMARY KEY,
    total_revenue FLOAT,
    total_profit FLOAT
);
""")
print("Gold table 1: sales_by_region created successfully.")
```

```
Gold table 1: sales_by_region created successfully.
```

```
from collections import defaultdict

# Aggregate total revenue and profit by region
region_sales = defaultdict(lambda: {'total_revenue': 0, 'total_profit': 0})

# Retrieve data from Silver table
rows = session.execute("SELECT region, total_revenue, total_profit FROM example.silver_sales")
for row in rows:
    region_sales[row.region]['total_revenue'] += row.total_revenue
    region_sales[row.region]['total_profit'] += row.total_profit

# Insert aggregated data into Gold table
for region, sales_data in region_sales.items():
    session.execute("""
    INSERT INTO example.gold_sales_by_region (region, total_revenue, total_profit)
    VALUES (%s, %s, %s);
    """, (region, sales_data['total_revenue'], sales_data['total_profit']))

print("Gold table 1: Data for total_sales_by_region inserted successfully.")
```

```
Gold table 1: Data for total_sales_by_region inserted successfully.
```

```
rows = session.execute("SELECT * FROM example.gold_sales_by_region")
for row in rows:
    print(row.region, row.total_profit, row.total_revenue)
```

```
Australia and Oceania 3486940.0 10711258.0
Europe 11267281.0 34964748.0
Middle East and North Africa 6514262.0 24765128.0
Central America and the Caribbean 4252300.0 17570836.0
Asia 6749896.0 28840812.0
Sub-Saharan Africa 7651892.0 24225438.0
North America 1404621.5 3611757.5
```

```
Start coding or generate with AI.
```

```
session.execute("""
CREATE TABLE IF NOT EXISTS example.gold_top_selling_products (
    item_type TEXT PRIMARY KEY,
    total_revenue FLOAT,
    total_units_sold INT
);
""")
print("Gold table 2: top_selling_products created successfully.")
```

```
Gold table 2: top_selling_products created successfully.
```

```
top_selling = defaultdict(lambda: {'total_revenue': 0, 'total_units_sold': 0})

# Aggregate total revenue and units sold by product type
rows = session.execute("SELECT item_type, total_revenue, units_sold FROM example.silver_sales")
for row in rows:
    top_selling[row.item_type]['total_revenue'] += row.total_revenue
    top_selling[row.item_type]['total_units_sold'] += row.units_sold

# Insert aggregated data into Gold table
for item_type, sales_data in top_selling.items():
```

```
        session.execute("""
        INSERT INTO example.gold_top_selling_products (item_type, total_revenue, total_units_sold)
        VALUES (%s, %s, %s);
        """, (item_type, sales_data['total_revenue'], sales_data['total_units_sold']))

    print("Gold table 2: Data for top_selling_products inserted successfully.")
```

```
Gold table 2: Data for top_selling_products inserted successfully.
```

```
    rows = session.execute("SELECT * FROM example.gold_top_selling_products")
    for row in rows:
        print(row.item_type, row.total_revenue, row.total_units_sold)
```

```
Household 38519084.0 57640
Office Supplies 27880904.0 42814
Vegetables 1135114.125 7368
Snacks 2193642.75 14377
Personal Care 3191147.75 39045
Meat 21278866.0 50437
Fruits 615033.625 65920
Beverages 2145024.75 45206
Cereal 9416123.0 45776
Cosmetics 28727100.0 65707
Baby Food 5200564.0 20372
Clothes 4387373.5 40148
```

```
    session.execute("""
    CREATE TABLE IF NOT EXISTS example.gold_sales_performance (
        year INT,
        month INT,
        total_revenue FLOAT,
        total_profit FLOAT,
        PRIMARY KEY (year, month)
    );
    """)
    print("Gold table 3: sales_performance created successfully.")
```

```
Gold table 3: sales_performance created successfully.
```

```
    from datetime import datetime

    # Aggregate sales performance by year and month
    monthly_sales = defaultdict(lambda: {'total_revenue': 0, 'total_profit': 0})

    # Convert order_date to year and month, and aggregate revenue and profit
    rows = session.execute("SELECT order_date, total_revenue, total_profit FROM example.silver_sales")
    for row in rows:
        if row.order_date:
            # Convert order_date to a Python datetime object
            order_date = datetime.strptime(str(row.order_date), '%Y-%m-%d')  # Ensure correct format

            # Extract year and month
            year, month = order_date.year, order_date.month

            # Aggregate sales data by year and month
            monthly_sales[(year, month)]['total_revenue'] += row.total_revenue
            monthly_sales[(year, month)]['total_profit'] += row.total_profit

    # Insert aggregated data into Gold table
    for (year, month), sales_data in monthly_sales.items():
        session.execute("""
        INSERT INTO example.gold_sales_performance (year, month, total_revenue, total_profit)
        VALUES (%s, %s, %s, %s);
        """, (year, month, sales_data['total_revenue'], sales_data['total_profit']))

    print("Gold table 3: Data for sales_performance inserted successfully.")
```

```
Gold table 3: Data for sales_performance inserted successfully.
```

```
    rows = session.execute("SELECT * FROM example.gold_sales_performance")
    for row in rows:
        print(row)
```

```
Row(year=2014, month=2, total_profit=562533.125, total_revenue=4081224.75)
Row(year=2014, month=3, total_profit=1592127.625, total_revenue=4003440.5)
Row(year=2014, month=6, total_profit=514581.84375, total_revenue=1205442.0)
Row(year=2014, month=7, total_profit=2102895.5, total_revenue=9280963.0)
Row(year=2014, month=10, total_profit=23133.58984375, total_revenue=89558.671875)
Row(year=2014, month=11, total_profit=143351.640625, total_revenue=434357.3125)
Row(year=2010, month=4, total_profit=1032559.3125, total_revenue=2585495.25)
Row(year=2010, month=6, total_profit=11423.400390625, total_revenue=44224.19921875)
Row(year=2010, month=8, total_profit=715456.4375, total_revenue=2884921.5)
Row(year=2010, month=10, total_profit=236007.3125, total_revenue=628499.375)
Row(year=2010, month=11, total_profit=1380006.25, total_revenue=3470056.5)
Row(year=2010, month=12, total_profit=959274.25, total_revenue=4159723.5)
Row(year=2012, month=2, total_profit=71738.4609375, total_revenue=217368.453125)
Row(year=2012, month=6, total_profit=1311052.375, total_revenue=6552552.0)
Row(year=2012, month=7, total_profit=1186156.0, total_revenue=3100162.0)
Row(year=2012, month=9, total_profit=57264.84375, total_revenue=188951.875)
Row(year=2012, month=10, total_profit=1178729.375, total_revenue=3256076.75)
Row(year=2012, month=11, total_profit=405388.8125, total_revenue=603225.625)
Row(year=2017, month=3, total_profit=614764.0, total_revenue=4283459.5)
Row(year=2017, month=4, total_profit=297783.1875, total_revenue=2196359.25)
Row(year=2017, month=5, total_profit=41273.28125, total_revenue=61415.359375)
Row(year=2015, month=2, total_profit=1300347.25, total_revenue=4493491.0)
Row(year=2015, month=4, total_profit=1071061.875, total_revenue=2536801.0)
Row(year=2015, month=5, total_profit=201069.0625, total_revenue=605694.0)
Row(year=2015, month=6, total_profit=215455.625, total_revenue=626742.8125)
Row(year=2015, month=8, total_profit=768429.6875, total_revenue=1784241.75)
Row(year=2015, month=9, total_profit=835619.25, total_revenue=2101183.25)
Row(year=2015, month=10, total_profit=1291202.375, total_revenue=5206491.5)
Row(year=2015, month=11, total_profit=479626.75, total_revenue=944631.0)
Row(year=2015, month=12, total_profit=1032718.4375, total_revenue=2604084.5)
Row(year=2011, month=1, total_profit=228973.21875, total_revenue=746767.0)
Row(year=2011, month=2, total_profit=1681149.0, total_revenue=4227287.0)
Row(year=2011, month=3, total_profit=1637996.75, total_revenue=8331209.5)
Row(year=2011, month=5, total_profit=1297765.625, total_revenue=3263260.75)
Row(year=2011, month=6, total_profit=910676.6875, total_revenue=1973257.5)
Row(year=2011, month=7, total_profit=115101.9375, total_revenue=289426.40625)
Row(year=2011, month=8, total_profit=26265.640625, total_revenue=69946.71875)
Row(year=2011, month=9, total_profit=1756572.25, total_revenue=7082993.5)
Row(year=2011, month=10, total_profit=794996.25, total_revenue=4100669.25)
Row(year=2011, month=11, total_profit=2171467.75, total_revenue=6907926.5)
Row(year=2016, month=1, total_profit=1457223.875, total_revenue=5793560.5)
Row(year=2016, month=2, total_profit=769781.3125, total_revenue=3897266.5)
Row(year=2016, month=3, total_profit=699842.0, total_revenue=5161824.0)
Row(year=2016, month=5, total_profit=1924869.125, total_revenue=4944264.0)
Row(year=2016, month=7, total_profit=539637.125, total_revenue=802989.4375)
Row(year=2016, month=8, total_profit=87540.4765625, total_revenue=130261.7578125)
Row(year=2016, month=9, total_profit=155472.484375, total_revenue=231345.765625)
Row(year=2016, month=10, total_profit=290615.59375, total_revenue=528379.625)
Row(year=2016, month=11, total_profit=324073.09375, total_revenue=900296.4375)
Row(year=2016, month=12, total_profit=937534.625, total_revenue=3780403.5)
Row(year=2013, month=1, total_profit=441607.40625, total_revenue=1234388.625)
Row(year=2013, month=5, total_profit=374026.96875, total_revenue=868465.375)
Row(year=2013, month=9, total_profit=486924.0625, total_revenue=933620.3125)
Row(year=2013, month=11, total_profit=13821.349609375, total_revenue=53507.55078125)
Row(year=2013, month=12, total_profit=570226.8125, total_revenue=4205821.5)
```