In [ ]:
```
import snowflake.connector as snowflake
from snowflake.connector import ProgrammingError
import pandas as pd
import numpy as np
from datetime import datetime
```

In [ ]:
```
connection = snowflake.connect(
    user = 'zihanyang',
    password = 'yzhF063180',
    account = 'hhlxbcu-zkb67684',
)
cur = connection.cursor()

cur.execute(f"USE WAREHOUSE LEGION_WH")
```

Out[ ]:
```
<snowflake.connector.cursor.SnowflakeCursor at 0x22296eecbd0>
```

In [ ]:
```
def create_database_table(dataframe: pd.DataFrame,
                          primary_key: str,
                          database_name:str,
                          table_name: str) -> None:

        # map the data type
    def map_dtype(dtype) -> str:
        if dtype == "object":
            return "STRING"
        elif dtype == "int64":
            return "INTEGER"
        elif dtype == "float64":
            return "FLOAT"
        elif dtype == "bool":
            return "BOOLEAN"
        elif dtype == "datetime64[ns]":
            return "TIMESTAMP"
        else:
            return "STRING"  # Default to STRING.


    try:
        # Connect it
        cur.execute(f"USE DATABASE {database_name}")
```

```
    except ProgrammingError:
        # Create it
        cur.execute(f"CREATE DATABASE {database_name}")


    # Check if primary_key is a column or a list in the dataframe, composite key
    primary_key = primary_key if isinstance(primary_key, list) else [primary_key]

    try:
        if not all(col in dataframe.columns for col in primary_key):
            raise ValueError("One or more primary key columns not found in the dataframe.")
    except TypeError:
            raise ValueError("Invalid primary key type. Expected string or list.")


   # Check if primary_key column(s) is/are unique
    if dataframe.duplicated(subset=primary_key).any():
            raise ValueError("Composite primary key values are not unique.")


    # Generate the SQL CREATE or REPLACE command
    columns = [f"{column} {map_dtype(dtype)}" for column, dtype in dataframe.dtypes.items()]
    create_table = f"CREATE OR REPLACE TABLE {table_name} (\n"
    create_table += ",\n".join(columns) + ",\n"
    create_table +=  f"PRIMARY KEY ({','.join(primary_key)}));\n"

    cur.execute(create_table)

    # Generate the SQL INSERT command
    insert_statement = f"INSERT INTO {table_name} ("
    insert_statement += f", ".join(dataframe.columns) + f") VALUES "

    # Iterate over each row in the dataframe
    for _, row in dataframe.iterrows():
        values = []
        for value in row.values:
            if pd.isna(value):
                values.append("NULL")
            elif isinstance(value, str):
                values.append(f"'{value}'")
            elif isinstance(value, pd.Timestamp):
                values.append(f"'{value}'")
            else:
                values.append(str(value))
```

```
        insert_statement = insert_statement + "(" + ", ".join(values) + "), "

    insert_statement = insert_statement[:-2]  #  Remove the final comma and space
    cur.execute(insert_statement)
```

In [ ]:
```
def create_initial_surrogate_key_mapping_table(table: pd.DataFrame,
                                               surrogate_key_name: str,
                                               start_date: str,
                                               end_date: str) -> tuple[pd.DataFrame, pd.DataFrame]:
    # Create an array of consecutive integers of the size the length of the table called surrogate_key
    num_rows = table.shape[0]

    # Create a new DataFrame to avoid SettingWithCopyWarning
    new_table = table.copy()

    # Assign the surrogate_key values to a new column in the new_table as the first column
    surrogate_key = range(num_rows)
    new_table.insert(0, surrogate_key_name, surrogate_key)  # Inserting at the first position

    # Create surrogate key mapping table containing surrogate key and original natural key
    surrogate_key_mapping_table = new_table.iloc[:, 0:2].copy()
    surrogate_key_mapping_table['Start_Date'] = start_date
    surrogate_key_mapping_table['End_Date'] = end_date
    surrogate_key_mapping_table['Current_Flag'] = True

    return new_table, surrogate_key_mapping_table
```

Create CUSTOMER_DIM Mapping Table

In [ ]:
```
cur.execute("SELECT c.customerid as customer_id, \
            c.contactname as customer_name, \
            c.country\
            FROM northwinds.public.customers c")
rows = cur.fetchall()
columns = [column[0] for column in cur.description]

# You seem to have repeated the code that defines the 'columns' variable twice
# I will remove the redundant line here
# columns = [column[0] for column in cur.description]

# Create a DataFrame using fetched rows and columns from the query
```

```
CUSTOMER = pd.DataFrame(rows, columns = columns)

# Assuming the create_initial_surrogate_key_mapping_table function is defined correctly
# and it is expected to return two values, 'customer_surrogate_key_mapping' and 'customer_surrogate_key_mapping'
CUSTOMER, customer_surrogate_key_mapping = create_initial_surrogate_key_mapping_table(table = CUSTOMER,
                                                                                       surrogate_key_name='CUSTOMER_KEY',
                                                                                       start_date= datetime(2000,1,1),
                                                                                       end_date=datetime(2099,12,31))
```

In [ ]:
```
create_database_table(CUSTOMER, 'CUSTOMER_KEY', 'northwinds_data_warehouse_1', 'customer_dim')
```

In [ ]:
```
create_database_table(customer_surrogate_key_mapping, 'CUSTOMER_KEY', 'northwinds_data_warehouse_1', 'customer_surrogate_mapping_tab
```

Create PRODUCT_DIM Mapping Table

In [ ]:
```
cur.execute("SELECT p.productid as product_id,\
            p.productname as product_name,\
            c.categoryname as category_name\
            FROM northwinds.public.products p\
            LEFT JOIN northwinds.public.categories c on p.categoryid = c.categoryid")
rows = cur.fetchall()
columns = [columns[0] for coumn in cur.description]

columns = [column[0] for column in cur.description]
PRODUCT = pd.DataFrame(rows, columns=columns)
PRODUCT, product_surrogate_key_mapping = create_initial_surrogate_key_mapping_table(table=PRODUCT,
                                                                                     surrogate_key_name='PRODUCT_KEY',
                                                                                     start_date=datetime(2000,1,1),
                                                                                     end_date=datetime(2099,12,31))
```

In [ ]:
```
PRODUCT
```

Out[ ]:

	PRODUCT_KEY	PRODUCT_ID	PRODUCT_NAME	CATEGORY_NAME
0	0	1	Chai	Beverages
1	1	2	Chang	Beverages
2	2	3	Aniseed Syrup	Condiments
3	3	4	Chef Antons Cajun Seasoning	Condiments
4	4	5	Chef Antons Gumbo Mix	Condiments
...	...	...	...	...
72	72	73	R d Kaviar	Seafood
73	73	74	Longlife Tofu	Produce
74	74	75	Rh nbr u Klosterbier	Beverages
75	75	76	Lakkalik ri	Beverages
76	76	77	Original Frankfurter gr ne So e	Condiments

77 rows × 4 columns

In [ ]:
```
product_surrogate_key_mapping
```

Out[ ]:

	PRODUCT_KEY	PRODUCT_ID	Start_Date	End_Date	Current_Flag
0	0	1	2000-01-01	2099-12-31	True
1	1	2	2000-01-01	2099-12-31	True
2	2	3	2000-01-01	2099-12-31	True
3	3	4	2000-01-01	2099-12-31	True
4	4	5	2000-01-01	2099-12-31	True
...	...	...	...	...	...
72	72	73	2000-01-01	2099-12-31	True
73	73	74	2000-01-01	2099-12-31	True
74	74	75	2000-01-01	2099-12-31	True
75	75	76	2000-01-01	2099-12-31	True
76	76	77	2000-01-01	2099-12-31	True

77 rows × 5 columns

In [ ]:
```
create_database_table(PRODUCT, 'PRODUCT_KEY', 'northwinds_data_warehouse_1', 'product_dim')
```

In [ ]:
```
create_database_table(product_surrogate_key_mapping, 'PRODUCT_KEY', 'northwinds_data_warehouse_1', 'product_surrogate_mapping_table'
```

Create EMPLOYEE_DIM Mapping Table

In [ ]:
```
cur.execute("SELECT e.employeeid as employee_id,\
             e.lastname,\
             e.firstname,\
             e.city,\
             e.country\
             FROM northwinds.public.employees e")
rows = cur.fetchall()
columns = [column[0] for column in cur.description]

EMPLOYEE = pd.DataFrame(rows, columns=columns)
EMPLOYEE, employee_surrogate_key_mapping = create_initial_surrogate_key_mapping_table(table= EMPLOYEE,
```

```
                                                                                surrogate_key_name= 'EMPLOYEE_KEY',
                                                                                start_date= datetime(2000,1,1),
                                                                                end_date=datetime(2099,12,31))
```

In [ ]:
```
create_database_table(EMPLOYEE, 'EMPLOYEE_KEY', 'northwinds_data_warehouse_1', 'employee_dim')
```

In [ ]:
```
create_database_table(employee_surrogate_key_mapping, 'EMPLOYEE_KEY', 'northwinds_data_warehouse_1', 'EMPLOYEE_surrogate_mapping_tal
```

Create SHIPPER_DIM Mapping Table

In [ ]:
```
cur.execute("SELECT s.shipperid as shipper_id,\
            s.companyname as company_name\
            FROM northwinds.public.shippers s")
rows = cur.fetchall()
columns = [column[0] for column in cur.description]

SHIPPER = pd.DataFrame(rows, columns=columns)
SHIPPER, shipper_surrogate_key_mapping = create_initial_surrogate_key_mapping_table(table=SHIPPER,
                                                                                    surrogate_key_name='SHIPPER_KEY',
                                                                                    start_date=datetime(2000,1,1),
                                                                                    end_date=datetime(2099,12,31))
```

In [ ]:
```
shipper_surrogate_key_mapping
```

Out[ ]:

	SHIPPER_KEY	SHIPPER_ID	Start_Date	End_Date	Current_Flag
0	0	1	2000-01-01	2099-12-31	True
1	1	2	2000-01-01	2099-12-31	True
2	2	3	2000-01-01	2099-12-31	True

In [ ]:
```
create_database_table(SHIPPER, 'SHIPPER_KEY', 'northwinds_data_warehouse_1', 'shipper_dim')
```

In [ ]:
```
create_database_table(shipper_surrogate_key_mapping, 'SHIPPER_KEY', 'northwinds_data_warehouse_1', 'SHIPPER_surrogate_mapping_table'
```

Create DATE_DIM Mappling Table

In [ ]:
```
cur.execute("SELECT orderdate AS date\
            FROM northwinds.public.orders")
rows = cur.fetchall()
columns = [column[0] for column in cur.description]

DATES = pd.DataFrame(rows, columns=['date'])

DATES['date'] = pd.to_datetime(DATES['date'])
DATES
```

Out[ ]:

	date
0	1996-07-04
1	1996-07-05
2	1996-07-08
3	1996-07-08
4	1996-07-09
...	...
825	1998-05-05
826	1998-05-06
827	1998-05-06
828	1998-05-06
829	1998-05-06

830 rows × 1 columns

In [ ]:
```
#  Use Python to add the other date attributes
from datetime import datetime
import holidays
from datetime import date
```

In [ ]:
```
DATES['week_of_year'] = DATES['date'].dt.isocalendar().week
DATES['month'] = DATES['date'].dt.month
```

```
DATES['year'] = DATES['date'].dt.year
DATES['day_of_week'] = DATES['date'].dt.day_name()
```

In [ ]:
```
DATES = DATES.drop_duplicates()
```

In [ ]:
```
create_database_table(DATES, 'date', 'northwinds_data_warehouse_1', 'dates_dim')
```

Create SALES_FACT Table

In [ ]:
```
cur.execute("SELECT c.customerid,\
            o.orderdate,\
            s.shipperid,\
            e.employeeid,\
            o.orderid, \
            p.productid, \
            od.unitprice,\
            od.quantity,\
            od.discount AS discount_number,\
            SUM (od.unitprice * od.quantity - od.discount) AS total_revenue\
            FROM northwinds.public.orders o\
            LEFT JOIN northwinds.public.orderdetails AS od ON o.orderid = od.orderid\
            LEFT JOIn northwinds.public.products AS p ON od.productid = p.productid\
            LEFT JOIN northwinds.public.customers AS c ON o.customerid = c.customerid\
            LEFT JOIN northwinds.public.employees AS e ON o.employeeid = e.employeeid\
            LEFT JOIN northwinds.public.shippers AS s On o.shipvia = s.shipperid\
            GROUP BY c.customerid, o.orderdate, s.shipperid, e.employeeid, o.orderid, p.productid, od.unitprice, od.quantity, od.dis
rows = cur.fetchall()
columns = [column[0] for column in cur.description]
Initial_Fact_Table = pd.DataFrame(rows, columns=columns)
Initial_Fact_Table
```

Out[ ]:

	CUSTOMERID	ORDERDATE	SHIPPERID	EMPLOYEEID	ORDERID	PRODUCTID	UNITPRICE	QUANTITY	DISCOUNT_NUMBER	TOTAL_REVENUE
0	VINET	1996-07-04	3	5	10248	42	10	10	0	100
1	HANAR	1996-07-08	2	4	10250	41	8	10	0	80
2	VICTE	1996-07-08	1	3	10251	65	17	20	0	340
3	SUPRD	1996-07-09	2	4	10252	20	65	40	0	2600
4	HANAR	1996-07-10	2	3	10253	39	14	42	0	588
...	...	...	...	...	...	...	...	...	...	...
2150	LILAS	1998-05-05	1	1	11071	7	30	15	0	450
2151	SAVEA	1998-03-11	2	7	10941	62	49	30	0	1470
2152	ERNSH	1998-03-26	2	8	10979	31	13	24	0	312
2153	VAFFE	1998-03-12	2	1	10946	10	31	25	0	775
2154	BONAP	1998-03-06	1	8	10932	62	49	14	0	686

2155 rows × 10 columns

In [ ]:
```
cur.execute("SELECT * FROM CUSTOMER_surrogate_mapping_table")
rows = cur.fetchall()
columns = [columns[0] for column in cur.description]
customer_surrogate_mapping_table = pd.DataFrame(rows, columns=columns)

cur.execute("SELECT * FROM PRODUCT_surrogate_mapping_table")
rows = cur.fetchall()
columns = [columns[0] for column in cur.description]
product_surrogate_mapping_table = pd.DataFrame(rows, columns=columns)

cur.execute("SELECT * FROM EMPLOYEE_surrogate_mapping_table" )
rows = cur.fetchall()
columns = [columns[0] for column in cur.description]
employee_surrogate_mapping_table = pd.DataFrame(rows, columns=columns)

cur.execute("SELECT * FROM SHIPPER_surrogate_mapping_table")
rows = cur.fetchall()
columns = [columns[0] for column in cur.description]
shipper_surrogate_mapping_table = pd.DataFrame(rows, columns=columns)
```

In [ ]:
```
Sales_Fact_Table = Initial_Fact_Table
Sales_Fact_Table['CUSTOMERID'] = Sales_Fact_Table['CUSTOMERID'].map(customer_surrogate_key_mapping.set_index('CUSTOMER_ID')['CUSTOM
Sales_Fact_Table['EMPLOYEEID'] = Sales_Fact_Table['EMPLOYEEID'].map(employee_surrogate_key_mapping.set_index('EMPLOYEE_ID')['EMPLOY
Sales_Fact_Table['SHIPPERID'] = Sales_Fact_Table['SHIPPERID'].map(shipper_surrogate_key_mapping.set_index('SHIPPER_ID')['SHIPPER_KE
Sales_Fact_Table['PRODUCTID'] = Sales_Fact_Table['PRODUCTID'].map(product_surrogate_key_mapping.set_index('PRODUCT_ID')['PRODUCT_KE
```

In [ ]:
```
Sales_Fact_Table
Sales_Fact_Table['ORDERDATE'] = pd.to_datetime(Sales_Fact_Table['ORDERDATE'])
```

In [ ]:
```
sales_fact_table, sales_surrogate_key_mapping = create_initial_surrogate_key_mapping_table(table = Sales_Fact_Table,
                                                                                            surrogate_key_name = 'SALES_KEY',
                                                                                            start_date = datetime(2000,1,1),
                                                                                            end_date = datetime(2099,12,31))
sales_fact_table
```

Out[ ]:

	SALES_KEY	CUSTOMERID	ORDERDATE	SHIPPERID	EMPLOYEEID	ORDERID	PRODUCTID	UNITPRICE	QUANTITY	DISCOUNT_NUMBER	TOTAL_RE
0	0	84	1996-07-04	2	4	10248	41	10	10	0	
1	1	33	1996-07-08	1	3	10250	40	8	10	0	
2	2	83	1996-07-08	0	2	10251	64	17	20	0	
3	3	75	1996-07-09	1	3	10252	19	65	40	0	
4	4	33	1996-07-10	1	2	10253	38	14	42	0	
...	...	...	...	...	...	...	...	...	...	...	
2150	2150	45	1998-05-05	0	1	11071	6	30	15	0	
2151	2151	70	1998-03-11	1	6	10941	61	49	30	0	
2152	2152	19	1998-03-26	1	7	10979	30	13	24	0	
2153	2153	82	1998-03-12	1	1	10946	9	31	25	0	
2154	2154	8	1998-03-06	0	7	10932	61	49	14	0	

2155 rows × 11 columns

In [ ]:
```
create_database_table(sales_fact_table, 'SALES_KEY', 'northwinds_data_warehouse_1', 'SALES_FACT')
```

In [ ]:
```
create_database_table(sales_surrogate_key_mapping, 'SALES_KEY', 'northwinds_data_warehouse_1', 'sales_surrogate_mapping_table')
```

Test Queries

In [ ]:
```
cur.execute("SELECT sf.total_revenue \
FROM northwinds_data_warehouse_1.public.sales_fact AS sf \
LEFT JOIN northwinds_data_warehouse_1.public.product_dim AS p ON sf.productid = p.product_key \
LEFT JOIN northwinds_data_warehouse_1.public.dates_dim AS d ON sf.orderdate = d.date \
WHERE p.product_name = 'Tofu' \
AND d.year = 1996\
AND d.month = 7")

rows = cur.fetchall()
rows
```

Out[ ]:
```
[(171,)]
```

In [ ]:
```
cur.execute("WITH TABLE1 AS(SELECT CONCAT(e.firstname, ' ', e.lastname) AS full_employee_name,\
            e.country,\
            SUM(total_revenue) AS total_order_value,\
            ROW_NUMBER() OVER (PARTITION BY e.country ORDER BY SUM(sf.total_revenue) DESC) AS rank\
            FROM northwinds_data_warehouse_1.public.employee_dim AS e\
            LEFT JOIN northwinds_data_warehouse_1.public.sales_fact AS sf ON sf.employeeid = e.employee_key\
            GROUP BY e.firstname, e.lastname, e.country)\
            SELECT full_employee_name, country, total_order_value\
            FROM TABLE1 \
            WHERE RANK = 1")
rows2 = cur.fetchall()
rows2
```

Out[ ]:
```
[('Robert King', 'UK', 141489), ('Margaret Peacock', 'USA', 250451)]
```