Week5_Implementation

Part I

`start my_sql`
`mysql -u root -p sales < sales.sql`

Python code:

```
# Import libraries required for connecting to mysql
import mysql.connector
# Import libraries required for connecting to DB2
import ibm_db
# Connect to MySQL
mysql_connection = mysql.connector.connect(user='root',
password='MTA0MzEtam9hb3Bh',host='127.0.0.1',database='sales')
cursor = mysql_connection.cursor()
# Connect to DB2
# connectction details

dsn_hostname = "9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud" # e.g.:
"dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"
dsn_uid = "sjc89808"        # e.g. "abc12345"
dsn_pwd = "3IQZkTgbhBgn98ag"      # e.g. "7dBZ3wWt9XN6$o0J"
dsn_port = "32459"                # e.g. "50000"
dsn_database = "bludb"            # i.e. "BLUDB"
dsn_driver = "{IBM DB2 ODBC DRIVER}" # i.e. "{IBM DB2 ODBC DRIVER}"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_security = "SSL"              # i.e. "SSL"

#Create the dsn connection string
dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname,
```

```
dsn_port, dsn_protocol, dsn_uid, dsn_pwd, dsn_security)

# create connection
db2_connection = ibm_db.connect(dsn, "", "")

# Find out the last rowid from DB2 data warehouse
# The function get_last_rowid must return the last rowid of the table
sales_data on the IBM DB2 database.

def get_last_rowid():
    SQL = """
    SELECT MAX(ROWID) FROM sales_data
    """
    row_id = ibm_db.exec_immediate(db2_connection, SQL)
    row_val = ibm_db.fetch_tuple(row_id)[0]
    return row_val



last_row_id = get_last_rowid()
print("Last row id on production datawarehouse = ", last_row_id)

# List out all records in MySQL database with rowid greater than the one on
the Data warehouse
# The function get_latest_records must return a list of all records that
have a rowid greater than the last_row_id in the sales_data table in the
sales database on the MySQL staging data warehouse.

def get_latest_records(rowid):
    SQL = f"""
    SELECT * FROM sales.sales_data where rowid > {rowid}
    """
    cursor.execute(SQL)
    list_of_tuples = []
    for row in cursor.fetchall():
        list_of_tuples.append(row)
    #mysql_connection.commit()
    return  list_of_tuples

new_records = get_latest_records(last_row_id)

print("New rows on staging datawarehouse = ", len(new_records))

# Insert the additional records from MySQL into DB2 data warehouse.
```

```
# The function insert_records must insert all the records passed to it into
the sales_data table in IBM DB2 database.

def insert_records(records):
    SQL = "INSERT INTO SALES_DATA(ROWID, PRODUCT_ID,
CUSTOMER_ID,QUANTITY,TIMESTAMP)  VALUES(?,?,?,?,CURRENT TIMESTAMP);"
    stmt = ibm_db.prepare(db2_connection, SQL)
    for element in records:
        ibm_db.execute(stmt, element)



insert_records(new_records)
print("New rows inserted into production datawarehouse = ",
len(new_records))

# disconnect from mysql warehouse
mysql_connection.close()
# disconnect from DB2 data warehouse
# close connection
ibm_db.close(db2_connection)
# End of program
```

Part II

code:

```
# import the libraries

from datetime import timedelta
# The DAG object; we'll need this to instantiate a DAG
from airflow import DAG
# Operators; we need this to write tasks!
from airflow.operators.bash_operator import BashOperator
# This makes scheduling easy
from airflow.utils.dates import days_ago

#defining DAG arguments

# You can override them on a per-task basis during operator initialization
default_args = {
    'owner': 'Joao Paulo',
    'start_date': days_ago(0),
```

```
    'email': ['joaopaulo@gmail.com'],
    'email_on_failure': True,
    'email_on_retry': True,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

# define the DAG
dag = DAG(
    dag_id='process_web_log',
    default_args=default_args,
    description='DAG to process web log',
    schedule_interval='@daily',
)

# define the first task named extract_data
extract = BashOperator(
    task_id='extract_data',
    bash_command='cut -d" " -f1 /home/project/accesslog.txt >
/home/project/extracted_data.txt',
    dag=dag,
)

# define the second task
transform = BashOperator(
    task_id='transform_data',
    bash_command='grep -v "198.46.149.143" /home/project/extracted_data.txt
>  /home/project/transformed_data.txt',
    dag=dag,
)

#define third task
load = BashOperator(
    task_id = 'load_data',
    bash_command= 'tar -zcvf weblog.tar /home/project/transformed_data.txt',
    dag = dag

)

extract >> transform >> load
```