

Data Analytics with Hadoop Project.

1. Tools

- Hadoop Server
- Visual Studio Code
- Python Version 3.10 or Upper
- Install the Package (psycopg2-binary, SQLAlchemy, sqlparse, pandas, hdfs, pywebhdfs, mrjob)
- DBeaver

2. Steps by Steps

- config.json

```
{
  "marketplace_prod": {
    "host": "103.150.197.96",
    "db": "project_4",
    "user": "postgres",
    "password": "dsbatch13!",
    "port": 5431
  },
  "dwh": {
    "host": "103.150.197.96",
    "db": "dwh_project_4",
    "user": "postgres",
    "password": "dsbatch13!",
    "port": 5431
  },
  "hadoop": {
    "client": "http://hadoop-server:9870"
  }
}
```

1. config.json

Script ini berfungsi untuk mengonfigurasi 3 koneksi serta pengaturan akses antara dua database (data source marketplace_prod dan destination dwh) dengan host, nama database, pengguna, kata sandi, dan port yang telah dibuat yang kemudian data tersebut tersimpan didalam Hadoop (<http://hadoop-server:9870>).

- Design DWH

```
DROP TABLE IF EXISTS dim_orders_alif;
CREATE TABLE dim_orders_alif (
  order_id INT NOT NULL,
  order_date DATE NOT NULL,
  user_id INT NOT NULL,
  payment_name VARCHAR(255),
  shipper_name VARCHAR(255),
  order_price INT,
  order_discount INT,
  voucher_name VARCHAR(255),
  voucher_price INT,
  order_total INT,
  rating_status VARCHAR(255)
);
```

2. Design DWH

Query diatas berfungsi untuk membuat Table dim_orders_alif yang akan digunakan sebagai tempat penyimpanan data dari data source.

c. Koneksi ke Hadoop

```
import os
import json
import psycopg2
import hdfs

from sqlalchemy import create_engine

def config(connection_db):
    path = os.getcwd()
    with open(path+'/'+ 'config.json') as file:
        conf = json.load(file)[connection_db]
    return conf

def psql_conn(conf, name_conn):
    try:
        conn = psycopg2.connect(
            host=conf['host'],
            database=conf['db'],
            user=conf['user'],
            password=conf['password'],
            port=conf['port']
        )
        print(f'[INFO] Success connect PostgreSQL {name_conn}')
        engine =
    create_engine(f"postgresql+psycopg2://{conf['user']}:{conf['password']}@{conf[
'host']}:{conf['port']}/{conf['db']}")
        return conn, engine
    except Exception as e:
        print(f"[INFO] Can't connect PostgreSQL {name_conn}")
        print(str(e))

def hadoop_conn(conf):
    client = conf['client']
    try:
        conn = hdfs.InsecureClient(client)
        print("[INFO] Success to connect HADOOP ...")
        return conn
    except:
        print("[INFO] Can't connect HADOOP ...")
```

3. Koneksi ke Hadoop

Script berfungsi untuk mengatur koneksi dan akses ke database PostgreSQL dan Hadoop. Pertama fungsi "config" digunakan untuk membaca file konfigurasi JSON ("config.json") yang telah dibuat sebelumnya. Kemudian fungsi "psql_conn" digunakan untuk membuat koneksi ke database PostgreSQL dengan menggunakan package psycopg2. Terakhir fungsi "hadoop_conn" digunakan untuk membuat koneksi ke Hadoop dengan menggunakan package hdfs.

d. App.py

```
# connection data source
conf = connection.config('marketplace_prod')
conn, engine = connection.psql_conn(conf, 'DataSource')
cursor = conn.cursor()

# connection dwh
conf_dwh = connection.config('dwh')
conn_dwh, engine_dwh = connection.psql_conn(conf_dwh, 'DataWarehouse')
cursor_dwh = conn_dwh.cursor()

# connection dwh to hadoop
conf_dwh_hadoop = connection.config('hadoop')
client_hadoop = connection.hadoop_conn(conf_dwh_hadoop)

# get query string
path_query = os.getcwd()+'/query/'
query = sqlparse.format(
    open(path_query+'query.sql', 'r').read(), strip_comments=True
).strip()

# get schema dwh design
path_dwh_design = os.getcwd()+'/query/'
dwh_design = sqlparse.format(
    open(path_dwh_design+'dwh_design.sql', 'r').read(),
strip_comments=True
).strip()

try:
    # get data
    print('[INFO] Service ETL is Running ...')
    df = pd.read_sql(query, engine)

    # upload file to hadoop as DWH
    filetype = datetime.now().strftime('%Y%m%d')

    my_file = f'dim_orders_{filetype}_alif.csv'
    my_file_with_path = f'/digitalskola/project4/{my_file}'
    with client_hadoop.write(my_file_with_path, encoding='utf-8') as
writer:
        df.to_csv(writer, index=False)
        print(f"[INFO] Upload Data in HADOOP Success ...")

    # get data from hadoop for create data mart
    print(f"[INFO] Get Data in HADOOP .....")
    hdfs=PyWebHdfsClient(host='hadoop-server',port='9870',
user_name='hduser')
    filetype = datetime.now().strftime('%Y%m%d')
    data = hdfs.read_file(str(my_file_with_path))
    data = data.decode().split('\n')
    data_list = []
    for item in data:
        item = item.replace('\r', '')
```

```

if item != '':
    data_list.append(item.split(','))
    pd.DataFrame(data_list[1:],
columns=data_list[0]).to_csv(f'output/{my_file}', index=False)
    os.system(f'python mapReduce.py output/{my_file} >
output/Wordercount_output_hadoop_map.txt')
    print(f"[INFO] Download Data in HADOOP Success and Created file
mart.....")

```

4b. App.py

Pada script diatas diatas menjelaskan Connection data source: Membuat koneksi ke data source "marketplace_prod" menggunakan fungsi "config" dan "psql_conn" dari modul "connection". Connection dwh: Membuat koneksi ke data warehouse "dwh" menggunakan fungsi "config" dan "psql_conn". Connection dwh to hadoop: Membuat koneksi ke Hadoop menggunakan fungsi "config" dan "hadoop_conn". Get query string: Membaca dan memformat string kueri dari file "query.sql" yang terletak di direktori "query/" menggunakan fungsi "sqlparse.format". Get schema dwh design: Membaca dan memformat schema desain data warehouse dari file "dwh_design.sql" yang terletak di direktori "query/". Get data: Mengeksekusi query SQL pada data source menggunakan Pandas dan menyimpan hasilnya dalam DataFrame "df". Upload file to Hadoop: Mengunggah DataFrame "df" ke Hadoop sebagai data warehouse. Nama file yang diunggah disesuaikan dengan timestamp. Get data from Hadoop: Membaca file yang diunggah dari Hadoop, melakukan manipulasi data, dan menyimpannya dalam file "dim_orders_{timestamp}_alif.csv". Perform MapReduce: Menjalankan skrip "mapReduce.py" untuk melakukan operasi MapReduce pada file hasil sebelumnya dan menyimpan hasilnya dalam file "output/Wordercount_output_hadoop_map.txt".

e. MapReduce.py

```

cols =
'order_id,order_date,user_id,payment_name,shipper_name,order_price,orde
r_discount,voucher_name,voucher_price,order_total,rating_status'.split(
',')

def csv_readline(line):
    """Given a sting CSV line, return a list of strings."""
    for row in csv.reader([line]):
        return row

class OrderDateCount(MRJob):
    def steps(self):
        return [
            MRStep(mapper=self.mapper, reducer=self.reducer),
            MRStep(reducer=self.sort)
        ]

    def mapper(self, _, line):
        # Convert each line into a dictionary
        row = dict(zip(cols, csv_readline(line)))

        #skip first row as header
        if row['order_id'] != 'order_id':
            # Yield the order_date

```

5a. MapReduce.py

```

        yield row['order_date'][0:7], 1

    def reducer(self, key, values):
        #for 'order_date' compute
        yield None, (key, sum(values))

    def sort(self, key, values):
        data = []
        for order_date, order_count in values:
            data.append((order_date, order_count))
            data.sort()

        for order_date, order_count in data:
            yield order_date, order_count

```

5a. MapReduce.py

Script ini merupakan implementasi dari MapReduce menggunakan framework MRJob. Tujuannya adalah untuk menghitung jumlah pesanan berdasarkan tanggal pesanan (order_date). Variabel "cols": Menyimpan daftar nama kolom yang digunakan dalam file CSV yang akan diproses. Kemudian fungsi "csv_readline": Membaca baris CSV sebagai string dan mengembalikan daftar string yang berisi nilai-nilai kolom. Lalu kelas "OrderDateCount": Merupakan kelas utama yang mengimplementasikan algoritma MapReduce. Method "steps" menentukan langkah-langkah MapReduce yang akan dilakukan, yaitu "mapper", "reducer", dan "sort". Method "mapper": Menerima baris input dan mengonversinya menjadi sebuah kamus (dictionary) menggunakan nama kolom dari "cols". Kemudian, untuk setiap baris yang bukan merupakan header, menghasilkan pasangan kunci-nilai dengan kunci berupa bulan dan tahun dari tanggal pesanan (order_date) dan nilai 1. Method "reducer": Menerima pasangan kunci-nilai dari mapper, yang memiliki kunci yang sama, yaitu bulan dan tahun pesanan. Menghasilkan pasangan kunci-nilai di mana kunci adalah None (tidak digunakan) dan nilai berupa pasangan kunci-nilai dari bulan dan jumlah pesanan pada bulan tersebut. Method "sort": Menerima pasangan kunci-nilai dari reducer dan mengumpulkannya dalam sebuah list. Kemudian, list tersebut diurutkan berdasarkan tanggal pesanan (order_date). Akhirnya, menghasilkan pasangan kunci-nilai dengan kunci berupa tanggal pesanan dan nilai berupa jumlah pesanan pada tanggal tersebut.

3. Hasil

a. App.py

Setelah semua script tidak ada peringatan error pada Visual Studio Code jalankan command "python app.py". Kemudian akan muncul hasil seperti dibawah.

```

Cerv@LAPTOP-HWKJLOOC MINGW64 /d/BOOTCAMP/bahan/PROJECT4/digitaskola_de_10_batch_processing-main/digitaskola_de_10_batch_processing-main
$ python app.py
[INFO] Service ETL is Starting ...
[INFO] Success connect PostgreSQL DataSource
[INFO] Success connect PostgreSQL DataWarehouse
[INFO] Success to connect HADOOP ...
[INFO] Service ETL is Running ...
[INFO] Upload Data in HADOOP Success ...
[INFO] Get Data in HADOOP .....
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory C:\Users\Cerv\AppData\Local\Temp\mapReduce.Cerv.20230630.070003.839655
Running step 1 of 2...
Running step 2 of 2...
job output is in C:\Users\Cerv\AppData\Local\Temp\mapReduce.Cerv.20230630.070003.839655\output
Streaming final output from C:\Users\Cerv\AppData\Local\Temp\mapReduce.Cerv.20230630.070003.839655\output...
Removing temp directory C:\Users\Cerv\AppData\Local\Temp\mapReduce.Cerv.20230630.070003.839655...
[INFO] Download Data in HADOOP Success and Created file mart.....
[INFO] Service ETL is Success ...

```

6. Python app.py

Pada gambar diatas menampilkan bahwa Service ETL sudah mulai hingga Service ETL berhasil.

b. Data di Dbeaver

	order_id	order_date	user_id	payment_name	shipper_name
1	1,110,001	2022-01-20	100,101	Debit	JNE Express
2	1,110,002	2022-01-29	100,102	Debit	JNE Express
3	1,110,003	2022-02-13	100,103	Credit	JNE Express
4	1,110,004	2022-03-06	100,102	Wallet	JNE Express
5	1,110,005	2022-04-28	100,105	Debit	Sicepat Express
6	1,110,006	2022-05-09	100,103	Debit	Sicepat Express
7	1,110,007	2022-05-21	100,106	Debit	JNE Express
8	1,110,008	2022-06-02	100,108	Credit	Sicepat Express
9	1,110,009	2022-06-23	100,103	Credit	Lazada Express
10	1,110,010	2022-07-01	100,102	Credit	Lazada Express
11	1,110,011	2022-07-21	100,110	Wallet	Sicepat Express
12	1,110,012	2022-07-30	100,110	Debit	JNE Express
13	11,100,012	2022-01-21	100,101	Debit	JNE Express
14	11,100,022	2022-01-22	100,102	Debit	JNE Express
15	11,100,032	2022-02-14	100,103	Credit	JNE Express
16	11,100,042	2022-03-05	100,102	Wallet	JNE Express

7. Data Dbeaver

Gambar diatas menampilkan bahwa berhasil mengtransform data ke DBeaver.

c. Hadoop

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	FZN	supergroup	10.07 KB	Jun 26 23:21	1	128 MB	dim_order_20230626_mfazima.csv
-rw-r--r--	Cerv	supergroup	10.07 KB	Jun 26 13:15	1	128 MB	dim_orders_20230626_alif.csv
-rw-r--r--	Admin	supergroup	10.07 KB	Jun 26 23:06	1	128 MB	dim_orders_20230626_fuad.csv
-rw-r--r--	PutuAndika	supergroup	10.07 KB	Jun 26 17:53	1	128 MB	dim_orders_20230626_putu_alvin.csv
-rw-r--r--	ASUS	supergroup	10.07 KB	Jun 26 21:41	1	128 MB	dim_orders_20230626_rendragi.csv
-rw-r--r--	andhika	supergroup	9.96 KB	Jun 27 16:06	1	128 MB	dim_orders_20230627_ardianto.csv
-rw-r--r--	andhika	supergroup	9.96 KB	Jun 27 23:13	1	128 MB	dim_orders_20230627_dhika.csv
-rw-r--r--	Babatunde	supergroup	10.07 KB	Jun 29 10:25	1	128 MB	dim_orders_20230629.csv
-rw-r--r--	Lenovo	supergroup	10.07 KB	Jun 29 23:44	1	128 MB	dim_orders_20230629_alwifhrz.csv
-rw-r--r--	Hendri	supergroup	10.07 KB	Jun 29 22:55	1	128 MB	dim_orders_20230629_hendri.csv
-rw-r--r--	Babatunde	supergroup	10.07 KB	Jun 29 10:37	1	128 MB	dim_orders_20230629_tommy.csv
-rw-r--r--	Wafir	supergroup	10.07 KB	Jun 29 18:28	1	128 MB	dim_orders_20230629_wafir.csv
-rw-r--r--	Lenovo	supergroup	10.07 KB	Jun 30 00:00	1	128 MB	dim_orders_20230630_alwifhrz.csv
-rw-r--r--	fbramantyo	supergroup	9.96 KB	Jun 30 04:00	1	128 MB	dim_orders_20230630_tyo.csv
-rw-r--r--	Andrew	supergroup	10.07 KB	Jun 22 12:19	1	128 MB	dim_orders_andrew20230622.csv
-rw-r--r--	rani	supergroup	9.96 KB	Jun 28 13:07	1	128 MB	dim_orders_rani.csv
-rw-r--r--	rani	supergroup	9.96 KB	Jun 28 13:11	1	128 MB	dim_orders_rani2.csv
-rw-r--r--	reza	supergroup	10.38 KB	Jun 24 17:39	1	128 MB	dim_orders_reza_20230624.csv

8. Hadoop

Gambar diatas menunjukkan berhasil menyimpan data di Hadoop

d. Output CSV

```
output > dim_orders_20230625_alif.csv
1 order_id,order_date,user_id,payment_name,shipper_name,order_price,order_discount,voucher_name,voucher_price,order_total,rating_status
2 1110001,2022-01-20,100101,Debit,JNE Express,250000,15000,New User,5000.0,230000,Medium Impact
3 1110002,2022-01-29,100102,Debit,JNE Express,620000,40000,New User,5000.0,575000,Medium Impact
4 1110003,2022-02-13,100103,Credit,JNE Express,6000000,1000000,New User,5000.0,4995000,Very Low Impact
5 1110004,2022-03-06,100102,Wallet,JNE Express,3150000,45000,,3105000,High Impact
6 1110005,2022-04-28,100105,Debit,Sicepat Express,4000000,1000000,New User,5000.0,2995000,Very Low Impact
7 1110006,2022-05-09,100103,Debit,Sicepat Express,4500000,1030000,,3470000,Medium High Impact
8 1110007,2022-05-21,100106,Debit,JNE Express,870000,25000,,845000,High Impact
9 1110008,2022-06-02,100108,Credit,Sicepat Express,2000000,0,New User,5000.0,1995000,Medium High Impact
10 1110009,2022-06-23,100103,Credit,Lazada Express,2000000,0,,2000000,High Impact
11 1110010,2022-07-01,100102,Credit,Lazada Express,1050000,45000,,1005000,Low Impact
12 1110011,2022-07-21,100110,Wallet,Sicepat Express,550000,15000,,535000,High Impact
13 1110012,2022-07-30,100110,Debit,JNE Express,490000,35000,Body Soap Promo,10000.0,445000,Medium High Impact
14 1110012,2022-01-21,100101,Debit,JNE Express,250000,15000,New User,5000.0,230000,Medium Impact
15 1110022,2022-01-22,100102,Debit,JNE Express,620000,40000,New User,5000.0,575000,Medium Impact
16 1110032,2022-02-14,100103,Credit,JNE Express,6000000,1000000,New User,5000.0,4995000,Very Low Impact
17 1110042,2022-03-05,100102,Wallet,JNE Express,3150000,45000,,3105000,High Impact
18 1110052,2022-04-28,100105,Debit,Sicepat Express,4000000,1000000,New User,5000.0,2995000,Very Low Impact
19 1110062,2022-05-07,100103,Debit,Sicepat Express,4500000,1030000,,3470000,Medium High Impact
20 1110072,2022-05-22,100106,Debit,JNE Express,870000,25000,,845000,High Impact
21 1110082,2022-06-03,100108,Credit,Sicepat Express,2000000,0,New User,5000.0,1995000,Medium High Impact
22 1110092,2022-06-24,100103,Credit,Lazada Express,2000000,0,,2000000,High Impact
23 1110102,2022-07-04,100102,Credit,Lazada Express,1050000,45000,,1005000,Low Impact
24 1110112,2022-07-23,100110,Wallet,Sicepat Express,550000,15000,,535000,High Impact
25 1110122,2022-07-23,100110,Debit,JNE Express,490000,35000,Body Soap Promo,10000.0,445000,Medium High Impact
26 1110013,2022-01-23,100101,Debit,JNE Express,250000,15000,New User,5000.0,230000,Medium Impact
27 1110023,2022-01-29,100102,Debit,JNE Express,620000,40000,New User,5000.0,575000,Medium Impact
28 1110033,2022-02-14,100103,Credit,JNE Express,6000000,1000000,New User,5000.0,4995000,Very Low Impact
```

9. Output CSV

Output pada data yang tersimpan.

e. Output MapReduce

```
output > Wordercount_output_hadoop_map.txt
1 "2022-01" 18
2 "2022-02" 9
3 "2022-03" 9
4 "2022-04" 9
5 "2022-05" 18
6 "2022-06" 18
7 "2022-07" 27
8
```

10. Output MapReduce

Output hasil dari MapReduce.