DigitalSkola
Uncover The World of Digital Skills with Us

"WEEK 11"

LEARNING PROGRESS REVIEW

BY : DREAM (KELOMPOK 1)

Data Realm Engineers And Maestros

ANGGOTA KELOMPOK

AFROH FAUZIAH

ALTHAF NAWADIR TAQIYYAH

ANDI ROSILALA

ANDREW BINTANG PRATAMA

ANDREW FORTINO MAHARDIKA SUADNYA

SPARK II

DATA ANALYTICS WITH SPARK

Different Type of Data

Booleans

- Terdiri dari 4 elemen: and, or, true, false
- Membentuk conditional requirement
- Terutama untuk filtering

Numbers

- Untuk filtering, counting
- Expression dilakukan pada numerical data types saat komputasi

String

- Selalu ada hampir di setiap data flow

Dates and timestamps

- Dates dan Timestamps disimpan sebagai String, terutama jika sumber datanya adalah CSV atau JSON
- Untuk track of timezones
- Timestamps, mengandung date dan time information

Complex Data

- Organize dan structure data lebih kompleks sesuai kebutuhan
- Tiga tipe complex types: structs, arrays, maps

Null Data

- Null untuk missing data
- Null values lebih baik secara eksplisit
- Explicitly drop nulls & Fill them with value

Data Aggreation

- Untuk summarize numerical data untuk grouping, penjumlahan, pengurangan, perkalian, dan pembagian.
- Aggregate value apapun menjadi array, list, map.

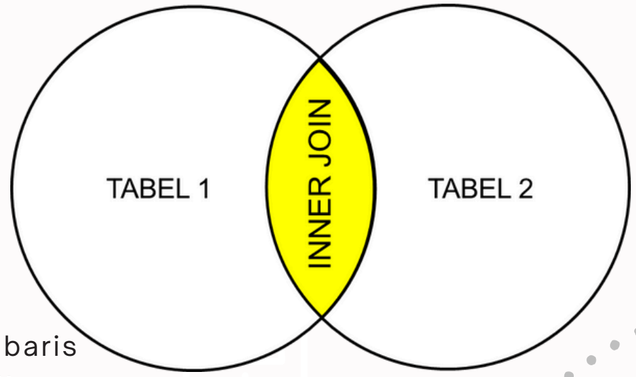
01 Aggregation langsung pada select statement

02 “group by” men-specify satu atau lebih key, dan satu atau lebih

03 “window” sama seperti “group by”. Bedanya rows input ke function berhubungan dengan current row

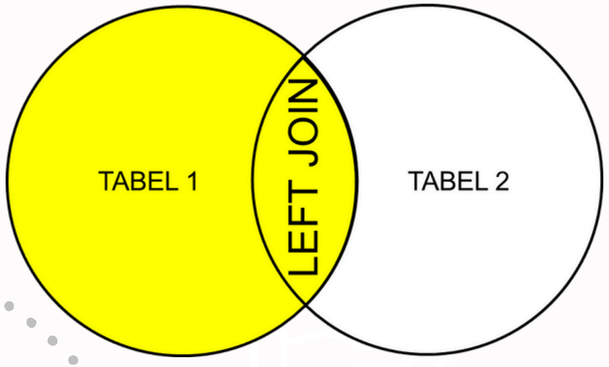
04 Function: count, sum, min, max, avg

Joint Data



Inner Join

Inner Join digunakan untuk menghasilkan baris data dengan menggabungkan 2 tabel atau lebih dengan menggunakan operator perbandingan pada kolom yang terdapat di tebel-tabel tersebut.



Left Join

Left Join merupakan penggabungan tabel dimana data akan ditampilkan secara keseluruhan pada tabel pertama (kiri) namun record pada tabel kedua (kanan) yang kosong akan ditampilkan dengan isi NULL.

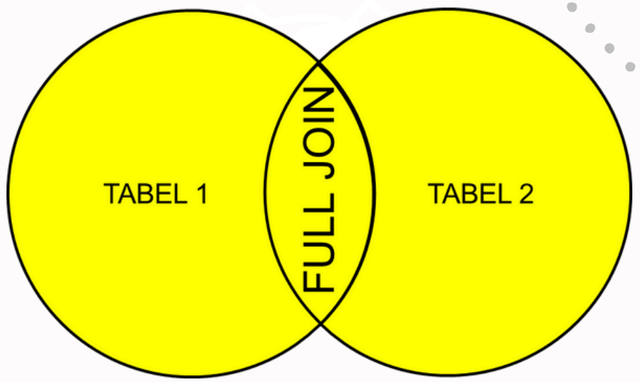
Right Join

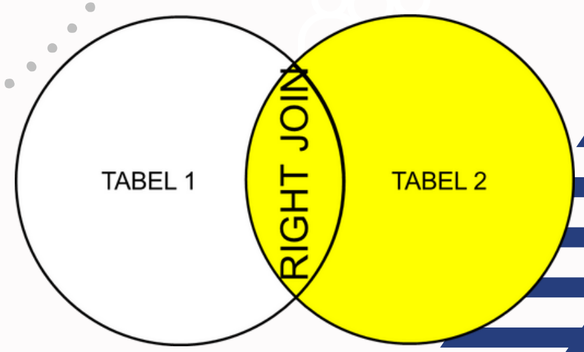
Right Join merupakan penggabungan tabel dimana data akan ditampilkan secara keseluruhan pada tabel kedua (kanan) namun record pada tabel pertama (kiri) yang kosong akan ditampilkan dengan isi NULL

Full Join

Full Join digunakan untuk menampilkan kedua tabel dengan record-record yang bersesuaian saja.

Jenis-Jenis Join Data





SPARK III

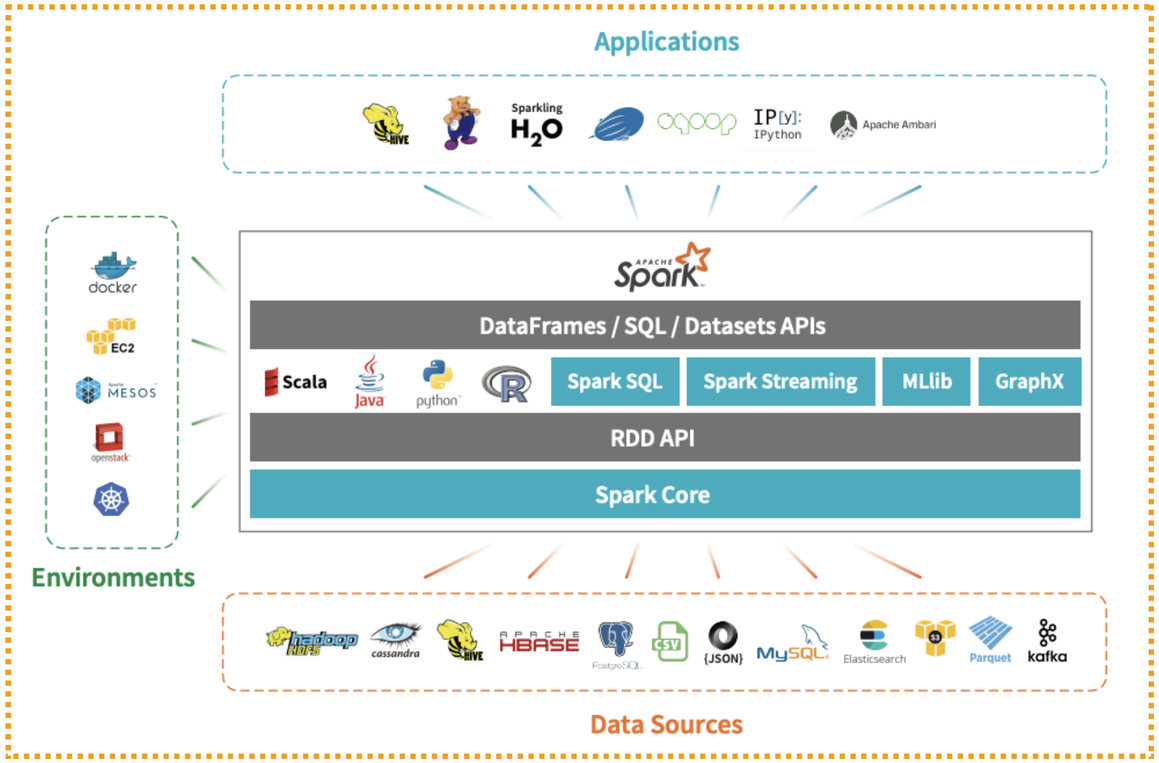
OPTIMIZING AND TUNING SPARK APPLICATIONS, BUILDING DATALAKE WITH SPARK (SPARK SQL AND DATAFRAME)

SparkSQL

- Spark SQL adalah library dan modul dari Apache Spark yang menyediakan antarmuka untuk bekerja dengan data terstruktur dan semi-terstruktur.
- Spark SQL memungkinkan pengguna untuk menjalankan SQL query dan juga menyediakan API untuk bekerja dengan DataFrame dan DataSet dalam berbagai bahasa pemrograman seperti Python, Scala, dan Java.

Kegunaan Spark SQL :

- Sumber DataFrame API > kumpulan library untuk bekerja dengan tabel data, juga membantu menentukan Frame Data.
- Catalyst Optimizer > modul pustaka yang dibuat berdasarkan perintah sistem untuk mengoptimasi kerangka kerja.



Fitur Spark SQL

Integrasi dengan Spark

Pengguna meng-query data terstruktur dengan SQL / DataFrame API di Java, Scala, Python, & R, juga membantu menggabungkan pemrosesan dengan komputasi Spark.

Penyeragaman Akses Data

DataFrames dan SQL mendukung akses ke berbagai sumber data (Hive, Avro, Parquet, ORC, JSON, dan JDBC) dan menggabungkannya.

Kompatibilitas dengan Hive

Menjalankan query Hive tanpa modifikasi data, memungkinkan kompatibilitas penuh dengan data, kueri, dan UDF Hive.

Konektivitas Standar dengan Alat Bisnis Intelijen

Spark SQL mendukung koneksi melalui JDBC dan ODBC, yang sering digunakan untuk konektivitas dengan BI Tools.

Fungsi Buatan Pengguna

Membuat fungsi buatan pengguna (UDF) untuk memperluas fungsionalitas Spark SQL sesuai kebutuhan.

Executing SQL commands on a DataFrame

```
from pyspark.sql import SparkSession
from pyspark.sql import Row
```

```
# Create a SparkSession
spark = SparkSession.builder.appName("SparkSQL").getOrCreate()

def mapper(line):
  fields = line.split(',')
  return Row(ID=int(fields[0]), name=str(fields[1]), age=int(fields[2]), numFriends=int(fields[3]))
```

```
lines = spark.sparkContext.textFile("fakefriends.csv")
people = lines.map(mapper)
```

```
# Infer the schema, and register the DataFrame as a table.
schemaPeople = spark.createDataFrame(people).cache()
schemaPeople.show()
```

```
schemaPeople.createOrReplaceTempView("people")
```

```
# SQL can be run over DataFrames that have been registered as a table
sample1 = spark.sql("select * from people where age >= 13 and age <= 19")
sample1.show()
```

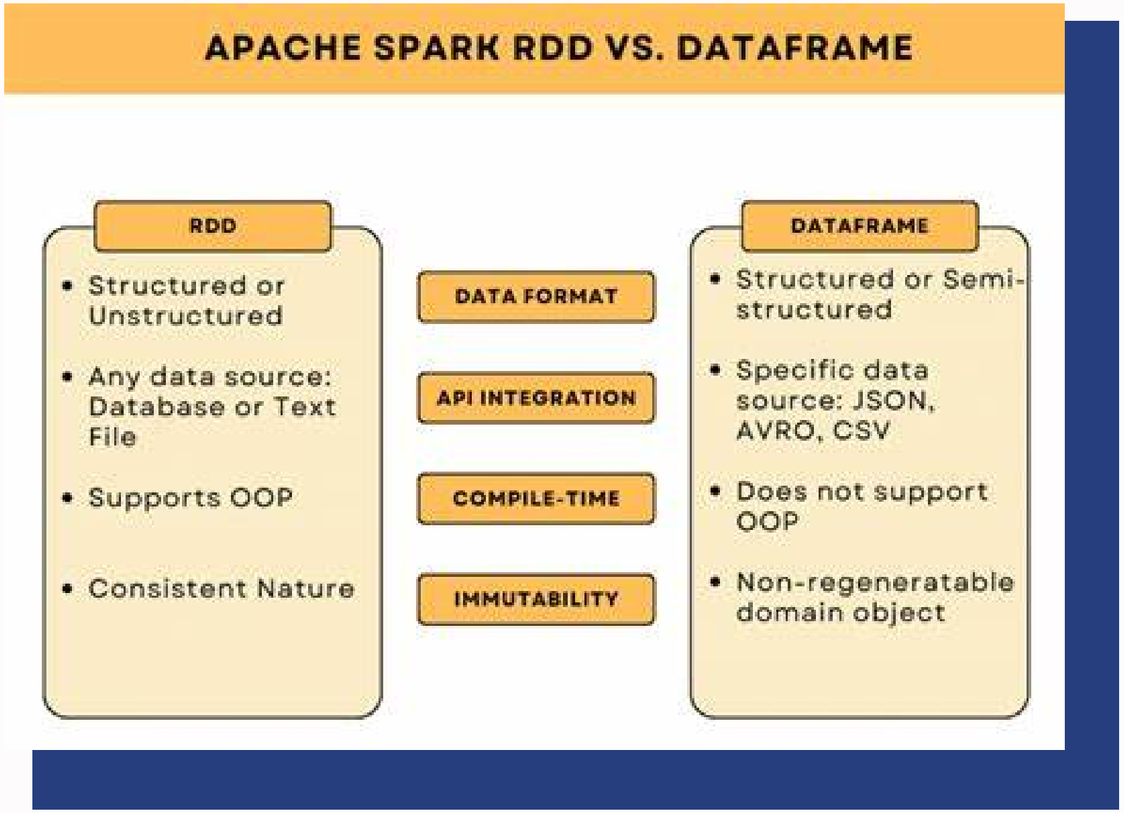
```
sample2 = spark.sql("select * from people where name='Will'")
sample2.show()
```

```
+---+----+---+----------+
| ID|name|age|numFriends|
+---+----+---+----------+
|  0|Will| 33|       385|
|  7|Will| 54|       307|
| 76|Will| 62|       201|
| 98|Will| 44|       178|
|107|Will| 64|       419|
|136|Will| 19|       335|
|164|Will| 31|       172|
|175|Will| 38|       459|
|206|Will| 21|       491|
|215|Will| 22|         6|
|252|Will| 36|       174|
|262|Will| 51|       334|
|275|Will| 47|        13|
|304|Will| 19|       404|
|338|Will| 28|       180|
|358|Will| 52|       276|
|387|Will| 43|       335|
|421|Will| 40|       261|
|423|Will| 44|       388|
+---+----+---+----------+
```

DigitalSkola

Using DataFrame instead of RDD

- RDD adalah API tingkat rendah yang memberikan kontrol penuh atas data, tetapi memerlukan lebih banyak kode dan pemahaman mendalam tentang distribusi data.
- DataFrame adalah abstraksi tingkat tinggi yang dibangun di atas RDD.
- DataFrame menyediakan API yang lebih kaya dan optimisasi otomatis menggunakan Catalyst optimizer.
- Menggunakan DataFrame daripada RDD (Resilient Distributed Dataset) > memanfaatkan abstraksi tingkat yang lebih tinggi memberikan beberapa keuntungan.



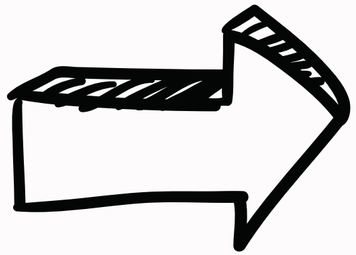
DigitalSkola

Using DataFrame instead of RDD

```
[5]  from pyspark.sql import SparkSession
     spark = SparkSession.builder.appName("SparkSQL").getOrCreate()

[10] people = spark.read.option("header", "true").option("inferSchema", "true").csv("worldcities.csv")
```

```
print("Here is our inferred schema:")
people.printSchema()
```

```
Here is our inferred schema:
root
 |-- city: string (nullable = true)
 |-- city_ascii: string (nullable = true)
 |-- lat: double (nullable = true)
 |-- lng: double (nullable = true)
 |-- country: string (nullable = true)
 |-- iso2: string (nullable = true)
 |-- iso3: string (nullable = true)
 |-- admin_name: string (nullable = true)
 |-- capital: string (nullable = true)
 |-- population: double (nullable = true)
 |-- id: integer (nullable = true)
```



```
print("Let's display the city column:")
people.select("city").show()
```

```
Let's display the city column:
+------------+
|        city|
+------------+
|       Tokyo|
|     Jakarta|
|       Delhi|
|      Mumbai|
|      Manila|
|    Shanghai|
|   São Paulo|
|       Seoul|
| Mexico City|
|   Guangzhou|
|     Beijing|
|       Cairo|
|    New York|
|     Kolkāta|
|      Moscow|
|     Bangkok|
|Buenos Aires|
|    Shenzhen|
|       Dhaka|
|       Lagos|
+------------+
only showing top 20 rows
```

```
print("Group by population")
people.groupBy("population").count().show()
```

```
Group by population
+----------+-----+
|population|count|
+----------+-----+
|  1.3528E7|    1|
| 1971704.0|    1|
| 1414100.0|    1|
|  330000.0|    4|
|  300000.0|    5|
|   94874.0|    1|
|    1051.0|    1|
|  924449.0|    1|
|  595118.0|    1|
|  431944.0|    1|
|  286240.0|    1|
|  217000.0|    1|
|  212474.0|    1|
|  203219.0|    1|
|  174564.0|    1|
|  168180.0|    1|
|  161925.0|    1|
|  154817.0|    1|
|  153717.0|    1|
|  153482.0|    1|
+----------+-----+
only showing top 20 rows
```

```
print("Increase the population of each city by 1000:")
people.select(people.city, people.population + 1000).show()
```

```
Increase the population of each city by 1000:
+------------+--------------------+
|        city|(population + 1000)|
+------------+--------------------+
|       Tokyo|           3.7978E7|
|     Jakarta|           3.4541E7|
|       Delhi|           2.9618E7|
|      Mumbai|           2.3356E7|
|      Manila|           2.3089E7|
|    Shanghai|           2.2121E7|
|   São Paulo|           2.2047E7|
|       Seoul|           2.1795E7|
| Mexico City|           2.0997E7|
|   Guangzhou|           2.0903E7|
|     Beijing|           1.9434E7|
|       Cairo|           1.9373E7|
|    New York|          1.871422E7|
|     Kolkāta|           1.7561E7|
|      Moscow|           1.7126E7|
|     Bangkok|           1.7067E7|
|Buenos Aires|           1.6158E7|
|    Shenzhen|            1.593E7|
|       Dhaka|           1.5444E7|
|       Lagos|            1.528E7|
+------------+--------------------+
only showing top 20 rows
```

DigitalSkola

Exercise

Find Avg Population by City using SparkSQL DataFrame

```
pip install pyspark
```

```
from pyspark.sql import SparkSession
from pyspark.sql import Row
from pyspark.sql import functions as func
```

```
spark = SparkSession.builder.appName("worldcities").getOrCreate()
```

```
lines = spark.read.option("header", "true").option("inferSchema", "true").csv("worldcities.csv")
```

```
people.createOrReplaceTempView("worldcities")
```

```
# Select only city and population columns
worldcities = lines.select("city", "population")
```

```
# From worldcities we group by "city" and then compute average
worldcities.groupBy("city").avg("population").show()
```

```
# From worldcities we group by "city" and then compute average
worldcities.groupBy("city").avg("population").show()
```

```
+----------------+-----------------+
|            city|  avg(population)|
+----------------+-----------------+
|       Bangalore|        1.3707E7|
|          Benxi|        1567000.0|
|       Volgograd|        1015586.0|
|        Guwāhāti|         957352.0|
|       Kisangani|         935977.0|
|         Antwerp|         920000.0|
|         Palermo|491266.6666666667|
|         Morelia|         784776.0|
|        Jaboatão|         702621.0|
|         Edogawa|         690457.0|
|    Villahermosa|         640359.0|
|        Montería|         505334.0|
|       Samarkand|         504423.0|
|       Worcester|         176790.0|
|  Mazār-e Sharīf|         427600.0|
|Florencio Varela|         286354.5|
|        Namangan|         408500.0|
|        Horlivka|         256714.0|
|       Magdeburg|         238697.0|
|        Borūjerd|         234997.0|
+----------------+-----------------+
only showing top 20 rows
```

```
# Sorted
worldcities.groupBy("city").avg("population").sort("city").show()
```

```
+--------------------+---------------+
|                city|avg(population)|
+--------------------+---------------+
|           A Coruña|       370610.0|
|              Aachen|       247380.0|
|              Aadorf|         9036.0|
|             Aalborg|       122219.0|
|               Aalen|        68456.0|
|            Aalsmeer|        31728.0|
|              Aalst|        85715.0|
|             Aalten|        27011.0|
|              Aarau|        21268.0|
|             Aarburg|         8068.0|
|             Aarhus|       237551.0|
|             Aasiaat|         3347.0|
|                Aba|       500183.0|
|              Abadan|        40813.0|
|Abadia dos Dourados|         6972.0|
|  Abadiano Celayeta|         7658.0|
|          Abadiânia|        15757.0|
|             Abadla|        14364.0|
|             Abadou|        10602.0|
|         Abaetetuba|       159080.0|
+--------------------+---------------+
```

```
# With a custom column name
worldcities.groupBy("city").agg(func.round(func.avg("population"), 2).alias("population_avg")).sort("city").show()
```

```
+--------------------+--------------+
|                city|population_avg|
+--------------------+--------------+
|           A Coruña|      370610.0|
|              Aachen|      247380.0|
|              Aadorf|        9036.0|
|             Aalborg|      122219.0|
|               Aalen|       68456.0|
|            Aalsmeer|       31728.0|
|              Aalst|       85715.0|
|             Aalten|       27011.0|
|              Aarau|       21268.0|
|             Aarburg|        8068.0|
|             Aarhus|      237551.0|
|             Aasiaat|        3347.0|
|                Aba|      500183.0|
|              Abadan|       40813.0|
|Abadia dos Dourados|        6972.0|
|  Abadiano Celayeta|        7658.0|
|          Abadiânia|       15757.0|
|             Abadla|       14364.0|
|             Abadou|       10602.0|
|         Abaetetuba|      159080.0|
+--------------------+--------------+
only showing top 20 rows
```

```
 # Formatted more nicely
worldcities.groupBy("city").agg(func.round(func.avg("population"), 2)).sort("city").show()
```

```
+--------------------+------------------------+
|                city|round(avg(population), 2)|
+--------------------+------------------------+
|           A Coruña|                370610.0|
|              Aachen|                247380.0|
|              Aadorf|                  9036.0|
|             Aalborg|                122219.0|
|               Aalen|                 68456.0|
|            Aalsmeer|                 31728.0|
|              Aalst|                 85715.0|
|             Aalten|                 27011.0|
|              Aarau|                 21268.0|
|             Aarburg|                  8068.0|
|             Aarhus|                237551.0|
|             Aasiaat|                  3347.0|
|                Aba|                500183.0|
|              Abadan|                 40813.0|
|Abadia dos Dourados|                  6972.0|
|  Abadiano Celayeta|                  7658.0|
|          Abadiânia|                 15757.0|
|             Abadla|                 14364.0|
|             Abadou|                 10602.0|
|         Abaetetuba|                159080.0|
+--------------------+------------------------+
only showing top 20 rows
```

Find Word Count using SparkSQL DataFrame

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as func

spark = SparkSession.builder.appName("WordCount").getOrCreate()

sample_text = """In the beginning, God created the heavens and the earth.
Now the earth was formless and empty, darkness was over the surface of the deep,
and the Spirit of God was hovering over the waters.

And God said, "Let there be light," and there was light. God saw that the light was good,
and he separated the light from the darkness. God called the light "day," and the darkness he called "night."
And there was evening, and there was morning—the first day.
"""

with open("book.txt", "w") as file:
    file.write(sample_text)

# Read each line of my book into a dataframe
inputDF = spark.read.text("book.txt")

# Split using a regular expression that extracts words
words = inputDF.select(func.explode(func.split(inputDF.value, "\\W+")).alias("word"))
words.filter(words.word != "")
```

```
DataFrame[word: string]
```

```
# Normalize everything to lowercase
lowercaseWords = words.select(func.lower(words.word).alias("word"))

# Count up the occurrences of each word
wordCounts = lowercaseWords.groupBy("word").count()

# Sort by counts
wordCountsSorted = wordCounts.sort("count")

# Show the results.
wordCountsSorted.show(wordCountsSorted.count())
```

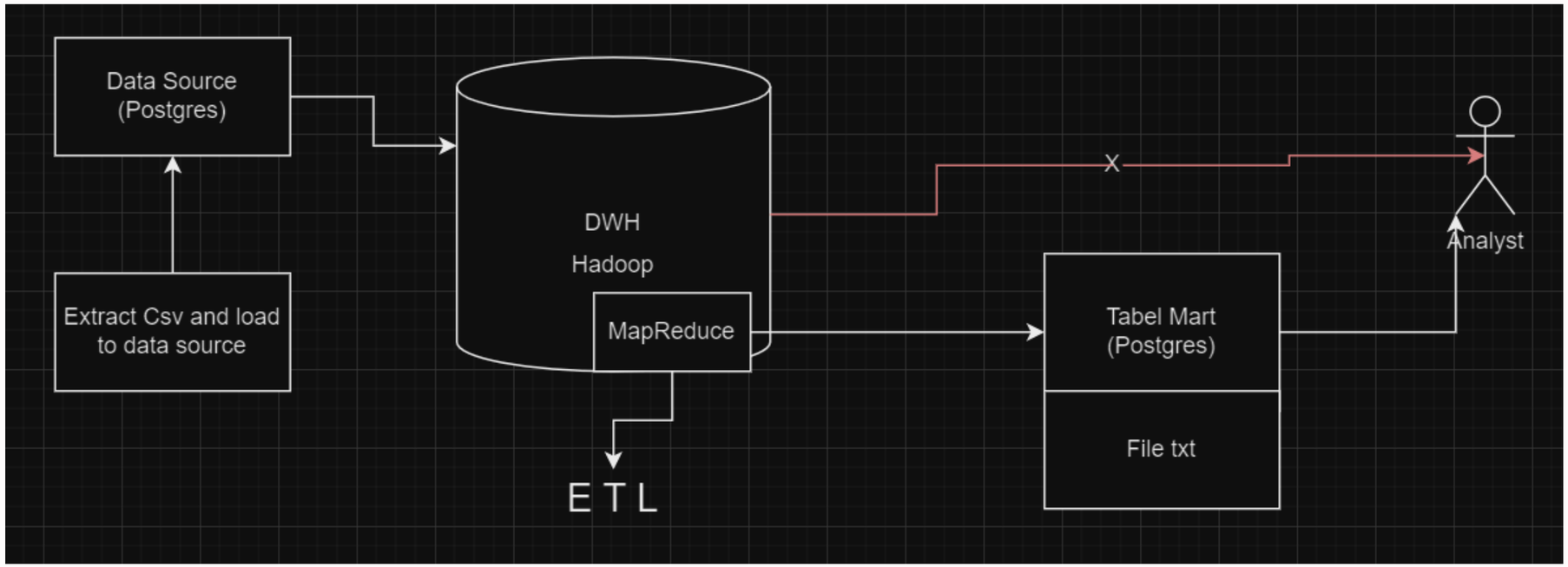
```
+---------+-----+
|     word|count|
+---------+-----+
|   waters|    1|
|separated|    1|
|   spirit|    1|
| hovering|    1|
|      saw|    1|
|  surface|    1|
|       in|    1|
|       be|    1|
|  created|    1|
|    night|    1|
|beginning|    1|
|      now|    1|
|  morning|    1|
|  heavens|    1|
|     said|    1|
|     from|    1|
|  evening|    1|
|     that|    1|
|     deep|    1|
|      let|    1|
|     good|    1|
| formless|    1|
|    first|    1|
|    empty|    1|
|      day|    2|
|    earth|    2|
|       of|    2|
|     over|    2|
|       he|    2|
|   called|    2|
| darkness|    3|
|    there|    4|
|    light|    5|
|      god|    5|
|      was|    7|
|         |    7|
|      and|    9|
|      the|   14|
+---------+-----+
```

DigitalSkola

PROJECT 6

Background Project

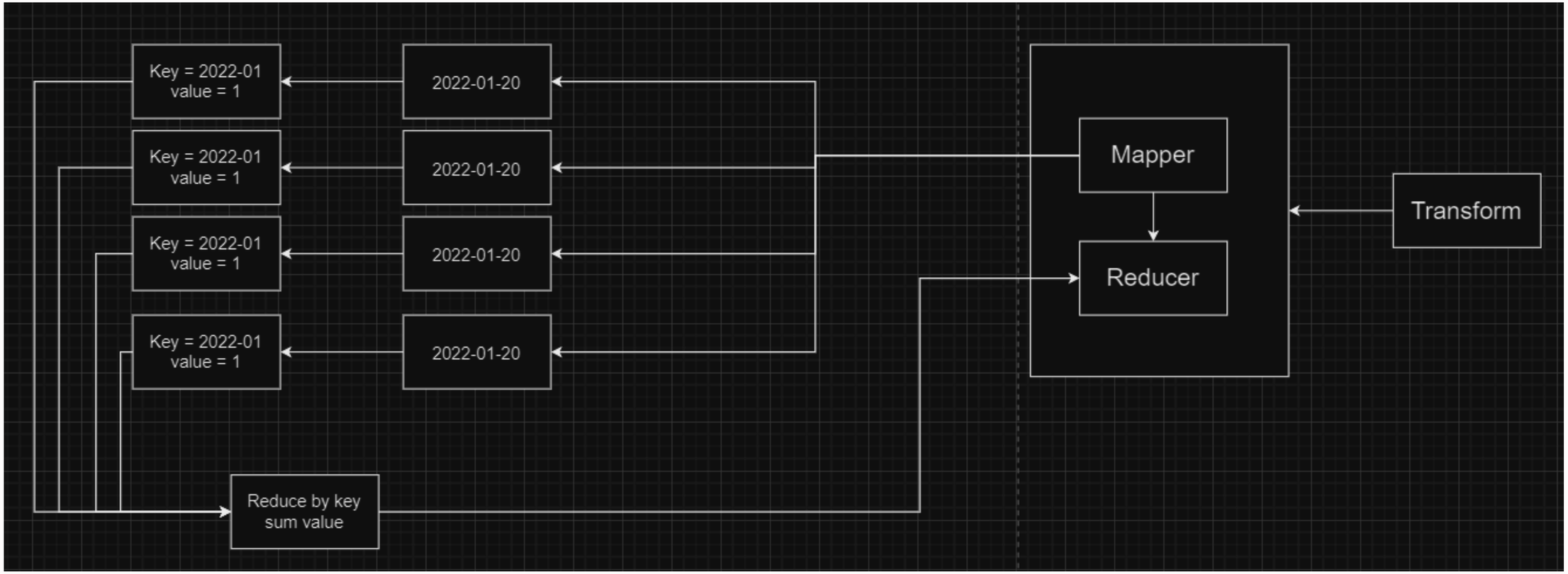
Projek ini merupakan projek lanjutan dari projek ke 3, yaitu mengenai pembuatan batch processing untuk data migrasi. Pada project ini, kita membuat tambahan task untuk membuat tabel data mart yang nantinya akan digunakan oleh data analis untuk membuat dashboard report order setiap bulannya. Proses yang dilakukan tidak boleh lebih dari 1 jam. Oleh karena itu, pada projek ini kita akan menggunakan HDFS agar proses migrasi datanya lebih cepat.



Flow Project 6

DigitalSkola

Background Project



Flow Mapper dan Reducer

DigitalSkola

Requirement tools

Hadoop

3 Opsi untuk menggunakan hadoop dalam project ini:

- install hadoop di os device masing2
- install hadoop via docker
- Menggunakan hadoop server dari tutor (yang akan digunakan kali ini)

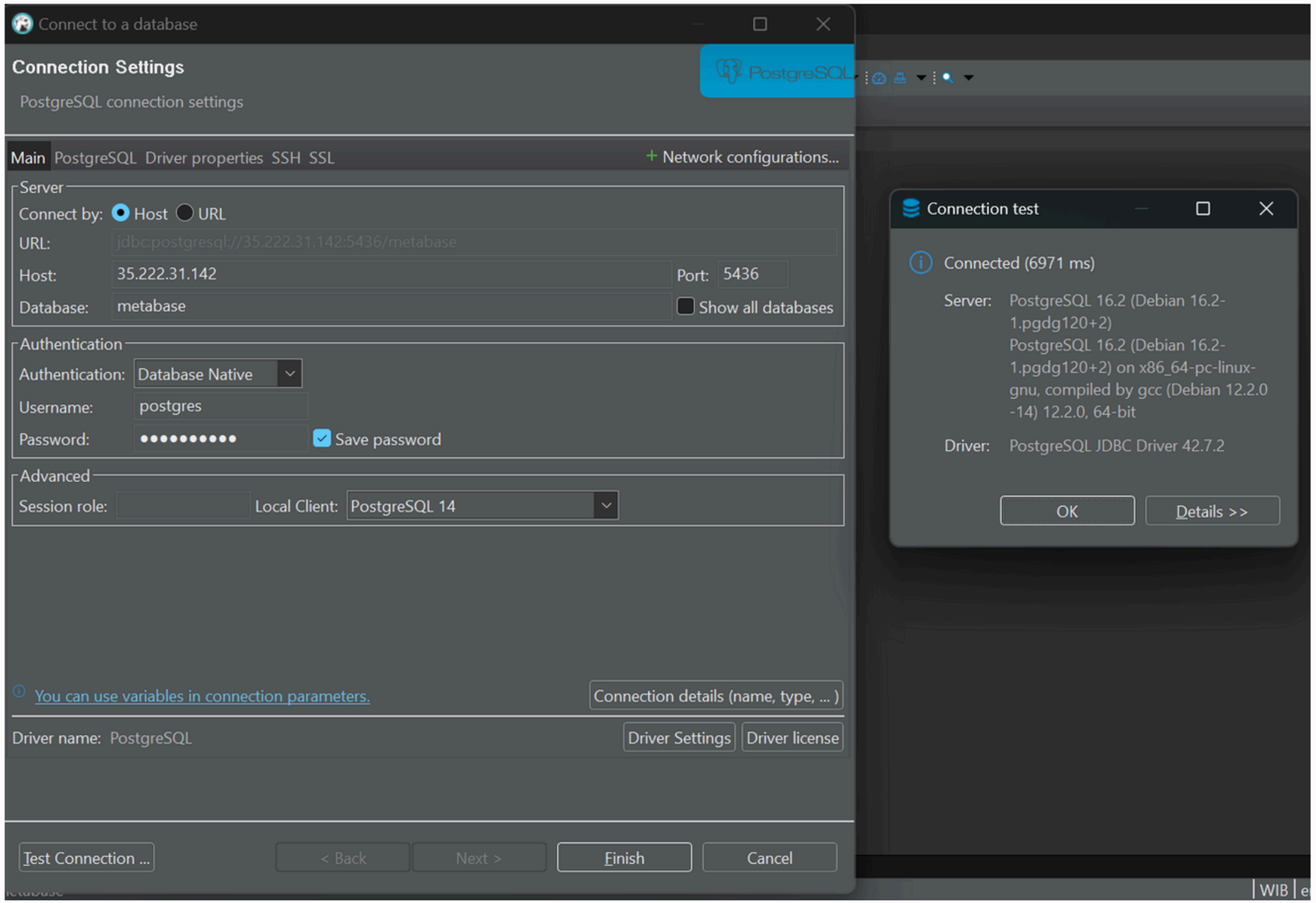
Postgresql

Postgresql bisa menggunakan via docker/server atau direct install. Pada project ini, kita akan menggunakan server dari tutor.

GUI Postgres

Beberapa gui postgresql yang dapat digunakan antara lain pgadmin, Dbeaver, OmniDB, dan lainnya. Pada project ini kita akan menggunakan Dbeaver.

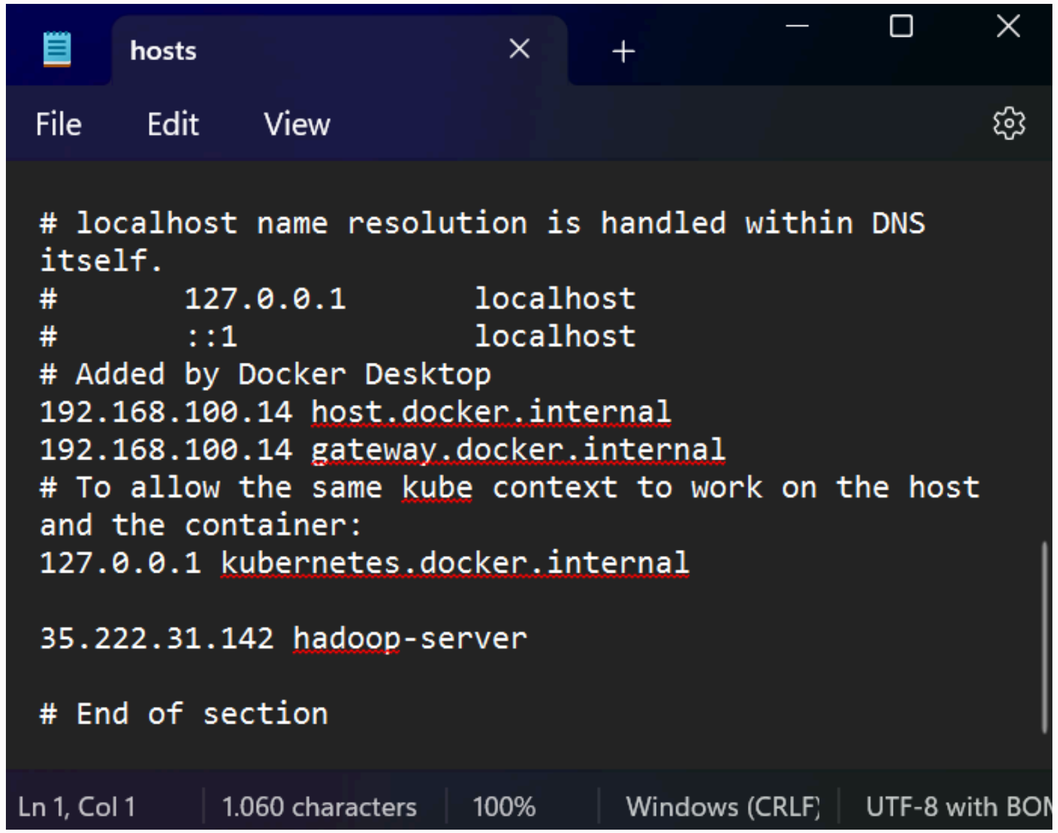
Step 1: Connect ke postgresql



Lakukan koneksi ke postgresql server yang disediakan tutor dengan konfigurasi berikut.

Host: 35.222.31.142
db: metabase
user: postgres
password: sib6_admin
port: 5436

Step 2: Add host ke device untuk mengakses hadoop-server



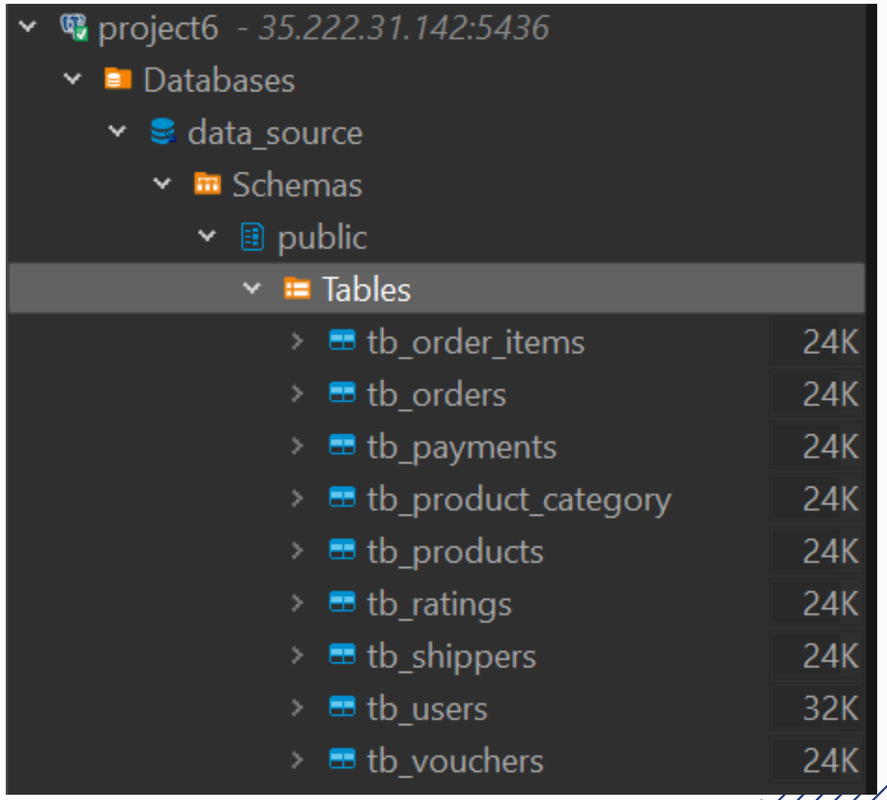
Tambahkan konfigurasi host pada file hosts di device: "35.222.31.142 hadoop-server"

note: file hosts bisa ditemukan pada directory C:\Windows\System32\drivers\etc

Step 3: Buat schema pada Postgres SQL sebagai raw data

```sql
CREATE TABLE tb_users (
    user_id INT NOT NULL,
    user_first_name VARCHAR(255) NOT NULL,
    user_last_name VARCHAR(255) NOT NULL,
    user_gender VARCHAR(50) NOT NULL,
    user_address VARCHAR(255),
    user_birthday DATE NOT NULL,
    user_join DATE NOT NULL,
    PRIMARY KEY (user_id)
);

CREATE TABLE tb_payments (
    payment_id INT NOT NULL,
    payment_name VARCHAR(255) NOT NULL,
    payment_status BOOLEAN NOT NULL,
    PRIMARY KEY (payment_id)
);

CREATE TABLE tb_shippers (
    shipper_id INT NOT NULL,
    shipper_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (shipper_id)
);

CREATE TABLE tb_ratings (
    rating_id INT NOT NULL,
    rating_level INT NOT NULL,
    rating_status VARCHAR(255) NOT NULL,
    PRIMARY KEY (rating_id)
);

CREATE TABLE tb_product_category (
    product_category_id INT NOT NULL,
    product_category_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (product_category_id)
);

CREATE TABLE tb_vouchers (
    voucher_id INT NOT NULL,
    voucher_name VARCHAR(255) NOT NULL,
    voucher_price INT,
    voucher_created DATE NOT NULL,
    user_id INT NOT NULL,
    PRIMARY KEY (voucher_id),
    CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES tb_users (user_id)
);
```

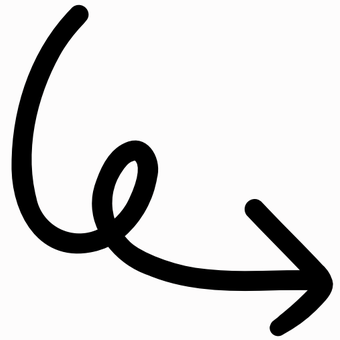
Query to create table on postgres



Terdapat 9 tabel pada data raw tersebut, yaitu order_items, orders, payments, product_category, products, ratings, shippers, users, dan vouchers.

Step 4: Setup connection ke postgres

File config.json digunakan untuk konfigurasi database marketplace_prod (data raw), data warehouse. dan hadoop server



```
{
    "marketplace_prod": {
        "host": "35.222.31.142",
        "db": "data_source",
        "user": "postgres",
        "password": "sib6_admin",
        "port": "5436"
    },
    "dwh": {
        "host": "35.222.31.142",
        "db": "dwh",
        "user": "postgres",
        "password": "sib6_admin",
        "port": "5436"
    },
    "hadoop":{
        "client": "http://hadoop-server:9870/"
    }
}
```

DigitalSkola

Step 4: Setup connection ke postgres

File connection.py digunakan untuk mengatur koneksi ke database PostgreSQL. File ini berisi dua fungsi utama, config dan get_conn.

Fungsi config membaca file config.json dari direktori kerja saat ini dan mengambil konfigurasi database berdasarkan parameter connection_db. Fungsi get_conn mencoba membuat koneksi ke database PostgreSQL menggunakan konfigurasi yang diberikan.

```python
import os
import json
import psycopg2
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 get_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 to postgress {name_conn}')
        engine = create_engine(f"postgresql+psycopg2://{conf['user']}:{co
        return conn, engine
    except Exception as e:
        print(f'[INFO] cannot connect to postgress {name_conn}')
        print(str(e))
```

DigitalSkola

Step 5: Buat query sql untuk design data warehouse dan query untuk mengambil data dari schema data raw

```sql
DROP TABLE IF EXISTS dim_order
CREATE TABLE dim_orders (
    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)
    );
```

query sql untuk schema data warehouse

query sql untuk mengambil data dari schema data raw

```sql
SELECT  order_id,
        order_date,
        a.user_id,
        b.payment_name,
        c.shipper_name,
        order_price,
        order_discount,
        d.voucher_name,
        d.voucher_price,
        order_total,
        e.rating_status
FROM tb_orders a
    LEFT JOIN tb_payments b ON a.payment_id = b.payment_id
    LEFT JOIN tb_shippers c ON a.shipper_id = c.shipper_id
    LEFT JOIN tb_vouchers d ON a.voucher_id = d.voucher_id
    LEFT JOIN tb_ratings e ON a.rating_id = e.rating_id ;
```

Step 6: Melakukan proses ETL

main.py

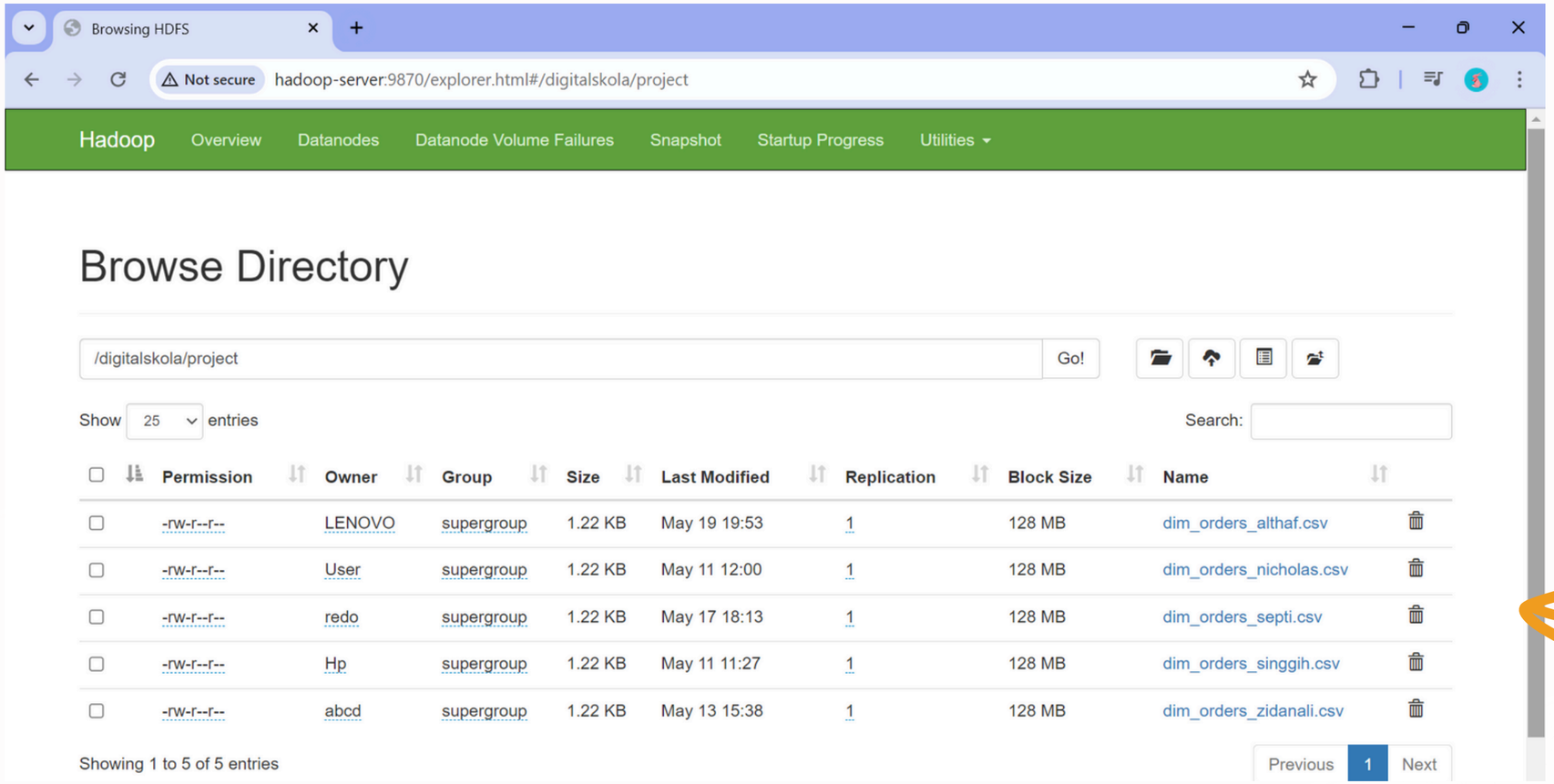
```python
import os
import connection
import sqlparse
import pandas as pd


if __name__ == '__main__':
    print('[INFO] Service ETL is Starting ...')

    # 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 hadoop
    conf_hadoop = connection.config("hadoop")
    client = connection.hadoop_conn(conf_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()
```

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

    # ingest data to hadoop
    with client.write(f'/digitalskola/project/dim_orders_althaf.csv', encoding='utf-8') as writer:
        df.to_csv(writer, index=False)

    print('[INFO] Service ETL is Success ...')
except Exception as e:
    print('[INFO] Service ETL is Failed ...')
    print(e)
```

DigitalSkola

Tampilan jika data migrasi berhasil dijalankan pada terminal

```
(project3) C:\Users\LENOVO\Stupen\SIB-Project6>python main.py
[INFO] Service ETL is Starting ...
[INFO] Success connect PostgreSQL DataSource
[INFO] Success connect PostgreSQL DataWarehouse
[INFO] Success connect to HADOOP ...
[INFO] Service ETL is Running ...
[INFO] Service ETL is Success ...
```



Tampilan jika data migrasi berhasil dijalankan pada hadoop server

DigitalSkola
Uncover The World of Digital Skills with Us

```python
from mrjob.job import MRJob
from mrjob.step import MRStep

import csv
import json

cols = 'order_id,order_date,user_id,payment_name,shipper_name,order_price,order_discount,v

def csv_readline(line):
    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):
        row = dict(zip(cols, csv_readline(line)))

        if row['order_id'] != 'order_id':
            yield row['order_date'][0:7], 1

    def reducer(self, key, values):
        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

if __name__ == '__main__':
    OrderDateCount.run()
```

Step 7: Melakukan proses map reduce

Siapkan file mapReduce.py untuk melakukan proses map reduce pada hadoop

DigitalSkola

Jalankan mapreduce hadoop menggunakan google colab

```
[1] !pip install hdfs
```

Show hidden output

```
[2] !pip install pywebhdfs
```

Show hidden output

```
[3] !pip install mrjob
```

Show hidden output

```
[5] # download hadoop
from pywebhdfs.webhdfs import PyWebHdfsClient
from pprint import pprint
import pandas as pd
from datetime import datetime

hdfs=PyWebHdfsClient(host='hadoop-server',port='9870', user_name='Hp')
filetime = datetime.now().strftime('%Y-%m-%d')

my_file = f'/digitalskola/project/dim_orders_althaf.csv'
data = hdfs.read_file(str(my_file))
data = data.decode().split('\n')
data_list = []
for item in data:
  data_list.append(item.split(','))
pd.DataFrame(data_list[1:], columns=data_list[0]).to_csv(f'dim_orders_althaf_{filetime}.csv', index=False)
```

```
[6] from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[8] # run mapreduce from file hadoop
!python mapReduce.py dim_orders_althaf_2024-05-19.csv > mart_table_2024-05-19.txt
```

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/mapReduce.root.20240519.152230.386446
Running step 1 of 2...
Running step 2 of 2...
job output is in /tmp/mapReduce.root.20240519.152230.386446/output
Streaming final output from /tmp/mapReduce.root.20240519.152230.386446/output...
Removing temp directory /tmp/mapReduce.root.20240519.152230.386446...
```

Hasil map-reduce

mart_table_2024-05-19.txt

```
1 ""   1
2 "2022-01" 2
3 "2022-02" 1
4 "2022-03" 1
5 "2022-04" 1
6 "2022-05" 2
7 "2022-06" 2
8 "2022-07" 3
9
```

TERIMA KASIH