Rakamin Academy X KALBE Nutritionals

Virtual Internship Experience - Final Project Data Engineer

Rakamin VIX Program

Presented by
Ricky Suhanry

Rakamin Academy

Ricky Suhanry

Data Analytics | Engineer

Enthusiast tech person with background in software engineer and specialist in big data engineering with cloud computing. I've applied my programming skills as data bootcamp student, where I produced clean, validation-ready code for various projects using pandas, numpy, numpy, and matplotlib. I'm well-equipped to contribute to challenging projects and drive innovation in the field of technology

Insert Your Experience

Kalbe Nutritionals - VIX Program

Data Engineer

July - August 2023

Kimia Farma - VIX Program

Big Data Analyst

February - March 2023

Fintech Startup

Software Engineer - Backend

May - August 2022

Case Study I

Rakamin Academy

Create a shell/bash script to check whether directory exists inside a given path.

Variables:
■ path=/hdfs/data/data1
name_of_directory=data1

○ Conditions:
■ If directory exists inside the path:
● Echo “There is [Directory Name] Directory Exists!”

■ If not:
● Echo “[Directory Name] Directory Not Exists!”
● Create a directory inside the path.

○ Final Step:
■ Create a crontab syntax to run the script at 07:00 AM Daily

main1.sh

```bash
#! /bin/bash
path="/hdfs/data/data1"
name_of_directory="data1"

full_path="${path} / ${name_of_directory}"

# This check if the directory is exists
if [ -d "${full_path}" ] ; then
  echo "There is $full_path Directory Exists!"
## note: -d = list directories only, do not included files

else
then
  echo "$full_path Directory Not Exists!"

  # create directory inside the path
  mkdir -p "${full_path}"
## note: -p = tells mkdir to create the specified directory
  exit
fi

# Schedule a crontab to run script at 07:00 AM Daily
0 7 * * * /path/to/script1.sh | crontab -l
```

Case Study II

Rakamin Academy

Using the question number 1 script, add another condition if directory exists inside the path

Variables:

- filename_excel=daily_market_price.xlsx
- source_dir=/local/data/market
- target_dir=Refer to Question Number 1 Path

○ Conditions:

- Copy file from source directory into target directory.
- Create a log file inside the same path with “File Moved Successfully” as a log content if success.

main2.sh

```bash
#! /bin/bash
path="/hdfs/data/data1"
name_of_directory="data1"
filename_excel="daily_market_price.xlsx"
source_dir="/local/data/amrket"
target_dir="${path} / ${name_of_directory}"

# This check if the directory is exists
if [ -d "${full_path}" ] ; then
  echo "There is $full_path Directory Exists!"

  # Copy file from source to target directory
  cp "${source_dir}" "${target_dir}"

  # Create a log file inside the same path
  echo "File Moved Successfully" >> ${target_dir}/report.log
  # When you redirect using > | >>, the contents of the target

else
  echo "$full_path Directory Not Exists!"

  # create directory inside the path
  mkdir -p "${full_path}"

  exit
fi
```

Case Study III

Rakamin Academy

3. Complete below Syntax {Highlighted Sentence} to insert data from Python to MySQL.

Install Database (pilih salah satu/ gunakan yang sudah ada)

- PostgreSQL: PostgreSQL: Downloads
- MySQL: MySQL :: MySQL Downloads

Code

Rakamin Academy

```python
# Melakukan percobaan koneksi
def connection():
  # Set the conn to 'None'
  conn = None
  try:
    print("Connecting to the PostgreSQL!")
    # Set the connection parameters
    conn = psycopg2.connect (
      database = "---", # database name
      host = "---", # host name
      port = "---", # port
      user = "---", # username
      password = "---" # password
    )
    print("Connecting successful!")

  except OperationalError as err:
    # call function for error
    db_error_tracing(err)
    # Rollback database if connection was fail
    conn.rollback()

  return conn
```

Case Study IV

Rakamin Academy

Convert this instruction into SQL Query Language.

- Create a database with 'KALBE' as the name.
- Inside the database, create a table with the name 'Inventory', with columns Item_code, Item_name, Item_price, and Item_total. Choose its own best data type and the length of it according to best practice. Choose one unique column as a primary key and decide columns constraints.
- Insert below data into the table:

Item_code	Item_name	Item_price	Item_total
2341	Promag Tablet	3000	100
2342	Hydro Coco 250ML	7000	20
2343	Nutrive Benecol 100ML	20000	30
2344	Blackmores Vit C 500Mg	95000	45
2345	Entrasol Gold 370G	90000	120

- Show Item_name that has the highest number in Item_total.
- Update the Item_price of the output of question bullet
- What will happen if we insert another Item_name with Item_code of 2343 into the table?
- Delete the Item_name that has the lowest number of Item_total.

Code

```python
# Melakukan pembuatan table baru di dalam postgreSQL

def create_postgres_tables():
  # Connect to the database
  conn = db_connection()
  conn.autocommit = True
  # Membuat object cursor koneksi
  cursor = conn.cursor()

  try:
    # Dropping table iris if exists
    cursor.execute("DROP TABLE IF EXISTS inventory")

    # Creating a table
    cursor.execute("""CREATE TABLE inventory (item_code INTEGER PRIMARY KEY,
                      item_name VARCHAR(25) NOT NULL, item_price INTEGER NOT NULL,
                      item_total INTEGER NOT NULL, Row_id INTEGER NOT NULL) """)
    print("inventory table is created successfully!")

    # Melakukan perubahan (commit) pada DB
    conn.commit()
    # Closing the cursor & connection
    cursor.close()

    conn.close()

  except OperationalError as err:
    # pass exception to function
    db_error_tracing(err)
    # Rollback database if connection was fail
    conn.rollback()
    # Set the close to cursor
    cursor.close()
```

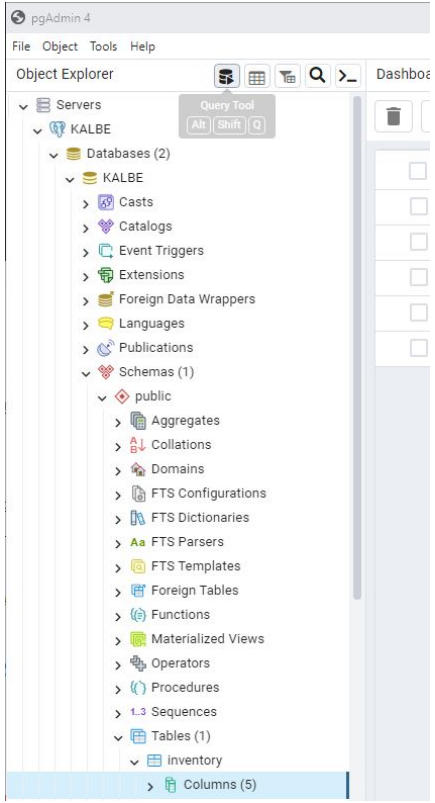
Output

```
# Close the connection
conn.close()
```

✓ 0.1s

```
Connecting to the PostgreSQL!
Connecting successful!
Connecting to the PostgreSQL!
Connecting successful!
inventory table is created successfully!
Connecting to the PostgreSQL!
Connecting successful!
Data convert to dataFrame successful!
C:\Users\RickyS-PC\AppData\Local\Temp\ipykernel_21176\27345185.py:21: FutureWarning:
  (int(row[0]), str(row[1]), str(row[2]), int(row[3]), int(row[4])))
```

	Item_code	Item_name	Item_price	Item_total	Row_id
0	2341	Promag Tablet	3000	100	1



Code

```python
# Insert data from CSV file into postgreSQL
def write_to_postgres():
    # Connect to the database
    conn = connection()
    conn.autocommit = True
    cursor = conn.cursor()

    inserted_row_count = 0

    for _, row in df_kalbe_store.iterrows():
        count_query = f"""SELECT COUNT(*) FROM inventory WHERE Row_id = {row['Row_id']}"""
        cursor.execute(count_query)
        result = cursor.fetchone()

        if result[0] == 0:
            inserted_row_count += 1
            cursor.execute("""INSERT INTO inventory (item_code, item_name,
                            item_price, item_total, Row_id)
                            VALUES (%s, %s, %s, %s, %s)""",
                            (int(row[0]), str(row[1]), str(row[2]), int(row[3]), int(row[4])))

    cursor = conn.cursor()
    try:
        # Melakukan perubahan (commit) pada DB
        conn.commit()
        print("Data convert to dataFrame successful!")

    except OperationalError as err:
        # call function for error
        db_error_tracing(err)
        # Rollback database if connection was fail
        conn.rollback()
        # Close cursor
        cursor.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!
Data convert to dataFrame successful!
C:\Users\RickyS-PC\AppData\Local\Temp\ipykernel_21176\27345185.py:21: FutureWarning: Series.
  (int(row[0]), str(row[1]), str(row[2]), int(row[3]), int(row[4])))
```

	Item_code	Item_name	Item_price	Item_total	Row_id
0	2341	Promag Tablet	3000	100	1
1	2342	Hydro Coco 250ML	7000	20	2

Output
In PostgreSQL
Terminal

Administrator: Command Prompt - psql --host=[redacted] --port=[redacted] --dbname=[redacted] --username=

```
ERROR:  syntax error at or near "SELECT"
LINE 2: SELECT * FROM Inventory;
        ^
KALBE=# SELECT * FROM Inventory;
 item_code |        item_name        | item_price | item_total | row_id
----------+-------------------------+------------+------------+--------
      2341 | Promag Tablet           |       3000 |        100 |      1
      2342 | Hydro Coco 250ML        |       7000 |         20 |      2
```

Code

```
#--- Mencari value item_name yang memiliki item_total paling banyak

# Connect to the database
conn = connection()
conn.autocommit = True

# Declare cursor for connection
cursor = conn.cursor()

# Execute query
query = """SELECT item_name FROM Inventory WHERE item_total =
           (SELECT MAX(item_total) FROM Inventory LIMIT 1)"""
show_highest_item = pd.read_sql_query(query, conn)
print(show_highest_item)

# Close cursor
cursor.close()

# Close the connection
conn.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!
            item_name
0  Entrasol Gold 370G
```

Output
In PostgreSQL
Terminal

Administrator: Command Prompt - psql --host=[redacted] --port=[redacted] --dbname=[redacted] --username=[redacted] --password

```
KALBE=# SELECT item_name FROM Inventory WHERE item_total = (SELECT MAX(item_total) FROM Inventory);
     item_name
--------------------
 Entrasol Gold 370G
(1 row)
```

Code

```python
#--- Mengubah value item_price dari row yang memiliki item_total terbanyak banyak

# Connect to the database
conn = connection()
conn.autocommit = True

# Declare cursor for connection
cursor = conn.cursor()

try:
    update_item_price = 125000
    # Execute query
    query = f"UPDATE Inventory SET item_price= {update_item_price} WHERE item_total = (SELECT MAX(item_total) F
    cursor.execute(query)
    # Melakukan perubahan (commit) pada DB
    conn.commit()
    print("Update item_price value successful!")

except OperationalError as err:
    # call function for error
    db_error_tracing(err)
    # Rollback database if connection was fail
    conn.rollback()
    # Close cursor
    cursor.close()

query = "SELECT * FROM Inventory"
show_update_item = pd.read_sql_query(query, conn)
print(show_update_item)

# Close cursor
cursor.close()

# Close the connection
conn.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!
Data convert to dataFrame successful!
C:\Users\RickyS-PC\AppData\Local\Temp\ipykernel_21176\27345185.py:21:
  (int(row[0]), str(row[1]), str(row[2]), int(row[3]), int(row[4])))
```

Item_code	Item_name	Item_price	Item_total	Row_id
4	2345	Entrasol Gold 370G	90000	120	5

```
Connecting to the PostgreSQL!
Connecting successful!
Update item_price value successful!
```

item_code	item_name	item_price	item_total	row_id
4	2345	Entrasol Gold 370G	125000	120	5

Output
In PostgreSQL
Terminal

Administrator: Command Prompt - psql --host= -port= --dbname= --username=

```
KALBE=# SELECT * FROM Inventory;
```

item_code	item_name	item_price	item_total	row_id
2345	Entrasol Gold 370G	125000	120	5

(5 rows)

Code

```python
#--- Menginput data baru ke dalam table invetory

# Connect to the database
conn = connection()
conn.autocommit = True

# Declare cursor for connection
cursor = conn.cursor()

try:
    # Input data as list of dicts and using named parameters to avoid duplicating data.
    input_new_data = [{'item_code': 2343, 'item_name': 'Vicks F44', 'item_price': 25000,
                       'item_total': 25, 'row_id': 6}]

    # Using execute_batch to inserts using a multi-line statement
    execute_batch(cursor, """INSERT INTO Inventory values(%(item_code)s, %(item_name)s,
                          %(item_price)s, %(item_total)s, %(row_id)s)', input_new_data)""")
    conn.commit()

except OperationalError as err:
    # call function for error
    db_error_tracing(err)
    # Rollback database if connection was fail
    conn.rollback()
    # Close cursor
    cursor.close()

# Execute query
query = "SELECT * FROM Inventory"
show_update_item = pd.read_sql_query(query, conn)
print(show_update_item)

# Close cursor
cursor.close()

# Close the connection
conn.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!

---------------------------------------------------------------------------
UniqueViolation                           Traceback (most recent call last)
[redacted]
     12 input_new_data = [{'item_code': 2343, 'item_name': 'Vicks F44', 'item_price'
     13 # Using execute_batch to inserts using a multi-line statement
---> 14 execute_batch(cursor, 'INSERT INTO Inventory values(%(item_code)s, %(item_na
     16 # Execute query #2- check data in the table
     17 query = "SELECT * FROM Inventory"

[redacted]
   1214 for page in _paginate(argslist, page_size=page_size):
   1215     sqls = [cur.mogrify(sql, args) for args in page]
-> 1216     cur.execute(b";".join(sqls))

UniqueViolation: duplicate key value violates unique constraint "inventory_pkey"
DETAIL:  Key (item_code)=(2343) already exists.
```

Code

```python
#--- Menghapus data baru ke dalam table

# Connect to the database
conn = connection()
conn.autocommit = True

# Declare cursor for connection
cursor = conn.cursor()

try:
    # Execute query
    query = "DELETE FROM Inventory WHERE item_total = (SELECT MIN(item_total) FROM Inventory LIMIT 1)"
    cursor.execute(query)

    # Melakukan perubahan (commit) pada DB
    conn.commit()
    print("Delete item_name value successful!")

except OperationalError as err:
    # call function for error
    db_error_tracing(err)
    # Rollback database if connection was fail
    conn.rollback()
    # Close cursor
    cursor.close()

# Execute query
query = "SELECT * FROM Inventory"
show_update_item = pd.read_sql_query(query, conn)
print(show_update_item)

# Close cursor
cursor.close()

# Close the connection
conn.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!
Delete item_name value successful!
```

Output
In PostgreSQL
Terminal

Administrator: Command Prompt - psql --host=[redacted] -port=[redacted] --dbname=[redacted] --username=[redacted] --password

```
        ^
KALBE=# DELETE FROM Inventory WHERE item_total = (SELECT MIN(item_total) FROM Inventory);
DELETE 1
KALBE=# SELECT * FROM Inventory;
 item_code |       item_name        | item_price | item_total | row_id
-----------+------------------------+------------+------------+--------
      2341 | Promag Tablet          |       3000 |        100 |      1
      2343 | Nutrive Benecol 100ML  |      20000 |         30 |      3
```

Case Study V

Rakamin Academy

5. Create a Query to display all customer orders where purchase amount is less than 100 or exclude those orders which order date is on or greater than 25 Aug 2022 and customer id is above 2001. Sample table: customer_orders

order_no	purchase_amount	order_date	customer_id	salesman_id
10001	150	2022-10-05	2005	3002
10009	270	2022-09-10	2001	3005
10002	65	2022-10-05	2002	3001
10004	110	2022-08-17	2009	3003
10007	948	2022-09-10	2005	3002
10005	2400	2022-07-27	2007	3001

Code

```
#--- Menampilkan semua daftar customer yang melakukan pembelian barang kurang dari
#--- 100 item atau customer yang melakukan pemesanan lewat dari tanggal 25 Agust 2022
#--- dan customer_id lebih besar dari 2001

# Connect to the database
conn = connection()
conn.autocommit = True

# Declare cursor for connection
cursor = conn.cursor()

# Execute query
query = """SELECT * FROM customer_transaction WHERE (purchase_amount < 100) OR
        (order_date > '2022-08-25' AND customer_id > 2001)"""
show_customer_data = pd.read_sql_query(query, conn)
print(show_customer_data)

# Close cursor
cursor.close()

# Close the connection
conn.close()
```

Output

Rakamin Academy

Output
In VSCode

```
Connecting to the PostgreSQL!
Connecting successful!
   order_no  purchase_amount  order_date  customer_id  salesman_id  row_id
0     10001              150  2022-10-05         2005         3002       1
1     10002               65  2022-10-05         2002         3001       3
```

Output
In PostgreSQL
Terminal

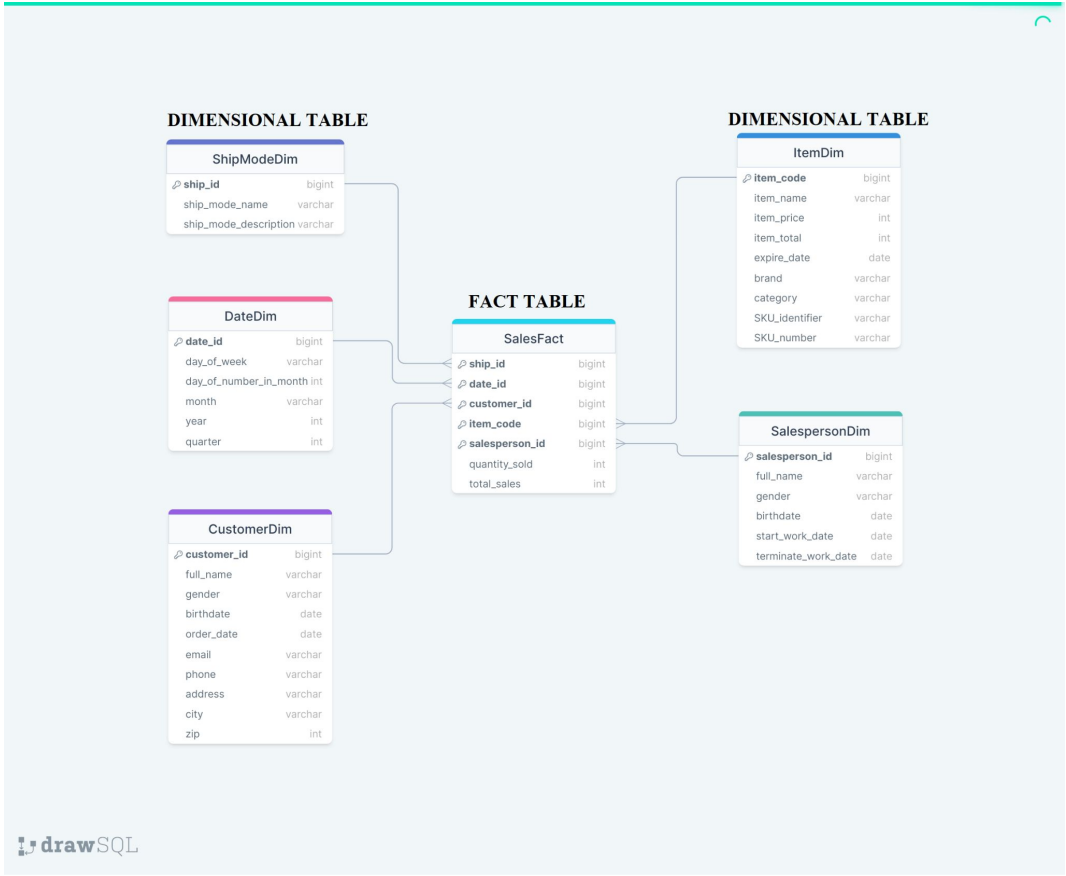
Administrator: Command Prompt - psql --host=[redacted] --port=[redacted] --dbname=[redacted] --username=[redacted] --password

```
KALBE=# SELECT * FROM customer_transaction WHERE (purchase_amount < 100) OR (order_date > '2022-08-25' AND customer_id >
 2001);
 order_no | purchase_amount | order_date | customer_id | salesman_id | row_id
----------+-----------------+------------+-------------+-------------+--------
    10001 |             150 | 2022-10-05 |        2005 |        3002 |      1
    10002 |              65 | 2022-10-05 |        2002 |        3001 |      3
```

Case Study VII

7. Create a simple star schema for KALBE database consist of 1 Fact and 5 Dimensions using Physical Data Model Theory.

Output


DIMENSIONAL TABLE
ShipModeDim
ship_id bigint
ship_mode_name varchar
ship_mode_description varchar
DateDim
date_id bigint
day_of_week varchar
day_of_number_in_month int
month varchar
year int
quarter int
CustomerDim
customer_id bigint
full_name varchar
gender varchar
birthdate date
order_date date
email varchar
phone varchar
address varchar
city varchar
zip int
FACT TABLE
SalesFact
ship_id bigint
date_id bigint
customer_id bigint
item_code bigint
salesperson_id bigint
quantity_sold int
total_sales int
DIMENSIONAL TABLE
ItemDim
item_code bigint
item_name varchar
item_price int
item_total int
expire_date date
brand varchar
category varchar
SKU_identifier varchar
SKU_number varchar
SalespersonDim
salesperson_id bigint
full_name varchar
gender varchar
birthdate date
start_work_date date
terminate_work_date date
drawSQL

Rakamin Academy

My Work Contact:

www.linkedin.com/in/ricky-suhanry

https://github.com/abliskan

ricky.suhanry107@gmail.com

More detail of this project:

https://github.com/abliskan/Rakamin-VIX-Kalbe-Nutritionals-DE

Thank You

Rakamin Academy X KALBE Nutritionals