Laporan Praktikum

Algoritma dan Pemrograman

“Studi kasus mengenai Pengolahan Data Penjualan Produk”

Disusun Oleh:

Muhammad Rossi Ramadhan (250535625057)

Shofiyah La Amiri (250535626476)

Nova Indriansyah (250535602240)

OFFERING C

PROGRAM STUDI S1 TEKNIK INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI MALANG

2025

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Allah SWT. yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas laporan praktikum ini tepat pada waktunya.

Adapun tujuan dari penulisan laporan ini adalah untuk memenuhi tugas pada mata kuliah Algoritma dan Pemrograman, serta sekaligus untuk menambah wawasan terkait pemrograman dengan menggunakan bahasa pemrograman python dalam menyelesaikan tugas dengan judul “Studi kasus mengenai Pengolahan Data Penjualan Produk”.

Terlebih dahulu penulis mengucapkan terima kasih kepada Bapak M. Jauharul Fuady, selaku dosen pengampu mata kuliah Alrgoritma dan Pemrograman yang telah memberikan tugas ini sehingga penulis dapat belajar dan berkembang.

Penulis juga mengucapkan terima kasih kepada semua pihak, yakni teman-teman kelompok yang telah berkontribusi dalam menyelesaikan tugas ini.

Terakhir, penulis menyadari bahwa tugas ini masih jauh dari kata sempurna, mengingat kemampuan yang dimiliki penulis. Oleh karena itu, kritik dan saran yang membangun dari semua pihak sangat penulis butuhkan.

Malang, 05 Oktober 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Setiap perusahaan terutama perusahaan dalam bidang bisnis pastinya memiliki data penjualan yang menunjukkan apakah sebuah produk yang dipasarkan laku atau tidak. Data penjualan inilah yang nantinya digunakan sebagai acuan dalam mengambil setiap keputusan perusahaan.

Agar mudah dianalisis, data biasanya disusun dalam bentuk matriks, dimana setiap baris mewakili produk dan setiap kolom mewakili bulan. Hal ini nantinya dapat mempermudah dalam menentukan jumlah penjualan setiap produk, mengidentifikasi produk dengan penjualan tertinggi dan terendah, serta menunjukkan lonjakan pemasaran pada tiap produk dalam suatu periode.

Dalam studi kasus ini, disajikan data penjualan tiga produk baru dalam bentuk matriks yang menunjukkan jumlah penjualan tiap bulan. Dengan menggunakan Algoritma Pemrograman, identifikasi data yang harus dilakukan mencakup tiga hal, yakni menghitung jumlah penjualan setiap produk, menentukan bulan dengan penjualan produk tertinggi dan terendah, serta mencari lonjakan yang signifikan pada setiap produknya.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, terdapat beberapa masalah yang dapat dirumuskan.

1 Bagaimana cara menghitung total penjualan setiap produk?
2 Bagaimana cara menentukan bulan dengan penjualan tertinggi dan terendah untuk setiap produk?
3 Bagaiamana cara mengidentifikasi produk yang memiliki lonjakan penjualan yang tidak biasa dibandingkan dengan rata-rata penjualan bulanan?

1.3 Tujuan

Adapun tujuan dibuatnya laporan praktikum ini adalah sebagai berikut.

1. Untuk mengetahui total penjualan setiap produk.
2. Untuk menentukan bulan dengan penjualan tertinggi dan terendah untuk setiap produk.
3. mengidentifikasi produk yang memiliki lonjakan penjualan yang tidak biasa dibandingkan dengan rata-rata penjualan bulanan.

1.4 Manfaat

Manfaat dari laporan ini adalah untuk membantu memahami cara mengolah data penjualan dalam bentuk matriks secara sederhana. Dengan begitu, kita dapat belajar bagaimana menghitung total penjualan, melihat pola penjualan tiap bulannya, dan mengenali lonjakan yang tidak biasa. Selain itu, laporan ini juga dapat menjadi latihan dalam menerapkan algoritma dan pemrograman dalam kasus nyata.

BAB II

PEMBAHASAN

2.1 Menghitung Total Penjualan

```
# Inisialisasi data yang akan diolah
data_bulan = ["Januari", "Februari", "Maret", "April", "Mei", "Juni"]
data_penjualan = [
    [120, 150, 130, 170, 200, 190],
    [80, 100, 90, 110, 130, 120],
    [200, 210, 190, 180, 220, 210]
]
```

Karena setiap kolom dari setiap produknya merepresentasikan jumlah penjualan selama 6 bulan, maka diberikan inisal data bulan yakni `["Januari", "Februari", "Maret", "April", "Mei", "Juni"]` agar lebih mudah dianalisis.

2.1.1 Fungsi Menghitung Total Penjualan Setiap Produk

```
# --- Bagian A: Menghitung total penjualan untuk setiap produk ---
print("\n---Bagian A---\n")
# Fungsi ini menghitung total per produk.
def total_penjualan_per_produk():
    # Menginisialisasi variabel untuk menyimpan total per produk
    retval = []
    # Iterasi melalui setiap baris (produk)
    for baris in data_penjualan:
        #jumlah setiap produk A,B,C akan ditambahkan ke variabel retval
        retval.append(sum(baris))

    # Mengembalikan banyaknya total penjualan untuk setiap produk A,B,C
    return f"Total Penjualan untuk setiap produk : Produk A =
{retval[0]}, Produk B = {retval[1]}, Produk C = {retval[2]}"
```

Penjelasan:

- `retval = [ ]`, menggunakan list kosong untuk menyimpan total per produk
- `for baris in data_penjualan:` akan diisi otomatis dengan daftar penjualan produk. Contoh (A`[120, 150, 130, 170, 200, 190]`).
- `retval.append(sum(baris))` artinya menghitung total penjualan satu produk dengan `(sum(baris))` dan disimpan ke dalam list `retval`, sehingga output `retval` nantinya berisi daftar total penjualan tiap produk, misal `[960, 630, 1210]`.

- `return f"Total Penjualan untuk setiap produk : Produk A = {retval[0]}, Produk B = {retval[1]}, Produk C = {retval[2]}"`
  fungsi akan mengembalikan teks (string) yang menampilkan total penjualan tiap produk dengan mengambil nilai dari list `retval` dengan indeks `0` untuk Produk A, indeks `1` untuk Produk B, dan indeks `2` untuk Produk C.

2.1.2 Fungsi Menghitung Total Penjualan Secara Keseluruhan

```
# Fungsi ini menghitung total penjualan keseluruhan, bukan per produk.
def total_penjualan():
    # Menginisialisasi variabel untuk menyimpan total
    retval = 0
    # Iterasi melalui setiap baris (produk)
    for i in range(len(data_penjualan)):
        # Menambahkan total penjualan per baris ke total keseluruhan
        retval += sum(data_penjualan[i])
    return f"Total Penjualan Keseluruhan : {retval}"
```

Penjelasan:

- `retval = 0` Sebagai penampung angka total dari seluruh produk.
- `for i in range(len(data_penjualan)):` *Looping* sebanyak jumlah baris dalam `data_penjualan`. Misalnya 3 produk, artinya `i = 0, 1, 2.` `data_penjualan[i])` akan mengambil daftar penjualan satu produk.
- `sum(data_penjualan[i])` menghitung jumlah satu produk. Misal produk A = `[120, 150, 130, 170, 200, 190] = 960`.
- `retval += sum(data_penjualan[i])` Hasil penjualan tiap produk tadi ditambahkan ke `retval`. Jadi *loop* ini terus menambahkan total tiap produk hingga semuanya dijumlah.
- `return f"Total Penjualan Keseluruhan : {retval}"` Setelah selesai *looping*, fungsi mengembalikan teks yang menampilkan jumlah total penjualan **semua produk digabung**.
- *Output* fungsi ini: `Total Penjualan Keseluruhan :2800`.

2.2 Menentukan Bulan dengan Penjualan Tertinggi & Terendah

2.2.1 Fungsi Menentukan Bulan dengan Penjualan Tertinggi

```
def maksimum():
    # List untuk menyimpan nilai penjualan tertinggi per produk
    retval = []
    # List untuk menyimpan nama bulan dengan penjualan tertinggi
    bulan = []
    # Iterasi setiap baris data penjualan
    for i in range(len(data_penjualan)):
        # Mengambil data satu baris (satu produk)
        baris = data_penjualan[i]
        # Menemukan nilai maksimum dari baris dan menyimpannya
        retval.append(max(baris))
        # Menemukan nama bulan berdasarkan indeks dari nilai maksimum
        bulan.append(data_bulan[baris.index(retval[i])])
    # Mengembalikan kedua list
    return retval, bulan

# Mencetak nilai dan bulan dengan penjualan tertinggi
retval_max, bulan_max = maksimum()
print(f"penjualan tertinggi A adalah {retval_max[0]} pada bulan
{bulan_max[0]}, penjualan tertinggi B adalah {retval_max[1]} pada bulan
{bulan_max[1]}, penjualan tertinggi C adalah {retval_max[2]} pada bulan
{bulan_max[2]}")
print("---")
```

Penjelasan:

- `retval = []` untuk menyimpan angka penjualan tertinggi dari tiap produk & `bulan = []` untuk menyimpan nama bulan yang memiliki angka tertinggi tersebut.
- *Loop* `for i in range(len(data_penjualan)):` ini berfungsi sesuai dengan jumlah produk. Misalnya terdapat 3 produk maka *loop* berjalan sebanyak 3 kali.
- `baris = data_penjualan[i]` untuk mengambil daftar penjualan untuk satu produk. Misal Produk A = `[120, 150, 130, 170, 200, 190]`
- `retval.append(max(baris))` untuk mencari angka penjualan tertinggi dari produk tersebut. Contoh Produk A -> max = `200`. Kemudian, angka ini yang nantinya masuk list `retval`.

- `bulan.append(data_bulan[baris.index(retval[i])])`

  `baris.index(retval[i])` Untuk mencari posisi (indeks) dari angka tertinggi tadi.

  `data_bulan[baris.index(retval[i])]` Mengambil nama bulan sesuai posisi tersebut.

  Misal `200` ada di indeks ke-4, berarti bulan ke-4 = Mei.

  Nama bulan itu masuk ke list `bulan`.

- `return retval, bulan` adalah fungsi untuk mengembalikan dua list:

  `retval = [200, 130, 220]` merupakan jumlah penjualan dari setiap produk.

  `bulan = [“Mei”, “Mei”, “Mei”]` yang merupakan nama bulan dengan penjualan tertinggi.

- Output:

  `retval_max` = list angka penjualan tertinggi.

  `bulan_max` = list bulan saat penjualan tertinggi.

  Contoh:

```
penjualan tertinggi A adalah 200 pada bulan Mei,
penjualan tertinggi B adalah 130 pada bulan Mei,
penjualan tertinggi C adalah 220 pada bulan Mei.
```

2.2.2 Fungsi Menentukan Bulan dengan Penjualan Terendah

```
def minimum():
    # List untuk menyimpan nilai penjualan terendah per produk
    retval = []
    # List untuk menyimpan nama bulan dengan penjualan terendah
    bulan = []
    # Iterasi setiap baris data penjualan
    for i in range(len(data_penjualan)):
        # Mengambil data satu baris (satu produk)
        baris = data_penjualan[i]
        # Menemukan nilai minimum dari baris dan menyimpannya
        retval.append(min(baris))
        # Menemukan nama bulan berdasarkan indeks dari nilai minimum
        bulan.append(data_bulan[baris.index(retval[i])])
    return retval, bulan
```

```
# Mencetak nilai dan bulan dengan penjualan terendah
retval_min, bulan_min = minimum()
print(f"penjualan terendah A adalah {retval_min[0]} pada bulan
{bulan_min[0]}, penjualan terendah B adalah {retval_min[1]} pada bulan
{bulan_min[1]}, penjualan terendah C adalah {retval_min[2]} pada bulan
{bulan_min[2]}")
print("---")
```

Penjelasan:

- `retval = []` List kosong untuk menyimpan penjuatan terendah tiap produk
  `bulan = []` List kosong untuk menyimpan nama bulan saat nilai terendah terjadi.
- Loop tiap produk `for i in range(len(data_penjualan))`
  `baris = data_penjualan[i]` Untuk mengambil data penjualan 1 produk.
  `min(baris)` Mencari angka paling kecil dari penjualan produk itu, lalu disimpan ke `retval`.
  `baris.index(retval[i])` Mencari posisi angka terendah dalam daftar penjualan produk itu.
  `data_bulan[baris.index(retval[i])]` Ambil nama bulan sesuai posisi angka tadi, lalu dimasukkan ke list `bulan`.
- `return retval, bulan` Hasil akhirnya adalah:
  `retval = [angka minimum produk A, angka minimum produk B, angka minimum produk C]`
  `bulan = [bulan produk A paling rendah, bulan produk B paling rendah, bulan produk C paling rendah].`
- Output:
  penjualan terendah A adalah 120 pada bulan Januari,
  penjualan terendah B adalah 80 pada bulan Januari,
  penjualan terendah C adalah 180 pada bulan April

2.3 Mengidentifikasi Lonjakan Penjualan

```
def lonjakan():
    # List untuk menyimpan rata-rata penjualan per produk
    rata_rata_perbaris = []

    # Menghitung rata-rata penjualan untuk setiap produk
    for i in range(len(data_penjualan)):
        rata_rata = sum(data_penjualan[i]) / len(data_penjualan[i])
        rata_rata_perbaris.append(rata_rata)
```

Fungsi `lonjakan()` ini pada dasarnya melakukan tugas berikut:

1. Mendifinisikan sebuah list kosong `rata_rata_perbaris` untuk menampung hasil.
2. Melakukan perulangan melalui setiap item (yang merepresentasikan data penjualan satu produk) di dalam list `data_penjualan`
3. Untuk setiap item (produk), ia menghitung rata-rata penjualan dengan menjumlahkan semua nilai penjualan dan membaginya dengan jumlah nilai tersebut.
4. Menyimpan hasuil rata-rata ini ke dalam list `rata_rata_perbaris`

```
# List untuk menyimpan informasi lonjakan yang ditemukan
    lonjakan_info = []

    # Iterasi melalui setiap produk dan setiap bulan
    for i in range(len(data_penjualan)):
        for j in range(len(data_penjualan[i])):
            penjualan = data_penjualan[i][j]
```

Kode ini menciptakan sistem perulangan ganda (*nested loop*) yang sangat umum digunakan untuk memproses data berstruktur dua dimensi (seperti *list of list* atau tabel):

1. Iterasi luar `i` : mengganti baris(produk)
2. Iterasi dalam `j` : mengganti kolom (periode waktu/bulan)

```
# Memeriksa apakah penjualan saat ini lebih besar dari ambang batas
lonjakan
            # Rumus: rata-rata * (1 + batasan)
            if penjualan > rata_rata_perbaris[i] * (1 +
batasan_lonjakan):
                # Menentukan nama produk (A, B, C, dst.) berdasarkan
indeks
                nama_produk = f"Produk {chr(65 + i)}"
                bulan_lonjakan = data_bulan[j]

                # Menambahkan informasi lonjakan ke list
                lonjakan_info.append(f"  - {nama_produk} mengalami
lonjakan di bulan {bulan_lonjakan} "
                                     f"dengan penjualan {penjualan}
(Rata-rata: {rata_rata_perbaris[i]:.2f})")
```

`if penjualan > rata_rata_perbaris[i] * (1 + batasan_lonjakan)`

Ini adalah kondisi utama. Jika nilai `penjualan` saat ini lebih besar dari ambang batas, maka penjualan tersebut dianggap lonjakan.

Fungsi inti dari kode ini adalah untuk membandingkan setiap nilai penjualan produk dengan ambang batas dinamis yang dihitung bedasarkan rata-rata produk itu sendiri.

Jika penjualan melebihi ambang batas:

1. Sistem secara otomatis menamai produk (`produk (A, B, C, dst.)`).
2. Sistem mencatat kapan terjadinya lonjakan (`bulan_lonjakan`).
3. Semua detail ini dikumpulkan dalam list `lonjakan_info` agar dapat dilaporkan atau diproses lebih lanjut setelah seluruh data selesai diperiksa.

```
# Mengembalikan hasil. Jika tidak ada lonjakan, tampilkan pesan.
    if not lonjakan_info:
        return "\nTidak ada lonjakan penjualan yang signifikan terdeteksi."
    else:
        return "Lonjakan penjualan yang terdeteksi:\n" +
"\n".join(lonjakan_info)

# Mencetak hasil identifikasi lonjakan
print(lonjakan())
print("---")
```

- Contoh hasil `batasan_lonjakan = 0.2 alias 20%`

```
Lonjakan penjualan yang terdeteksi:

  - Produk A mengalami lonjakan di bulan Mei dengan penjualan 200 (Rata-rata: 160.00)

    Produk B mengalami lonjakan di bulan Mei dengan penjualan 130 (Rata-rata: 105)
```

- Sedangkan untuk produk C tidak mengalami lonjakan yang signifikan.

BAB III

PENUTUP

3.1 Kesimpulan

Dari pembahasan di atas, dapat kita simpulkan beberapa hal berikut.

1. Dengan menggunakan algoritma pemrograman, dapat diketahui total penjualan tiap produk yaitu
    - Produk A = 960,
    - Produk B = 630,
    - Produk C = 1210

2. Dalam penjabaran pemrograman di atas, dapat ditentukan bulan dimana terdapat penjualan tertinggi dan terendah pada setiap produknya.
    - Produk A = Tertinggi 200 pada bulan Mei, terendah 120 pada bulan Januari.
    - Produk B = Tertinggi 130 pada bulan Mei, terendah 80 pada bulan Januari.
    - Produk C = Tertinggi 220 pada bulan Mei, terendah 180 pada bulan April.

3. Dari hasil penghitungan rata-rata penjualan tiap produk dalam periode 6 bulan, produk yang mengalami lonjakan penjualan adalah yang belebihi 20% penjualan dari rata-rata.

    - Produk A rata-rata 160, lonjakan ada di bulan Mei dengan penjualan produk 200.
    - Produk B rata-rata 105, lonjakan ada di bulan Mei dengan penjualan produk 130.
    - Sedangkan Produk C tidak mengalami lonjakan karena data penjualan cenderung stabil tiap bulannya.

3.2 Saran

Dalam pembuatan algoritma sistem pengolahan data penjualan produk ini, diharapkan dapat dikembangkan kembali untuk dipadukan dengan evaluasi strategi pemasaran tiap produk. Produk yang mengalami lonjakan dapat dijadikan fokus promosi, semantara produk dengan penjualan stabil perlu

ditinjau kembali strategi penjualannya. Melalui sistem sederhana ini, pengambilan keputusan dalam suatu perusahaaan bisnis dapat lebih efisien