LAPORAN TUGAS AKHIR
APLIKASI CATATAN KEUANGAN

Mata Kuliah Pemrograman Berorientasi Objek

Disusun Oleh:

1. Dwijananda Galih Prameswara (2213020138)
2. Toni Gunawan (2213020148)
3. David Satria Fahmi (2213020197)

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS NUSANTARA PGRI KEDIRI

TAHUN 2024

KATA PENGANTAR

Dengan rasa syukur dan hormat, kami mempersembahkan makalah ini sebagai bagian dari upaya kami untuk memahami dan mendalami suatu topik yang relevan. Makalah ini disusun dengan tujuan memberikan gambaran umum tentang "APLIKASI CATATAN KEUANGAN", serta mendiskusikan beberapa aspek penting yang terkait.

Kami juga ingin mengucapkan terima kasih kepada semua pihak yang telah memberikan kami semangat dan motivasi dalam pembuatan tugas makalah ini. Kepada kedua orang tua kami yang telah memberikan banyak kontribusi bagi kami, dosen pembimbing kami M. Bahrul Subkhi, M.Kom dan juga kepada teman-teman seperjuangan yang membantu kami dalam berbagai hal. Harapan kami, informasi dan materi yang terdapat dalam makalah ini dapat bermanfaat bagi pembaca.

Tentu saja, makalah ini tidak luput dari kekurangan dan keterbatasan. Oleh karena itu, kami menghargai setiap masukan dan saran yang dapat meningkatkan kualitas makalah ini di masa mendatang.

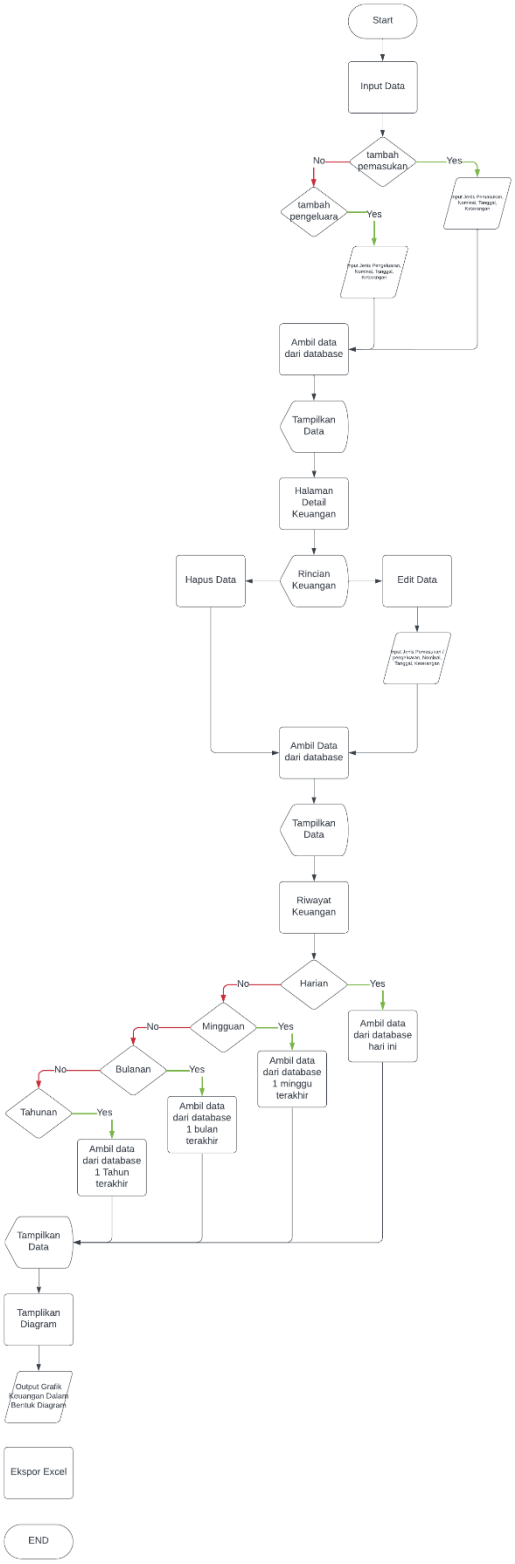
Akhir kata, kami berharap makalah ini dapat memberikan wawasan tambahan dan memberikan kontribusi kecil pada pemahaman umum mengenai "APLIKASI CATATAN KEUANGAN".

Kediri, 02 Januari 2024

Penyusun

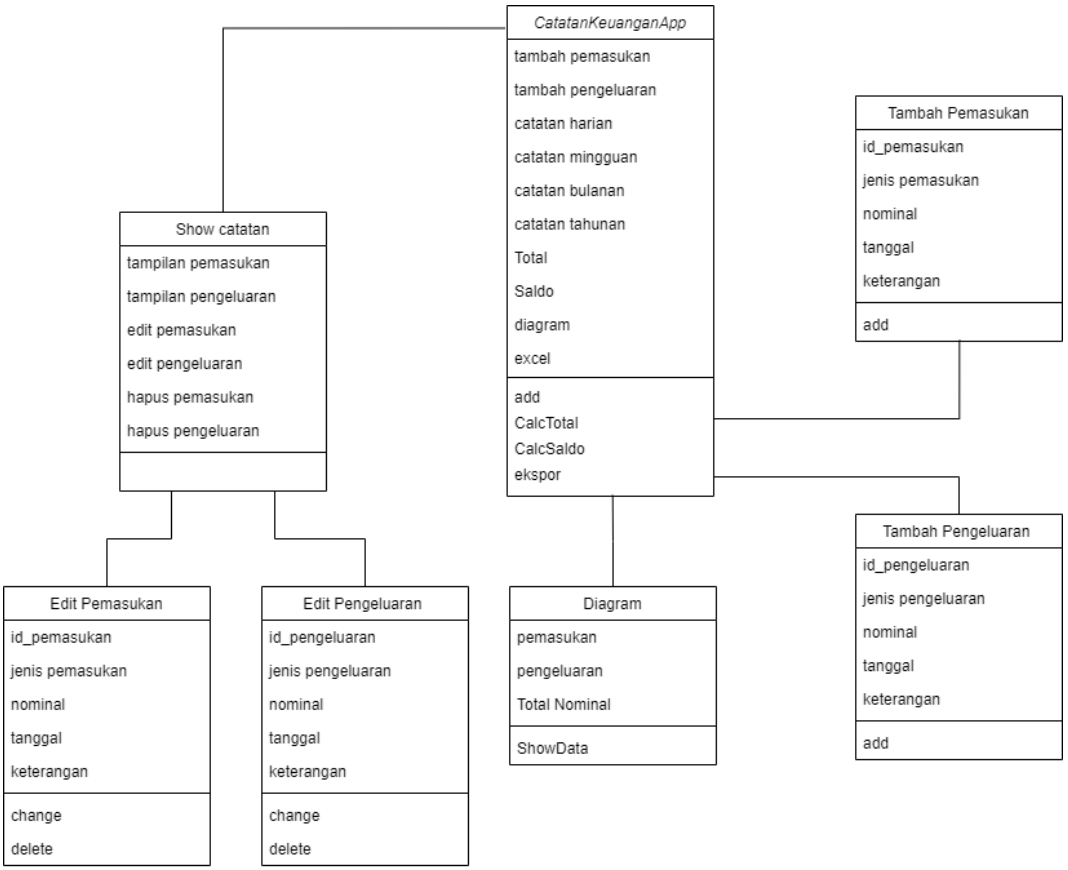
DAFTAR ISI

FLOWCHART SISTEM



CLASS DIAGRAM

Berikut gambar dari Class Diagram dari aplikasi Catatan Keuangan.



Terdapat Class sebagai berikut:

1. CatatanKeuanganApp
2. Tambah Pemasukan
3. Tambah Pengeluaran
4. Diagram
5. Show Catatan
6. Edit Pemasukan
7. Edit pengeluaran

Terdapat Atribut sebagai berikut:

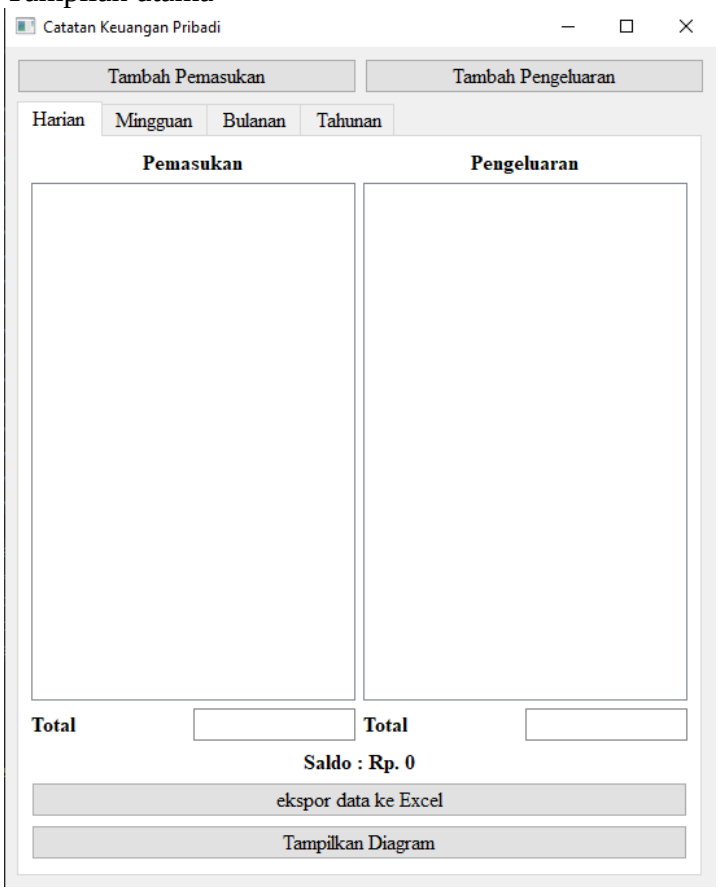
1. CatatanKeuanganApp: tambah pemasukan, tambah pengeluaran, catatan harian, catatan mingguan, catatan bulanan, catatan tahunan, Total, Saldo, diagram, excel
2. Tambah Pemasukan: id_pemasukan, jenis pemasukan, nominal, tanggal, keterangan
3. Tambah Pengeluaran: id_pengeluaran, jenis pengeluaran, nominal, tanggal, keterangan
4. Diagram: pemasukan, pengeluaran, total nominal
5. Show Catatan: tampilan pemasukan, tampilan pengeluaran, edit pemasukan, edit pengeluaran, hapus pemasukan, hapus pengeluaran
6. Edit Pemasukan: id_pemasukan, jenis pemasukan, nominal, tanggal, keterangan
7. Edit pengeluaran: id_pengeluaran, jenis pemasukan, nominal, tanggal, keterangan

Terdapat metode sebagai berikut:

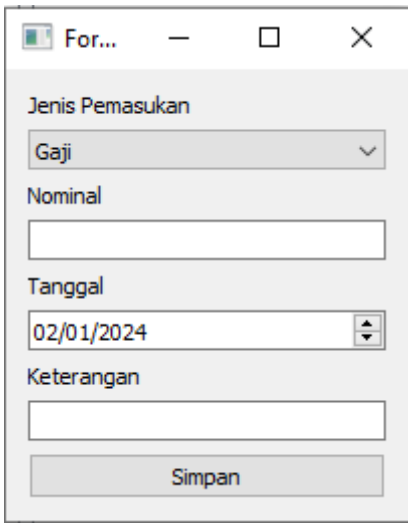
1. CatatanKeuanganApp: add, CalcTotal, CalcSaldo, ekspor
2. Tambah Pemasukan: add
3. Tambah Pengeluaran: add
4. Diagram: ShowData
5. Show Catatan:
6. Edit Pemasukan: change, delete
7. Edit pengeluaran: change, delete

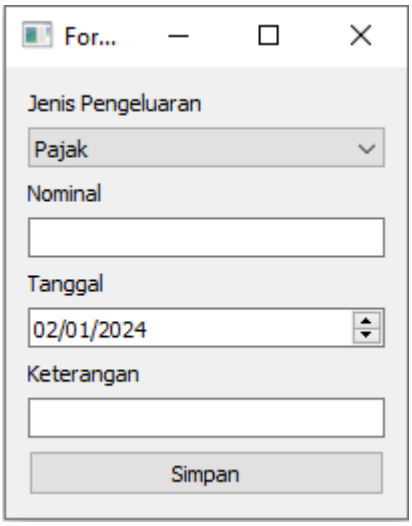
HASIL PROGRAM

- Tampilan utama

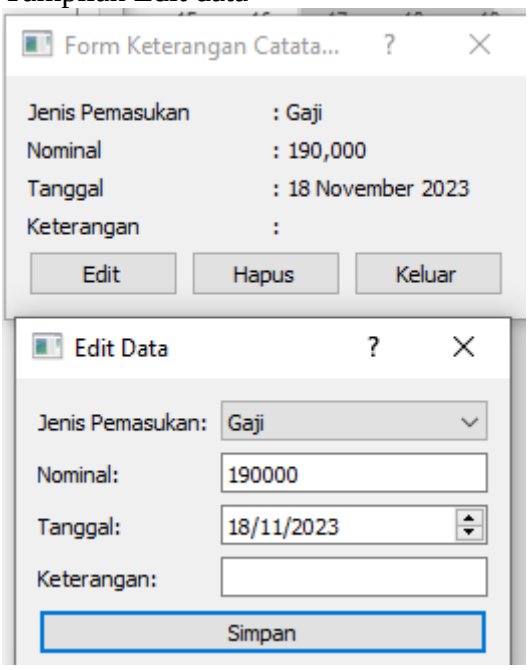


- Tampilan Tambah Pemasukan dan Tambah Pengeluaran

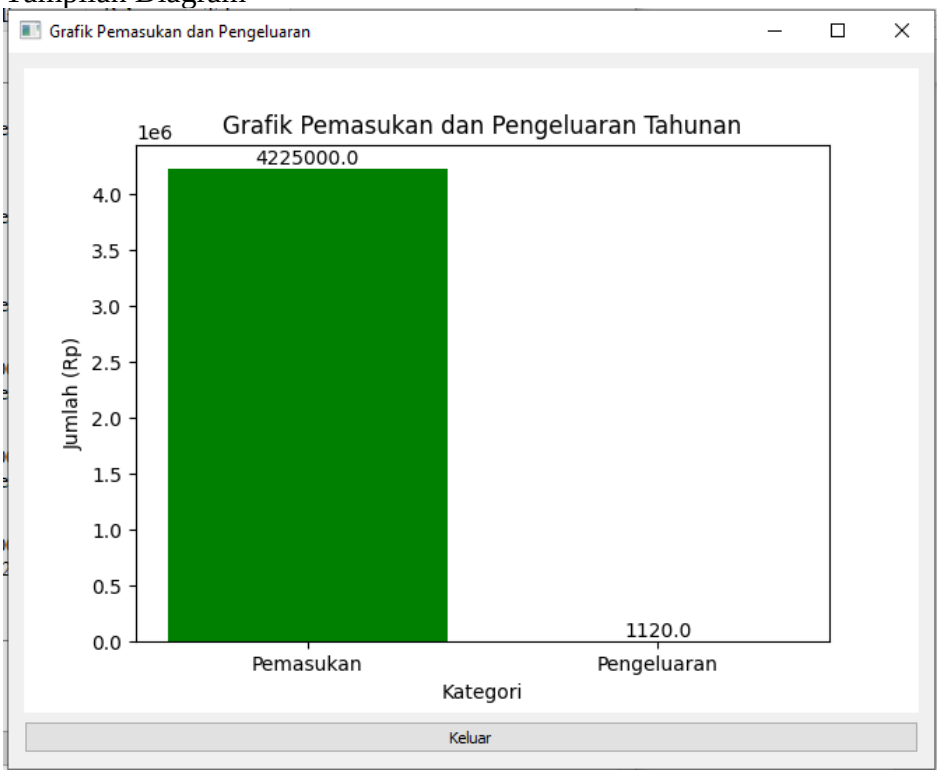




- Tampilan Edit data



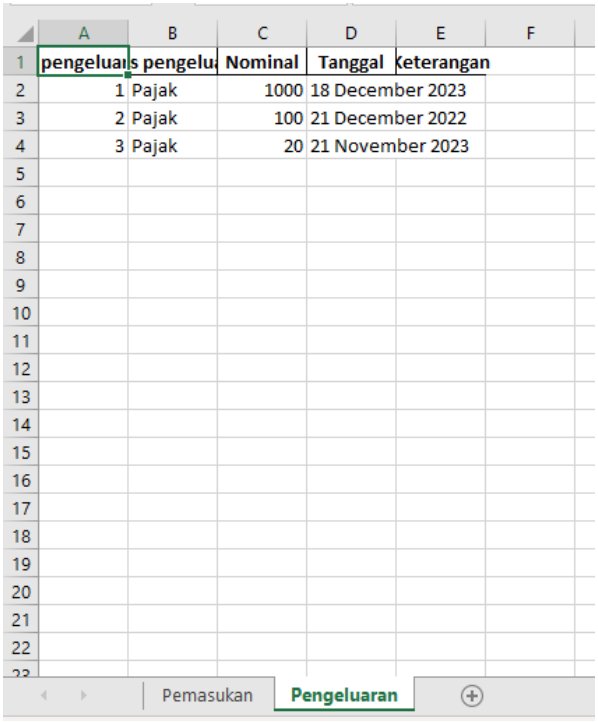
- Tampilan Diagram



- Tampilan saat ekspor ke excel

pemasuka	is pemasu	Nominal	Tanggal	keterangan
1	Gaji	10000	18 December 2023	
2	Gaji	11000	18 December 2023	
3	Gaji	7000	17 December 2023	
4	Gaji	190000	18 November 2023	
5	Gaji	5000	21 December 2022	
6	Gaji	2000	22 December 2023	
7	Investasi	1000000	30 December 2022	
8	Investasi	2000000	31 December 2020	
9	Gaji	1000000	31 October 2022	

Pemasukan | Pengeluaran



	A	B	C	D	E	F
1	pengeluar	s pengelu	Nominal	Tanggal	keterangan	
2	1	Pajak	1000	18 December 2023		
3	2	Pajak	100	21 December 2022		
4	3	Pajak	20	21 November 2023		

Pemasukan | Pengeluaran

PENJELASAN PROGRAM

- Class CatatanKeunanganApp

```
31  class CatatanKeuanganApp(QMainWindow):
32      def __init__(self):
33          super().__init__()
34          self.setWindowTitle("Catatan Keuangan Pribadi")
35          self.setGeometry(200, 200, 500, 540)
36
37          font = QtGui.QFont()
38          font.setFamily("Times New Roman")
39          font.setPointSize(12)
40
41          font1 = QtGui.QFont()
42          font1.setFamily("Times New Roman")
43          font1.setPointSize(12)
44          font1.setBold(True)
45
46          self.font2 = QtGui.QFont()
47          self.font2.setFamily("Times New Roman")
48          self.font2.setPointSize(8)
```

Line 31 untuk membuat class dengan nama catatan keuangan yang merupakan kelas turunan dari class QmainWindow

Line 32-48 untuk memodifikasi judul, tata letak, jenis font pada widget

```
51          button1 = QPushButton("Tambah Pemasukan")
52          button2 = QPushButton("Tambah Pengeluaran")
53          button1.clicked.connect(self.tambahCatatanPemasukan)
54          button2.clicked.connect(self.tambahCatatanPengeluaran)
55          label9 = QLabel("Total")
56          label1 = QLabel("Pemasukan")
57          label1.setAlignment(Qt.AlignCenter)
58          label1.setFont(font1)
59          label2 = QLabel("Pengeluaran")
60          label2.setAlignment(Qt.AlignCenter)
61          label2.setFont(font1)
62          label3 = QLabel("Pemasukan")
63          label3.setAlignment(Qt.AlignCenter)
64          label3.setFont(font1)
65          label4 = QLabel("Pengeluaran")
66          label4.setAlignment(Qt.AlignCenter)
67          label4.setFont(font1)
68          label5 = QLabel("Pemasukan")
69          label5.setAlignment(Qt.AlignCenter)
70          label5.setFont(font1)
71          label6 = QLabel("Pengeluaran")
72          label6.setAlignment(Qt.AlignCenter)
73          label6.setFont(font1)
74          label7 = QLabel("Pemasukan")
75          label7.setAlignment(Qt.AlignCenter)
76          label7.setFont(font1)
77          label8 = QLabel("Pengeluaran")
78          label8.setAlignment(Qt.AlignCenter)
79          label8.setFont(font1)
```

Line 51-54 untuk membuat button Tambah Pemasukan dan Tambah Pengeluaran dan menghubungkannya ke method tambahCatatanPemasukan dan tambahCatatanPengeluaran.

Line 56-79 untuk membuat label pemasukan dan pengeluaran pada tab harian,mingguan, bulanan dan tahunan.

```
79          self.list1 = QListWidget()
80          self.list1.itemDoubleClicked.connect(self.show_catatan)
81          self.list1.setStyleSheet("QListWidget::item { margin: 5px; }")
82          self.list1.setFont(QtGui.QFont("Times New Roman", 11))
83          self.list1.setFixedSize(250, 400)
84          self.list2 = QListWidget()
85          self.list2.itemDoubleClicked.connect(self.show_catatan)
86          self.list2.setStyleSheet("QListWidget::item { margin: 5px; }")
87          self.list2.setFont(QtGui.QFont("Times New Roman", 11))
88          self.list2.setFixedSize(250, 400)
89          self.list3 = QListWidget()
90          self.list3.itemDoubleClicked.connect(self.show_catatan)
91          self.list3.setStyleSheet("QListWidget::item { margin: 5px; }")
92          self.list3.setFont(QtGui.QFont("Times New Roman", 11))
93          self.list3.setFixedSize(250, 400)
94          self.list4 = QListWidget()
95          self.list4.itemDoubleClicked.connect(self.show_catatan)
96          self.list4.setStyleSheet("QListWidget::item { margin: 5px; }")
97          self.list4.setFont(QtGui.QFont("Times New Roman", 11))
98          self.list4.setFixedSize(250, 400)
99          self.list5 = QListWidget()
100         self.list5.itemDoubleClicked.connect(self.show_catatan)
101         self.list5.setStyleSheet("QListWidget::item { margin: 5px; }")
102         self.list5.setFont(QtGui.QFont("Times New Roman", 11))
103         self.list5.setFixedSize(250, 400)
104         self.list6 = QListWidget()
105         self.list6.itemDoubleClicked.connect(self.show_catatan)
106         self.list6.setStyleSheet("QListWidget::item { margin: 5px; }")
107         self.list6.setFont(QtGui.QFont("Times New Roman", 11))
108         self.list6.setFixedSize(250, 400)
109         self.list7 = QListWidget()
110         self.list7.itemDoubleClicked.connect(self.show_catatan)
111         self.list7.setStyleSheet("QListWidget::item { margin: 5px; }")
112         self.list7.setFont(QtGui.QFont("Times New Roman", 11))
113         self.list7.setFixedSize(250, 400)
114         self.list8 = QListWidget()
115         self.list8.itemDoubleClicked.connect(self.show_catatan)
116         self.list8.setStyleSheet("QListWidget::item { margin: 5px; }")
117         self.list8.setFont(QtGui.QFont("Times New Roman", 11))
118         self.list8.setFixedSize(250, 400)
```

Line 79-118 Membuat ListWidget untuk menampilkan daftar pemasukan dan pengeluaran.

```
119         label9 = QLabel("Total")
120         label9.setFont(font1)
121         label10 = QLabel("Total")
122         label10.setFont(font1)
123         label11 = QLabel("Total")
124         label11.setFont(font1)
125         label12 = QLabel("Total")
126         label12.setFont(font1)
127         label13 = QLabel("Total")
128         label13.setFont(font1)
129         label14 = QLabel("Total")
130         label14.setFont(font1)
131         label15 = QLabel("Total")
132         label15.setFont(font1)
133         label16 = QLabel("Total")
134         label16.setFont(font1)
135         self.label17 = QLabel()
136         self.label17.setAlignment(Qt.AlignCenter)
137         self.label17.setFont(font1)
138         self.label18 = QLabel()
139         self.label18.setAlignment(Qt.AlignCenter)
140         self.label18.setFont(font1)
141         self.label19 = QLabel()
142         self.label19.setAlignment(Qt.AlignCenter)
143         self.label19.setFont(font1)
144         self.label20 = QLabel()
145         self.label20.setAlignment(Qt.AlignCenter)
146         self.label20.setFont(font1)
```

119-134 untuk membuat label total pemasukan dan pengeluaran.

Line 135-146 untuk membuat label yang nantinya akan dibuat sebagai label saldo.

```
147         self.labeltotal1 = QLabel()
148         self.labeltotal1.setStyleSheet("border: 1px solid gray; padding: 5px;")
149         self.labeltotal2 = QLabel()
150         self.labeltotal2.setStyleSheet("border: 1px solid gray; padding: 5px;")
151         self.labeltotal3 = QLabel()
152         self.labeltotal3.setStyleSheet("border: 1px solid gray; padding: 5px;")
153         self.labeltotal4 = QLabel()
154         self.labeltotal4.setStyleSheet("border: 1px solid gray; padding: 5px;")
155         self.labeltotal5 = QLabel()
156         self.labeltotal5.setStyleSheet("border: 1px solid gray; padding: 5px;")
157         self.labeltotal6 = QLabel()
158         self.labeltotal6.setStyleSheet("border: 1px solid gray; padding: 5px;")
159         self.labeltotal7 = QLabel()
160         self.labeltotal7.setStyleSheet("border: 1px solid gray; padding: 5px;")
161         self.labeltotal8 = QLabel()
162         self.labeltotal8.setStyleSheet("border: 1px solid gray; padding: 5px;")
```

Line 147-162 untuk membuat label yang nantinya akan dibuat untuk mengisi nilai total

```
163         button_ekspor1 = QPushButton("ekspor data ke Excel")
164         button_ekspor1.clicked.connect(self.ekspor_excel1)
165         button_ekspor2 = QPushButton("ekspor data ke Excel")
166         button_ekspor2.clicked.connect(self.ekspor_excel2)
167         button_ekspor3 = QPushButton("ekspor data ke Excel")
168         button_ekspor3.clicked.connect(self.ekspor_excel3)
169         button_ekspor4 = QPushButton("ekspor data ke Excel")
170         button_ekspor4.clicked.connect(self.ekspor_excel4)
171         button_diagram1 = QPushButton("Tampilkan Diagram")
172         button_diagram1.clicked.connect(self.show_diagram1)
173         button_diagram2 = QPushButton("Tampilkan Diagram")
174         button_diagram2.clicked.connect(self.show_diagram2)
175         button_diagram3 = QPushButton("Tampilkan Diagram")
176         button_diagram3.clicked.connect(self.show_diagram3)
177         button_diagram4 = QPushButton("Tampilkan Diagram")
178         button_diagram4.clicked.connect(self.show_diagram4)
```

Line 163-170 untuk membuat button ekspor ke excel

Line 171-178 untuk membuat button tampilkan diagram.

```
180         layoutH1 = QHBoxLayout()
181         layoutH1.addWidget(label9)
182         layoutH1.addWidget(self.labeltotal1)
183         layoutH1.setSpacing(0)
184 
185         layoutH2 = QHBoxLayout()
186         layoutH2.addWidget(label10)
187         layoutH2.addWidget(self.labeltotal2)
188         layoutH2.setSpacing(0)
189 
190         layoutH3 = QHBoxLayout()
191         layoutH3.addWidget(label11)
192         layoutH3.addWidget(self.labeltotal3)
193         layoutH3.setSpacing(0)
194 
195         layoutH4 = QHBoxLayout()
196         layoutH4.addWidget(label12)
197         layoutH4.addWidget(self.labeltotal4)
198         layoutH4.setSpacing(0)
199 
200         layoutH5 = QHBoxLayout()
201         layoutH5.addWidget(label13)
202         layoutH5.addWidget(self.labeltotal5)
203         layoutH5.setSpacing(0)
204 
205         layoutH6 = QHBoxLayout()
206         layoutH6.addWidget(label14)
207         layoutH6.addWidget(self.labeltotal6)
208         layoutH6.setSpacing(0)
209 
210         layoutH7 = QHBoxLayout()
211         layoutH7.addWidget(label15)
212         layoutH7.addWidget(self.labeltotal7)
213         layoutH7.setSpacing(0)
214 
215         layoutH8 = QHBoxLayout()
216         layoutH8.addWidget(label16)
217         layoutH8.addWidget(self.labeltotal8)
218         layoutH8.setSpacing(0)
```

Line 180-218 untuk membuat layout horizontal pada tab harian,mingguan,bulanan,tahunan yang isinya label total dan nilai total.

```
220         layouttab1 = QGridLayout()
221         layouttab1.addWidget(label1, 0, 0)
222         layouttab1.addWidget(label2, 0, 1)
223         layouttab1.addWidget(self.list1, 1, 0)
224         layouttab1.addWidget(self.list2, 1, 1)
225         layouttab1.addLayout(layoutH1, 2, 0)
226         layouttab1.addLayout(layoutH2, 2, 1)
227         layouttab1.addWidget(self.label17, 3, 0, 1, 2)
228         layouttab1.addWidget(button_ekspor1, 4, 0, 1, 2)
229         layouttab1.addWidget(button_diagram1, 5, 0, 1, 2)
230 
231         layouttab2 = QGridLayout()
232         layouttab2.addWidget(label3, 0, 0)
233         layouttab2.addWidget(label4, 0, 1)
234         layouttab2.addWidget(self.list3, 1, 0)
235         layouttab2.addWidget(self.list4, 1, 1)
236         layouttab2.addLayout(layoutH3, 2, 0)
237         layouttab2.addLayout(layoutH4, 2, 1)
238         layouttab2.addWidget(self.label18, 3, 0, 1, 2)
239         layouttab2.addWidget(button_ekspor2, 4, 0, 1, 2)
240         layouttab2.addWidget(button_diagram2, 5, 0, 1, 2)
241 
242         layouttab3 = QGridLayout()
243         layouttab3.addWidget(label5, 0, 0)
244         layouttab3.addWidget(label6, 0, 1)
245         layouttab3.addWidget(self.list5, 1, 0)
246         layouttab3.addWidget(self.list6, 1, 1)
247         layouttab3.addLayout(layoutH5, 2, 0)
248         layouttab3.addLayout(layoutH6, 2, 1)
249         layouttab3.addWidget(self.label19, 3, 0, 1, 2)
250         layouttab3.addWidget(button_ekspor3, 4, 0, 1, 2)
251         layouttab3.addWidget(button_diagram3, 5, 0, 1, 2)
252 
253         layouttab4 = QGridLayout()
254         layouttab4.addWidget(label7, 0, 0)
255         layouttab4.addWidget(label8, 0, 1)
256         layouttab4.addWidget(self.list7, 1, 0)
257         layouttab4.addWidget(self.list8, 1, 1)
258         layouttab4.addLayout(layoutH7, 2, 0)
259         layouttab4.addLayout(layoutH8, 2, 1)
260         layouttab4.addWidget(self.label20, 3, 0, 1, 2)
261         layouttab4.addWidget(button_ekspor4, 4, 0, 1, 2)
262         layouttab4.addWidget(button_diagram4, 5, 0, 1, 2)
```

Line 220-262 untuk membuat GridLayout dan menyesuaikan posisi pada setiap widget pada tab harian,mingguan,bulanan dan tahunan.

```
264         tab1 = QWidget()
265         tab1.setObjectName("Harian")
266         tab1.setFont(font)
267         tab1.setLayout(layouttab1)
268         tab2 = QWidget()
269         tab2.setObjectName("Mingguan")
270         tab2.setFont(font)
271         tab2.setLayout(layouttab2)
272         tab3 = QWidget()
273         tab3.setObjectName("Bulanan")
274         tab3.setFont(font)
275         tab3.setLayout(layouttab3)
276         tab4 = QWidget()
277         tab4.setObjectName("Tahunan")
278         tab4.setFont(font)
279         tab4.setLayout(layouttab4)
280
281         tabWidget = QTabWidget()
282         tabWidget.addTab(tab1, "Harian")
283         tabWidget.addTab(tab2, "Mingguan")
284         tabWidget.addTab(tab3, "Bulanan")
285         tabWidget.addTab(tab4, "Tahunan")
```

Line 264-279 untuk mengatur nama dan font pada widget.

Line 281-285 untuk menambahkan tab ke objek QtabWidget.

```
288         layout1 = QGridLayout()
289         layout1.addWidget(button1, 0, 0)
290         layout1.addWidget(button2, 0, 1)
291
292         mainLayout = QVBoxLayout()
293         mainLayout.addLayout(layout1)
294         mainLayout.addWidget(tabWidget)
295
296         widget = QWidget()
297         widget.setLayout(mainLayout)
298         widget.setFont(font)
299         self.setCentralWidget(widget)
300         self.adjustSize()
301
302         self.nilai_angka1 = 0
303         self.nilai_angka2 = 0
304         self.nilai_angka3 = 0
305         self.nilai_angka4 = 0
306         self.nilai_angka5 = 0
307         self.nilai_angka6 = 0
308         self.nilai_angka7 = 0
309         self.nilai_angka8 = 0
```

Line 288-290 untuk mengatur grid layout pada button tambah pemasukan dan tambah pengeluaran.

Line 191-194 untuk mengatur layout utama yang isinya layout1 diatas dan tabWidget.

Line 296-300 untuk mengatur mainLayout,font,letak dan menyesuaikan ukuran widget.

Line 302-309 untuk mengatur nilai default menjadi 0.

```
311         self.mydb = mc.connect(
312                 host="localhost",
313                 user="root",
314                 password="",
315                 database="catatankeuangan"
316             )
317 
318         self.load_catatan1()
319         self.load_catatan2()
320         self.load_catatan3()
321         self.load_catatan4()
322         self.load_catatan5()
323         self.load_catatan6()
324         self.load_catatan7()
325         self.load_catatan8()
326         self.total1()
327         self.total2()
328         self.total3()
329         self.total4()
330         self.total5()
331         self.total6()
332         self.total7()
333         self.total8()
334         self.saldo1()
335         self.saldo2()
336         self.saldo3()
337         self.saldo4()
```

Line 311-316 untuk menyambungkan ke database yang ada.

Line 318-337 untuk memanggil fungsi loadcatatan,total dan saldo.

```
339     def load_catatan1(self):
340         self.list1.clear()
341         try:
342             self.mydb._open_connection()
343             mycursor = self.mydb.cursor()
344 
345             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
346 
347             sql = "SELECT id_pemasukan, nominal, tanggal FROM pemasukan WHERE tanggal = %s"
348             val = (tanggal_sekarang,)
349 
350             mycursor.execute(sql, val)
351             result1 = mycursor.fetchall()
352 
353             for id_pemasukan, nominal, tanggal in result1:
354                 tanggal_baru = tanggal.strftime("%d %B %Y")
355                 format_nominal = f"{nominal:,}"
356                 item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pemasukan}")
357                 self.list1.addItem(item)
358 
359             mycursor.close()
360             self.mydb.close()
361 
362         except mc.Error as e:
363             print("gagal menampilkan data:", e)
```

```
365     def load_catatan2(self):
366         self.list2.clear()
367         try:
368             self.mydb._open_connection()
369             mycursor = self.mydb.cursor()
370 
371             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
372 
373             sql = "SELECT id_pengeluaran, nominal, tanggal FROM pengeluaran WHERE tanggal = %s"
374             val = (tanggal_sekarang,)
375 
376             mycursor.execute(sql, val)
377             result = mycursor.fetchall()
378 
379             for id_pengeluaran, nominal, tanggal in result:
380                 tanggal_baru = tanggal.strftime("%d %B %Y")
381                 format_nominal = f"{nominal:,}"
382                 item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pengeluaran}")
383                 self.list2.addItem(item)
384 
385             mycursor.close()
386             self.mydb.close()
387 
388         except mc.Error as e:
389             print("gagal menampilkan data:", e)
390 
391     def load_catatan3(self):
392         self.list3.clear()
393         try:
394             self.mydb._open_connection()
395             mycursor = self.mydb.cursor()
396 
397             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
398             tanggal_7hari = timedelta(days=6)
399 
400             tanggal_mingguan = (datetime.now() - tanggal_7hari).date().strftime("%Y-%m-%d")
401 
402             sql = "SELECT id_pemasukan, nominal, tanggal FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
403             val = (tanggal_mingguan, tanggal_sekarang)
404 
405             mycursor.execute(sql, val)
406             result = mycursor.fetchall()
407 
408             for id_pemasukan, nominal, tanggal in result:
409                 tanggal_baru = tanggal.strftime("%d %B %Y")
410                 format_nominal = f"{nominal:,}"
411                 item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pemasukan}")
412                 self.list3.addItem(item)
413 
414             mycursor.close()
415             self.mydb.close()
416 
417         except mc.Error as e:
418             print("gagal menampilkan data:", e)
```

```python
    def load_catatan4(self):
        self.list4.clear()
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_7hari = timedelta(days=6)

            tanggal_mingguan = (datetime.now() - tanggal_7hari).date().strftime("%Y-%m-%d")

            sql = "SELECT id_pengeluaran, nominal, tanggal FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_mingguan, tanggal_sekarang)

            mycursor.execute(sql, val)
            result = mycursor.fetchall()
            for id_pengeluaran, nominal, tanggal in result:
                tanggal_baru = tanggal.strftime("%d %B %Y")
                format_nominal = f"{nominal:,}"
                item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pengeluaran}")
                self.list4.addItem(item)

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menampilkan data:", e)
    def load_catatan5(self):
        self.list5.clear()
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_30hari = timedelta(days=29)

            tanggal_bulanan = (datetime.now() - tanggal_30hari).date().strftime("%Y-%m-%d")

            sql = "SELECT id_pemasukan, nominal, tanggal FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_bulanan, tanggal_sekarang)

            mycursor.execute(sql, val)
            result = mycursor.fetchall()

            for id_pemasukan, nominal, tanggal in result:
                tanggal_baru = tanggal.strftime("%d %B %Y")
                format_nominal = f"{nominal:,}"
                item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pemasukan}")
                self.list5.addItem(item)

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menampilkan data:", e)
```

```python
    def load_catatan6(self):
        self.list6.clear()
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_30hari = timedelta(days=29)

            tanggal_bulanan = (datetime.now() - tanggal_30hari).date().strftime("%Y-%m-%d")

            sql = "SELECT id_pengeluaran, nominal, tanggal FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_bulanan, tanggal_sekarang)

            mycursor.execute(sql, val)
            result = mycursor.fetchall()

            for id_pengeluaran, nominal, tanggal in result:
                tanggal_baru = tanggal.strftime("%d %B %Y")
                format_nominal = f"{nominal:,}"
                item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pengeluaran}")
                self.list6.addItem(item)

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menampilkan data:", e)
    def load_catatan7(self):
        self.list7.clear()
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_awal = "0000-00-00"
            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_365hari = timedelta(days=364)

            sql = "SELECT id_pemasukan, nominal, tanggal FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_awal, tanggal_sekarang)

            mycursor.execute(sql, val)
            result = mycursor.fetchall()

            for id_pemasukan, nominal, tanggal in result:
                tanggal_baru = tanggal.strftime("%d %B %Y")
                format_nominal = f"{nominal:,}"
                item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pemasukan}")
                self.list7.addItem(item)

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menampilkan data:", e)
```

```python
    def load_catatan8(self):
        self.list8.clear()
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_awal = "0000-00-00"
            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_365hari = timedelta(days=364)

            sql = "SELECT id_pengeluaran, nominal, tanggal FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_awal, tanggal_sekarang)

            mycursor.execute(sql, val)
            result = mycursor.fetchall()

            for id_pengeluaran, nominal, tanggal in result:
                tanggal_baru = tanggal.strftime("%d %B %Y")
                format_nominal = f"{nominal:,}"
                item = QListWidgetItem(f"Rp. {format_nominal}\n{tanggal_baru}\nID: {id_pengeluaran}")
                self.list8.addItem(item)

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menampilkan data:", e)
```

Membuat fungsi load catatan 1-8 yang nantinya dimasukkan pada masing-masing widget tab.

Pertama kita koneksikan ke database lalu kita atur tanggal ke hari ini dan sesuaikan selisih tanggal sesuai tab masing-masing lalu kita tulis query untuk memanggil data dari database diantara tanggal sekarang dan tanggal sesuai tab dan kita eksekusi querynya kemudian perulangan untuk mengambil data dari setiap baris dan kita tampilkan ke widgedlist jika terjadi error akan menampilkan pesan gagal menampilkan data.

```python

    def total1(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")

            sql = "SELECT SUM(nominal) FROM pemasukan WHERE tanggal = %s"
            val = (tanggal_sekarang, )

            mycursor.execute(sql, val)
            result = mycursor.fetchone()

            if not result == (None,):
                self.nilai_angka1 = float(result[0])
                format_jumlah = f"{self.nilai_angka1:,}"
                hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
                self.labeltotal1.setText(f"Rp. {hasil}")
            else:
                self.labeltotal1.setText("")

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menjumlahkan data:", e)

```

```
587     def total2(self):
588         try:
589             self.mydb._open_connection()
590             mycursor = self.mydb.cursor()
591 
592             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
593 
594             sql = "SELECT SUM(nominal) FROM pengeluaran WHERE tanggal = %s"
595             val = (tanggal_sekarang, )
596 
597             mycursor.execute(sql, val)
598             result = mycursor.fetchone()
599 
600             if not result == (None,):
601                 self.nilai_angka2 = float(result[0])
602                 format_jumlah = f"{self.nilai_angka2:,}"
603                 hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
604                 self.labeltotal2.setText(f"Rp. {hasil}")
605             else:
606                 self.labeltotal2.setText("")
607 
608             mycursor.close()
609             self.mydb.close()
610 
611         except mc.Error as e:
612             print("gagal menjumlahkan data:", e)
613     def total3(self):
614         try:
615             self.mydb._open_connection()
616             mycursor = self.mydb.cursor()
617 
618             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
619             tanggal_7hari = timedelta(days=6)
620 
621             tanggal_mingguan = (datetime.now() - tanggal_7hari).date().strftime("%Y-%m-%d")
622 
623             sql = "SELECT SUM(nominal) FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
624             val = (tanggal_mingguan, tanggal_sekarang )
625 
626             mycursor.execute(sql, val)
627             result = mycursor.fetchone()
628 
629             if not result == (None,):
630                 self.nilai_angka3 = float(result[0])
631                 format_jumlah = f"{self.nilai_angka3:,}"
632                 hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
633                 self.labeltotal3.setText(f"Rp. {hasil}")
634             else:
635                 self.labeltotal3.setText("")
636 
637             mycursor.close()
638             self.mydb.close()
639 
640         except mc.Error as e:
641             print("gagal menjumlahkan data:", e)
```

```
643     def total4(self):
644         try:
645             self.mydb._open_connection()
646             mycursor = self.mydb.cursor()
647 
648             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
649             tanggal_7hari = timedelta(days=6)
650 
651             tanggal_mingguan = (datetime.now() - tanggal_7hari).date().strftime("%Y-%m-%d")
652 
653             sql = "SELECT SUM(nominal) FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
654             val = (tanggal_mingguan, tanggal_sekarang )
655 
656             mycursor.execute(sql, val)
657             result = mycursor.fetchone()
658 
659             if not result == (None,):
660                 self.nilai_angka4 = float(result[0])
661                 format_jumlah = f"{self.nilai_angka4:,}"
662                 hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
663                 self.labeltotal4.setText(f"Rp. {hasil}")
664             else:
665                 self.labeltotal4.setText("")
666 
667             mycursor.close()
668             self.mydb.close()
669 
670         except mc.Error as e:
671             print("gagal menjumlahkan data:", e)
672     def total5(self):
673         try:
674             self.mydb._open_connection()
675             mycursor = self.mydb.cursor()
676 
677             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
678             tanggal_30hari = timedelta(days=29)
679 
680             tanggal_bulanan = (datetime.now() - tanggal_30hari).date().strftime("%Y-%m-%d")
681 
682             sql = "SELECT SUM(nominal) FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
683             val = (tanggal_bulanan, tanggal_sekarang )
684 
685             mycursor.execute(sql, val)
686             result = mycursor.fetchone()
687 
688             if not result == (None,):
689                 self.nilai_angka5 = float(result[0])
690                 format_jumlah = f"{self.nilai_angka5:,}"
691                 hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
692                 self.labeltotal5.setText(f"Rp. {hasil}")
693             else:
694                 self.labeltotal5.setText("")
695 
696             mycursor.close()
697             self.mydb.close()
698 
699         except mc.Error as e:
700             print("gagal menjumlahkan data:", e)
701 
```

```python
    def total6(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_30hari = timedelta(days=29)

            tanggal_bulanan = (datetime.now() - tanggal_30hari).date().strftime("%Y-%m-%d")

            sql = "SELECT SUM(nominal) FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_bulanan, tanggal_sekarang )

            mycursor.execute(sql, val)
            result = mycursor.fetchone()

            if not result == (None,):
                self.nilai_angka6 = float(result[0])
                format_jumlah = f"{self.nilai_angka6:,}"
                hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
                self.labeltotal6.setText(f"Rp. {hasil}")
            else:
                self.labeltotal6.setText("")

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menjumlahkan data:", e)

    def total7(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_awal = "0000-00-00"
            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_365hari = timedelta(days=364)

            sql = "SELECT SUM(nominal) FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_awal, tanggal_sekarang )

            mycursor.execute(sql, val)
            result = mycursor.fetchone()

            if not result == (None,):
                self.nilai_angka7 = float(result[0])
                format_jumlah = f"{self.nilai_angka7:,}"
                hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
                self.labeltotal7.setText(f"Rp. {hasil}")
            else:
                self.labeltotal7.setText("")

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menjumlahkan data:", e)
```

```python
    def total8(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_awal = "0000-00-00"
            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_365hari = timedelta(days=364)

            sql = "SELECT SUM(nominal) FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val = (tanggal_awal, tanggal_sekarang )

            mycursor.execute(sql, val)
            result = mycursor.fetchone()

            if not result == (None,):
                self.nilai_angka8 = float(result[0])
                format_jumlah = f"{self.nilai_angka8:,}"
                hasil = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
                self.labeltotal8.setText(f"Rp. {hasil}")
            else:
                self.labeltotal8.setText("")

            mycursor.close()
            self.mydb.close()

        except mc.Error as e:
            print("gagal menjumlahkan data:", e)
```

Membuat fungsi untuk menghitung total jumlah pengeluaran dan pemasukan

pertama kita koneksikan kedatabase lalu kita set tanggal sesuai tab masing-masing,kemudian tulis query untuk mengambil data dari database lalu kita jalankan, jika terdapat nilai maka kita tampilkan nilai pada ke label total,jika nilai tidak ada atau none kosongkan label total.jika terjadi error kita tampilkan output gagal menjumlahkan data.

```python
    def saldo1(self):
        hasil = self.nilai_angka1 - self.nilai_angka2
        format_jumlah = f"{hasil:,}"
        saldo = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
        self.label17.setText(f"Saldo : Rp. {saldo}")

    def saldo2(self):
        hasil = self.nilai_angka3 - self.nilai_angka4
        format_jumlah = f"{hasil:,}"
        saldo = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
        self.label18.setText(f"Saldo : Rp. {saldo}")

    def saldo3(self):
        hasil = self.nilai_angka5 - self.nilai_angka6
        format_jumlah = f"{hasil:,}"
        saldo = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
        self.label19.setText(f"Saldo : Rp. {saldo}")

    def saldo4(self):
        hasil = self.nilai_angka7- self.nilai_angka8
        format_jumlah = f"{hasil:,}"
        saldo = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
        self.label20.setText(f"Saldo : Rp. {saldo}")
```

Membuat fungsi untuk menghitung jumlah saldo. Pertama kurangkan nilai pemasukan dan pengeluarannya, kemudian kita tampilkan hasil dari selisih pemasukan dan pengurangan ke widget label saldo sebelumnya.

```
815     def tambahCatatanPemasukan(self):
816         self.tambah = FormTambahPemasukan()
817         self.tambah.show()
818 
819     def tambahCatatanPengeluaran(self):
820         self.tambah = FormTambahPengeluaran()
821         self.tambah.show()
822 
```

Membuat fungsi untuk menampilkan widget pemasukan dan pengeluaran. Kita buat objek dari class FormTambahPemasukan dan class FormTambahPengeluaran lalu kita panggil class tersebut dengan fungsi show.

```
823     def show_catatan(self, item):
824         itemTerpilih = item.text()
825         baris = itemTerpilih.split(": ")
826         id_item = baris[1]
827         tampilkan = ShowCatatan(id_item)
828         tampilkan.exec_()
829 
```

Membuat fungsi show catatan untuk menampilkan detail dari pemasukan dan pengeluaran.

```
830     def show_diagram1(self):
831         sizes = [self.nilai_angka1, self.nilai_angka2]
832         labels = ["Pemasukan", "Pengeluaran"]
833         judul = "Grafik Pemasukan dan Pengeluaran Harian"
834         self.diagram = ShowDiagram(judul, sizes, labels)
835         self.diagram.show()
836 
837     def show_diagram2(self):
838         sizes = [self.nilai_angka3, self.nilai_angka4]
839         labels = ["Pemasukan", "Pengeluaran"]
840         judul = "Grafik Pemasukan dan Pengeluaran Mingguan"
841         self.diagram = ShowDiagram(judul, sizes, labels)
842         self.diagram.show()
843 
844     def show_diagram3(self):
845         sizes = [self.nilai_angka5, self.nilai_angka6]
846         labels = ["Pemasukan", "Pengeluaran"]
847         judul = "Grafik Pemasukan dan Pengeluaran Bulanan"
848         self.diagram = ShowDiagram(judul, sizes, labels)
849         self.diagram.show()
850 
851     def show_diagram4(self):
852         sizes = [self.nilai_angka7, self.nilai_angka8]
853         labels = ["Pemasukan", "Pengeluaran"]
854         judul = "Grafik Pemasukan dan Pengeluaran Tahunan"
855         self.diagram = ShowDiagram(judul, sizes, labels)
856         self.diagram.show()
857 
```

Membuat fungsi show diagram untuk membuat diagram dari total nilai data pemasukan dan pengeluaran yang sesuai tab masing masing.

```
858     def ekspor_excel1(self):
859         try:
860             self.mydb._open_connection()
861             mycursor = self.mydb.cursor()
862 
863             tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
864 
865             sql1 = "SELECT * FROM pemasukan WHERE tanggal = %s"
866             val1 = (tanggal_sekarang,)
867 
868             mycursor.execute(sql1, val1)
869             result1 = mycursor.fetchall()
870 
871             id_pemasukan = []
872             jenis_pemasukan = []
873             nominal_pemasukan = []
874             tanggal_pemasukan = []
875             keterangan_pemasukan = []
876 
877             for kolom1, kolom2, kolom3, kolom4, kolom5 in result1:
878                 tanggal_baru = kolom4.strftime("%d %B %Y")
879                 id_pemasukan.append(kolom1)
880                 jenis_pemasukan.append(kolom2)
881                 nominal_pemasukan.append(kolom3)
882                 tanggal_pemasukan.append(tanggal_baru)
883                 keterangan_pemasukan.append(kolom5)
884 
885             data1 = {"ID pemasukan" : id_pemasukan,
886                      "Jenis pemasukan" : jenis_pemasukan,
887                      "Nominal" : nominal_pemasukan,
888                      "Tanggal" : tanggal_pemasukan,
889                      "Keterangan" : keterangan_pemasukan}
890 
891             df1 = pd.DataFrame(data1)
892 
893             sql2 = "SELECT * FROM pengeluaran WHERE tanggal = %s"
894             val2 = (tanggal_sekarang,)
895 
896             mycursor.execute(sql2, val2)
897             result2 = mycursor.fetchall()
898 
899             id_pengeluaran = []
900             jenis_pengeluaran = []
901             nominal_pengeluaran = []
902             tanggal_pengeluaran = []
903             keterangan_pengeluaran = []
904 
905             for kolom1, kolom2, kolom3, kolom4, kolom5 in result2:
906                 tanggal_baru = kolom4.strftime("%d %B %Y")
907                 id_pengeluaran.append(kolom1)
908                 jenis_pengeluaran.append(kolom2)
909                 nominal_pengeluaran.append(kolom3)
910                 tanggal_pengeluaran.append(tanggal_baru)
911                 keterangan_pengeluaran.append(kolom5)
```

```
912
913             data2 = {"ID pengeluaran" : id_pengeluaran,
914                      "Jenis pengeluaran" : jenis_pengeluaran,
915                      "Nominal" : nominal_pengeluaran,
916                      "Tanggal" : tanggal_pengeluaran,
917                      "Keterangan" : keterangan_pengeluaran}
918
919             df2 = pd.DataFrame(data2)
920
921             default_file_name = "data_pemasukan_dan_pengeluaran_harian.xlsx"
922
923             file_path = filedialog.asksaveasfilename(
924                 defaultextension=".xlsx",
925                 filetypes=[("Excel Files", "*.xlsx")],
926                 initialfile=default_file_name,
927                 title="Simpan Sebagai"
928             )
929
930             if not file_path:
931                 return
932
933             if os.path.exists(file_path):
934                 base, extension = os.path.splitext(file_path)
935                 count = 1
936                 new_file_path = f"{base} ({count}){extension}"
937                 while os.path.exists(new_file_path):
938                     count += 1
939                     new_file_path = f"{base} ({count}){extension}"
940                 file_path = new_file_path
941
942             with pd.ExcelWriter(file_path, engine='xlsxwriter') as writer:
943                 df1.to_excel(writer, sheet_name='Pemasukan', index=False)
944                 df2.to_excel(writer, sheet_name='Pengeluaran', index=False)
945
946             mycursor.close()
947             self.mydb.close()
948
949             QMessageBox.information(self, 'Info', f"Berhasil ekspor data ke excel dengan nama '{file_path}'")
950
951         except mc.Error as e:
952             print("gagal ekspor data ke excel:", e)
953             QMessageBox.warning(self, 'Peringatan', 'Gagal ekspor data ke excel.')
954
```

```python
    def ekspor_excel2(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_7hari = timedelta(days=6)

            tanggal_mingguan = (datetime.now() - tanggal_7hari).date().strftime("%Y-%m-%d")

            sql1 = "SELECT * FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
            val1 = (tanggal_mingguan, tanggal_sekarang)

            mycursor.execute(sql1, val1)
            result1 = mycursor.fetchall()

            id_pemasukan = []
            jenis_pemasukan = []
            nominal_pemasukan = []
            tanggal_pemasukan = []
            keterangan_pemasukan = []

            for kolom1, kolom2, kolom3, kolom4, kolom5 in result1:
                tanggal_baru = kolom4.strftime("%d %B %Y")
                id_pemasukan.append(kolom1)
                jenis_pemasukan.append(kolom2)
                nominal_pemasukan.append(kolom3)
                tanggal_pemasukan.append(tanggal_baru)
                keterangan_pemasukan.append(kolom5)

            data1 = {"ID pemasukan" : id_
                     "Jenis pemasukan" : jenis_pemasukan,
                     "Nominal" : nominal_pemasukan,
                     "Tanggal" : tanggal_pemasukan,
                     "Keterangan" : keterangan_pemasukan}

            df1 = pd.DataFrame(data1)

            sql2 = "SELECT * FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
            val2 = (tanggal_mingguan, tanggal_sekarang)

            mycursor.execute(sql2, val2)
            result2 = mycursor.fetchall()

            id_pengeluaran = []
            jenis_pengeluaran = []
            nominal_pengeluaran = []
            tanggal_pengeluaran = []
            keterangan_pengeluaran = []

            for kolom1, kolom2, kolom3, kolom4, kolom5 in result2:
                tanggal_baru = kolom4.strftime("%d %B %Y")
                id_pengeluaran.append(kolom1)
                jenis_pengeluaran.append(kolom2)
                nominal_pengeluaran.append(kolom3)
```

(variable) jenis_pemasukan: list

```
                tanggal_pengeluaran.append(tanggal_baru)
                keterangan_pengeluaran.append(kolom5)

            data2 = {"ID pengeluaran" : id_pengeluaran,
                     "Jenis pengeluaran" : jenis_pengeluaran,
                     "Nominal" : nominal_pengeluaran,
                     "Tanggal" : tanggal_pengeluaran,
                     "Keterangan" : keterangan_pengeluaran}

            df2 = pd.DataFrame(data2)

            default_file_name = "data_pemasukan_dan_pengeluaran_mingguan.xlsx"

            file_path = filedialog.asksaveasfilename(
                defaultextension=".xlsx",
                filetypes=[("Excel Files", "*.xlsx")],
                initialfile=default_file_name,
                title="Simpan Sebagai"
            )

            if not file_path:
                return

            if os.path.exists(file_path):
                base, extension = os.path.splitext(file_path)
                count = 1
                new_file_path = f"{base} ({count}){extension}"
                while os.path.exists(new_file_path):
                    count += 1
                    new_file_path = f"{base} ({count}){extension}"
                file_path = new_file_path

            with pd.ExcelWriter(file_path, engine='xlsxwriter') as writer:
                df1.to_excel(writer, sheet_name='Pemasukan', index=False)
                df2.to_excel(writer, sheet_name='Pengeluaran', index=False)

            mycursor.close()
            self.mydb.close()

            QMessageBox.information(self, 'Info', f"Berhasil ekspor data ke excel dengan nama '{file_path}'")

        except mc.Error as e:
            print("gagal ekspor data ke excel:", e)
            QMessageBox.warning(self, 'Peringatan', 'Gagal ekspor data ke excel.')

    def ekspor_excel3(self):
        try:
            self.mydb._open_connection()
            mycursor = self.mydb.cursor()

            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
            tanggal_30hari = timedelta(days=29)

            tanggal_bulanan = (datetime.now() - tanggal_30hari).date().strftime("%Y-%m-%d")

```

```
1065            sql1 = "SELECT * FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
1066            val1 = (tanggal_bulanan, tanggal_sekarang)
1067
1068            mycursor.execute(sql1, val1)
1069            result1 = mycursor.fetchall()
1070
1071            id_pemasukan = []
1072            jenis_pemasukan = []
1073            nominal_pemasukan = []
1074            tanggal_pemasukan = []
1075            keterangan_pemasukan = []
1076
1077            for kolom1, kolom2, kolom3, kolom4, kolom5 in result1:
1078                tanggal_baru = kolom4.strftime("%d %B %Y")
1079                id_pemasukan.append(kolom1)
1080                jenis_pemasukan.append(kolom2)
1081                nominal_pemasukan.append(kolom3)
1082                tanggal_pemasukan.append(tanggal_baru)
1083                keterangan_pemasukan.append(kolom5)
1084
1085            data1 = {"ID pemasukan" : id_pemasukan,
1086                     "Jenis pemasukan" : jenis_pemasukan,
1087                     "Nominal" : nominal_pemasukan,
1088                     "Tanggal" : tanggal_pemasukan,
1089                     "Keterangan" : keterangan_pemasukan}
1090
1091            df1 = pd.DataFrame(data1)
1092
1093            sql2 = "SELECT * FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
1094            val2 = (tanggal_bulanan, tanggal_sekarang)
1095
1096            mycursor.execute(sql2, val2)
1097            result2 = mycursor.fetchall()
1098
1099            id_pengeluaran = []
1100            jenis_pengeluaran = []
1101            nominal_pengeluaran = []
1102            tanggal_pengeluaran = []
1103            keterangan_pengeluaran = []
1104
1105            for kolom1, kolom2, kolom3, kolom4, kolom5 in result2:
1106                tanggal_baru = kolom4.strftime("%d %B %Y")
1107                id_pengeluaran.append(kolom1)
1108                jenis_pengeluaran.append(kolom2)
1109                nominal_pengeluaran.append(kolom3)
1110                tanggal_pengeluaran.append(tanggal_baru)
1111                keterangan_pengeluaran.append(kolom5)
1112
1113            data2 = {"ID pengeluaran" : id_pengeluaran,
1114                     "Jenis pengeluaran" : jenis_pengeluaran,
1115                     "Nominal" : nominal_pengeluaran,
1116                     "Tanggal" : tanggal_pengeluaran,
1117                     "Keterangan" : keterangan_pengeluaran}
1118
1119            df2 = pd.DataFrame(data2)
```

```
1120
1121            default_file_name = "data_pemasukan_dan_pengeluaran_bulanan.xlsx"
1122
1123            file_path = filedialog.asksaveasfilename(
1124                defaultextension=".xlsx",
1125                filetypes=[("Excel Files", "*.xlsx")],
1126                initialfile=default_file_name,
1127                title="Simpan Sebagai"
1128            )
1129
1130            if not file_path:
1131                return
1132
1133            if os.path.exists(file_path):
1134                base, extension = os.path.splitext(file_path)
1135                count = 1
1136                new_file_path = f"{base} ({count}){extension}"
1137                while os.path.exists(new_file_path):
1138                    count += 1
1139                    new_file_path = f"{base} ({count}){extension}"
1140                file_path = new_file_path
1141
1142            with pd.ExcelWriter(file_path, engine='xlsxwriter') as writer:
1143                df1.to_excel(writer, sheet_name='Pemasukan', index=False)
1144                df2.to_excel(writer, sheet_name='Pengeluaran', index=False)
1145
1146            mycursor.close()
1147            self.mydb.close()
1148
1149            QMessageBox.information(self, 'Info', f"Berhasil ekspor data ke excel dengan nama '{file_path}'")
1150
1151        except mc.Error as e:
1152            print("gagal ekspor data ke excel:", e)
1153            QMessageBox.warning(self, 'Peringatan', 'Gagal ekspor data ke excel.')
1154
1155    def ekspor_excel4(self):
1156        try:
1157            self.mydb._open_connection()
1158            mycursor = self.mydb.cursor()
1159
1160            tanggal_awal = "0000-00-00"
1161            tanggal_sekarang = datetime.now().date().strftime("%Y-%m-%d")
1162
1163            sql1 = "SELECT * FROM pemasukan WHERE tanggal BETWEEN %s AND %s"
1164            val1 = (tanggal_awal, tanggal_sekarang)
1165
1166            mycursor.execute(sql1, val1)
1167            result1 = mycursor.fetchall()
1168
1169            id_pemasukan = []
1170            jenis_pemasukan = []
1171            nominal_pemasukan = []
1172            tanggal_pemasukan = []
1173            keterangan_pemasukan = []
```

```
1174
1175            for kolom1, kolom2, kolom3, kolom4, kolom5 in result1:
1176                tanggal_baru = kolom4.strftime("%d %B %Y")
1177                id_pemasukan.append(kolom1)
1178                jenis_pemasukan.append(kolom2)
1179                nominal_pemasukan.append(kolom3)
1180                tanggal_pemasukan.append(tanggal_baru)
1181                keterangan_pemasukan.append(kolom5)
1182
1183            data1 = {"ID pemasukan" : id_pemasukan,
1184                     "Jenis pemasukan" : jenis_pemasukan,
1185                     "Nominal" : nominal_pemasukan,
1186                     "Tanggal" : tanggal_pemasukan,
1187                     "Keterangan" : keterangan_pemasukan}
1188
1189            df1 = pd.DataFrame(data1)
1190
1191            sql2 = "SELECT * FROM pengeluaran WHERE tanggal BETWEEN %s AND %s"
1192            val2 = (tanggal_awal, tanggal_sekarang)
1193
1194            mycursor.execute(sql2, val2)
1195            result2 = mycursor.fetchall()
1196
1197            id_pengeluaran = []
1198            jenis_pengeluaran = []
1199            nominal_pengeluaran = []
1200            tanggal_pengeluaran = []
1201            keterangan_pengeluaran = []
1202
1203            for kolom1, kolom2, kolom3, kolom4, kolom5 in result2:
1204                tanggal_baru = kolom4.strftime("%d %B %Y")
1205                id_pengeluaran.append(kolom1)
1206                jenis_pengeluaran.append(kolom2)
1207                nominal_pengeluaran.append(kolom3)
1208                tanggal_pengeluaran.append(tanggal_baru)
1209                keterangan_pengeluaran.append(kolom5)
1210
1211            data2 = {"ID pengeluaran" : id_pengeluaran,
1212                     "Jenis pengeluaran" : jenis_pengeluaran,
1213                     "Nominal" : nominal_pengeluaran,
1214                     "Tanggal" : tanggal_pengeluaran,
1215                     "Keterangan" : keterangan_pengeluaran}
1216
1217            df2 = pd.DataFrame(data2)
1218
1219            default_file_name = "data_pemasukan_dan_pengeluaran_tahunan.xlsx"
1220
1221            file_path = filedialog.asksaveasfilename(
1222                defaultextension=".xlsx",
1223                filetypes=[("Excel Files", "*.xlsx")],
1224                initialfile=default_file_name,
1225                title="Simpan Sebagai"
1226            )
1227
```

```
1228            if not file_path:
1229                return
1230
1231            if os.path.exists(file_path):
1232                base, extension = os.path.splitext(file_path)
1233                count = 1
1234                new_file_path = f"{base} ({count}){extension}"
1235                while os.path.exists(new_file_path):
1236                    count += 1
1237                    new_file_path = f"{base} ({count}){extension}"
1238                file_path = new_file_path
1239
1240            with pd.ExcelWriter(file_path, engine='xlsxwriter') as writer:
1241                df1.to_excel(writer, sheet_name='Pemasukan', index=False)
1242                df2.to_excel(writer, sheet_name='Pengeluaran', index=False)
1243
1244            mycursor.close()
1245            self.mydb.close()
1246
1247            QMessageBox.information(self, 'Info', f"Berhasil ekspor data ke excel dengan nama '{file_path}'")
1248
1249        except mc.Error as e:
1250            print("gagal ekspor data ke excel:", e)
1251            QMessageBox.warning(self, 'Peringatan', 'Gagal ekspor data ke excel.')
1252
```

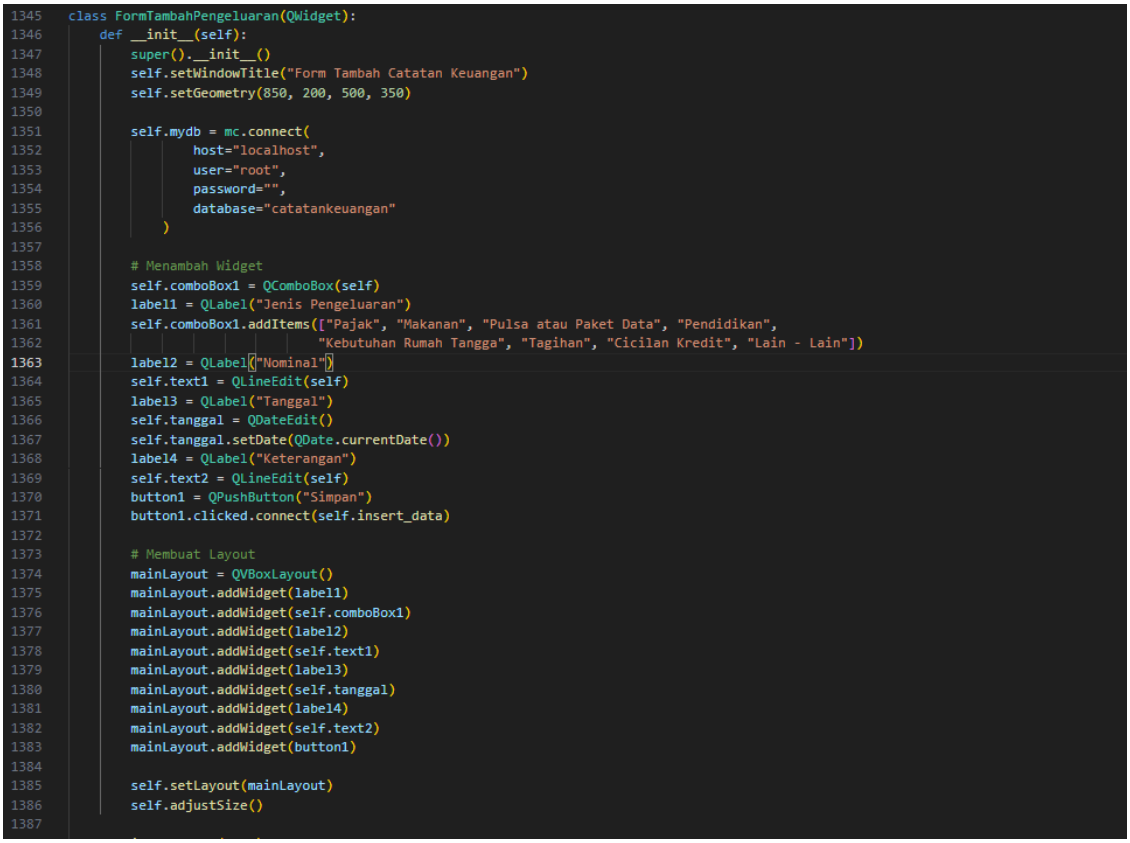
Membuat fungsi untuk ekspor data ke excel. Membuat data yang diperoleh dari hasil yang dimasukkan dalam list terpisah yaitu pemasukan dan pengeluaran. Pada saat ingin menyimpan terdapat nama yang sudah dipersiapkan lalu dapat menyimpan file sesuai folder yang kita inginkan.

```
1253  class FormTambahPemasukan(QWidget):
1254      def __init__(self):
1255          super().__init__()
1256          self.setWindowTitle("Form Tambah Catatan Keuangan")
1257          self.setGeometry(850, 200, 500, 350)
1258
1259          self.mydb = mc.connect(
1260                  host="localhost",
1261                  user="root",
1262                  password="",
1263                  database="catatankeuangan"
1264              )
1265
1266          # Menambah Widget
1267          self.comboBox1 = QComboBox(self)
1268          label1 = QLabel("Jenis Pemasukan")
1269          self.comboBox1.addItems(["Gaji", "Investasi", "Pengembalian Dana",
1270                                   "Penjualan", "Penyewaan", "Tabungan", "Lain - lain"])
1271          label2 = QLabel("Nominal")
1272          self.text1 = QLineEdit(self)
1273          label3 = QLabel("Tanggal")
1274          self.tanggal = QDateEdit()
1275          self.tanggal.setDate(QDate.currentDate())
1276          label4 = QLabel("Keterangan")
1277          self.text2 = QLineEdit(self)
1278          button1 = QPushButton("Simpan")
1279          button1.clicked.connect(self.insert_data)
1280
1281          # Membuat Layout
1282          mainLayout = QVBoxLayout()
1283          mainLayout.addWidget(label1)
1284          mainLayout.addWidget(self.comboBox1)
1285          mainLayout.addWidget(label2)
1286          mainLayout.addWidget(self.text1)
1287          mainLayout.addWidget(label3)
1288          mainLayout.addWidget(self.tanggal)
1289          mainLayout.addWidget(label4)
1290          mainLayout.addWidget(self.text2)
1291          mainLayout.addWidget(button1)
1292
1293          self.setLayout(mainLayout)
1294          self.adjustSize()
1295
```

Membuat class FormTambahPemasukan yang terdapat widget jenis pemasukan, Nominal, Tanggal, Keterangan, dan simpan

```
1296    def insert_data(self):
1297        try:
1298            self.mydb._open_connection()
1299            mycursor = self.mydb.cursor()
1300
1301            jenis = self.comboBox1.currentText()
1302            nominal = self.text1.text()
1303            selected_tanggal = self.tanggal.date()
1304            tanggal = selected_tanggal.toString("yyyy-MM-dd")
1305            text = self.text2.text()
1306
1307            sql = "INSERT INTO pemasukan (jenis_pemasukan, nominal, tanggal, keterangan) VALUES (%s, %s, %s, %s)"
1308            val = (jenis, nominal, tanggal, text)
1309
1310            mycursor.execute(sql, val)
1311            self.mydb.commit()
1312
1313            mycursor.close()
1314            self.mydb.close()
1315
1316            QMessageBox.information(self, 'Info', 'Catatan Pemasukan berhasil ditambahkan.')
1317
1318        except Exception as e:
1319            # print(f"Exception Type: {type(e).__name__}")
1320            print(f"Gagal Menambahkan data pemasukan: {str(e)}")
1321            QMessageBox.warning(self, 'Peringatan', 'Gagal menambahkan catatan pemasukan.')
1322
1323        self.close()
1324        window.load_catatan1()
1325        window.load_catatan2()
1326        window.load_catatan3()
1327        window.load_catatan4()
1328        window.load_catatan5()
1329        window.load_catatan6()
1330        window.load_catatan7()
1331        window.load_catatan8()
1332        window.total1()
1333        window.total2()
1334        window.total3()
1335        window.total4()
1336        window.total5()
1337        window.total6()
1338        window.total7()
1339        window.total8()
1340        window.saldo1()
1341        window.saldo2()
1342        window.saldo3()
1343        window.saldo4()
1344
```

Membuat fungsi insert data untuk menjalankan query dan memasukkan data ke dalam tabel pemasukan. jika kita menambah pemasukan maka akan tampil pesan informasi, dan jika terdapat kesalahan maka akan terjadi pesan gagal.



```
class FormTambahPengeluaran(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Form Tambah Catatan Keuangan")
        self.setGeometry(850, 200, 500, 350)

        self.mydb = mc.connect(
                host="localhost",
                user="root",
                password="",
                database="catatankeuangan"
            )

        # Menambah Widget
        self.comboBox1 = QComboBox(self)
        label1 = QLabel("Jenis Pengeluaran")
        self.comboBox1.addItems(["Pajak", "Makanan", "Pulsa atau Paket Data", "Pendidikan",
                                 "Kebutuhan Rumah Tangga", "Tagihan", "Cicilan Kredit", "Lain - Lain"])
        label2 = QLabel("Nominal")
        self.text1 = QLineEdit(self)
        label3 = QLabel("Tanggal")
        self.tanggal = QDateEdit()
        self.tanggal.setDate(QDate.currentDate())
        label4 = QLabel("Keterangan")
        self.text2 = QLineEdit(self)
        button1 = QPushButton("Simpan")
        button1.clicked.connect(self.insert_data)

        # Membuat Layout
        mainLayout = QVBoxLayout()
        mainLayout.addWidget(label1)
        mainLayout.addWidget(self.comboBox1)
        mainLayout.addWidget(label2)
        mainLayout.addWidget(self.text1)
        mainLayout.addWidget(label3)
        mainLayout.addWidget(self.tanggal)
        mainLayout.addWidget(label4)
        mainLayout.addWidget(self.text2)
        mainLayout.addWidget(button1)

        self.setLayout(mainLayout)
        self.adjustSize()

```

Membuat class FormTambahPengeluaran yang terdapat widget jenis pemasukan, Nominal, Tanggal, Keterangan, dan simpan

```
1388    def insert_data(self):
1389        try:
1390            self.mydb._open_connection()
1391            mycursor = self.mydb.cursor()
1392
1393            jenis = self.comboBox1.currentText()
1394            nominal = self.text1.text()
1395            selected_tanggal = self.tanggal.date()
1396            tanggal = selected_tanggal.toString("yyyy-MM-dd")
1397            text = self.text2.text()
1398
1399            sql = "INSERT INTO pengeluaran (jenis_pengeluaran, nominal, tanggal, keterangan) VALUES (%s, %s, %s, %s)"
1400            val = (jenis, nominal, tanggal, text)
1401
1402            mycursor.execute(sql, val)
1403            self.mydb.commit()
1404
1405            mycursor.close()
1406            self.mydb.close()
1407
1408            QMessageBox.information(self, 'Info', 'Catatan Pengeluaran berhasil ditambahkan.')
1409
1410        except Exception as e:
1411            # print(f"Exception Type: {type(e).__name__}")
1412            print(f"Gagal menambahkan data pengeluaran: {str(e)}")
1413
1414            QMessageBox.warning(self, 'Peringatan', 'Gagal menambahkan catatan pengeluaran.')
1415
1416        self.close()
1417        window.load_catatan1()
1418        window.load_catatan2()
1419        window.load_catatan3()
1420        window.load_catatan4()
1421        window.load_catatan5()
1422        window.load_catatan6()
1423        window.load_catatan7()
1424        window.load_catatan8()
1425        window.total1()
1426        window.total2()
1427        window.total3()
1428        window.total4()
1429        window.total5()
1430        window.total6()
1431        window.total7()
1432        window.total8()
1433        window.saldo1()
1434        window.saldo2()
1435        window.saldo3()
1436        window.saldo4()
1437
```

Membuat fungsi insert data untuk menjalankan query dan memasukkan data ke dalam tabel pengeluaran. jika kita menambah pengeluaran maka akan tampil pesan informasi, dan jika terdapat kesalahan maka akan terjadi pesan gagal.

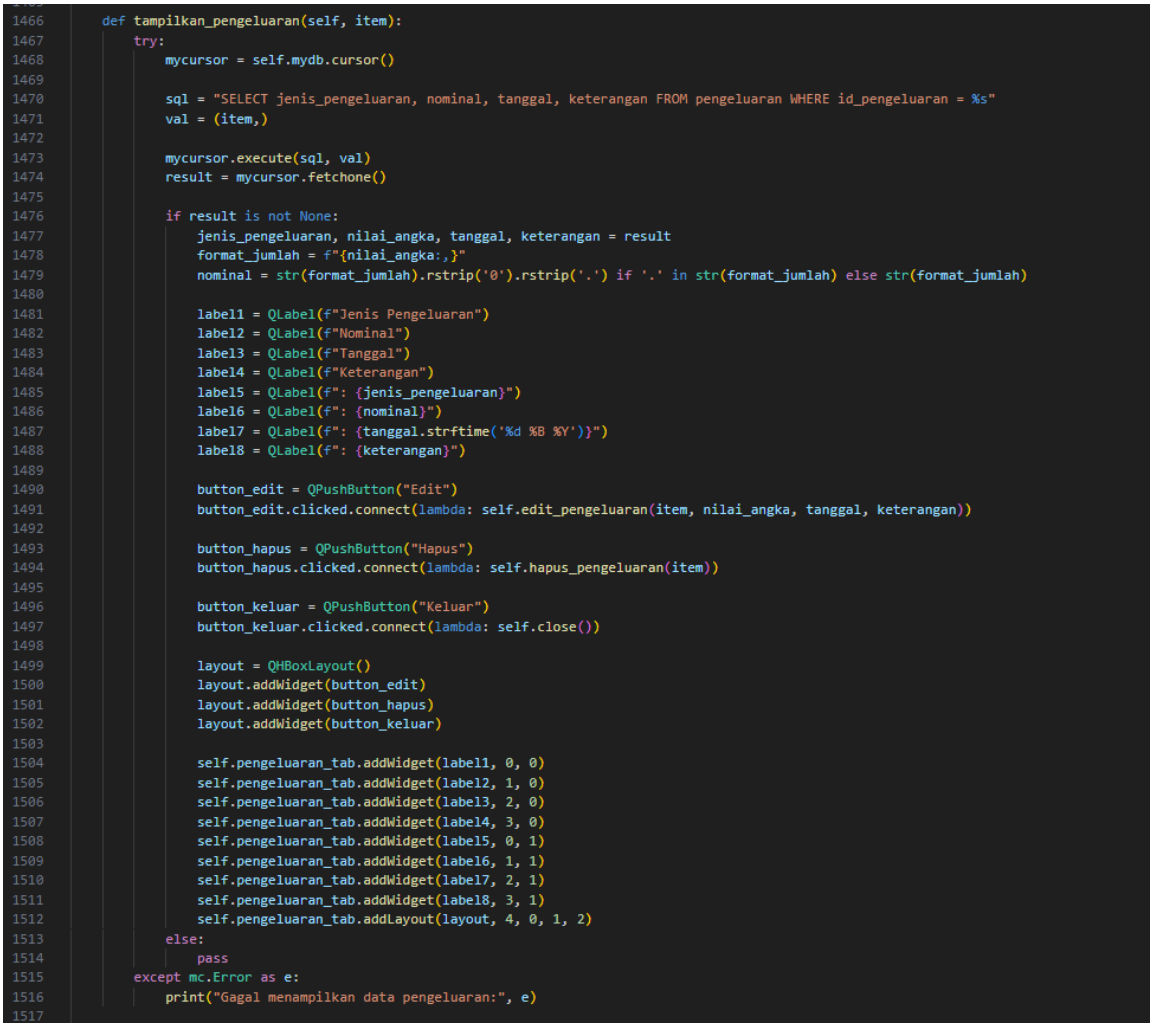
- Class ShowCatatan

```
1438  class ShowCatatan(QDialog):
1439      def __init__(self, item):
1440          super().__init__()
1441
1442          self.setWindowTitle("Form Keterangan Catatan Keuangan")
1443          self.setGeometry(850, 200, 500, 350)
1444
1445          self.mydb = mc.connect(
1446              host="localhost",
1447              user="root",
1448              password="",
1449              database="catatankeuangan"
1450          )
1451
1452          self.pengeluaran_tab = QGridLayout()
1453          self.pemasukan_tab = QGridLayout()
1454          self.item = item
1455          self.setup_ui()
1456
```

Membuat class ShowCatatan yang terdapat tabel tab pengeluaran dan pemasukan

```
1457      def setup_ui(self):
1458          self.tampilkan_pengeluaran(self.item)
1459          self.tampilkan_pemasukan(self.item)
1460
1461          mainLayout = QVBoxLayout(self)
1462          mainLayout.addLayout(self.pengeluaran_tab)
1463          mainLayout.addLayout(self.pemasukan_tab)
1464          self.adjustSize()
1465
```

Membuat metode untuk menampilkan informasi pengeluaran dan pemasukan



```
1466    def tampilkan_pengeluaran(self, item):
1467        try:
1468            mycursor = self.mydb.cursor()
1469
1470            sql = "SELECT jenis_pengeluaran, nominal, tanggal, keterangan FROM pengeluaran WHERE id_pengeluaran = %s"
1471            val = (item,)
1472
1473            mycursor.execute(sql, val)
1474            result = mycursor.fetchone()
1475
1476            if result is not None:
1477                jenis_pengeluaran, nilai_angka, tanggal, keterangan = result
1478                format_jumlah = f"{nilai_angka:,}"
1479                nominal = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)
1480
1481                label1 = QLabel(f"Jenis Pengeluaran")
1482                label2 = QLabel(f"Nominal")
1483                label3 = QLabel(f"Tanggal")
1484                label4 = QLabel(f"Keterangan")
1485                label5 = QLabel(f": {jenis_pengeluaran}")
1486                label6 = QLabel(f": {nominal}")
1487                label7 = QLabel(f": {tanggal.strftime('%d %B %Y')}")
1488                label8 = QLabel(f": {keterangan}")
1489
1490                button_edit = QPushButton("Edit")
1491                button_edit.clicked.connect(lambda: self.edit_pengeluaran(item, nilai_angka, tanggal, keterangan))
1492
1493                button_hapus = QPushButton("Hapus")
1494                button_hapus.clicked.connect(lambda: self.hapus_pengeluaran(item))
1495
1496                button_keluar = QPushButton("Keluar")
1497                button_keluar.clicked.connect(lambda: self.close())
1498
1499                layout = QHBoxLayout()
1500                layout.addWidget(button_edit)
1501                layout.addWidget(button_hapus)
1502                layout.addWidget(button_keluar)
1503
1504                self.pengeluaran_tab.addWidget(label1, 0, 0)
1505                self.pengeluaran_tab.addWidget(label2, 1, 0)
1506                self.pengeluaran_tab.addWidget(label3, 2, 0)
1507                self.pengeluaran_tab.addWidget(label4, 3, 0)
1508                self.pengeluaran_tab.addWidget(label5, 0, 1)
1509                self.pengeluaran_tab.addWidget(label6, 1, 1)
1510                self.pengeluaran_tab.addWidget(label7, 2, 1)
1511                self.pengeluaran_tab.addWidget(label8, 3, 1)
1512                self.pengeluaran_tab.addLayout(layout, 4, 0, 1, 2)
1513            else:
1514                pass
1515        except mc.Error as e:
1516            print("Gagal menampilkan data pengeluaran:", e)
1517
```

Membuat fungsi tampilkan pengeluaran, menjalankan Query untuk menampilkan detail informasi berdasarkan id_pengeluaran. Membuat label untuk menampilkan jenis pengeluaran, Nominal, tanggal, dan keterangan,

```python
    def tampilkan_pemasukan(self, item):
        try:
            mycursor = self.mydb.cursor()

            sql = "SELECT jenis_pemasukan, nominal, tanggal, keterangan FROM pemasukan WHERE id_pemasukan = %s"
            val = (item,)

            mycursor.execute(sql, val)
            result = mycursor.fetchone()

            if result is not None:
                jenis_pemasukan, nilai_angka, tanggal, keterangan = result
                format_jumlah = f"{nilai_angka:,}"
                nominal = str(format_jumlah).rstrip('0').rstrip('.') if '.' in str(format_jumlah) else str(format_jumlah)

                label1 = QLabel(f"Jenis Pemasukan")
                label2 = QLabel(f"Nominal")
                label3 = QLabel(f"Tanggal")
                label4 = QLabel(f"Keterangan")
                label5 = QLabel(f": {jenis_pemasukan}")
                label6 = QLabel(f": {nominal}")
                label7 = QLabel(f": {tanggal.strftime('%d %B %Y')}")
                label8 = QLabel(f": {keterangan}")

                button_edit = QPushButton("Edit")
                button_edit.clicked.connect(lambda: self.edit_pemasukan(item, nilai_angka , tanggal, keterangan))

                button_hapus = QPushButton("Hapus")
                button_hapus.clicked.connect(lambda: self.hapus_pemasukan(item))

                button_keluar = QPushButton("Keluar")
                button_keluar.clicked.connect(lambda: self.close())

                layout = QHBoxLayout()
                layout.addWidget(button_edit)
                layout.addWidget(button_hapus)
                layout.addWidget(button_keluar)

                self.pemasukan_tab.addWidget(label1, 0, 0)
                self.pemasukan_tab.addWidget(label2, 1, 0)
                self.pemasukan_tab.addWidget(label3, 2, 0)
                self.pemasukan_tab.addWidget(label4, 3, 0)
                self.pemasukan_tab.addWidget(label5, 0, 1)
                self.pemasukan_tab.addWidget(label6, 1, 1)
                self.pemasukan_tab.addWidget(label7, 2, 1)
                self.pemasukan_tab.addWidget(label8, 3, 1)
                self.pemasukan_tab.addLayout(layout, 4, 0, 1, 2)
            else:
                pass
        except mc.Error as e:
            print("Gagal menampilkan data pemasukan:", e)

```

Membuat fungsi tampilkan pemasukan, menjalankan Query untuk menampilkan detail informasi berdasarkan id_pemasukan. Membuat label untuk menampilkan jenis pengeluaran, Nominal, tanggal, dan keterangan,

```python
    def edit_pengeluaran(self, item, nominal, tanggal, keterangan):
        dialog = EditPengeluaran(nominal, tanggal, keterangan, parent=self)
        result = dialog.exec_()

        if result == QDialog.Accepted:
            new_jenis_pengeluaran, new_nominal, new_tanggal, new_keterangan = dialog.get_data()

            try:
                mycursor = self.mydb.cursor()

                sql = "UPDATE pengeluaran SET jenis_pengeluaran = %s, nominal = %s, tanggal = %s, keterangan = %s " \
                      "WHERE id_pengeluaran = %s"
                val = (new_jenis_pengeluaran, new_nominal, new_tanggal, new_keterangan, item)

                mycursor.execute(sql, val)
                self.mydb.commit()

                print("Data pengeluaran berhasil diupdate.")
                QMessageBox.information(self, 'Info', 'Data pengeluaran berhasil diupdate.')

                window.load_catatan1()
                window.load_catatan2()
                window.load_catatan3()
                window.load_catatan4()
                window.load_catatan5()
                window.load_catatan6()
                window.load_catatan7()
                window.load_catatan8()
                window.total1()
                window.total2()
                window.total3()
                window.total4()
                window.total5()
                window.total6()
                window.total7()
                window.total8()
                window.saldo1()
                window.saldo2()
                window.saldo3()
                window.saldo4()
                self.close()
                self.tampilkan_pengeluaran(item)

            except mc.Error as e:
                print("Gagal mengupdate data pengeluaran:", e)
                QMessageBox.warning(self, 'Peringatan', 'Gagal mengupdate data pengeluaran.')

```

Membuat fungsi edit pengeluaran, jika pengeluaran ingin kita edit maka akan menjalakan query update pengeluaran ke database. Lalu jika sudah berhasil maka terdapat pesan informasi, dan jika gagal maka terdapat pesan gagal mengupdate.

```
1617    def edit_pemasukan(self, item, nominal, tanggal, keterangan):
1618        dialog = EditPemasukan(nominal, tanggal, keterangan, parent=self)
1619        result = dialog.exec_()
1620
1621        if result == QDialog.Accepted:
1622            new_jenis_pemasukan, new_nominal, new_tanggal, new_keterangan = dialog.get_data()
1623
1624            try:
1625                mycursor = self.mydb.cursor()
1626
1627                sql = "UPDATE pemasukan SET jenis_pemasukan = %s, nominal = %s, tanggal = %s, keterangan = %s " \
1628                      "WHERE id_pemasukan = %s"
1629                val = (new_jenis_pemasukan, new_nominal, new_tanggal, new_keterangan, item)
1630
1631                mycursor.execute(sql, val)
1632                self.mydb.commit()
1633
1634                print("Data pemasukan berhasil diupdate.")
1635                QMessageBox.information(self, 'Info', 'Data pemasukan berhasil diupdate.')
1636
1637                # Bersihkan tampilan
1638                window.load_catatan1()
1639                window.load_catatan2()
1640                window.load_catatan3()
1641                window.load_catatan4()
1642                window.load_catatan5()
1643                window.load_catatan6()
1644                window.load_catatan7()
1645                window.load_catatan8()
1646                window.total1()
1647                window.total2()
1648                window.total3()
1649                window.total4()
1650                window.total5()
1651                window.total6()
1652                window.total7()
1653                window.total8()
1654                window.saldo1()
1655                window.saldo2()
1656                window.saldo3()
1657                window.saldo4()
1658                self.close()
1659                self.tampilkan_pemasukan(item)
1660
1661            except mc.Error as e:
1662                print("Gagal mengupdate data pemasukan:", e)
1663                QMessageBox.warning(self, 'Peringatan', 'Gagal mengupdate data pemasukan.')
1664
```

Membuat fungsi edit pemasukan, jika pemasukan ingin kita edit maka akan menjalakan query update pengeluaran ke database. Lalu jika sudah berhasil maka terdapat pesan informasi, dan jika gagal maka terdapat pesan gagal mengupdate.

```python
    def hapus_pengeluaran(self, item):
        reply = QMessageBox.question(self, 'Hapus Pengeluaran', 'Anda yakin ingin menghapus pengeluaran ini?',
                                     QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

        if reply == QMessageBox.Yes:
            try:
                mycursor = self.mydb.cursor()

                sql = "DELETE FROM pengeluaran WHERE id_pengeluaran = %s"
                val = (item,)

                mycursor.execute(sql, val)
                self.mydb.commit()

                QMessageBox.information(self, 'Info', 'Data pengeluaran berhasil dihapus.')
                window.load_catatan1()
                window.load_catatan2()
                window.load_catatan3()
                window.load_catatan4()
                window.load_catatan5()
                window.load_catatan6()
                window.load_catatan7()
                window.load_catatan8()
                window.total1()
                window.total2()
                window.total3()
                window.total4()
                window.total5()
                window.total6()
                window.total7()
                window.total8()
                window.saldo1()
                window.saldo2()
                window.saldo3()
                window.saldo4()
                self.close()

            except mc.Error as e:
                print("Gagal menghapus data pengeluaran:", e)
                QMessageBox.warning(self, 'Peringatan', 'Gagal menghapus data pengeluaran.')

```

membuat fungsi hapus pengeluran, jika pengeluaran kita hapus maka akan menampilkan pesan konfirmasi. Jika memilih yes maka akan menjalankan query delete pada pengeluaran yang akan kita hapus.

```
1706    def hapus_pemasukan(self, item):
1707        reply = QMessageBox.question(self, 'Hapus Pemasukan', 'Anda yakin ingin menghapus pemasukan ini?',
1708                                     QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
1709
1710        if reply == QMessageBox.Yes:
1711            try:
1712                mycursor = self.mydb.cursor()
1713
1714                sql = "DELETE FROM pemasukan WHERE id_pemasukan = %s"
1715                val = (item,)
1716
1717                mycursor.execute(sql, val)
1718                self.mydb.commit()
1719
1720                QMessageBox.information(self, 'Info', 'Data pemasukan berhasil dihapus.')
1721
1722                window.load_catatan1()
1723                window.load_catatan2()
1724                window.load_catatan3()
1725                window.load_catatan4()
1726                window.load_catatan5()
1727                window.load_catatan6()
1728                window.load_catatan7()
1729                window.load_catatan8()
1730                window.total1()
1731                window.total2()
1732                window.total3()
1733                window.total4()
1734                window.total5()
1735                window.total6()
1736                window.total7()
1737                window.total8()
1738                window.saldo1()
1739                window.saldo2()
1740                window.saldo3()
1741                window.saldo4()
1742                self.close()
1743
1744            except mc.Error as e:
1745                print("Gagal menghapus data pemasukan:", e)
1746                QMessageBox.warning(self, 'Peringatan', 'Gagal menghapus data pemasukan.')
1747
```

membuat fungsi hapus pemasukan, jika pemasukan kita hapus maka akan menampilkan pesan konfirmasi. Jika memilih yes maka akan menjalankan query delete pada pemasukan yang akan kita hapus.

- Class EditPemasukan

```
1748  class EditPemasukan(QDialog):
1749      def __init__(self, nominal, tanggal, keterangan, parent=None):
1750          super().__init__(parent)
1751
1752          self.setWindowTitle("Edit Data")
1753          self.setGeometry(850, 200, 400, 200)
1754
1755          self.jenis_edit = QComboBox(self)
1756          self.jenis_edit.addItems(["Gaji", "Investasi", "Pengembalian Dana",
1757                                    "Penjualan", "Penyewaan", "Tabungan", "Lain - lain"])
1758          self.nominal_edit = QLineEdit(str(nominal))
1759          self.tanggal_edit = QDateEdit(tanggal)
1760          self.keterangan_edit = QLineEdit(keterangan)
1761
1762          form_layout = QFormLayout()
1763          form_layout.addRow("Jenis Pemasukan:", self.jenis_edit)
1764          form_layout.addRow("Nominal:", self.nominal_edit)
1765          form_layout.addRow("Tanggal:", self.tanggal_edit)
1766          form_layout.addRow("Keterangan:", self.keterangan_edit)
1767
1768          save_button = QPushButton("Simpan")
1769          save_button.clicked.connect(self.accept)
1770
1771          layout = QVBoxLayout()
1772          layout.addLayout(form_layout)
1773          layout.addWidget(save_button)
1774
1775          self.setLayout(layout)
1776          self.adjustSize()
1777
1778      def get_data(self):
1779          return (
1780              self.jenis_edit.currentText(),
1781              float(self.nominal_edit.text()),
1782              self.tanggal_edit.date().toPyDate(),
1783              self.keterangan_edit.text()
1784          )
1785
```

Membuat class edit pemasukan, membuat tata letak form, tombol simpan, dan tata letak utama pada edit pemasukan.

- Class EditPengeluaran

```
1786  class EditPengeluaran(QDialog):
1787      def __init__(self, nominal, tanggal, keterangan, parent=None):
1788          super().__init__(parent)
1789
1790          self.setWindowTitle("Edit Data")
1791          self.setGeometry(850, 200, 400, 200)
1792
1793          self.jenis_edit = QComboBox(self)
1794          self.jenis_edit.addItems(["Pajak", "Makanan", "Pulsa atau Paket Data", "Pendidikan",
1795                                    "Kebutuhan Rumah Tangga", "Tagihan", "Cicilan Kredit", "Lain - Lain"])
1796          self.nominal_edit = QLineEdit(str(nominal))
1797          self.tanggal_edit = QDateEdit(tanggal)
1798          self.keterangan_edit = QLineEdit(keterangan)
1799
1800          form_layout = QFormLayout()
1801          form_layout.addRow("Jenis Pemasukan:", self.jenis_edit)
1802          form_layout.addRow("Nominal:", self.nominal_edit)
1803          form_layout.addRow("Tanggal:", self.tanggal_edit)
1804          form_layout.addRow("Keterangan:", self.keterangan_edit)
1805
1806          save_button = QPushButton("Simpan")
1807          save_button.clicked.connect(self.accept)
1808
1809          layout = QVBoxLayout()
1810          layout.addLayout(form_layout)
1811          layout.addWidget(save_button)
1812
1813          self.setLayout(layout)
1814          self.adjustSize()
1815
1816      def get_data(self):
1817          return (
1818              self.jenis_edit.currentText(),
1819              float(self.nominal_edit.text()),
1820              self.tanggal_edit.date().toPyDate(),
1821              self.keterangan_edit.text()
1822          )
1823
```

Membuat class edit pengeluaran, membuat tata letak form, tombol simpan, dan tata letak utama pada edit pengeluaran.

- Class ShowDiagram

```
1824  class ShowDiagram(QWidget):
1825      def __init__(self, judul, sizes, labels):
1826          super().__init__()
1827
1828          self.judul = judul
1829          self.sizes = sizes
1830          self.labels = labels
1831
1832          self.setWindowTitle('Grafik Pemasukan dan Pengeluaran')
1833          self.setGeometry(850, 200, 400, 200)
1834
1835          self.figure, self.axes = plt.subplots()
1836          self.canvas = FigureCanvas(self.figure)
1837
1838          button_keluar = QPushButton("Keluar")
1839          button_keluar.clicked.connect(self.close)
1840
1841          layout = QVBoxLayout()
1842          layout.addWidget(self.canvas)
1843          layout.addWidget(button_keluar)
1844
1845          self.setLayout(layout)
1846          self.adjustSize()
1847
1848          self.drawChart()
1849
```

Membuat class show diagram. Mengatur judul yang grafik pemasukan dan pengeluaran, dan membuat tombol keluar. Lalu menggunakan metode drawChart untuk membuat gambar.

```
1850      def drawChart(self):
1851          self.axes.clear()
1852          bars = self.axes.bar(self.labels, self.sizes, color=['green', 'red'])
1853          self.axes.set_title(self.judul)
1854          self.axes.set_xlabel('Kategori')
1855          self.axes.set_ylabel('Jumlah (Rp)')
1856
1857          for bar in bars:
1858              yval = bar.get_height()
1859              self.axes.text(bar.get_x() + bar.get_width()/2, yval + 10, round(yval, 2), ha='center', va='bottom')
1860
1861          self.canvas.draw()
1862
1863
1864
```

Membuat metode drawChart untuk membuat diagram batang berdasarkan data yang terdapat warna hijau dan merah.

```
1865  if __name__ == "__main__":
1866      app = QApplication(sys.argv)
1867      window = CatatanKeuanganApp()
1868      window.show()
1869      sys.exit(app.exec_())
1870
```

Membuat objek utama yaitu CatatanKeuanganApp, lalu menampilkan jendela utama.

DAFTAR PUSTAKA

Parwiz. (2020, Okotober 11). *PyQt5 Tutorial – How to Insert Data in MySQL Database*. Retrieved from CodeLoop: https://codeloop.org/pyqt5-tutorial-how-to-insert-data-in-mysql-database/?ref=morioh.com&utm_source=morioh.com

PyQt Documentation v5.15.7. (n.d.). Retrieved from riverbankcomputing: https://www.riverbankcomputing.com/static/Docs/PyQt5/