Expense.py

```python
# All Modules
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
import mysql.connector
from tkinter import font
import random

# Database Connection
def connect_to_db():
    return mysql.connector.connect(
        host="localhost",
        user="Swagatam",
        password="Swagatam@1509",
        database="ExpenseTracker"
    )

# Add Expense to Database
def add_expense():
    date = entry_date.get()
    category = combo_category.get()
    amount = entry_amount.get()
    description = entry_description.get()
    while 1:
        idx=random.randint(1,100)
        if idx not in l:
            l.add(idx)
            break

    if not (date and category and amount):
        messagebox.showerror("Input Error", "All fields are required except description!")
        return

    if not entry_description.get():
        description="No Description Added !"

    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        query = "INSERT INTO expenses (id, date, category, amount, description) VALUES (%s, %s,
%s, %s, %s)"
        cursor.execute(query, (idx, date, category, amount, description))
        conn.commit()
        conn.close()
        messagebox.showinfo("Success", "Expense added successfully!")
        clear_fields()
        display_expenses()
    except Exception as e:
        messagebox.showerror("Database Error", str(e))

# Display Expenses with Optional Filters
def display_expenses():
    for row in tree.get_children():
        tree.delete(row)

    query = "SELECT * FROM expenses WHERE 1=1"
    params = []

    # Apply filters
    if filter_start_date.get():
        query += " AND date >= %s"
        params.append(filter_start_date.get())
    if filter_end_date.get():
        query += " AND date <= %s"
        params.append(filter_end_date.get())
    if filter_category.get() and filter_category.get()!="None":
        query += " AND category = %s"
        params.append(filter_category.get())
    if filter_start_amt.get():
        query += " AND amount >= %s"
        params.append(filter_start_amt.get())
    if filter_end_amt.get():
        query += " AND amount <= %s"
        params.append(filter_end_amt.get())

    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        cursor.execute(query, params)
        rows = cursor.fetchall()
        conn.close()
        cnt = 0
        for row in rows:
            tree.insert("", tk.END, values=row)
            cnt+=1
        tk.Label(root, text="Number of Expenses Shown: 
{}".format(cnt),font=f2,bg="Yellow",fg="Black",relief="sunken").place(x=700, y=190)
    except Exception as e:
        messagebox.showerror("Database Error", str(e))

# Clear Input Fields
def clear_fields():
    entry_date.delete(0, tk.END)
    combo_category.set("")
    entry_amount.delete(0, tk.END)
    entry_description.delete(0, tk.END)

# Edit Selected Expense
```

```python
def edit_expense():
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showerror("Selection Error", "No expense selected!")
        return

    expense_id = tree.item(selected_item, "values")[0]
    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        if len(entry_date.get())>0:
            date = entry_date.get()
        else:
            date = tree.item(selected_item, "values")[1]
        if len(combo_category.get())>0:
            category = combo_category.get()
        else:
            category = tree.item(selected_item, "values")[2]
        if len(entry_amount.get())>0:
            amount = entry_amount.get()
        else:
            amount = tree.item(selected_item, "values")[3]
        if len(entry_description.get())>0:
            description = entry_description.get()
        else:
            description = tree.item(selected_item, "values")[4]
        cursor.execute("UPDATE expenses SET date = %s, category = %s, amount = %s, description = %s WHERE id = %s", (date,category,amount,description,expense_id,))
        conn.commit()
        conn.close()
        messagebox.showinfo("Success", "Expense Edited successfully!")
        display_expenses()
    except Exception as e:
        messagebox.showerror("Database Error", str(e))

# Delete Selected Expense
def delete_expense():
    selected_item = tree.selection()
    if not selected_item:
        messagebox.showerror("Selection Error", "No expense selected!")
        return

    expense_id = tree.item(selected_item, "values")[0]
    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        l.discard(int(expense_id))
        cursor.execute("DELETE FROM expenses WHERE id = %s", (expense_id,))
        conn.commit()
        conn.close()
        messagebox.showinfo("Success", "Expense deleted successfully!")
        display_expenses()
    except Exception as e:
        messagebox.showerror("Database Error", str(e))

# Clear Filters
def clear_filters():
    filter_start_date.delete(0, tk.END)
    filter_end_date.delete(0, tk.END)
    filter_category.set("")
    filter_start_amt.delete(0,tk.END)
    filter_end_amt.delete(0,tk.END)
    display_expenses()

# Generate Reports
def generate_report():
    query = """
    SELECT
        category,
        COUNT(category) AS expense_count,
        SUM(amount) AS total_amount
    FROM expenses
    GROUP BY category
    ORDER BY SUM(amount) desc;
    """
    try:
        conn = connect_to_db()
        cursor = conn.cursor()
        cursor.execute(query)
        rows = cursor.fetchall()
        conn.close()

        # Display report in a new window
        report_window = tk.Toplevel(root)
        report_window.title("Expense Report")
        report_window.geometry("600x400")

        report_columns = ("Category", "Expense Count", "Total Amount")
        report_tree = ttk.Treeview(report_window, columns=report_columns, show="headings", 
height=15)
        for col in report_columns:
            report_tree.heading(col, text=col)
            report_tree.column(col, anchor=tk.W)
        report_tree.pack(fill=tk.BOTH, expand=True)

        for row in rows:
            report_tree.insert("", tk.END, values=row)

    except Exception as e:
        messagebox.showerror("Database Error", str(e))
```

```python

# GUI Setup
root = tk.Tk()
root.title("Enhanced Expense Tracker")
root.geometry("1400x750")

l = set()

f1 = font.Font(family="Times New Roman",size=14)
f2 = font.Font(family="Times New Roman",size=18)

for i in range(10,1300,200):
    for j in range(0,750,50):
        t1=tk.Label(root, text="My Expense Tracker", font=f1, fg="Dark Grey")
        t1.place(x=i,y=j)

img = tk.PhotoImage(file="Expensetrack.png")
imglabel = tk.Label(root, image=img)
imglabel.place(x=950,y=150)

# Input Fields
tk.Label(root, text="Date (YYYY-MM-DD) --",font=f1).place(x=100, y=20)
entry_date = tk.Entry(root,font=f1)
entry_date.place(x=300, y=20)

tk.Label(root, text="Category --",font=f1).place(x=100, y=60)
combo_category = ttk.Combobox(root, values=["Food", "Travel", "Shopping", "Medical", 
"Education", "Bills", "Others"],font=f1)
combo_category.place(x=300, y=60)

tk.Label(root, text="Amount --",font=f1).place(x=100, y=100)
entry_amount = tk.Entry(root,font=f1)
entry_amount.place(x=300, y=100)

tk.Label(root, text="Description --",font=f1).place(x=100, y=140)
entry_description = tk.Entry(root,font=f1)
entry_description.place(x=300, y=140)

# Buttons
btn_add = tk.Button(root, text="Add Expense", bg="Green", fg="White", font=f1, 
command=add_expense)
btn_add.place(x=50, y=180)

btn_clear = tk.Button(root, text="Clear", bg="Red", fg="White", font=f1, command=clear_fields)
btn_clear.place(x=500, y=180)

btn_delete = tk.Button(root, text="Delete Expense", bg="Blue", fg="White", font=f1, 
command=delete_expense)
btn_delete.place(x=350, y=180)

```

```python
btn_edit = tk.Button(root, text="Edit Expense", bg="Brown", fg="White", font=f1,
command=edit_expense)
btn_edit.place(x=200, y=180)

# Filter Fields
tk.Label(root, text="Filter Start Date --",font=f1).place(x=600, y=20)
filter_start_date = tk.Entry(root,font=f1)
filter_start_date.place(x=750, y=20)

tk.Label(root, text="Filter End Date --",font=f1).place(x=950, y=20)
filter_end_date = tk.Entry(root,font=f1)
filter_end_date.place(x=1100, y=20)

tk.Label(root, text="Filter Category --",font=f1).place(x=700, y=100)
filter_category = ttk.Combobox(root, values=["None", "Food", "Travel", "Shopping", "Medical",
"Education", "Bills", "Others"],font=f1)
filter_category.place(x=850, y=100)

tk.Label(root, text="Filter Start Amount --",font=f1).place(x=600, y=60)
filter_start_amt = tk.Entry(root,font=f1)
filter_start_amt.place(x=770, y=60)

tk.Label(root, text="Filter End Amount --",font=f1).place(x=970, y=60)
filter_end_amt = tk.Entry(root,font=f1)
filter_end_amt.place(x=1140, y=60)

btn_filter = tk.Button(root, text="Apply Filters", bg="Green", fg="White", font=f1,
command=display_expenses)
btn_filter.place(x=700, y=140)

btn_clear_filters = tk.Button(root, text="Clear Filters", bg="Red", fg="White", font=f1,
command=clear_filters)
btn_clear_filters.place(x=900, y=140)

btn_report = tk.Button(root, text="Generate Report", bg="Maroon", fg="White", font=f2,
relief="sunken", command=generate_report)
btn_report.place(x=1100,y=150)

# Expense List (Treeview)
columns = ("ID", "Date", "Category", "Amount", "Description")
tree = ttk.Treeview(root, columns=columns, show="headings", height=24)
for col in columns:
    tree.heading(col, text=col)
    tree.column(col, anchor=tk.W)
tree.place(x=20, y=230)

# Populate Expenses
display_expenses()

root.mainloop()

```