

Functions

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)

```
import json, os

SAVE_PATH = "/content/drive/MyDrive/expenses.json"

if os.path.exists(SAVE_PATH):
    with open(SAVE_PATH, "r") as f:
        expenses = json.load(f)
    print("✅ Loaded saved expenses from Drive.")
else:
    expenses = []
    print("⚠️ No saved file found, starting fresh.")

def save_expenses():
    with open(SAVE_PATH, "w") as f:
        json.dump(expenses, f)
```

✅ Loaded saved expenses from Drive.

```
def partition(arr, low, high):
    i = low - 1
    pivot = arr[high]
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i+1], arr[high] = arr[high], arr[i+1]
    return i+1

def quickSort(arr, low, high):
    if len(arr) == 1:
        return arr
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

def binaray_search(arr, low, high, x):
    if high >= low:
        mid = (high + low) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binaray_search(arr, low, mid - 1, x)
        else:
            return binaray_search(arr, mid + 1, high, x)
    return -1

def to_amounts(expenses):
    return [e["amount"] for e in expenses]

def find_insert_pos_amount(expenses, new_amount):
    arr = to_amounts(expenses)
    low, high = 0, len(arr)
```

```
while low < high:
    mid = (low + high) // 2
    if arr[mid] < new_amount:
        low = mid + 1
    else:
        high = mid
return low

def insert_sorted(expenses, expense):
    pos = find_insert_pos_amount(expenses, expense["amount"])
    expenses.insert(pos, expense)
    return pos

def delete_expense_by_item(expenses, item_name):
    for i, e in enumerate(expenses):
        if e["item"].casefold() == item_name.casefold():
            expenses.pop(i)
            return True
    return False

def count_by_category(expenses):
    out = {}
    for e in expenses:
        c = e["category"]
        out[c] = out.get(c, 0) + e["amount"]
    return out

def sort_expenses(expenses, key="amount", reverse=False):
    arr = [e[key] for e in expenses]
    if key != "amount":
        arr = list(map(str, arr))
    if arr:
        quickSort(arr, 0, len(arr)-1)
    if reverse:
        arr = arr[::-1]
    sorted_exp = []
    used = set()
    for v in arr:
        for idx, e in enumerate(expenses):
            val = str(e[key]) if key != "amount" else e[key]
            if val == v and idx not in used:
                sorted_exp.append(e)
                used.add(idx)
                break
    expenses[:] = sorted_exp

def search_items_by_name(expenses, query):
    q = query.casefold()
    return [e for e in expenses if q in e["item"].casefold()]

def search_by_category(expenses, category):
    q = category.casefold()
    return [e for e in expenses if e["category"].casefold() == q]

def find_indices_by_item(expenses, item_name):
    q = item_name.casefold()
    return [i for i, e in enumerate(expenses) if e["item"].casefold() == q]

def delete_expense_by_index(expenses, idx):
    if 0 <= idx < len(expenses):
        expenses.pop(idx)
        return True
```

```

    return False

def delete_expenses_by_indices(expenses, indices):
    for offset, idx in enumerate(sorted(indices)):
        expenses.pop(idx - offset)

def parse_selection_list(s, max_n):
    s = s.strip()
    if not s:
        return []
    out = set()
    for part in s.split(","):
        part = part.strip()
        if "-" in part:
            a, b = part.split("-", 1)
            if a.isdigit() and b.isdigit():
                start = int(a); end = int(b)
                if start <= end:
                    for k in range(start, end + 1):
                        if 1 <= k <= max_n:
                            out.add(k - 1)
        else:
            if part.isdigit():
                k = int(part)
                if 1 <= k <= max_n:
                    out.add(k - 1)
    return sorted(out)

def delete_subset_of_matches(expenses, matches, chosen_1based_indices):
    targets = [matches[i] for i in chosen_1based_indices]
    delete_expenses_by_indices(expenses, targets)

```

```

from IPython.display import clear_output
import pandas as pd

def _print_expenses(expenses):
    if not expenses:
        print("(no expenses yet)")
        return

    df = pd.DataFrame(expenses)
    df.index.name = "Index"
    display(df)  # shows a nice table in Colab

def _prompt_sort_and_apply():
    print("\nSort by:")
    print("1. Amount")
    print("2. Item name")
    print("3. Date")
    print("4. Category")
    sub = input("Enter choice: ").strip()
    key_map = {"1": "amount", "2": "item", "3": "date", "4": "category"}
    key = key_map.get(sub, "amount")
    order = input("Ascending (A) or Descending (D)? ").strip().lower()
    reverse = (order == "d")
    sort_expenses(expenses, key=key, reverse=reverse)
    print(f"Sorted by {key} ({'descending' if reverse else 'ascending'})")

def _prompt_search_and_show():
    print("\nSearch type:")
    print("1. Amount (binary search)")

```

```

print("1. Amount (binary search) ")
print("2. Item name (substring)")
print("3. Category (exact)")
s = input("Enter choice: ").strip()
if s == "1":
    arr = to_amounts(expenses)
    if arr:
        quickSort(arr, 0, len(arr)-1)
        x = int(input("Enter amount to search: ").strip())
        idx = binaray_search(arr, 0, len(arr)-1, x) if arr else -1
        print("Found at index:", idx if idx != -1 else "Not found")
elif s == "2":
    q = input("Enter item name substring: ").strip()
    res = search_items_by_name(expenses, q)
    print("No matches." if not res else "\n".join(map(str, res)))
elif s == "3":
    c = input("Enter category: ").strip()
    res = search_by_category(expenses, c)
    print("No matches." if not res else "\n".join(map(str, res)))
else:
    print("Invalid search choice.")

def _prompt_delete_with_choices():
    item = input("Enter item name to delete: ").strip()
    matches = find_indices_by_item(expenses, item)
    if not matches:
        print("No item with that exact name.")
        return
    if len(matches) == 1:
        confirm = input(f"Delete this? {expenses[matches[0]]} (y/n): ").strip().lower()
        if confirm == "y":
            delete_expense_by_index(expenses, matches[0])
            print("Deleted.")
        else:
            print("Cancelled.")
        return
    print("Multiple matches:")
    for k, idx in enumerate(matches, 1):
        e = expenses[idx]
        print(f"{k}. idx={idx} | {e}")
    sel = input("Type a number, 'all', a list like '1,3,5', a range like '1-3', or 'c': ")
    if sel == "all":
        delete_expenses_by_indices(expenses, matches)
        print("Deleted all matches.")
    elif sel == "c":
        print("Cancelled.")
    else:
        chosen = parse_selection_list(sel, len(matches))
        if not chosen:
            print("No valid selection.")
        else:
            delete_subset_of_matches(expenses, matches, chosen)
            print(f"Deleted {len(chosen)} record(s).")

def safe_input(prompt):
    import sys, time
    sys.stdout.flush()
    time.sleep(0.1)
    try:
        return input(prompt)
    except EOFError:
        time.sleep(0.2)
        return input(prompt)

```

```

def menu_loop():
    while True:
        clear_output(wait=True)
        _print_expenses(expenses)
        print("\n\n=== Expense Tracker ===")
        print("1. Add new expense (keeps sorted by amount)")
        print("2. Delete expense by item (choose records if duplicates)")
        print("3. Sort expenses (choose key + order)")
        print("4. Search (amount / item / category)")
        print("5. Show totals by category")
        print("0. Exit")
        choice = safe_input("Enter your choice: ").strip()
        _print_expenses(expenses)

        if choice == "1":
            date = input("Date (YYYY-MM-DD): ").strip()
            category = input("Category: ").strip()
            item = input("Item: ").strip()
            amount = int(input("Amount: ").strip())
            insert_sorted(expenses, {"date": date, "category": category, "item": item, "amount": amount})
            save_expenses()
            print("✅ Added & saved.")

        elif choice == "2":
            _prompt_delete_with_choices()
            save_expenses()

        elif choice == "3":
            _prompt_sort_and_apply()
            save_expenses()

        elif choice == "4":
            _prompt_search_and_show()

        elif choice == "5":
            totals = count_by_category(expenses)
            print("Totals by category:", totals)

        elif choice == "0":
            save_expenses()
            print("📁 Saved before exit. Goodbye!")
            break

        else:
            print("Invalid choice.")

        input("\nPress Enter to return to menu...")

```

▼ Run

menu_loop()


	date	category	item	amount
Index				
0	2022-10-14	Gaming	time	50
1	2023-10-23	Computer	CPU	2000
2	2023-10-25	Transport	Bus	100
3	2025-9-19	Food	Drink	34
4	2025-9-19	Food	Drink	50

=== Expense Tracker ===

1. Add new expense (keeps sorted by amount)
2. Delete expense by item (choose records if duplicates)
3. Sort expenses (choose key + order)
4. Search (amount / item / category)
5. Show totals by category
0. Exit

Enter your choice: 0

	date	category	item	amount
Index				
0	2022-10-14	Gaming	time	50
1	2023-10-23	Computer	CPU	2000
2	2023-10-25	Transport	Bus	100
3	2025-9-19	Food	Drink	34
4	2025-9-19	Food	Drink	50

 Saved before exit. Goodbye!