DOCUMENTATION: EXPENSE TRACKER

Starting Variables:

- store: (user input) store name, location of transaction
- expense_cat: (user input) category of the expense – groceries, supplies, splurge; can self-define
- amount: (user input) transaction amount in CAD
- date: (user input) date of the transaction
- shared: (default: False) True if the expense is shared with other people, False if not
- shared_number: (default: 0) number of people the expense is shared with
- percent_shared: (default: 0) percent of the expense you spend
- recurring_charge: (default: False) True if the expense is recurring, False if not
- name: (default: empty) the name(s) of the person/people the expense is shared with
- user_id: (user input) stores the name of the user

Library used:

- mysql.connector
- datetime
- calendar
- PrettyTable: to print out the monthly expense in a table format

FEATURES:

Class/Methods explanation:

Class: Expense – stores the details of the expense, allows user to call a report based on the specified date, summing amount spent monthly, edit (date or amount) or delete the recurring charge

1. calc_amount_per_person(): calculates the decimal value and therefore the amount shared with the person if the expense is shared
    a. No parameter
    b. Returns amount shared among people based on the decimal value calculated
2. get_store_id(): obtain store id from MySQL
    a. No parameter
    b. Returns the store id
3. get_cat_id(): obtain expense category id from MySQL
    a. No parameter
    b. Returns the expense category id
4. shared_person(): stores the details about the expense with people, expense can be shared with more than one person (in MySQL the same expense is stored on different line if there's more than one name input by the user)
    a. Parameters:
        i. amount_shared: the amount shared per person
        ii. store_id: the store's id
5. store_data(): store user input data about the expense into the database

   a. Parameters:
      i. store: name of store
      ii. expense_cat: expense’s category define by user
      iii. amount: full amount paid by user
      iv. date: date of the expense
      v. user_id: name of the user
      vi. shared: (default = False) if the expense is shared with another person
      vii. shared_number: (default = 0) number of people that shared the expense
      viii. percent_shared: (default = 0) percent of the expense user spend
      ix. recurring_charge: (default = False) if the expense is a recurring charge
      x. name: a list of name or names of people the expense is shared with
   b. Stores user’s input data in the designated variables
   c. calls both get_store_id() and get_cat_id() to obtain the store id and category id, respectively
   d. If shared variable is true, calls calc_amount_per_person() to get the amount shared per person and calls shared_person() method to store the shared expense information. If the shared variable is false, the amount_shared variable is set to 0.
   e. After obtaining all the information, the store_data() method stores the expense in MySQL
   f. Returns “Data Stored” to indicate that the data has been stored

6. sum_monthly(): allow user to see report of monthly expense
   a. Parameters:
      i. start_range: start date from user’s input
      ii. end_range: end date from user’s input
      iii. user_id: user’s name
      iv. leap_year: (default = False) whether the year is a leap year
   b. New varibles:
      i. start_date: date at the start of the expense from user’s input
      ii. end_date: date at the end of the expense from user’s input
   c. Calls the stored procedure in MySQL to get the monthly data dependent on range of date and user’s id
   d. Uses the data obtained from MySQL to print out the report:
      i. Loops through results stored in mycursor from MySQL and stores the results in a list (list name: list_results) in the case of multiple rows
         1. Each item in the list is stored as a tuple
         2. Tuple contains: (date, store’s name, expense’s category, expense’s amount, if the expense is shared, percent shared per person, if the expense is a recurring charge, amount per person, name the expense is shared with)
      ii. Set new temporary variables so these variables can be used to print out individual rows and weed out repeated rows (eg. Rows that corresponds to the shared_expense with others)
         1. total_amount: sums up the total amount for the time range called by user
         2. total_shared_amount: total amount that is shared with others

            3. name_shared_person: a list of name(s) of the person shared, used for printing out the total amount shared with each person at the end of report
            4. current/previous_store: store's name
            5. current/previous_amount: expense's amount
            6. current/previous_date: expense's date
            7. current/previous_category: expense's category
            8. shared: whether the expense is shared or not
        iii. Loops through list_results and set corresponding variable names with the correct data
            1. Set the "current" variables to the corresponding values
            2. Conditional statement: if the expense is shared, set "shared" variable to 'Yes'. If not, then set to 'No'.
            3. Conditional statement: checks if the row is a repeat of the previous row. If it is a repeat, print_status is set to 0, meaning it won't print. Else: current_amount is added to the total_amount, print_status is set to 1, and if the expense is a shared amount, the amount is added to the total_shared_amount.
            4. Use the same set of values to set the "previous" variables
            5. Conditional statement: checks if the expense is a recurring charge, if True, recurring is set to 'Yes'. If False, set to 'No'.
            6. Conditional statement: checks the print_status. If it is True, then the tuple of values are added to the print_table.
        iv. Prints out the final report
        v. Prints the total amount
        vi. Calls the get_sum_by_category with parameter start_date and end_date to get all the expense categories within the date range and summation by category. Summation by category is printed
        vii. Prints the total amount of the expense shared with others
        viii. Calls stored procedure in MySQL to get summation of expense shared with each person, which is then printed
7. get_sum_by_category():
    a. Parameters:
        i. start_range: start date from user's input
        ii. end_range: end date from user's input
    b. New variables:
        i. cat_dic: stores the results from calling fetch_expense_categories()
        ii. expense_by_cat: a list of tuples: (category id, sum of amount)
    c. Calls fetch_expense_categories to obtain all the expense's categories present within the date range indicated by user
    d. Calls stored procedure in MySQL to get total amount of each category – stores in expense_by_cat
    e. Returns expense_by_cat
8. change_amount_recuring_charge(): allows the user to edit the charge of a recurring charge on a particular date
    a. Parameters:

            i. date: date of the charge user want to edit
            ii. store: name of the store of the recurring charge
            iii. old_amount: the original amount of the recurring charge
            iv. new_amount: the amount the user wants to change to
            v. category: the category of the charge
    b. Calls the stored procedure in MySQL
    c. Returns a status whether the charge was changed or the charge doesn't exit

9. change_date_recurring_charge(): allows the user to edit the date of the recurring charge for a particular date
    a. Parameters:
        i. old_date: the original date of the charge
        ii. new_date: the new date the recurring charge would be changed to
        iii. store: name of the store of the charge
        iv. old_amount: the original amount of the recurring charge
        v. category: category of the charge
    b. Calls stored procedure in MySQL to edit the date of the recurring charge
    c. Returns a status whether the charge was changed or the charge doesn't exit
10. delete_recurring_charge(): allows user to delete a range of recurring charge
    a. Parameters:
        i. start_date: start of the range the user want the recurring charge to be deleted
        ii. end_date: end of the range the user want to recurring charge to be deleted
        iii. store: name of the store of the charge
        iv. amount: amount of the charge
        v. category: category of the charge
    b. Calls stored procedure looking the rows of the charge in MySQL fetching the row id(s)
    c. Calls another stored procedure to delete the rows obtained from the previous query
    d. Returns status that the rows were deleted

MySQL tables:

Expense: primary key: ID (auto-generated) main expense table, stores all the expenses inputted by user

#	Field	Table	Type
1	ID	expense	INT
2	STORE_ID	expense	INT
3	Amount	expense	FLOAT
4	Shared_expense	expense	BOOL
5	Shared_number	expense	INT
6	Percent_shared	expense	FLOAT
7	Category_id	expense	INT
8	Recurring_charge	expense	BOOL
9	Shared_amount	expense	FLOAT
10	Date	expense	DATE
11	USER_ID	expense	VARCHAR(500)

Store: primary key: Store_id (auto-generated) stores name of the stores and their id

#	Field	Table	Type
1	Store_name	store	VARCHAR(500)
2	Store_id	store	INT

Expense_categories: primary key: Category_id (auto-generated) stores the expense categories and their id

#	Field	Table	Type
1	Expense_cat	expense_categories	VARCHAR(500)
2	Category_id	expense_categories	INT

Shared_person: primary key: ID (auto-generated) stores the name

#	Field	Table	Type
1	ID	shared_person	INT
2	Name	shared_person	VARCHAR(500)
3	expense_date	shared_person	DATE
4	amount	shared_person	FLOAT
5	store_id	shared_person	INT