|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 1  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1(6) | Описати неіменовану функцію для об'єднання голів трьох списків в один список, вихідні дані взяти з таблиці 1 |
| Реалізація | Анонімна функція, що містить в собі використання функції list та first(аналог head в SML/LISP) |
| Результати |  |
|  |  |
| Завдання № 2(6) | Описати іменовану функцію для створення нового списку з елементів декількох вихідних списків. В якості вихідних списків використовувати списки таблиці 1. Номери елементів списків взяти в таблиці 2. |
| Реалізація | Функція foo що приймає 3 листа та повертає один лист |
| Результати |  |
|  |  |
| Завдання № 3(6) | Описати іменовану функцію для обчислення результату відповідно до варіанта індивідуального завдання зі списку. Набори для множин взяти в таблиці 1 |
| Реалізація | Foo\_t приймає 3 листа та повертає 1 сет |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 2  Звіт з лабораторної роботи № 2  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1 | Реалізувати функцію для генерації послідовності із заданою кількістю елементів на виході |
| Реалізація | Функція foo1 приймає число і повертає ліст значень, використовуючи функцію factorials |
| Результати |  |
|  |  |
| Завдання № 2 | Реалізувати сортування послідовності із першого завдання(merge sort) |
| Реалізація | Функція merge sort з відомим алгоритмом і допоміжною функцією merge step |
| Результати |  |
|  |  |
| Завдання № 3 | Реалізувати сортування за алфавітом послідовності слів використовуючи алгоритм з попереднього завдання 2. Реалізувати сортування за довжиною слова послідовності слів використовуючи алгоритм з попереднього завдання |
| Реалізація | Функція mergesort фільтрує додатковий параметр, який вказує на метод сортування. |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 3  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1(6) | Підготувати дані з файлів для роботи з ними у форматі списків згідно до вашої мови програмування |
| Реалізація |  |
| Результати |  |
|  |  |
| Завдання № 2(6) | Реалізувати cli(command line interface) для запитів до основного програмного файлу з лабораторними |
| Реалізація | Встановив Leiningen, підключив Calva |
| Результати |  |
|  |  |
| Завдання № 3(6) | Реалізувати форматований вивід обраних таблиць в cli за запитом load(“filename”) |
| Реалізація | Функція main приймає на вхід строку, парсить з неї назву файлу та додає в пам’ять дані файлу модулем clojure/csv. Після чого відпрацьовує функція PrintTable, яка друкує таблицю в термінал. |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 4  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1 | Реалізувати запит для cli згідно таблиць з минулої лабораторної роботи SELECT column1, column2, ... FROM table\_name; |
| Реалізація | На всі подальші лаборабторні роботи створені 2 дополвновальні функції: parseSQL та parseRequest. Перша потрібна для створення відповідних запиту датафреймів, друга потрібна для виявлення ключових слів у запиті. На цьому етапі додатково створені наступні функції:   * simpleColsParser(прибирає коми з назв колонок у векторі) * oneRowSelect(робить мапу з мапи існуючої таблиці, яка мають саме обрані колонки) * newSimpleExpressions(використовує попередню функцію на всій таблиці) |
| Результати |  |
|  |  |
| Завдання № 2 | Реалізувати опцію DISTINCT для запиту в cli згідно таблиць з минулої лабораторної роботи SELECT DISTINCT column1, column2, ... FROM table\_name; |
| Реалізація | Функція distinctSelect приймає вектор мап і повертає вектор мап без повторів. |
| Результати |  |
|  |  |
| Завдання № 3(5) | Реалізувати опцію WHERE з умовами згідно варіанту для запиту в cli згідно таблиць з минулої лабораторної роботи SELECT \* FROM Customers WHERE CustomerID=1 |
| Реалізація | Додані функції:   * twoDFintoOne(склеює два датафрейми в залежності від оператора or/and) * recursiveDFs(використовує попередню функцію для склеювання декількох датафреймів) * recursiveWheresParser (формує два вектори: один з виразами >/=/not, другий з виразами and/or) * oneExpressionParser (формує датафрейм на основі одного виразу) * whereExpressionParser (формує цілий датафрейм на основі попередніх функцій) |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 5  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1 | Додати опції AND OR NOT до запиту SELECT для cli |
| Реалізація | Реалізація цієї функції описана у попередній лабораторній роботі |
| Результати |  |
|  |  |
| Завдання № 2 | Додати опцію ORDER BY для запиту SELECT в cli |
| Реалізація | Додані наступні функції:   * OrderMaker(сортує таблицю за заданими колонками і порядком) * OrderParser(парсить колонки, за якими треба сортувати і повертає датафрейм відсортованої таблиці; використовує попередню функцію) |
| Результати |  |
|  |  |
| Завдання № 3(5) | Додати функції для запиту для запиту SELECT в cli(Min/Avg/count) |
| Реалізація | Усі три реалізовані за допомогою функції вищого порядку HigherOrderFun, в яку передаються таблиця, min/count/avg та повертається таблиця з одним значенням. |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 6  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1 | Додати опції JOIN до запиту SELECT для cli згідно таблиць з минулої лабораторної роботи |
| Реалізація | Створені наступні функції:   * getJoinedTable—повертає вже з’єднані дві таблиці в залежності від методу поєднання. * ColumnsForJoin—парсить колонки, за якими треба з’єднувати таблиці. * allKeysChecker—перевіряє, чи всі назви колонок було використано у рядках таблиці. * innerJoin—сам джойн, з’єднує тільки рядки зі спільним ключем. |
| Результати |  |
|  |  |
| Завдання № 2 | Додати опцію FULL OUTER для запиту SELECT в cli згідно таблиць з минулої лабораторної роботи |
| Реалізація | Додана функція outterJoin, яка поєднує таблиці за принципом логічної дії UNION. |
| Результати |  |
|  |  |
| Завдання № 3 | Додати функції для запиту для запиту SELECT в cli RIGHT JOIN |
| Реалізація | Додана функція rightJoin, яка поєднує таблиці, використовуючи outterJoin та залишаючи тільки спільні рядки і рядки, що були у правій таблиці. |
| Результати |  |
|  |  |

|  |  |
| --- | --- |
| Звіт з лабораторної роботи № 7  Сучасні технології програмування – 1 : Функціональне програмування  Студент : Шабанов Бунямін Група : ІП-82  Мова виконання: Clojure | |
| Завдання № 1(6) | Описати неіменовану функцію для об'єднання голів трьох списків в один список, вихідні дані взяти з таблиці 1 |
| Реалізація | Анонімна функція, що містить в собі використання функції list та first(аналог head в SML/LISP) |
| Результати |  |
|  |  |
| Завдання № 2(6) | Описати іменовану функцію для створення нового списку з елементів декількох вихідних списків. В якості вихідних списків використовувати списки таблиці 1. Номери елементів списків взяти в таблиці 2. |
| Реалізація | Функція foo що приймає 3 листа та повертає один лист |
| Результати |  |
|  |  |
| Завдання № 3(6) | Описати іменовану функцію для обчислення результату відповідно до варіанта індивідуального завдання зі списку. Набори для множин взяти в таблиці 1 |
| Реалізація | Foo\_t приймає 3 листа та повертає 1 сет |
| Результати |  |