1. **Цель и задачи метода**

Метод GetSeasonSales решает задачу расчета сезонных продаж для продуктов, с вычислением процентов роста продаж по каждому месяцу. Он объединяет данные о продажах и вычисляет процентное соотношение проданных единиц каждого продукта за каждый месяц в году.

Метод GetSeasonSales() выполняет следующие действия:

* Группировка по имени продукта. Метод использует LINQ для группировки всех продаж по имени продукта, что позволяет вычислить статистику по каждому продукту отдельно.
* Вычисление общего объема продаж для каждого продукта в течение всего года.
* Вычисление роста продаж для каждого месяца. Для каждого месяца метод вычисляет, какой процент от общего объема продаж составляют продажи в этом месяце.
* Округление процентов до двух знаков после запятой с помощью Math.Round().

Метод возвращает список объектов типа SeasonProductInfo, в которых хранятся проценты роста продаж по месяцам для каждого продукта. Эти данные затем используются в тестах для проверки корректности вычислений.

1. **Как воспользоваться сценарием в приложении**

Пошаговое руководство для работы с функционалом сезонного анализа продаж:

1. Открытие формы. Нажмите кнопку “Сезонные товары”, открывающую форму сезонных продаж. При загрузке формы таблица сразу заполнится обобщенными данными за год для каждого товара.
2. Просмотр данных. В таблице отображается распределение продаж по месяцам для каждого продукта, что помогает увидеть пиковые периоды.
3. Фильтрация. Выберите нужный месяц из выпадающего списка и введите пороговое значение (в процентах) в текстовом поле. Нажмите кнопку фильтрации – таблица обновится и покажет товары, удовлетворяющие выбранным критериям.
4. Сброс фильтра. Для возврата к исходному набору данных нажмите кнопку сброса.
5. Экспорт данных. Если необходимо сохранить результаты, воспользуйтесь кнопкой сохранения для экспорта таблицы в HTML.

Эта инструкция позволяет быстро и удобно использовать модуль сезонного анализа продаж для получения нужной информации и последующего анализа.

1. **Формирование чек-листов**

Чек-лист для проверки работоспособности функции:

✅ Проверка на пустой список продаж.

✅ Проверка на корректный расчет процента для одного продукта в июне.

✅ Проверка на несколько продуктов с продажами в разные месяцы.

✅ Проверка на корректное округление процентов для месяца.

✅ Проверка на корректную группировку по продукту и вычисление процентов для каждого месяца.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Проверка** | **Ожидаемый результат** | **Метод проверки** |
| 1 | Пустой список продаж | Метод должен вернуть пустую коллекцию (Count = 0) | Assert.AreEqual(0, result.Count) |
| 2 | Расчет процентов для продукта в июне | Для продукта с продажей только в июне: процент для июня = 100%, для остальных месяцев = 0 | CollectionAssert.AreEqual(new double[]{0,0,0,0,0,100,0,0,0,0,0,0}, actualArray) |
| 3 | Расчет для нескольких продаж одного продукта | Для продукта с продажами, распределёнными по месяцам (например, 20 в январе, 80 во втором месяце). | CollectionAssert.AreEqual(new double[]{20,80,0,0,0,0,0,0,0,0,0,0}, actualArray) |
| 4 | Корректное округление процентов | Значения процентов должны быть округлены до двух знаков | CollectionAssert.AreEqual(new double[]{0,0,33.33,66.67,0,0,0,0,0,0,0,0}, actualArray) |
| 5 | Группировка по имени продукта | Записи с одинаковым именем объединяются с суммированием продаж | CollectionAssert.AreEqual(new double[]{0,0,0,0,10,10,80,0,0,0,0,0}, actualArray) |

*Таблица 1. – Формирование чек-листов*

1. **Чек-лист для проверки работоспособности функции**

Чек-лист (с успешным/неуспешным результатом):[[1]](#footnote-1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Данные 1** | **Данные 2** | **Результат** | **Статус** |
| 1 | Нет данных | Пустой список продаж | Возвращается пустой список | ✅ |
| 2 | Одна продажа | Товар "Product1": 50 шт. (продажа в июне) | Процент для июня = 100%, для остальных месяцев = 0% | ✅ |
| 3 | Несколько продаж одного товара | Товар "Product2": 20 шт. (январь), 30 и 50 шт. (февраль) | Процент для каждого месяца для каждого продукта | ✅ |
| 4 | Округление процентов | Товар "Product3": 1 шт. (март), 2 шт. (апрель) | Процент для марта = 33.33%, для апреля = 66.67% | ✅ |
| 5 | Группировка по товарам | Товар "Product4": 10 шт. (май), 10 шт. (июнь), 80 шт. (июль) | Правильное распределение процентов по месяцам | ✅ |

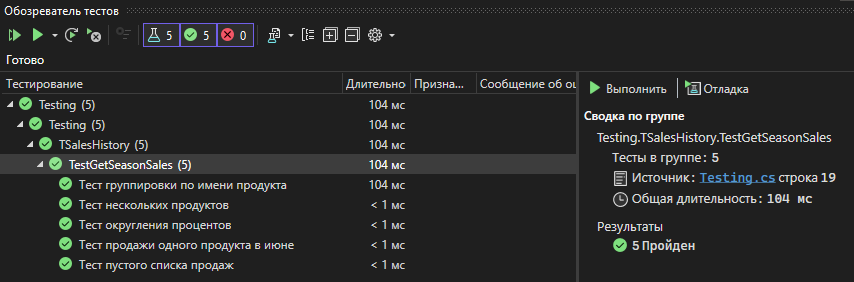
*Таблица 2. – Чек-лист для проверки работоспособности функции*

1. **Результаты тестирования**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Тест-кейс** | **Данные** | **Ожидаемый результат** | **Фактический результат** | **Статус** |
| 1 | Пустой список | Нет продаж | Метод возвращает пустую коллекцию (Count = 0) | Count = 0 | ✅ |
| 2 | Одна продажа | Product1:  50 шт. (июнь) | Продажи в июне составляют 100%, в остальных месяцах – 0% | Июнь: 100%, остальные: 0% | ✅ |
| 3 | Несколько продаж | Product2:  20 (янв), 50 (фев) | Январь – 20%, Февраль – 80%, остальные – 0% | Январь: 20%, Февраль: 80%, остальные: 0% | ✅ |
| 4 | Округление | Product3:  1 (март), 2 (апр) | Март ≈ 33.33%, Апрель ≈ 66.67%, остальные – 0% | Март: 33.33%, Апрель: 66.67%, остальные: 0% | ✅ |
| 5 | Группировка | Product4:  10 (май), 10 (июнь), 80 (июль) | Для Product4 продажи с разных записей суммируются. | Май: 10%, Июнь: 10%, Июль: 80%, остальные: 0% | ✅ |

*Таблица 3. – Результаты тестирования*

1. **Автоматизированное тестирование**

****

|  |
| --- |
| **[**TestClass**]**  **public** class TSalesHistory  **{**  **private** SalesHistory salesHistory**;**  // Тестовые случаи: каждая коллекция продаж для одного теста  **private** List**<**List**<**ProductInfo**>>** \_allTestCases **=** **new** List**<**List**<**ProductInfo**>>();**  **[**TestInitialize**]**  **public** void TestInitialize**()**  **{**  salesHistory **=** **new** SalesHistory**();**  \_allTestCases**.**Clear**();**  // 0: Пустой список  \_allTestCases**.**Add**(new** List**<**ProductInfo**>());**  // 1: Один продукт, одна продажа в июне  \_allTestCases**.**Add**(new** List**<**ProductInfo**>**  **{**  **new** ProductInfo **{** Name **=** "Product1"**,** QuantitySold **=** 50**,** LastSell **=** **new** DateTime**(**2023**,** 6**,** 15**)** **}**  **});**  // 2: Несколько продаж одного продукта  \_allTestCases**.**Add**(new** List**<**ProductInfo**>**  **{**  **new** ProductInfo **{** Name **=** "Product2"**,** QuantitySold **=** 20**,** LastSell **=** **new** DateTime**(**2023**,** 1**,** 10**)** **},**  **new** ProductInfo **{** Name **=** "Product2"**,** QuantitySold **=** 30**,** LastSell **=** **new** DateTime**(**2023**,** 2**,** 15**)** **},**  **new** ProductInfo **{** Name **=** "Product2"**,** QuantitySold **=** 50**,** LastSell **=** **new** DateTime**(**2023**,** 2**,** 20**)** **}**  **});**  // 3: Тест округления процентов  \_allTestCases**.**Add**(new** List**<**ProductInfo**>**  **{**  **new** ProductInfo **{** Name **=** "Product3"**,** QuantitySold **=** 1**,** LastSell **=** **new** DateTime**(**2023**,** 3**,** 5**)** **},**  **new** ProductInfo **{** Name **=** "Product3"**,** QuantitySold **=** 2**,** LastSell **=** **new** DateTime**(**2023**,** 4**,** 5**)** **}**  **});**  // 4: Группировка по имени продукта  \_allTestCases**.**Add**(new** List**<**ProductInfo**>**  **{**  **new** ProductInfo **{** Name **=** "Product4"**,** QuantitySold **=** 10**,** LastSell **=** **new** DateTime**(**2023**,** 5**,** 1**)** **},**  **new** ProductInfo **{** Name **=** "Product4"**,** QuantitySold **=** 10**,** LastSell **=** **new** DateTime**(**2023**,** 6**,** 1**)** **},**  **new** ProductInfo **{** Name **=** "Product4"**,** QuantitySold **=** 80**,** LastSell **=** **new** DateTime**(**2023**,** 7**,** 1**)** **}**  **});**  **}**  **[**DataTestMethod**]**  **[**DataRow**(**0**,** "Product1"**,** "0,0,0,0,0,100,0,0,0,0,0,0"**,** DisplayName **=** "Тест пустого списка продаж"**)]**  **[**DataRow**(**1**,** "Product1"**,** "0,0,0,0,0,100,0,0,0,0,0,0"**,** DisplayName **=** "Тест продажи одного продукта в июне"**)]**  **[**DataRow**(**2**,** "Product2"**,** "20,80,0,0,0,0,0,0,0,0,0,0"**,** DisplayName **=** "Тест нескольких продаж одного продукта"**)]**  **[**DataRow**(**3**,** "Product3"**,** "0,0,33.33,66.67,0,0,0,0,0,0,0,0"**,** DisplayName **=** "Тест округления процентов"**)]**  **[**DataRow**(**4**,** "Product4"**,** "0,0,0,0,10,10,80,0,0,0,0,0"**,** DisplayName **=** "Тест группировки по имени продукта"**)]**  **public** void TestGetSeasonSales**(**int testCaseIndex**,** string expectedName**,** string expectedGrowthCsv**)**  **{**  // Arrange: создаём новый экземпляр SalesHistory и добавляем данные из тестового случая  salesHistory **=** **new** SalesHistory**();**  **foreach** **(**var product **in** \_allTestCases**[**testCaseIndex**])**  **{**  salesHistory**.**AddSales**(**product**);**  **}**  // Act: вызываем метод GetSeasonSales  BindingList**<**SeasonProductInfo**>** result **=** salesHistory**.**GetSeasonSales**();**  // Assert: если список пустой, проверяем, что он пустой  **if** **(**testCaseIndex **==** 0**)**  **{**  Assert**.**AreEqual**(**0**,** result**.**Count**,** "Ожидается пустой список при отсутствии продаж."**);**  **return;**  **}**  // Создаем ожидаемый объект SeasonProductInfo на основе входных данных  double**[]** expectedGrowth **=** expectedGrowthCsv**.**Split**(**','**)**  **.**Select**(**s **=>** Math**.**Round**(**double**.**Parse**(**s**.**Trim**(),** CultureInfo**.**InvariantCulture**),** 2**))**  **.**ToArray**();**  SeasonProductInfo expected **=** **new** SeasonProductInfo  **{**  Name **=** expectedName**,**  JanuaryGrowth **=** expectedGrowth**[**0**],**  FebruaryGrowth **=** expectedGrowth**[**1**],**  MarchGrowth **=** expectedGrowth**[**2**],**  AprilGrowth **=** expectedGrowth**[**3**],**  MayGrowth **=** expectedGrowth**[**4**],**  JuneGrowth **=** expectedGrowth**[**5**],**  JulyGrowth **=** expectedGrowth**[**6**],**  AugustGrowth **=** expectedGrowth**[**7**],**  SeptemberGrowth **=** expectedGrowth**[**8**],**  OctoberGrowth **=** expectedGrowth**[**9**],**  NovemberGrowth **=** expectedGrowth**[**10**],**  DecemberGrowth **=** expectedGrowth**[**11**]**  **};**  // Получаем фактический объект (например, первый в списке)  SeasonProductInfo actual **=** result**.**First**();**  // Помещаем объекты в списки и сравниваем их через CollectionAssert  var expectedList **=** **new** List**<**SeasonProductInfo**>** **{** expected **};**  var actualList **=** **new** List**<**SeasonProductInfo**>** **{** actual **};**  CollectionAssert**.**AreEqual**(**expectedList**,** actualList**);**  **}**  **}** |

**Финальный вердикт:**

✅ Функция работоспособна, все проверки пройдены.

1. Примечание. Пункт чек-листа успешен, если все проверки с результатами успешны. [↑](#footnote-ref-1)