LINQ Query Usage Report

This document details the usage of LINQ (Language Integrated Query) in the **Smart Health Insurance System** backend to demonstrate proficiency in data manipulation, reporting, and efficient database querying.

1. Filtering (Where **Clause)**

Used to narrow down data based on specific criteria.

- **Location:**

  Services/ClaimService.cs

- **Use Case:** Filtering claims by User ID, Hospital ID, Status, and Search Term for the dashboard.

Code:

```
Expression<Func<Claim, bool>> filter = c =>
  (!userId.HasValue || c.UserId == userId.Value) &&
  (!targetHospitalId.HasValue || c.HospitalId == targetHospitalId.Value) &&
  (!targetStatus.HasValue || c.Status == targetStatus.Value) &&
  (searchTerm == null || c.ClaimNumber.Contains(searchTerm));
```

2. Grouping (GroupBy)

Used in reports to categorize data.

- **Location:** Services/ReportService.cs
- **Use Case 1:** Grouping payments by Month to show revenue trends.

  Code:

  ```
  var trend = payments
    .GroupBy(p => new { p.PaymentDate.Year, p.PaymentDate.Month })
    .Select(...)
  ```

- **Use Case 2:** Grouping payments by Insurance Plan to compare premium income vs. claim payouts.

  Code:

  ```
  var paymentGroups = allPayments
    .GroupBy(p => p.Policy?.Plan?.PlanName ?? "Unknown Plan")
    .ToDictionary(g => g.Key, g => g.Sum(p => p.Amount));
  ```

3. Aggregation (Sum, Count)

Used to calculate totals for analytic dashboards.

- **Location:** Services/ReportService.cs
- **Use Case:** Calculating total revenue, total active claims, and coverage amounts.

  Code:

  Revenue = g.Sum(p => p.Amount) // Calculating total revenue per month

  var activeClaims = await _claimRepository.CountAsync(c => c.Status != ClaimStatus.Paid);

4. Pagination (Skip & Take)

Used for efficient data loading in grids/tables.

- **Location:** Repositories/Repository.cs (Generic Repository)
- **Use Case**: Fetching data in "Pages" (e.g., 10 records at a time) to improve performance.

  Code:

  var items = await query

  .Skip((page - 1) * pageSize)

  .Take(pageSize)

  .ToListAsync();

5. Sorting (OrderBy, OrderByDescending)

Used to organize report data.

- **Location:** Services/ReportService.cs
- **Use Case:** Sorting revenue trend data chronologically.

  Code:

  .OrderBy(x => DateTime.Parse(x.Month))

6. Projections (Select)

Used to shape data into DTOs (Data Transfer Objects) to avoid exposing full database entities.

- **Location:** Services/ReportService.cs

  Code:

  .Select(g => new RevenueTrendDto

  {

  Month = ...,

  Revenue = g.Sum(p => p.Amount)

  })