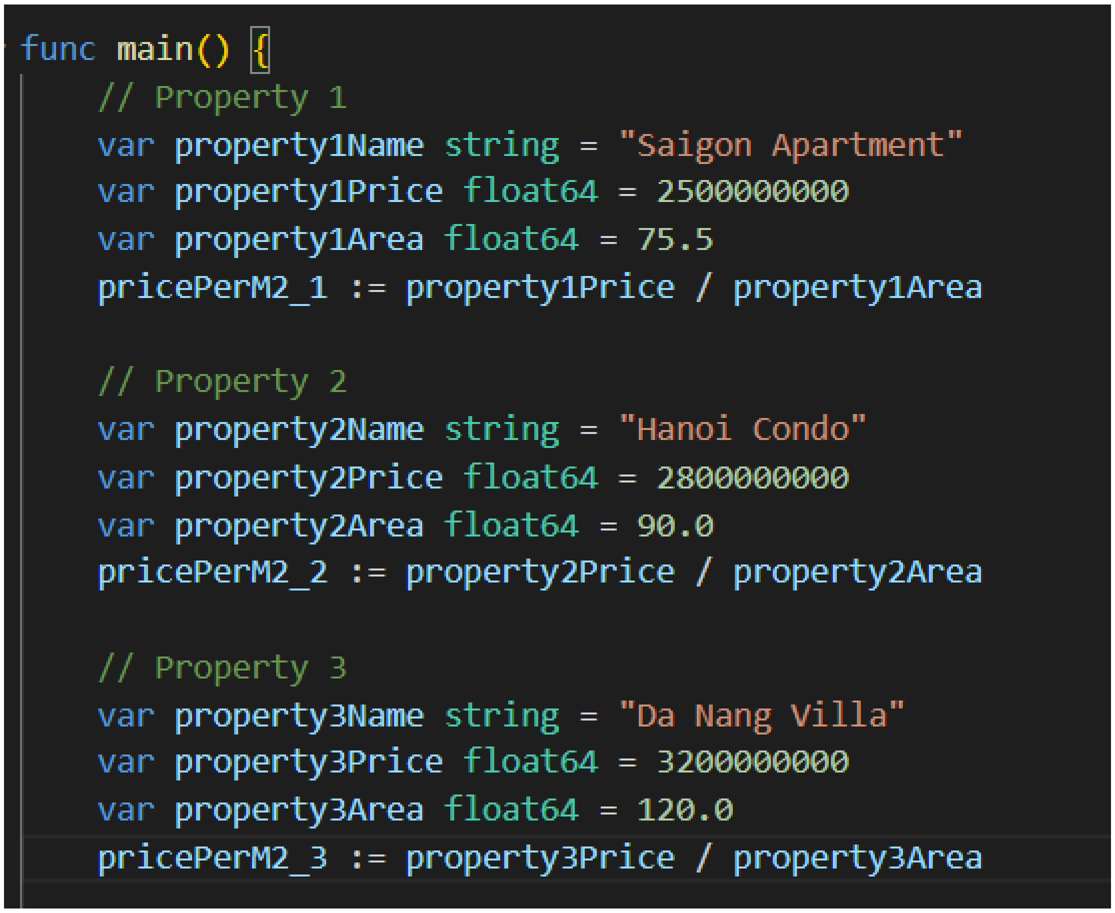
Part 1:

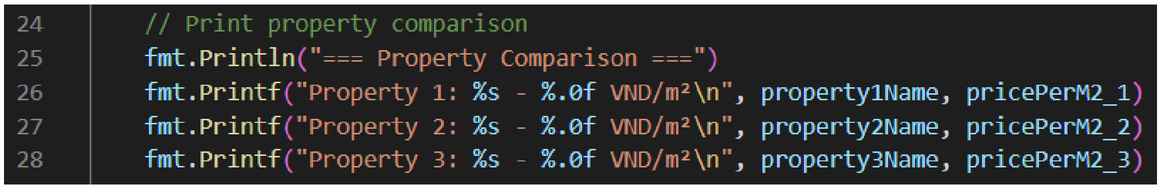
Task 1.1:

For each of the three properties, we declare variables holding (name, price, area) and calculate price per square meter for each property.



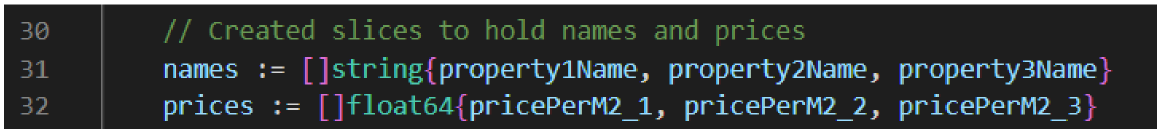
```go
func main() {
    // Property 1
    var property1Name string = "Saigon Apartment"
    var property1Price float64 = 2500000000
    var property1Area float64 = 75.5
    pricePerM2_1 := property1Price / property1Area

    // Property 2
    var property2Name string = "Hanoi Condo"
    var property2Price float64 = 2800000000
    var property2Area float64 = 90.0
    pricePerM2_2 := property2Price / property2Area

    // Property 3
    var property3Name string = "Da Nang Villa"
    var property3Price float64 = 3200000000
    var property3Area float64 = 120.0
    pricePerM2_3 := property3Price / property3Area
```

Prints the price per m² for all three properties.



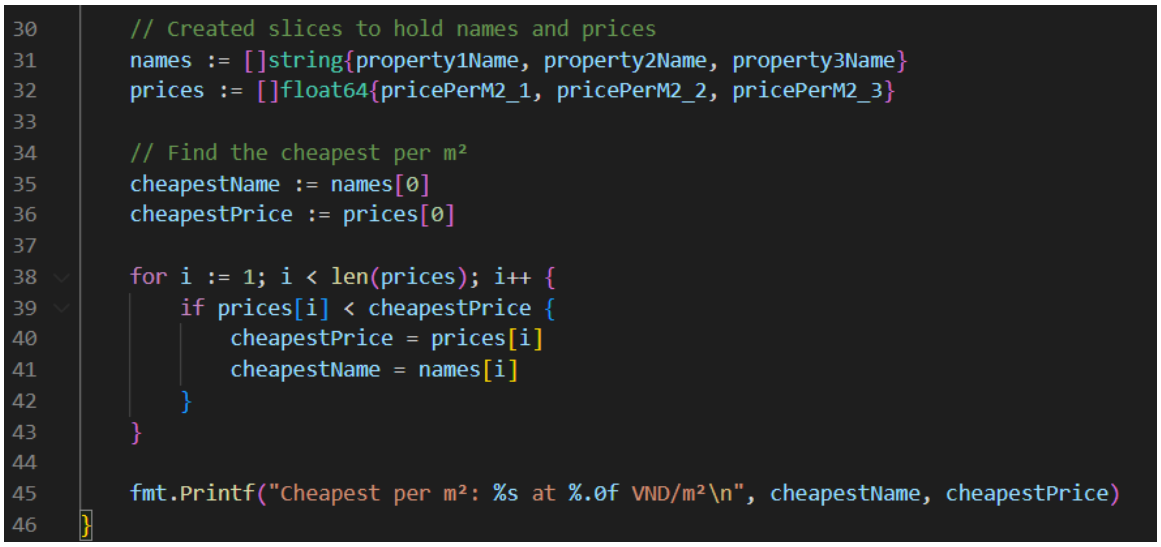
```go
    // Print property comparison
    fmt.Println("=== Property Comparison ===")
    fmt.Printf("Property 1: %s - %.0f VND/m²\n", property1Name, pricePerM2_1)
    fmt.Printf("Property 2: %s - %.0f VND/m²\n", property2Name, pricePerM2_2)
    fmt.Printf("Property 3: %s - %.0f VND/m²\n", property3Name, pricePerM2_3)
```

Uses slices to store property names and prices.



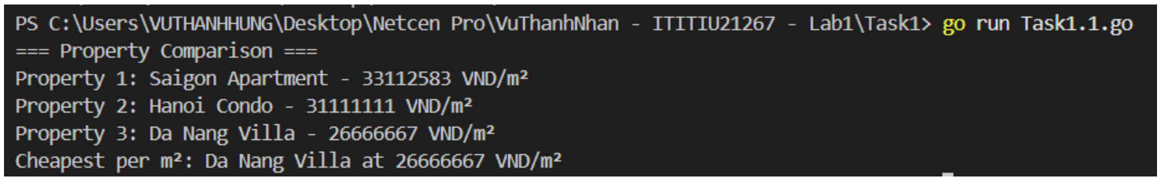
```go
    // Created slices to hold names and prices
    names := []string{property1Name, property2Name, property3Name}
    prices := []float64{pricePerM2_1, pricePerM2_2, pricePerM2_3}
```

Loops through prices to find the cheapest property and prints the name and price of the cheapest property.



```go
    // Created slices to hold names and prices
    names := []string{property1Name, property2Name, property3Name}
    prices := []float64{pricePerM2_1, pricePerM2_2, pricePerM2_3}

    // Find the cheapest per m²
    cheapestName := names[0]
    cheapestPrice := prices[0]

    for i := 1; i < len(prices); i++ {
        if prices[i] < cheapestPrice {
            cheapestPrice = prices[i]
            cheapestName = names[i]
        }
    }

    fmt.Printf("Cheapest per m²: %s at %.0f VND/m²\n", cheapestName, cheapestPrice)
}
```

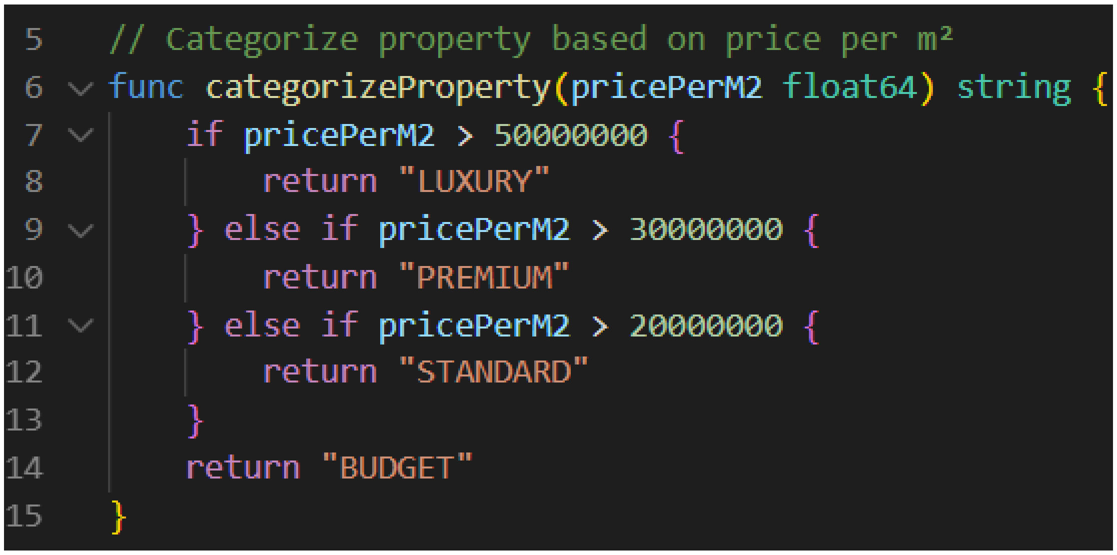
⇨ Result:



```
PS C:\Users\VUTHANHHUNG\Desktop\Netcen Pro\VuThanhNhan - ITITIU21267 - Lab1\Task1> go run Task1.1.go
=== Property Comparison ===
Property 1: Saigon Apartment - 33112583 VND/m²
Property 2: Hanoi Condo - 31111111 VND/m²
Property 3: Da Nang Villa - 26666667 VND/m²
Cheapest per m²: Da Nang Villa at 26666667 VND/m²
```

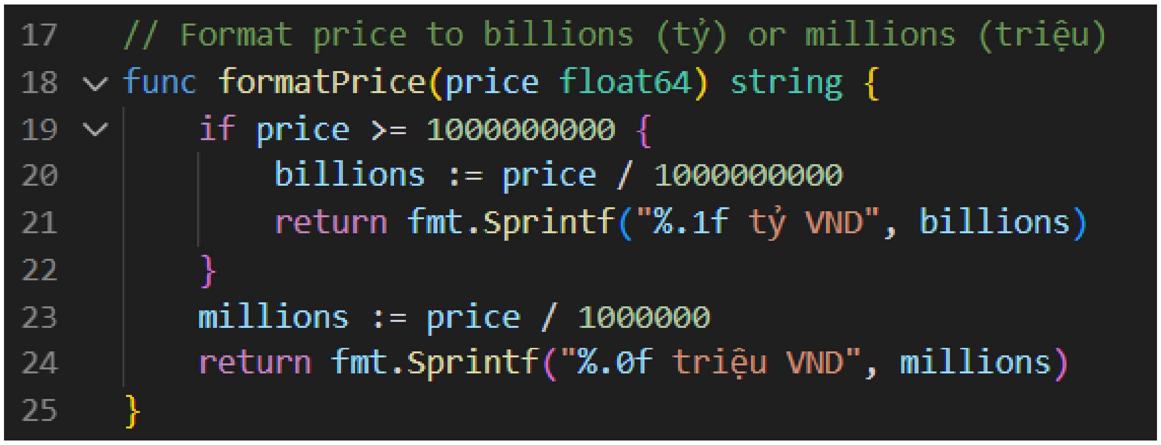
Task 1.2:

We add Function that assigns a category based on price per m²:



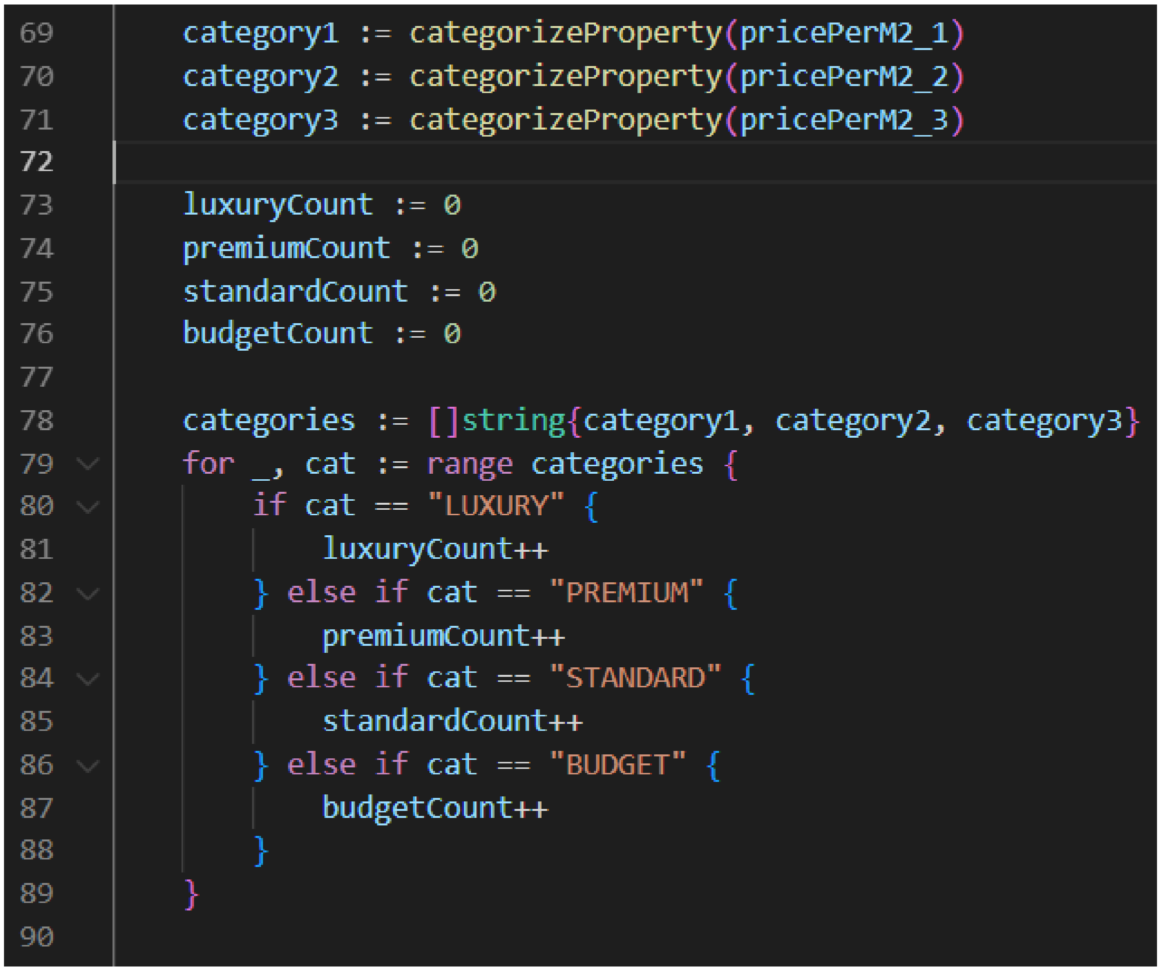
```go
// Categorize property based on price per m²
func categorizeProperty(pricePerM2 float64) string {
    if pricePerM2 > 50000000 {
        return "LUXURY"
    } else if pricePerM2 > 30000000 {
        return "PREMIUM"
    } else if pricePerM2 > 20000000 {
        return "STANDARD"
    }
    return "BUDGET"
}
```

Convert large VND numbers to readable format:



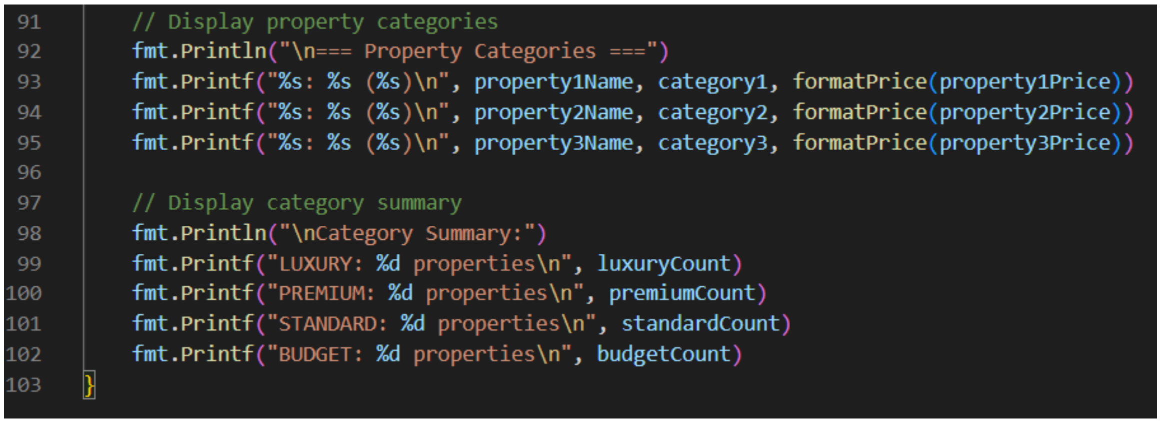
```go
// Format price to billions (tỷ) or millions (triệu)
func formatPrice(price float64) string {
    if price >= 1000000000 {
        billions := price / 1000000000
        return fmt.Sprintf("%.1f tỷ VND", billions)
    }
    millions := price / 1000000
    return fmt.Sprintf("%.0f triệu VND", millions)
}
```

Apply categorization to each property and count how many properties fall into each category.



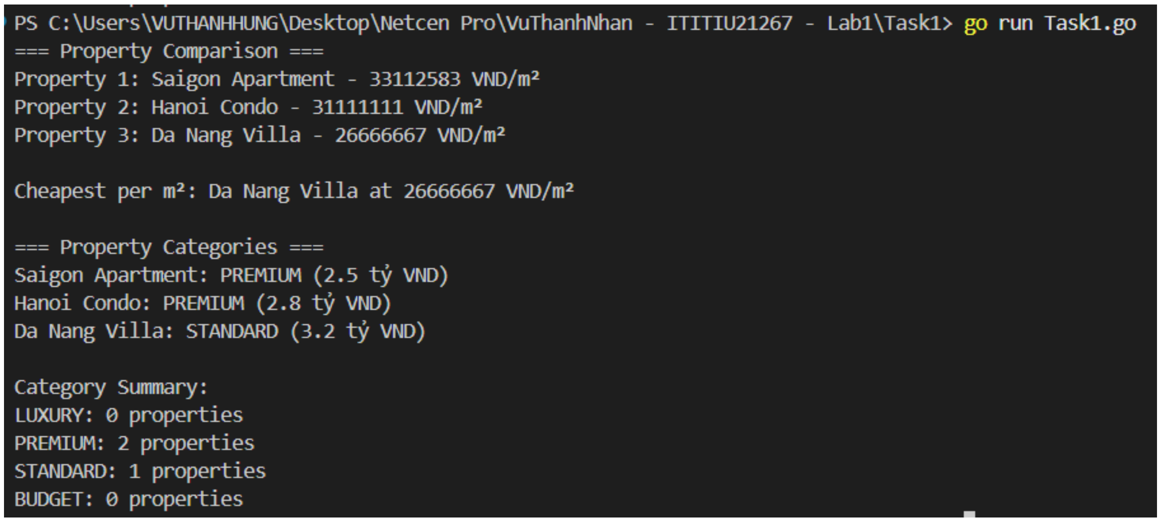
```go
    category1 := categorizeProperty(pricePerM2_1)
    category2 := categorizeProperty(pricePerM2_2)
    category3 := categorizeProperty(pricePerM2_3)

    luxuryCount := 0
    premiumCount := 0
    standardCount := 0
    budgetCount := 0

    categories := []string{category1, category2, category3}
    for _, cat := range categories {
        if cat == "LUXURY" {
            luxuryCount++
        } else if cat == "PREMIUM" {
            premiumCount++
        } else if cat == "STANDARD" {
            standardCount++
        } else if cat == "BUDGET" {
            budgetCount++
        }
    }

```

Display the result:



```go
    // Display property categories
    fmt.Println("\n=== Property Categories ===")
    fmt.Printf("%s: %s (%s)\n", property1Name, category1, formatPrice(property1Price))
    fmt.Printf("%s: %s (%s)\n", property2Name, category2, formatPrice(property2Price))
    fmt.Printf("%s: %s (%s)\n", property3Name, category3, formatPrice(property3Price))

    // Display category summary
    fmt.Println("\nCategory Summary:")
    fmt.Printf("LUXURY: %d properties\n", luxuryCount)
    fmt.Printf("PREMIUM: %d properties\n", premiumCount)
    fmt.Printf("STANDARD: %d properties\n", standardCount)
    fmt.Printf("BUDGET: %d properties\n", budgetCount)
}
```

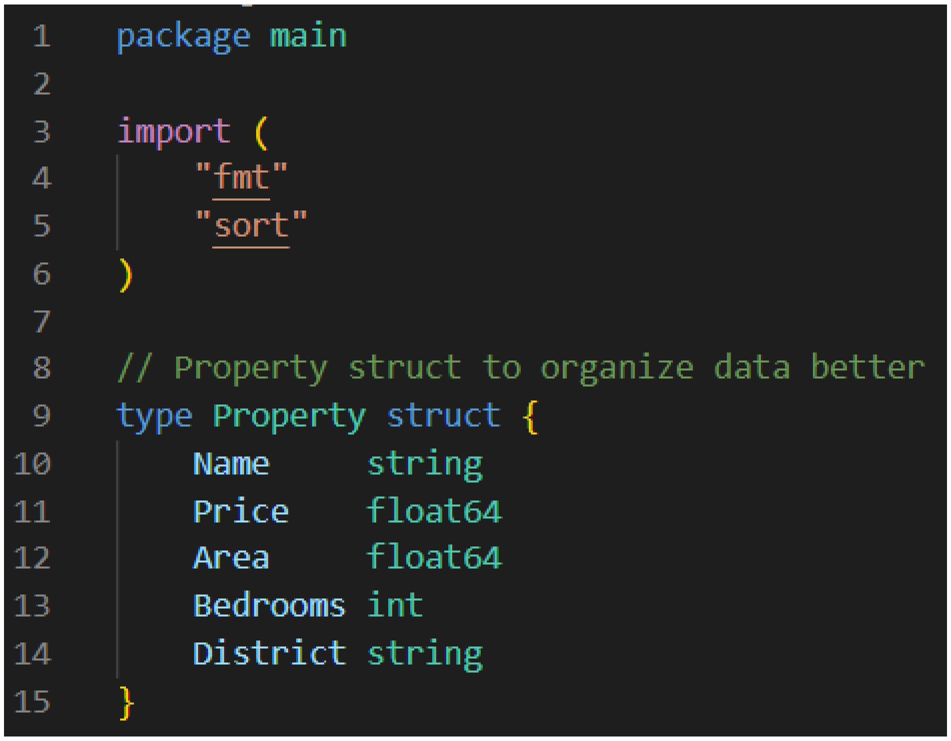
⇨ Result:



```
PS C:\Users\VUTHANHHUNG\Desktop\Netcen Pro\VuThanhNhan - ITITIU21267 - Lab1\Task1> go run Task1.go
=== Property Comparison ===
Property 1: Saigon Apartment - 33112583 VND/m²
Property 2: Hanoi Condo - 31111111 VND/m²
Property 3: Da Nang Villa - 26666667 VND/m²

Cheapest per m²: Da Nang Villa at 26666667 VND/m²

=== Property Categories ===
Saigon Apartment: PREMIUM (2.5 tỷ VND)
Hanoi Condo: PREMIUM (2.8 tỷ VND)
Da Nang Villa: STANDARD (3.2 tỷ VND)

Category Summary:
LUXURY: 0 properties
PREMIUM: 2 properties
STANDARD: 1 properties
BUDGET: 0 properties
```

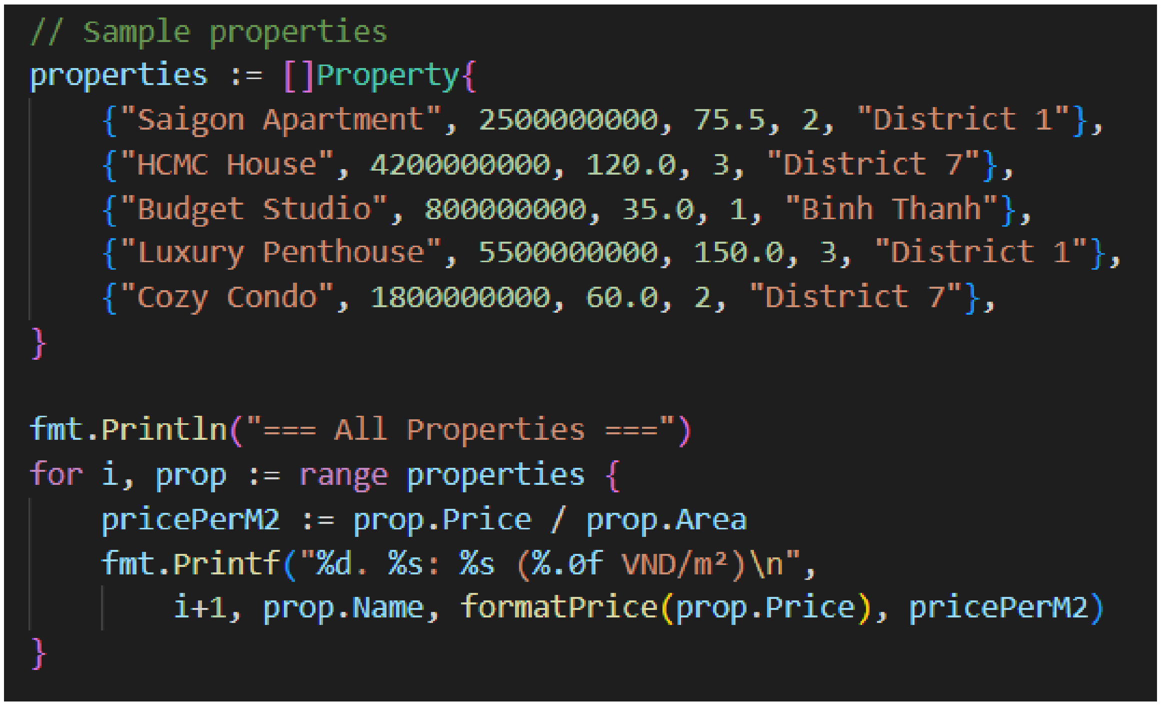
Part 2:

Property structure:



```go
package main

import (
    "fmt"
    "sort"
)

// Property struct to organize data better
type Property struct {
    Name     string
    Price    float64
    Area     float64
    Bedrooms int
    District string
}
```

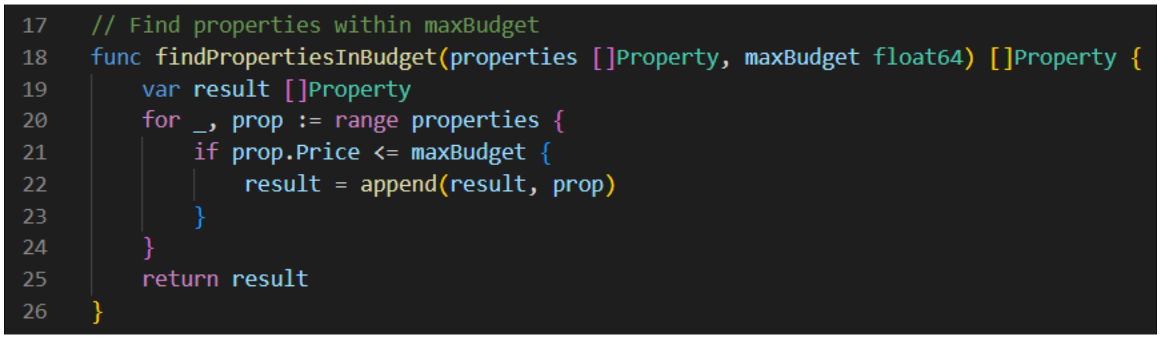
Sample properties:



```go
// Sample properties
properties := []Property{
    {"Saigon Apartment", 2500000000, 75.5, 2, "District 1"},
    {"HCMC House", 4200000000, 120.0, 3, "District 7"},
    {"Budget Studio", 800000000, 35.0, 1, "Binh Thanh"},
    {"Luxury Penthouse", 5500000000, 150.0, 3, "District 1"},
    {"Cozy Condo", 1800000000, 60.0, 2, "District 7"},
}

fmt.Println("=== All Properties ===")
for i, prop := range properties {
    pricePerM2 := prop.Price / prop.Area
    fmt.Printf("%d. %s: %s (%.0f VND/m²)\n",
        i+1, prop.Name, formatPrice(prop.Price), pricePerM2)
}
```

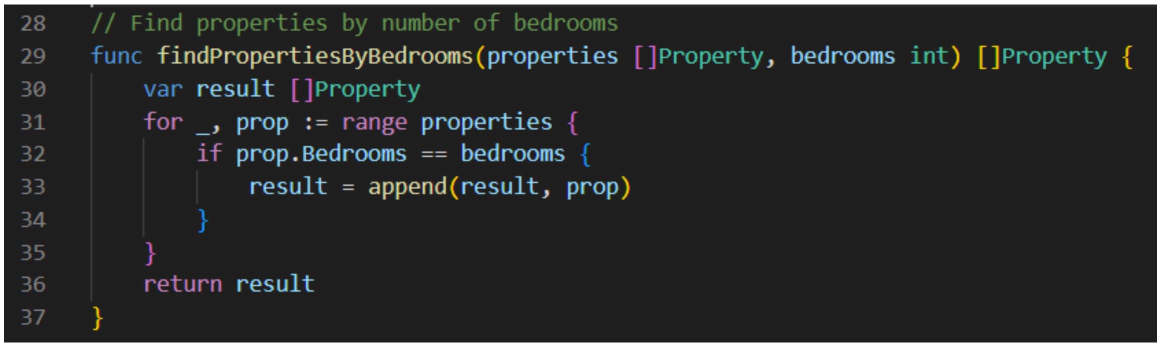
Task 2.1:

Implements a function that filters and returns all properties with prices less than or equal to a given maximum budget by iterating over the list and checking each price:



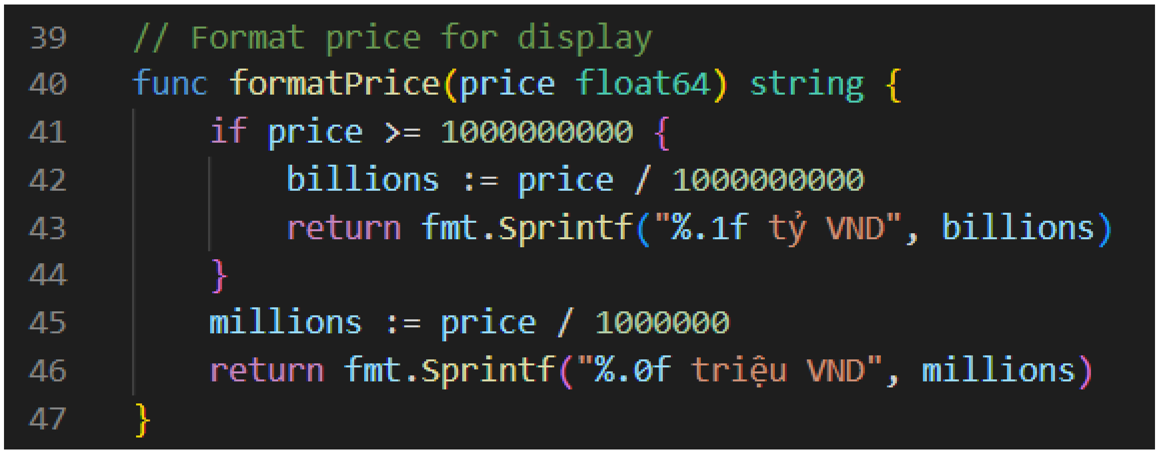
```go
// Find properties within maxBudget
func findPropertiesInBudget(properties []Property, maxBudget float64) []Property {
    var result []Property
    for _, prop := range properties {
        if prop.Price <= maxBudget {
            result = append(result, prop)
        }
    }
    return result
}
```

Function that finds and returns properties matching an exact number of bedrooms by looping through all properties and comparing the bedroom count:

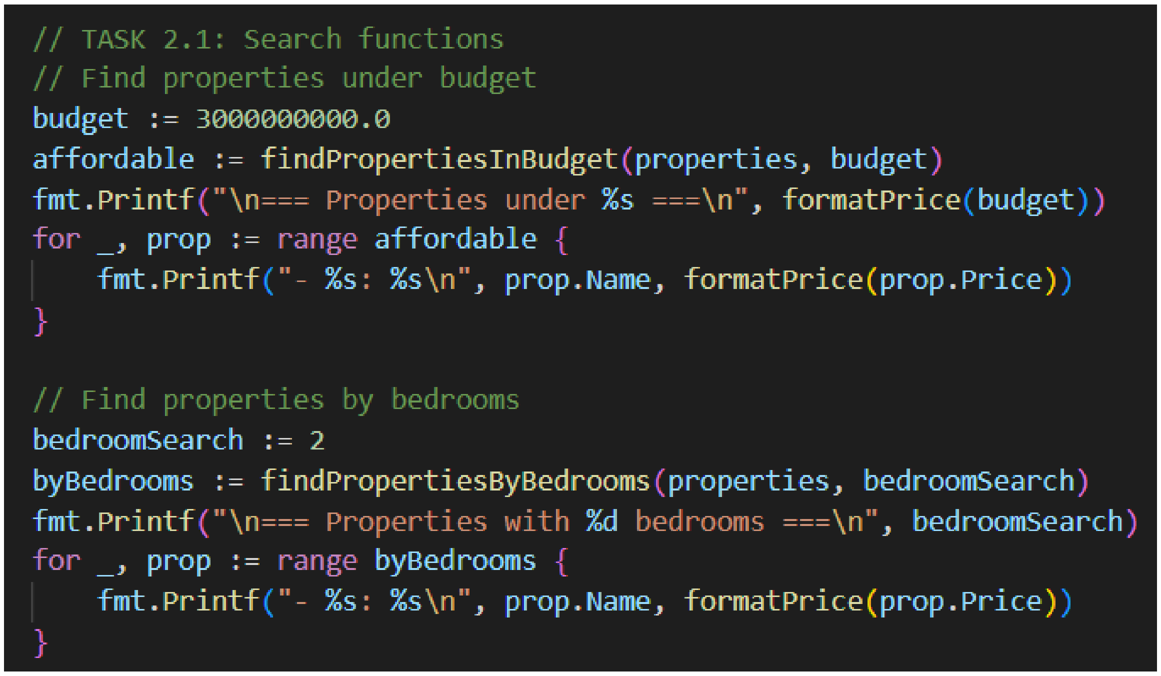


```go
// Find properties by number of bedrooms
func findPropertiesByBedrooms(properties []Property, bedrooms int) []Property {
    var result []Property
    for _, prop := range properties {
        if prop.Bedrooms == bedrooms {
            result = append(result, prop)
        }
    }
    return result
}
```

Defines a helper function to format property prices into readable strings, converting large values into billions or millions of Vietnamese Dong:

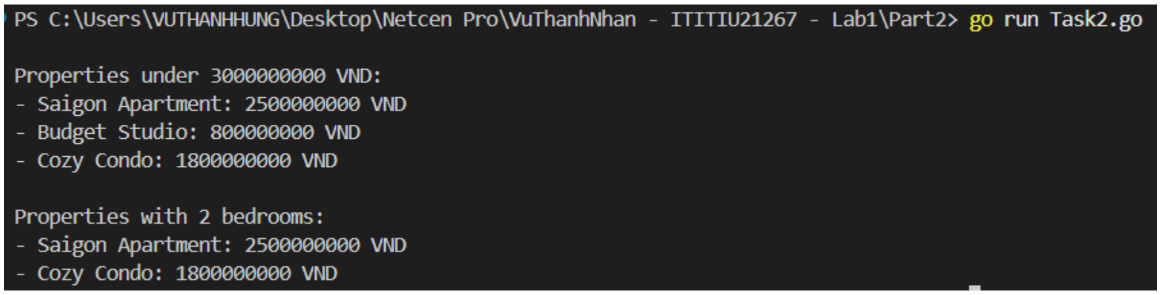


```go
// Format price for display
func formatPrice(price float64) string {
    if price >= 1000000000 {
        billions := price / 1000000000
        return fmt.Sprintf("%.1f tỷ VND", billions)
    }
    millions := price / 1000000
    return fmt.Sprintf("%.0f triệu VND", millions)
}
```



```go
// TASK 2.1: Search functions
// Find properties under budget
budget := 3000000000.0
affordable := findPropertiesInBudget(properties, budget)
fmt.Printf("\n=== Properties under %s ===\n", formatPrice(budget))
for _, prop := range affordable {
    fmt.Printf("- %s: %s\n", prop.Name, formatPrice(prop.Price))
}

// Find properties by bedrooms
bedroomSearch := 2
byBedrooms := findPropertiesByBedrooms(properties, bedroomSearch)
fmt.Printf("\n=== Properties with %d bedrooms ===\n", bedroomSearch)
for _, prop := range byBedrooms {
    fmt.Printf("- %s: %s\n", prop.Name, formatPrice(prop.Price))
}
```

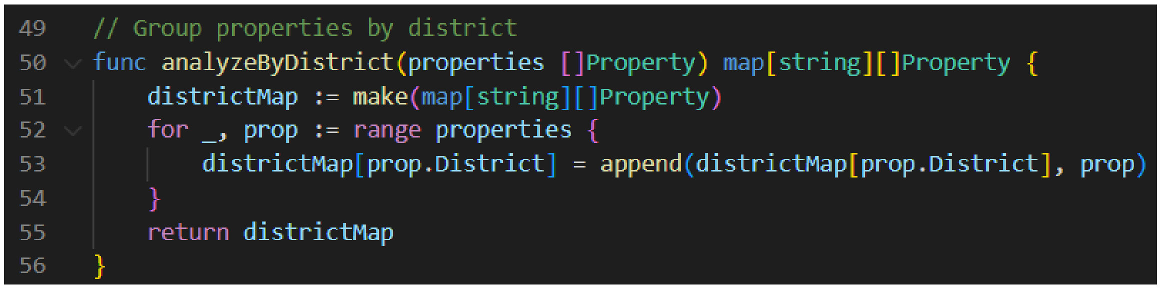
Result:



```
PS C:\Users\VUTHANHHUNG\Desktop\Netcen Pro\VuThanhNhan - ITITIU21267 - Lab1\Part2> go run Task2.go

Properties under 3000000000 VND:
- Saigon Apartment: 2500000000 VND
- Budget Studio: 800000000 VND
- Cozy Condo: 1800000000 VND

Properties with 2 bedrooms:
- Saigon Apartment: 2500000000 VND
- Cozy Condo: 1800000000 VND
```

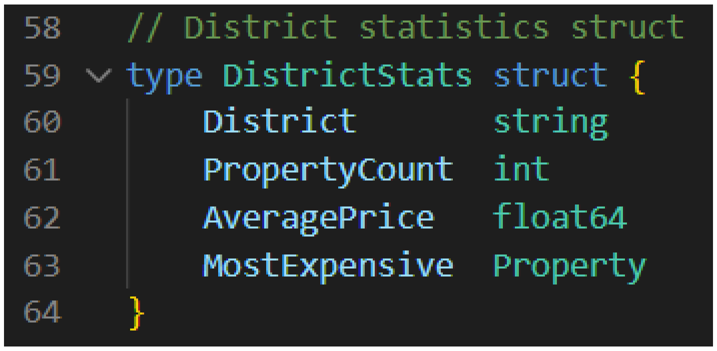
Task 2.2:

Groups properties by their district using a map where each key is a district name and the value is a list of properties in that district.



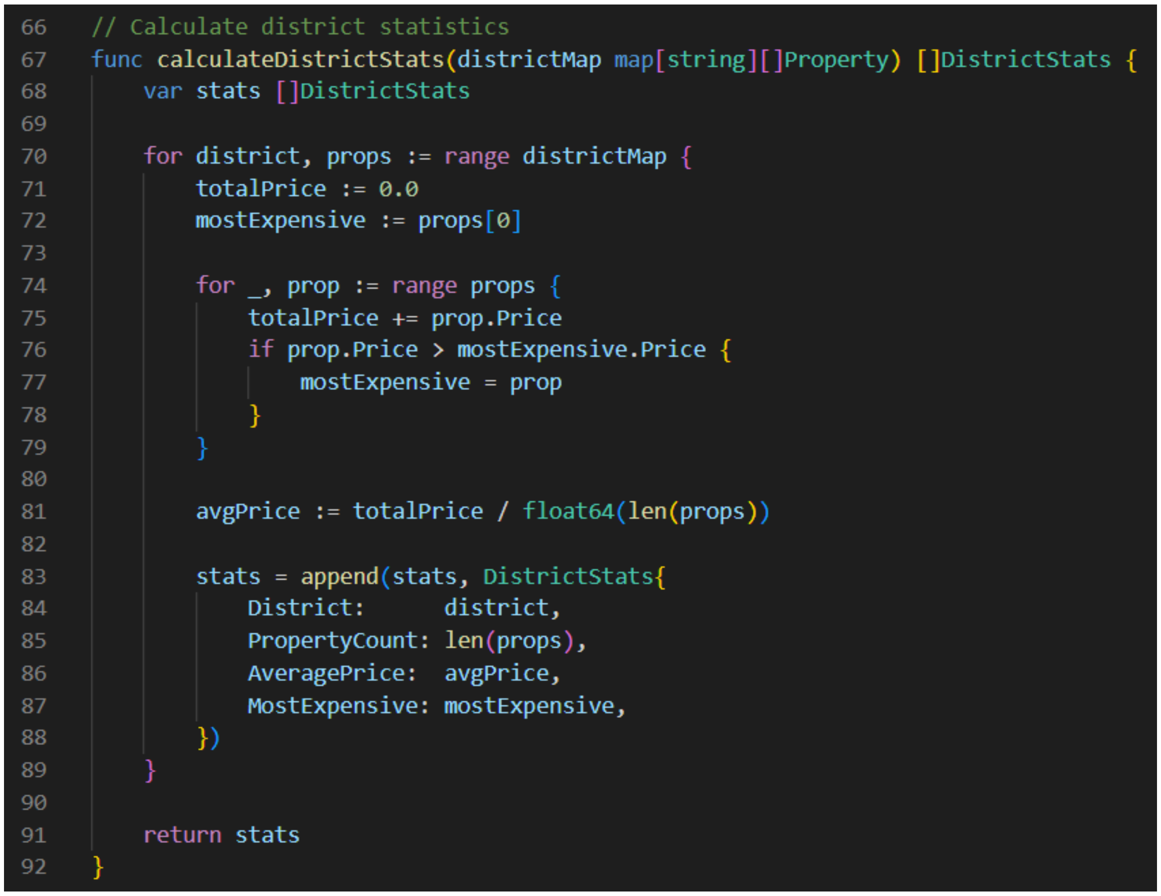
```go
// Group properties by district
func analyzeByDistrict(properties []Property) map[string][]Property {
    districtMap := make(map[string][]Property)
    for _, prop := range properties {
        districtMap[prop.District] = append(districtMap[prop.District], prop)
    }
    return districtMap
}
```

Introduces a DistrictStats struct to hold statistics:

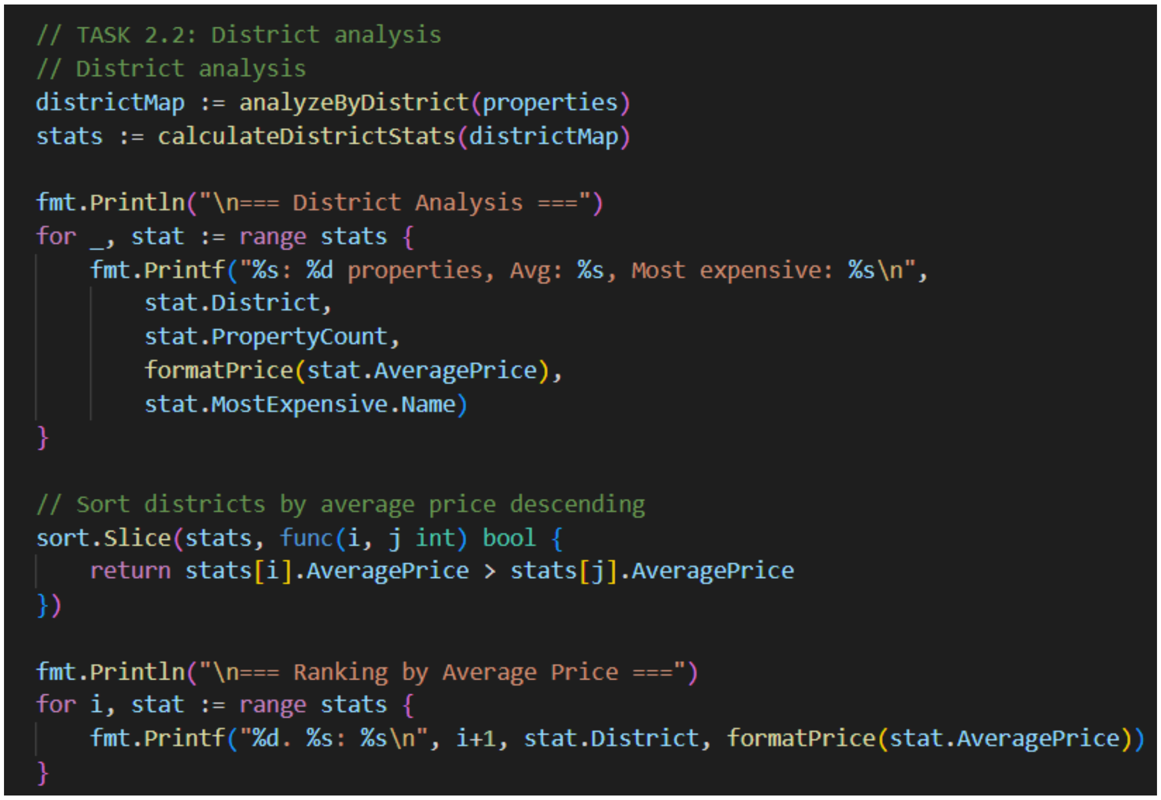


```go
// District statistics struct
type DistrictStats struct {
	District      string
	PropertyCount int
	AveragePrice  float64
	MostExpensive Property
}
```

Calculates these statistics by iterating over each district's properties, summing prices, finding the highest price, computing the average, and returning the results as a slice of DistrictStats:

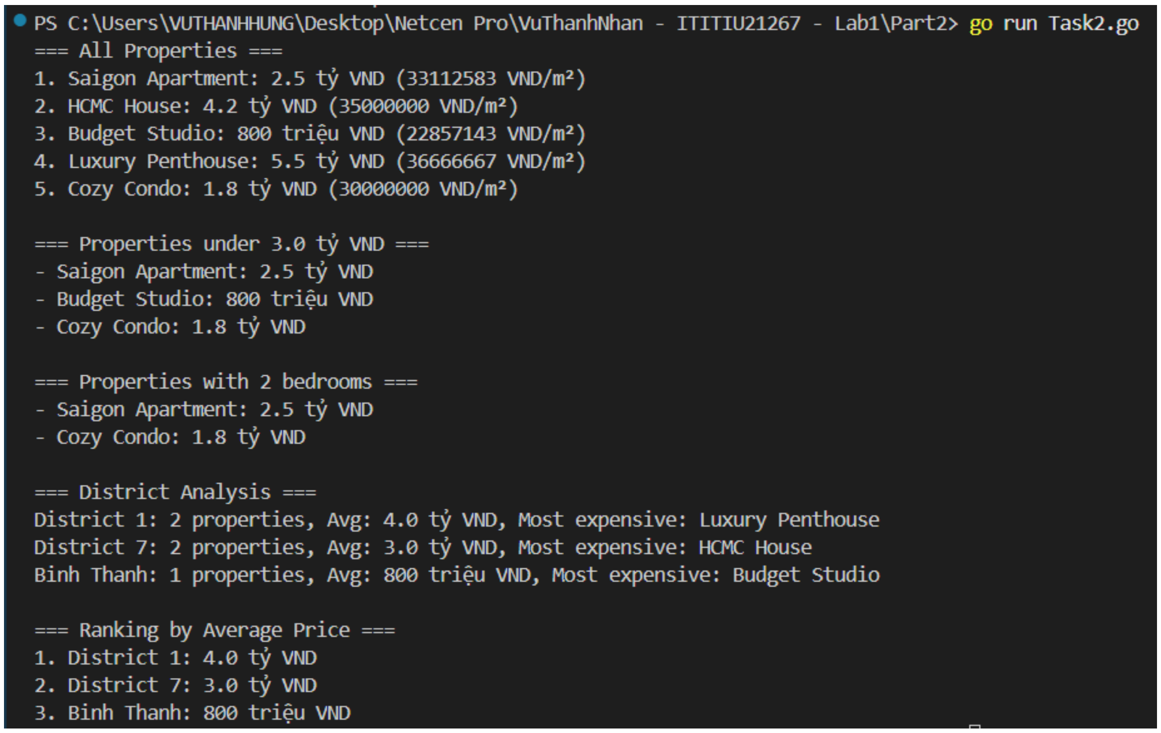


```go
// Calculate district statistics
func calculateDistrictStats(districtMap map[string][]Property) []DistrictStats {
	var stats []DistrictStats

	for district, props := range districtMap {
		totalPrice := 0.0
		mostExpensive := props[0]

		for _, prop := range props {
			totalPrice += prop.Price
			if prop.Price > mostExpensive.Price {
				mostExpensive = prop
			}
		}

		avgPrice := totalPrice / float64(len(props))

		stats = append(stats, DistrictStats{
			District:      district,
			PropertyCount: len(props),
			AveragePrice:  avgPrice,
			MostExpensive: mostExpensive,
		})
	}

	return stats
}
```



```go
// TASK 2.2: District analysis
// District analysis
districtMap := analyzeByDistrict(properties)
stats := calculateDistrictStats(districtMap)

fmt.Println("\n=== District Analysis ===")
for _, stat := range stats {
    fmt.Printf("%s: %d properties, Avg: %s, Most expensive: %s\n",
        stat.District,
        stat.PropertyCount,
        formatPrice(stat.AveragePrice),
        stat.MostExpensive.Name)
}

// Sort districts by average price descending
sort.Slice(stats, func(i, j int) bool {
    return stats[i].AveragePrice > stats[j].AveragePrice
})

fmt.Println("\n=== Ranking by Average Price ===")
for i, stat := range stats {
    fmt.Printf("%d. %s: %s\n", i+1, stat.District, formatPrice(stat.AveragePrice))
}
```

⇨ Final result:



```
PS C:\Users\VUTHANHHUNG\Desktop\Netcen Pro\VuThanhNhan - ITITIU21267 - Lab1\Part2> go run Task2.go
=== All Properties ===
1. Saigon Apartment: 2.5 tỷ VND (33112583 VND/m²)
2. HCMC House: 4.2 tỷ VND (35000000 VND/m²)
3. Budget Studio: 800 triệu VND (22857143 VND/m²)
4. Luxury Penthouse: 5.5 tỷ VND (36666667 VND/m²)
5. Cozy Condo: 1.8 tỷ VND (30000000 VND/m²)

=== Properties under 3.0 tỷ VND ===
- Saigon Apartment: 2.5 tỷ VND
- Budget Studio: 800 triệu VND
- Cozy Condo: 1.8 tỷ VND

=== Properties with 2 bedrooms ===
- Saigon Apartment: 2.5 tỷ VND
- Cozy Condo: 1.8 tỷ VND

=== District Analysis ===
District 1: 2 properties, Avg: 4.0 tỷ VND, Most expensive: Luxury Penthouse
District 7: 2 properties, Avg: 3.0 tỷ VND, Most expensive: HCMC House
Binh Thanh: 1 properties, Avg: 800 triệu VND, Most expensive: Budget Studio

=== Ranking by Average Price ===
1. District 1: 4.0 tỷ VND
2. District 7: 3.0 tỷ VND
3. Binh Thanh: 800 triệu VND
```