```
package retailers

import (
        "fmt"
        "github.com/pkg/errors"
        "io/ioutil"
        "net/http"
        "net/url"
        "strings"
)

const (
        PC_URL_PREFIX = "https://poisoncityestore.com"
        PC_SEARCH_URL = "https://poisoncityestore.com/search?q=%s+vinyl"
)

type PoisonCity struct{}

func (a *PoisonCity) GetArtistQueryURL(artist string) string {
        query := fmt.Sprintf(PC_SEARCH_URL, url.QueryEscape(artist))
        return query
}

func (a *PoisonCity) ScrapeArtistReleases(artist string) (findings []SKU, err error) {

        findings = []SKU{}
        query := a.GetArtistQueryURL(artist)
        resp, err := http.Get(query)
        if err != nil {
                return nil, errors.Wrapf(err, "failed to retrieve search query %s", query)
        }
        body, err := ioutil.ReadAll(resp.Body)
        if err != nil {
                return nil, errors.Wrapf(err, "failed to extract body of search results %s", query)
        }
        toks := strings.Split(string(body), ">")

        for idx, t := range toks {
                if strings.Index(t, "row results") >= 0 {
                        sku := SKU{}
                        subUrl := strings.TrimPrefix(strings.TrimSpace(toks[idx+3]), "<a href=\"")
                        subUrl = PC_URL_PREFIX + strings.TrimSuffix(subUrl, "\"")

                        // now read THAT url...
                        query := fmt.Sprintf(subUrl)
                        resp, err := http.Get(query)
                        if err != nil {
                                fmt.Printf("[FAILED: %s]..", err.Error())
                                return nil, errors.Wrapf(err, "failed to open sub url %s", subUrl)
                        }
                        body, err := ioutil.ReadAll(resp.Body)
                        if err != nil {
                                fmt.Printf("[FAILED: %s]..", err.Error())
                                return nil, errors.Wrapf(err, "failed to open body of sub url %s",
subUrl)
                        }

                        subToks := strings.Split(string(body), ">")
                        for idx, tok := range subToks {
                                if strings.Index(tok, "itemprop=\"image\"") > 0 {
                                        sku.Image = "https:" +
strings.TrimPrefix(strings.TrimSpace(tok), "<meta itemprop=\"image\" content=\"")
                                        sku.Image = strings.TrimSuffix(sku.Image, "\" /")
                                }
                                if strings.Index(tok, "itemprop=\"name\"") >= 0 {
                                        sku.Name = strings.TrimSpace(subToks[idx+1])
                                        sku.Name = strings.TrimSuffix(sku.Name, "</h1")
                                }
                                if strings.Index(tok, "itemprop=\"price\"") >= 0 {
                                        sku.Price = strings.TrimSpace(subToks[idx+1])
                                        sku.Price = strings.TrimSuffix(sku.Price, "</span")
                                }
                                if strings.Index(tok, "\"Sold Out\"") >= 0 {
```

```
                                sku.Price = "sold out"
                        }
                }

                if !strings.HasPrefix(strings.ToLower(sku.Name), strings.ToLower(artist)) {
                        continue
                }

                sku.Image = fmt.Sprintf("<img width=\"150px\" height=\"150px\"
src=\"%s\">", sku.Image)

                // store new price
                findings = append(findings, sku)
        }
    }
    return findings, nil
}
```