> Really don't understand how to implement Proposition 12 and could use an example. How do I efficiently compute the partial derivatives of f and then also efficiently compute the gcd of all of them?

I've added some sections to our Sage reference about computing partial derivatives (using the `gradient` function) and gcds and reductions (there's a custom `reduction` function defined there). See the sections titled "Multivariable polynomials: Partial derivatives" and "Multivariable polynomials: GCDs, factoring, and reduction."

So, for example, if you wanted to compute $\sqrt{I}$ where $I = \langle (x+y^2)^3(x-y) \rangle$, you might do the following:

```
R.<x,y> = PolynomialRing(QQ, order='lex')
f = (x+y^2)^3*(x-y)
reduction(f)
```

The output of this is `x^2 + x*y^2 - x*y - y^3`, and this is a generator for $\sqrt{I}$.

Note though that the above only works for principal ideals. That being said, Sage can compute radicals of non-principal ideals as well, using the `radical` function. For example, if you want to compute a Gröbner basis for $\sqrt{I}$ when $I = \langle x^2+y^2-1, y-1 \rangle$, you can do the following:

```
R.<x,y> = PolynomialRing(QQ, order='deglex')
I = Ideal(x^2+y^2-1,y-1)
I.radical().groebner_basis()
```

The output is `[x, y - 1]`.