AgGroups by pc-presentations in GAP:

```
gap> F := FreeGroup( 3 );
Group( f.1, f.2, f.3 )

gap> f := F.generators;
[ f.1, f.2, f.3 ]

gap> rels := [f[1]^3,
>             f[2]^2,
>             f[3]^2,
>             Comm(f[2],f[1])/f[3],
>             Comm(f[3], f[1])/f[2],
>             Comm(f[3],f[2])];
[ f.1^3, f.2^2, f.3^2,
  f.2^-1*f.1^-1*f.2*f.1*f.3^-1,
  f.3^-1*f.1^-1*f.3*f.1*f.2^-1,
  f.3^-1*f.2^-1*f.3*f.2 ]

gap> H := F/rels;
Group( f.1, f.2, f.3 )

gap> G := AgGroupFpGroup( H );
Group( f.1, f.2, f.3 )

gap> IsConsistent(G);
false
```

2. Try:

```
gap> rels := [f[1]^3,
>              f[2]^2,
>              f[3]^2,
>              Comm(f[2],f[1])/f[3],
>              Comm(f[3], f[1])/(f[2]*f[3]),
>              Comm(f[3],f[2])];;

gap> H := F/rels;
Group( f.1, f.2, f.3 )

gap> G := AgGroupFpGroup( H );
Group( f.1, f.2, f.3 )

gap> IsConsistent(G);
true
```

AgGroups constructed from permutation groups:

```
gap> G := TransitiveGroup( 9, 10 );
[3^2]S(3)_6

gap> G.generators;
[ (1,4,7)(2,8,5), (1,2,3,4,5,6,7,8,9), (1,8)(2,7)(3,6)(4,5) ]

gap> IsSolvable(G);
true

gap> A := AgGroup( G );
Group( g1, g2, g3, g4 )

gap> A.bijection;
GroupHomomorphismByImages(
    Group( g1, g2, g3, g4 ),
    [3^2]S(3)_6,
    [ g1, g2, g3, g4 ],
    [ (1,8)(2,7)(3,6)(4,5), (1,2,3,4,5,6,7,8,9),
      (1,4,7)(2,8,5), (2,5,8)(3,9,6) ] )
```

AgGroups constructed from matrix groups:

```
gap> G := IrreducibleSolvableGroup( 5, 3, 9 );;

gap> Length(G.generators);
6

gap> G.generators[1];
[ [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
  [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3) ] ]

gap> A := AgGroup( G );
Group( g1, g2, g3, g4, g5, g6 )

gap> Size(A);
160

gap> min := MinimalGeneratingSet( A );
[ g2, g1*g3 ]

gap> List( min, x -> Image( A.bijection, x ) );
[ [ [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ],
      [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3) ],
      [ Z(3), 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
      [ 0*Z(3), Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
      [ 0*Z(3), 0*Z(3), Z(3), 0*Z(3), 0*Z(3) ] ],
  [ [ 0*Z(3), 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3) ],
      [ 0*Z(3), 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
      [ 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3) ],
      [ Z(3)^0, 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3) ],
      [ 0*Z(3), 0*Z(3), 0*Z(3), 0*Z(3), Z(3) ] ] ]
```

AgGroup does not work for f.p. groups:

```
gap> IsFpGroup( H );
true

gap> AgGroup(H);
Error, this function will not work for finitely
presented groups.

Please use 'OperationCosetsFpGroup' in order to
construct an isomorphic permutation group in

G.operations.Factorization( G, g ) called from
Factorization( Q[i + 1], B[i] ^ BP[i] ) called from
G.operations.AgGroup( G ) called from
AgGroup( H ) called from
main loop

brk> quit;
gap>
```

The small group catalogue in GAP:

```
gap> NumberSmallGroups( 192 );
1543

gap> G := SmallGroup( 192, 1500 );
192_1500

gap> G.generators;
[ a, b, c, d, e, f, g ]

gap> IsAgGroup(G);
true
```

Find the catalogue number of a group:

```
gap> G := TransitiveGroup( 9, 10 );
[3^2]S(3)_6

gap> IdGroup( G );
[ 54, 6 ]
```

The catalogue of solvable groups in GAP:

```
gap> G := SolvableGroup( 64, 10 );
8^2

gap> G.generators;
[ a, b, c, d, e, f ]

gap> IsAgGroup( G );
true

gap> GroupId(G);
rec(
  catalogue := [ 64, 10 ],
  names := [ "8^2" ],
  size := 64,
  abelianInvariants := [ 8, 8 ],
  pGroupId := 2 )

gap> IdGroup( G );
[ 64, 2 ]
```

A basic construction function:

```
gap> G := SymmetricGroup( AgWords, 4 );
Group( g1, g2, g3, g4 )

gap> IdGroup(G);
[ 24, 12 ]
```

Direct products, semidirect products and wreath products:

```
gap> G := SmallGroup( 24, 12 );
24_12

gap> D := DirectProduct( G, G );
Group( a1, a2, a3, a4, b1, b2, b3, b4 )


gap> mods := ModulesAgGroup( G, GF(2), 2 );;
gap> mats := mods[2].generators;;

gap> M := Group( mats, mats[1]^0 );
Group( [ [ Z(2)^0, Z(2)^0 ], [ 0*Z(2), Z(2)^0 ] ],
       [ [ 0*Z(2), Z(2)^0 ], [ Z(2)^0, Z(2)^0 ] ] )

gap> hom := GroupHomomorphismByImages(
            G, M, G.generators, mats );;

gap> S := SemidirectMatProduct( G, hom );
Group( g1, g2, g3, g4, h1, h2 )

gap> Size(S);
96
```

The PQuotient in GAP:

```
gap> F := FreeGroup( 2 );
Group( f.1, f.2 )

gap> f := F.generators;
[ f.1, f.2 ]

gap> rels := [f[1]^3 / (f[1]*f[2])^3,
              Comm(f[1], f[2])^3,
>            (f[1]^f[2])^9];
[ f.1^3*f.2^-1*f.1^-1*f.2^-1*f.1^-1*f.2^-1*f.1^-1,
  f.1^-1*f.2^-1*f.1*f.2*f.1^-1*f.2^-1*f.1*f.2*\
  f.1^-1*f.2^-1*f.1*f.2,
  f.2^-1*f.1^9*f.2 ]

gap> H := F / rels;
Group( f.1, f.2 )

gap> AbelianInvariants(H);
[ 3, 9 ]
```

```
gap> PQuotient(H, 3, 10 );;
#I  PQuotient: class 1 : 2
#I  PQuotient: Runtime : 0
#I  PQuotient: class 2 : 2
#I  PQuotient: Runtime : 10
#I  PQuotient: class 3 : 2
#I  PQuotient: Runtime : 30
#I  PQuotient: class 4 : 1
#I  PQuotient: Runtime : 70
#I  PQuotient: class 5 : 2
#I  PQuotient: Runtime : 110
#I  PQuotient: class 6 : 2
#I  PQuotient: Runtime : 200
#I  PQuotient: class 7 : 4
#I  PQuotient: Runtime : 300
#I  PQuotient: class 8 : 3
#I  PQuotient: Runtime : 460
#I  PQuotient: class 9 : 6
#I  PQuotient: Runtime : 680
#I  PQuotient: class 10 : 7
#I  PQuotient: Runtime : 1120

gap> A := AgGroup( last );
Group( g1, g2, a3, a4, a6, a7, a12, a15, a16, a18, a19,
       a21, a22, a23, a24, a26, a27, a30, a33, a34, a35,
       a36, a37, a38, a40, a41, a44, a45, a46, a47, a48 )

gap> Collected(Factors(Size(A)));
[ [ 3, 31 ] ]
```

The GAP SolvableQuotient:

```
gap> H;
Group( a, b, c, d )

gap> H.relators;
[ a^2, c^3, b^2*d^2, d*b*d*b, d^4, c^-1*b^-1*c*b,
  c^-1*a*c*a, b^-1*a*b*a, d^-1*a*d*a, d^-1*c*d*c*d^-1*c,
  c^-1*d^-1*c^-1*d^-1*c^-1*d^-1 ]

gap> AbelianInvariants(H);
[ 2, 2, 3 ]

gap> SolvableQuotient( H, [2,3] );
#I  commutator group = 2^2*3
#I  trying    = [ 2, 0 ]
#I  quotient = 2^5*3
#I  trying    = [ 3, 0 ]
#I  quotient = 2^5*3
Group( e1, e2, e3, m4, m5, m6 )

gap> A := AgGroupFpGroup( last );

gap> Size( A );
96
```

The PCQA - polycyclic quotient:

```
gap> RequirePackage("pcqa");

gap> F := FreeGroup( 2 );;
gap> f := F.generators;;

gap> rels := [f[1]*f[2]*f[1]*f[2]^-1*f[1]^-1*f[2]^-1];
[ f.1*f.2*f.1*f.2^-1*f.1^-1*f.2^-1 ]

gap> H := F/rels;
Group( f.1, f.2 )

gap> C := CallPCQA( H, 3 );;

gap> AbelianComponent(C);
[ [ 0 ], [ 0, 0 ] ]

gap> HirschLength(C.PolycyclicPresentation);
3

gap> C.PolycyclicPresentation;
rec(
  Generators := 3,
  ExponentList := [ 0, 0, 0 ],
  ppRelations := [ [ [0,1,-1], [0,1,0] ], [ [0,0,1] ] ],
  pnRelations := [ [ [0,-1,1], [0,-1,0] ], [ [0,0,-1] ] ],
  npRelations := [ [ [0,0,1], [0,-1,1] ], [ [0,0,1] ] ],
  nnRelations := [ [ [0,0,-1], [0,1,-1] ], [ [0,0,-1] ] ],
  PowerRelations := [  ] )
```

Try some slightly different relators:

```
gap> rels := [f[1]*f[2]*f[1]*f[2]^-1*f[1]^-1*f[2]^-1,
              f[2]^7];;

gap> H := F/rels;
Group( f.1, f.2 )

gap> C := CallPCQA( H, 3 );;

gap> AbelianComponent(C);
[ [ 7 ] ]

gap> HirschLength(C.PolycyclicPresentation);
0

gap> C.PolycyclicPresentation;
rec(
  Generators := 1,
  ExponentList := [ 7 ],
  ppRelations := [  ],
  pnRelations := [  ],
  npRelations := [  ],
  nnRelations := [  ],
  PowerRelations := [ [ 0 ], [ 6 ] ] )
```

You cannot convert it to an Ag-Group!

The NQ - nilpotent quotient:

```
gap> RequirePackage("nq");

gap> F := FreeGroup( 2 );
Group( f.1, f.2 )

gap> f := F.generators;
[ f.1, f.2 ]

gap> rels := [f[1]^3/(f[1]*f[2])^9, Comm(f[1], f[2])^3];;

gap> H := F/rels;
Group( f.1, f.2 )

gap> NilpotentQuotient( H, 6 );
[ [ 3, 0 ], [ 3 ], [ 3, 3 ], [ 3, 3 ], [ 3, 3, 3, 3 ],
  [ 3, 3, 3, 3, 3 ] ]
```