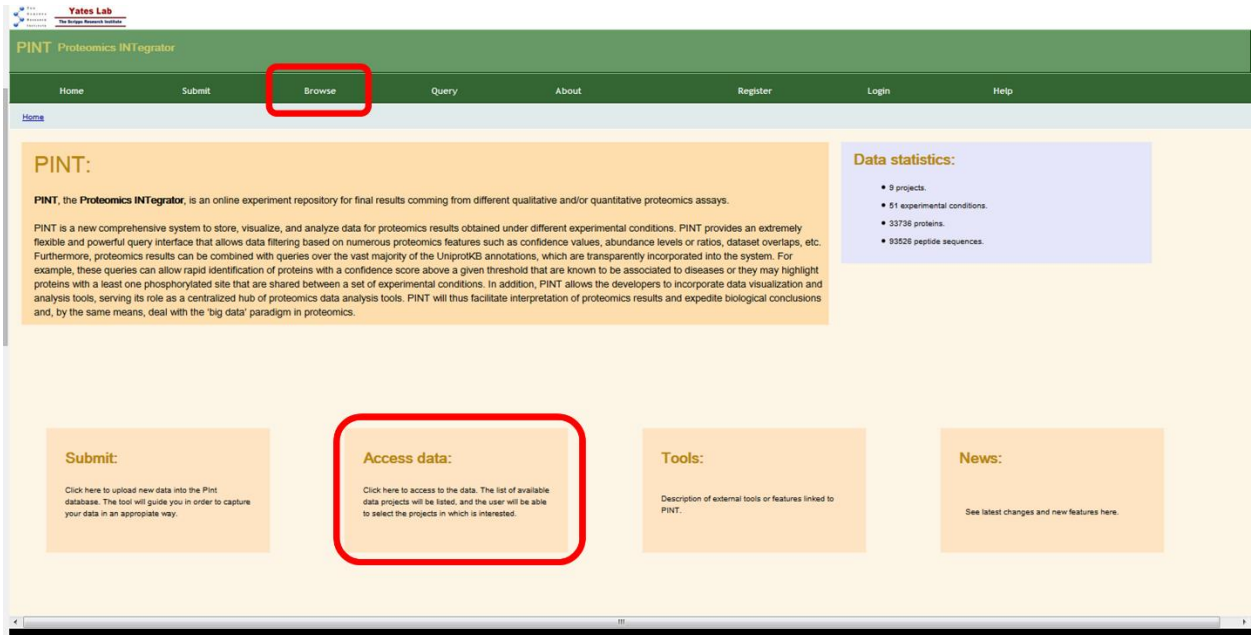
Commands for query data in DB

PINT provides a powerful data query / filter system on the data stored in the system. For performing any query or filtering, you will have to load the project or projects from the “Browse” menu option or by clicking on “Access data” square at the main page:



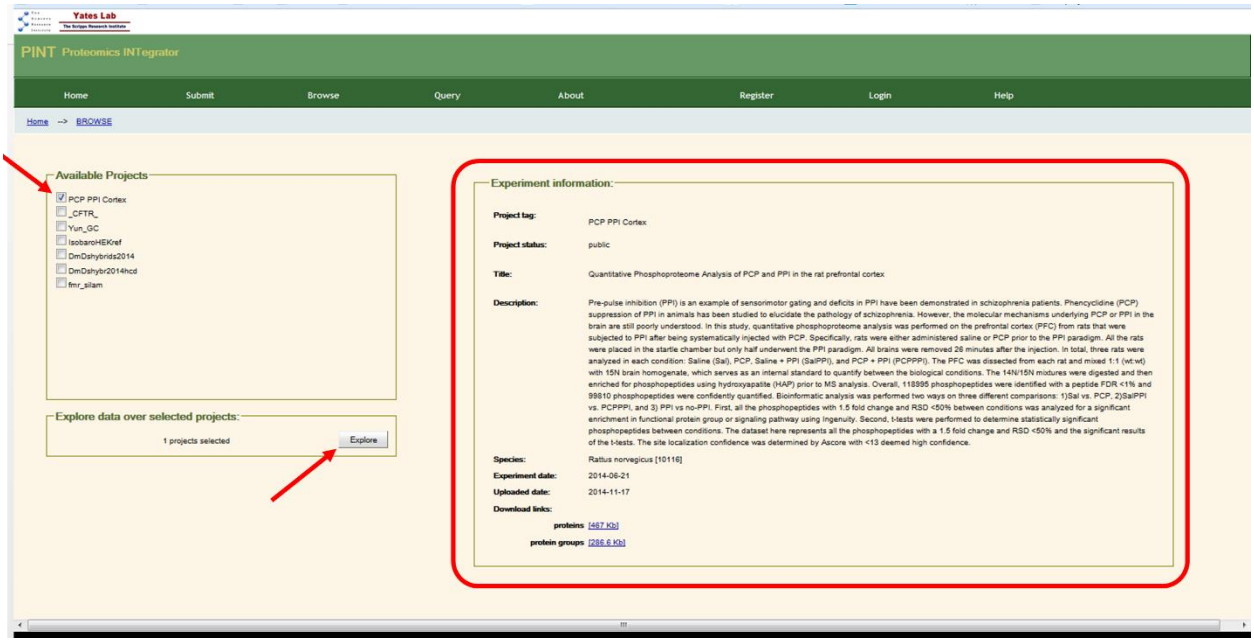
Then, you will have to select the desired project from the list of projects. Note that by positioning the mouse over each project name, the associated information will be showed at the right part of the page. Note that by default, the projects that are uploaded to PINT will be private, which means that they will be accessible through this page.

***Note**: By default all projects will be uploaded to the system as **private**, so they will be not available through the “Browse” page. In order to make them public available, modify the database entry using the following SQL command:*

update Project set private = 0 where project_tag = “your_project_tag”

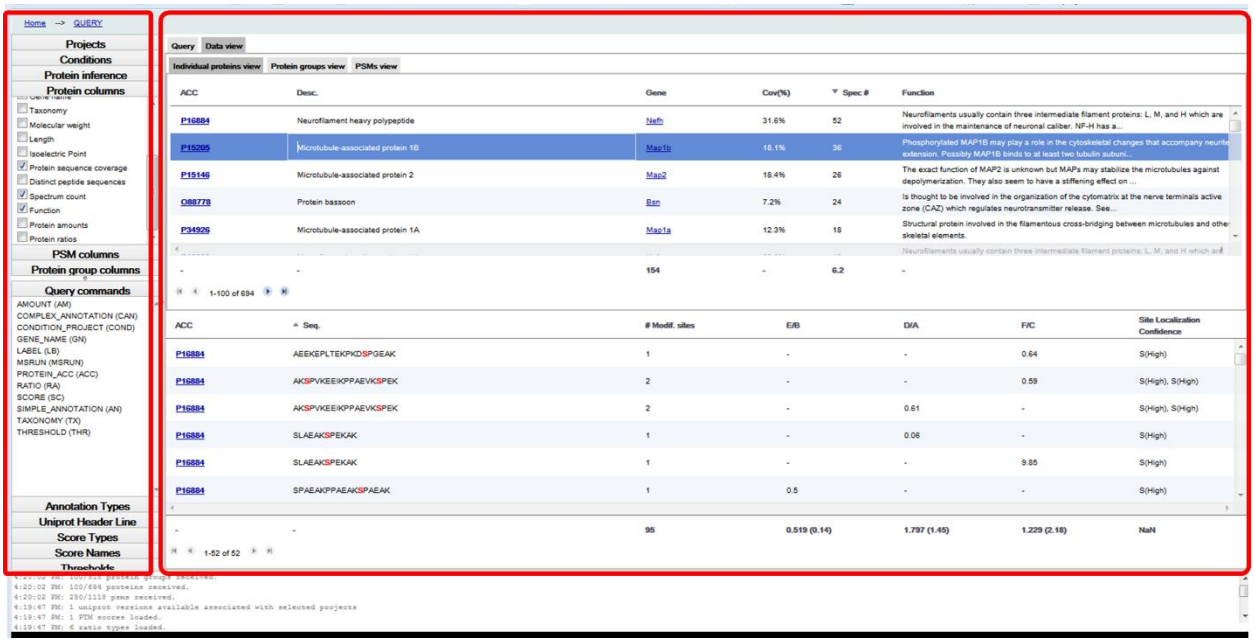
In order to access to a private project, you will have to use a direct unique link to it, which will be created at the time it is saved into the database. You will find the link in the email sent after the submission. The link should be something like this:

http://yourserver/pint/?project=03fg4aa7791e28e4



It is possible to select more than one project in order to load all the data of them. All the information will be integrated in a single data view.

Once you select the project(s), you should click on “Explore” button (or enter in a direct link to the project), and the data view page will be loaded:

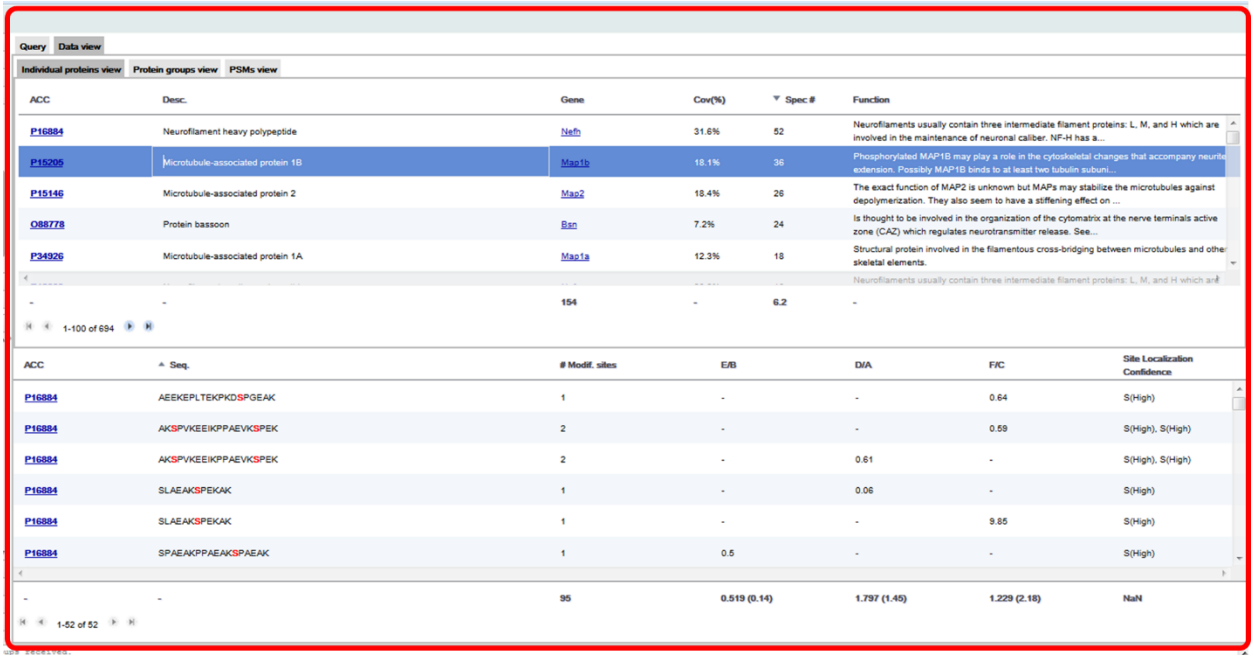


The data viewer page is divided into two parts. The left part which contains all the data related to the project, such as experimental condition names, score names and score type names associated with the data, as well as data visualization options such as the names of the columns of the different tables (used for show or hide the columns), and other name lists containing text that can be used in the different query/filtering commands. The right part contains different tabs:

Data view tab

In the "Data view" you will see three different sub-tabs, one of each corresponding to one level of aggregation: "Individual proteins view", "Protein groups view" and "PSMs view". By selecting each one of them you will see the corresponding data table, and in case of "Individual proteins view", "Protein groups view", you will be able to see also a PSM table containing the PSMs of the protein or protein-group selected.

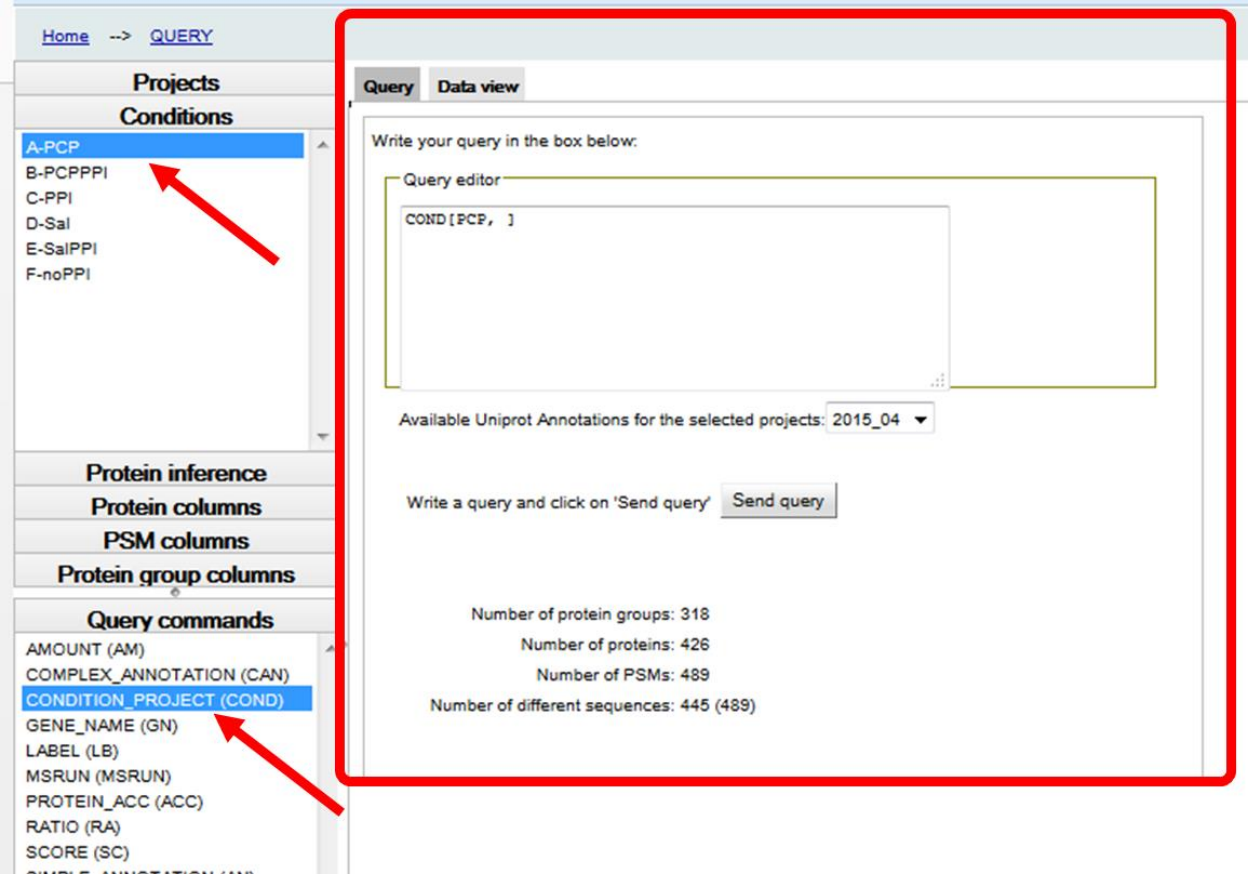
By clicking in the name of any column of the tables, you will be able to sort the data accordingly. If you click twice, the data will be sorted in the opposite direction.



Query tab

In the "Query" tab, you will be able to write a logical combination of query commands and execute it by clicking on "Send query" button. Below that button, you will see a summary of the current dataset numbers (number of protein groups, number of proteins, number of PSMs, number of different sequences…), as well as the links to download the data in Tab-separated values files (if available).

By double-clicking on some text on the left menus, you will be able to transfer the selected text on the query text box. In the example below, a double click on the CONDITION_PROJECT command wrote the command in the text box and then, a double click on the condition PCP, added the PCP condition in the CONDITION_PROJECT command.



The list of available commands are listed in the "Query commands" left menu. Double clicking on any of them, will transfer an appropriate text to the query text box as a template for completing the command (see screenshot before).

Commands are usually formatted as a capital letters code, followed by a certain number of arguments, separated by commands and between brackets.

> ***Note**: Usually, leaving one argument empty means that you allow that any value can be taken in that particular argument. That is not allowed in all of the arguments. If some error is made in the query, PINT will show an error message in the status text box (at the bottom), describing as possible the error and showing, if necessary, the correct syntax of the command.*

Before describing the commands, in the next sections we will describe some common types of arguments that the commands usually use:

Logical operators

PINT provides a set of commands for query the data. These commands can be combined with logical operators such as the ones in the table:

Logical Operators		Examples	
AND	*Intersection logical operator*	*COND[A1,]* ***AND*** *COND[A2,]*	*Proteins present in condition A1 and condition A2*
OR	*Disjunction logical operator*	*COND[A1,]* ***OR*** *COND[A2,]*	*Proteins present in condition A1 or condition A2*
!	*Negative logical operator*	*COND[A1,]* ***AND !****COND[A2,]*	*Proteins present in condition A1 and not in condition A2*
XOR	*Exclusive OR*	*COND[A1,]* ***XOR*** *COND[A2,]*	*Proteins present in condition A1 or condition A2 but not in both*

Note*: You can also use parenthesis to make more sophisticated commands like:*
(COND[A1,] ***OR*** *COND[A2,]) AND SC[PSM, , SEQUEST:xcorr, >4.5]*

Aggregation level:

Depending on the filter, it can be may applied at different aggregation levels. So, for example, the command for filtering on scores, needs the aggregation level in order to know if the score is assigned to the proteins or to the PSMs…etc.

Aggregation levels	
PROTEIN	*Protein aggregation level*
PSM	*Peptide Spectrum Match level*
PEPTIDE	*Not implemented yet*
PROTEIN GROUP	*Not implemented yet*

Numerical condition:

Numerical condition is composed by an operator and a numerical value. In some cases instead of a numerical value, a text can be used (see examples in table below).

Numerical condition			
Operators	*Number*	*Example*	Meaning of the example
<	*Less than*	*< 2.5*	Less than 2.5
>	*Greater than*	*> 1.0*	Greater than 1.0
=	*Equals to*	*= 0.0*	Equals to 0.0
		= High	Equals to "High" (*in case of text-based scores*)
		= Nan	Equals to Nan, which means that is not present
		*= **	Equals to *, which means that is equals to anything not null (this example is actually equivalent to write "!= Nan")
		= INF	Equals to a POSITIVE_INFINITY (*valid for the RATIO command*)
!=	*Not equals to*	*!= 3*	Not equals to 3
		!= Low	Not equals to "Low" (in case of text-based scores)
		!= Nan	Not equals to nothing, which means that is
		!= -INF	Not equals to NEGATIVE_INFINITY (valid for the RATIO command)
>=	*Greater or equals to*	*>= 3.0*	Greater or equals to 3.0
<=	*Less or equals to*	*<= 5.0*	Less or equals to 5.0

***Note**: A numerical condition used in a RATIO command (see below) can use the INF and –INF annotations, which are referring to a POSITIVE_INFINITY and to a NEGATIVE_INFINITY respectively.*

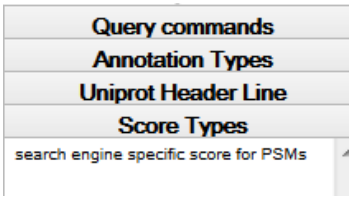
Amount type:

Amount type can be used in the Amount command. The allowed types are the ones in the table below.

Amount types				
INTENSITY	***NSAF***	***dNSAF***	***SPC***	**AREA**
NORMALIZED_INTENSITY	***EMPAI***	***EMPAI_COV***	***XIC***	

Score type:

- **Score type** can be any value from the list below. However, the system will detect the ones that are used in the loaded projects and the possible values will be loaded in one of the lists at the left side of the query page:



Amanda:AmandaScore, Andromeda:score, Ascore:Ascore, Byonic: Peptide AbsLogProb, Byonic: Peptide AbsLogProb2D, Byonic: Protein AbsLogProb, Byonic:Best LogProb, Byonic:Best Score, Byonic:Delta Score, Byonic:DeltaMod Score, Byonic:PEP, Byonic:Peptide LogProb, Byonic:Protein LogProb, Byonic:Score, Comet:deltacn, Comet:deltacnstar, Comet:expectation value, Comet:matched ions, Comet:sprank, Comet:spscore, Comet:total ions, Comet:xcorr, DeBunker:score, Expect value, FDRScore, FDRScore for proteins, H-Score, IdentityE Score, MRMaid:peptide score, MS-GF:DeNovoScore, MS-GF:EValue, MS-GF:Energy, MS-GF:PEP, MS-GF:PepQValue, MS-GF:QValue, MS-GF:RawScore, MS-GF:SpecEValue, MSFit:Mowse score, MSQuant:PTM-score, Mascot:PTM site assignment confidence, Mascot:expectation value, Mascot:homology threshold, Mascot:identity threshold, Mascot:matched ions, Mascot:score, Mascot:total ions, MaxQuant:P-site localization probability, MaxQuant:PTM Delta Score, MaxQuant:PTM Score, MaxQuant:Phospho (STY) Probabilities, MaxQuant:Phospho (STY) Score Diffs, MyriMatch:MVH, MyriMatch:mzFidelity, OMSSA:evalue, OMSSA:pvalue, PEAKS:peptideScore, PEAKS:proteinScore, PSM-level FDRScore, PSM-level combined FDRScore, PTM localization score, Paragon:confidence, Paragon:contrib, Paragon:expression change p-value, Paragon:expression error factor, Paragon:score, Paragon:total protscore, Paragon:unused protscore, Phenyx:AC, Phenyx:Auto, Phenyx:ID, Phenyx:Modif, Phenyx:NumberOfMC, Phenyx:PepPvalue, Phenyx:Peptides1, Phenyx:Peptides2, Phenyx:Pepzscore, Phenyx:Score, Phenyx:User, Profound:Cluster, Profound:ClusterRank, Profound:z value, ProteinExtractor:Score, ProteinLynx:Ladder Score, ProteinLynx:Log Likelihood, ProteinProspector:expectation value, ProteinProspector:score, ProteinScape:IntensityCoverage, ProteinScape:PFFSolverExp, ProteinScape:PFFSolverScore, ProteinScape:ProfoundProbability, ProteinScape:SearchEventId, ProteinScape:SearchResultId, ProteinScape:SequestMetaScore, ProteoGrouper:PAG score, ProteomeDiscoverer:Mascot:Protein CutOff Score, ProteomeDiscoverer:phosphoRS score, ProteomeDiscoverer:phosphoRS sequence probability, ProteomeDiscoverer:phosphoRS site probability, SEQUEST:PeptideIdnumber, SEQUEST:PeptideNumber, SEQUEST:PeptideRankSp, SEQUEST:PeptideSp, SEQUEST:Sequences, SEQUEST:Sum, SEQUEST:TIC, SEQUEST:Uniq, SEQUEST:consensus score, SEQUEST:deltacn, SEQUEST:deltacnstar, SEQUEST:expectation value, SEQUEST:matched ions, SEQUEST:probability, SEQUEST:sf, SEQUEST:sp, SEQUEST:sprank, SEQUEST:spscore, SEQUEST:total ions, SEQUEST:xcorr, SQID:deltaScore, SQID:protein score, SQID:score, Scaffold:Peptide Probability, Scaffold:Protein Probability, Sonar:Score, SpectraST:delta, SpectraST:discriminant score F, SpectraST:dot, SpectraST:dot_bias, SpectrumMill:Discriminant Score, SpectrumMill:SPI, SpectrumMill:Score, X!Tandem:expect, X!Tandem:hyperscore, ZCore:probScore, cluster identifier, combined FDRScore, combined FDRScore for proteins, confidence score, distinct peptide-level FDRScore, distinct peptide-level combined FDRScore, distinct peptide-level probability, higher score better, local FDR, lower score better, manual validation, p-value, peptide identification confidence metric, percolator:PEP, percolator:Q value, percolator:score, probability for proteins, protein ambiguity group result details, protein group passes threshold, protein group-level FDRScore,

```
protein group-level combined FDRScore, protein identification confidence metric,
protein rank, protein-level e-value, protein-level p-value, protein-level q-value,
search engine specific score, search engine specific score for PSMs.
```

Query Commands:

Here you will find a description and some examples for all the available commands for data querying and filtering in PINT.

- **SCORE (SC):** Selects the proteins/PSMs complaining a certain numerical condition in one of their associated scores.

Syntax: **SC[Aggregation_level, Score_type, Score_name, Numerical_condition]**

Examples:

Command	**Explanation**
SC[PROTEIN, p-value, ,]	Proteins detected with a certain score of type "p-value" associated
SC[PROTEIN, , my new p-value,<0.001]	Proteins detected with an associated score name "my new p-value" less than 0.001
SC[PSM, , SEQUEST:xcorr, >4.5]	Psms detected with an associated score name "SEQUEST:xcorr " greater than 4.5
SC[PSM, PTM localization score, Site Localization Confidence, High]	Psms detected with an associated score of type "PTM localization score" named "Site Localization Confidence" equals to "High"

- **CONDITION_PROJECT (COND):** Selects proteins detected on a certain experimental condition in a certain project.

Syntax: **COND[condition_name, project_name]**

Examples:

Command	**Explanation**
COND[saha, project_01032014]	proteins detected in saha condition in the project "*project _01032014*"
!COND[control,]	proteins detected in any other condition than control in any project
!COND[, project _01032014]	proteins detected in any other project than "*project _01032014*"

- **RATIO (RA)**: Selects the proteins/PSMs complaining a certain numerical condition in one of their ratios between two certain experimental conditions.

Syntax: **RA[Aggregation_level, CONDITION_PROJECT, CONDITION_PROJECT, Ratio_name, Numerical_condition, SCORE]**

> ***Note****: This command is always referring to a ratio between two certain experimental conditions. That is why two CONDITION_PROJECT commands are embedded in the command. The SCORE command also embedded here is referring to any score associated with that particular ratio.*

> ***Note'****: Even if a ratio is internally referring to a ratio between condition A and B, that is A/B, if the command is written like 'B/A' the numerical condition will be transformed according to the appropriate value in order to compare the ratio value.*

> ***Note''****: All ratios stored in PINT should be treated as log2 ratios.*

> ***Note'''****: Numerical_condition here can contain POSITIVE_INFINITY and NEGATIVE_INFINITY values (see Numerical condition definition above).*

Examples:

Command	**Explanation**
RA[PROTEIN, COND[saha, proj2], COND[control, proj*1], ,]*	Proteins with any quantitative ratio measured between condition saha from project "proj2" vs condition control from project "proj1"
RA[PROTEIN, COND[saha,], COND[control, proj *2], >= 2.0,]*	Proteins with a quantitative ratio measured between condition "saha" from any project vs condition control from project "proj 2" with a log2 value greater or equal to 2.0
RA[PROTEIN, COND[saha,], COND[control,], =INF, SC[, p-value saha, <= 0.001]]	Proteins with a quantitative ratio measured between condition "saha" from any project vs condition control from any project with a value equals to POSITIVE_INFINITY and an associated score value named "p-value saha" less or equal to 0.001
RA[PROTEIN, COND[saha, proj*1], COND[control,* proj*1], , SC[p-value, , <= 0.001]]*	Proteins with a quantitative ratio measured between condition "saha" vs condition "control"

	both from experiment “proj1“ with an associated score of type “p-value” less or equal to 0.001
RA[PROTEIN, COND[saha_TSA, exp3_saha_tsa], COND[control, exp3_saha_tsa], , SC[p-value, new_p-value, <= 0.001]]	Proteins with a quantitative ratio measured between condition “saha_TSA“ and control from experiment “exp3_saha_tsa“ with an associated score of type “p-value” named “new_p-value” less or equal to 0.001
RA[PSM,COND[LIGHT_DROME,062114_DmDv_isogenic],COND[HEAVY_DROVI,062114_DmDv_isogenic],<-4,]	PSMs with a quantitative ratio measured between condition “LIGHT_DROME” and “HEAVY_DROVI” in the project “*062114_DmDv_isogenic*” with a log2 value less than -4

- **AMOUNT (AM):** Selects proteins/PSMS complaining a certain numerical condition in one of their quantitative amount measurements.

Syntax: **AM[Aggregation level, Amount_type, CONDITION_PROJECT, Numerical_condition]**

Examples:

Command	**Explanation**
AM[PROTEIN, SPC, COND[saha, last_project],]	Proteins with any quantitative spectral count value measured in condition “saha“ from “last_project” project
AM[PSM, , COND[saha AND control,], >= 2.0]	PSMs with a quantitative value measured (of any type) in condition named “saha AND control“ from any experiment with a value (in both cases) greater or equal to 2.0

- **THRESHOLD (TH):** Selects proteins with a certain threshold which only can be have been defined when the dataset is uploaded using a custom formatted Excel file. The idea is that the user can define certain subsets of the dataset by using some thresholds assigned to the proteins.

Syntax: **THR[Threshold_name, Boolean_value]**

Examples:

Command	Explanation
THR[Xscorefilter_TSA, true]	Proteins that contains an applied filter Xscorefilter_TSA and that has been passed that filter
THR[Xscorefilter_TSA, false]	Proteins that contains an applied filter Xcscorefilter and that has NOT been passed that filter

- **SIMPLE_ANNOTATION (AN):** Selects proteins that contains a certain text in their UniProtKB annotations.

> ***Note**: This command is similar to COMPLEX_ANNOTATION. They both try to select proteins containing certain annotations. This one allows less specific queries, since the Annotation_string passed as parameter is searched in every type of annotation.*

Syntax: **AN [Uniprot_version, Annotation_string, Numerical_condition]**

Where ***Uniprot_version*** is the version of the UniProtKB database that will be used as "YYYY_MM", i.e. "2014_07". Note that the available UniProtKB version to query will depend on the data you are querying on, so only the versions from date in which your project(s) were uploaded to Pint will be available. If not stated, the current (last) available UniProtKB official version will be used (the one stated at: ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/reldate.txt).

Where ***Annotation_string*** is any string you want to query in the Uniprot annotations. This field is mandatory for this command.

Where ***Numerical_condition*** means the number of times the annotation should appear in the queried proteins (see third example of the table).

Examples:

Command	Explanation
AN[, cornified ,]	Proteins with the annotation "*cornified"* in any field, using the current Uniprot version
AN[2014_07, cornified ,]	Proteins with the annotation "*cornified"* in any field, using the Uniprot version *2014_07*

AN[2014_07, transmembrane region , =8]	Proteins containing 8 annotations containing the text "transmembrane region" in any field, using the Uniprot version *2014_07*

- **COMPLEX_ANNOTATION (CAN):** This command will allow a more specific search along the UniProtKB annotations by specifying additional fields, such as the uniprot header line, the annotation type, the annotation name and value (see below). So it allows a more advanced and specific query than the "simple annotation command" described before.

***Note**: This command is similar to SIMPLE_ANNOTATION. They both try to select proteins containing certain annotations. This one allows a much more specific queries, since the command can specify the Annotation_Type, the Annotation_name, the Annoration_value and the Uniprot_header_line (see explanations below).*

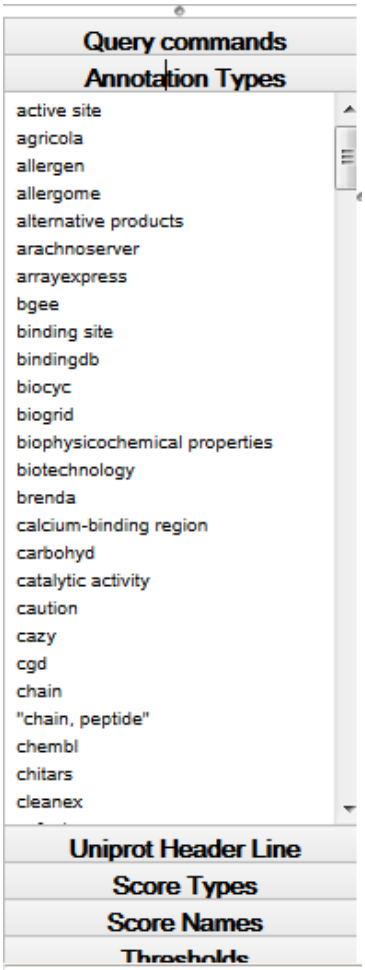
Syntax: **CAN [Uniprot_version, Uniprot_header_line, Annotation_type, Annotation_name, Annotation_value, Numerical condition]**

Where ***Uniprot_version*** is the version of the UniProtKB database that will be used as "YYYY_MM", i.e. "2014_07". Note that the available UniProtKB version to query will depend on the data you are querying on, so only the versions from date in which your project(s) were uploaded to Pint will be available. If not stated, the current (last) available UniProtKB official version will be used (the one stated at: ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/reldate.txt).

Where ***Uniprot_header_line*** is a two letters code equivalent to the two letters of the Uniprot annotation lines. Allowed values are show in the table below:

Two-three letters code	Definition
CC	Comment
FT	Feature
PE	Protein existence
DR	Database cross-Reference
RC	Reference Comment
GO	Gene Ontology
RX	Reference cross-reference
KWR	Keyword
STA	Status
MAN	Manual annotation

DT	Entry and sequence dates and versions



Where ***Annotation_type*** is the type of the Uniprot annotation. Allowed types for Annotation_type are shown in the "Annotation types" menu at the left of the query page (see screeshot at left).

Where ***Annotation_name*** is the name of the Uniprot annotation. Depending on the annotation type, the annotation name can have different values, so these values are not showed here.

Where ***Annotation_value*** is the text that is searched in the annotation, and can be included in the annotation, not necessarily have to be the exact text.

And ***Numerical_condition*** refers to the number of annotations found.

Examples:

Command	**Explanation**
CAN[2014_07,FT, , , ,]	Proteins with any *FT* type annotation
CAN[2014_07, , domain, , ,]	Proteins with a *domain* annotation
CAN[2014_07, , modified residue, N-acetylserine, ,]	Proteins with an annotation of a *modified residue* in a *N-acetylserine*
CAN[2014_07, , transmembrane region, , , =7]	Proteins with 7 different *transmembrane region* annotations
CAN[2014_07, , mass spectrometry, , ,]	Proteins with a *mass spectrometry* annotation

CAN[2014_08, , PTM, , , >=2]	Proteins with a 2 or more different *PTM* annotations
CAN[2014_06, , disease, chromosomal aberration, ,]	Proteins with a *disease* annotation containing the text '*chromosomal aberration*' on it.
CAN[2014_07, CC, interaction, , P56945,]	Proteins with an *interaction* annotation containing the text '*P56945*' in the value, that is, proteins interacting with Uniprot protein *P56945*
CAN[2014_07, , subunit, ,Interacts with EPHA3,]	Proteins with a *subunit* annotation containing the text '*Interacts with EPHA3*' in the annotation value, that is, proteins containing a subunit that interacts with EPHA3

- **GENE_NAME (GN):** This command will select the proteins associated to a certain gene name.

> ***Note**: This command is not case sensitive and a partial gene name can be submitted. For example querying for GN[myo] could return proteins with gene names such as: Myo18a, Myo5c, Myo9a…etc.*

Syntax: **GN [Gene_name]**

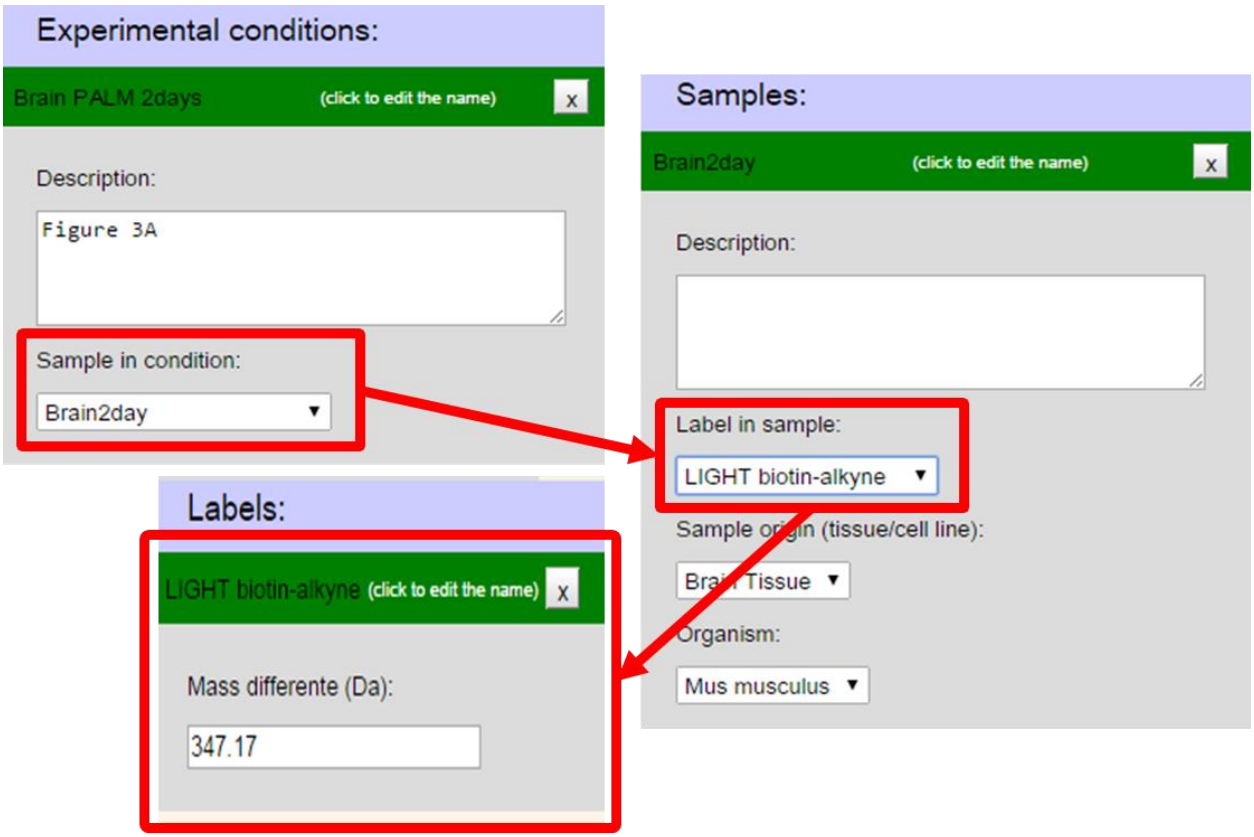
Examples:

Command	**Explanation**
GN[myo]	Proteins with a gene name containing "myo"

- **LABEL (LB):** This command will select the proteins/PSMs associated to a certain label name.

Syntax: **LB[Aggregation_level, Label_name, ONLY]**

Where ***Label_name*** is the name of the label associated to the Protein or PSM. The label is usually defined in the graphical interface for submitting new projects (see screenshot below). Each sample has a label associated, and each experimental condition has a label associated. Therefore, any protein or PSM detected in that experimental condition will have that associated label.



Where ***ONLY*** is a parameter that can only be used when *Aggregation_level* is PSM and if it is present forces the PSM to only have that particular label associated.

> ***Note**: Usually each PSM has two labels associated, and therefore a quantitative ratio can be calculated. Some quantitative approaches allow to get information from the singleton peptides, which are the peptides (or PSMs) that only have one signal of the two conditions to compare, and therefore, they will have ONLY one associated label.*

Examples:

Command	**Explanation**
LB[PROTEIN, heavy,]	Proteins labeled as “heavy”
LB[PROTEIN, 116,] AND LB[PROTEIN, 114,]	Proteins labeled as “116” and “114”
LB[PSM, light, ONLY]	PSMs labeled as “light” but no other label.
LB[PSM, light,]	PSMs labeled as “light”, no matter if other labels are also associated or not.

- **MSRUN (MSRUN):** This command will select the proteins/PSMs associated to a certain label name.

Syntax: **MSRUN[CVS_MS_run_ids]**

Where ***CVS_MS_run_ids*** is a list of MS Run ids separated by commas, meaning that proteins detected in at least one of the MS runs listed are going to be retrieved.

Examples:

Command	**Explanation**
MSRUN[runA1]	Proteins detected in the MS run "runA1"
MSRUN[repA1, repA2, repA3]	Proteins detected in any of the MS runs: "repA1", "repA2" or "repA3".

- **PROTEIN_ACC (ACC):** This command will select the proteins with a certain accession.

Syntax: **ACC[CVS_accessions]**

Where ***CVS_accessions*** is a list of accessions separated by commas, meaning that proteins with any of these accessions are going to be retrieved.

Examples:

Command	**Explanation**
ACC[IPI00763970.1]	Proteins with accession "IPI00763970.1"
ACC[D3ZWC6, D3ZAF7, IPI00763970.1]	Proteins with any of these accessions: "D3ZWC6", "D3ZAF7" or "IPI00763970.1".

***Note**: The type of the accession can be different between the accessions in the list.*

***Note'**: When submitting proteins with NCBI or IPI accessions, PINT will try to convert them as UniProtKB ones, but both of them will be kept in the system and therefore will be queriables.*

- **TAXONOMY (TX):** This command will select the proteins or PSMs belonging to a certain taxonomy. In case of proteins, we assume that one protein can belongs to a single taxonomy/organism. However, in case of PSMs, they can belong to more than one taxonomy/organism since they can be shared by different proteins belonging to different species.

Syntax: **TX[Aggregation_level, Organism_name, Ncbi_tax_id, ONLY]**

Where ***Organism_name*** is the common name of the organism according to the NCBI taxonomy database (http://www.ncbi.nlm.nih.gov/taxonomy).

Where ***Ncbi_tax_id*** is the identifier of the taxonomy in the NCBI taxonomy database as a integer number (i.e. 7227).

Where ***ONLY*** is a parameter that can only be used when *Aggregation_level* is PSM and that means that the PSM can ONLY belong to the provided taxonomy.

> ***Note****: The taxonomy can be provided either with the Organism_name or the NCBI_tax_id or both (see examples below).*

Examples:

Command	**Explanation**
TX[PROTEIN, Drosophila melanogaster, ,]	Proteins from the Drosophila melanogaster species
TX[PROTEIN, , 7227,]	Proteins from the Drosophila melanogaster species (tax id: 7227)
TX[PSM, , 7227, ONLY]	PSMs mapped ONLY to proteins from the Drosophila melanogaster species (tax id: 7227)
TX[PSM, , 7227,]	PSMs mapped to proteins from the Drosophila melanogaster species (tax id: 7227) no matter if they are also mapped to any other protein from any other species.