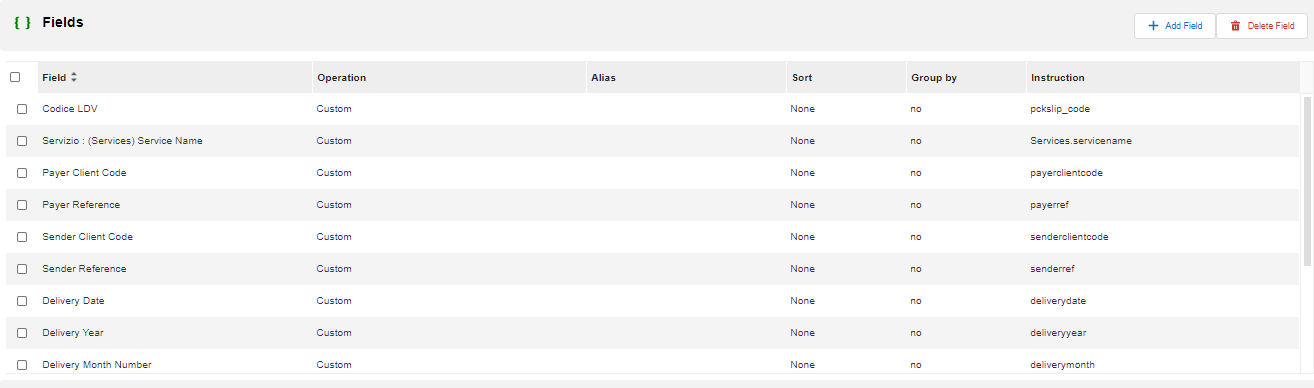
1. **QUERY BUILDER UI MODE**

**Query Builder** is a tool useful for non-technical user and also tech users to build SQL query-s in a more faster and graphical way. The user interface of the builder is divided in 3 important sections as ONE BASIC-SQL-QUERY

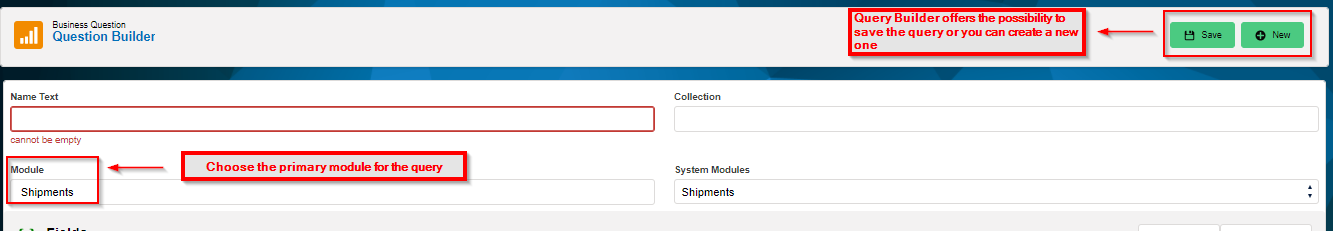
*SELECT [field1, field2, field3]  
FROM [table]  
WHERE [condition]*

With query builder the user is able to build query and join table without prior knowledge of SQL. Using the modular structure of Corebos the creation of SQL JOIN-s is easier, where all you have to do is the selection of the right field in First Section as you see in the images below. It had to be mentioned that the Builder we can JOIN direct relation module from example Sales Order-Vendor, Project-Account etc.

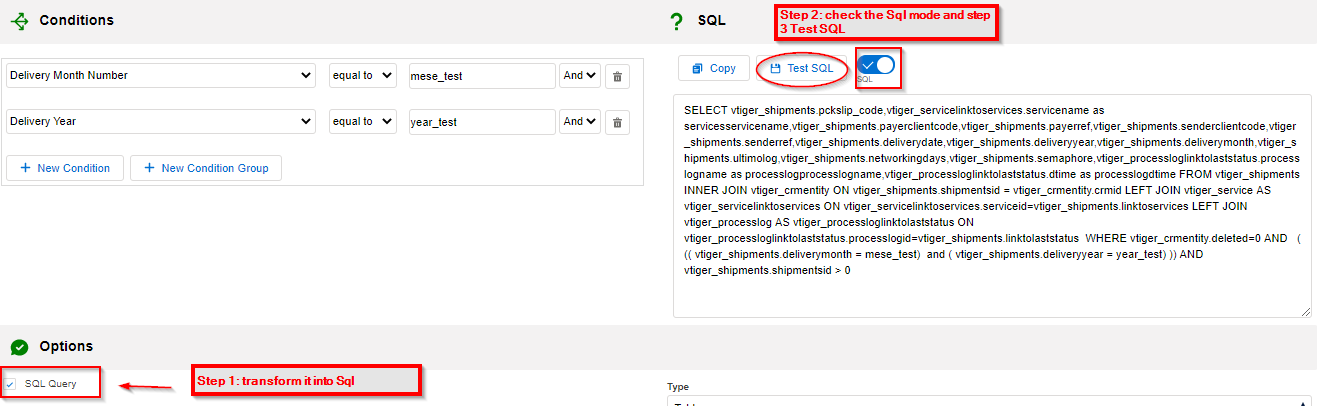


1. How to create a cbQuestion using **QueryBuilder**

* For cbQuestion you have to choose a parent module, where is the source of the data that you are retrieving and type a name that describes the purpose of your query.

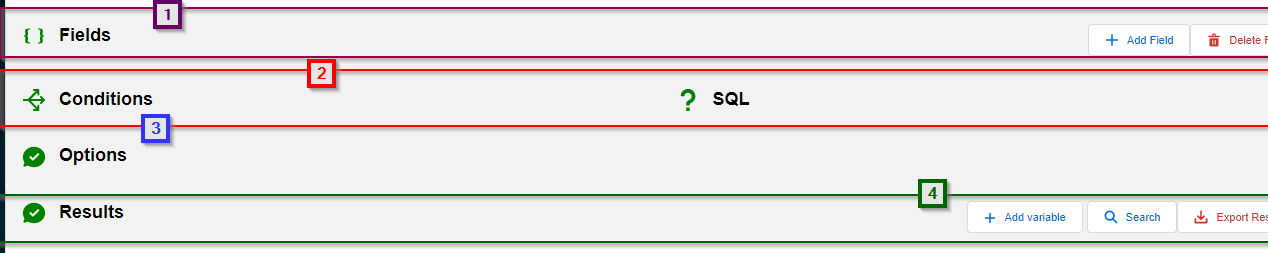


* Query Builder offers the possibility to modify the query and save it or you can create a new one and store it.
* Query Builder is able to transform it into a SQL query and in a JSON format code
* If you want to transform a query into a SQL language you just need to check the checkbox below and test the Question in the Editor like showed in the image.



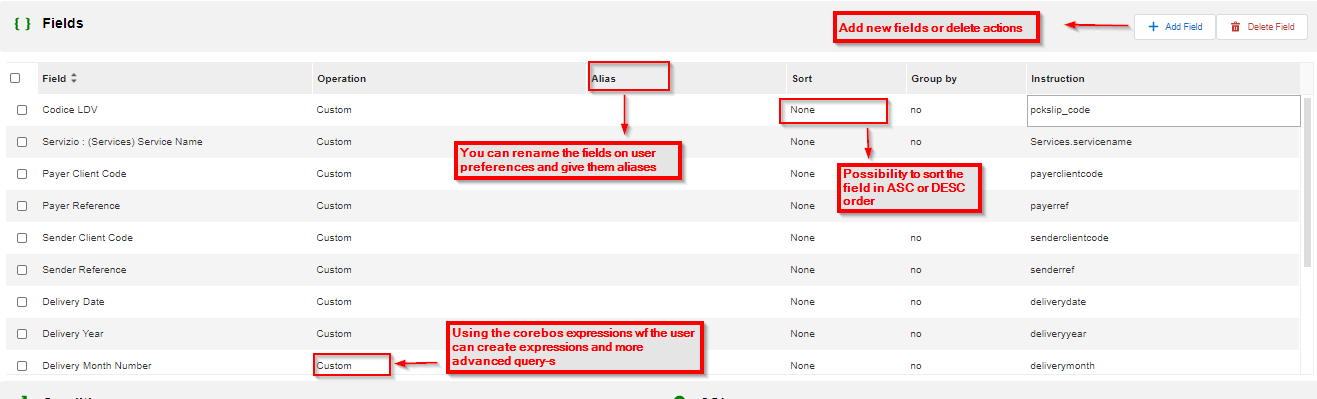
1. **FIELD OPERATIONS**

The **[FIELD\_SECTION]1** is equivalent with the SELECT syntax of a basic standard query. In this section the user can add field in a query, delete them or transform them using the workflow-s expression that Corebos offers. Also using Query Builder you can group the data accordingly to specific required field.



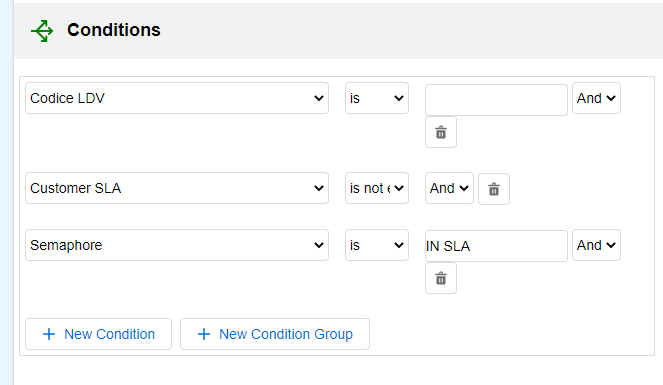
**Some of the fields operations that Query Builder offers are listed below:**

1. **AS** - you can rename field in your query and give them more insightful name accordingly to the business logic. Alias-es are important also when you want to re-use the same field and create a workflow expression above of them.
2. **ORDER** – you can sort based on field types ordering them into Ascending or descending order
3. ***FUNCTIONS*** - this section is a more advanced option, where for each field you can transform them using the internal workflow system and create more advanced metrics. For example we have functions like: concat, substring, uppercase, min/max value, avg/count/sum operations. Keep in mind that Corebos is very versatile and the implementers can create any function using the Workflows function and include it in the Query Builder accordingly to the applied business logic.
4. **GROUP BY** - The GROUP BY statement group’s rows that have the same values into summary rows, like "find the number of customers in each country". The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.



1. **CONDITIONS**

The **[CONDITIONS\_SECTION]2** is equivalent with the WHERE syntax of a basic standard query. In this section the user can add diverse where conditions using AND, OR operations as shown in the example below.



**2. QUERY BUILDER SQL MODE**

**Scope:** The initial idea of including the SQL mode in the Editor was because using the Interface we can’t join tables on other related modules. For example, the shipments join ProcessLog and the ProcessLog join “cbStatus” table to retrieve the name of the status, code and other fields needed. In these cases we have to use SQL because we need to get fields from modules that don't have a direct relation with each other. For this purpose, we proposed a way for the users to use the Builder with Native SQL.

The **SQL Builder Mode** permits the user to write SQL queries and to be able to load them without entering in the database and perform actions like exporting the data, visualizing them etc. The interface of the SQL mode is a little different from the one we have explained before. The SQL editor is useful for end-user with prior knowledge of Corebos ER and is able to write SQL queries in Corebos. Graphically this mode contains only the SQL converted from the **Query\_Generator** and the Results where the user has the possibility to load them and export.

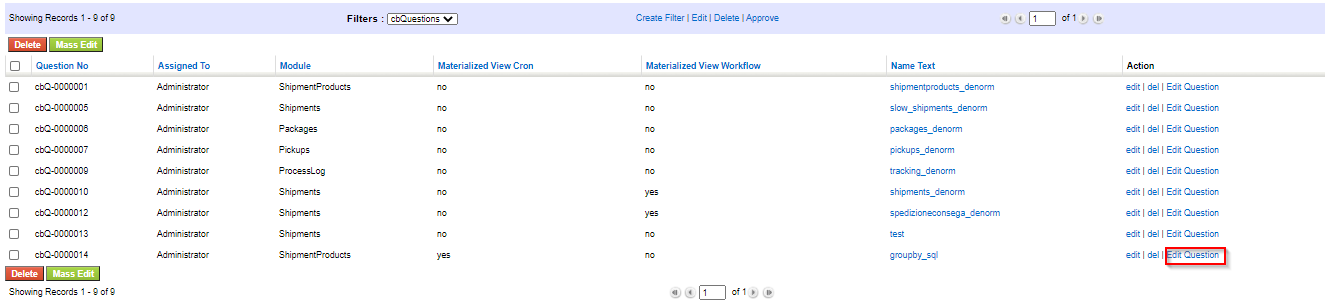
**UI Changes:**  
• we hide the fields and the Conditions column [note: that since we hide the conditions column the SQL editor will stretch to full length of the window]  
• block the web service/SQL switch in SQL (you can't type in a webserivice query)  
• add a property with the names of the context variables, so we can preload them in the context variable table  
• modify the test button to load the context variables

With these changes we get a manual SQL editor that you can execute and see the results.

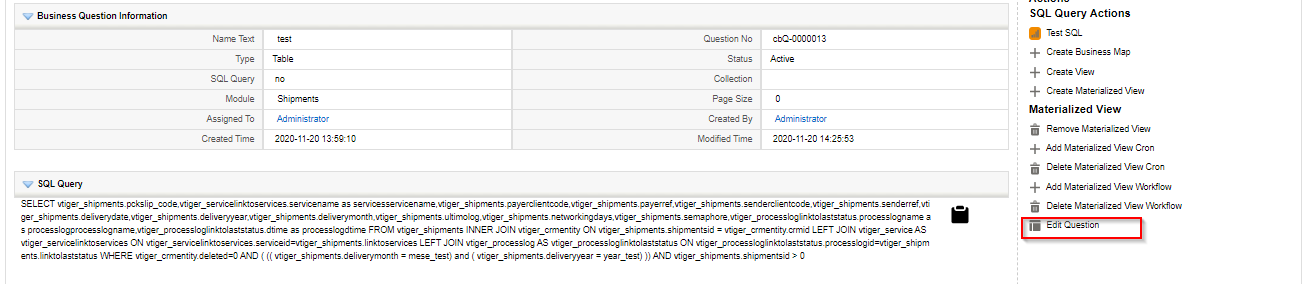
**INSTALLO CASE**

Since the volumes of the Installo denormalized views are usually very big, they can contain more than +180k, is needed to be added a filtering possibility [so giving context to the query] in order to be able exporting results based on context parameters defined. The final output of Installo is to create the denorm tables accordingly to the clients requests and to be able to export the results that will serve as an input for other BI tools. We have to provide the end user the denormalized tables as an input for their monthly reports that they perform with being able to export them externally. For this reason we need to add these actions in the cbquestion Editor [Query Builder]. So when you are in a DV of a denorm\_view the user will be able to Edit\_cbQuestion.

**Image 1: ListView Actions to Edit the created Questions**



**Image 2: DetailView Action in a cbQuestion record**



These actions can be created in each Corebos installation in the Business Actions module. This functionality can serve also for the other Clients purposes as a fast SQL exporter of the field offering the possibility to use Corebos application as a database and perform various query-s in order to get the needed result in one system.

Also giving context to query-s is very useful for the implementers or other end-user to retrieve only the needed information by just giving the parameters values and save time in working with big data files or even using external software to perform those actions for you. In short work this functionality helps for other BI operations needed for data analysis.

1. **CONTEXT VARIABLES**

**Context variables** are the variables which can have different values in different environments. You can create a context group which can hold multiple context variables. The including of context variables offers the possibility to give any query context and using the SQL mode you can load results depending on the context set by the end-user. ***To give a real example: “When we want to extract data of the last month Shipments of the year 2020. “***

We need to simply declare the month,year field as context variables and is not more needed to change the query each time or save a new cbQuestion In this case the context can have different values and using the QueryBuilder the user can give the context different values and load the query and export all the data needed according to the business logic decided by the client.

**Below are real cases of Installo project and examples how to create a cbQuestion with context.**

**Step 1:** Define Module

**Step 2:** Define fields, Conditions and Field/JOIN operations as explained above in the non-sql mode.

[The SQL editor is useful for end-user with prior knowledge of Corebos ER and able to write SQL queries in Corebos.]

**Step 3:** Set Context Variables

In the query is needed to define which fields are going to be context variables.

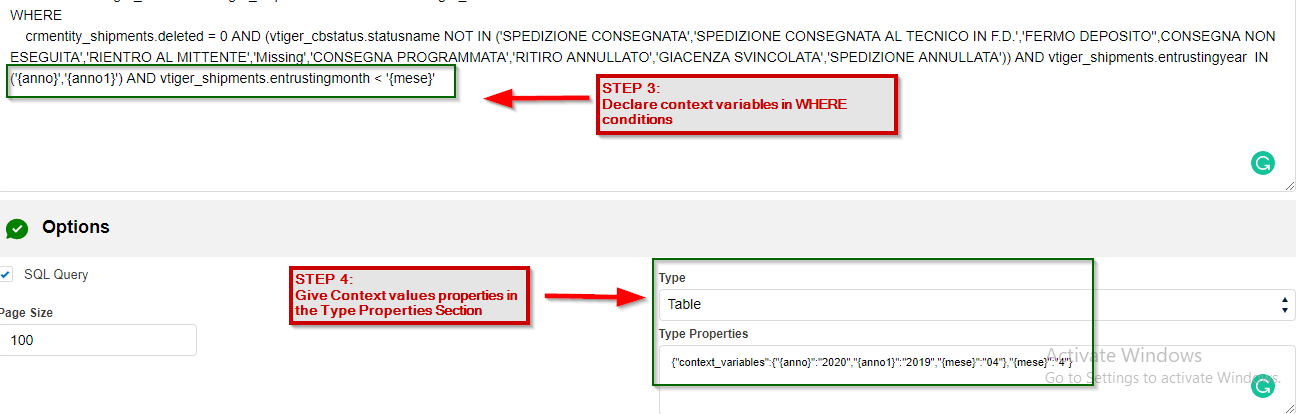
* The context has to be set as variable in the WHERE condition for example table.fieldname = ‘{context1}’

In the example below is set as context the field entrustingyear [the year when the shipments is created].

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | The context initially it should be set in the WHERE CONDITON | Description | TYPE PROPERTIES |
| vtiger\_shipments.entrustingyear = '{anno}’ | | Retrieve shipments of 2019 | {"context\_variables":{"{anno}":"2019}} |
| 2 | OR it can be declared using the operator if you need to expand the results | **Description** | **TYPE PROPERTIES** |
| vtiger\_shipments.entrustingyear > '{anno}’ | | Retrieve only the shipments where year>2018 | {"context\_variables":{"{anno}":"2018}} |
| 3 | **OR you can create group of context variables depending on the case** | **Description** | **TYPE PROPERTIES** |
| vtiger\_shipments.entrustingyear IN ('{anno}','{anno1}') | | Retrieve only the shipments of year [2020,2019] | {"context\_variables":{"{anno}":"2020","{anno1}":"2019"}} |
| 4 | The same operations can be done with more than one field declared as context. | **Description** | **TYPE PROPERTIES** |
| vtiger\_shipments.entrustingyear IN ('{anno}','{anno1}') AND vtiger\_shipments.entrustingmonth < '{mese}' | | we can use month and year as context to retrieve Shipments data of the last 3 months of year 2019 and 2020. | {"context\_variables":{"{anno}":"2020","{anno1}":"2019","{ mese}":"4"}} |

**Step 4:** Add Properties

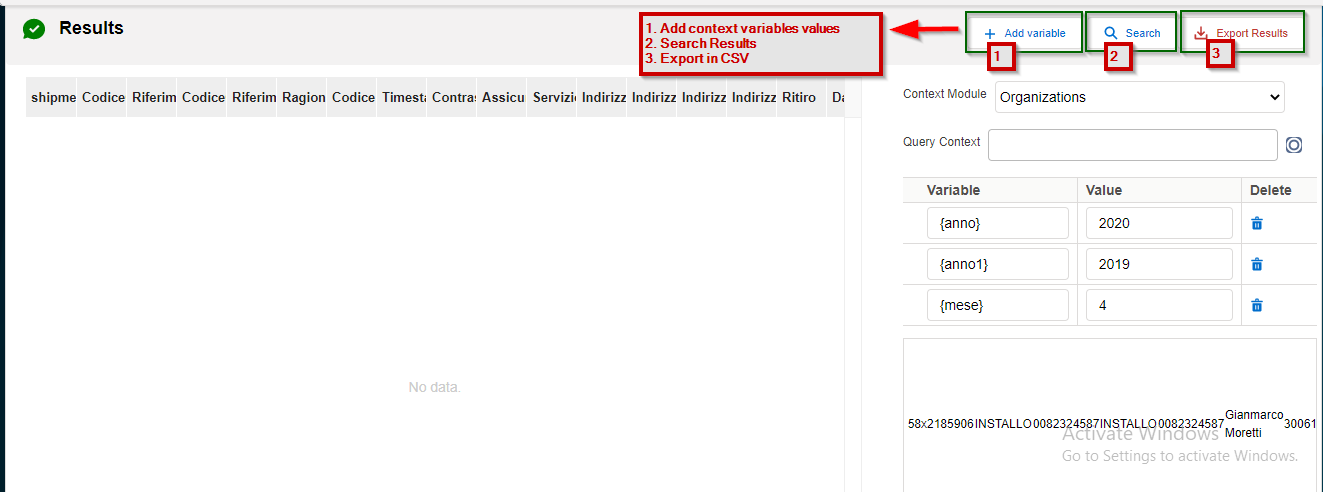
* In the Options Section, in the Type Properties we need to save the combinations of context-value for each variable we have declared as context in the WHERE condition.



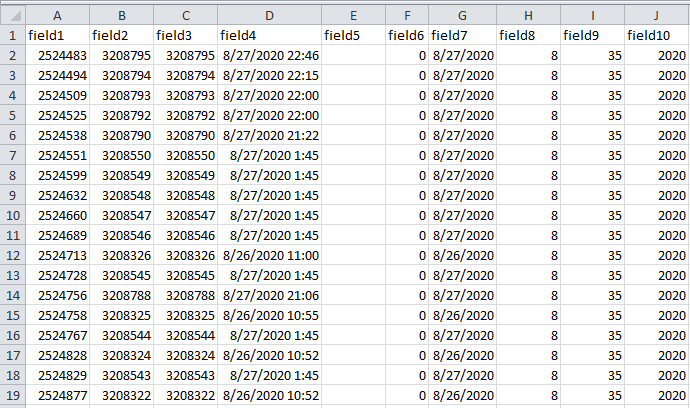
1. **EXECUTE QUERY**

In the results section the end-user can execute the query inside the application. Apart from loading the query also the user is able to:

* Load query in base of context variables
* Export Results in csv



**EXPORTING-ALIASES SUPPORTED**



**COMPLEX EXAMPLE QUERY: Use month and year as context to retrieve Shipments data of the last 3 months of year 2019 and 2020.**

**In the example are tested:**

1. Multiple direct JOIN and non-direct JOIN
2. **Warning:** The ALIAS-es has to be declared without parenthesis AS fieldname
3. Supported IF Expression and CASE statement for the cases of Installo

SELECT vtiger\_shipments.shipmentsid AS shipmentsid,

vtiger\_shipments.payerclientcode AS CodiceClienteOrdinante,

vtiger\_shipments.payerref AS RiferimentoOrdinante,

vtiger\_shipments.senderclientcode AS CodiceClienteMitt,

vtiger\_shipments.senderref AS RiferimentoMittente,

vtiger\_shipments.companyname AS RagioneSociale,

vtiger\_shipments.pckslip\_code AS CodiceLDV,

IF(

vtiger\_shipments.entrusting\_date = '',

NULL,

STR\_TO\_DATE(entrusting\_date, '%Y-%m-%d %H:%i:%s')

) AS TimestampAffidamento,

vtiger\_shipments.cash\_on\_delivery AS Contrassegno,

vtiger\_shipments.insured\_value AS Assicurate,

vtiger\_shipments.linktoservices AS Servizio,

vtiger\_shipments.deliverydate AS DataConsegna,

vtiger\_shipments.entrustingdate AS DataAffidamento,

vtiger\_shipments.entrustingyear AS AnnoAffidamento,

vtiger\_shipments.entrustingmonth AS MeseAffidamento,

vtiger\_shipments.entrustingweek AS WeekAffidamento,

IF(

vtiger\_shipments.delivery\_date = '',

NULL,

STR\_TO\_DATE(

vtiger\_shipments.delivery\_date,

'%Y-%m-%d %H:%i:%s'

)

) AS TimestampConsegna,

vtiger\_service.servicename AS ServiceName,

vtiger\_service.service\_no AS ServiceNo,

vtiger\_service.servicesrcid AS IDServizio,

vtiger\_service.servicecode AS CodiceServizio,

vtiger\_processlog.dtime AS DataUltimoEsito,

vtiger\_cbstatus.statusname AS UltimoEsito,

CASE

WHEN vtiger\_service.servicecode IN ('R','RP','RR') THEN

CASE

WHEN (vtiger\_cbstatus.statusname = 'IN ARRIVO' AND vtiger\_service.servicecode != 'RP') THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN (vtiger\_cbstatus.statusname = 'IN ARRIVO' AND vtiger\_service.servicecode = 'RP') THEN 'RITIRO ESEGUITO IN ATTESA DI RIENTRO'

WHEN vtiger\_cbstatus.statusname ='IN RITIRO' THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN vtiger\_cbstatus.statusname ='IN TRANSITO' THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN vtiger\_cbstatus.statusname ='IN CONSEGNA' THEN 'IN CONSEGNA'

WHEN vtiger\_cbstatus.statusname ='SPEDIZIONE PARTITA' THEN 'IN CONSEGNA'

WHEN vtiger\_cbstatus.statusname ='ORDINE SOSPESO' THEN 'ORDINE SOSPESO'

WHEN vtiger\_cbstatus.statusname ='CARICO VETTURA' THEN 'RIENTRO ESEGUITO'

WHEN vtiger\_cbstatus.statusname ='SPEDIZIONE RITIRATA' THEN 'RITIRO ESEGUITO'

WHEN vtiger\_cbstatus.statusname ='ASSENTE' THEN 'RITIRO FALLITO'

WHEN vtiger\_cbstatus.statusname ='MERCE NON PRONTA' THEN 'RITIRO FALLITO'

END

ELSE

CASE

WHEN vtiger\_cbstatus.statusname ='IN GESTIONE' THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN vtiger\_cbstatus.statusname ='IN RITIRO' THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN vtiger\_cbstatus.statusname ='IN TRANSITO' THEN 'IN ATTESA DI PROGRAMMAZIONE'

WHEN vtiger\_cbstatus.statusname ='IN CONSEGNA' THEN 'IN CONSEGNA'

WHEN vtiger\_cbstatus.statusname ='CARICO VETTURA' THEN 'IN CONSEGNA'

WHEN vtiger\_cbstatus.statusname ='FERMO DEPOSITO' THEN 'IN GIACENZA'

WHEN vtiger\_cbstatus.statusname ='IN GIACENZA' THEN 'IN GIACENZA'

WHEN vtiger\_cbstatus.statusname ='RICEZIONE FILE' THEN 'RICEZIONE FILE'

WHEN vtiger\_cbstatus.statusname ='ASSENTE' THEN 'Consegna fallita'

WHEN vtiger\_cbstatus.statusname ='MERCE RESPINTA' THEN 'Consegna fallita'

WHEN vtiger\_cbstatus.statusname ='ORDINE SOSPESO' THEN 'ELIMINATA'

WHEN vtiger\_cbstatus.statusname ='CONSEGNATA A MITTENTE' THEN 'CONSEGNATA A MITTENTE'

END

END AS Raggruppamenti

**FROM** vtiger\_shipments INNER JOIN vtiger\_crmentity crmentity\_shipments ON vtiger\_shipments.shipmentsid = crmentity\_shipments.crmid

LEFT JOIN vtiger\_processlog ON vtiger\_shipments.linktolaststatus = vtiger\_processlog.processlogid

LEFT JOIN vtiger\_cbstatus ON vtiger\_processlog.linktostatus = vtiger\_cbstatus.statusid

LEFT JOIN vtiger\_service ON vtiger\_shipments.linktoservices = vtiger\_service.serviceid

**WHERE**

crmentity\_shipments.deleted = 0 AND (vtiger\_cbstatus.statusname NOT IN ('SPEDIZIONE CONSEGNATA','SPEDIZIONE CONSEGNATA AL TECNICO IN F.D.','FERMO DEPOSITO'',CONSEGNA NON ESEGUITA','RIENTRO AL MITTENTE','Missing','CONSEGNA PROGRAMMATA','RITIRO ANNULLATO','GIACENZA SVINCOLATA','SPEDIZIONE ANNULLATA')) AND **vtiger\_shipments.entrustingyear IN ('{anno}','{anno1}') AND vtiger\_shipments.entrustingmonth < '{mese}'**;