Ab initio GDE

Lab Book

1. Filter by expression & Reformat

Develop a small graph filtering out the records having level A, B and C and replace that level with 1, 2 and 3 and according to this level give bonus to every employee with multiple of 1000 with level per month, from the date of joining to each individual. Add a column bonus in the desired output. For levels other than A, B, C gives a flat bonus of 5000 per month from the date of Joining. (Ref Emp DML).

1. **Sort & Dedup**

A feed file from a banking system consists of the various transactions taking place over a period of time.

Process this file as below.

1) Obtain the transactions with the highest transaction amount for each account.

2) Obtain the transactions with the lowest transaction amount for each account.

3) Obtain the accounts which have undergone only a single transaction over this period.

References

1) Input file dml

****

1. Partition and Departition components partition by key, gather, concatenate

Code the join example using parallelism

Reference:

Employee (emp)

|  |  |  |
| --- | --- | --- |
| Structure | Field | Mapping Rule |
| String | EMPID | EMPLOEE ID |
| String | NAME | NAME |
| String | LEVEL | A,B,C,D,E |
| String | GENDER | M or F |
| Decimal | SALARY | 9999999.99 |
| Date (dd/mm/yyyy) | DOJ | Date > 01/02/2005 and <05/06/2005 |
| String | SKILL | Skill |
| String | PNONENO | 99999999 |
| String | REGION | Region |

Region (Look Up)

|  |  |  |
| --- | --- | --- |
| Structure | Field | Mapping Rule |
| String | Region | Region |
| Decimal | Code | Code |

1. **Join ( Inner, Outer and Explicit)**
2. The claims table in PIW is populated by a complex legacy etl code. The business intends to populate the claims table in PLW by unloading data from the claims PIW table to avoid the efforts of reverse engineering the complex legacy etl code. Claims table in PIW does not maintain history; however it is required to retain history in the claims table present in PLW. Claims table in PIW & PLW have the same structure besides an additional field called history status in PLW. history\_status can have the following values.

C🡪CURRENT

H🡪HISTORY

D🡪DELETED

Build the above code using the below guidelines.

1) If the claim\_nbr is present in both PIW & PLW (having history\_status=C) with all fields identical, ignore such records.

2) If the claim\_nbr is present in both PIW & PLW (having history\_status=C) with atleast one field different, update the record in PLW by setting the history\_status=H and insert the record from PIW with history\_status=C.

3) If claim\_nbr is present in PIW but not in PLW insert it into PLW with history\_status=C.

4) If claim\_nbr is present in PLW (having history\_status=C) but not in PIW, update the record in PLW by setting the history\_status=D

Note: PLW is implemented in teradata. In teradata delete and insert is preferred over update operation as updates take longer/costly. Please build the above code keeping this in mind. Use flat files in place of tables if necessary.

References

1) Input table dml

2) Output table dml



1. **Rollup**

A feed file from a supermarket management system provides transaction data. It consists details like customer\_id, purchase\_date, purchase\_amount etc. Process this file to obtain the date of first transaction, date of the last transaction and the total amount purchased by each customer.

References

1) Input file dml

2) Output file dml

****

1. **Scan**

A feed file consists of transaction data from a credit card system. Details like account\_nbr, transaction\_date & time and the transaction amount are present. However the available credit limit after each transaction is not available. Assume that before the first transaction for each account the available credit limit is Rs 100000. Use the scan component to provide details about the available credit limit after each transaction.

References

1) Input file dml

2) Output file dml

1. **Lookup**

Consider a feed file from a telecom system consisting details like phone number, region, customer\_id etc. Develop a graph to match the region with the look up and add the std\_code to phone number from the lookup in the format of (9999-99999999). If the region is not found in lookup then add default std\_code as “0000” in phone number field. (Ref. EMP DML and lookup Region)

1. **Read and Write multiple files**

The employee detail files from departments A, B and C are available having the fields

mentioned below. Every department maintains the data of all the employees working

in it only. The departments have 3, 4, and 2 employees respectively.

(a) Develop a graph to read the three files from the departments A, B and C and create as output a different file for each of the employees. The output files should contain a field called DEPARTMENT in addition to the fields in the input file.

(b) Develop a graph to read the three files from the departments A, B and C and create as output five different files – one each for employees at levels A,B,C,D,E. The output files should contain a field AVGEXP in addition to the fields from the input files which depicts the average work experience of the employees at that level.

Reference:

Employee (emp)

|  |  |  |
| --- | --- | --- |
| Structure | Field | Mapping Rule |
| String | EMPID | EMPLOYEE ID |
| String | NAME | NAME |
| String | LEVEL | A,B,C,D,E |
| String | GENDER | M or F |
| Decimal | SALARY | 9999999.99 |
| Date (dd/mm/yyyy) | DOJ | Date > 01/02/2005 and <05/06/2005 |
| String | SKILL | Skill |
| String | PNONENO | 99999999 |
| String | REGION | Region |

Extract all the row and column from the financial.accts table and match the data on cust\_id with the out\_financials.dat file.

a) Set acct\_end\_date to “31/12/9999” if the cust\_id is found in both the financial.accts table and out\_financials.dat file, else set acct\_end\_date to Null.

b) Insert records from out\_financials.dat file to financial.accts table if the cust\_id is not present in financial.accts table, else update financial.accts table with records from out\_financials.dat file.



1. **Partition by expression & rollup**

Consider a feed file consisting of a header, footer and some data records. The data records consist of homes policy data. These policies can be for rented or owned property. More than one rented property can be insured against a single policy while only a single owned property can be insured against a single policy. The header and footer records give the creation date, cycle no and total of the number of records present in the feed file.

Process this file as below.

1) Verify that the number of data records matches the count mentioned in the header and footer. If not make an entry to an alert file about the mismatch which can be later used to raise an alert and abort the process.

2) Check that there is no more than one owned property insured against a policy. Reject policies which do not comply and store them in a reject file.

3) For policies on rented property store the count of IE insured against the policy in field IE\_count.

References

1) Input file dml

2) Output file dml

