Talend Open Studio for DI

Creating an ETL layer for Data Warehousing

For Healthcare atHome India Pvt Ltd

1. DOCUMENT OBJECTIVE

To provide detailed information on the creation of data warehouse for the purpose of business intelligence for Health Care At Home India Pvt Ltd.

Instructions for One time setup to connection to Talend Server:
Please note these steps are already done in the Latitude Laptop reserved for BI purpose. You don't need to repeat unless you change the machine.

1. Install openvpn client in your desktop/laptop
from http://help.unotelly.com/support/solutions/articles/xxxxxxx-openvpn-info-and-files-please-read (Do not download the openvpn config file, it is attached with this mail - ronak.ovpn)
Download the attached files to a location of your choice.
Open ronak.ovpn in an editor:
Replace the path of the ca, cert and key to the path of your downloaded files and save the file.
Double click client.ovpn. Openvpn client will open. Click on the client icon, it will ask for a username and password.
username : xxxxx
password : xxxxxxxxxxxxxx

2. After openvpn successfully connects, open remote desktop connection in your windows pc and put xx.xx.xx.xxx in computer name. A new window will open with login to xrdp message. Enter following credentials
username – ubuntu
password - xxxxxxxxxx
You will now connect to the server desktop.

3. To start Talend server, double click on the desktop shortcut - open_talend.sh

Instructions to connect to Talend on Daily basis (It is recommended to keep the BI laptop switched on at all times)

The server is on AWS in Singapore. To connect we need to establish an Open VPN connection and MSTSC. These are already configured in the Latitude Laptop (reserved for BI purpose)

1. In the bottom right corner, click Open VPN GUI
2. Click Connect
3. In 5-10 seconds, the display will show Management:> STATE: xxxxxxxx Connected Success
4. Now minimize the window
5. Open Windows Remote Desktop and connect using the below creds

Singapore talend server xx.xx.xx.xxx
username - ubuntu
password - xxxxxxxxxxx

6. Click open_talend.sh

2. SOFTWARE DESCRIPTION

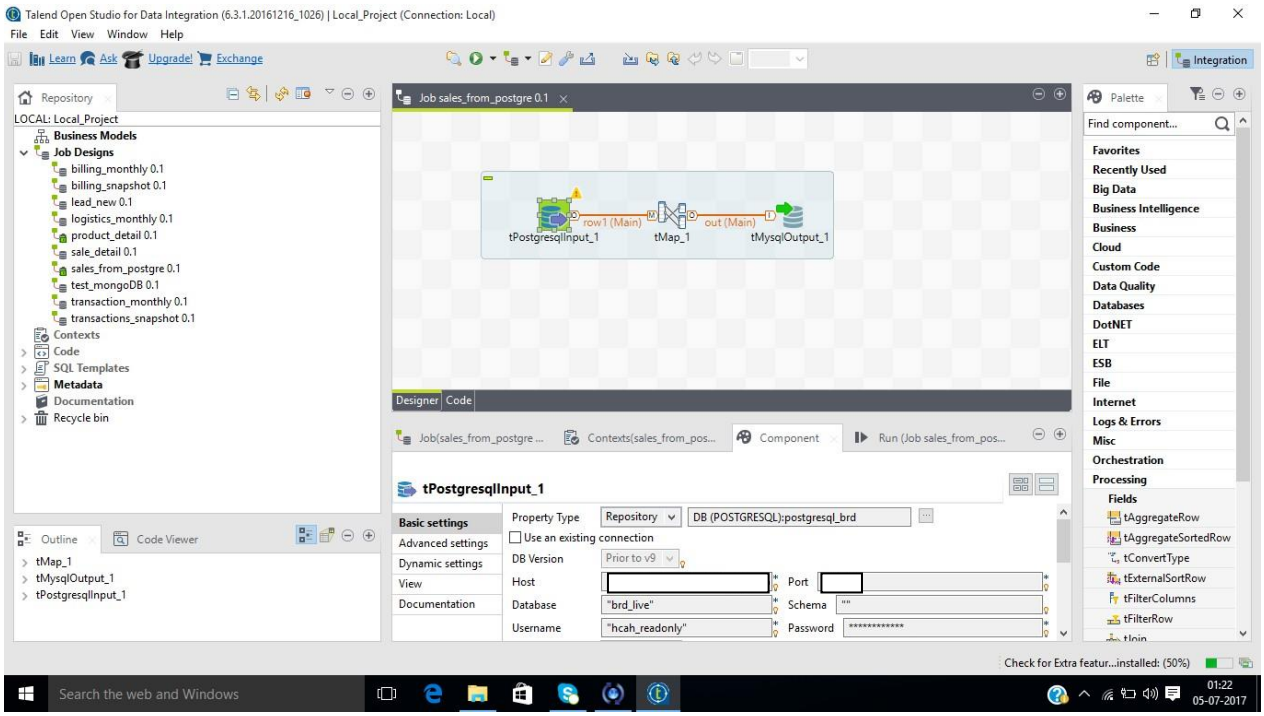
Talend Open Studio for Data Integration is an open source data integration product developed by Talend and designed to combine, convert and update data in various locations across a business. Talend Open Studio for Data Integration operates as a code generator, producing data-transformation scripts and underlying programs in Java. Its GUI gives access to a metadata repository and to a graphical designer. The metadata repository contains the definitions and configuration for each job - but not the actual data being transformed or moved. All of the components of Talend Open Studio for Data Integration use the information in the metadata repository.

Users can design individual jobs using graphical components, from a set of over 900, for transformation, connectivity, or other operations. These jobs created can be executed from within the studio or as standalone scripts.

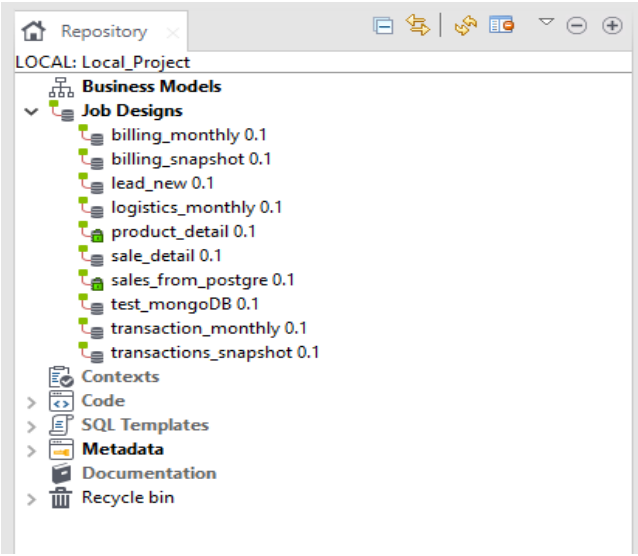
Talend Studio main window is the interface from which you manage all types of data integration processes. The Talend Studio multi-panel window is divided into:

- Repository tree view
- Design workspace
- Palette
- Various configuration views in a tab system, for any of the elements in the data integration Job designed in the workspace
- Outline view and Code Viewer

The figure below illustrates Talend Studio main window and its panels and views

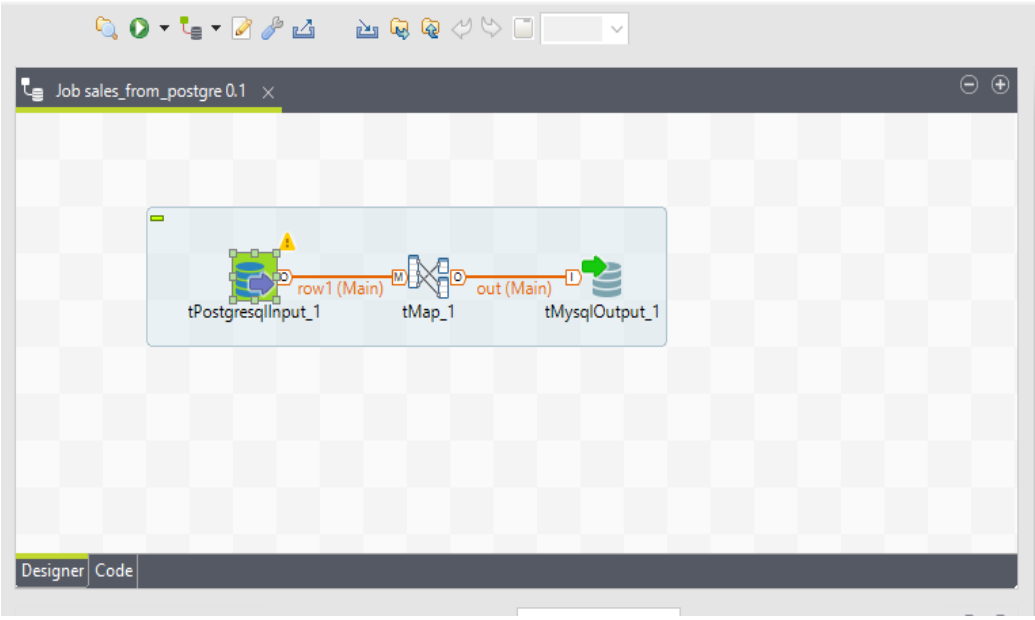


(i) **Repository Tree View:** The Repository centralizes and stores all necessary elements for any Job design and business modeling contained in a project. It stores all your data (Business, Jobs) and metadata (Routines, DB/File connections, any meaningful Documentation and so on). The figure below illustrates the elements stored in the Repository:

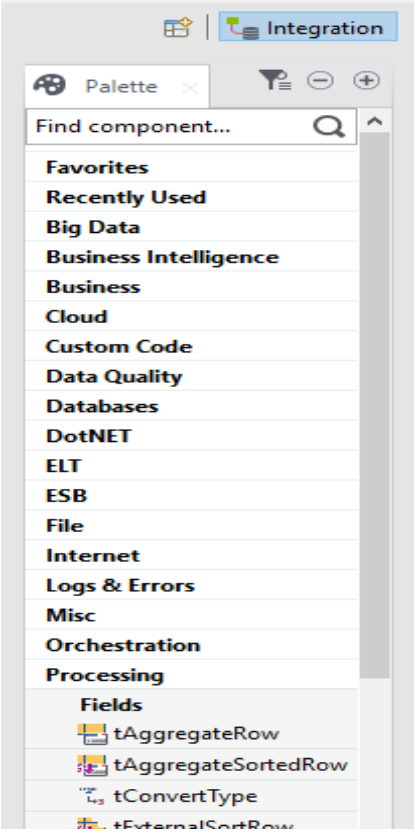


The Job Designs node shows the tree view of the designed Jobs for the current project. Double-click the name of a Job to open it on the design workspace. The Metadata node stores files holding redundant information you want to reuse in various Jobs, such as schemas and property data.

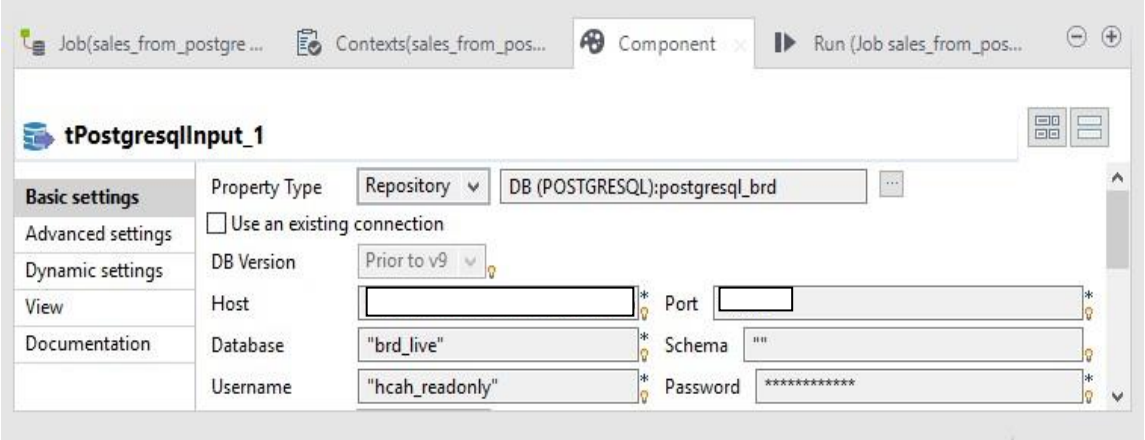
(ii) **Design Workspace:** Located in the top middle portion of the window, it is used to design jobs or business models.



(iii) **Pallete:** Located at the top right corner of the window, it contains all the technical components or shapes, branches and notes which can be dropped to the design workspace for Job design or business modelling.



(iv) **Configuration Tabs:** The configuration tabs are located in the lower half of the design workspace. Each tab opens a view that displays the properties of the selected element in the design workspace. These properties can be edited to change or set the parameters related to a particular component or to the Job as a whole.



3. JOB TABLE

S.No.	Job Name	Description	Tables created in RDS	Frequency
1	Billing_3	Creates Billing_3 table using PCS billing data	Billing_2, Billing_3, First_Package_Dates	Daily
2	BRD_Data_to_RDS	Load sales and product details of BRD from postgres server to MySQL RDS	Sales_detail, product_detail	Daily
3	MR_Table	Creates a table containing MR details from brd postgres server to MySQL RDS	MR_Table	Daily
4	lead_detail	Load crm_lead table from postgres server to MySQL RDS	lead_detail	Daily
5	PCS_monthly_data	Update PCS data on RDS from monthly files on S3 bucket	Billing_snapshot, logistics_monthly, transaction_snapshot	Daily
6	Feedback_data	Load billing_snapshot to Google Drive	None	Daily
7	daily_date	Load daily_date.csv from desktop to RDS	daily_date	One time
8	manufacturer_lookup	Load manufacturer_lookup table to RDS	Manufacturer_lookup	One time
9	Month_year	Load Month_Year table to RDS	Month_Year	One time
10	PCS_billing	Load PCS billing data (snapshot) to RDS	billing_snapshot	One time
11	PCS_transactions	Load PCS transactions data (snapshot) to RDS	transaction_snapshot	One time
12	Service_Package_Lookup	Load Midnight census data to RDS	Midnight_census	One time

CronTab $crontab –e

Three cron jobs are currently scheduled to run

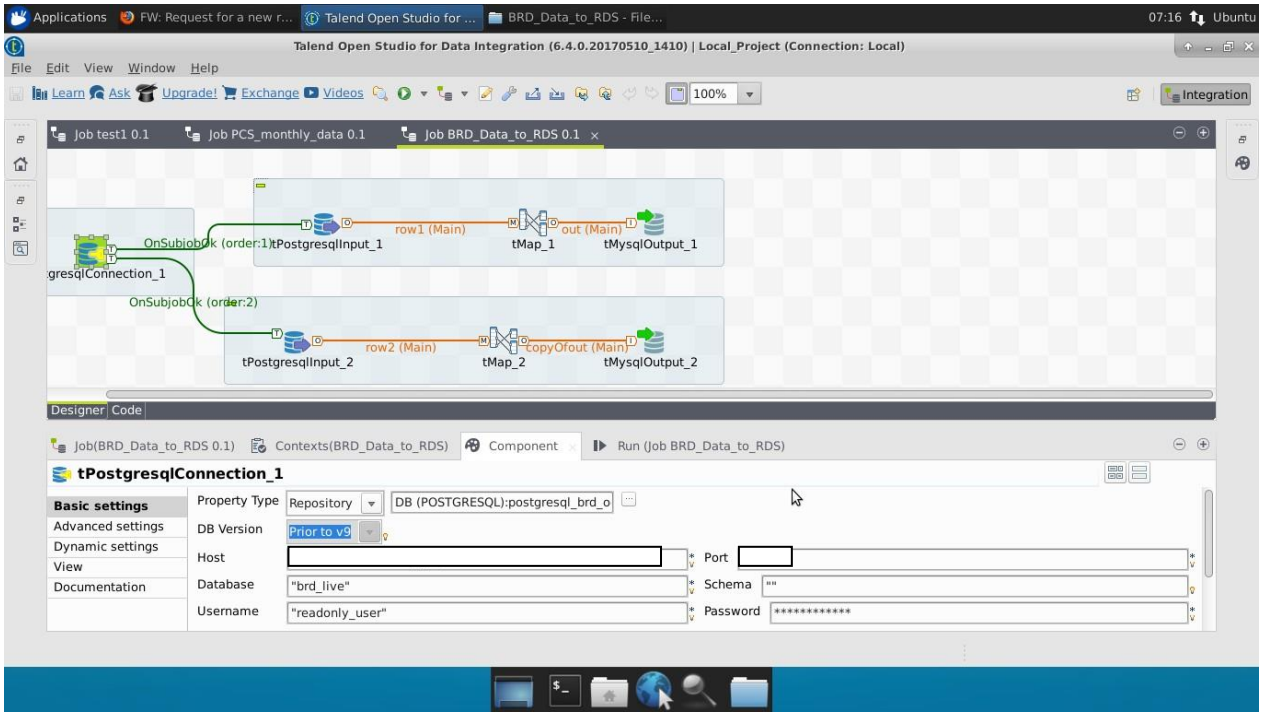
- PCS_monthly data (updates the billing,transactions,logistics and billing_3 tables)
- BRD_data(updates sale_detail,order_detail,MR details)
- Leads_data (updates lead_detail)

4. JOB DESCRIPTION

4.1 Billing_3

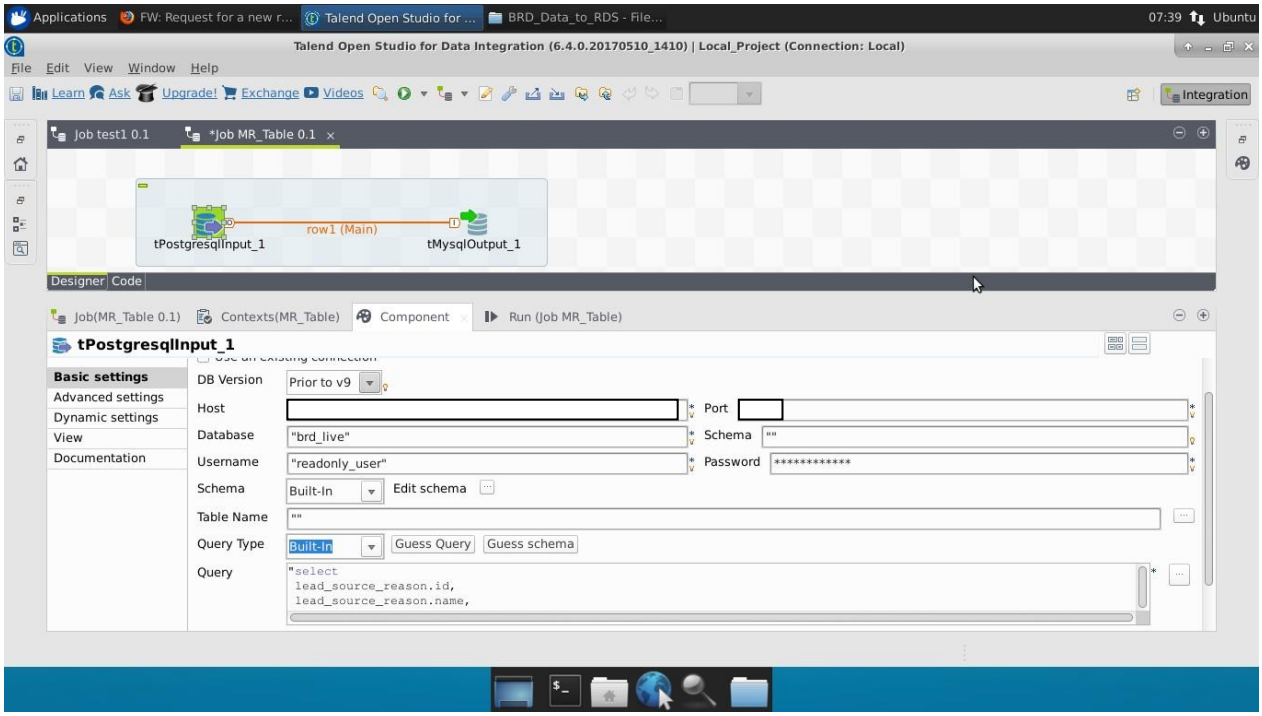
Calls a stored procedure in MySQL RDS server. This Stored procedure in turn generates a Billing_3 table using intermetiade tables such as First_Package_dates, Billing_2

4.2 BRD_Data_to_RDS



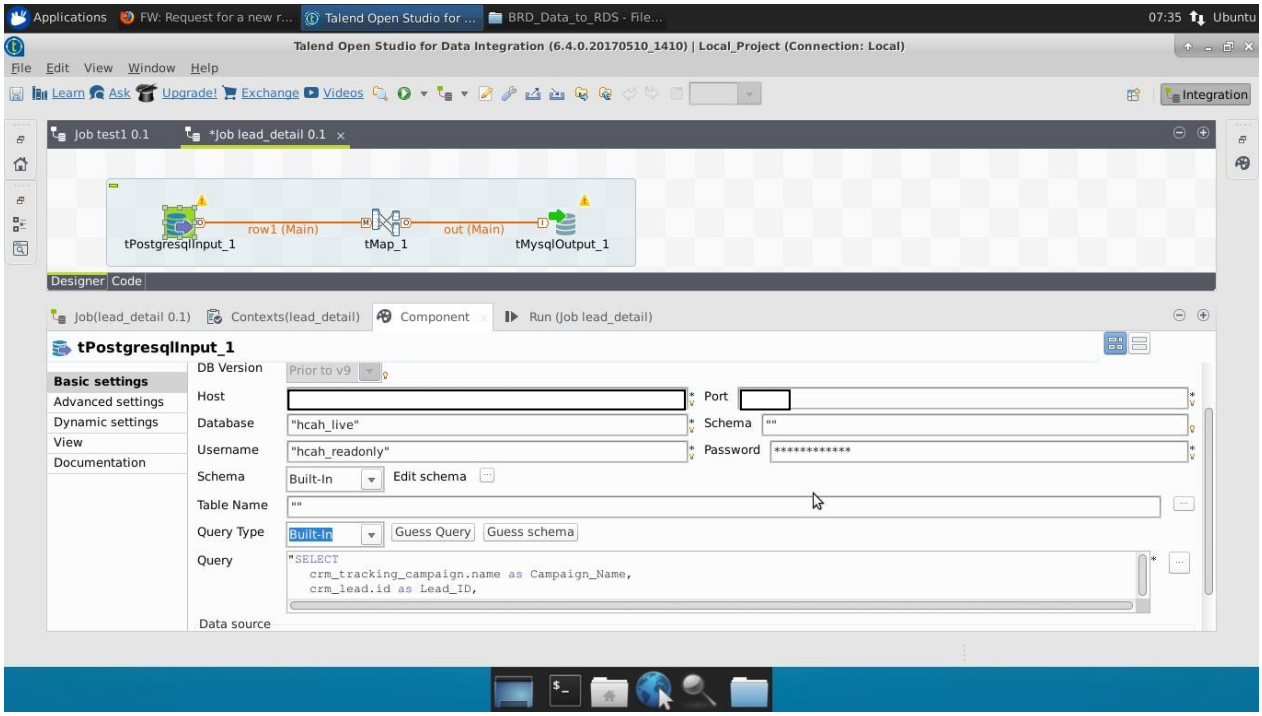
Source: brd_live (Odoo PostGreSQL)
Main Input tables: sale_order, sale_order_line
Other joined tables: hr_employee_service_area, res_partner, product_product, product_template
Queries (in appendix):
sale_detail
product_detail
Destination: Talend_rds (MySQL DWh)
Output Tables: Sales_detail, product_detail

4.3 MR_Table Job



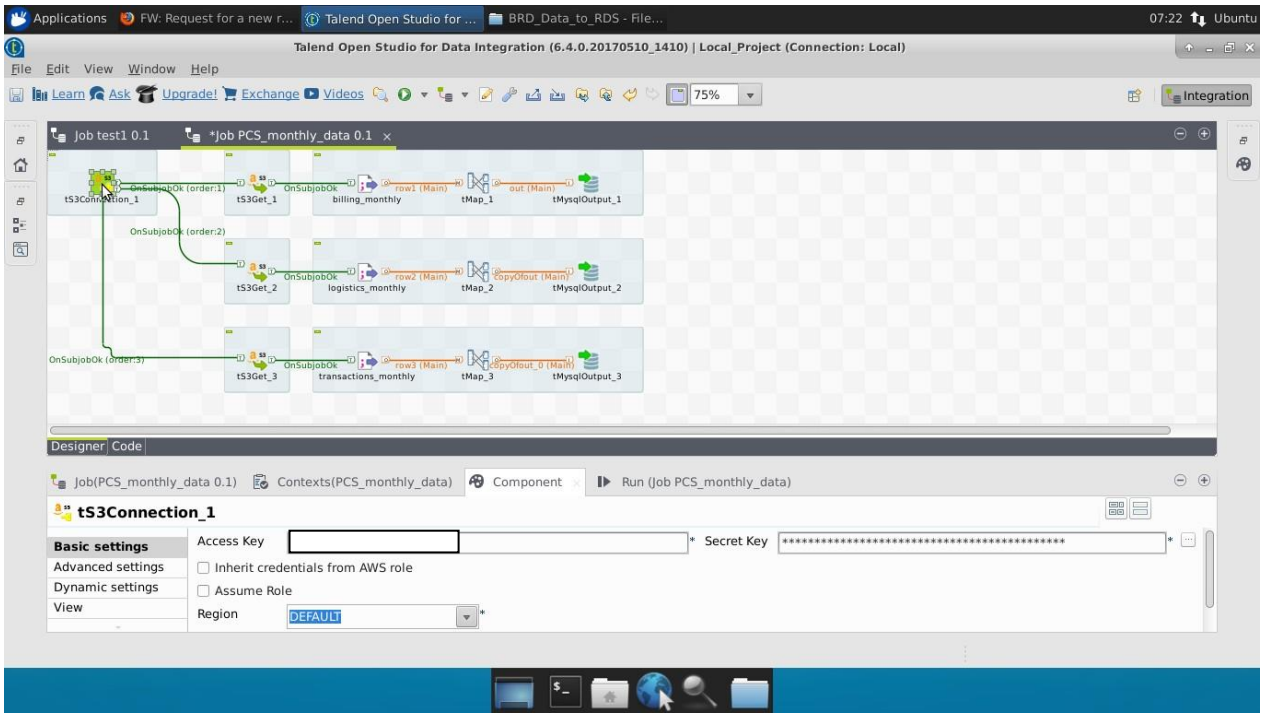
Source: brd_live (Odoo PostGreSQL)
Main Input tables: lead_source_reason
Other joined tables: hr_employee_service_area, res_partner
Queries (in appendix):
MR_Table
Destination: Talend_rds (MySQL DWh)
Output Tables: MR_Table

4.4 Lead_detail Job



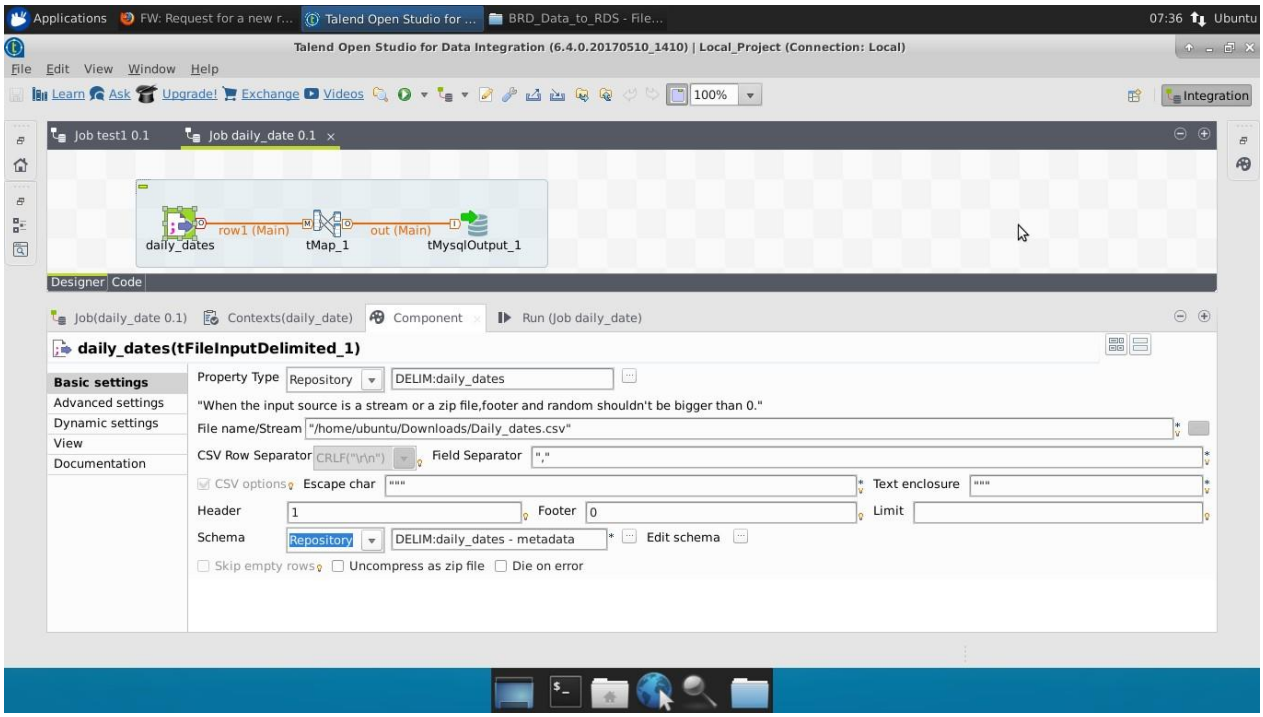
Source: hcah_live (Odoo PostGreSQL)
Main Input tables: crm_lead
Other joined tables: crm_tracking_campaign, status_reason, lead_source_reason, hr_employee_service_area, crm_case_stage
Queries (in appendix):
lead_detail
Destination: Talend_rds (MySQL DWh)
Output Tables: lead_detail

4.5 PCS_monthly_data Job



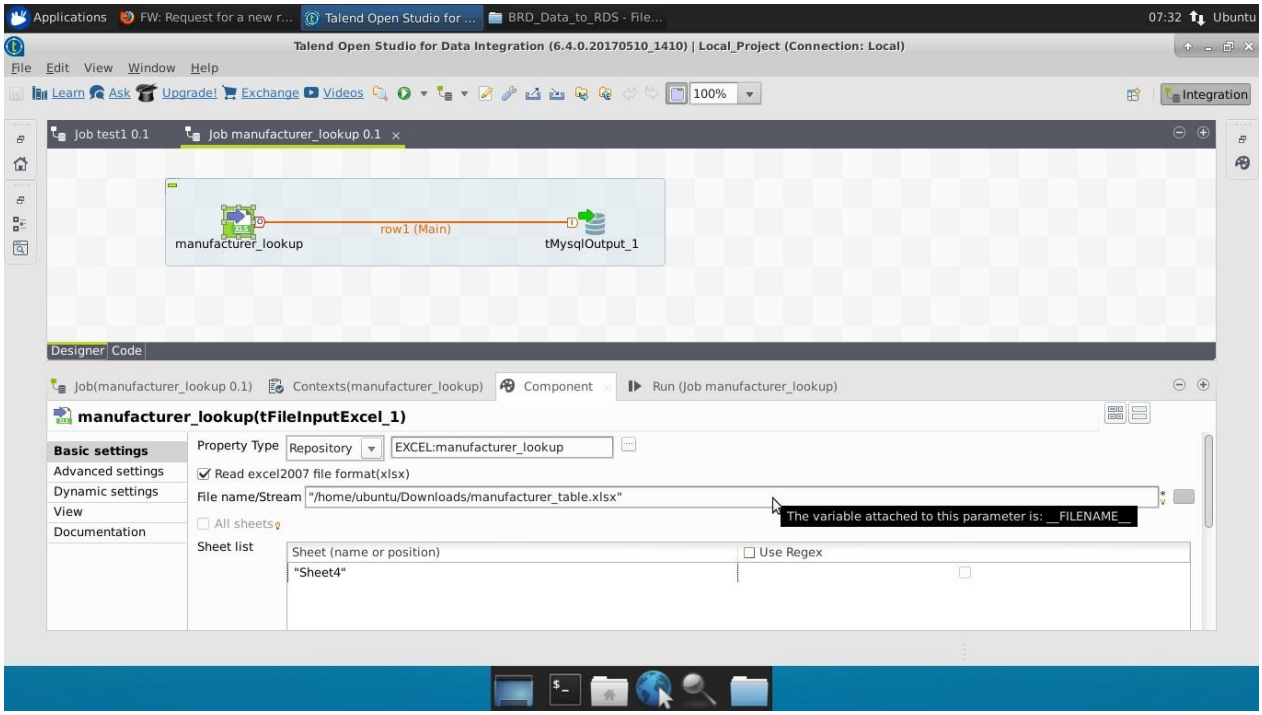
Source: AWS S3 Bucket
Main Input tables: billing_monthly, transactions_monthly, logistics_monthly, (patients_monthly, documents_monthly TBD)
Queries: No query. Select Insert or update action.
Destination: Talend_rds (MySQL DWh)
Output Tables: billing_snapshot, logistics_snapshot, transactions_snapshot, (patients_snapshot, documents_snapshot TBD)

4.6 Daily_date Job



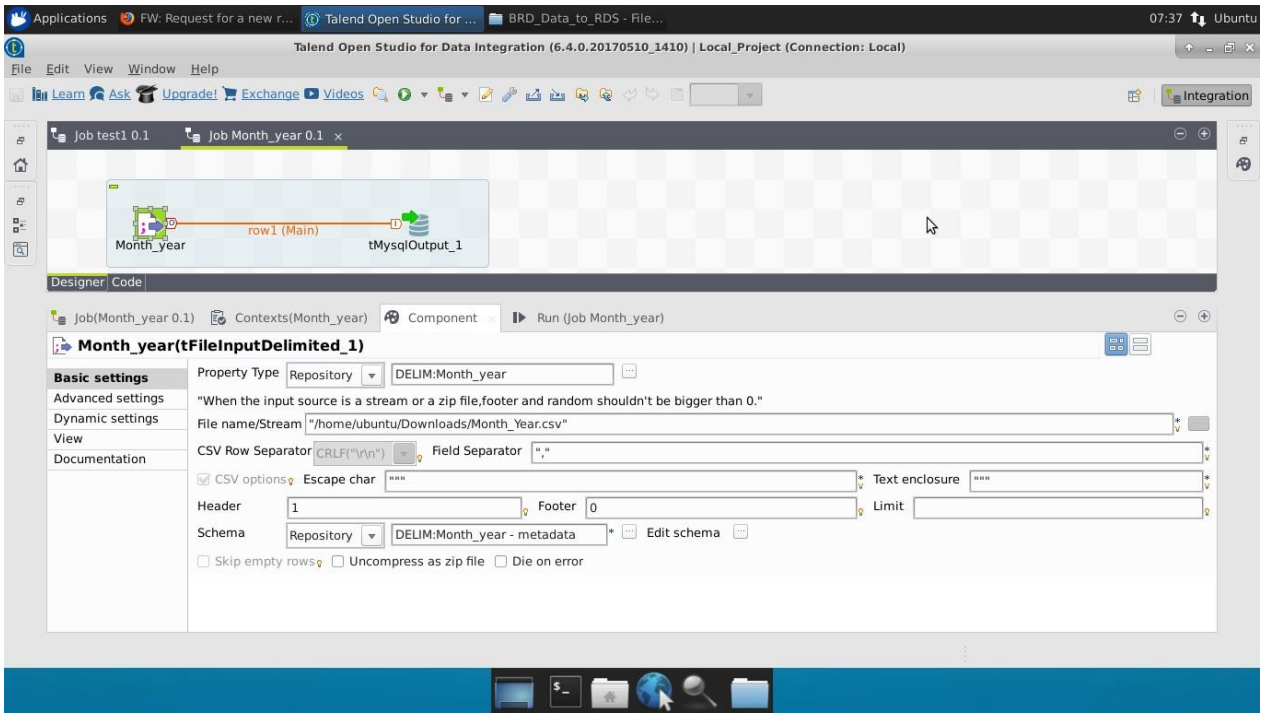
Source: local machine (Ubuntu)
Input tables: daily_dates
Destination: Talend_rds (MySQL DWh)
Output Tables: daily_date

4.7 Manufacturer_lookup Job



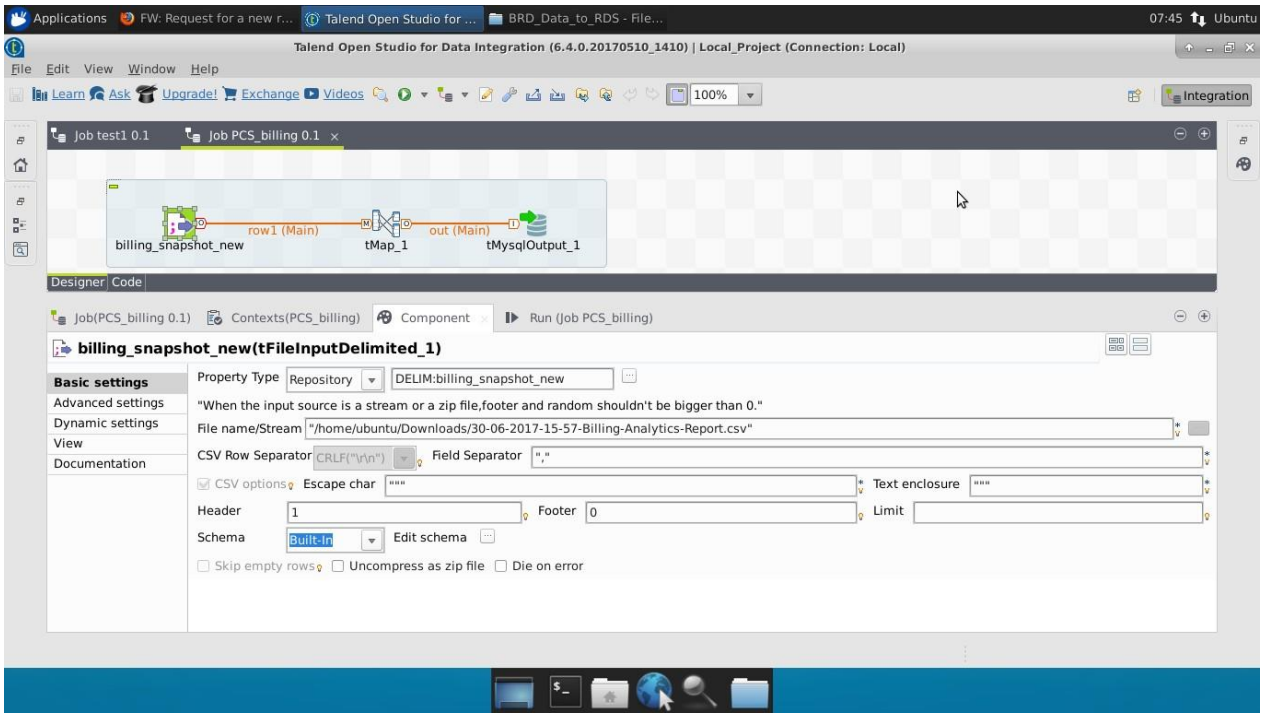
Source: local machine (Ubuntu)
Input tables: manufacturer_lookup (sheet 4)
Destination: Talend_rds (MySQL DWh)
Output Tables: manufacturer_lookup

4.8 Month_year Job



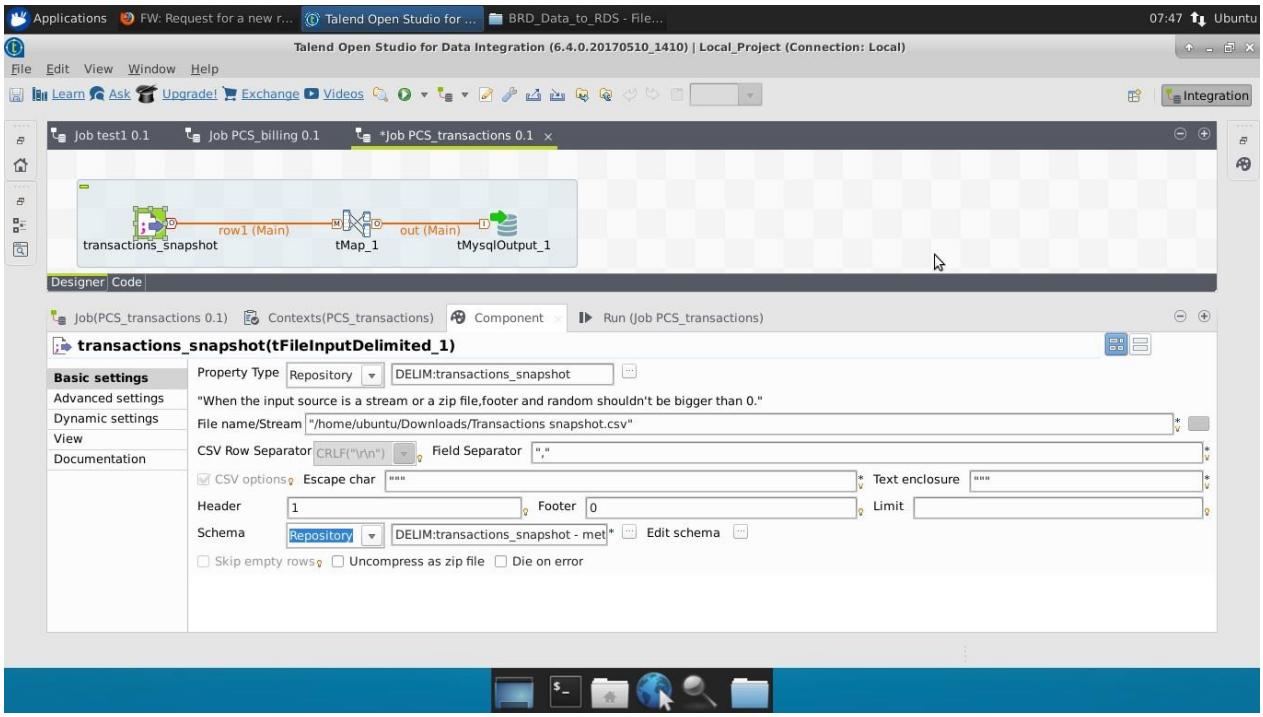
Source: local machine (Ubuntu)
Input tables: Month_Year
Destination: Talend_rds (MySQL DWh)
Output Tables: Month_Year

4.9 PCS_billing



Source: local machine (Ubuntu)
Main Input tables: billing_snapshot
Destination: Talend_rds (MySQL DWh)
Output Tables: billing_snapshot

4.10 PCS_transactions job



Source: local machine (Ubuntu)
Main Input tables: transaction_snapshot
Destination: Talend_rds (MySQL DWh)
Output Tables: transaction_snapshot

5. APPENDIX

5.1 Sale_detail Query

```
select
sale_order.name as Order_Reference,
hr_employee_service_area.name as Service_Area,
sale_order.date_order as Creation_Date,
sale_order.delivery_date as Delivery_Date,
res_partner.id as Customer_ID,
res_partner.name as Customer_Name,
res_partner.hcah_id as Customer_HCAH_ID,
res_partner.mobile as Mobile_No_,
res_partner.is_company as Is_Company,
res_partner.is_droupout as Is_Dropout,
dropout_reason.reason as Dropout_Reason,
sale_order.doc_name as Doctor_Name,
lead_source.name as Lead_Source,
lead_source_reason.name as Source_Info,
lead_source_reason.id as MR_ID,
lead_source_reason.code as MR_Code,
sale_order.amount_total as Total_Revenue,
sale_order.state as Status,
sale_order.amount_untaxed as Revenue_without_tax

from
        sale_order

    LEFT OUTER JOIN
        lead_source ON sale_order.lead_source = lead_source.id

    LEFT OUTER JOIN
        lead_source_reason ON sale_order.source_info = lead_source_reason.id,

        res_partner

    LEFT OUTER JOIN
        dropout_reason ON res_partner.reason_id = dropout_reason.id,

        hr_employee_service_area
     where
        res_partner.id = sale_order.partner_id and
        sale_order.service_area_id = hr_employee_service_area.id;
```

5.2 Product_detail query

```
select
         so.name as Order_reference,
         sa.name as Service_Area,
         pp.name_template as Product_Name,
         manu.name as Manufacturer,
         sol.product_uom_qty as Quantity,
         sol.price_unit as Unit_Price,
         sol.discount as Discount,
         round((sol.product_uom_qty * sol.price_unit) * (100 - sol.discount) / 100,2) AS "Subtotal",
         so.state as Status,
         rp.id as Customer_ID,
         rp.name as Customer_Name
```

```
    from
       res_partner rp,
       res_partner manu,
       sale_order so,
       sale_order_line sol,
       product_product pp,
       product_template pt,
       hr_employee_service_area sa
    where
       manu.id = pt.manufacturer AND
       pp.product_tmpl_id = pt.id AND
       so.partner_id = rp.id AND
       sol.order_id = so.id AND
       pp.id = sol.product_id AND
       so.service_area_id = sa.id;
```

5.3 Lead_detail query

```
SELECT
  crm_tracking_campaign.name as Campaign_Name,
  crm_lead.id as Lead_ID,
  crm_lead.create_date as Creation_Date,
  crm_lead.city as City,
  crm_lead.lead_source as Lead_Source,
  lead_source_reason.name as Source_Info,
  crm_lead.lead_status as Lead_Status,
  status_reason.name Status_Reason,
  crm_lead.service_requirment as Service_Requirement,
  crm_lead.type as Type,
  crm_lead.opportunity_date as Opportunity_Date,
  crm_lead.pcs_patient_id as PCS_Account_Number,
  crm_lead.pcs_service_no as PCS_Service_Number,
  crm_lead.convr_revenue as Converted_Revenue,
  crm_lead.planned_revenue as Planned_Revenue,
  hr_employee_service_area.name as Service_Area,
  crm_case_stage.name as Stage
FROM
  crm_lead
LEFT OUTER JOIN
        crm_tracking_campaign ON crm_lead.campaign_id = crm_tracking_campaign.id
LEFT OUTER JOIN
        status_reason ON crm_lead.status_reason_id = status_reason.id
LEFT OUTER JOIN
        lead_source_reason ON crm_lead.source_info = lead_source_reason.id
LEFT OUTER JOIN
        hr_employee_service_area ON crm_lead.service_area = hr_employee_service_area.id
LEFT OUTER JOIN
       crm_case_stage ON crm_lead.stage_id = crm_case_stage.id;
```

5.3 MR_Table query

```
select
lead_source_reason.id,
lead_source_reason.name,
lead_source_reason.phone,
lead_source_reason.lead_source,
lead_source_reason.code,
hr_employee_service_area.name as Service_Area,
res_partner.name as Company_Name
from lead_source_reason, hr_employee_service_area, res_partner
WHERE
```

res_partner.id = lead_source_reason.company_id AND
lead_source_reason.service_area_id = hr_employee_service_area.id