

Business Intelligence Module

Talend Project

XYZ INSURANCE Datawarehouse

Staging, Loading and Reporting

A report submitted in partial fulfilment of the degree of
MSc in Computing in Big Data Analytics



Department of Computing

Letterkenny Institute of Technology

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Student Name: Yunus Salman Shaik

Date of Submission: 18-12-2019

Signature:

Table of Contents

1	Abstract:.....	4
2	Introduction:	5
2.1	Aim:	5
2.2	Problem scope:.....	5
2.3	Technical review:.....	5
3	Design Requirements.....	6
3.1	Tools:	6
3.2	Schema Design:	6
3.3	Design:	8
4	Implementation:	9
4.1	Metadata in Talend:	9
4.2	Staging:	10
4.3	Data warehouse Loading.....	18
5	Visualization:	31
5.1	Business questions:	32
6	Conclusion:.....	39
7	Bibliography:	40
8	Appendices:.....	41

1 Abstract:

The project is entirely based on the data warehouse and the project is regarding the module of BUSINESS INTELLIGENCE in the MSC Big Data Analytics course. The name of the project is “XYZ INSURANCE”, which is an insurance company. This insurance company is holding all the data in the form of excel files.

The data contains all the information regarding customers, agents, calendar and their location and transactions. The project uses tools like MySQL for creating the database and tables, Talend open studio data integration for creating the metadata and executing the database files. With the help of talend, all the data has been insert, insert or update and update.

All the refined data has been stored into the database, these files are non-readable by humans and thus making decisions with these types of data impossible. So, the tool like Tableau data visualization is used to convert this data in the form of bar graphs, pie charts, etc., All the business questions which were asked previously are answered with the help of MYSQL and also with Tableau data visualization.

2 Introduction:

2.1 Aim:

We are developing a data warehouse project for a company called “XYZ INSURANCE”, from the initial days this insurance company is following a traditional approach to save the data. They have been storing the data in the form of excel files. Excel is the traditional form of storing the data, which is very simple and useful, but the downside is, retrieving the data and refining the data are the most difficult things when the data is stored in the excel files. So, the company needs some useful Business Intelligence tools for refining the data and storing the data in a database, where the data can be easily accessed and analyzed.

The data warehouse project has been divided into three different parts for better utilizing the tools, the first tool we have used is the “talend open studio data integration” which is the most essential, as all the data refining has been done with the help of this talend, the excel files are stored as the metadata files, then the data will be inserted, insert or update and updated, in the talend we will use the slowly changing dimensions (SCD) component because the data in the files need to be refined properly. All the refined data will be stored in the database. These data are further visualized with the tool called “Tableau data visualization”.

2.2 Problem scope:

The financial companies have a huge amount of the data, as the transactions are very huge and the data need to be stored safely. This is a huge challenge for a company like “XYZ INSURANCE” as the majority of the transactions are done by the agents in the insurance companies. All the agents don’t have enough knowledge to store the data in a secure form, so all the data is commonly stored in the form of excel files. But the excel data is not good enough for data refining and decision makings.

2.3 Technical review:

Data is arguably the most valuable currency of the digital age, and insurance companies are sitting on a veritable goldmine. Insurance companies experience a near-constant influx of data — actuarial data, market data, claims data, customer data, and so on. While many businesses in other industries are still figuring out how to derive value from data, the insurance industry has discovered that business intelligence is the key to realizing its potential.

In fact, according to a study from Dresner Advisory Services, the insurance industry is the leader in business intelligence adoption. The study reveals that, when it comes to business intelligence, insurance companies prioritize the following: (‘4 Ways You Can Benefit From Insurance Business Intelligence’ 2019)

- Data warehousing
- Data mining
- Data discovery
- Reporting
- Dashboards
- End user self-services

3 Design Requirements

3.1 Tools:

The tools used for this data warehouse project are:

- MYSQL workbench
- Talend open studio data integration
- Tableau data visualization

The data warehouse project needs these different kinds of tools or applications for making the normal excel data into much more advanced refined data and the visualized data which will further used for companies' growth.

MYSQL workbench:

The tool is designed for a database where the data will be gathered and stored with the help of queries. As the name indicates its structured query language (SQL), the data is stored in the form of tables and all the data refining like insert, update and delete are done with the help of the tool.

Talend open studio data integration:

As the name indicates this tool is used for data integration, Talend is connected with the database of the MYSQL workbench where the tables can retrieve and store the data. All the data which are in the form of excel can be added as metadata files and there are different components like input, output and mapping which will help in data refining. Slowly changing dimensions are the most advanced feature in the talend, which are used for advanced data integration.

Tableau data visualization:

As the name indicates this tool is used for visualization, the data generated by the talend are stored in the database, but these refined data are not enough for decision making by the company officials. So, the tableau data visualization tool is used for visualizing the refined data and used to answer all the business questions. These visualized data are enough for decision making. ('What is Tableau? Uses and Applications' 2019)

3.2 Schema Design:

The schema design of the insurance company has MySQL related queries, as the data of the company like customer data, agent, location, calendar, and transaction are stored in the form of tables in the workbench which are further accessed by the "talend open studio data integration" tool for insert, insert or update and update operations the end results (data) will be stored in the database, further the refined data is visualized by the tableau data visualization tool.

Model:

For any company designing a proper model is most important, as the stages are involved in the model, we have designed a model based upon the star schema design. Policy_fact is the main table that is linked to the other table. Surrogate keys in the policy tables and primary keys in other tables are linked with each other. The connections are in the form of one to one and one to many.

The below figure will represent our data warehouse model.

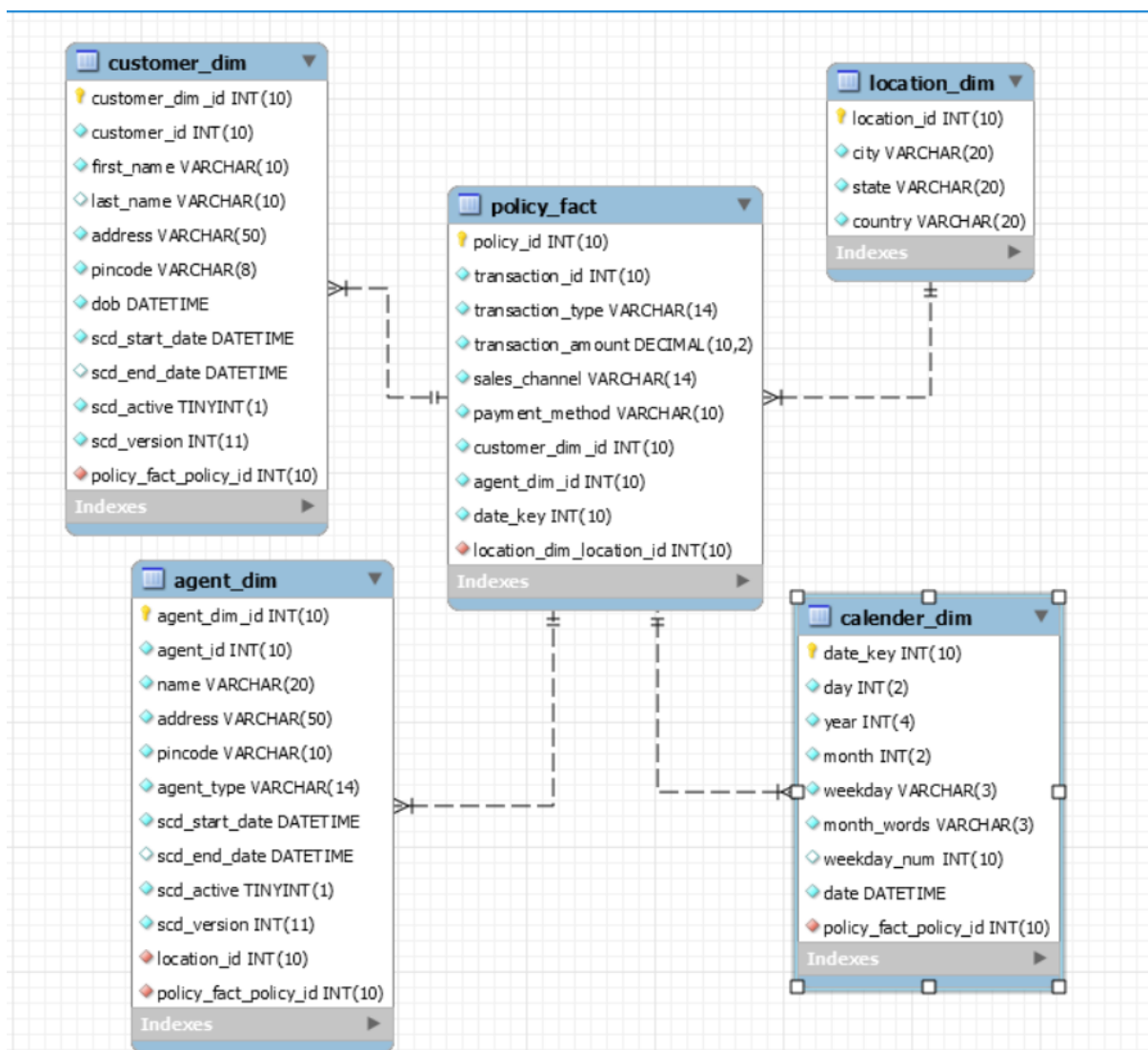


Figure 1: ER diagram

3.3 Design:

The design of the data warehouse project has been defined in the structural form:

As shown in the figure below;

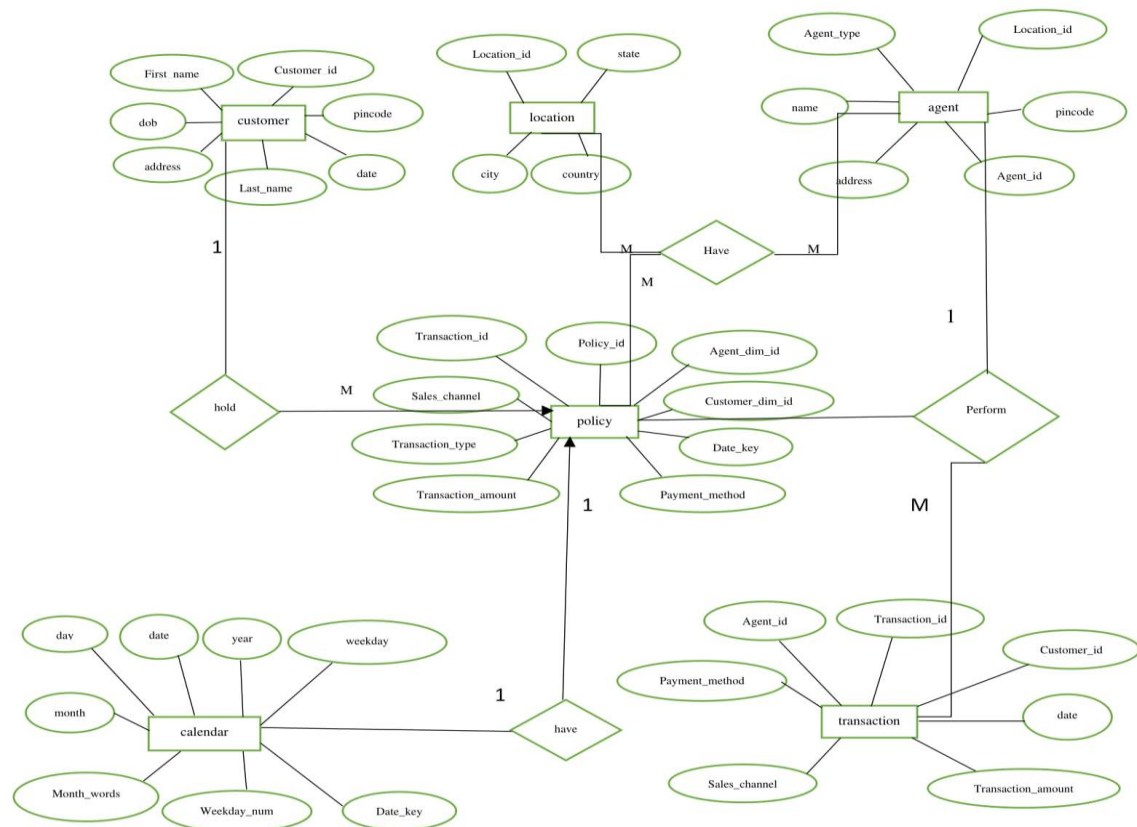


Figure 2: Data warehouse design

4 Implementation:

The implementation of the data warehouse project for the “XYZ INSURANCE” company is staged into different sectors. We will discuss them briefly:

4.1 Metadata in Talend:

Metadata in the Talend open studio data integration is most important because all the data (Excel files) of the company are saved in the form of “metadata” in talend. The metadata is essential data, which are further refined with the help of creating the jobs adding the components and with the help of slowly changing the dimension component, all the data is refined properly. The metadata won’t change throughout the project, changes in the metadata might be a bad decision as each and every company wants to save the original raw data for further use.

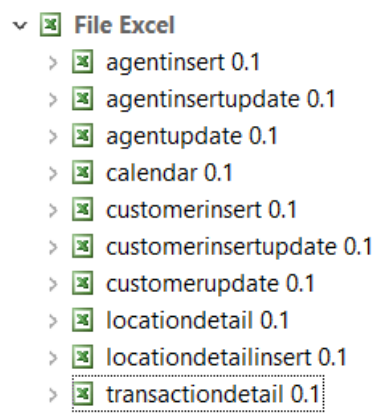


Figure 3:Metadata files

Data warehouse tables:

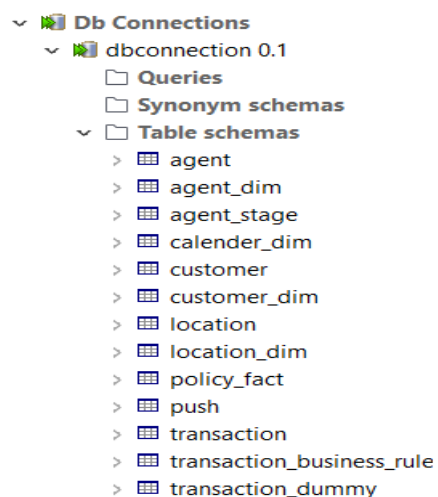


Figure 4:Data warehouse tables

4.2 Staging:

stage loading is the initial feature in the talend. In this step, each and every metadata is added into the component which is a “tFileInputExcel” and with the help of tMap component, all the data get refined as insert, insert or update and update. Then the data is pushed into the output component, which is “tDBOutput”. In the slowly changing dimension (SCD) the output file is taken and it further get refined with the help of “tDBSCD” component, wherewith the help of fields like type 0, type 1, type 2 and type 3, all the data will be properly refined. The refined data is pushed into dimension tables for further use of visualization.

1) Customer data:

The customer data has been refined in three ways insert, insert or update and update.

A job will be created for each and every operation and the different types of components are used. They are:

- tFileInputExcel
- tMap and
- tDBOutput

we will read the metadata file into the “tFileInputExcel” and with the help of “tMap” component. The tfileinputexcel and tdboutput file will be mapped with the help of tmap component and all the columns in these two components will be mapped and executed.

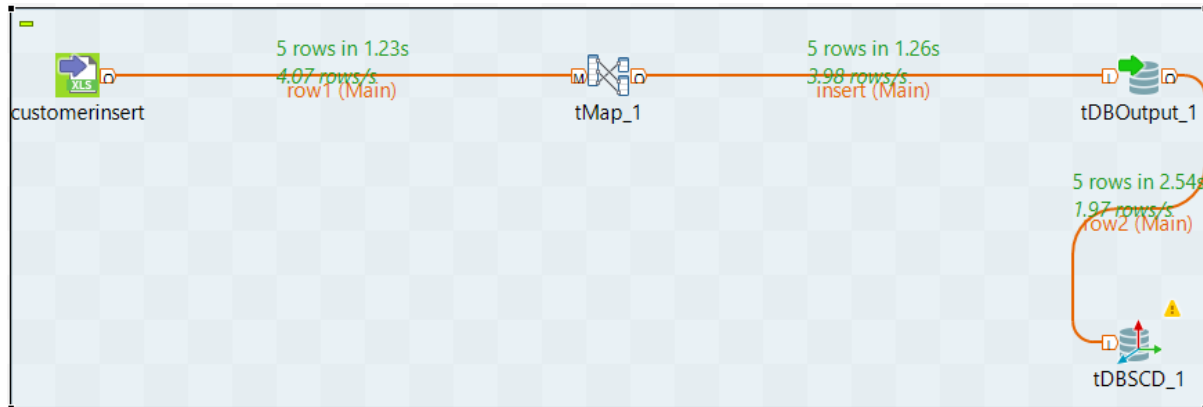


Figure 5:customer insert table

We have chosen the option of “insert” in the insert job for the “tDBOutput” component.

The screenshot shows the 'tDBOutput_1(MySQL)' configuration window. The 'Basic settings' tab is active. The 'Database' is set to 'MySQL'. The 'Property Type' is 'Repository' with a value of 'DB (MYSQL):dbconnection'. The 'DB Version' is 'Mysql 8'. The 'Host' is empty, and the 'Port' is '3306'. The 'Database' is 'salmaninsurance', the 'Username' is 'root', and the 'Password' is masked with '*****'. The 'Table' is 'customer'. The 'Action on table' is 'Default' and the 'Action on data' is 'Insert'. The 'Schema' is 'Built-In'.

Figure 6: Insert field in the customer insert

For the “insert or update” in the job we need to choose the same in the “tDBOutput” component.

The screenshot shows the 'tDBOutput_1(MySQL)' configuration window. The 'Basic settings' tab is active. The 'Database' is set to 'MySQL'. The 'Property Type' is 'Built-In'. The 'DB Version' is 'Mysql 8'. The 'Host' is empty, and the 'Port' is '3306'. The 'Database' is 'salmaninsurance', the 'Username' is 'root', and the 'Password' is masked with '*****'. The 'Table' is 'customer'. The 'Action on table' is 'Default' and the 'Action on data' is 'Insert or update'. The 'Schema' is 'Built-In'.

Figure 7: insert or update field in the customer insert or update component.

We have chosen the option of “update” in the update job for the “tDBOutput” component.

The screenshot shows the 'tDBOutput_1(MySQL)' configuration window. The 'Basic settings' tab is active. The 'Database' is set to 'MySQL'. The 'Property Type' is 'Built-In'. The 'DB Version' is 'Mysql 8'. The 'Host' is empty, and the 'Port' is '3306'. The 'Database' is 'salmaninsurance', the 'Username' is 'root', and the 'Password' is masked with '*****'. The 'Table' is 'customer'. The 'Action on table' is 'Default' and the 'Action on data' is 'Update'. The 'Schema' is 'Built-In'.

Figure 8: update field in the customer update component.

The tMap is used to cleanse the data, which can be seen in figure 5. The name field and address field in the source file has a trailing ‘!’, this was trimmed using:

First name field has some trailing spaces between the names so we right the code in expression builder to remove the trailings: `row1.customer_first_NAME.trim()`

Last name field has some trailing ‘!’ so we right the code in expression builder to remove the trailings: `row1.customer_last_NAME.replace("!", "")`

The address field has some trailing so we right the code in expression builder to remove the trailings: `row1.customer_ADDRESS.replace("#", "")` in a variable map expression in the tMap component.

All the column names in input and output can be seen in figure 9. And automap is successfully mapped the components.

row1		Var	insert	
Column			Expression	Column
CUSTOMER_ID			row1.CUSTOMER_ID	customer_id
CUSTOMER_FIRST_NAME			row1.CUSTOMER_FIRST_NAME.trim()	first_name
CUSTOMER_LAST_NAME			row1.CUSTOMER_LAST_NAME.replace("!", "")	last_name
CUSTOMER_ADDRESS			row1.CUSTOMER_ADDRESS.replace("#", "")	address
PINCODE			row1.PINCODE	pincode
DATE			row1.DATE	date
CUSTOMER_DOB			row1.CUSTOMER_DOB	dob

Figure 9:tmap editor for customer table

2) Location data:

The location data has been refined in two ways insert, and update.

Three types of components are used in the location data table, they are:

- tFileInputExcel
- tMap and
- tDBOutput

we will read the metadata file into the “tFileInputExcel” and with the help of “tMap” component. The tfileinputexcel and tdboutput file will be mapped with the help of tmap component and all the columns in these two components will be mapped and executed.

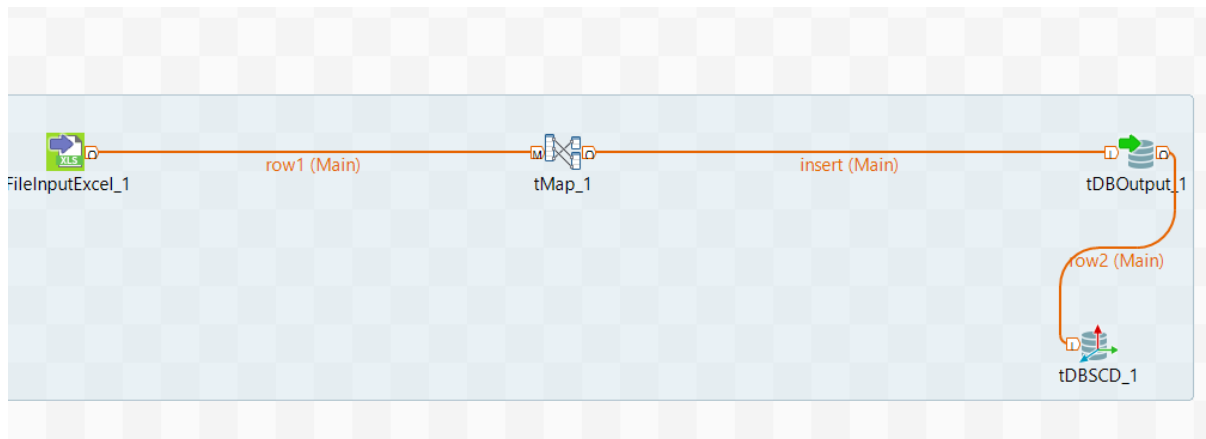


Figure 10:location insert job

The tMap is used to cleanse the data. The country field in the given excel file has a trailing ‘,’ , this was trimmed.

Country: row1.COUNTRY.replace(", ", "") in a variable map expression in the tMap component.

row1	Var	insert
Column		Expression
LOCATION_ID		row1.LOCATION_ID
CITY		row1.CITY
STATE		row1.STATE
COUNTRY		row1.COUNTRY.replace(", ", "")
		Column
		location_id
		city
		state
		country

Figure 11:tMap editor for location table

The location table have the insert field where the tDBOutput has been given as ‘insert’.

tDBOutput_1(MySQL)	
Basic settings	Database: MySQL Apply
Advanced settings	Property Type: Built-In Save
Dynamic settings	DB Version: Mysql 8
View	<input type="checkbox"/> Use an existing connection
Documentation	Host: "" Port: "3306"
	Database: "salmaninsurance"
	Username: "root" Password: "*****"
	Table: "location"
	Action on table: Default Action on data: Insert
	Schema: Built-In

Figure 12:insert option for insert job for location

The location table has the update field where the tDBOutput has been given as ‘update’.

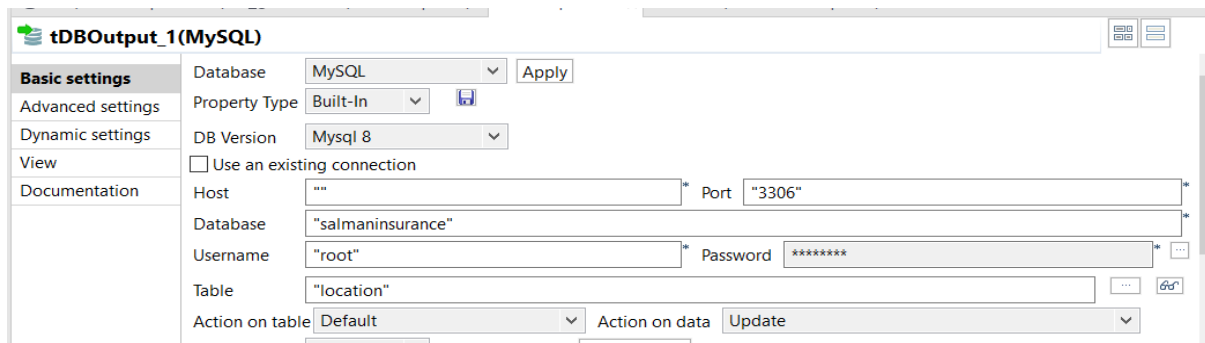


Figure 13: update option for update job for location

3) Agent data:

The agent data has been refined in three ways insert, insert or update and update.

Three type of components are used in the agent data table, they are:

- tFileInputExcel
- tMap and
- tDBOutput

we will read the metadata file into the “tFileInputExcel” and with the help of “tMap” component. The tfileinputexcel and tdboutput file will be mapped with the help of tmap component and all the columns in these two components will be mapped and executed.

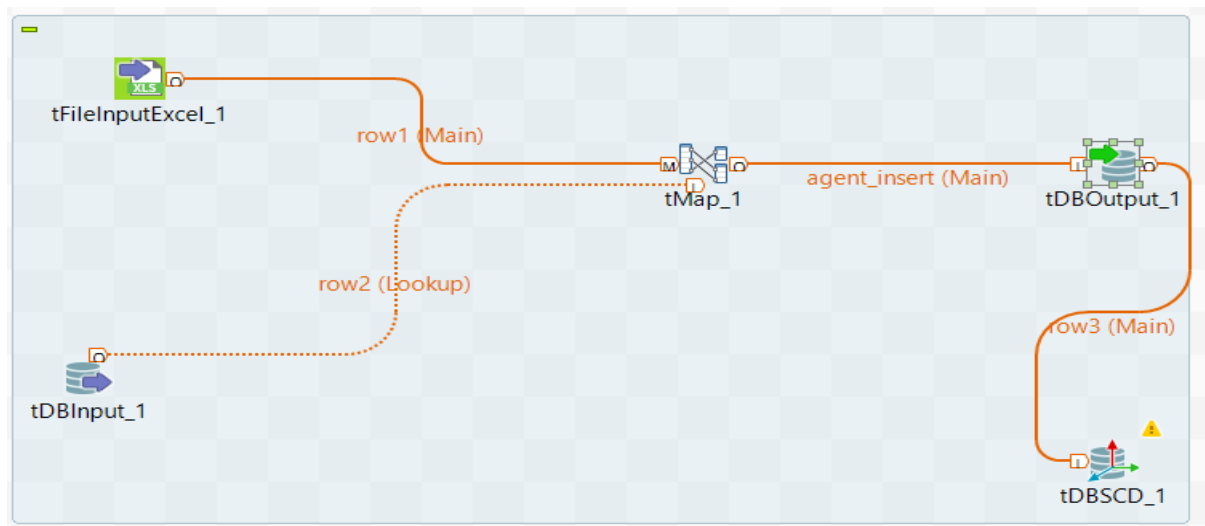


Figure 14:job created for the agent data.

The address field in the source file has a trailing '#', this was trimmed using

-Address: row1.customer_ADDRESS.replace("#", "") in a variable map expression in the tMap component.

The tmap component has the two input files and an output file. The input file has a foreign key which were mentioned in the tmap as "row1.LOCATION_ID"

The 'match model' has either 'All matches' or 'unique match' to be selected, we have opted for the 'all matches'.

Further the two input rows are matched with the output row.

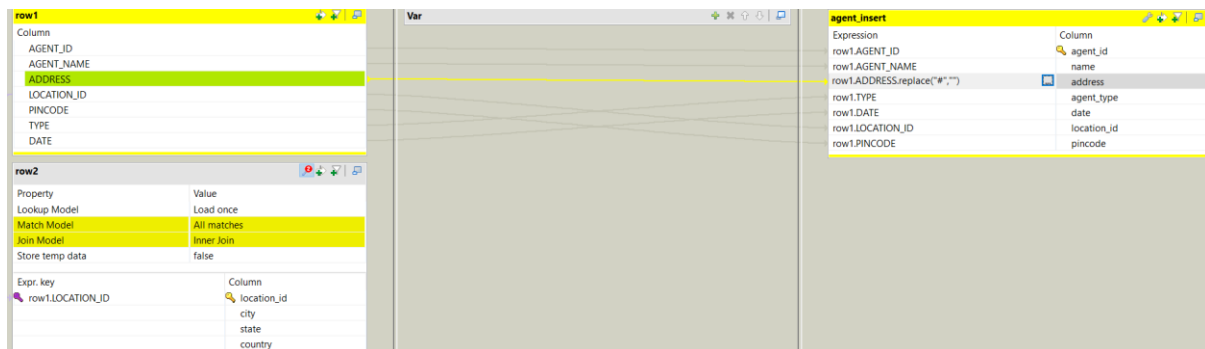


Figure 15:tmap editor for agent data

The agent table have the insert field where the tDBOutput has been given as 'insert'.

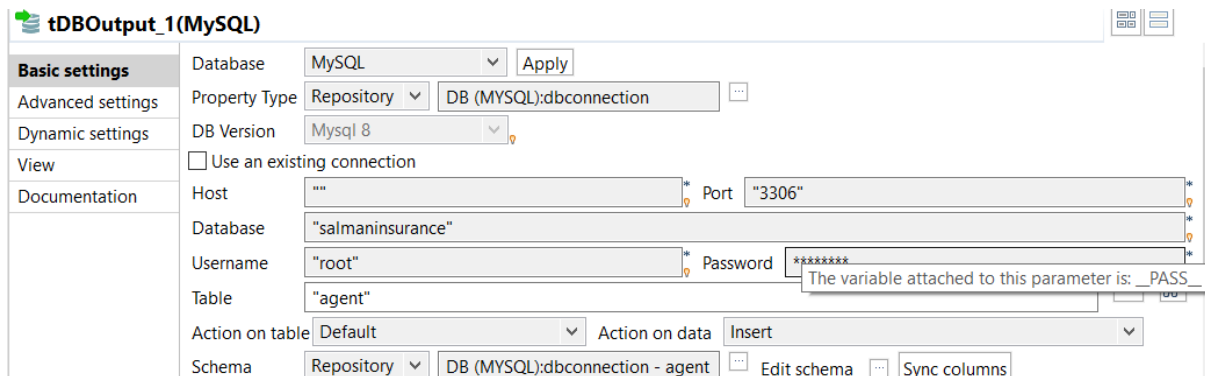


Figure 16: insert option for the insert job for the agent data.

The agent table have the insert or update field where the tDBOutput has been given as 'insert or update'.

Figure 17: insert or update option for the insert or update job for the agent data.

The agent table have the insert or update field where the tDBOutput has been given as ‘insert or update’.

Figure 18: update option for the update job for the agent data.

4) Transaction data:

Transaction table uses the four components in the talend, they are:

- tFileInputExcel
- tMap
- two tDBInput and
- tDBOutput.

tDBInput 1: customer data

tDBInput 2: Agent data

tDBOutput: transaction table

we will read the metadata file into the “tFileInputExcel” and with the help of “tMap” component, and the two tDBInput files are linked with the tmap. The tfileinputexcel and tdboutput file will be mapped with the help of tmap component and all the columns in these two components will be mapped and executed.

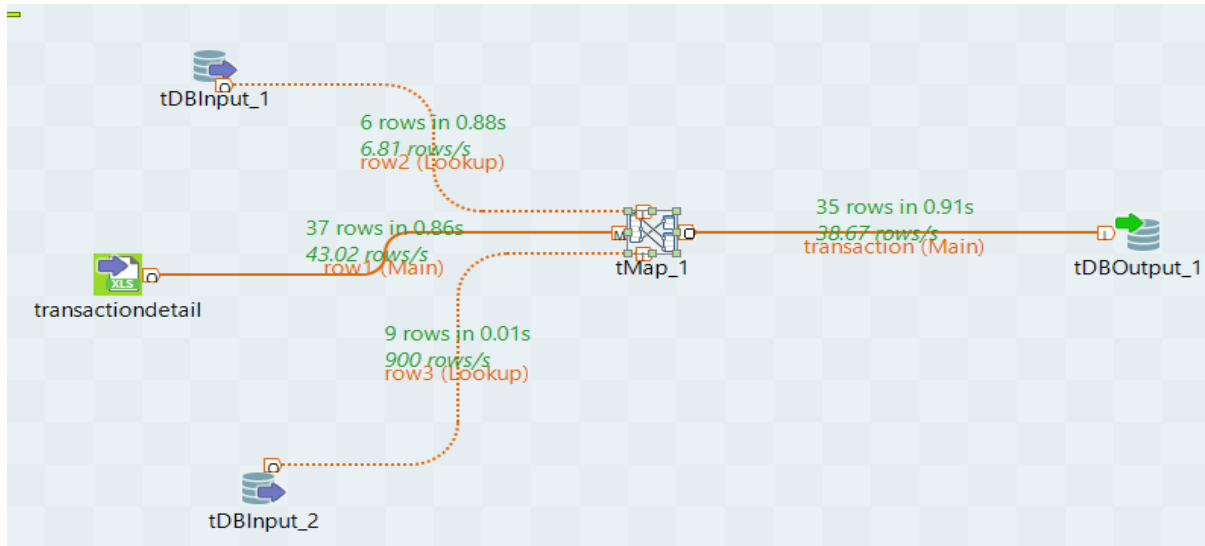


Figure 19: job for transaction data

tmap component has to be sorted in the way that its primary key is defined as 'transaction_id', the transaction table has two foreign keys

'row1.customer_id' and 'row1.agent_id'

all these are foreign keys that are dragged from the transaction table. All the match models are selected as 'unique match' and the join model is selected as 'inner join' in other rows. Then the output row is mapped with the help of the input rows after both the rows are mapped.

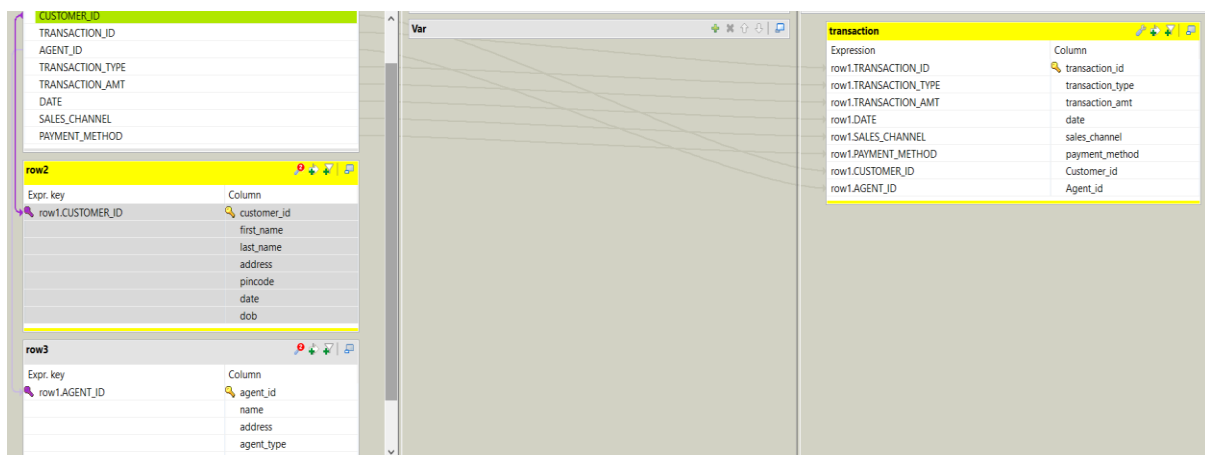


Figure 20: tMap editor for transaction data

We have done with the staging tables and all the refine operations like insert, insert or update and update have been done in the staging.

The next phase is to do the dimension tables, this is the essential element as the proper refinement will be done in this phase. Slowly changing dimensions are the helpful component in the talend for data refinement with fields like type 0, type 1, type 2 and type 3.

Type 0: it's a fixed dimension field, which stays constant.

Type 1: the history won't store in this field; every time new data is updated.

Type 2: it simply mentioned as a “row versioning” with the active dates.

Type 3: it has the previous data, which will add as a separate column with the existing data.(‘Introduction to Slowly Changing Dimensions (SCD) Types - Adatis’ 2019)

4.3 Data warehouse Loading

Dimension tables:

Customer:

The process is similar as staging but we will add an extra slowly changing dimension (SCD) component for the job.

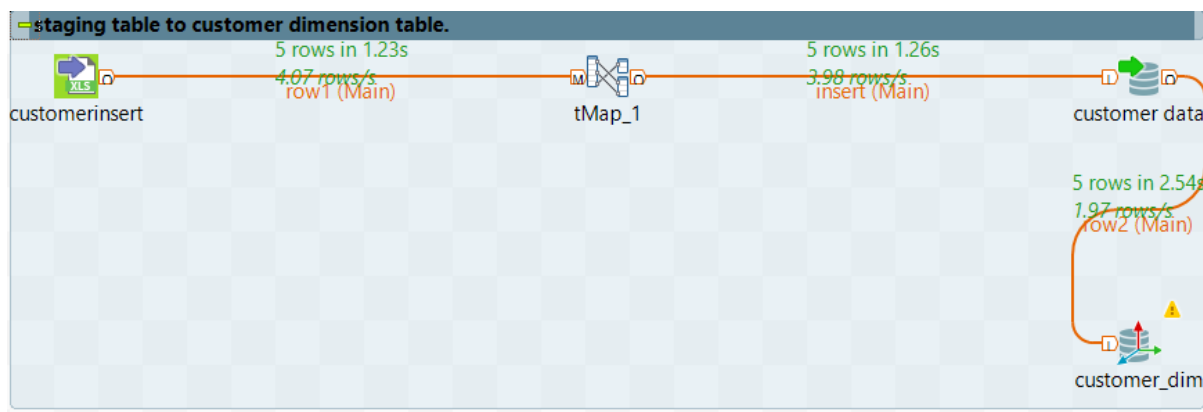


Figure 21: job created with the SCD component for customer.

As given in the business requirements for the company requirement we will choose the type 2 field for the customer dimension table.

To restore the previous data for the first_name, last_name, address, dob and Pincode we will add these components into the type 2 filed. Which will also generate the new data and also keep the historical data.

The surrogate key in SCD will act as a primary key, which is, in this case, is “customer_dim_id” and the source key will act as a foreign key, which is “customer_id”.

We need to tick the options for the “scd_version” and “scd_active” and mention the other types as “scd_start_date” and “scd_end_date”.

We can see all the options and fields in the figure below.

<input type="text"/> filter	
Unused	
date	
Source keys	
customer_id	
Surrogate keys	
name	customer_dim_id
creation	Auto increment
complement	

Type 0 fields			
Type 1 fields			
Type 2 fields			
first_name			
last_name			
address			
pincode			
dob			
Versioning			
type	name	creation	compleme...
start	scd_start_date	Job start time	
end	scd_end_date	NULL	
<input checked="" type="checkbox"/> versi...	scd_version		
<input checked="" type="checkbox"/> active	scd_active		
Type 3 fields			
current value	previous value		

Figure 22: SCD component for the customer dimension table.

Records:

For insertion:

customer_id	first_name	last_name	address	pincode	date	dob
12345	FIRNAME	A!!	#123,1ST MAIN, 2ND CROSS#	520083	2010-01-01	1977-04-04
678910	FNAME	B!!	#342,2ND MAIN, 15TH CROSS#	520084	2011-01-02	1971-01-01
11121314	FIRNAME	C!!	#32, 1ST MAIN, 1ST CROSS#	520085	2010-01-03	1979-01-01
15161718	FAME	D!!	#151, 15TH MAIN, 39TH CROSS#	520086	2013-01-04	1978-01-01
19202122	FIRSTAME	E!!	#155, 2ND CROSS,11TH AVENUE#	520087	2012-05-04	1984-01-01

Figure 23: Insertion data for the customer table.

As we can see in the image above, the last name has “!” and address field have “#”, so we need to delete this and update.

For updating:

After applying the trim() function for the first name, replace(“!”, “”) for the last name and replace(“#”, “”) for the address function in the expression builder.

We will save the job and run again.

```
mysql> select * from customer;
```

customer_id	first_name	last_name	address	pincode	date	dob
12345	FIRNAME	A	123,1ST MAIN, 2ND CROSS	520083	2010-01-01 00:00:00	1977-04-04
678910	FNAME	B	342,2ND MAIN, 15TH CROSS	520084	2011-01-02 00:00:00	1971-01-01
11121314	FIRNAME	C	32, 1ST MAIN, 1ST CROSS	520085	2010-01-03 00:00:00	1979-01-01
15161718	FAME	D	151, 15TH MAIN, 39TH CROSS	520086	2013-01-04 00:00:00	1978-01-01
19202122	FIRSTAME	E	155, 2ND CROSS,11TH AVENUE	520087	2012-05-04 00:00:00	1984-01-01

Figure 24:updating the record in the customer table.

Location dimension table:

The location_dim job will take the four components which are

- tfileinputexcel
- tmap
- tdboutput and
- tdbscd

fields, the SCD field will have type 1 field to be filled. Which indicated the current data.

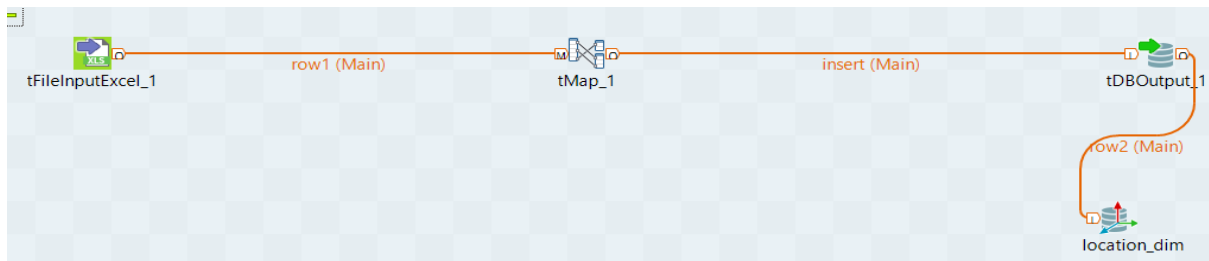


Figure 25:location dimension table job

As this is a type 1 field, we will add the city, state and country components into the type 1 field.

The surrogate key is “location_dim_id”.

The source key is “location_id”.

Unused		Type 0 fields																									
Source keys		Type 1 fields																									
location_id		<div>city</div> <div>country</div> <div>< [] ></div>																									
Surrogate keys		Type 2 fields																									
name	<input type="text" value="location_dim_id"/>																										
creation	<input type="text" value="Auto increment"/> ▼	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="4" style="background-color: #ffff00;">Versioning</th> </tr> <tr> <th>type</th> <th>name</th> <th>creation</th> <th>compleme...</th> </tr> </thead> <tbody> <tr> <td>start</td> <td>scd_start</td> <td>Job start time ▼</td> <td></td> </tr> <tr> <td>end</td> <td>scd_end</td> <td>NULL ▼</td> <td></td> </tr> <tr> <td><input type="checkbox"/> versi...</td> <td>scd_version</td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/> active</td> <td>scd_active</td> <td></td> <td></td> </tr> </tbody> </table>		Versioning				type	name	creation	compleme...	start	scd_start	Job start time ▼		end	scd_end	NULL ▼		<input type="checkbox"/> versi...	scd_version			<input type="checkbox"/> active	scd_active		
Versioning																											
type	name	creation	compleme...																								
start	scd_start	Job start time ▼																									
end	scd_end	NULL ▼																									
<input type="checkbox"/> versi...	scd_version																										
<input type="checkbox"/> active	scd_active																										
complement																											
		Type 3 fields																									
		<div>current value</div> <div>previous value</div>																									

Figure 26:scd editor for the location dimension.

Agent dimension table:

The agent_dim will use the five components

- tFileInputExcel,
- tDBInput,
- tMap,
- tDBOutput and
- tDBSCD

Which has the agentinsert metadata, tDBInput has the “location data”, tDBOutput will be logged with agent table and the SCD will be added at the end which will store the agent_dim data.

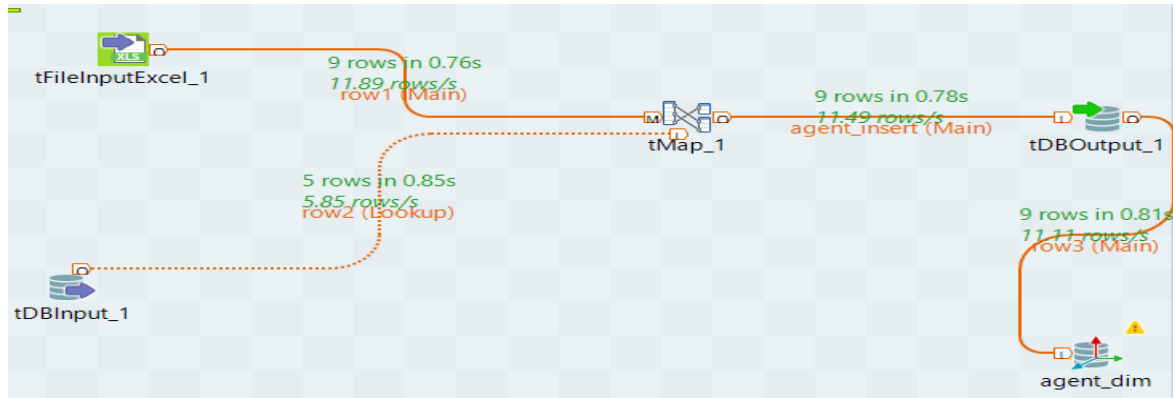


Figure 27:agent dimension table job.

The tmap is used to auto map the components of the input and output rows when the rows are mapped with each other, then the output will be generated.

The agent_dim has the foreign key as “location_id” which is dragged and dropped in row 2.

The address filed in the agent_insert has the “#” at the start. So with the use of the replace function, we will change as row1.ADDRESS.replace(“#”, “ “).

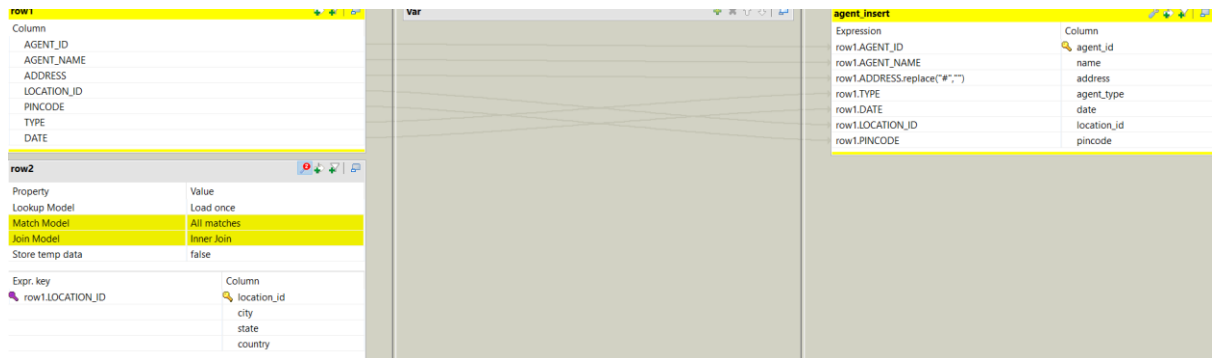


Figure 28:tMap component for agent dimension table.

The matching model is changed to all matches and the join model is changed to an inner join.

The SCD editor will use the type 2 filed mostly, which will keep the new data and also keep the historical data.

The components like address, agent_type, name and pincode has been added to the type 2 field. Whereas the “location_id” need not to be changed so it is kept under type 0 field.

The surrogate key is “agent_dim_id”

The source key is “agent_id”.

Unused	
date	

Source keys	
agent_id	

Surrogate keys	
name	agent_dim_id
creation	Auto increment ▼
complement	

Type 2 fields				
location_id				

Versioning				
type	name	creation	completion	
start	scd_start_date	Job start time	▼	
end	scd_end_date	NULL	▼	
<input checked="" type="checkbox"/> versi...	scd_version			
<input checked="" type="checkbox"/> active	scd_active			

Type 3 fields	
current value	previous value

Figure 29:SCD component for agent dimension table.

Dimension table:

The job has been taken three components like

- `tfileinputexcel`,
- `tmap` and
- `tdboutput`.

The calendar dimension table should be directly loaded from the supplied data file.

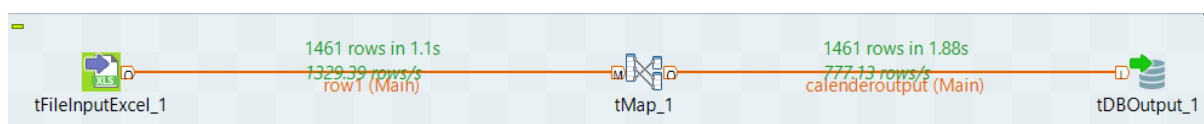


Figure 30:calendar dimension table job

The tmap editor of the calendar dimension table can be seen in the below figure.

row1	Var	calenderoutput	
Column		Expression	Column
DATE		row1.DATE	DATE
WEEKDAY_NUM		row1.WEEKDAY_NUM	WEEKDAY_NUM
YEAR		row1.YEAR	YEAR
MONTH		row1.MONTH	MONTH
DAY		row1.DAY	DAY
WEEKDAY_WORDS		row1.WEEKDAY_WORDS	WEEKDAY_WORDS
MONTH_WORDS		row1.MONTH_WORDS	MONTH_WORDS

Figure 31:tmap editor for the calendar dimension table.

Business rule:

Transaction business rule is specially defined in the business scenario as its very important as the company needs to know that the data in the columns are matching or not.

So, we need to design a business rule where it will indicate 1 if the data matches unless it will exist without displaying anything.


Column	K...	Type	<input checked="" type="checkbox"/> N..	Date Pattern (Ctrl+S...
 transaction_id	<input checked="" type="checkbox"/>	int	<input type="checkbox"/>	
transaction_type	<input type="checkbox"/>	String	<input type="checkbox"/>	
transaction_amount	<input type="checkbox"/>	BigDecimal	<input type="checkbox"/>	
date	<input type="checkbox"/>	Date	<input type="checkbox"/>	"dd-MM-yyyy"
sales_channel	<input type="checkbox"/>	String	<input checked="" type="checkbox"/>	
payment_method	<input type="checkbox"/>	String	<input type="checkbox"/>	
Customer_id	<input type="checkbox"/>	int	<input type="checkbox"/>	
Agent_id	<input type="checkbox"/>	int	<input type="checkbox"/>	
Ready_indicator	<input type="checkbox"/>	Boolean	<input checked="" type="checkbox"/>	

Figure 32:transaction business rule table.

Raw data of the business rule:

transaction_id	transaction_type	transaction_amt	date	sales_channel	payment_method	Customer_id	Agent_id
9001	REN	4500.00	2012-05-05	AGENT	PAYPAL	12345	1001
9002	CANCEL	1000.00	2012-05-05	AGENT	DD	12345	1006
9003	REN	6000.00	2011-07-08	AGENT	DD	678910	1002
9004	CANCEL	2000.00	2012-06-05	AGENT	PAYPAL	12345	1001
9005	CANCEL	4000.00	2010-10-03	AGENT	DD	11121314	1008
9006	NB	5000.00	2011-04-03	AGENT	CASH	12345	1001
9007	NB	9000.00	2012-05-06	AGENT	DD	15161718	1004
9008	CANCEL	1500.00	2011-08-08	AGENT	DD	678910	1007
9009	NB	10000.00	2011-05-05	AGENT	DD	12345	1001
9010	CANCEL	5000.00	2011-07-08	AGENT	DD	12345	1006
9011	NB	3400.00	2011-10-10	AGENT	DD	678910	1002
9012	CANCEL	1200.00	2011-11-10	AGENT	DD	678910	1002
9013	REN	6500.00	2011-12-12	AGENT	DD	678910	1002
9014	CANCEL	500.00	2011-12-13	AGENT	DD	678910	1007
9015	CANCEL	2500.00	2011-12-15	AGENT	DD	678910	1007
9016	NB	5000.00	2011-12-17	AGENT	DD	678910	1002
9017	CANCEL	8000.00	2011-12-31	AGENT	DD	678910	1002
9018	REN	4200.00	2011-12-08	AGENT	DD	11121314	1003
9019	REN	2500.00	2011-08-14	AGENT	DD	11121314	1003
9020	CANCEL	1600.00	2011-08-14	AGENT	DD	11121314	1008
9021	REN	8000.00	2011-09-17	AGENT	DD	11121314	1003
9022	CANCEL	2000.00	2011-09-18	AGENT	DD	11121314	1003
9023	CANCEL	4000.00	2012-09-20	AGENT	DD	11121314	1003
9024	NB	7000.00	2012-09-22	AGENT	DD	11121314	1003
9025	NB	9000.00	2012-08-08	AGENT	DD	12345	1001
9026	NB	5000.00	2012-09-09	AGENT	DD	12345	1001
9027	CANCEL	4000.00	2012-10-10	AGENT	DD	12345	1006
9028	REN	3000.00	2012-06-06	AGENT	DD	15161718	1004
9029	CANCEL	1000.00	2012-07-06	AGENT	CHEQUE	15161718	1009
9030	CANCEL	4000.00	2011-08-06	AGENT	CHEQUE	15161718	1009
9031	NB	5000.00	2012-06-06	AGENT	CASH	19202122	1005
9032	NB	5000.00	2012-07-06	AGENT	CASH	19202122	1005
9033	NB	5000.00	2012-08-06	AGENT	CASH	19202122	1005
9034	CANCEL	8000.00	2012-10-06	AGENT	CHEQUE	19202122	1005
9035	NB	3000.00	2012-12-06	AGENT	CASH	19202122	1005

Figure 33:raw data of the business rule.

Ready indictor of the business rule:

transaction_id	transaction_type	transaction_amount	date	sales_channel	payment_method	Customer_id	Agent_id	Ready_indicator
9001	REN	4500.00	2012-05-05	AGENT	PAYPAL	12345	1001	1
9002	CANCEL	1000.00	2012-05-05	AGENT	DD	12345	1006	1
9003	REN	6000.00	2011-07-08	AGENT	DD	678910	1002	1
9004	CANCEL	2000.00	2012-06-05	AGENT	PAYPAL	12345	1001	1
9005	CANCEL	4000.00	2010-10-03	AGENT	DD	11121314	1008	1
9006	NB	5000.00	2011-04-03	AGENT	CASH	12345	1001	1
9007	NB	9000.00	2012-05-06	AGENT	DD	15161718	1004	1
9008	CANCEL	1500.00	2011-08-08	AGENT	DD	678910	1007	1
9009	NB	10000.00	2011-05-05	AGENT	DD	12345	1001	1
9010	CANCEL	5000.00	2011-07-08	AGENT	DD	12345	1006	1
9011	NB	3400.00	2011-10-10	AGENT	DD	678910	1002	1
9012	CANCEL	1200.00	2011-11-10	AGENT	DD	678910	1002	1
9013	REN	6500.00	2011-12-12	AGENT	DD	678910	1002	1
9014	CANCEL	500.00	2011-12-13	AGENT	DD	678910	1007	1
9015	CANCEL	2500.00	2011-12-15	AGENT	DD	678910	1007	1
9016	NB	5000.00	2011-12-17	AGENT	DD	678910	1002	1
9017	CANCEL	8000.00	2011-12-31	AGENT	DD	678910	1002	1
9018	REN	4200.00	2011-12-08	AGENT	DD	11121314	1003	1
9019	REN	2500.00	2011-08-14	AGENT	DD	11121314	1003	1
9020	CANCEL	1600.00	2011-08-14	AGENT	DD	11121314	1008	1
9021	REN	8000.00	2011-09-17	AGENT	DD	11121314	1003	1
9022	CANCEL	2000.00	2011-09-18	AGENT	DD	11121314	1003	1
9023	CANCEL	4000.00	2012-09-20	AGENT	DD	11121314	1003	1
9024	NB	7000.00	2012-09-22	AGENT	DD	11121314	1003	1
9025	NB	9000.00	2012-08-08	AGENT	DD	12345	1001	1
9026	NB	5000.00	2012-09-09	AGENT	DD	12345	1001	1
9027	CANCEL	4000.00	2012-10-10	AGENT	DD	12345	1006	1
9028	REN	8000.00	2012-06-06	AGENT	DD	15161718	1004	1
9029	CANCEL	1000.00	2012-07-06	AGENT	CHEQUE	15161718	1009	1
9030	CANCEL	4000.00	2011-08-06	AGENT	CHEQUE	15161718	1009	1
9031	NB	5000.00	2012-06-06	AGENT	CASH	19202122	1005	1
9032	NB	5000.00	2012-07-06	AGENT	CASH	19202122	1005	1
9033	NB	5000.00	2012-08-06	AGENT	CASH	19202122	1005	1
9034	CANCEL	8000.00	2012-10-06	AGENT	CHEQUE	19202122	1005	1
9035	NB	3000.00	2012-12-06	AGENT	CASH	19202122	1005	1

Figure 34:implementing the ready indicator for the business rule.

As in the above figure, we can see that the ready indicator is showing the value as “1” which means the data is true. The data is matching.

Fact table:

We have designed the policy fact table, which are designed as per the star schema model.

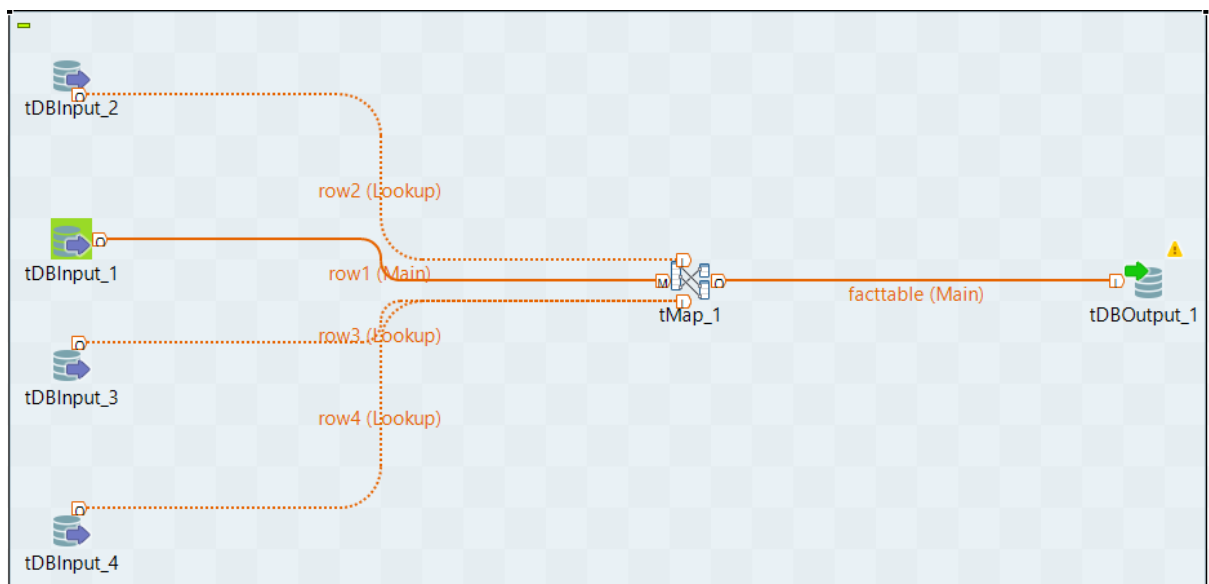


Figure 35:policy fact table

The policy fact table have these components:

- tDBInput 1: transaction_business_rule

- tDBInput 2: calendar_dimension table
- tDBInput 3: customer dimension table
- tDBInput 4: agent dimension table

All these components will be linked with the tmap and the obtained refined output will be stored in the tDBOutput component.

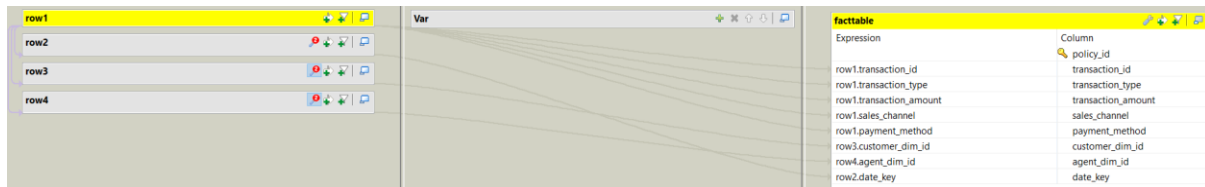


Figure 36:tmap component for policy fact

The surrogate key is the “policy_id”.

The Foreign key will be “date”.

“customer_id” and “agent_id” all are auto mapped with the output column.

It only takes the record, if the ready indicator displays the value “1” it is set true by the transaction_business_rule table (True=1, false=0).

Transaction_business_rule will be the main input for the tmap, which has the primary key as “transaction_id”.

tDBInput 2 component has the calendar dimension table which has the foreign key as “date”.

tDBInput 3 component has the customer dimension table which has the foreign key as “customer_id”.

tDBInput 4 component has the agent dimension table which has the foreign key as “agent_id”.

All the foreign keys are matched from the “transaction_business_rule” table to other input component tables.

We need to define the match model as “unique match” and the join model has the “inner join”.

transaction_id
transaction_type
transaction_amount
date
sales_channel
payment_method
Customer_id
Agent_id
Ready_indicator

row2	
Property	Value
Lookup Model	Load once
Match Model	Unique match
Join Model	Inner Join
Store temp data	false

Expr. key	Column
	date_key
	day
	year
	month
	weekday
	month_words
	weekday_num
row1.date	date

Figure 37:tmap for transaction_business_rule

transaction_id
transaction_type
transaction_amount
date
sales_channel
payment_method
Customer_id
Agent_id
Ready_indicator

row2	
row3	
Expr. key	Column
	customer_dim_id
row1.Customer_id	customer_id
	first_name
	last_name
	address
	pincode
	dob
	scd_start_date
	scd_end_date
	scd_active
	scd_version

Figure 38:tmap for customer dimension table

transaction_id
transaction_type
transaction_amount
date
sales_channel
payment_method
Customer_id
Agent_id
Ready_indicator

row2
row3
row4

Expr. key	Column
row1.Agent_id	agent_dim_id
	agent_id
	name
	address
	pincode
	agent_type
	scd_start_date
	scd_end_date
	scd_active
	scd_version

Figure 39:tmap for agent dimension table

Policy fact table output:

policy_id	transaction_id	transaction_type	transaction_amount	sales_channel	payment_method	customer_dim_id	agent_dim_id	date_key
1	9001	REN	4500.00	AGENT	PAYPAL	30	15	2317
2	9002	CANCEL	1000.00	AGENT	DD	30	20	2317
3	9003	REN	6000.00	AGENT	DD	24	16	2015
4	9004	CANCEL	2000.00	AGENT	PAYPAL	30	15	2348
5	9005	CANCEL	4000.00	AGENT	DD	25	22	1737
6	9006	NB	5000.00	AGENT	CASH	30	15	1919
7	9007	NB	9000.00	AGENT	DD	26	18	2318
8	9008	CANCEL	1500.00	AGENT	DD	24	21	2046
9	9009	NB	10000.00	AGENT	DD	30	15	1951
10	9010	CANCEL	5000.00	AGENT	DD	30	20	2015
11	9011	NB	3400.00	AGENT	DD	24	16	2109
12	9012	CANCEL	1200.00	AGENT	DD	24	16	2140
13	9013	REN	6500.00	AGENT	DD	24	16	2172
14	9014	CANCEL	500.00	AGENT	DD	24	21	2173
15	9015	CANCEL	2500.00	AGENT	DD	24	21	2175
16	9016	NB	5000.00	AGENT	DD	24	16	2177
17	9017	CANCEL	8000.00	AGENT	DD	24	16	2191
18	9018	REN	4200.00	AGENT	DD	25	17	2168
19	9019	REN	2500.00	AGENT	DD	25	17	2052
20	9020	CANCEL	1600.00	AGENT	DD	25	22	2052
21	9021	REN	8000.00	AGENT	DD	25	17	2086
22	9022	CANCEL	2000.00	AGENT	DD	25	17	2087
23	9023	CANCEL	4000.00	AGENT	DD	25	17	2455
24	9024	NB	7000.00	AGENT	DD	25	17	2457
25	9025	NB	9000.00	AGENT	DD	30	15	2412
26	9026	NB	5000.00	AGENT	DD	30	15	2444
27	9027	CANCEL	4000.00	AGENT	DD	30	20	2475
28	9028	REN	8000.00	AGENT	DD	26	18	2349
29	9029	CANCEL	1000.00	AGENT	CHEQUE	26	23	2379
30	9030	CANCEL	4000.00	AGENT	CHEQUE	26	23	2044
31	9031	NB	5000.00	AGENT	CASH	31	19	2349
32	9032	NB	5000.00	AGENT	CASH	31	19	2379
33	9033	NB	5000.00	AGENT	CASH	31	19	2410
34	9034	CANCEL	8000.00	AGENT	CHEQUE	31	19	2471
35	9035	NB	3000.00	AGENT	CASH	31	19	2532

Figure 40:result for the policy fact table

Dimension table results:

Insert:

Customer dimension table insert result.

```
mysql> select * from customer_dim;
```

customer_dim_id	customer_id	first_name	last_name	address	pincode	dob	scd_start_date	scd_end_date	scd_active	scd_version
1	12345	FIRNAME	A	123,1ST MAIN, 2ND CROSS	520083	1977-04-04 00:00:00	2019-11-19 00:44:48	NULL	1	1
2	678910	FNAME	B	342,2ND MAIN, 15TH CROSS	520084	1971-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
3	11121314	FIRNAME	C	32, 1ST MAIN, 1ST CROSS	520085	1979-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
4	15161718	FNAME	D	151, 15TH MAIN, 39TH CROSS	520086	1978-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
5	19202122	FIRSTNAME	E	155, 2ND CROSS,11TH AVENUE	520087	1984-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1

Figure 41:insertion result for customer dimension table

Location dimension table insert result.

```
mysql> select * from location_dim;
```

location_id	city	state	country
1	DUNGLOE	DONEGAL	IRELAND
2	CLIFFONY	SLIGO	IRELAND
3	LISTOWEL	KERRY	IRELAND
4	CLONES	MONAGHAN	IRELAND
5	LONDON	NA	UNITED KINGDOM
6	TEST	TEST	TEST

Figure 42: insertion result for location dimension table

Agent dimension table insert result.

```
mysql> select * from agent_dim;
```

agent_dim_id	agent_id	name	address	pincode	agent_type	scd_start_date	scd_end_date	scd_active	scd_version	location_id
1	1001	XYZ ROSSES	MAIN CENTRAL	570001	EXC	2019-11-19 00:46:43	NULL	1	1	1
2	1002	XYZ INS HO	POLE CENTRAL	570002	EXC	2019-11-19 00:46:43	NULL	1	1	2
3	1003	XYZ LISTOWEL	MAIN STREET CENTRAL	570003	EXC	2019-11-19 00:46:43	NULL	1	1	3
4	1004	HEGARTY HO	POLE STREET CENTRAL	570004	EXC	2019-11-19 00:46:43	NULL	1	1	4
5	1005	LONDON HO	LONDON CENTRAL	570005	EXC	2019-11-19 00:46:43	NULL	1	1	5
6	1006	XYZ AREA 1	92, TALL TOWERS	570006	IND	2019-11-19 00:46:43	NULL	1	1	1
7	1007	XYZ AREA 1	8, BANK ARCADE	570007	IND	2019-11-19 00:46:43	NULL	1	1	2
8	1008	HEGARTY AREA 1	51, 22 BUILDING	570008	IND	2019-11-19 00:46:43	NULL	1	1	3
9	1009	XYZ AGENT AREA 1	42, MAINSTREET	570009	IND	2019-11-19 00:46:43	NULL	1	1	4

Figure 43:insertion result for agent dimension table

Calendar dimension table insert result.

```
mysql> select * from calendar_dim;
```

date_key	day	year	month	weekday	month_words	weekday_num	date
1	1	2010	1	FRI	JAN	5	2010-01-01 00:00:00
2	2	2010	1	SAT	JAN	6	2010-01-02 00:00:00
3	3	2010	1	SUN	JAN	7	2010-01-03 00:00:00
4	4	2010	1	MON	JAN	1	2010-01-04 00:00:00
5	5	2010	1	TUE	JAN	2	2010-01-05 00:00:00
6	6	2010	1	WED	JAN	3	2010-01-06 00:00:00
7	7	2010	1	THU	JAN	4	2010-01-07 00:00:00
8	8	2010	1	FRI	JAN	5	2010-01-08 00:00:00
9	9	2010	1	SAT	JAN	6	2010-01-09 00:00:00
10	10	2010	1	SUN	JAN	7	2010-01-10 00:00:00
11	11	2010	1	MON	JAN	1	2010-01-11 00:00:00
12	12	2010	1	TUE	JAN	2	2010-01-12 00:00:00
13	13	2010	1	WED	JAN	3	2010-01-13 00:00:00
14	14	2010	1	THU	JAN	4	2010-01-14 00:00:00
15	15	2010	1	FRI	JAN	5	2010-01-15 00:00:00
16	16	2010	1	SAT	JAN	6	2010-01-16 00:00:00
17	17	2010	1	SUN	JAN	7	2010-01-17 00:00:00
18	18	2010	1	MON	JAN	1	2010-01-18 00:00:00
19	19	2010	1	TUE	JAN	2	2010-01-19 00:00:00
20	20	2010	1	WED	JAN	3	2010-01-20 00:00:00
21	21	2010	1	THU	JAN	4	2010-01-21 00:00:00
22	22	2010	1	FRI	JAN	5	2010-01-22 00:00:00
23	23	2010	1	SAT	JAN	6	2010-01-23 00:00:00
24	24	2010	1	SUN	JAN	7	2010-01-24 00:00:00
25	25	2010	1	MON	JAN	1	2010-01-25 00:00:00
26	26	2010	1	TUE	JAN	2	2010-01-26 00:00:00
27	27	2010	1	WED	JAN	3	2010-01-27 00:00:00
28	28	2010	1	THU	JAN	4	2010-01-28 00:00:00
29	29	2010	1	FRI	JAN	5	2010-01-29 00:00:00
30	30	2010	1	SAT	JAN	6	2010-01-30 00:00:00
31	31	2010	1	SUN	JAN	7	2010-01-31 00:00:00
32	1	2010	2	MON	FEB	1	2010-02-01 00:00:00
33	2	2010	2	TUE	FEB	2	2010-02-02 00:00:00

Figure 44:insertion result for calendar dimension table

Insert or update results for the dimension table.

Customer dimension table result for insert or update

```
mysql> select * from customer_dim;
```

customer_dim_id	customer_id	first_name	last_name	address	pincode	dob	scd_start_date	scd_end_date	scd_active	scd_version
1	12345	FIRNAME	A	123,1ST MAIN, 2ND CROSS	520083	1977-04-04 00:00:00	2019-11-19 00:40:32	NULL	1	1
2	678910	FINAME	B	342,2ND MAIN, 15TH CROSS	520084	1971-01-01 00:00:00	2019-11-19 00:40:32	NULL	1	1
3	11121314	FIRNAME	C	32, 1ST MAIN, 1ST CROSS	520085	1979-01-01 00:00:00	2019-11-19 00:40:32	NULL	1	1
4	15161718	FAME	D	151, 15TH MAIN, 39TH CROSS	520086	1978-01-01 00:00:00	2019-11-19 00:40:32	NULL	1	1
5	19202122	FIRSTNAME	E	155, 2ND CROSS,11TH AVENUE	520087	1984-01-01 00:00:00	2019-11-19 00:40:32	NULL	1	1
6	23242526	FINAMES	AD	123,1ST MAIN, 2ND CROSS	520099	1971-01-01 00:00:00	2019-11-19 00:41:32	2019-11-19 00:41:32	0	1
7	23242526	FINAMES	AM	155, 2ND CROSS,11TH AVENUE	520087	1971-01-01 00:00:00	2019-11-19 00:41:32	NULL	1	2

Figure 45:insert or update for customer dimension table

Agent dimension table result for insert or update

```
mysql> select * from agent_dim;
```

agent_dim_id	agent_id	name	address	pincode	agent_type	scd_start_date	scd_end_date	scd_active	scd_version	location_id
1	1001	XYZ ROSSES	MAIN CENTRAL	570001	EXC	2019-11-19 00:46:43	NULL	1	1	1
2	1002	XYZ INS HO	POLE CENTRAL	570002	EXC	2019-11-19 00:46:43	NULL	1	1	2
3	1003	XYZ LISTOWEL	MAIN STREET CENTRAL	570003	EXC	2019-11-19 00:46:43	NULL	1	1	3
4	1004	HEGARTY HO	POLE STREET CENTRAL	570004	EXC	2019-11-19 00:46:43	NULL	1	1	4
5	1005	LONDON HO	LONDON CENTRAL	570005	EXC	2019-11-19 00:46:43	NULL	1	1	5
6	1006	XYZ AREA 1	92, TALL TOMERS	570006	TND	2019-11-19 00:46:43	NULL	1	1	1
7	1007	XYZ AREA 1	8, BANK ARCADE	570007	TND	2019-11-19 00:46:43	NULL	1	1	2
8	1008	HEGARTY AREA 1	51, 22 BUILDING	570008	TND	2019-11-19 00:46:43	NULL	1	1	3
9	1009	XYZ AGENT AREA 1	42, MAINSTREET	570009	TND	2019-11-19 00:46:43	NULL	1	1	4
10	2003	LONDON AREA 1	#6, BIG BEN BUILDING	570009	TND	2019-11-19 00:48:15	2019-11-19 00:48:15	0	1	5
11	2003	LONDON AREA 1	#6, BIG BEN BUILDING,CENTRAL	570009	TND	2019-11-19 00:48:15	NULL	1	2	5

Figure 46:insert or update for agent dimension table

Update result for the dimension table.

Customer dimension table result for the update

```
mysql> select * from customer_dim;
```

customer_dim_id	customer_id	first_name	last_name	address	pincode	dob	scd_start_date	scd_end_date	scd_active	scd_version
1	12345	FIRNAME	A	123,1ST MAIN, 2ND CROSS	520083	1977-04-04 00:00:00	2019-11-19 00:44:48	2019-11-19 00:50:42	0	1
2	678910	FINAME	B	342,2ND MAIN, 15TH CROSS	520084	1971-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
3	11121314	FIRNAME	C	32, 1ST MAIN, 1ST CROSS	520085	1979-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
4	15161718	FAME	D	151, 15TH MAIN, 39TH CROSS	520086	1978-01-01 00:00:00	2019-11-19 00:44:48	NULL	1	1
5	19202122	FIRSTNAME	E	155, 2ND CROSS,11TH AVENUE	520087	1984-01-01 00:00:00	2019-11-19 00:44:48	2019-11-19 00:50:42	0	1
6	12345	FIRNAME	AB	123,1ST MAIN, 2ND CROSS	520099	1977-04-04 00:00:00	2019-11-19 00:50:42	NULL	1	2
7	19202122	FIRSTNAME	EC	155, 2ND CROSS,11TH AVENUE	520087	1984-01-01 00:00:00	2019-11-19 00:50:42	NULL	1	2

Figure 47:update result for customer dimension table

Agent dimension table result for the update

```
mysql> select * from agent_dim;
```

agent_dim_id	agent_id	name	address	pincode	agent_type	scd_start_date	scd_end_date	scd_active	scd_version	location_id
1	1001	XYZ ROSSES	MAIN CENTRAL	570001	EXC	2019-11-19 00:46:43	2019-11-19 00:54:03	0	1	1
2	1002	XYZ INS HO	POLE CENTRAL	570002	EXC	2019-11-19 00:46:43	2019-11-19 00:54:03	0	1	2
3	1003	XYZ LISTOMEL	MAIN STREET CENTRAL	570003	EXC	2019-11-19 00:46:43	NULL	1	1	3
4	1004	HEGARTY HO	POLE STREET CENTRAL	570004	EXC	2019-11-19 00:46:43	NULL	1	1	4
5	1005	LONDON HO	LONDON CENTRAL	570005	EXC	2019-11-19 00:46:43	NULL	1	1	5
6	1006	XYZ AREA 1	92, TALL TOWERS	570006	IND	2019-11-19 00:46:43	NULL	1	1	1
7	1007	XYZ AREA 1	8, BANK ARCADE	570007	IND	2019-11-19 00:46:43	NULL	1	1	2
8	1008	HEGARTY AREA 1	51, 22 BUILDING	570008	IND	2019-11-19 00:46:43	NULL	1	1	3
9	1009	XYZ AGENT AREA 1	42, MAINSTREET	570009	IND	2019-11-19 00:46:43	NULL	1	1	4
10	2003	LONDON AREA 1	#6, BIG BEN BUILDING	570009	IND	2019-11-19 00:48:15	2019-11-19 00:48:15	0	1	5
11	2003	LONDON AREA 1	#6, BIG BEN BUILDING,CENTRAL	570009	IND	2019-11-19 00:48:15	NULL	1	2	5
12	1001	XYZ ROSSES	MAIN STREET	570001	EXE	2019-11-19 00:54:03	NULL	1	2	1
13	1002	XYZ INS HO	POLE STREET	570002	EXE	2019-11-19 00:54:03	NULL	1	2	2

Figure 48:update result for agent dimension table

Location dimension table result for the update

```
mysql> select * from location_dim;
```

location_id	city	state	country
1	GLENTIES	DONEGAL	IRELAND
2	CLIFFONY	SLIGO	IRELAND
3	LISTOWEL	KERRY	IRELAND
4	CLONES	MONAGHAN	IRELAND
5	LONDON	NA	UNITED KINGDOM
6	TEST	TEST	TEST

Figure 49: update result for location dimension table

5 Visualization:

As all the data has been refined and restored into the database, these data need to be visualized in a way that it is easily understandable to the officials and there are few business questions that are asked previously need to be answered with the help of MYSQL and Tableau tools.

For tableau data visualization, we need to create a database connection with the MySQL and all the dimension tables need to be joined for data gathering.

In the below figure we can see the connection established in tableau:

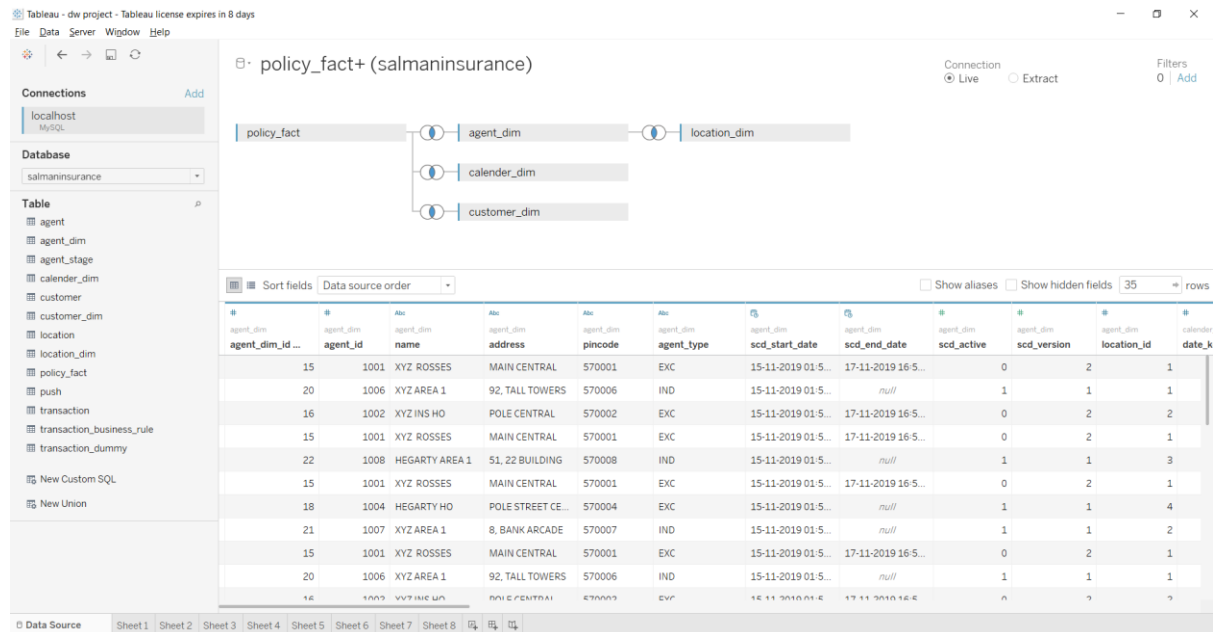


Figure 50:connection in tableau

5.1 Business questions:

Before developing this data warehouse project, we have asked a few business questions for the “XYZ INSURANCE” company. If the questions were answered with the help of MySQL and tableau tools, it will be very useful for the company officials for the decision making and predicting the business analysis.

- What is the total revenue for New business in 2011?

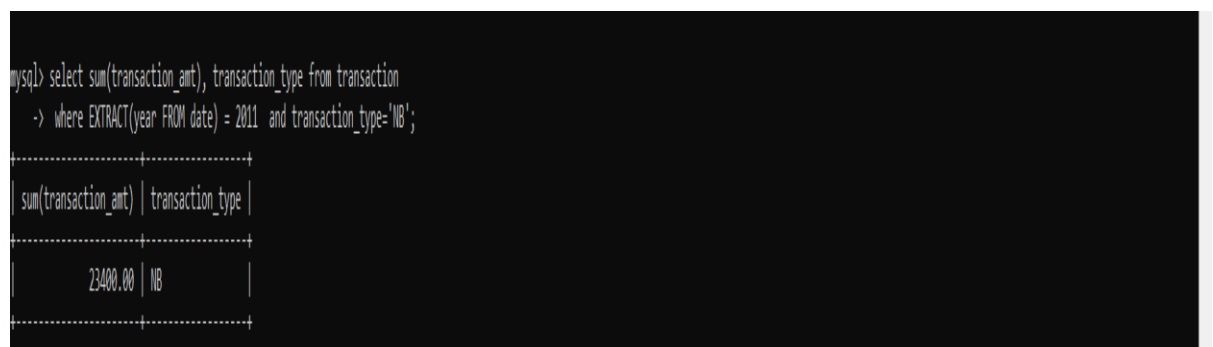


Figure 51:result for the total revenue for new business in 2011

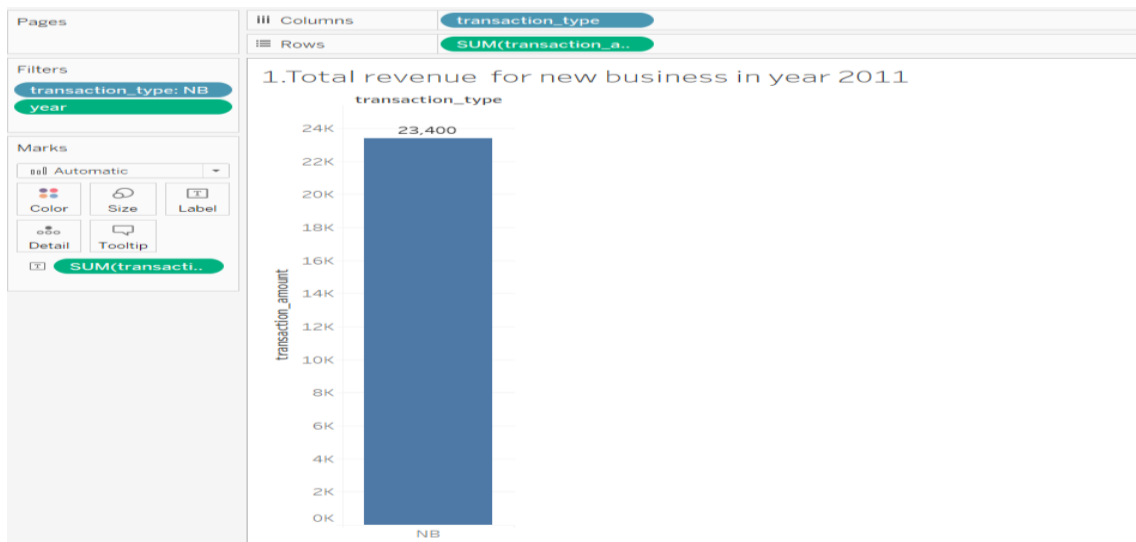


Figure 52: visualization result for the total revenue for new business in 2011

- What is the sum of the total cancelations by agent for 2012? List the name of the agent and their city.

```
mysql> select agent_dim.name, location_dim.city, sum(policy_fact.transaction_amount) from policy_fact
-> INNER JOIN calender_dim on calender_dim.date_key = policy_fact.date_key
-> INNER JOIN agent_dim on agent_dim.agent_dim_id = policy_fact.agent_dim_id
-> INNER JOIN location_dim on location_dim.location_id = agent_dim.location_id
-> where calender_dim.year = '2012' and policy_fact.transaction_type = 'CANCEL'
-> group by calender_dim.year, agent_dim.agent_id order by sum(policy_fact.transaction_amount);
```

name	city	sum(policy_fact.transaction_amount)
XYZ AGENT AREA 1	CLONES	1000.00
XYZ ROSSES	DUNGLOE	2000.00
XYZ LISTOWEL	LISTOWEL	4000.00
XYZ AREA 1	DUNGLOE	5000.00
LONDON HO	LONDON	8000.00

5 rows in set (0.00 sec)

Figure 53: result for the sum of total cancelations by agent for 2012.

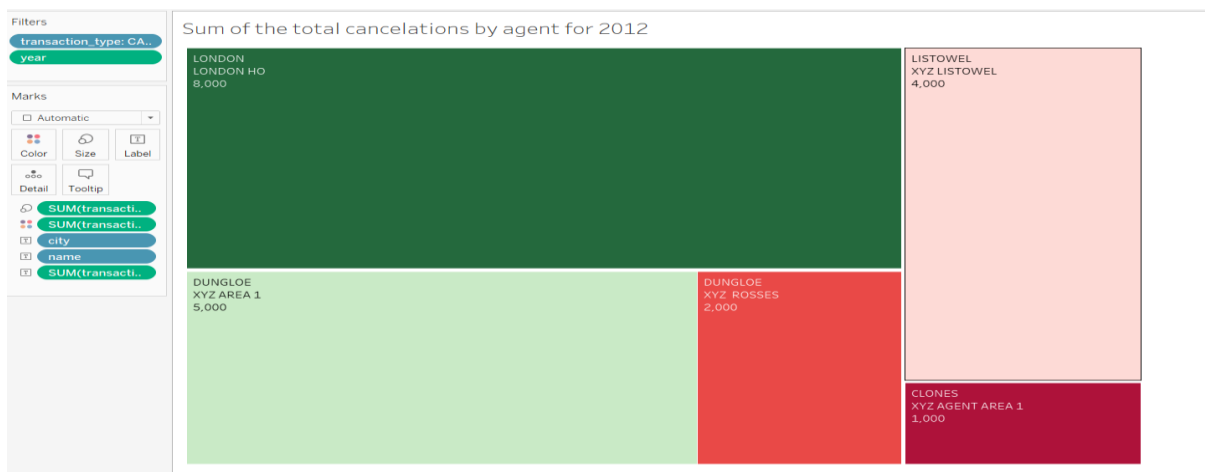


Figure 54: visualization result for the sum of total cancelations by agent for 2012.

- What is the total new business transactions amount per Agent?

```
mysql> select transaction.transaction_type,agent.name,sum(transaction.transaction_amt) from transaction INNER JOIN agent ON transaction.Agent_id=agent.agent_id where transaction_type='NB' group by 2;
```

transaction_type	name	sum(transaction.transaction_amt)
NB	XYZ ROSSES	29000.00
NB	HEGARTY HQ	9000.00
NB	XYZ INS HQ	8400.00
NB	XYZ LISTOWEL	7000.00
NB	LONDON HQ	18000.00

5 rows in set (0.00 sec)

Figure 55: result for the total new business transactions amount per agent.

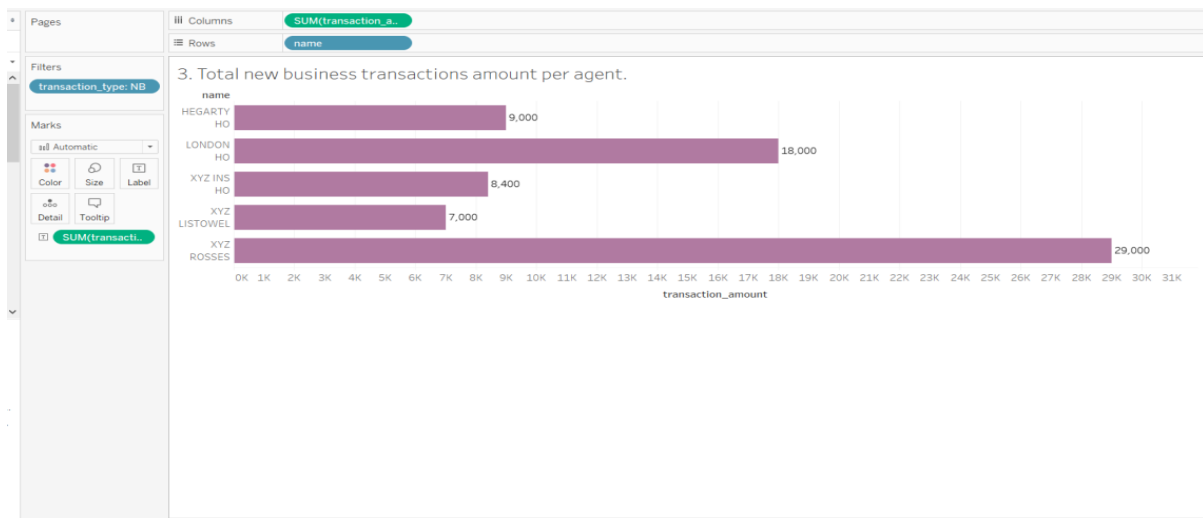


Figure 56: visualization result for the total new business transactions amount per agent.

- Who is the top selling agent for New Business?

```
mysql> select transaction.transaction_type,agent.name,sum(transaction.transaction_amt) from transaction INNER JOIN agent ON transaction.Agent_id=agent.agent_id where transaction_type='NB' group by 2 limit 1;
```

transaction_type	name	sum(transaction.transaction_amt)
NB	XYZ ROSSES	29000.00

1 row in set (0.00 sec)

Figure 57 result for the top selling agent for new business.

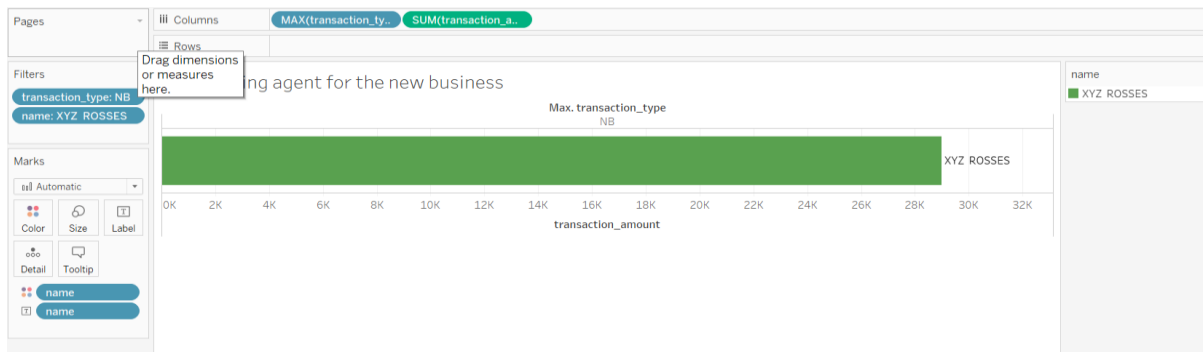


Figure 58: visualization result for the top selling agent for new business.

- Who are the top 3 performing agents for renewals?

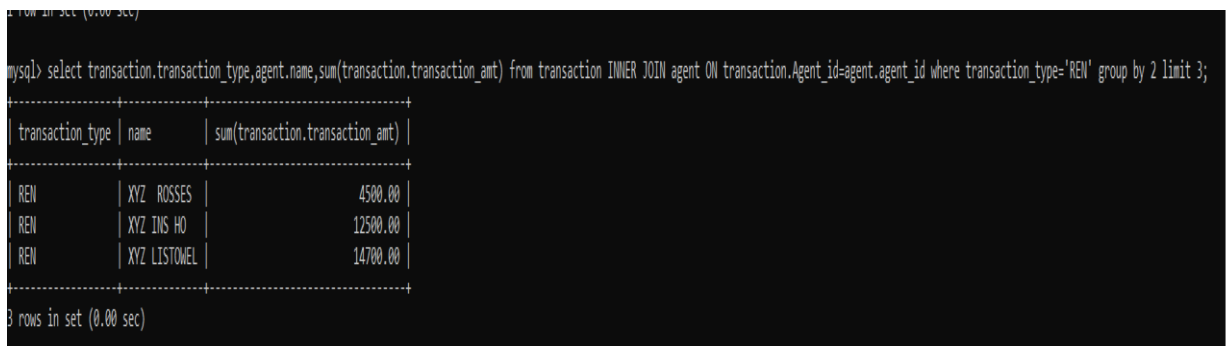


Figure 59: result for top3 performing agents for renewals



Figure 60: visualization result for top3 performing agents for renewals

- Who are making the most in transaction total, exclusive or independent agents?

```
mysql> select agent.agent_type,sum(transaction.transaction_amt) from transaction INNER JOIN agent ON transaction.Agent_id=agent.agent_id group by 1;
```

agent_type	sum(transaction.transaction_amt)
EXE	65600.00
EXC	70700.00
IND	25100.00

3 rows in set (0.00 sec)

Figure 61:result for the most transaction total, exclusive or independent agents

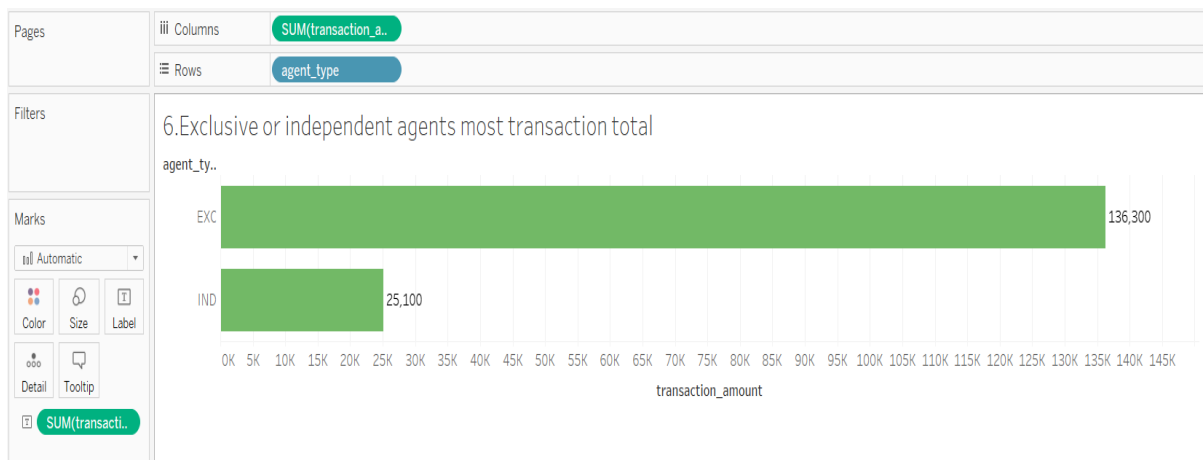


Figure 62: visualization result for the most transaction total, exclusive or independent agents

- What is the name and age of the highest value new business customer?

```
mysql> select CONCAT(a.first_name , ' ',a.last_name) AS Name,max(b.transaction_amt),TIMESTAMPDIFF (YEAR, a.dob, CURDATE()) AS AGE from customer a, transaction b where b.transaction_type='NB';
```

Name	max(b.transaction_amt)	AGE
FIRMINES AB	10000.00	42

1 row in set (0.00 sec)

Figure 63:result for name and age of highest value new business customer

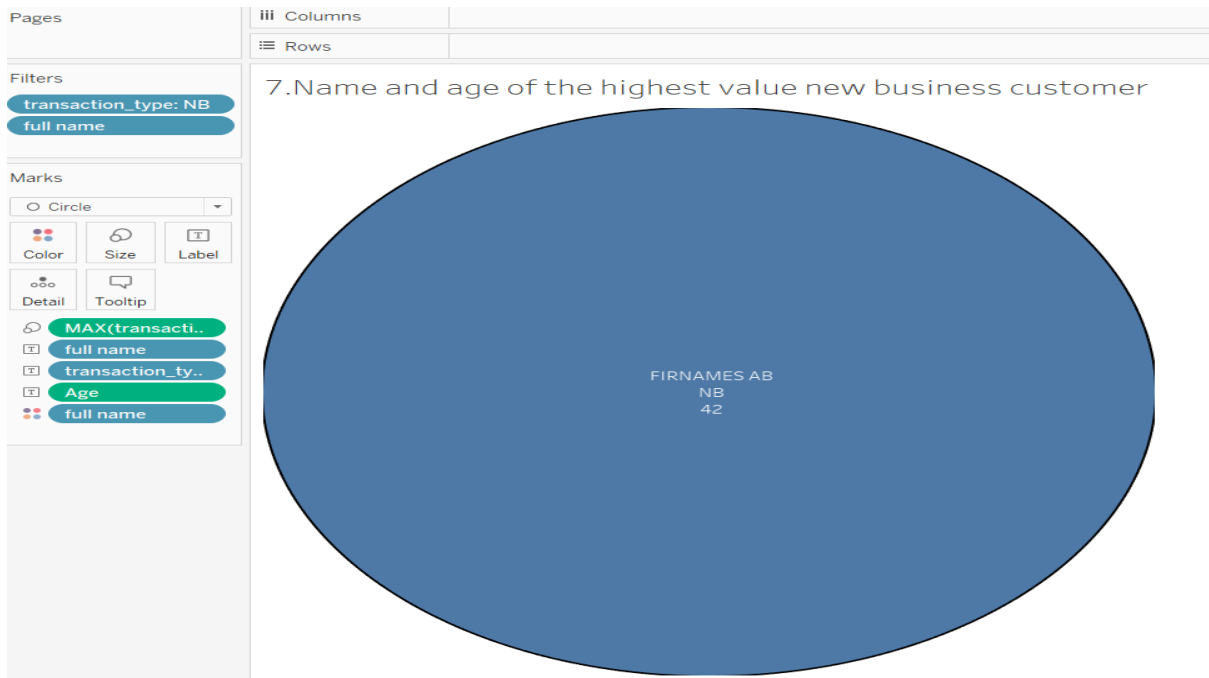


Figure 64:visualization result for name and age of highest value new business customer

- What is the most popular payment method?

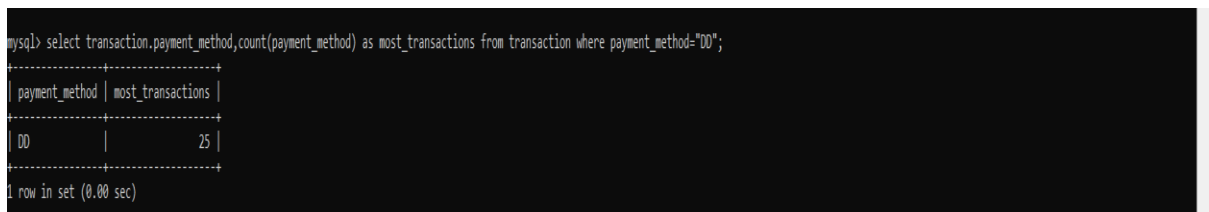


Figure 65: result for most popular payment method.

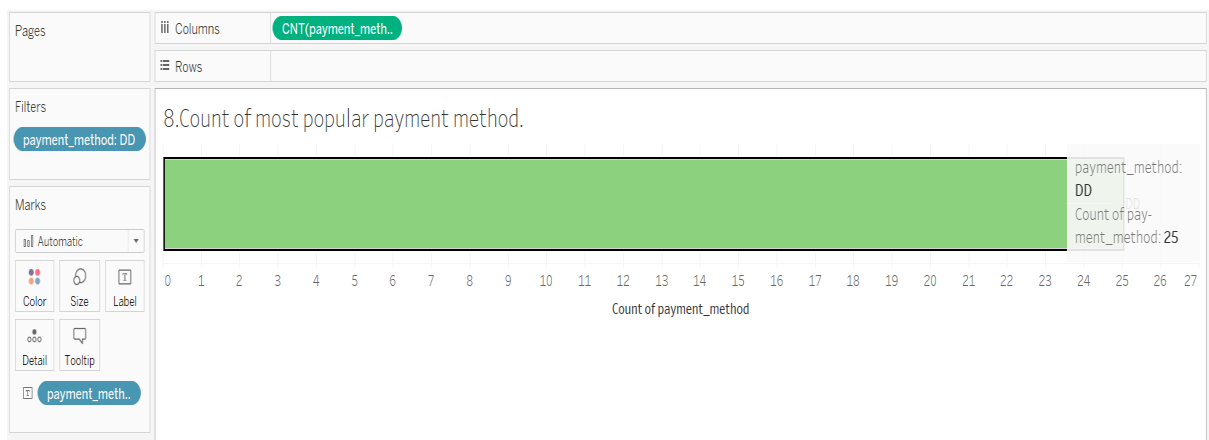


Figure 66: visualization result for most popular payment method.

- List total revenue generated from new business and renewals by agent. In your output list the agent's name, address and city and whether they are independent or exclusive agents.

```
mysql> select agent_dim.name, agent_dim.address, location_dim.city, policy_fact.transaction_type, sum(policy_fact.transaction_amount) from policy_fact
-> INNER JOIN agent_dim on agent_dim.agent_dim_id = policy_fact.agent_dim_id
-> INNER JOIN location_dim on location_dim.location_id = agent_dim.location_id
-> where policy_fact.transaction_type IN ('NB','REN')
-> group by policy_fact.transaction_type,agent_dim.name
-> ;
```

name	address	city	transaction_type	sum(policy_fact.transaction_amount)
XYZ ROSSES	MAIN CENTRAL	DUNGLOE	REN	4500.00
XYZ INS HO	POLE CENTRAL	CLIFFONY	REN	12500.00
XYZ ROSSES	MAIN CENTRAL	DUNGLOE	NB	29000.00
HEGARTY HO	POLE STREET CENTRAL	CLONES	NB	9000.00
XYZ INS HO	POLE CENTRAL	CLIFFONY	NB	8400.00
XYZ LISTOMEL	MAIN STREET CENTRAL	LISTOMEL	REN	14700.00
XYZ LISTOMEL	MAIN STREET CENTRAL	LISTOMEL	NB	7000.00
HEGARTY HO	POLE STREET CENTRAL	CLONES	REN	8000.00
LONDON HO	LONDON CENTRAL	LONDON	NB	18000.00

0 rows in set (0.00 sec)

Figure 67: result for total revenue generated from new business and renewals by agent

6 Conclusion:

The main agenda of this project is to build a data warehouse project for the “XYZ INSURANCE” company. The financial company like this will be majorly focused on their data for future decisions and making the company profitable. In this data warehouse project, we have used different types of tools for data arrangement and refinement. All the data of the insurance company has been saved in the form of excel files. These data need to be stored in the database and all the files are saved in the database tables like customer, agent, location, calendar and transaction tables. Database and tables are created in MYSQL workbench. Then this database is connected with the Talend open studio data integration, as we have created metadata files, all these metadata files are assigned with the excel files. We have created individual jobs in the Talend and run the jobs and refine the data as insert, insert or update, and update. All the refined data has been stored in the dimension table. The next step is to refine the data with the help of Slowly changing Dimension (SCD), the fields like type 0 to type 3 are used for proper refinement of the data. Business rule has been developed to know the data in other tables are matched or not. All the refined data has been saved in the database as a dimension table. All these refined data are non-readable, we need proper visualization tool like tableau for data visualization. These data visualization is done in the form of pie charts, bar graphs etc. Before developing the project, we have asked different business questions for the company. The business questions are answered with the help of “MYSQL queries” and “tableau data visualization”. The visualization graphs and answers are very useful for the company officials for decision making and market predictions.

7 Bibliography:

- 4 Ways You Can Benefit From Insurance Business Intelligence [online] (2019) available:
<https://us.hitachi-solutions.com/blog/insurance-business-intelligence/> [accessed 19 Nov 2019].
- Introduction to Slowly Changing Dimensions (SCD) Types - Adatis [online] (2019) available:
<https://adatis.co.uk/introduction-to-slowly-changing-dimensions-scd-types/> [accessed 16 Dec 2019].
- What Is Tableau? Uses and Applications [online] (2019) available: <https://www.guru99.com/what-is-tableau.html> [accessed 19 Nov 2019].

8 Appendices:

```
CREATE DATABASE IF NOT EXISTS `salmaninsurance` /*!40100 DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;

USE `salmaninsurance`;

-- MySQL dump 10.13 Distrib 8.0.16, for Win64 (x86_64)

--

-- Host: localhost Database: salmaninsurance

-- -----

-- Server version      8.0.16

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0
*/;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--

-- Table structure for table `agent`

--

DROP TABLE IF EXISTS `agent`;

/*!40101 SET @saved_cs_client = @@character_set_client */;
```

```

SET character_set_client = utf8mb4 ;

CREATE TABLE `agent` (

  `agent_id` int(10) NOT NULL AUTO_INCREMENT,

  `name` varchar(20) NOT NULL,

  `address` varchar(50) NOT NULL,

  `agent_type` varchar(14) NOT NULL,

  `date` date NOT NULL,

  `location_id` int(10) NOT NULL,

  `pincode` int(10) NOT NULL,

  PRIMARY KEY (`agent_id`),

  KEY `location_id` (`location_id`),

  CONSTRAINT `agent_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES `location` (`location_id`)

) ENGINE=InnoDB AUTO_INCREMENT=2004 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `agent_dim`
--

DROP TABLE IF EXISTS `agent_dim`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `agent_dim` (

  `agent_dim_id` int(10) NOT NULL AUTO_INCREMENT,

  `agent_id` int(10) NOT NULL,

  `name` varchar(20) NOT NULL,

  `address` varchar(50) NOT NULL,

```

```

`pincode` varchar(10) NOT NULL,

`agent_type` varchar(14) NOT NULL,

`scd_start_date` datetime NOT NULL,

`scd_end_date` datetime DEFAULT NULL,

`scd_active` tinyint(1) NOT NULL,

`scd_version` int(11) NOT NULL,

`location_id` int(10) NOT NULL,

PRIMARY KEY (`agent_dim_id`),

KEY `location_id` (`location_id`),

CONSTRAINT `agent_dim_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES `location` (`location_id`)

) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Table structure for table `agent_stage`

--

DROP TABLE IF EXISTS `agent_stage`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `agent_stage` (

  `agent_id` int(10) NOT NULL AUTO_INCREMENT,

  `name` varchar(20) NOT NULL,

  `address` varchar(50) NOT NULL,

  `agent_type` varchar(14) NOT NULL,

  `date` date NOT NULL,

  `location_id` int(10) NOT NULL,

```

```

PRIMARY KEY (`agent_id`),

KEY `location_id` (`location_id`),

CONSTRAINT `agent_stage_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES `location`
(`location_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `calender_dim`
--

DROP TABLE IF EXISTS `calender_dim`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `calender_dim` (

  `date_key` int(10) NOT NULL AUTO_INCREMENT,

  `day` int(2) NOT NULL,

  `year` int(4) NOT NULL,

  `month` int(2) NOT NULL,

  `weekday` varchar(3) NOT NULL,

  `month_words` varchar(3) NOT NULL,

  `weekday_num` int(10) DEFAULT NULL,

  `date` datetime NOT NULL,

  PRIMARY KEY (`date_key`)

) ENGINE=InnoDB AUTO_INCREMENT=5845 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

```

```

--

-- Table structure for table `customer`

--


DROP TABLE IF EXISTS `customer`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `customer` (

  `customer_id` int(10) NOT NULL AUTO_INCREMENT,

  `first_name` varchar(10) NOT NULL,

  `last_name` varchar(10) DEFAULT NULL,

  `address` varchar(50) NOT NULL,

  `pincode` varchar(8) NOT NULL,

  `date` timestamp NULL DEFAULT NULL,

  `dob` date NOT NULL,

  PRIMARY KEY (`customer_id`)

) ENGINE=InnoDB AUTO_INCREMENT=19202123 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Table structure for table `customer_dim`

--


DROP TABLE IF EXISTS `customer_dim`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `customer_dim` (

```

```

`customer_dim_id` int(10) NOT NULL AUTO_INCREMENT,

`customer_id` int(10) NOT NULL,

`first_name` varchar(10) NOT NULL,

`last_name` varchar(10) DEFAULT NULL,

`address` varchar(50) NOT NULL,

`pincode` varchar(8) NOT NULL,

`dob` datetime NOT NULL,

`scd_start_date` datetime NOT NULL,

`scd_end_date` datetime DEFAULT NULL,

`scd_active` tinyint(1) NOT NULL,

`scd_version` int(11) NOT NULL,

PRIMARY KEY (`customer_dim_id`)

) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `location`
--

DROP TABLE IF EXISTS `location`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `location` (

`location_id` int(10) NOT NULL AUTO_INCREMENT,

`city` varchar(20) NOT NULL,

`state` varchar(20) NOT NULL,

`country` varchar(20) NOT NULL,

```

```

PRIMARY KEY (`location_id`)

) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `location_dim`
--

DROP TABLE IF EXISTS `location_dim`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `location_dim` (
  `location_id` int(10) NOT NULL AUTO_INCREMENT,
  `city` varchar(20) NOT NULL,
  `state` varchar(20) NOT NULL,
  `country` varchar(20) NOT NULL,
  PRIMARY KEY (`location_id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `policy_fact`
--

DROP TABLE IF EXISTS `policy_fact`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

```

```

SET character_set_client = utf8mb4 ;

CREATE TABLE `policy_fact` (

  `policy_id` int(10) NOT NULL AUTO_INCREMENT,

  `transaction_id` int(10) NOT NULL,

  `transaction_type` varchar(14) NOT NULL,

  `transaction_amount` decimal(10,2) NOT NULL,

  `sales_channel` varchar(14) NOT NULL,

  `payment_method` varchar(10) NOT NULL,

  `customer_dim_id` int(10) NOT NULL,

  `agent_dim_id` int(10) NOT NULL,

  `date_key` int(10) NOT NULL,

  PRIMARY KEY (`policy_id`)

) ENGINE=InnoDB AUTO_INCREMENT=36 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `push`
--

DROP TABLE IF EXISTS `push`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `push` (

  `transaction_type` varchar(10) NOT NULL,

  `name` varchar(30) NOT NULL,

  `transaction_amt` decimal(10,2) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```



```

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `transaction`
--

DROP TABLE IF EXISTS `transaction`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `transaction` (
  `transaction_id` int(10) NOT NULL AUTO_INCREMENT,
  `transaction_type` varchar(14) NOT NULL,
  `transaction_amt` decimal(10,2) NOT NULL,
  `date` date NOT NULL,
  `sales_channel` varchar(14) DEFAULT NULL,
  `payment_method` varchar(10) NOT NULL,
  `Customer_id` int(10) NOT NULL,
  `Agent_id` int(10) NOT NULL,
  PRIMARY KEY (`transaction_id`),
  KEY `Customer_id` (`Customer_id`),
  KEY `Agent_id` (`Agent_id`),
  CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`Customer_id`) REFERENCES `customer`
(`customer_id`),
  CONSTRAINT `transaction_ibfk_2` FOREIGN KEY (`Agent_id`) REFERENCES `agent` (`agent_id`)
) ENGINE=InnoDB AUTO_INCREMENT=9036 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

```

```

--
-- Table structure for table `transaction_business_rule`
--

DROP TABLE IF EXISTS `transaction_business_rule`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `transaction_business_rule` (

  `transaction_id` int(10) NOT NULL AUTO_INCREMENT,

  `transaction_type` varchar(14) NOT NULL,

  `transaction_amount` decimal(10,2) NOT NULL,

  `date` date NOT NULL,

  `sales_channel` varchar(14) DEFAULT NULL,

  `payment_method` varchar(10) NOT NULL,

  `Customer_id` int(10) NOT NULL,

  `Agent_id` int(10) NOT NULL,

  `Ready_indicator` tinyint(1) DEFAULT '0',

  PRIMARY KEY (`transaction_id`),

  KEY `Customer_id` (`Customer_id`),

  KEY `Agent_id` (`Agent_id`),

  CONSTRAINT `transaction_business_rule_ibfk_1` FOREIGN KEY (`Customer_id`) REFERENCES `customer` (`customer_id`),

  CONSTRAINT `transaction_business_rule_ibfk_2` FOREIGN KEY (`Agent_id`) REFERENCES `agent` (`agent_id`)

) ENGINE=InnoDB AUTO_INCREMENT=9036 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

```

```

-- Table structure for table `transaction_dummy`

--

DROP TABLE IF EXISTS `transaction_dummy`;

/*!40101 SET @saved_cs_client = @@character_set_client */;

SET character_set_client = utf8mb4 ;

CREATE TABLE `transaction_dummy` (

  `transaction_id` int(10) NOT NULL AUTO_INCREMENT,

  `transaction_type` varchar(14) NOT NULL,

  `transaction_amount` decimal(10,2) NOT NULL,

  `date` date NOT NULL,

  `sales_channel` varchar(14) DEFAULT NULL,

  `payment_method` varchar(10) NOT NULL,

  PRIMARY KEY (`transaction_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Dumping events for database 'salmaninsurance'

--

--

-- Dumping routines for database 'salmaninsurance'

--

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

```

```
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;  
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;  
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

```
-- Dump completed on 2019-11-19 13:23:53
```