

Technical Design Document

PSQL to SQL Server Replication

And

CUSTOMER MARKETING LIST WIZARD

P2S-CMLW-001

|  |  |
| --- | --- |
| Prepared by: | Tamilselvan Palani |
| Last Revised Date: |  |
| Revision: |  |

Contents

[1 Introduction 4](#_Toc481012452)

[2 POC Background 5](#_Toc481012453)

[3 Initial Requirements 6](#_Toc481012454)

[4 Implementation Options 7](#_Toc481012455)

[4.1.1 Description 7](#_Toc481012456)

[4.1.2 Technology Selection 7](#_Toc481012457)

[4.1.3 Architecture 8](#_Toc481012458)

[4.1.4 Challenges 14](#_Toc481012459)

[5 Appendix A 15](#_Toc481012460)

[6 Appendix B 15](#_Toc481012461)

[7 Appendix C 17](#_Toc481012462)

[8 Appendix D 18](#_Toc481012463)

[9 Appendix E 18](#_Toc481012464)

[10 Customer marketing list wizard 19](#_Toc481012465)

[10.1.1 Introduction 19](#_Toc481012466)

[10.1.2 Application Background 19](#_Toc481012467)

[10.1.3 Existing System issue 19](#_Toc481012468)

[10.1.4 Initial Requirement 19](#_Toc481012469)

[10.1.5 Implementation Details 20](#_Toc481012470)

[10.1.6 CMLW Architecture 21](#_Toc481012471)

[Tableau Components 21](#_Toc481012472)

[ODBC Component 22](#_Toc481012473)

[Data Sources 23](#_Toc481012474)

[Filters 24](#_Toc481012475)

[WorkSheet 26](#_Toc481012476)

[10.1.7 Challenges 26](#_Toc481012477)

[11 Team contact list 27](#_Toc481012478)

Document Control

| Revision | Date | Revised By | Description |
| --- | --- | --- | --- |
| 1.0 | 01/12/2016 | Mani Raman |  |
|  |  |  |  |

Contributors

| Name | Email ID | Title |
| --- | --- | --- |
| Mani Raman | [mani@yotabitesllc.com](mailto:mani@yotabitesllc.com) | Chief Technical Officer |
|  |  |  |
| Tamilselvan Palani | [tamil@yotabitesllc.com](mailto:tamil@yotabitesllc.com) | Senior Software Engineer |

## Introduction

The purpose of this document is to outline the technical design of the Database Replication from Pervasive Database to Microsoft SQL Server and provide an overview for the SSIS implementation. Its main purpose is to –

* Provide a basis for the replication of PSQL into MS SQL Server detailed design and development

* Provide a detailed design and implementation of SSIS package
* Provide the detailed design and implementation of Automation in Windows scheduler with SSIS package
* Provide the detailed challenges and limitation of implementation.

## POC Background

The previous implementation of the system was in .Net. Though it was able to handle initial load, it could not accommodate the incremental load and the CDC (Capture Data Change) properly. The .Net implementation is also had performance issues.

The new design and approach is made to overcome the above issues.

## Initial Requirements

The Replication Design outlined in this document builds upon the scope defined in the Requirements phase as Replicate the PSQL DB to MS SQL Server by batch process as near real time.

## Implementation Options

## Description

Database Replication from PSQL to MS SQL Server using the generic connectivity of ODBC and OLE DB respectively, which are provided by software venders. The Microsoft Visual Studio SSIS service used to design the integration package for replicate from Source database tables to Destination database tables.

The designed integration package is deployed into destination database platform using CLI. This implementation will run as a batch process with defined time intervals using Windows scheduler.

## Technology Selection

This design implemented as POC in the following technologies and Tools

|  |  |
| --- | --- |
| Operating System | Windows 64 bit |
| Databases | Pervasive with ODBC 32 bit (64 bit option)  Microsoft SQL Server Express Edition with DTS Tools |
| Development Tools | Microsoft Visual Studio with Integration Service |

## Architecture

Replication architecture defines the various components and their interactions in context of a whole system



The Replication Architecture is consist of following components

* Pervasive Database as Source
* SSIS Package as ETL (Extract, Transform and Load)
* MS SQL Server as Destination
* Windows Task/Job Scheduler

**SSIS Package**

SSIS is an ETL tool (Extract, Transform and Load), which is very much needed for the Data warehousing applications. Also SSIS is used to perform operations like loading the data based on the need, performing different transformations on the data like doing data conversion, calculations (Sum, Average, etc.) and to define a workflow of how the process should flow and perform some tasks on the day to day activity.

SSIS is designed by using MS Visual Studio. This Replication architecture consists of the following SSIS Tasks and Connectivity’s

1. ODBC Source
2. OLE DB Destination
3. Execute SQL Task
4. Data Flow Task
5. Data Converter Task

**ODBC Source**

The ODBC source extracts data from ODBC-supported database by using a database table, a view, or an SQL statement. The ODBC source has the following data access modes for extracting data:

* A table or view.
* The results of an SQL statement.

In this design, ODBC is used to connect PSQL Database and read data from the table by Transact-SQL statements with SSIS Task.

**OLE DB Destination**

The OLE DB destination loads data into a variety of OLE DB-compliant databases using a database table or view or an SQL command. The OLE DB destination provides five different data access modes for loading data:

* A table or view. You can specify an existing table or view, or you create a new table.
* A table or view using fast-load options. You can specify an existing table or create a new table.
* A table or view specified in a variable.
* A table or view specified in a variable using fast-load options.
* The results of an SQL statement.

This architecture, perform to connect MS SQL Sever, write the data into table with SSIS Tasks and Execute the DDL Statement with Execute SQL Task.

**Execute SQL Task**

The Execute SQL task runs SQL statements or stored procedures from a package. The task can contain either a single SQL statement or multiple SQL statements that run sequentially.

You can use the Execute SQL task for the following purposes:

* Truncate a table or view in preparation for inserting data.
* Create, alter, and drop database objects such as tables and views.
* Re-create fact and dimension tables before loading data into them.
* Run stored procedures. If the SQL statement invokes a stored procedure that returns results from a temporary table, use the WITH RESULT SETS option to define metadata for the result set.
* Save the rowset returned from a query into a variable.

In this design, Used to Create and Re-create the tables.

**Data Flow Task**

The Data Flow task encapsulates the data flow engine that moves data between sources and destinations, and lets the user transform, clean, and modify data as it is moved. Addition of a Data Flow task to a package control flow makes it possible for the package to extract, transform, and load data.

A data flow consists of at least one data flow component, but it is typically a set of connected data flow components: sources that extract data; transformations that modify, route, or summarize data; and destinations that load data.

This is used to moves data from PSQL source to SQL Sever destinations

**Data Converter Task**

The Data Conversion transformation converts the data in an input column to a different data type and then copies it to a new output column.

This design used the Data Converter Task to convert the following PSQL data type to SQL SERVER Data type.

See [Appendix B](#_Appendix_B_1)

**Windows Task/Job Scheduler**

The Task Scheduler enables you to automatically perform routine tasks on a chosen computer. The Task Scheduler does this by monitoring whatever criteria you choose to initiate the tasks (referred to as triggers) and then executing the tasks when the criteria is met.

The MS Visual Studio Integration service performs to design the SSIS Package using above SSIS Tasks and component. The created SSIS package is deployed in Windows batch file with DTexec utility

See [Appendix E](#_Appendix_E)

The created batch file is configure in windows task scheduler for perform task as a batch process with defined time intervals.

The current implementation of POC, Windows task scheduler to run the task at every hour.

*4.1.4. High Level Process Flow*

CONTROL TASK

EXECUTE SQL TASK

SQL SERVER - OLE DB

DDL Statement

Drop Tables

Create/Re-Create Tables

PSQL – ODBC

Read Data

DML Statement

**SELECT \* FROM Tables**

SQL SERVER

OLE DB

Write DATA

To Tables

Data Converter

Map the columns to Convert

Data Flow Task

The SSIS Package is started by windows Scheduler, The Execute SQL Task (EST) triggered very first. The EST connects the MS SQL SERVER through OLE DB. Its drop the tables if exists and Re-created the Tables using the DDL statement.

See [**Appendix C**](#_Appendix_C_2)

After completion of Execute SQL Task, the Data Flow Task is trigged. Its read the data from PSQL via ODBC source connection. The return the dataset passed to data converter. The Data Converter has converted the data from PSQL data type to MS SQL data type and stored as copy columns. These copied columns are renamed and passed to SQL SERVER OLE DB Destination to write the data into SQL Server tables one by one.

The current implementation is designed with fixed schema source. The schema changes with required the re-create the SSIS package.

## Challenges

This design is implemented as a POC in the production of more the 20 tables. The following data issues have been addressed at time of implementation of POC.

1. Invalid date format of 9/1/8826 in following columns of WRKORDER tables

duedate, contactdate, promiseddate, dateout

2. The row of agreementnbr 3564,5496 is corrupted in AGRPERIOD.

3. ARTTEMP, BOBTEMP can’t access

The more filters are used to avoid the above issues. It may be degrade the performance.

See [**Appendix D**](#_Appendix_D_1)

**So the above data issue may be fixed the in production to achieve high performance**

## Appendix A

|  |  |
| --- | --- |
| SSIS | Sql Server Integration Service |
| ODBC | Open Database Connectivity |
| OLE DB | Object Linking and Embedding Database |
| CLI | Command Line Interrupter |
| DDL | Data Definition Language |
| DML | Data Manipulation Language |

[BACK](#_Architecture)

## Appendix B

**Data Type Converter**

|  |  |  |
| --- | --- | --- |
| **Pervasive** | **MS SQL(Converter)** | **Description** |
| CHAR VARCHA. | DT\_STR | A null-terminated ANSI/MBCS character string with a maximum length of 8000 characters. (If a column value contains additional null terminators, the string will be truncated at the occurrence of the first null.) |
| BIGINT UBIGINT INTEGER  UINTEGER | DT\_NUMERIC | An exact numeric value with a fixed precision and scale. This data type is a 16-byte unsigned integer with a separate sign, a scale of 0 - 38, and a maximum precision of 38. |
| TINYINT UTINYINT | DT\_I1  DT\_I2 | A one-byte, signed integer.  A two-byte, signed integer.  A four-byte, signed integer. |
| SMALLINT  USMALLINT | DT\_I4  DT\_I8 | A four-byte, signed integer.  An eight-byte, signed integer. |
| DATE | DT\_DATE  DT\_DBDATE | A date structure that consists of year, month, day, hour, minute, seconds, and fractional seconds. The fractional seconds have a fixed scale of 7 digits.  A date structure that consists of year, month, and day. |
| TIME | DT\_DBTIME | A time structure that consists of hour, minute, and second |
| TIMESTAMP | DT\_DBTIMESTAMP | A timestamp structure that consists of year, month, day, hour, minute, second, and fractional seconds. The fractional seconds have a maximum scale of 3 digits. |
| DECIMAL  NUMERIC  NUMERICSA  NUMERICSTS | DT\_DECIMAL | An exact numeric value with a fixed precision and a fixed scale. This data type is a 12-byte unsigned integer with a separate sign, a scale of 0 to 28, and a maximum precision of 29. |
| * MONEY CURRENCY | DT\_CY | A currency value. This data type is an eight-byte signed integer with a scale of 4 and a maximum precision of 19 digits. |
| FLOAT  REAL  DOUBLE BFLOAT4 BFLOAT8 | DT\_R4  DT\_R8  DT\_DECIMAL(Optional) | A single-precision floating-point value.  A double-precision floating-point value. |
| * LONGVARCHAR | DT\_NTEXT | A Unicode character string with a maximum length of 2^30 - 1 (1,073,741,823) characters. |
| BINARY LONGVARBINARY | DT\_BYTES | A binary data value. The length is variable and the maximum length is 8000 bytes. |

[BACK](#_Architecture)

## Appendix C

**DDL Statement**

DROP TABLE AGREEMENT;

CREATE TABLE [AGREEMENT] (

[SS\_AGREEMENTNBR] int,

[SS\_ALTAGRNBR] varchar(10),

[SS\_AGREEMENTSEQ] int,

[SS\_CUSTOMERNBR] int,

[SS\_BILLINGNBR] int,

[SS\_CENTERNBR] int,

[SS\_SALESMANNBR] int,

[SS\_TECHNBR] int,

[SS\_DATEENTER] date,

[SS\_TIMEENTER] datetime,

[SS\_ENTERBY] int,

[SS\_PAYMETHOD] int,

[SS\_CUSTOMERPO] varchar(20),

[SS\_CONTACT] varchar(25),

[SS\_CREDCARDTYPE] int,

[SS\_CREDCARDNBR] varchar(17),

[SS\_CREDCARDEXP] smallint,

[SS\_QTAXABLE] varchar(1),

[SS\_QTAXATCENTER] varchar(1),

[SS\_TAXCODE] int,

[SS\_COMMENTS] varchar(120),

[SS\_LEADNBR] int,

[SS\_LEADSRCNBR] int,

[SS\_REPORTNBR] int,

[SS\_DATEMOD] date,

[SS\_TIMEMOD] datetime,

[SS\_MODBY] int,

[SS\_QINACTIVE] varchar(1),

[SS\_QSPARE] varchar(1),

[SS\_PPROFNBR] int,

[SS\_SPARE] varchar(46));

[BACK](#_Architecture)

## Appendix D

**DML Statement**

SELECT \* FROM wrkorder

WHERE duedate IS NULL

AND contactdate IS NULL

AND promiseddate IS NULL

AND dateout IS NULL;

SELECT \* FROM AGRPERIOD

WHERE ( (startdate >={d '1966-8-27'} AND startdate <={d '2001-2-28'})

OR (startdate >={d '2001-3-1'} AND startdate <={d '2098-12-31'}))

AND ( (expiredate >={d '1970-1-1'} AND expiredate <={d '2001-2-28'})

OR (expiredate >={d '2001-3-1'} AND expiredate <={d '9999-12-31'}))

AND ( (quoteexpdate >={d '1970-1-1'} AND quoteexpdate <={d '2001-2-28'})

OR (quoteexpdate >={d '2001-3-1'} AND quoteexpdate <={d '9999-12-31'}))

AND (agreementnbr <> 3564 AND agreementnbr <> 5496)

[BACK](#_Architecture)

## Appendix E

**Windows Batch File**

The batch file consist of following script

E:

cd \SQL Server\SQL\110\DTS\BInn

DTEXEC /F C:\Your\Location\of\Batchfile.bat

[BACK](#_Architecture)

## Customer marketing list wizard

## Introduction

The purpose of this document is to outline the technical design of the Customer Marketing List Wizard (CMLW) using tableau. This is the breakdown of the entire process:

* Define of Data Source from MS SQL Server to tableau
* Outline the detailed design and implementation of tableau worksheet.
* List the implementation of filters.
* Detail the various challenges and limitation of implementation.

## Application Background

This application was drafted with new design and approach to address the issues and limitations with the existing system as detailed below.

## Existing System issue

The existing application is implemented using old technology. While it could work with only Pervasive Database, the existing system has these other limitations :

* Non-User Friendly interface
* No Interactive reports

Selection of filters using too many navigations of screens

## Initial Requirement

The CMLW Design outlined in this document builds upon the scope defined in the Requirements Document.

## Implementation Details

**Connectivity**

CMLW of the source tables from MS SQL Server is designed to happen using the generic connectivity of ODBC, which is provided by the respective software venders.

**Tableau Tool**

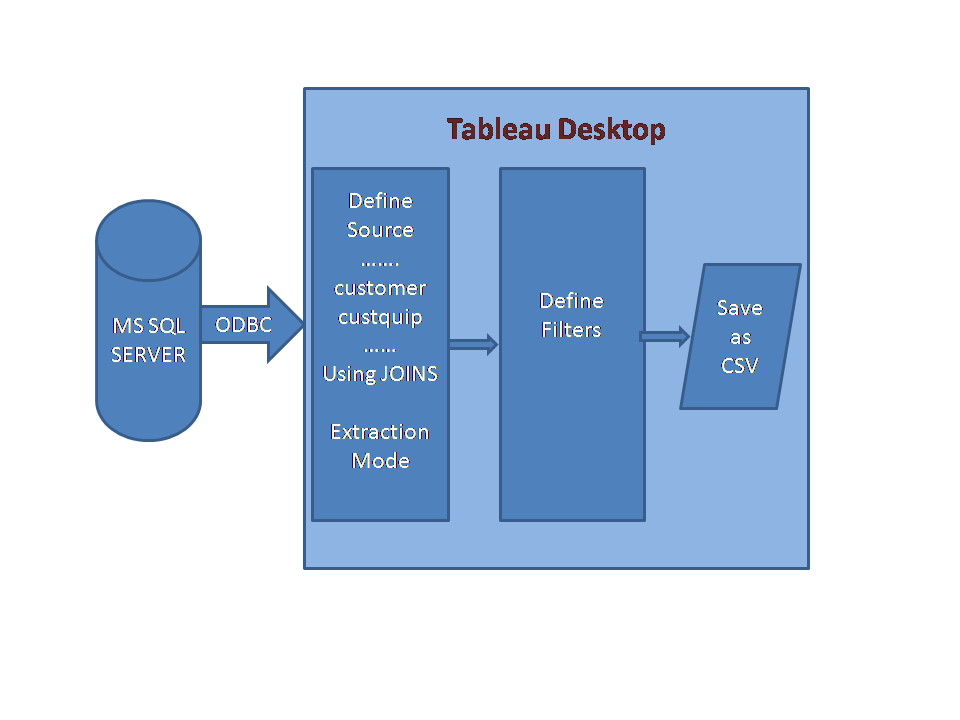
Tableau desktop version provided design mode and Presentation mode. The design mode is used to create a work sheet for CMLW. The Presentation mode used for end user to create the CMLW and save as csv.

**Technology Used**

|  |  |
| --- | --- |
| Operating System | Windows 64 bit |
| Source Database | Microsoft SQL Server |
| Development Tools | Tableau Desktop |

## CMLW Architecture

CMLW architecture defines the various components and their interactions in context of a whole system



## Tableau Components

This CMLW architecture consists of the following components:

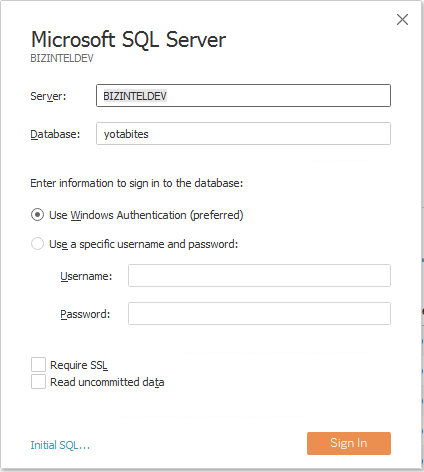
* ODBC Components
* Data Sources
* Filters
* Worksheet

## ODBC Component

The ODBC component is used to connect MS SQL database table, a view, or an SQL statement. The ODBC component has the following data access modes

* A table or view.
* The results of an SQL statement.

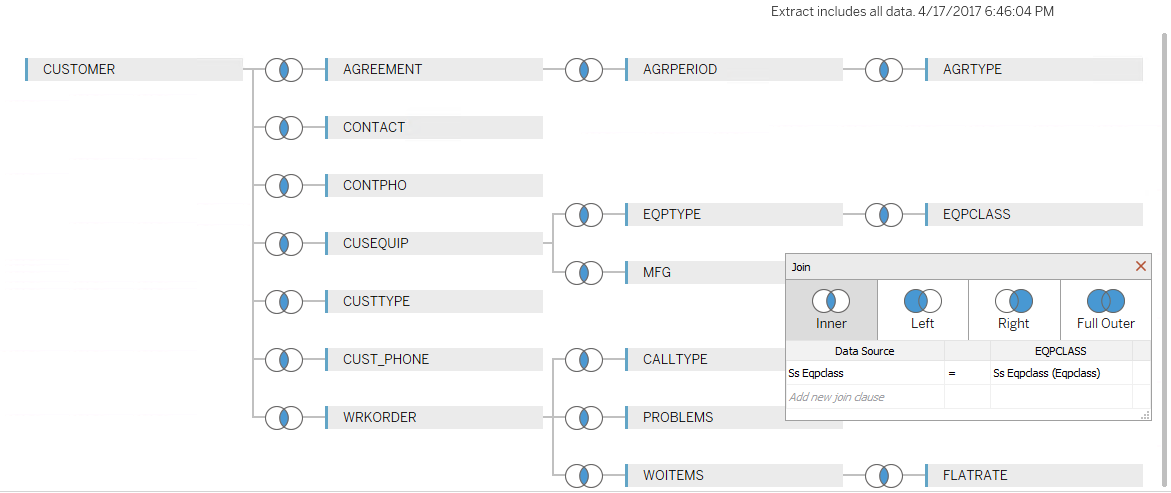
In this design, ODBC is used to connect to MS SQL Server to read data from the table by using a tableau data source components.



## Data Sources

The Data source is used to create joins of required tables from MS SQL server via ODBC components. Extract the data from this data source using the Extract mode.

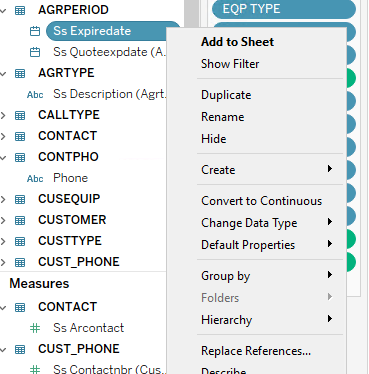
The following tables are used to create join (inner)



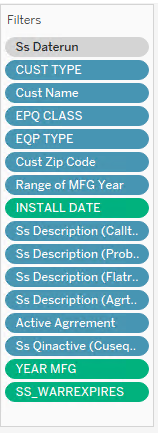
## 

## Filters

From the data source we can apply the filters for varies condition which is described in the requirement documents to generate the CMLW. The creating filters are used some customized the calculated filed as per requirement documents. To create a filter, Right click on the dimension column and select show filter.



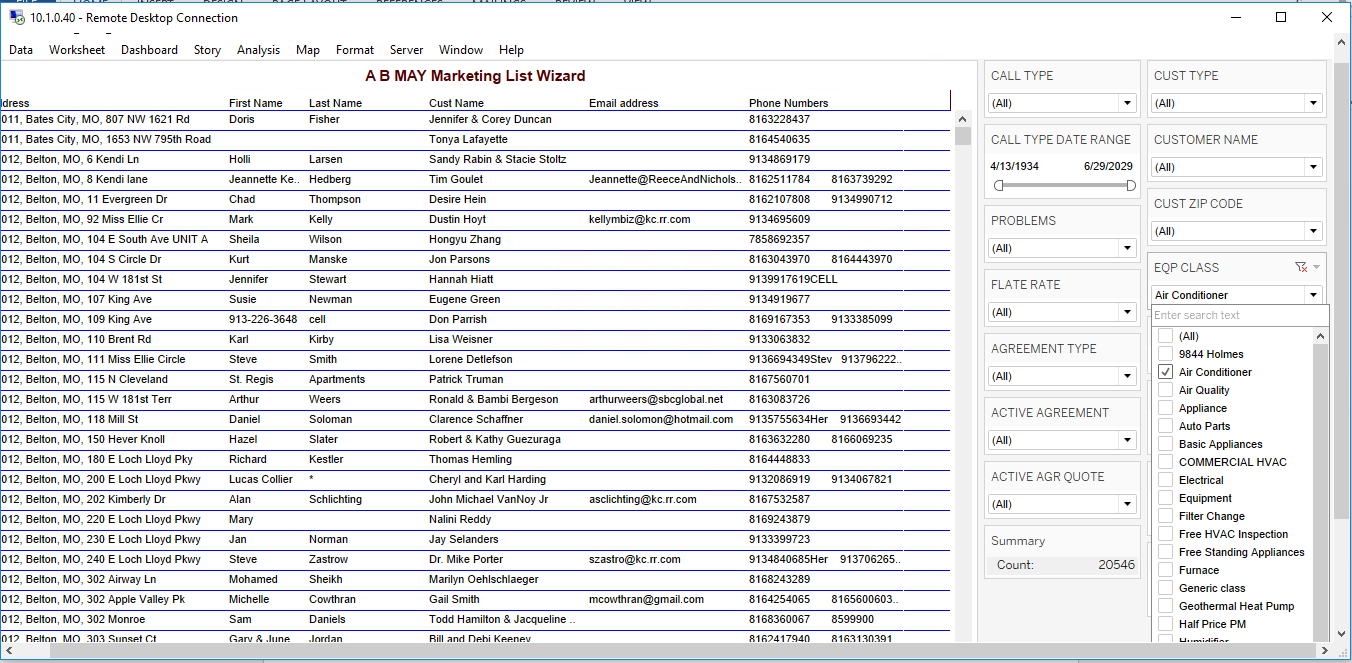
This implementation uses the following filters.



## 

## WorkSheet

The worksheet is used to embed the dimension columns, filters and generate the interactive report. The results are changed as the filter values are selected. The report is saved as csv from this worksheet.



## Challenges

This design is implemented in the production. The Tableau desktop version does not have automation capabilities. So, this design is not implemented automation.

## 11 Team contact list

|  |  |  |
| --- | --- | --- |
| **Contact Name** | **Job Title** | **Email ID** |
| MANI RAMAN | Chief Technical Officer | [mani@yotabitesllc.com](mailto:mani@yotabitesllc.com) |
| ATUL | VP | atulk@yotabitesllc.com |
| TAMILSELVAN PALANI | Senior Software Engineer | [tamil@yotabitesllc.com](mailto:tamil@yotabitesllc.com) |