The major components of this pipeline are the Sqoop jobs, the Hive tables and queries, the Oozie scheduler, and Power Query / Power View to fetch the data and create reports on top of it.

**Oozie** – The workflow is laid out in **workflow.xml**. In this particular workflow, the replication data is deleted, then the sqoop jobs are executed to get the updated replication data, and then the hive queries are executed to do some cleansing/aggregating of the data. The **coordinator.xml** takes care of executing the workflow, telling the start and end times, as well as the frequency of how often the workflow should run. The workflow.xml, coordinator.xml, as well as the Hive query files are uploaded via the **oozie\_prep.ps1** script. The oozie job is started by running the **oozie\_run.ps1** script.

**Sqoop** – Sqoop transfers from SQL databases to blob storage can be done in two ways. For testing and incremental development, you can start a job manually through the Azure PowerShell. An example script is uploadTable.ps1.

Otherwise, the sqoop jobs can be automated and run through the oozie scheduler. In certain <action/> nodes of workflow.xml, you can see how a sqoop job is outlined.

**Hive** – The two interactions with Hive in this pipeline are creating the metatables on top of the replication data, and running Hive queries to cleanse/aggregate the data. I usually created the *external* tables straight through the Hive editor. You can access the hive editor through (Azure Management Portal 🡪 [click on cluster] 🡪 Manage Cluster).

Sometimes the hive editor is buggy and refuses to execute anything, giving ambiguous errors. In this case, you can enable Remote Desktop Connection for the cluster (Azure Management Portal 🡪 [click on cluster] 🡪 Configuration 🡪 Enable Remote). After opening the remote connection, open the Hadoop command line. The following commands will open up a Hive console where you can enter any Hive commands.

cd %HIVE\_HOME%/bin

hive

The hive files that contain the queries that oozie will execute are: **activityphonejoin\_query.hql, activitypointerbase\_query.hql,** and **phonecallbase\_query.hql.**

The files that contain the hive ‘create external table’ commands are **create\_activitypointerbase\_hive.hql,** and **create\_phonecallbase\_hive.hql.**

**Power Query to Power Pivot / Power View -**

The following steps are how to import Hive Query results using Power Query

1. Under Power Query tab, click **From Other Sources 🡪 From Windows Azure HDInsight**
2. Enter the name of the storage account associated with the cluster
3. Enter the key associated with the storage account
   1. The key can be found in the Azure Management Portal. Click on the storage account, and click “Manage Keys”
4. In the resulting Navigator Pane, click on the cluster name
5. A list of cluster contents will appear. Since the Hive results are written into specified directories, like “JoinActivityPhoneQuery” for example, you can filter by folder path to find the files that contains those specific query results.
6. After filtering, the contents will be part of the binaries labeled **00000\_0, 00000\_1**, etc., depending on how large the query is.
7. Click on the **binary** of the contents, and open the file as a CSV document.
8. This view will show the contents in a table form.
9. You must rename the columns so they are more readable.
10. Most of the types are as strings, so some of them can be changed so they work better with power pivot and power view down the line, like text 🡪 DateTime for example.
11. Rename the results, and “Apply and Close” to load contents into Excel Workbook.
12. Once the relevant queries are loaded into the workbook, you can add relationships and calculated metrics using Power Pivot, and generate reports using Power View.

**Miscellaneous**

**sql\_to\_hive.py** is a python script that takes an input file of SQL column names and types, what is required for a create table command in SQL, and outputs a file that contains the equivalent hive column names and types. It takes an output filename as another optional parameter.

Example use: ./python sql\_to\_hive.py input.txt [phoneTableHiveTypes.txt]

Example input file contents:

[OwningBusinessUnit] [uniqueidentifier] NULL,

[ActualEnd] [datetime] NULL,

[VersionNumber] [timestamp] NULL,

[ActivityId] [uniqueidentifier] NOT NULL,

[IsBilled] [bit] NULL,

[CreatedBy] [uniqueidentifier] NULL,

[Description] [nvarchar](max) NULL,

……. etc.