ETL Project

**Introduction**

Welcome to the **course project**. This project will test your knowledge of the various tools related to batch processing, which you have learnt throughout this course. The project mainly revolves around **Apache Sqoop, Apache PySpark, Amazon S3 and Amazon RedShift**, which are some of the most widely used tools in the industry.

In this video, our expert will give a brief introduction to the ETL project that we will be going through in this module.

In this project, you will go through a real-world use case from the banking sector.

Your task, essentially, would be to build a batch ETL pipeline to read transactional data from RDS, transform it and then load it into Redshift Tables, after which you will have to perform some analytical queries on the loaded data.

People you will hear from in this project

**Subject Matter Expert**

[Ganesh Gurusiddaiah](https://www.linkedin.com/in/ganesh-gurusiddaiah-b5634348/)

**Big Data Technology Lead**

Ganesh is a Big Data ETL Leader with expertise in data warehouse solutions. He has almost 10 years of experience in the software domain and holds a master’s degree from BITS Pilani in Software Systems.

**Problem Statement**

In this segment, you will get to know more about the problem statement for the ETL project, which you will be going through in this project.

In the next video, our expert will walk you through the detailed problem statement for the ETL project.

As discussed in the video, banks have to refill the ATMs when the money goes below a specific threshold limit.

This depends on the activity and the area where a particular ATM is located as well as the weather, day of the week, etc.

In our project, **Spar Nord Bank** is trying to observe the withdrawal behavior and the corresponding dependent factors to optimally manage the refill frequency. Apart from this, other insights also have to be drawn from the data.

Coming to the analysis part, you will be tasked to carry out the calculations to perform the following analytical queries:

* Top 10 ATMs where most transactions are in the ’inactive’ state
* Number of ATM failures corresponding to the different weather conditions recorded at the time of the transactions
* Top 10 ATMs with the most number of transactions throughout the year
* Number of overall ATM transactions going inactive per month for each month
* Top 10 ATMs with the highest total amount withdrawn throughout the year
* Number of failed ATM transactions across various card types
* Top 10 records with the number of transactions ordered by the ATM\_number, ATM\_manufacturer, location, weekend\_flag and then total\_transaction\_count, on weekdays and on weekends throughout the year
* Most active day in each ATMs from location "Vejgaard"

Your overall task in this project will be to build a batch ETL pipeline to read transactional data from RDS, transform and load it into target dimensions and facts on Redshift**Data Mart(Schema)**.

**Please note that the source data and target schema details are provided to better understand the source and targets, which would help design the ETL pipeline.**Once the data is loaded into Redshift, you would have to write the analytical queries discussed above.

We have data from more than 100 ATMs across Denmark. Data is captured for every transaction including, card type, location, date, time, ATM type, etc.

Also, the transaction amount field in the data set was added separately using a random function for the analysis purpose.

Spar Nord Bank has also built a dimensional model datastore (**ATM Data Mart**) on this ATM transaction data to understand the ATM usage pattern. This exact schema(target schema) of this Data Mart will be provided to you for the sake of this project. Basically, this will be the schema according to which you will be creating tables in Redshift.

**Broadly you will be performing the following task-**

* Extracting the transactional data from a given MySQL RDS server to HDFS(EC2) instance using Sqoop.
* Transforming the transactional data according to the given target schema using PySpark.
* This transformed data is to be loaded to an S3 bucket.
* Creating the Redshift tables according to the given schema.
* Loading the data from Amazon S3 to Redshift tables.
* Performing the analysis queries.

In the next segment, you will learn more about the data set, which you will be dealing with in this project.

Additional Content

* If you want to get clarity on the concepts of **Dimension Model** and **Data Mart,** you can refer to the relevant segments in the **Data Warehousing and ETL** module. Anyways the Dimensional Model(target schema) to be used in this project will be provided to you.
* [Data Marts](https://docs.oracle.com/cd/A81042_01/DOC/server.816/a76994/marts.htm) - A data mart is a simple form of a data warehouse that is focused on a single subject (or functional area), such as Sales or Finance or Marketing.
* [Dimensional modelling](https://en.wikipedia.org/wiki/Dimensional_modeling) - Dimensional modelling includes a set of methods, techniques and concepts for use in data warehouse design.

Additional Soft-Skills Content

Navigate to the respective links to access the soft skills content related to the [Approach to Problem Solving](https://learn.upgrad.com/course/3161?courseId=32823), [Aids to Problem Solving](https://learn.upgrad.com/course/3161?courseId=32823) & [Job Interview Skills](https://learn.upgrad.com/course/3161?courseId=32822)

**Dataset Description**

Now, in this segment, you will get to know about the data set that you are dealing with in this project. Our expert Ganesh will walk you through the details of the data set in the next video.

Play Video

4031055

As discussed in the video, you will be working on a **Danish ATM Transactions Data Set** in this project.

This dataset comprises around **2.5 million records** of withdrawal data along with weather information at the time of the transactions from around **113 ATMs from the year 2017**.

The actual data set is divided into **two part files**, both amounting to about **503** MB in total.

There is also a data dictionary present along with the data set, which defines all of the **33 columns**present in the data set. The data dictionary is given below.

**RDS Data Dictionary**

**Download**

This data set contains various types of transactional data as well as the weather data at the time of the transaction, such as:

* **Transaction Date and Time**: Year, month, day, weekday, hour
* **Status of the ATM**: Active or inactive
* **Details of the ATM**: ATM ID, manufacturer name along with location details such as longitude, latitude, street name, street number and zip code
* **The weather of the area near the ATM during the transaction**: Location of measurement such as longitude, latitude, city name along with the type of weather, temperature, pressure, wind speed, cloud and so on
* **Transaction details:** Card type, currency, transaction/service type, transaction amount and error message (if any)

 As discussed in the video, you will be working on a **Danish ATM Transactions Data Set** in this project.

This dataset comprises around **2.5 million records** of withdrawal data along with weather information at the time of the transactions from around **113 ATMs from the year 2017**.

The actual data set is divided into **two part files**, both amounting to about **503** MB in total.

There is also a data dictionary present along with the data set, which defines all of the **33 columns**present in the data set. The data dictionary is given below.

This data set contains various types of transactional data as well as the weather data at the time of the transaction, such as:

* **Transaction Date and Time**: Year, month, day, weekday, hour
* **Status of the ATM**: Active or inactive
* **Details of the ATM**: ATM ID, manufacturer name along with location details such as longitude, latitude, street name, street number and zip code
* **The weather of the area near the ATM during the transaction**: Location of measurement such as longitude, latitude, city name along with the type of weather, temperature, pressure, wind speed, cloud and so on
* **Transaction details:** Card type, currency, transaction/service type, transaction amount and error message (if any)

Acknowledgement

[Spar Nord Bank](https://www.sparnord.dk/) has published this dataset at [Kaggle](https://www.kaggle.com/) under [Database Contents License (DbCL) v1.0 — Open Data Commons](https://opendatacommons.org/licenses/dbcl/1-0/).

Reference

Source of the Data Set in Kaggle: [Kaggle Data Set](https://www.kaggle.com/sparnord/danish-atm-transactions)

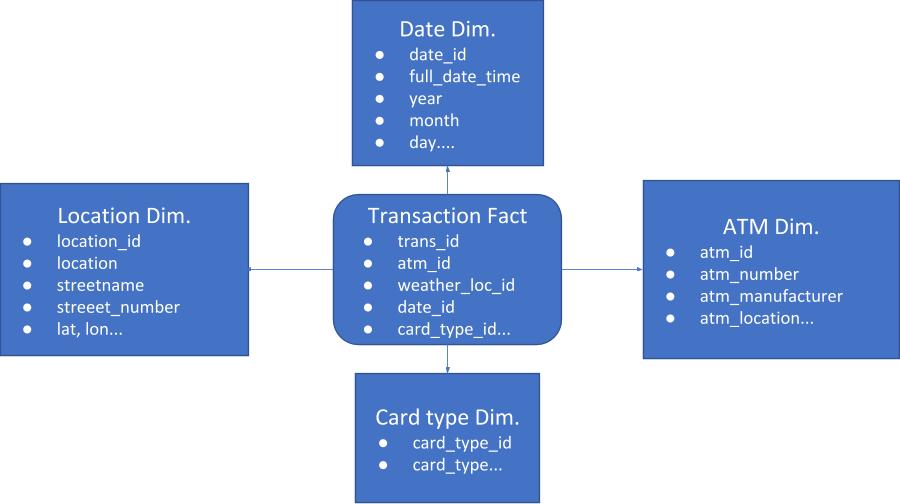
**Target Dimension Model**

Now, in this segment, you will get to know about the**target schema(dimension model)** according to which you will have to transform the data set using Spark as well as create and load it accordingly in RedShift.  Simply put this will be the schema according to which you'll create the tables in Redshift.

In the next video, our expert will walk you through the target dimension model.

If you want to read about the dimensions and facts please refer to the Additional Resources.

As discussed in the video, you will be following the provided target dimension model schema for this ETL project.



Target Dimension Model

For this project, you will need four dimension tables and one fact table. They are as follows:

* **ATM dimension** - This dimension will have the data related to the various ATMs present in the dataset along with the ATM number(ATM ID in the original dataset), ATM manufacturer and a reference to the ATM location and is very important for solving analytical queries related where ATM data will be used.
* **Location dimension**- This is a very important dimension containing all the location data including location name, street name, street number, zip code and even the latitude and longitude. This information will be very important for solving problems related to the particular location at which a transaction took place and can help banks in things like pinpointing locations where ATMs where demand is higher as compared to other locations. Combined with weather data in the transaction table, this can be used to further do analysis such as how weather affects the demand at ATMs at a particular location.
* **Date dimension** - This is another very important dimension which is almost always present where data such as transactional data is being dealt with. This dimension includes fields such as the full date and time timestamp, year, month, day, hour as well as the weekday for a transaction. This all can help in analysing the transaction behaviour with respect to the time at which the transaction took place and also how the transaction activity varies between weekdays and weekends.
* **Card type dimension** - This dimension has the information about the particular card type with which a particular transaction took place. This can help in performing analysis on how the number of transactions varies with respect to each different card type.
* **Transaction fact** - This is the actual fact table for the data set which contains all of the numerical data such as the currency of the transaction, service, transaction amount, message code and text as well as weather info such as description, weather id etc.

The schemas for these tables are provided in the following document.

**Schema for Dimensions and Fact**

**Download**

The analytical queries defined in the next segment will also be based on the Redshift tables.

In the next segment, you will look at the solution approach as well as the tasks and analytical queries, which you will solve in this ETL project.

Additional Resources

* If you want to get clarity on the concepts of **Dimensions & Fact Tables**, you can refer to the relevant segments in the Data Warehousing and ETL module. Anyways the Dimensional Model(target schema) to be used in this project will be provided to you.
* [Dimension](https://en.wikipedia.org/wiki/Dimension_(data_warehouse)) - This is a Wikipedia page describing the concept of Dimension in a Data Warehouse
* [Fact Tables](https://en.wikipedia.org/wiki/Fact_table) - This is a Wikipedia page describing the concept of Fact Tables in a Data Warehouse
* **Solution Approach and Tasks**
* In the next video, our expert will walk you through the entire Solution Approach for this project.

As discussed in the video, the tasks needed to be performed for this project are as follows:

Sqoop

* **Data ingestion from RDS to HDFS using Sqoop**
  + Sqoop import command
  + Command to see the list of imported data in HDFS

For this step, the RDS connection string and credentials are as follows:

* RDS **Connection String** -

jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase

* **Username** - student
* **Password** - STUDENT123
* **Table Name** - SRC\_ATM\_TRANS

PySpark

* **Reading the data from the files in HDFS by a specific schema using PySpark**
  + Command to create an input schema using **StructType**(We recommend you to create a custom schema using the StructType class of PySpark, to avoid any data type mismatch.)
  + Commands to read the data using the input schema created and verifying the data using the count function
* **Creation of dimension tables using PySpark**
  + Command to create a data frame for the dimension according to the target schema(dimension model) provided
  + Commands to clean and transform the data:
    - Making sure that duplicate records are cleaned(Avoid duplicate info especially in the dimension tables.
    - Making sure that appropriate primary keys are present for the dimensions( You need to generate a primary key for each dimension table. For example for the '**Date**' dimension one way to generate the primary key can be by adding "date" as the prefix  to the row number i.e. 'date1', 'date2' and so on.)
    - Rearranging the fields if necessary(According to the target schema)

**Note:** Here, the tasks given above have to be done for all four dimension tables.

* **Creation of transaction fact table using PySpark**
  + Commands to set proper alias for the various **PySpark DataFrames** before proceeding with creating the fact table (**optional**)
  + Commands for various stages where the original data frame is appropriately joined with the dimension tables created above
  + Commands to clean and transform the data:
    - Making sure that the appropriate primary key is present for the fact table
    - Rearranging the fields if necessary
* **Loading the dimension and fact tables into Amazon S3 bucket**
  + Write the DataFrames containing the dimensions and fact table directly to an S3 bucket folder. [You should create different folders on your S3 bucket for different dimensions and fact table.]

Redshift

* **Creation of a Redshift Cluster**
  + You need to create a Redshift cluster in the same way as it was done in the **Amazon Redshift**module.
* **Setting up a database in the Redshift cluster and running queries to create the dimension and fact tables**
  + Queries to create the various dimension and fact tables with appropriate primary and foreign keys
* **Loading data into a Redshift cluster from Amazon S3 bucket**
  + Queries to **copy** the data from S3 buckets to the Redshift cluster in the appropriate tables
* **Using queries on a Redshift cluster to find the solution to the following analytical queries.**
  + Top 10 ATMs where most transactions are in the ’inactive’ state
  + Number of ATM failures corresponding to the different weather conditions recorded at the time of the transactions
  + Top 10 ATMs with the most number of transactions throughout the year
  + Number of overall ATM transactions going inactive per month for each month
  + Top 10 ATMs with the highest total amount withdrawn throughout the year
  + Number of failed ATM transactions across various card types
  + Top 10 records with the number of transactions ordered by the ATM\_number, ATM\_manufacturer, location, weekend\_flag and then total\_transaction\_count, on weekdays and on weekends throughout the year
  + Most active day in each ATMs from location "Vejgaard"

**\*\*\***

**Important Note:** Amazon Redshift is a costly service of AWS. Hence to avoid burning-up your monthly AWS budget, Please make sure to **Pause** your Redshift cluster during the project. And make sure to terminate it once you are done with the project.

Please follow the steps below to delete the Redshift cluster. This must be done to avoid burning up your AWS budget.

**How to Delete the Redshift Cluster**

**Download**

The steps to create a Redshift cluster are straight forward, hence we would highly recommend you to create a new cluster every time you want to go through the RedShift Analysis Queries(if possible) and delete the cluster once you are done.  Please plan your practice accordingly so that you are able to sit in long stretch for the practice.

**\*\*\***

For analytical tasks at the end, the tasks to be performed are to provide the appropriate queries along with the screenshot of the result (limit to the first page of the query result page if the result is bigger than that). All the Queries are designed in a way that the maximum records that should be outputted should not be more than 15.

Refer to the sample solution format given in the ‘**Submission**’ segment of the next ‘**Submission Guidelines**’ session to get more clarity on the things to be submitted.

Finally, in the next video, our expert would like to conclude the introduction of the project and also give a few tips before you start developing the ETL project.

**Additional Resources**

In this segment, you will learn some concepts in Spark SQL, such as Window function, Partition and row\_number, and you will understand how to use them. These concepts will help you while working on the project.

row\_number() - Spark SQL

During the creation of Dimension tables, you will have to add a row number/index to each table so that a primary key can be established.

Spark SQL provides a function called **row\_number()** as part of the **window**functions group, which can be used to assign a sequential integer number to each row in a DataFrame according to the partition decided.

Let's assume that you have a dataset of employee information of a certain company. You have a dataframe called **emp\_df**having the following data:

| employee table | | |
| --- | --- | --- |
| **emp\_name** | **dept** | **salary** |
| Abhishek | Sales | 1000 |
| Kumar | Legal | 2000 |
| Saif | Finance | 1200 |
| Singh | HR | 1500 |

You can use the following code to add another column, having sequential numbers as values, to the DataFrame:

**from** **pyspark.sql.types** **import** \*

**from** **pyspark.sql.window** **import** Window

**import** **pyspark.sql.functions** **as** **F**

**from** **pyspark.sql.functions** **import** row\_number

res\_df = emp\_df.select("emp\_name","dept","salary",F.row\_number().over(Window.partitionBy().orderBy(res\_df['emp\_name'])).alias("index"))

This code snippet will essentially add another column at the end of the table called **index**, which will have sequential integer values. The final DataFrame would look as follows:

| employee table final dataframe | | | |
| --- | --- | --- | --- |
| **emp\_name** | **dept** | **salary** | **index** |
| Abhishek | Sales | 1000 | 1 |
| Kumar | Legal | 2000 | 2 |
| Saif | Finance | 1200 | 3 |
| Singh | HR | 1500 | 4 |

Using this method, you will be able to create a unique value for each row of your dimension tables.

The concepts learned in this segment will help you during the creation of the dimensions and fact tables.

**Submission Guidelines**

Submissions Required

Upload a **zip** file containing:

**Document-01:** A PDF document(**SqoopDataIngestion.pdf**) containing the Sqoop Code used for ingesting data from the RDS server. This should have the code for ingestion along with the screenshots of the EC2 instance showing the list of the files generated in the HDFS cluster along with proper explanation and comments.

The Sample template for this document is as follows:

**Sample Template - SqoopDataIngestion**

**Download**

**Document-02:**  A Jupyter Notebook(**SparkETLCode.ipynb**) containing the PySpark codes to read the data to Spark, Creating the Dimension and Fact tables and then loading them to S3 bucket. The Jupyter Notebook should be properly commented and should explain all the steps taken. It should also contain appropriate Markup cells for explanation purposes.

**Document-03:** A PDF document(**RedshiftSetup.pdf**) containing the following:

Screenshots of the configuration of the Redshift cluster that you create for the project

Queries used for creating the Dimension and Fact tables on the Redshift cluster along with screenshots of the successful status of the query

Queries used for loading the data into the Dimension and Fact tables in the Redshift cluster from the S3 bucket along with screenshots of the successful status of the query

The Sample template for this document is as follows:

**Sample Template - RedshiftSetup**

**Download**

**Document-04:** A PDF document(**RedshiftQueries.pdf**) containing the queries used for solving the Analytical queries in the Redshift cluster. It should also have the screenshots of the first page of the Tables which are produced after running the queries on Redshift.

The Sample template for this document is as follows:

**Sample Template - RedshiftQueries**

**Download**

**Please make sure that you are not changing any of the file names that have been provided above in brackets. The code that you are submitting should run at our end without any modifications in the code.**

**Make sure that you have not made any changes to the original data set. You will be graded based on the queries and documentation submitted.**

**Final Submission**

For submissions obtained within 1 week of the deadline, there will be a 30% penalty. Submissions beyond 1 week of the deadline will be provided only feedback and score zero marks (100% penalty).

You must go through these guidelines-

1. Make sure you have not made any changes to the original dataset provided to you. Your  code should work on the dataset given to you as part of the problem statement. You are not allowed to make modifications in data set using excel and then use it in your Python code. Entire data processing must be done in as mentioned only. During grading we will be running your code on the dataset provided by us, in case your code gives errors with that, then marks will be deducted accordingly.
2. All penalties are automatically applied by the system based on time of submission. Hence, submissions that are late, even by a second, will attract penalties.

For e.g.- If the deadline is 2nd August 2019, 11:59:00 PM IST, the submissions at 2nd August 2019, 11:59:01 PM IST will attract a penalty of 30%. Hence we recommend that assignments are submitted at least 30 minutes before the deadline to avoid any last minute issues.

Also, note the all the deadlines are in IST (UTC +5.5), hence, if you are in a different time zone, then your deadline may vary according to local time. For eg - If you are in London and following BST (British Summer Time) which is UTC +1 then deadline for you in local time would be 7:29:00 PM BST when the deadline in India is 11:59:00 PM IST.

1. Make sure you click “Submit for Grading” only if you are 100% sure, else you must just upload your file and leave it there. It will be automatically submitted before the deadline. If you click “Submit for grading” after uploading the file, under no circumstance you will be allowed to resubmit or change your upload.

Here are the steps that you must follow during submitting any assignment-

1. Collect all the files (if there are multiple files) and compress(zip file) them together.
2. Try to upload this compressed file latest by 11:30 PM
3. Download your submission and check that you have included all the required files.
4. Check that none of the files or the zip is corrupt. If it is found to be corrupt during grading, you will NOT be allowed to re-submit.
5. If you are 100% sure that you will not need to make any more changes in the assignment, click “Submit for Grading”, else, just let it be. Unless you remove it, it will be automatically submitted at the upcoming deadline.