A REPORT

ON

DATA WAREHOUSING IN IOT ANALYTICS

By

SOUMYADEEP BASU

AT

Samsung Research Institute, Bangalore

A Practice School II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,

PILANI (June, 2021)

A REPORT

ON

DATA WAREHOUSING IN IOT ANALYTICS

By

Name of the student	**ID No.**	**Discipline**
Soumyadeep Basu	2017B5A31056H	Electrical and Electronics Engineering

Prepared in the partial fulfilment of the

Practice School II Course

AT

Samsung Research Institute, Bangalore

A Practice School II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(June, 2021)

Acknowledgements

I am highly indebted to my mentor Praveen Kumar Charugundla, Staff Engineer, IoT Analytics, SRI-Bangalore and Amogha D Shanbag Staff Engineer, IoT Analytics, SRI-Bangalore for providing me such a wonderful opportunity to work on this project. I owe my sincere gratitude to them for their constant guidance, invaluable suggestions and advice at every stage of the project.

I am grateful to Dr. Lucy J. Gudino, Associate Professor, BITS Pilani for her constant guidance and availability throughout the PS II program.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

Practice School Division

Station: Samsung Research Institute	**Centre:** Bangalore
Duration: 5 months	**Date of Start:** 19th July, 2021

Date of Submission: 20th September, 2021

Title of the Project: Data warehousing in IoT Analytics

ID No: 2017B5A31056H
Name: Soumyadeep Basu
Discipline: Electrical and Electronics Engineering

Expert Details

1. Praveen Kumar Charugundla, Staff Engineer, IoT Analytics, SRI-Bangalore
2. Amogha D Shanbag Staff Engineer, IoT Analytics, SRI-Bangalore

Name of the PS Faculty: Lucy J. Gudino

Key Words: ETL, MySQL, Amazon Redshift, Database Management

Project Areas: Data Warehousing

Abstract: The project discusses the analysis of IoT data in Data Warehousing. The entire project is schematically divided into three parts to Extract, Transform and Load (ETL) data using an end-to-end Data Pipeline. We are consuming IoT data from a system of sensors in a home and are extracting data from the dataset using a data pipeline. The extracted data is observed to identify the key performance indicators (KPIs) which are transformed using various SQL queries. The transformed data, hence, is loaded into a data warehouse for further consumption by a dashboard. The system is constructed in the Amazon Web Services (AWS) platform. This entire process is scheduled daily at a particular time by a scheduler named Airflow. The system, hence, derives analytics over KPIs on a particular dataset using the ETL processes.

Signature of Student **Signature of PS Faculty**

Date

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)

PRACTICE SCHOOL DIVISION

Response Option Sheet

Station: Samsung Research Institute **Center**: Bangalore

ID No. & Name: 2017B5A31056H, Soumyadeep Basu

Title of the Project: Data warehousing in IoT Analytics

Some portion of this project can be added to the Internet of Things and help understand the practical applications of such protocols. The basic concepts are taught in the DBMS course.

Code No.	Response Option	Course No.(s) & Name
1.	A new course can be designed out of this project.	
2.	The project can help modification of the course content of some of the existing Courses	
3.	The project can be used directly in some of the existing Compulsory Discipline Courses (CDC)/ Discipline Courses Other than Compulsory (DCOC)/ Emerging Area (EA), etc. Courses	EEE F411 – Internet of Things CS F212 – Database Management System
4.	The project can be used in preparatory courses like Analysis and Application Oriented Courses (AAOC)/ Engineering Science (ES)/ Technical Art (TA) and Core Courses.	
5.	This project cannot come under any of the above-mentioned options as it relates to the professional work of the host organization.	

Signature of Student
Date:

Signature of Faculty
Date:

Table of Contents

INTRODUCTION

IoT Analytics is used for analysing huge amounts of data which are obtained by various Internet of Things devices which are interconnected among each other and performing a certain function. Such analysis requires passing queries and running tests to analyse the data. Also, certain operations can be done on the data to bear results which can be used to derive certain conclusions. In our project for Samsung, we are focusing on home automation and the use of IoT for analysis of data is the foundation of the entire project. We have used concepts of Database Management in the project for the analysis of the data. Softwares like MySQL and Amazon Redshift have been used thoroughly as the basic building blocks.

DATA WAREHOUSING

What is Data warehousing? For that let's start with the meaning of the phrase "single source of truth". In Information Systems Theory, the single source of truth is the practice of structuring all the best quality data in one place. A data warehouse exists to fill that need. A data warehouse exactly is the place where companies store their valuable data assets including customer data, sales data, employee data, and so on. In short, a data warehouse is the de-facto single source of data truth for an organization. It is usually created and used primarily for data reporting and analysis purposes.

There are unique characteristics of a data warehouse:

- **Subject-oriented:** They can analyse data about a particular subject or functional area (such as sales).
- **Integrated:** Data warehouses create consistency among different data types from disparate sources.
- **Non-volatile:** Once data is in a data warehouse, it's stable and doesn't change.
- **Time-variant:** Data warehouse analysis looks at change over time.

A data warehouse is widely used for data analytics. Often it is aggregated or segmented in some ways in order to facilitate analysis and reporting. All the analysis is done through SQL and Amazon Redshift.

STRUCTURED QUERY LANGUAGE (SQL)

SQL is a language which is used to interact with relational database management systems. And a relational database management system is basically just a software application which we can use to create and manage different databases. MySQL is one of the most popular database management systems.

What is a database? A database is any collection of related information, e.g., a phonebook, a shopping lists, a to-do list, Facebook's userbase. A Database Management System (DBMS) is a special software that helps users create and maintain a database. It makes it ways to manage large amounts of data, handles security, maintain backups, importing/exporting data, maintain concurrency, and it interacts with software applications. A good DBMS is able to execute the C.R.U.D functions on its database. C.R.U.D stands for Create, Read, Update, Delete.

There are two types of Databases: Relational Database (SQL) and Non-Relational Database(noSQL). Relational databases organize data into one or more tables. Each table has columns and rows and a unique key identifies each row. In non-relational databases, organized data is anything but a traditional table. A Relational Database Management System (RDBMS) helps users create and maintain a relational database. For example, MySQL, Oracle, PostgreSQL, MariaDB, etc. Structured Query Language (SQL) is a standardized language for interacting with RDBMS. It is used to perform C.R.U.D operations, as well as other administrative tasks (user management, security, backup, etc). It can be used to define tables and structures.

Database Queries:

Queries are requests made to the database management system for specific information. As the database's structure becomes more and more complex, it becomes more difficult to et the specific pieces of information we want. For example, a Google search is a query.

Tables:

Tables are database objects that contain all the data in a database. In tables, data is logically organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field in the record. For example, a table that contains employee data for a company might contain a row for each employee and columns representing employee information such as employee number, name, address, job title, and telephone number.

Data Types:

INT (size) – Whole numbers

VARCHAR (size) – String of length size

DECIMAL (m,n) – Decimal numbers – exact value

BLOB – Binary Large Objects, Stores large data

DATE – YYYY-MM-DD

TIMESTAMP – YYYY-MM-DD HH:MM:SS

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

DDL (Data Definition Language): DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

Examples of DDL commands:

- CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- DROP – is used to delete objects from the database.
- ALTER-is used to alter the structure of the database.
- TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed.
- COMMENT –is used to add comments to the data dictionary.
- RENAME –is used to rename an object existing in the database.

DQL (Data Query Language): DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

Example of DQL:

- SELECT – is used to retrieve data from the database.

DML (Data Manipulation Language): The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Examples of DML:

- INSERT – is used to insert data into a table.
- UPDATE – is used to update existing data within a table.
- DELETE – is used to delete records from a database table.

Complex SQL queries:

Complex SQL is the use of SQL queries which go beyond the standard SQL of using the SELECT and WHERE commands. Complex SQL often involves using complex joins and sub-queries, where queries are nested in WHERE clauses. Complex queries frequently involve heavy use of AND and OR clauses. These queries make it possible for perform more accurate searches of a database.

Examples:

TOP Nth MARKS USING DENSE_RANK():

```
WITH Result AS
(
        SELECT Studentname, Subject,
                Marks,  DENSE_RANK()
                OVER(ORDER BY Marks
                DESC) DenseRank
        FROM ExamResult
)
SELECT TOP 1 StudentName, Subject, Marks
FROM Result
WHERE DenseRank = N
```

TO FIND EMPLOYEES WHOSE SALARY IS HIGHER THAN THE AVERAGE SALARY OF THE EMPLOYEES IN THEIR DEPARTMENTS:

```
SELECT
   employee_id,
   first_name,
   last_name,
   salary,
   department_id
FROM
   employees e
WHERE
   salary > (SELECT
         AVG(salary)
      FROM
         employees
      WHERE
         department_id = e.department_id)
ORDER BY
   department_id ,
   first_name ,
   last_name;
```

AMAZON REDSHIFT

Before we proceed with the concepts of Amazon Redshift, we need to know what is Amazon Web Services (AWS). AWS or Amazon Web Services is a secure cloud service platform from Amazon. AWS services can be used to create and deploy any application in the cloud. It provides services over the internet. With AWS, we pay for only what we use. These cloud computing web services provide a variety of basic abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Redshift.

Before Amazon Redshift, developers used to fetch the data from data warehouse. There are certain cons of traditional data warehouse service:

- Time consuming
- Maintenance costs outweigh the benefits
- loss of information
- Data rigidity

These problems were solved with the help of Amazon Redshift.

Amazon redshift is a cloud-based service or a data warehouse service that is used for collecting and storing data. Also, it enables a user to analyse the data using BI tools and simplifies the process of handling large scale data sets. Amazon Redshift cost less to operate than any other cloud data warehouse service. Performance matters and Amazon Redshift is the fastest data warehouse available.

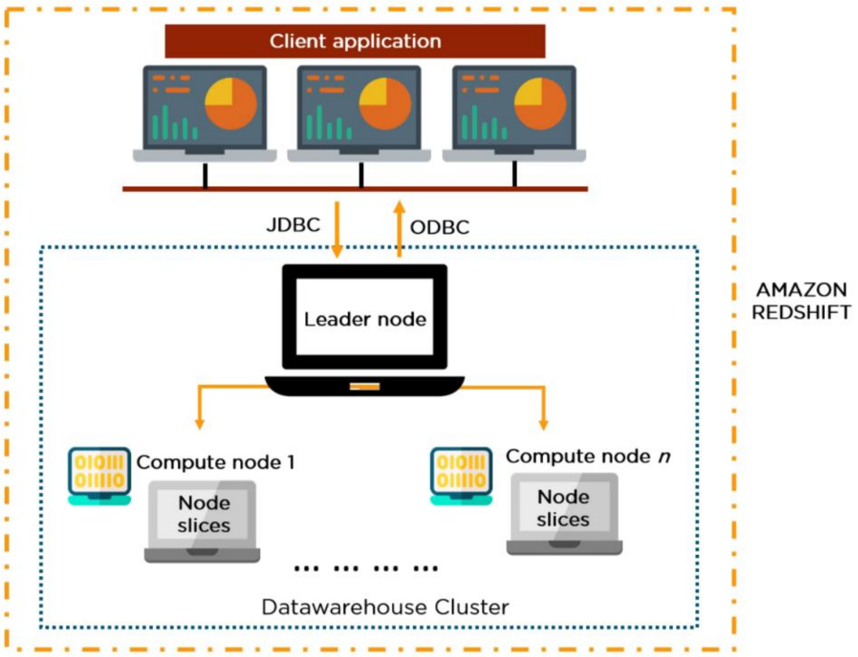
Advantages of Amazon Redshift:

- High performance - Fastest available data warehouse service.
- Low cost – Inexpensive service and we only pay for the services we use.
- Scalability – We can add or delete nodes any time we want without depending on the procurement of the hardware or the infrastructure.
- Availability – Highly available source since it is available across multiple availability zones,
- Security – Whenever we access redshift we create clusters in redshift, we can define a specific private cloud for our own clusters and also we can create our own security groups and attach it to our cluster.

- Flexibility – We can remove clusters, create under clusters, take snapshots and move it across different regions.
- Simple database migration – Very simple way to migrate from traditional data centre over the cloud to Redshift.

Architecture of Amazon Redshift:

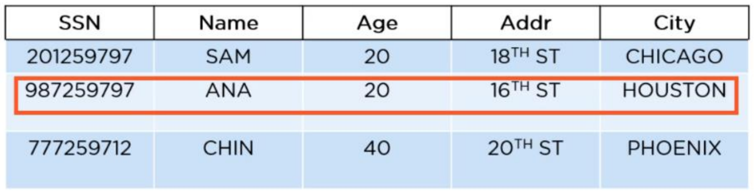
The client applications of Amazon redshift interact using two drivers. They are JDBC and ODB. Amazon Redshift services can monitor connections from other applications using JDBC connections. ODBC allows a user (directly from any application) to interact with live data of Amazon Redshift. Amazon Redshift has a set of computing resources called nodes.



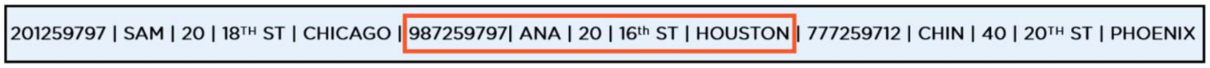
The nodes when gathered into a group are called a data warehouse cluster. Each cluster has one or more databases. The node manages the interaction between the client application and computer nodes. It analyses and develops designs in order to carry out database operations. The leader node runs the program and assigns the code to individual compute nodes. The compute node executes the program and shares the result back to the leader node for final aggregation. Compute nodes are categorized into slices. Each node slice is allotted with specific memory space, where it processes its workload. These node slices work in parallel in order to finish their work.

Column storage:

Column storage is an essential query performance and resulting in quicker outputs. Here is an example how database tables store record into disk blocks by row:



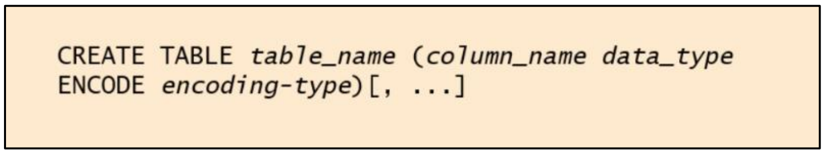
SSN	Name	Age	Addr	City
201259797	SAM	20	18^{TH} ST	CHICAGO
987259797	ANA	20	16^{TH} ST	HOUSTON
777259712	CHIN	40	20^{TH} ST	PHOENIX



201259797 | SAM | 20 | 18^{TH} ST | CHICAGO | 987259797| ANA | 20 | 16^{th} ST | HOUSTON | 777259712 | CHIN | 40 | 20^{TH} ST | PHOENIX

Compression:

Compression is a column-level operation which decreases storage requirements. It eventually improves query performance. The syntax for column compression is:



```
CREATE TABLE table_name (column_name data_type
ENCODE encoding-type)[, ...]
```

DATA PIPELINE

It is a series of tools and actions for organizing and transferring the data to different storage and analysis systems. It automates the ETL process, extraction, transformation, load. As a data pipeline example, we can collect information about our customers devices, location and session duration and track their purchases and interaction with our brand's customer service.

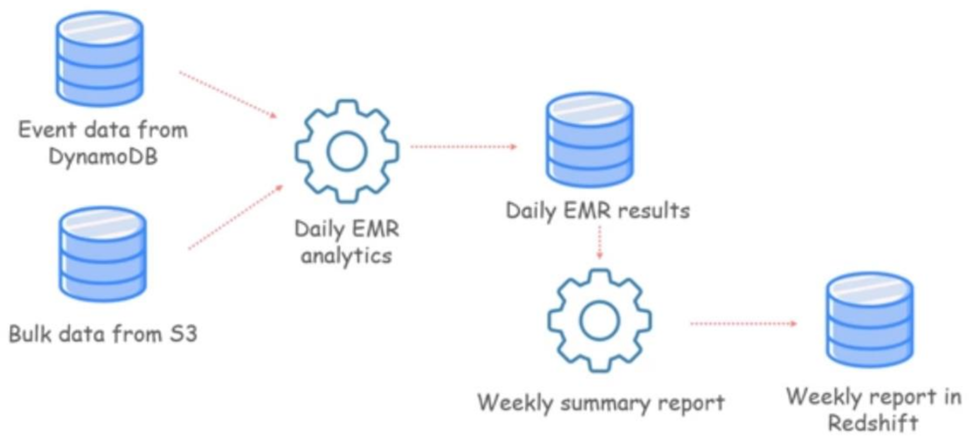
How does a data pipeline work?

The raw unstructured data is located at the beginning of the pipeline, then it passes a series of steps and each of them transforms the data.

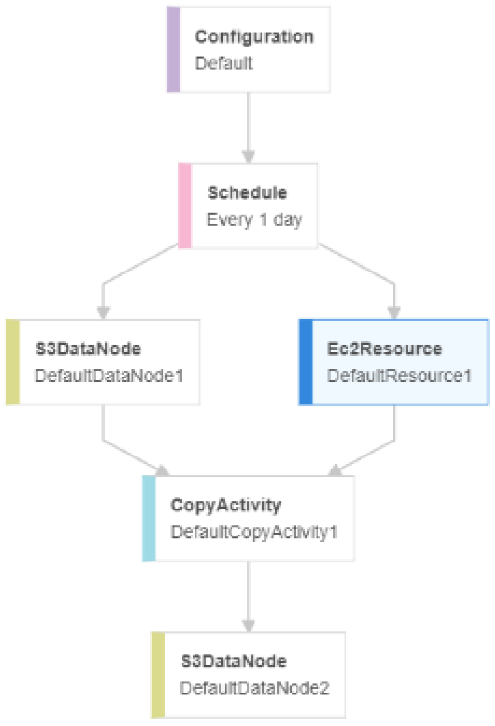
- Step one is collecting the data. At this stage the system gathers the data from thousands of sources such as databases, APIs, cloud sources and social media.
- Step two is extraction. After the raw data is collected, the system starts reading each piece of data using the data sources' API and after the data is extracted it goes through processing. If the sets of records are extracted and counted as one group, batch processing is applied. Real-time processing passes individual records as soon as they are created or recognized. By default, the companies use batch processing since it is easier and cheaper.
- Step 3 is transformation and standardization. Now we need to adjust the structure or format of the data. Among the most common types of transformation are basic transformations in which only the appearance and format of the data is affected without severe content changes, and advanced transformations in which the content and the relationship between data sets are changed.
- Step 4 is destination and this is the final point where the clean data is transferred further they can go to data warehouses while less structured data is stored in data lakes.
- Step five is monitoring to ensure that the data is accurate. Engineers continuously check the pipeline data by monitoring, logging and alerting the code.

AWS Data Pipeline

AWS data pipeline is a web service allowing data processing and moving it between different computing services AWS storage and local data sources. It helps to easily create complex data processing, pipeline operations and guarantee their fault tolerance and high availability.



This is how an AWS data pipeline works. For our project we have created a data pipeline which is going to copy data from an S3 bucket to another S3 bucket. The data pipeline is scheduled to run once everyday and copy the data from a source bucket to a destination bucket.



```
{
          "objects": [
            {
              "schedule": {
                "ref": "DefaultSchedule"
              },
              "filePath": "s3://interndestinationbucket/data",
              "name": "DefaultDataNode2",
              "id": "DataNodeId_u3HDM",
              "type": "S3DataNode"
            },
            {
              "output": {
                "ref": "DataNodeId_u3HDM"
              },
              "input": {
                "ref": "DataNodeId_601HK"
              },
              "schedule": {
                "ref": "DefaultSchedule"
              },
              "name": "DefaultCopyActivity1",
              "runsOn": {
                "ref": "ResourceId_VtFyg"
              },
              "id": "CopyActivityId_22zdi",
              "type": "CopyActivity"
            },
            {
              "schedule": {
                "ref": "DefaultSchedule"
              },
              "resourceRole": "DataPipelineDefaultResourceRole",
              "role": "DataPipelineDefaultRole",
              "name": "DefaultResource1",
              "id": "ResourceId_VtFyg",
              "type": "Ec2Resource"
            },
            {
              "failureAndRerunMode": "CASCADE",
              "schedule": {
                "ref": "DefaultSchedule"
              },
              "resourceRole": "DataPipelineDefaultResourceRole",
              "role": "DataPipelineDefaultRole",
              "scheduleType": "cron",
              "name": "Default",
              "id": "Default"
            },
            {
              "period": "1 days",
              "startDateTime": "2021-11-23T10:39:00",
              "name": "Every 1 day",
              "id": "DefaultSchedule",
              "type": "Schedule"
            },
            {
              "schedule": {
                "ref": "DefaultSchedule"
              },
              "filePath": "s3://internsourcebucket/data",
              "name": "DefaultDataNode1",
              "id": "DataNodeId_601HK",
              "type": "S3DataNode"
            }
          ],
          "parameters": []
        }
```

The above code is a JSON version of the data pipeline that we created. It is used to copy data from the bucket ‘s3://internsourcebucket’ to the bucket ‘s3://interndestinationbucket’ and it is scheduled to run once every day at 10:39 am UTC.

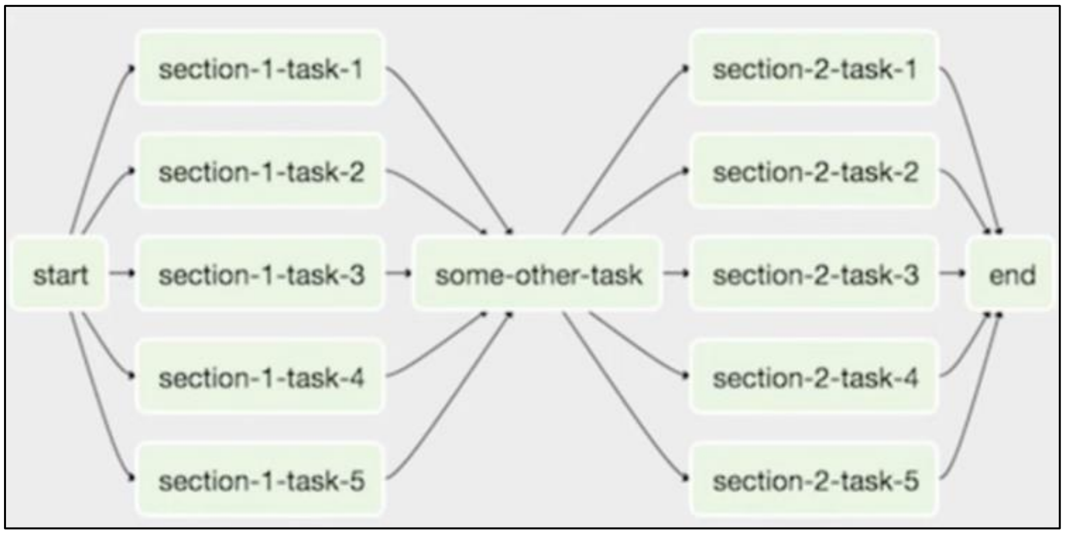
Airflow DAG

Airflow is a platform to programmatically author, schedule and monitor workflows or data pipelines. A workflow is a sequence of tasks started on a schedule or triggered by an event and frequently used to handle big data processing pipelines. A typical workflow works like this:

- Download data from source
- Send data somewhere else to process
- Monitor when the process is completed
- Get the result and generate the report
- Send the report out.

Apache Airflow is a workflow management system developed by Airbnb. It has a framework to define tasks and dependencies in Python. It is used for executing, scheduling, distributing tasks across worker nodes. We can view present and past runs, and there is a logging feature as well. It is extensible thorough plugins. It has a nice UI and it interacts well with the database.

A workflow is known as a Directed Acyclic Graph (DAG) with multiple tasks which can be executed independently. Airflow DAGs are composed of tasks.



Airflow has a great application in Data warehousing. It cleanses, organizes, does data quality checks, and publish/stream data into our growing warehouse. It is widely used in the study of analytics. Apache Airflow can be implemented in AWS using Amazon Managed Workflows for Apache Airflow (MWAA).

MWAA deploys Apache Airflow rapidly with the same open-source Airflow and the same Airflow UI. It implements the AWS Management console, AWS CLI, API, or AWS CloudFormation with Airflow. We can create an MWAA environment, copy our DAGs and plugins to Amazon S3 and access the Airflow. The MWAA environment is created under the appropriate IAM roles, VPC subnets and the S3 buckets for the DAG are to be allotted. The DAG code for copying data from an S3 bucket to another S3 bucket is:

```
from airflow import DAG

from airflow.operators.python_operator import PythonOperator
from airflow.operators import S3KeySensor
from airflow.hooks import S3Hook

from datetime import datetime
from io import StringIO

s3_bucket_name = 'airflow-dag-bucket'
s3_inkey = 'input.csv'
s3_outkey = 'output.csv'

def transform_file_fn():
  hook = S3Hook()
  s3c = hook.get_conn()
  obj = s3c.get_Object(Bucket=s3_bucket_name,Key=s3_inkey)
  infileStr = obj['Body'].read().decode('utf-8')
  outfileStr=infileStr.replace('"',' ')
  outfile = StringIO(outfileStr)
  s3c.put_object(Bucket=s3_bucket_name,Key=s3_outkey,Body=outfile.getvalue())

with DAG(dag_id='copy-s3-file',
         schedule_interval='@once',
         start_date=datetime(2021,11,29),
         catchup=False) as dag:

         wait_for_file = S3KeySensor(
           task_id='wait_for_file',
           bucket_key=s3_inkey,
           bucket_name=s3_bucket_name
         )

         transform_file = PythonOperator(
           task_id='transform_file',
           python_callable=transform_file_fn
         )

         wait_for_file >> transform_file
```

WORKING WITH THE DATATSET

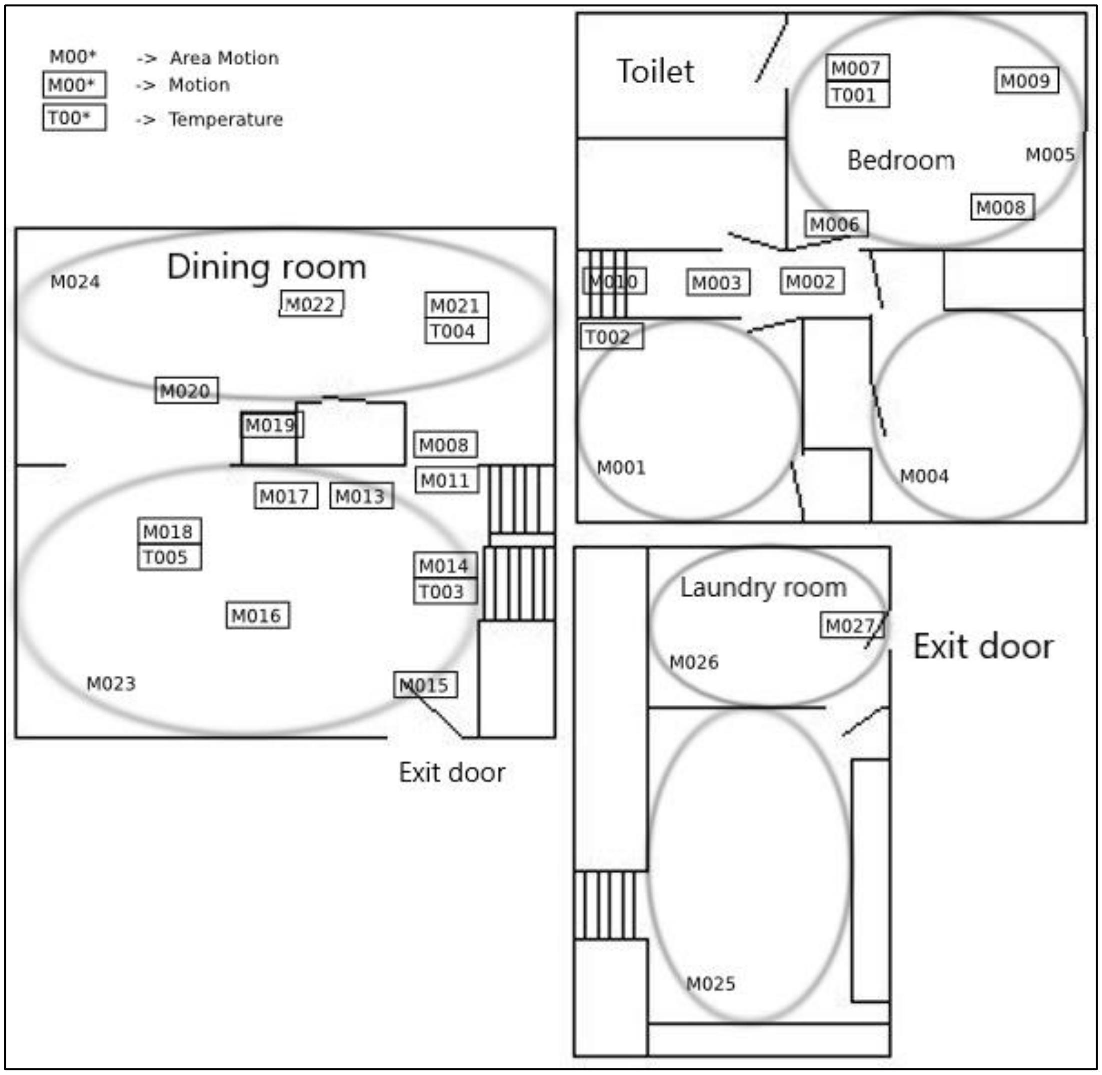
Now that we have created the data pipeline, the dataset needs to be uploaded to the S3 bucket for analysis. Before we proceed with the analysis, we need to observe the dataset and introduce some terms related to the dataset.

The dataset is a set of values conformed over time and stores data in a historical manner with the timestamps. The dataset we used contains sensor data that was collected in the home of a volunteer adult couple. The residents in the home were a man, a woman, and a dog. The couple's children also visited the home on at least one occasion.

The following activities are annotated within the dataset:

- Bed to toilet
- Breakfast
- R1 sleep
- R1 wake
- R1 work in office
- Dinner
- Laundry
- Leave home
- Lunch
- Night wandering
- R2 sleep
- R2 take medicine
- R2 wake

The sensor events are generated from motion sensors (these sensor IDs begin with "M") and temperature sensors (these sensor IDs begin with "T"). The layout of the sensor is as follows:



The certain section of the dataset is shown below to show how the data is recorded:

2009-06-10 06:14:59.029206	M005	ON
2009-06-10 06:14:59.087507	M007	ON
2009-06-10 06:15:06.053773	M007	OFF
2009-06-10 06:15:11.004895	M007	ON
2009-06-10 06:15:15.082658	M007	OFF
2009-06-10 06:15:16.019862	M005	OFF
2009-06-10 06:15:37.008132	T004	16
2009-06-10 06:15:37.035657	T003	16.5
2009-06-10 06:15:53.049202	T003	17
2009-06-10 06:16:08.030746	T003	16.5
2009-06-10 06:16:24.030179	T004	16.5
2009-06-10 06:16:24.005713	T003	17
2009-06-10 06:16:28.097232	M005	ON

The date and timestamp are show along with the sensor name and its status. We need to program certain codes to detect the activities from the sensor data.

KPI

A Key Performance Indicator is a measurable value that demonstrates how effectively a company is achieving key business objectives. Organizations use KPIs at multiple levels to evaluate their success at reaching targets. High-level KPIs may focus on the overall performance of the enterprise, while low-level KPIs may focus on processes in departments such as sales, marketing or a call centre. KPIs allow users, managers and companies to measure and manage targets and goals. We can use simple KPIs to track project performance and report to the project shareholders.

The KPIs for our dataset are:

- Bed to toilet
- Breakfast
- R1 sleep
- R1 wake
- R1 work in office
- Dinner
- Laundry
- Leave home
- Lunch
- Night wandering
- R2 sleep
- R2 take medicine
- R2 wake

We can use this KPIs to track and analyse the user's daily routine and schedule and can draw certain conclusions. Before we analyse the data, we need to set up a platform to carry out our analysis. We need to run SQL queries on the dataset to determine the results. To run SQL queries in AWS, we need to use AWS Athena/Spectrum as a platform. AWS Athena can be set up with the proper IAM roles, subnets and S3 locations.

To analyse the dataset, we need to select one KPI from the dataset. Let's take "Lunch" KPI as an example. With proper observation, we realise that there is movement in the dining room between 12pm to 1pm. The sensors M021, M024, M022 are in ON status during that time, this indicates that lunch is happening. So, our necessary requirement to detect the beginning of lunch is:

- Time should be between 12:00 and 13:00.
- M021, M024, M022 needs to be active and in 'ON' status.

With these requirements, we build our code.

The SQL code for beginning of lunch is:

```
with
cte
as
        (select cast(Time as date)[Date], *, ROW_NUMBER() OVER (PARTITION BY cast(Time as
        date) ORDER BY Time ASC) AS rn
        from [cairo].[dbo].[data - Copy]
        where cast(Time as time) between '12:00:00.000' and '13:00:00.000'
        and (sensor ='M021' or sensor = 'M024') and status = 'ON'
        )
        select Date,cast(Time as time)[Time], 'Lunch begins expected' as KPI_expected
        from cte
        where rn=1
```

The output is:

```
2009-06-10    12:23:01.0000000   Lunch begins expected
2009-06-11    12:06:16.0000000   Lunch begins expected
2009-06-13    12:14:11.0000000   Lunch begins expected
2009-06-14    12:00:56.0000000   Lunch begins expected
2009-06-15    12:00:04.0000000   Lunch begins expected
2009-06-18    12:00:29.0000000   Lunch begins expected
2009-06-19    12:00:11.0000000   Lunch begins expected
2009-06-20    12:00:04.0000000   Lunch begins expected
2009-06-21    12:00:01.0000000   Lunch begins expected
2009-06-22    12:00:02.0000000   Lunch begins expected
2009-06-23    12:01:43.0000000   Lunch begins expected
2009-06-24    12:00:06.0000000   Lunch begins expected
2009-06-25    12:12:13.0000000   Lunch begins expected
2009-06-26    12:01:42.0000000   Lunch begins expected
2009-06-28    12:12:58.0000000   Lunch begins expected
2009-06-29    12:00:20.0000000   Lunch begins expected
2009-06-30    12:05:04.0000000   Lunch begins expected
2009-07-01    12:00:11.0000000   Lunch begins expected
2009-07-03    12:20:20.0000000   Lunch begins expected
2009-07-04    12:00:17.0000000   Lunch begins expected
2009-07-05    12:01:10.0000000   Lunch begins expected
2009-07-08    12:00:58.0000000   Lunch begins expected
2009-07-09    12:04:20.0000000   Lunch begins expected
2009-07-11    12:00:11.0000000   Lunch begins expected
2009-07-12    12:35:10.0000000   Lunch begins expected
2009-07-13    12:48:20.0000000   Lunch begins expected
2009-07-14    12:33:43.0000000   Lunch begins expected
2009-07-16    12:09:14.0000000   Lunch begins expected
2009-07-17    12:00:44.0000000   Lunch begins expected
2009-07-18    12:56:19.0000000   Lunch begins expected
2009-07-20    12:02:30.0000000   Lunch begins expected
```

```
2009-07-22    12:00:13.0000000   Lunch begins expected
2009-07-23    12:18:17.0000000   Lunch begins expected
2009-07-24    12:00:03.0000000   Lunch begins expected
2009-07-25    12:05:48.0000000   Lunch begins expected
2009-07-26    12:00:50.0000000   Lunch begins expected
2009-07-27    12:02:45.0000000   Lunch begins expected
2009-07-28    12:00:31.0000000   Lunch begins expected
2009-07-29    12:00:26.0000000   Lunch begins expected
2009-07-31    12:06:34.0000000   Lunch begins expected
2009-08-01    12:00:16.0000000   Lunch begins expected
2009-08-02    12:00:04.0000000   Lunch begins expected
2009-08-03    12:00:14.0000000   Lunch begins expected
2009-08-04    12:00:00.0000000   Lunch begins expected
2009-08-05    12:00:02.0000000   Lunch begins expected
```

The output shows the beginning of “Lunch” activity every day. The list is sorted according to the date and the time.

Similarly, we could show the output of “Lunch ends”. The criteria then become different. We observe that between 1pm and 1:30pm, the sensors in the dining room turn off. Hence, lunch ends at this time interval. The conditions are:

- Time should be between 13:00 and 13:30.
- M021, M022, M024 should be in ‘OFF’ state.

The SQL code for end of lunch is:

```
with cte as
(select cast(Time as date)[Date], *, ROW_NUMBER() OVER (PARTITION BY cast(Time as date)
ORDER BY Time ASC) AS rn
from [cairo].[dbo].[data - Copy]
where cast(Time as time) between '13:00:00.000' and '13:30:00.000'
and (sensor ='M021' or sensor = 'M024' or sensor = 'M022') and status = 'OFF'
)
select Date,cast(Time as time)[Time], 'Lunch ends expected' as KPI_expected
from cte
where rn=1
```

The output is:

```
2009-06-10    13:00:21.0000000  Lunch ends expected
2009-06-11    13:03:51.0000000  Lunch ends expected
2009-06-13    13:11:23.0000000  Lunch ends expected
2009-06-14    13:19:07.0000000  Lunch ends expected
2009-06-15    13:04:32.0000000  Lunch ends expected
2009-06-18    13:18:56.0000000  Lunch ends expected
2009-06-19    13:10:59.0000000  Lunch ends expected
2009-06-22    13:00:02.0000000  Lunch ends expected
2009-06-24    13:00:00.0000000  Lunch ends expected
2009-06-25    13:00:19.0000000  Lunch ends expected
2009-06-26    13:04:22.0000000  Lunch ends expected
2009-06-27    13:02:58.0000000  Lunch ends expected
2009-06-30    13:02:40.0000000  Lunch ends expected
2009-07-03    13:03:23.0000000  Lunch ends expected
2009-07-05    13:13:09.0000000  Lunch ends expected
2009-07-08    13:20:01.0000000  Lunch ends expected
2009-07-09    13:06:39.0000000  Lunch ends expected
2009-07-11    13:00:09.0000000  Lunch ends expected
2009-07-12    13:00:29.0000000  Lunch ends expected
2009-07-13    13:01:55.0000000  Lunch ends expected
2009-07-14    13:00:01.0000000  Lunch ends expected
2009-07-17    13:00:51.0000000  Lunch ends expected
2009-07-18    13:00:05.0000000  Lunch ends expected
2009-07-20    13:00:34.0000000  Lunch ends expected
2009-07-22    13:00:41.0000000  Lunch ends expected
2009-07-23    13:01:23.0000000  Lunch ends expected
2009-07-26    13:23:20.0000000  Lunch ends expected
2009-07-29    13:22:35.0000000  Lunch ends expected
2009-07-31    13:20:05.0000000  Lunch ends expected
2009-08-01    13:02:33.0000000  Lunch ends expected
2009-08-02    13:00:35.0000000  Lunch ends expected
2009-08-03    13:04:55.0000000  Lunch ends expected
2009-08-04    13:00:03.0000000  Lunch ends expected
2009-08-05    13:00:02.0000000  Lunch ends expected
```

Using these two tables we can calculate, the duration of lunch, the change in timings on daily basis. Using the data, we can study the daily schedules and life of the people and draw conclusions. But those are for future motives. For the time being, we are concentrating on the derivation of data from the dataset.

Similarly, we can detect the other KPIs using the various conditions that we observe.

- In case of breakfast, we can apply the same conditions of lunch, only the time criteria need to be changed to early morning timings.
- Similarly for dinner, we need to change the time criteria to night timings.
- For ‘Bed to Toilet’ movements, we need to check the sequential activity of the sensors around that area.
- For night wandering we need to check for the motion sensors throughout the house at late night timings (post dinner, so that it doesn’t overlap).
- For sleep, we check if the bedroom sensors at night are in ‘OFF’ state.
- For wake, we check if the bedroom sensors at early morning are in ‘ON’ state.
- For exit, we check if the sensors near the exit are in ‘ON’ or not.
- We can check the motions of R1 and R2 using the reading of temperature sensors.

We can put such conditions and check for the other KPIs and could say that such an event has occurred.

PRACTICAL APPLICATION

Now that we discussed all the topics required for the analysis of a dataset, we need to know how this project id applicable in practical situations. What is the real-life motive of this project? To answer this question, we need to consider the real-life scenario of the conditions in which we recorded the data in the dataset. We imagine a house with various rooms, with a simple system of two types of sensors, that is motion and temperature sensors, located at various corners of each room. The dataset that we observe in the project is a combination of data recorded over a span of 3 months. In real life scenario, the data would be recorded everyday by these sensors and stored in an S3 bucket. The movements of the individuals and their temperatures are stored on a daily basis. Everyday new data is recorded and a new dataset is stored in the bucket. This forms a data lake.

The S3 bucket which acts as the data lake is connected to a destination S3 bucket through a data pipeline which is scheduled to run on a daily basis at a fixed point of time. The data pipeline organises the raw data and stores it in a format which is ready for processing. The structured dataset is stored in the destination S3 bucket and is updated daily as the pipeline runs. The data is processed and is ready for testing.

The final dataset is analysed using various SQL codes and queries and the KPIs are detected. Based on the results, the observations and conclusions can be drawn. This system demonstrated is a simple one which consists of two sensors and simple KPIs. Using the KPIs, we can derive further more KPIs and hence, a deeper analysis. Future improvements involve using of more sensors, to fetch more data. As the system grows complex, data will also be available in abundance and hence, more complex analysis can be done. This is how we can do data warehousing and analysis on inter-connected data.

Home automation is a very promising field in today's generation. The entire core of the project is home automation. With products such as Amazon Alexa, Google Home, inter-connected smart watches, smart phones, smart bulbs, and even smart electronic appliances, technology is in the grip of our hand. Data analytics is a very crucial topic is such field of electronics. With huge amounts of data in our hand, we can formulate programs to calculate and analyse the data to give us a better understanding of everything happening with us and around us. For example, smart watches in our hands are capable of measuring our pulse, oxygen saturation, blood pressure, footsteps, and at the same time it can be connected to our phones to get better accessibility of the inter-connected devices. With the biometric data collected by the smart watch, certain programs formulated can track the data and also inform us of any anomalies in the body. By doing simple calculations, the data can be processed to give us information on our body. This is a huge step in technology and it is becoming widespread.

PROJECT FILES

The entire project is stored in a GitHub repository:

https://github.com/sbasu033/Samsung-project

The sample dataset used for demonstration is download from CASAS website.

http://casas.wsu.edu/datasets/cairo.zip

The entire project is ran using Amazon Web Services features.

CONCLUSION

The different databases management software languages and softwares discussed in the project form the building blocks of our project. IoT Analytics, as the name suggests, deals with the analysis of data obtained from interconnected IoT devices. The data obtained are stored in data warehouses in certain databases. Hence, the advanced concepts of DBMS come in handy when dealing with data analysis. Furthermore, the data can be processed and analysed using AWS. The analysed data can give us information about the various conclusions drawn from the results of the dataset. The model demonstrated is a minute version of home automation. Two simple sensors are used to record the movements and with that only, we are able to construct so many KPIs an derive so many results.

REFERENCES

- https://www.w3schools.com/sql/
- https://docs.aws.amazon.com/redshift/latest/dg/c_designing-tables-best-practices.html
- https://en.wikipedia.org/wiki/MySQL
- https://github.com/sbasu033/Samsung-project
- http://casas.wsu.edu/datasets/cairo.zip
- https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/what-is-datapipeline.html
- https://docs.aws.amazon.com/mwaa/latest/userguide/what-is-mwaa.html