

BI System Specification Document

Version 6

Daniel Strauss

**Content**

1. General
   1. Project Objective
   2. Project Contents
2. Distribution of tasks and schedules for execution (Gantt)
3. Technical Specification
   1. Prerequisites - A comprehensive list of systems including access methods
   2. Solution Architecture and Flow Chart (HLD)
4. Functional Specification:
   1. Creation of final Source to Target and ERD models.
   2. Detailed description of all ETL processes.
   3. Description of data tables in the Data Warehouse (DWH), including history tables.
   4. Specification of measures, tables, and filters for reports in Power BI

**1. General**

**1.1. Project Objective**

The project aims to establish a comprehensive Business Intelligence (BI) solution for Amazon, utilizing the PriorityERP Database. It involves condensing key data tables, with a focus on sales, customer details, employee records, etc. The goal is to provide Amazon's leadership with visually represented data through dashboards and reports, facilitating efficient decision-making. The analysis of customer data aims to offer insights into preferences, purchasing behaviours, and product performance, helping Amazon shape targeted marketing campaigns and optimize production strategies.

**1.2. Project Contents**

In this project, we aim to construct a comprehensive Data Mart encompassing various facets of real-world business order details from an end-to-end perspective.

Key summary tables designed to meet the company's requirements:

**Fact\_DWH**- Encompasses details about all orders, including transaction dates, pricing, and quantities per customer from the transactions in the transactional database.

**Dim\_Employees** - Comprises information about the company's workforce.

**Dim\_Customers** - Encompasses information about the company's clientele.

**Dim\_Products** - Includes details about the products offered by the company.

**Dim\_ProductsHistory** - Captures information about products and the changes that have occurred to them over time.

**Dim\_Stores** - Provides insights into the stores involved in the transactions.Top of Form

The project is loaded automatically every morning

**Sales Employee Department Analysis:**

The Sales Employee report will focus on analysing data related to sales activities, including order timestamps, customer details, and the initiating agent. The department's primary objective is to identify trends in sales, considering factors such as product category and store type. A comparative analysis with the previous year's results will be conducted to provide a comprehensive view of goal achievement. The department consistently gathers information with each new order entry into the system, aiding users in monitoring ongoing transactions.

Employee performance will be evaluated based on the top 5 most beneficial agents, considering factors like total sales, year-over-year sales variation, and the popularity of the brands they promote. Additionally, the top 3 best-selling brands will be highlighted, showcasing total sales categorized by stores and locations. To derive future business insights and implement strategic actions for growth, a detailed analysis of sales data by store type will be performed.

**Customer Department Analysis:**

Customers will have access to a variety of data related to their purchases and interactions with the brand. This information includes total sales, sales categorized by category, top-performing brands, Leveraging this information enables customers to make well-informed choices for future purchases and deepen their understanding of the diverse array of **products offered by the company.**

**Executive Dashboard Analysis:**

The executive department will focus on overall data related to the highest scope of the organization, incorporating merged data between products, employees, and sales. Measurable information will include the number of products sold, the most profitable products, a comparison of income revenues between store types, and the most profitable countries by sales. A thorough comparison of the company's revenue with the previous year will be conducted, providing executives with insights to guide strategic decisions.Top of Form

**2. Distribution of tasks and schedules for execution (Gantt)**

Gantt Document[**attached**](https://docs.google.com/spreadsheets/d/1hz9cJMrAYFiWEHtLHq2NX3AoRBet49Bk/edit?usp=sharing&ouid=105715906283239387124&rtpof=true&sd=true)

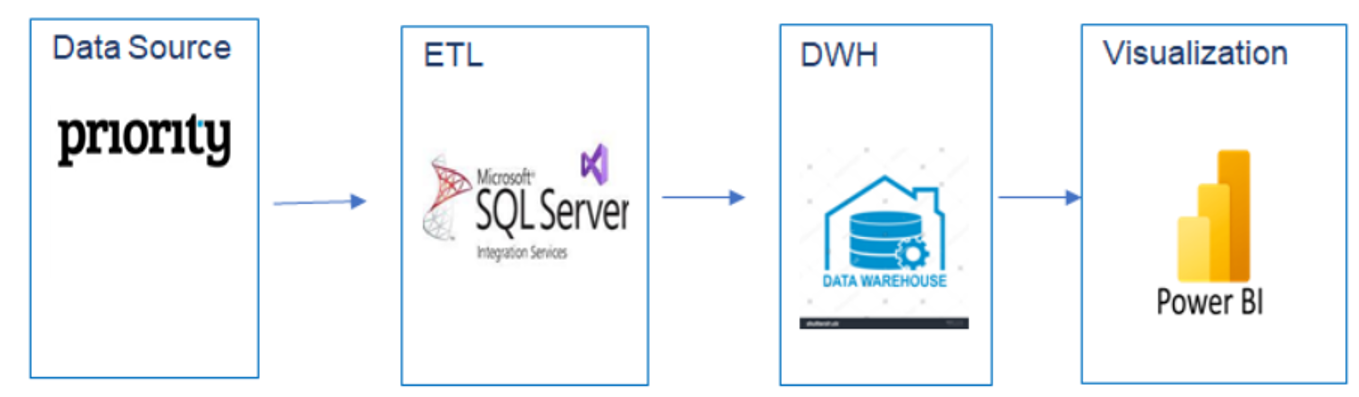
**3. Technical Specification**

**3.1. Prerequisites**

|  |  |
| --- | --- |
| System/Process | Explanation |
| SQL Server | Operational DB-tables-data (SQL Files) |
| SSIS | ETL processes using SSIS in Visual Studio |
| Data Refreshing | Refreshing processes through an attribute of Employees in SSMS |
| Power BI | Creating reports & dashboards for Power BI usage |

**3.2. Solution Architecture and Flow Chart (HLD)**

HLD:



**Data Collection and Exploration:** Data collection and exploration from the ERP system will be conducted using SQL Server. The extracted data will then undergo the Extract, Transform, Load (ETL) process through SQL Server Integration Services (SSIS) for organization and arrangement into a Data Mart.

**Customer Department Report:** The report for the Customer department will encompass the following key insights:

1. **Sales, Quantity ,Orders ,Customers, Avg Customers sales:** General measure on project data
2. **Sells by Category**
3. **Average Customers sales**
4. **Sales by Product**

**Management Department Dashboard:** The report for the Management department will include the following key metrics:

1. **Sales, Quantity ,Orders ,Employees ,Customers:** General measure on project data
2. **Sales by Country**
3. **Monthly Sales**
4. **Sales by store**

**Employees Department Report:** The report for the Employees department will focus on the performance of individual agents and overall sales:

1. **Sales, Quantity ,Orders , Employees ,Avg Employees sales:**
2. **top 5 agents based on total sales.**
3. **Sales by Country**
4. **Month Sales change by Month**

**4. Functional Specification**

**4.1. Creation of final Source to Target and ERD models.**

**4.1.1 Source to Target**

Source To Target document [**attached**](https://docs.google.com/spreadsheets/d/1g-hGzTpweKSAMjNubKuQbwoYput7--O_/edit?usp=sharing&ouid=105715906283239387124&rtpof=true&sd=true)

The data tables that will be used from the OLTP database consist of a total of 11 tables as seen in the S2T

**4.1.2 ERD model**

Erd document [**attached**](https://drive.google.com/file/d/1_Wf8WkK5bzIRp0wcjqXERMn-mbpGZfQI/view?usp=sharing)

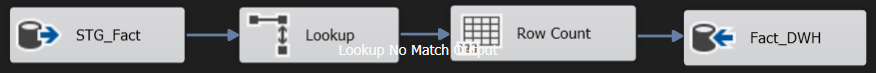
**4.2 Detailed description of the ETL Process**

In SSIS, We have packages for the required tables.

1. Fact\_DWH Table:

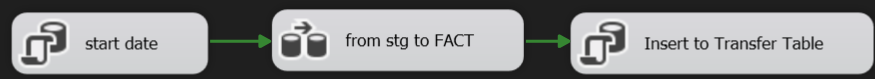
**FACT\_Sales:**

The FACT\_Sales data is sourced from the Stg\_Fact\_DWH table. During the loading process, and an additional variable is introduced to track the row count. Subsequently, the transformed data is moved to the Fact\_DWH table.



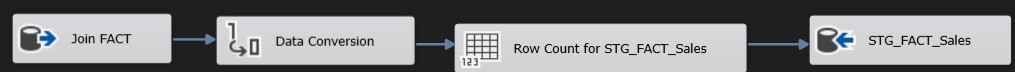
After that the records of: Package name, table name, StartDate, End Date, the amount of

transferred rows are inserted inside the table Transfer table



**STG\_FACT\_Sales**

Creating a table stg\_fact\_sales that includes a united results of the SQL query that joins between the different tables into our desired result. During the loading process, certain columns undergo data type conversion to align with the loaded data After that we count the amount of the transferred rows. After that the data will be loaded inside stg\_fact\_sales



Every time the package runs, all the previous records from stg\_fact\_sales will be deleted, ONLY the new records will be inserted inside stg\_fact\_sales from the mrr tables. After that the records of: Package name, table name, StartDate, End Date, the amount of transferred rows are inserted inside the table 'ransfer table

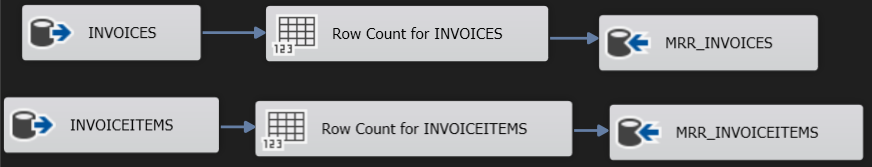


**MRR FOR FACT**

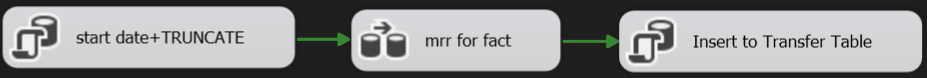
Transferring the Tables 'INVOICES', 'INVOICEITEMS' into mrr tables.

The load action will load ONLY new records that did not appear in the 'FactOrders' table before

to avoid a case that we load duplicated data. All the new records will be counted.



Upon each execution of the package, the existing records in the mrr\_INVOICES and mrr\_INVOICEITEMS tables are cleared, making room for only the latest data to be inserted. Following this, the details including the package name, table names, start date, end date, and the count of transferred rows are appended to the 'Transfertable' for reference.



**Dim\_Employees**

The process involves loading data from the **Stg\_Employees** table using the Upsert Merge Command. This command is designed to manage data in various scenarios:

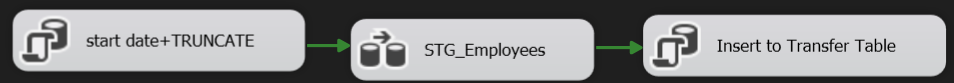
1. **Load New Data:** Newly added data is seamlessly incorporated into the system.
2. **Update Existing Data:** Changes in the Online Transaction Processing Tables (**EMPLOYEES**) are identified and applied to ensure the most up-to-date information.
3. **Deleted Data Handling:** The process also addresses instances where data has been deleted from the Online Transaction Processing, ensuring accurate reflection in the system.

Top of Form



**Stg\_Employees:**

Every time the package runs, all the previous records from stg\_Employees will be deleted, ONLY the new records will be inserted inside 'stg\_Employees' from the mrr tables. After that the records of: Package name, table name, StartDate, End Date, the amount of transferred rows are inserted inside the table 'Transfer table'



Creating a table 'stg\_Employees' that includes a united results of the SQL query that joins between the different tables into our desired result. Next, we convert column datatypes to match those of the destination and lastly: count the amount of the transferred rows. After that the data will be loaded inside 'stg\_Employees'



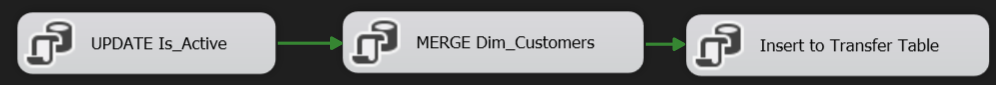
**Dim\_Customers:**

The process involves extracting and loading data from the **Stg\_Customers**. This data is managed using the Upsert Merge Command, accommodating various scenarios:

1. **Load New Data:** Newly acquired data is integrated seamlessly into the system.
2. **Update Existing Data:** Changes in the Online Transaction Processing Tables (**CUSTOMERS**) are identified and applied to maintain updated records.
3. **Deleted Data Handling:** The process ensures that data deleted from the Online Transaction Processing is appropriately managed.

Following the successful loading of the 'dim\_Customers' table, relevant details including the package name, table name, start date, end date, and the count of transferred rows are logged in the 'Transfer table' for reference.

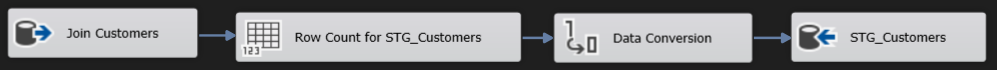
Top of Form



**Stg\_Customers:**

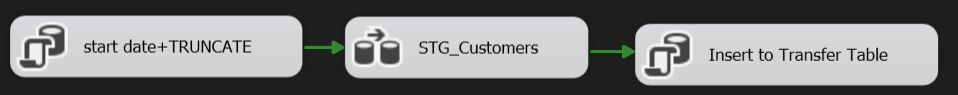
A table named 'stg\_Customers' is created to consolidate the results of an SQL query involving joins between different tables to achieve the desired outcome. Following this, column datatypes are adjusted to align with the destination requirements. Subsequently, the process involves counting the number of rows to be transferred. Finally, the processed data is loaded into the 'stg\_Customers' table.

Top of Form



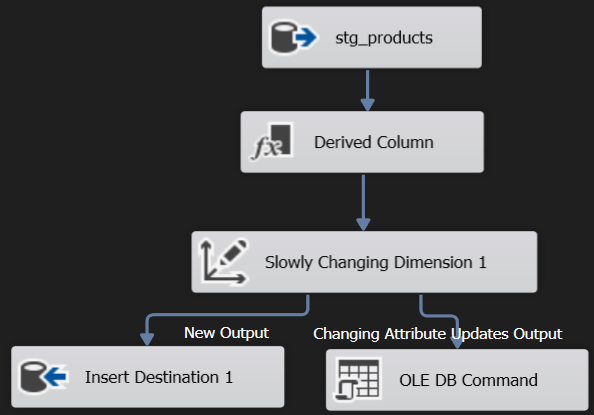
Upon each execution of the package, the existing records in the 'stg\_Customers' table are cleared, making room for only the new records from the 'mrr' tables to be inserted. Following this, pertinent details including the package name, table name, start date, end date, and the count of transferred rows are appended to the 'Transfer table' for reference.

Top of Form

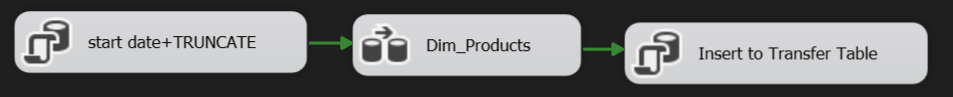


**Dim\_Products:**

The data loading process involves extracting information from 'Stg\_Products' using Slowly Changing Dimensions (SCD). Only records with changes, including new entries, updates, or deletions, will be loaded into the 'Dim\_Products' table. Following this, the process generates an updated version of the 'Dim\_Products'.



After Loading the 'dim\_Products' table, the records of: Package name, table name, Date, the amount of transferred rows are inserted inside the table 'Transfer table'



The data loading process involves extracting information from 'DIM\_Products' using Slowly Changing Dimensions (SCD). Only records with changes, including new entries, updates, or deletions, will be loaded into the 'Dim\_Products\_history' table. Following this, the process generates an updated version of the 'Dim\_Products\_history'.

**A diagram of a product

Description automatically generated**

**Stg\_Products:**

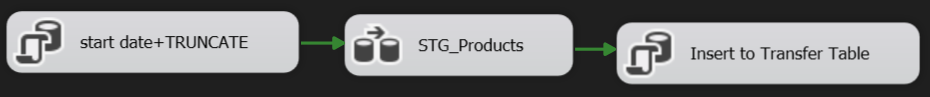
A table named 'stg\_Products' is created, consolidating the results of an SQL query that incorporates joins between various tables to achieve the desired output. Subsequently, column datatypes are adjusted to align with the destination requirements. Following this, the process involves counting the number of rows to be transferred. Finally, the processed data is loaded into the 'stg\_Products' table.

Top of Form



Upon each execution of the package, the existing records in the 'stg\_Products' table are cleared. Subsequently, only the new records from the 'mrr' tables are inserted into 'stg\_Products'. Following this data transfer, relevant details, including the package name, table name, start date, end date, and the count of transferred rows, are appended to the 'Transfer table' for reference.

Top of Form



**Dim\_Stores:**

The process involves loading data from **STG\_Stores** using the Upsert Merge Command, which manages data in various scenarios:

1. **Load New Data:** Incorporating newly acquired data seamlessly into the system.
2. **Update Existing Data:** Identifying and applying changes in the Online Transaction Processing Tables (**BRANCHES**) to maintain updated records.
3. **Deleted Data Handling:** Ensuring proper management of data that has been deleted from the Online Transaction Processing.

Following the successful loading of the 'dim\_Stores' table, relevant details, including the package name, table name, start date, end date, and the count of transferred rows, are recorded in the 'Transfer table' for future reference.

Top of Form



**Stg\_Stores:**

A table named 'stg\_Stores' is created, consolidating the results of an SQL query that involves joining different tables to achieve the desired output. Subsequently, column datatypes are adjusted to align with the destination requirements. Following this, the process involves counting the number of rows to be transferred. Finally, the processed data is loaded into the 'stg\_Stores' table.

Top of Form



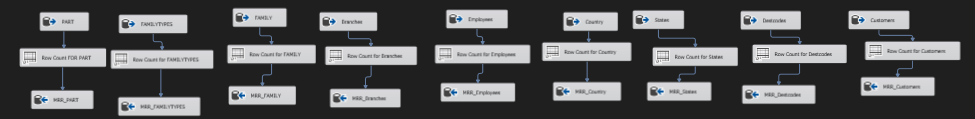
After Loading the 'stg\_Stores' table, the records of: Package name, table name,

StartDate,EndDate, the amount of transferred rows are inserted inside the table 'Transfer table'



**mrr:**

Loading the source tables from the Online Transaction Processing tables that related to the stg’s into new mrr tables in our database, after loading each mrr table, the amount of the transferred rows will be counted to check how many rows transferred and recognize what happened during the process.



**4.1.**

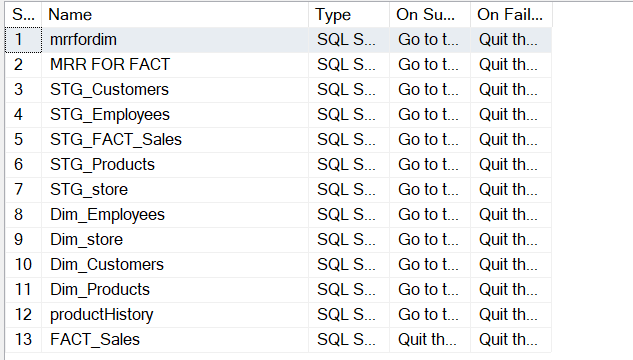
**Jobs Overview**

SQL Agent jobs consist of one or more steps, each representing a specific action or set of actions. These jobs can include tasks such as data extraction, transformation, loading, backups, and other database maintenance activities. The scheduling capabilities of SQL Agent enable users to plan and automate these tasks, reducing manual intervention and ensuring timely and consistent execution.

These SQL Agent jobs collectively form a robust and automated ETL process, enabling the Sales Department to harness the power of data for informed decision-making and strategic insights.Top of Form

**4.2**

**Job definition**



**Jobs Scheduling**

The scheduling capabilities of SQL Agent enable users to plan and automate these tasks, reducing manual intervention and ensuring timely and consistent execution.



**Development Environment**

Upon verifying the successful operation of the end-to-end ETL process in the production environment, including accurate and planned data transfers into dimension and fact tables, I set up a dedicated development environment. In this environment, I replicated all tables along with their production data. This strategic move enables us to conduct further development work in the new environment without causing disruptions to the existing, deployed, and operational production environment. This task was accomplished using specific functionalities within SQL Server Integration Services (SSIS).

A close up of a sign

Description automatically generated  

**4.3. Description of data tables in the Data Warehouse (DWH):**

**Tables in the DWH:**

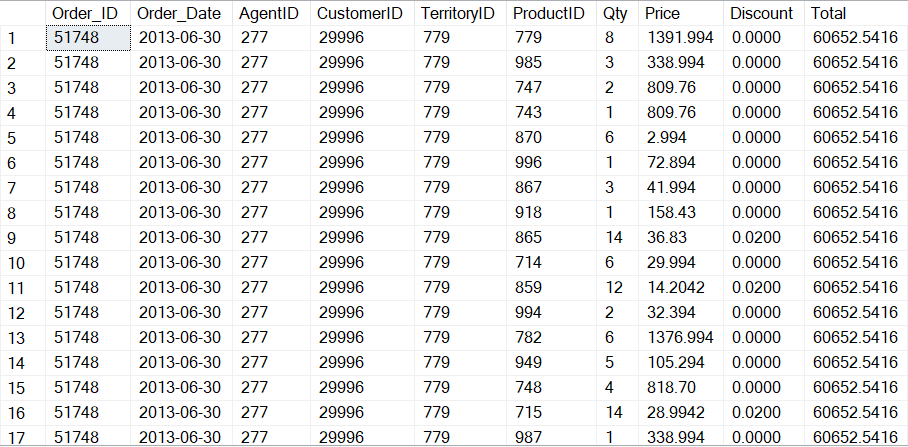
**Fact\_DWH**

1. **Order\_ID:**
   * A unique identifier for each sales order.

**Order\_Date:**

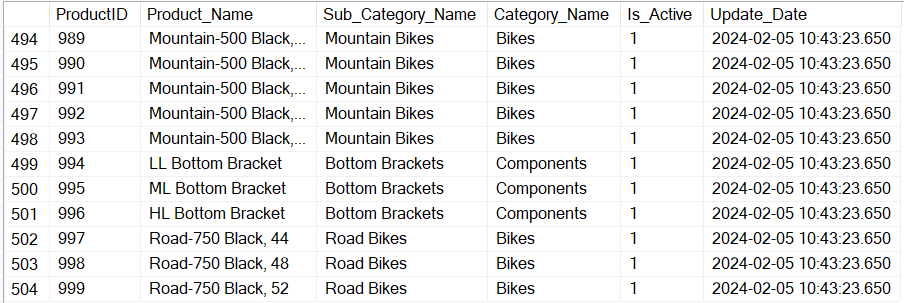
* + The date on which the sales order was placed.

1. **AgentID:**
   * Identifier for the sales agent associated with the transaction.
   * Constraint: Foreign Key (FK\_Fact\_DWH\_Agent) references the "Emp\_ID" column in the "Dim\_Employees" table.
2. **CustomerID:**
   * Identifier for the customer who placed the sales order.
   * Constraint: Foreign Key (FK\_Fact\_DWH\_Customer) references the "CustomerID" column in the "Dim\_Customers" table.
3. **TerritoryID:**
   * Identifier for the sales territory associated with the transaction.
4. **ProductID:**
   * Identifier for the product sold in the transaction.
   * Constraint: Foreign Key (FK\_Fact\_DWH\_Product) references the "ProductID" column in the "Dim\_Products" table.
5. **Qty:**
   * Quantity of the product sold in the transaction.
6. **Price:**
   * The unit price of the product at the time of the transaction.
7. **Discount:**
   * The discount applied to the product in the transaction.
8. **Total:**
   * The total amount for the sales transaction, taking into account quantity, price, and discounts.



**Dim\_Products**

1. **ProductID:**
   * A unique identifier for each product.
   * Constraint: PRIMARY KEY, indicating that the "ProductID" column is the primary key for this table.
2. **Product\_Name:**
   * The name or description of the product.
3. **Sub\_Category\_Name:**
   * The name of the sub-category to which the product belongs.
4. **Category\_Name:**
   * The name of the category to which the product belongs.
5. **Is\_Active:**
   * Indicates whether the product is active or not. It is a flag (1 for active, 0 for inactive).
6. **Update\_Date:**
   * The date and time when the record was last updated.



**Dim\_Products\_History**

1. **ProductID:**
   * A unique identifier for each product.
   * Constraint: PRIMARY KEY, indicating that the "ProductID" column is the primary key for this table.
2. **Product\_Name:**
   * The name or description of the product.
3. **Sub\_Category\_Name:**
   * The name of the sub-category to which the product belongs.
4. **Category\_Name:**
   * The name of the category to which the product belongs.
5. **Insert\_Date:**
   * The date and time when the historical record for the product was inserted.
6. **End\_Date:**
   * The date and time when the historical record for the product ends. It signifies the point in time when the information in the record is no longer active.

A screenshot of a computer

Description automatically generated

**Dim\_Customers**

1. **CustomerID:**
   * A unique identifier for each customer.
   * Constraint: PRIMARY KEY, indicating that the "CustomerID" column is the primary key for this table.
2. **StoreID:**
   * Identifier for the store associated with the customer.
   * Constraint: FOREIGN KEY references the "store\_ID" column in the "Dim\_Store" table
3. **Name:**
   * The name of the customer.
4. **Address:**
   * The address of the customer.
5. **City:**
   * The city where the customer is located.
6. **Region:**
   * The region or state where the customer is located.
7. **Country:**
   * The country where the customer is located.
8. **Is\_Active:**
   * Indicates whether the product is active or not. It is a flag (1 for active, 0 for inactive).
9. **Update\_Date:**
   * The date and time when the record was last updated.



**Dim\_Employees**

1. **Emp\_ID:**
   * A unique identifier for each employee.
   * Constraint: PRIMARY KEY, indicating that the "Emp\_ID" column is the primary key for this table.
2. **First\_Name:**
   * The first name of the employee.
3. **Last\_Name:**
   * The last name of the employee.
4. **Job\_Title:**
   * The job title or position of the employee.
5. **Hire\_Date:**
   * The date when the employee was hired.
6. **Phone\_Number:**
   * The phone number of the employee.
7. **Email\_Address:**
   * The email address of the employee.
8. **Territory\_Name:**
   * The name of the territory associated with the employee.
9. **Is\_Engineer:**
   * A flag indicating whether the employee is an engineer. It is a character field that presumably holds 'Y' for Yes or 'N' for No.
10. **Is\_Active:**
    * Indicates whether the product is active or not. It is a flag (1 for active, 0 for inactive).



**Dim\_Stores**

1. **store\_ID:**
   * A unique identifier for each store.
   * Constraint: PRIMARY KEY, indicating that the "store\_ID" column is the primary key for this table.
2. **name:**
   * The name of the store.
3. **location:**
   * The location or address of the store.

A screenshot of a computer

Description automatically generated

**4.4. Power BI**

In Power BI, I have created several measures:

1 total sales = sum(Fact\_DWH[LineTotal])

2 Total Quantity = SUM(Fact\_DWH[Qty])

3 Total Orders = DISTINCTCOUNT(Fact\_DWH[Order\_ID])

4 Total Employees = DISTINCTCOUNT(Dim\_Employees[Emp\_ID])

5 Total Customers = DISTINCTCOUNT(Dim\_Customers[CustomerID])

6 Products Sold = DISTINCTCOUNT(Dim\_Products[ProductID])

7 MTD Sales = CALCULATE(SUM(Fact\_DWH[LineTotal]),DATESMTD(Dim\_Date[Date]))

8 Average Employees sales = SUM(Fact\_DWH[LineTotal]) / [Total Employees]

9 Average Customers sales = SUM(Fact\_DWH[LineTotal]) / [Total Customers]

10 % Month Sales Chage = VAR MTD\_Sales = [MTD Sales] var Pre\_MTD\_Sales=[Sales SamePeriodLastMounth] RETURN IF(ISBLANK(MTD\_Sales) || ISBLANK(Pre\_MTD\_Sales),BLANK(),DIVIDE(MTD\_Sales,Pre\_MTD\_Sales,1)-1)

11 MTD Sales = CALCULATE(SUM(Fact\_DWH[LineTotal]),DATESMTD(Dim\_Date[Date]))

12 YTD Units = TOTALYTD([Total Quantity], Dim\_Date[Date])

13 YTD Sales = TOTALYTD([Total Sales], Dim\_Date[Date])

14 LY Sales = CALCULATE([Total Sales], SAMEPERIODLASTYEAR(Dim\_Date[Date]))

15 Sales Difference = [YTD Sales] - [LY Sales]

16 Sales Difference % = DIVIDE([YTD Sales] - [LY Sales], [LY Sales],0)

17 Sales SamePeriodLastMounth = Var MTDStartMounth = MINX(FILTER(ALL(Dim\_Date),[MTD Sales]<>BLANK()),MONTH(Dim\_Date[Date])) RETURN IF(AND(NOT ISBLANK([MTD Sales]),ISBLANK(CALCULATE([MTD Sales],DATEADD(Dim\_Date[Date],-1,MONTH)))),0,CALCULATE([MTD Sales],DATEADD(Dim\_Date[Date],-1,MONTH)))

**Dashboard:**

A screenshot of a computer

Description automatically generated

On the Dashboard, you can view the Total Sales, Quantity ,Orders ,Employees ,Customers ,And each of them has useful indicators and also Sales by Country, Monthly Sales , Sales by store you can slice the data for each of the top-selling countries. For example, for Canada or 2014

**Customers report:**

**A screenshot of a computer

Description automatically generated**

On the Customers report, you can view the Sales, Quantity ,Orders ,Customers,Avg Customers sales ,And each of them has useful indicators and also Sells by Category ,Average Customers sales ,Sales by Product. you can slice the data for each of the top-selling countries. For example, for Canada or 2014

**Employees Report:**

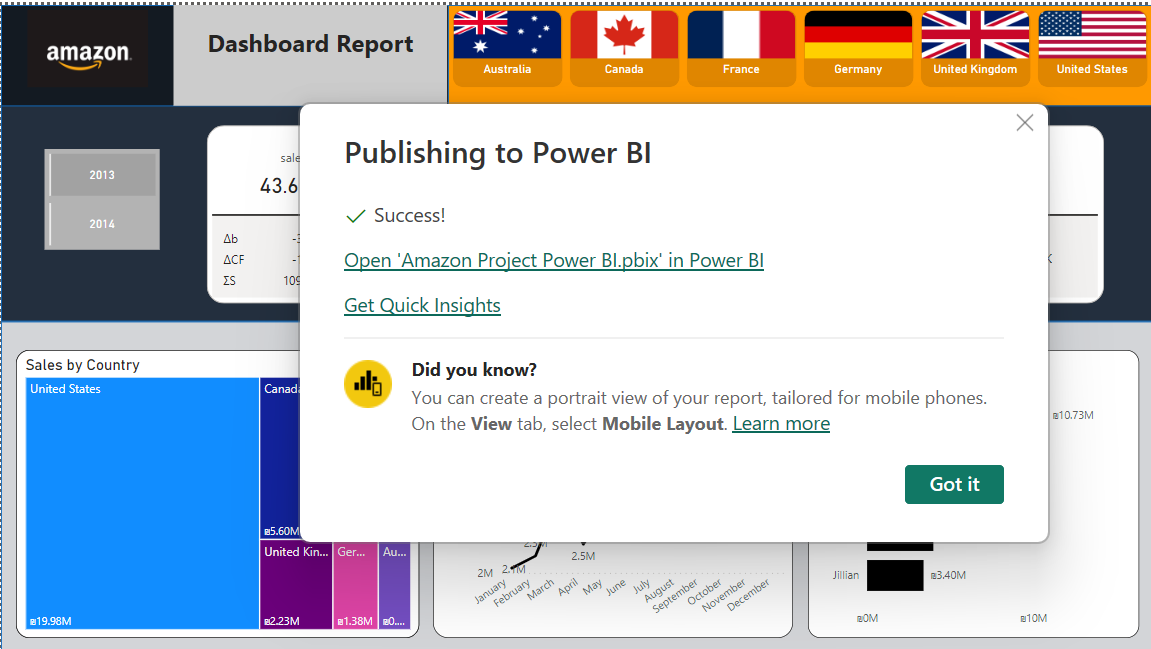
**A screenshot of a computer

Description automatically generated**

On the Employees Report, you can view the Sales, Quantity ,Orders , Employees,Avg Employees sales ,And each of them has useful indicators and also Sales by Country , Month Sales change by Month , Sales by store. you can slice the data for each of the top-selling countries. For example, for Canada or 2014

**Publish to Power BI Service**

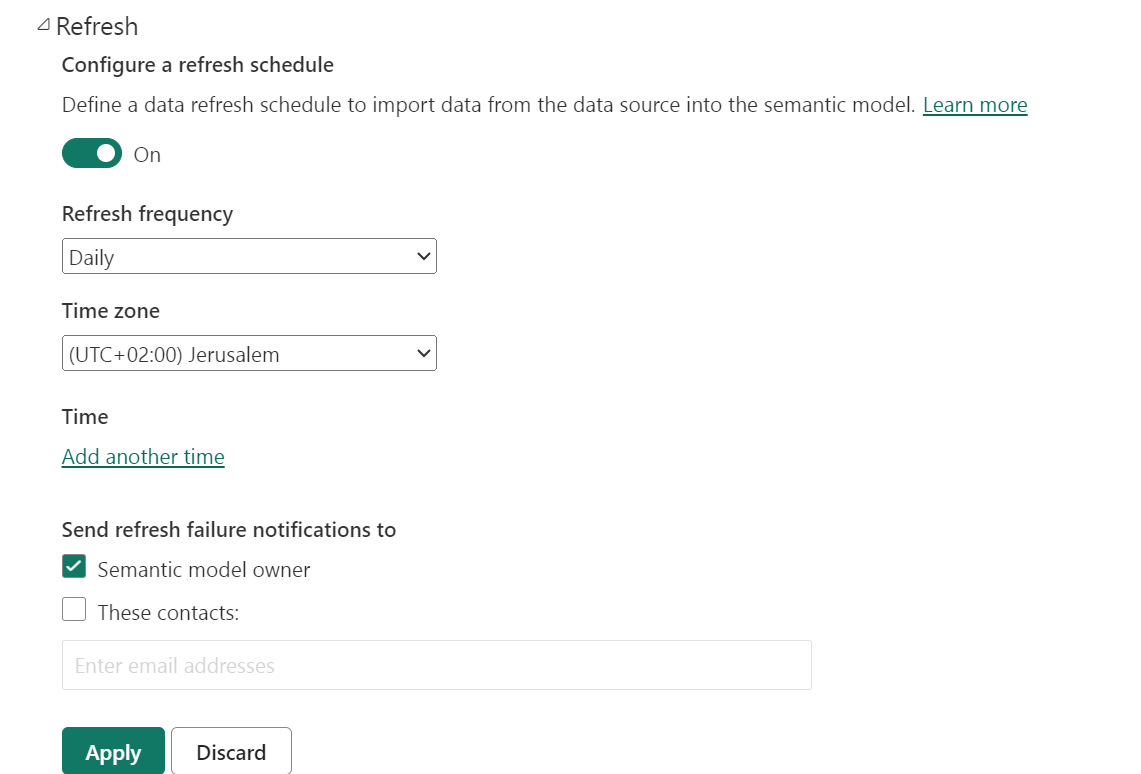
At the end of the project, I publish my report to Power BI Service.



In Power BI Service, I have activated the Gateway, activated the data source and added an automatic update:

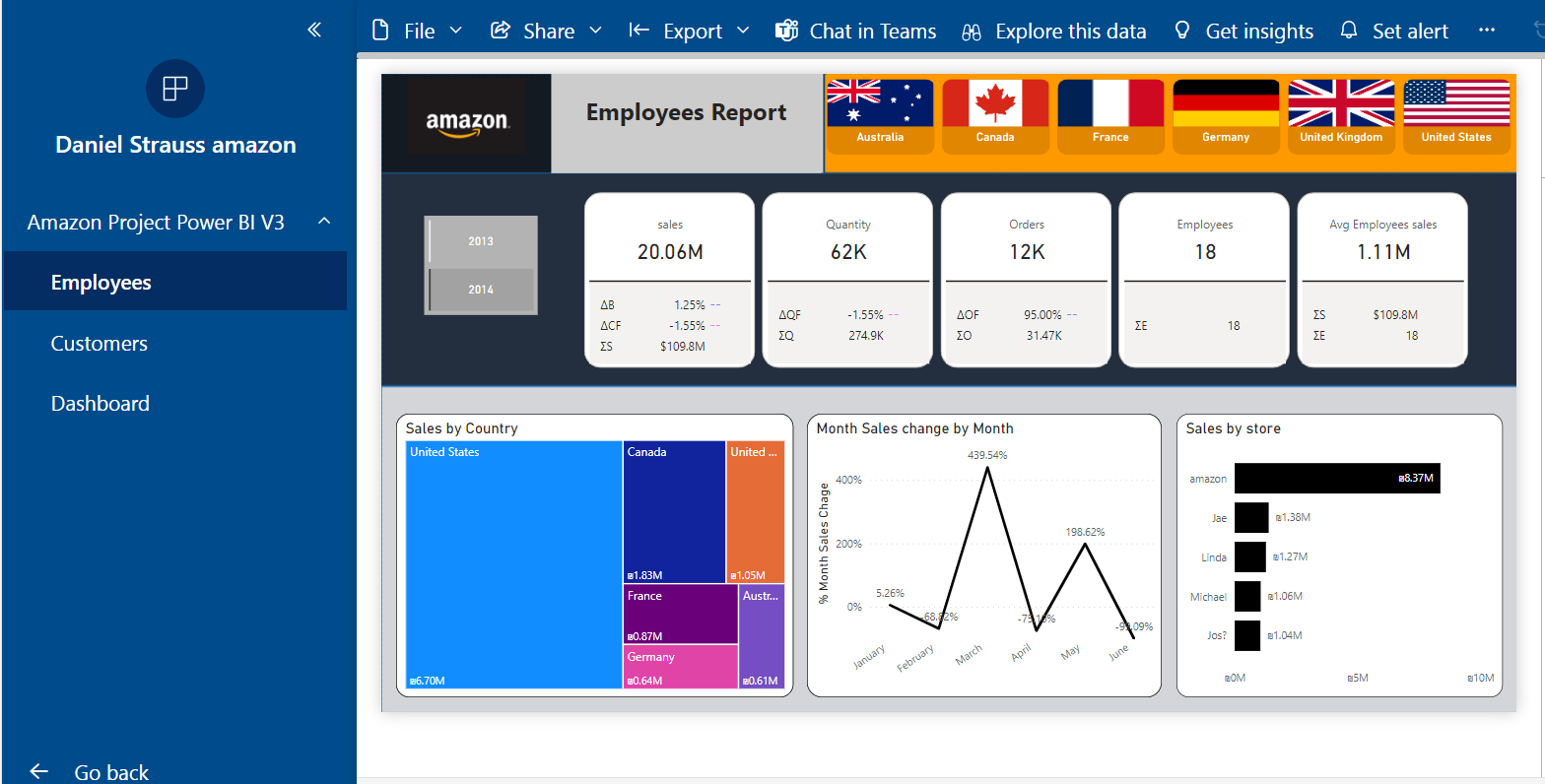
A close-up of a contact

Description automatically generated

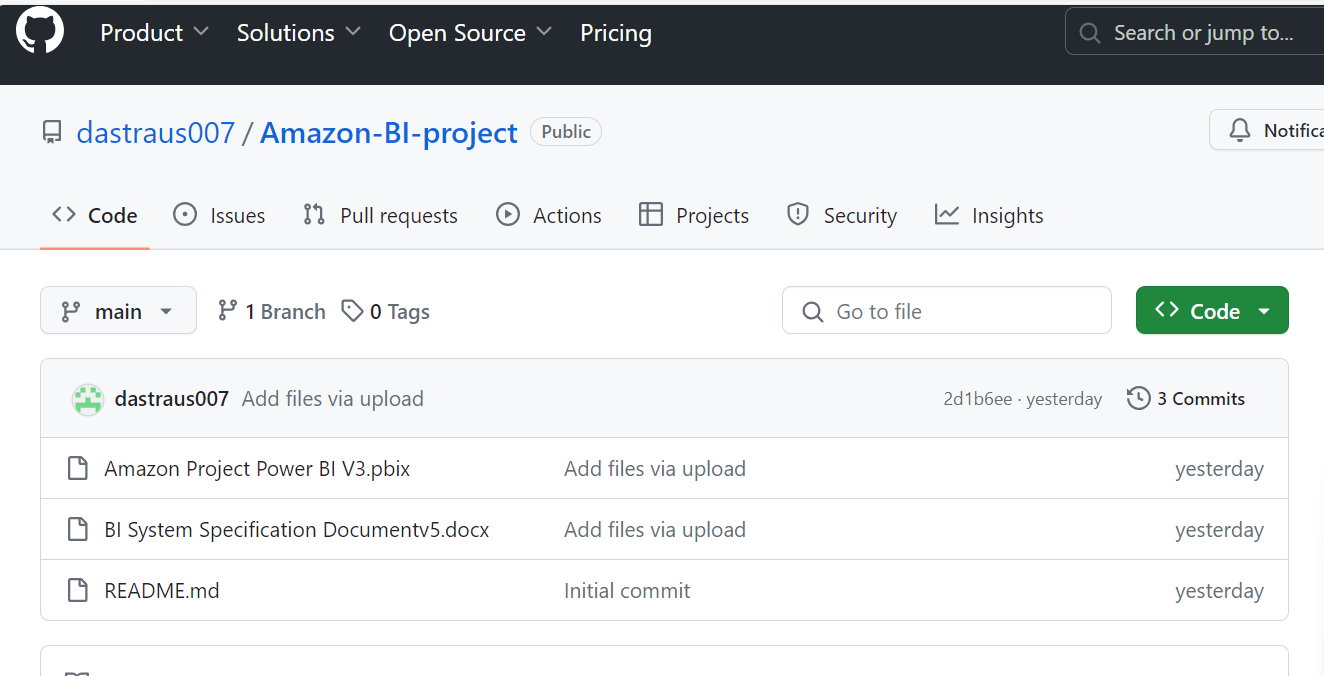


Then created an App called **Amazon Project Power BI**

[to share](https://app.powerbi.com/Redirect?action=OpenApp&appId=45cbc7e1-a94b-420b-ae04-4edd29ce7b26&ctid=155e691f-6455-40f2-b075-37035eb699f8)



And upload the project to GitHub



[Link to git](https://github.com/dastraus007/Amazon-BI-project)