Ministry of Communications and Information Technology
Digital Egypt Pioneers Initiative DEPI

Financial Data Aggregation
Data Engineering Application in Sales Financial Data

Engr. Karem Ashraf Hosny

Engr. Philopatir Sorial

Engr. Abdelmasih Fawzy Waqiem

Engr. Youssef Mahmoud

Engr. Barakat Ramadan Barakat

Engr.Feras Hatem Omar

Supervised By

Dr Eslam Hussain

Submitted in partial fulfillment of the DEPI project.

OCT, 2024

ABSTRACT

The development of data warehouses is critical in financial data management to ensure efficient reporting and analysis. In our graduation project, we focus on designing and implementing a financial data warehouse to integrate, clean, and transform raw financial data from multiple sources into an organized repository. The purpose is to enhance decision-making through a structured system for querying and data visualization. Two primary approaches are considered for this task, both involving well-known tools in the data engineering field.

Our first approach utilizes SQL Server Management Studio (SSMS) and SQL Server Integration Services (SSIS) for the extraction, transformation, and loading (ETL) processes, followed by Power BI for data visualization and dashboarding. The second approach incorporates Python for initial data cleaning to handle complex transformations, then employs SSIS for ETL and mapping, leading into the data warehouse creation. In both cases, Power BI is used to develop dynamic dashboards that allow stakeholders to interpret financial data effectively.

In the second phase of the project (as covered in Report 2), we will validate our ETL process and data models through comprehensive testing. The tools employed, including SSMS, SSIS, Python, and Power BI, will be evaluated for their performance, usability, and accuracy in delivering actionable insights. The results will focus on the efficiency of our ETL pipelines, data consistency, and the quality of the financial dashboards produced. Our efforts aim to achieve a streamlined and scalable financial reporting system that provides clarity and supports strategic financial decision-making.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Eng. Eslam Hussain for his invaluable guidance and support throughout the course of this project. His expertise and dedication have been instrumental in helping us achieve our objectives. We also extend our thanks to the Digital Egypt Pioneers Initiative (DEPI) for providing us with the resources and opportunities that enabled the successful completion of this project. Their efforts and encouragement have greatly contributed to our growth and learning in the field of data engineering.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

In today's competitive market, companies generate a vast amount of sales data. However, this raw data is often fragmented across various systems, making it difficult to derive actionable insights. To enhance decision-making, companies need a centralized data warehouse that consolidates sales data from different sources and provides insights through dashboards.

1.2 Project Objectives

This project aims to design and implement a data warehouse for a company's sales database. The primary objectives are:

- Consolidate sales data into a centralized data warehouse.
- Design an efficient ETL process to clean, transform, and load data.
- Develop intuitive dashboards for business intelligence and reporting.

1.3 Scope of the Project

The scope of this project includes:

- Developing a data warehouse to store sales data.
- Implementing an ETL process using three different approaches:
    1. SQL Server, SSMS, and SSIS.
    2. Microsoft Azure.
    3. Python (Pandas, NumPy).
- Creating dashboards using Power BI to visualize key sales metrics.

The project will explore each approach's benefits and challenges and provide recommendations based on performance, scalability, and ease of use.

1.4 Methodology Overview

The project methodology includes three different approaches to building the data warehouse and ETL process:

- **Approach 1:** Using SQL Server, SSMS, and SSIS to perform ETL and Power BI for dashboards.
- **Approach 2:** Utilizing Microsoft Azure to perform cloud-based ETL and Power BI for dashboards.
- **Approach 3:** Using Python for data cleaning, followed by SSIS for ETL and Power BI for dashboards.

Each approach will be evaluated based on performance, ease of implementation, and suitability for scaling.

1.5 Structure of the Report

This report is organized into several chapters. Chapter 1 provides the introduction and project background. Chapter 2 reviews relevant literature, including data warehousing concepts, ETL processes, and the use of cloud computing for data management. Chapters 3 through 6 cover the system design and implementation of each approach. Chapter 7 discusses testing and validation, and Chapter 8 concludes with a comparison of the three approaches, followed by recommendations.

CHAPTER 2

Literature Review

(Literature Review, Existing Work)

2.1 Data Warehousing Concepts

Data warehousing is a critical component of modern business intelligence and analytics, serving as a centralized repository for storing, retrieving, and managing large volumes of data. A data warehouse integrates data from various sources, including operational databases, external data sources, and transactional systems, to provide a unified view of the organization's data. This process facilitates better decision-making by enabling users to perform complex queries and generate reports.

The architecture of a data warehouse typically comprises three main layers: the staging area, the data warehouse, and the presentation layer. The staging area is where data is initially collected and transformed; the data warehouse stores cleaned and structured data; and the presentation layer provides access to end-users through reporting tools and dashboards. Commonly used data warehouse models include star schema and snowflake schema, which define the structure of data organization for efficient querying and reporting.

2.2 ETL Process

The Extract, Transform, Load (ETL) process is a fundamental procedure in data warehousing, responsible for collecting data from various sources, transforming it into a suitable format, and loading it into the data warehouse. The **Extract** phase involves retrieving data from different operational systems, databases, and files. The data can come from various sources such as ERP systems, CRM systems, and flat files.

In the **Transform** phase, data is cleaned, enriched, and transformed to ensure consistency and accuracy. This step may include data validation, deduplication, standardization, and aggregation. The final phase, **Load**, involves moving the transformed data into the data warehouse, where it

can be accessed for analysis and reporting. An efficient ETL process is crucial for maintaining data integrity and ensuring that users have access to up-to-date and relevant information.

2.3 Business Intelligence and Dashboards

Business Intelligence (BI) refers to the technologies, applications, and practices used to collect, analyze, and present business data. BI enables organizations to make informed decisions based on data-driven insights. Dashboards are a key component of BI, providing a visual representation of data through charts, graphs, and metrics that summarize key performance indicators (KPIs).

Effective dashboards allow stakeholders to monitor performance, identify trends, and track the progress of strategic initiatives in real-time. They can be customized to suit the needs of different users, allowing for both high-level overviews and detailed analyses. The integration of dashboards with data warehouses facilitates timely access to relevant data, enabling users to derive actionable insights quickly.

2.4 Cloud Computing and Azure

Cloud computing has revolutionized the way organizations manage and store their data. It provides scalable, flexible, and cost-effective solutions for data storage and processing. Microsoft Azure is a leading cloud computing platform that offers a wide range of services, including data storage, analytics, and machine learning. Azure provides tools for building, deploying, and managing applications in the cloud, allowing organizations to leverage powerful resources without the need for extensive on-premises infrastructure.

Azure's data services, such as Azure SQL Database, Azure Data Factory, and Azure Synapse Analytics, support the implementation of data warehousing and ETL processes. These tools enable organizations to build data pipelines, perform complex data transformations, and create robust analytical solutions. The cloud-based approach to data warehousing allows for increased collaboration, reduced operational costs, and enhanced scalability to meet evolving business needs.

CHAPTER 3

System Design

3.1 Data Warehouse Architecture

The architecture of a data warehouse is designed to support data integration, storage, and retrieval efficiently. A typical architecture consists of three main components: the staging area, the data warehouse, and the presentation layer. The **staging area** serves as the initial point for data extraction, where raw data is collected and temporarily stored before transformation. This area ensures that data is not directly loaded into the warehouse without prior cleansing and validation.

The **data warehouse** is the core component, where processed and structured data is stored. It is organized using dimensional modeling techniques, such as star or snowflake schemas, which allow for efficient querying and reporting. The **presentation layer** consists of reporting tools and dashboards, providing users with access to the data warehouse for analysis. This architecture promotes a separation of concerns, ensuring that data processing and presentation are distinct, thus enhancing performance and maintainability.

3.2 Data Sources Overview

The data sources for the data warehouse include both internal and external systems. Internal sources may comprise operational databases, customer relationship management (CRM) systems, enterprise resource planning (ERP) systems, and point-of-sale (POS) systems. These sources generate transactional data that is critical for understanding sales performance, inventory levels, and customer behavior.

External data sources may include market research, social media data, and third-party data providers that offer additional insights into customer preferences and market trends. Identifying and integrating diverse data sources is essential for creating a comprehensive view of the organization’s operations, enabling more accurate forecasting and strategic planning.

3.3 Data Transformation Process

The data transformation process is integral to the ETL workflow, ensuring that raw data is converted into a usable format before being loaded into the data warehouse. This process involves several key steps, including data cleaning, validation, and transformation.

Data cleaning focuses on identifying and correcting inaccuracies, such as duplicate records and inconsistent formats. Validation checks ensure that the data meets specified quality criteria, such as range checks and referential integrity. Transformation activities may involve aggregating data,

converting data types, and calculating derived metrics, such as sales totals and profit margins. This process is crucial for maintaining high-quality data in the warehouse, which directly impacts the accuracy of analytics and reporting.

3.4 Data Modeling

Data modeling is the process of creating a structured representation of the data warehouse's data. This involves defining entities, attributes, and relationships between different data elements. Two common approaches to data modeling are **dimensional modeling** and **normalized modeling**.

In dimensional modeling, data is organized into fact and dimension tables. Fact tables store quantitative data for analysis, such as sales figures and quantities sold, while dimension tables contain descriptive attributes related to the facts, such as product details, customer information, and time periods. This approach facilitates easy querying and reporting, as users can analyze data across multiple dimensions.

Normalized modeling, on the other hand, minimizes data redundancy by organizing data into smaller, related tables. This approach is more complex and may be better suited for transactional databases than for analytical purposes. The choice of data modeling technique depends on the specific requirements of the business and the intended use of the data.

CHAPTER 4

SQL Server and SSIS

4.1 System Setup

Database Name: EO_AdventureWorksDW2022
Tools Used:

- SQL Server: For database creation and management.
- SQL Server Integration Services (SSIS): For ETL operations.
- AdventureWorks2022: Source database for loading data.

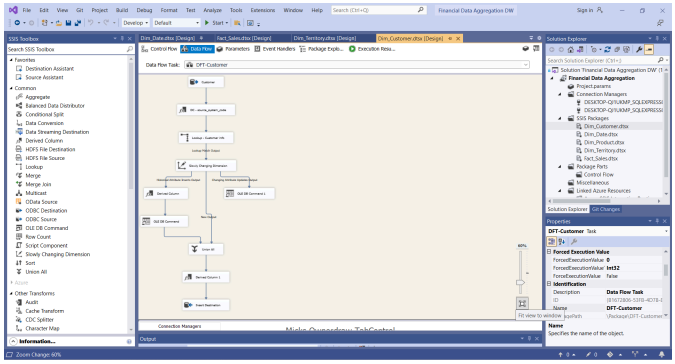
Tables Created:

- Dimension Tables: dim_product, dim_customer, dim_date, dim_territory
- Fact Table: fact_sales

Relationships:

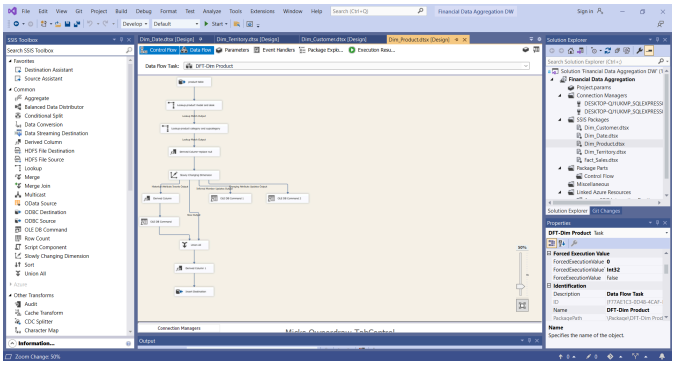
- A star schema was implemented, where the fact_sales table is at the center, connected to the dimension tables using foreign key relationships**.**

4.2 ETL Process using SQL Server



6 4.1.1

Displays the operations performed on a table fact_sales

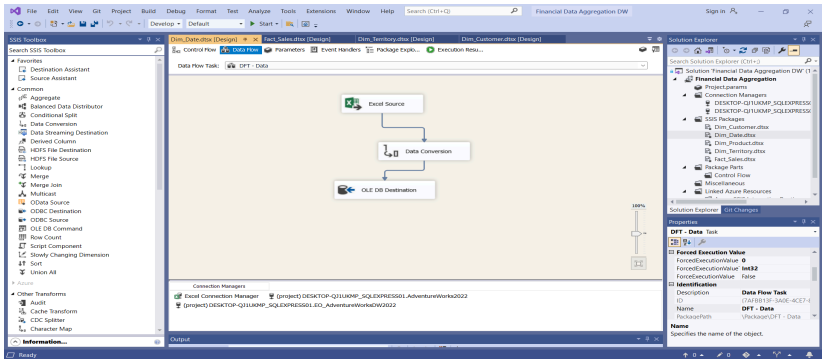

3.1.2
Displays the operations performed on a table Dim_Product

The ETL (Extract, Transform, Load) process was designed and implemented using SSIS. Key operations include:

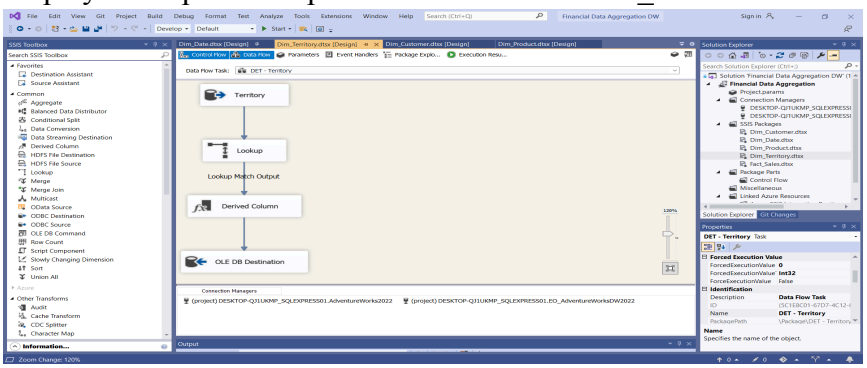
- Data Extraction: Data was sourced from AdventureWorks2022, a relational database.
- Data Transformation: Various transformations were applied to clean and structure the data as per the requirements of the data warehouse.
    - Merge Join: Used to combine data from different sources based on common columns.
    - Lookup: Implemented for data validation and to find related values from reference tables.
    - Derived Column: Added calculated fields based on existing data.
    - Slowly Changing Dimension (SCD): Handled historical changes in the dimension tables, ensuring both current and previous data states were retained.
- Data Loading: Data was loaded into EO_AdventureWorksDW2022 using:
    - OLE DB Destination: For loading data into SQL Server tables.

4.3 Data Mapping and Transformation

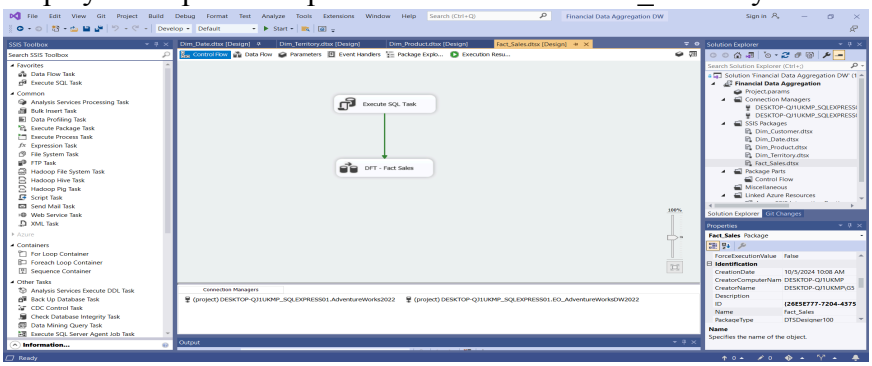
- Source Tables: The data from source tables in AdventureWorks2022 (e.g., Product, Customer, Sales) were mapped to corresponding dimension and fact tables in EO_AdventureWorksDW2022.
    - Product → dim_product
    - Customer → dim_customer
    - Territory → dim_territory
    - Sales → fact_sales
- Transformations:
    - Date formats were standardized and loaded into the dim_date table.
    - Sales amounts and quantities were calculated and loaded into fact_sales.
    - Slowly Changing Dimensions were managed for attributes like customer and product, ensuring that historical changes are captured appropriately.

- 4.1.  3

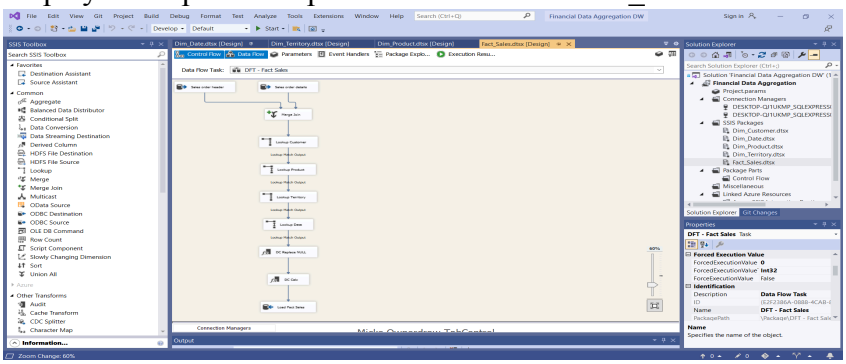
Displays the operations performed on a table Dim_Date

 4.1.4

Displays the operations performed on a table Dim_Territory

 4.1.5.1

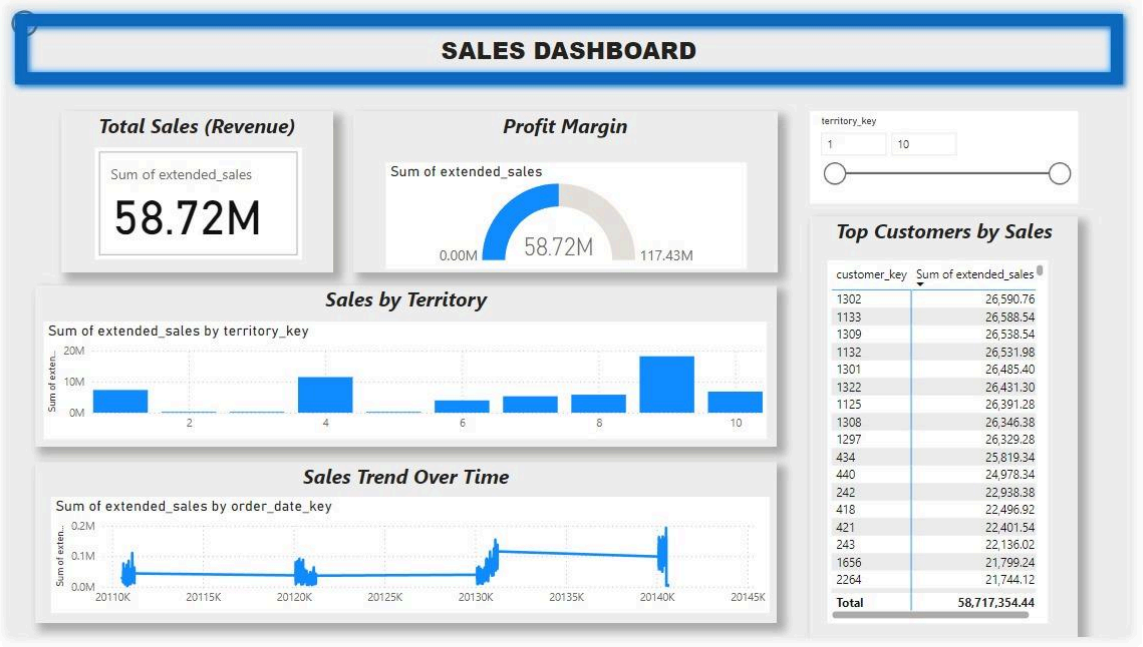
Displays the operations performed on a table fact_sales

 4.1.5.2

Displays the operations performed on a table fact_sales

4.4 Dashboards in Power BI

- Total Sales and Total Cost:
  At the top of the dashboard, two key performance indicators (KPIs) are displayed prominently: Total Sales and Total Cost. Total Sales, represented as the sum of the extended_sales field, reflects the overall revenue generated within a specified timeframe. Conversely, Total Cost, derived from the extended_cost field, indicates the total expenses incurred during sales activities. These metrics allow stakeholders to quickly assess the financial performance and profitability of the organization.
- Profit Analysis:
  A calculated KPI for Profit is presented, showcasing the difference between Total Sales and Total Cost. This metric is crucial for evaluating the organization's profitability and helps in identifying trends over time. Profit is visualized through a visually engaging gauge or card, allowing for quick interpretation.
- Sales Quantity and Average Metrics:
  The dashboard includes visualizations for Sales Quantity and Average Unit Price, presenting data on the total number of units sold and the average selling price per unit. These metrics help in understanding sales volume and pricing strategies, providing insights into customer purchasing behavior.
- Sales Breakdown by Product:
  A bar chart or pie chart illustrates Sales by Product, highlighting the revenue contribution from each product category. This visualization aids in identifying top-performing products, allowing the organization to strategize inventory and marketing efforts effectively.
- Customer and Territory Analysis:
  The dashboard features a section dedicated to Sales by Customer and Sales by Territory. These visualizations provide insights into revenue generated from various customer segments and geographical areas. Analyzing this data helps in recognizing key accounts and optimizing sales strategies to target specific markets.
- Sales Trend Over Time:
  A line chart visualizes the Sales Trend Over Time, allowing stakeholders to monitor sales performance over days, months, or years. This trend analysis is essential for identifying seasonal patterns, understanding fluctuations in sales, and making informed predictions for future sales performance.
- Freight Costs:
  Lastly, the dashboard includes a visualization for Freight Costs, which aggregates the freight field. Understanding freight expenses is vital for assessing overall profitability, particularly in industries where transportation costs significantly impact the bottom line.



7

4.4.1 Generated dashboard based on business KPIs

Overall, the sample dashboard in Power BI provides a dynamic and interactive interface, enabling users to drill down into specific metrics and gain insights from the sales data. The visualizations and analytics included are designed to support strategic decision-making, ultimately driving improved business performance.

CHAPTER 4

azure

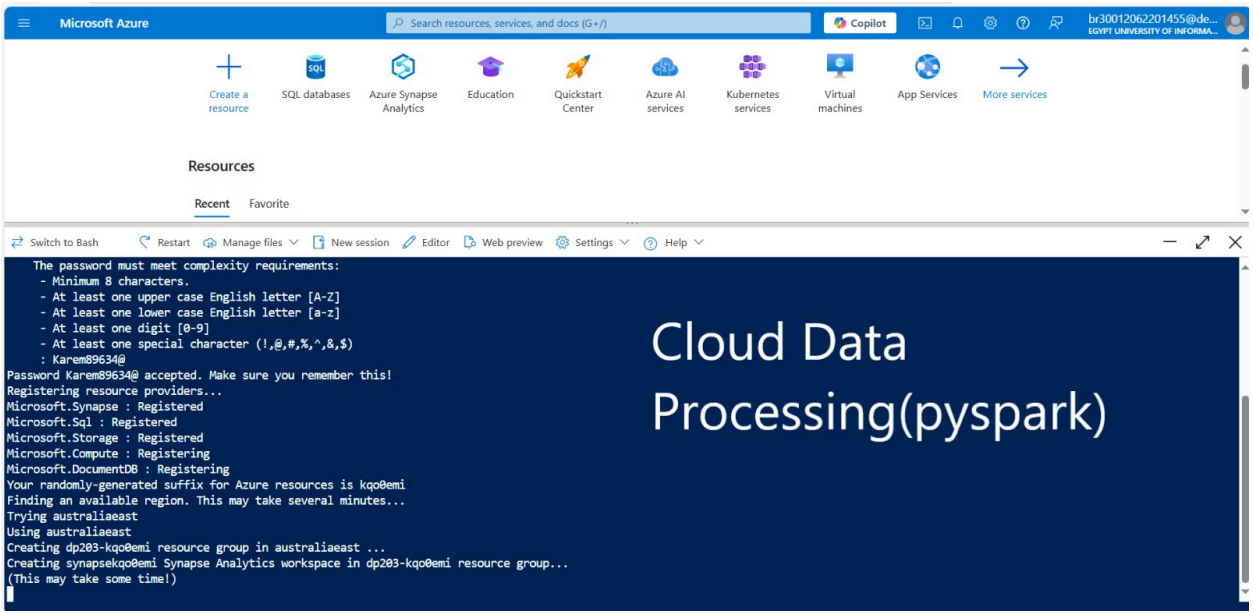
Data Visualization Using

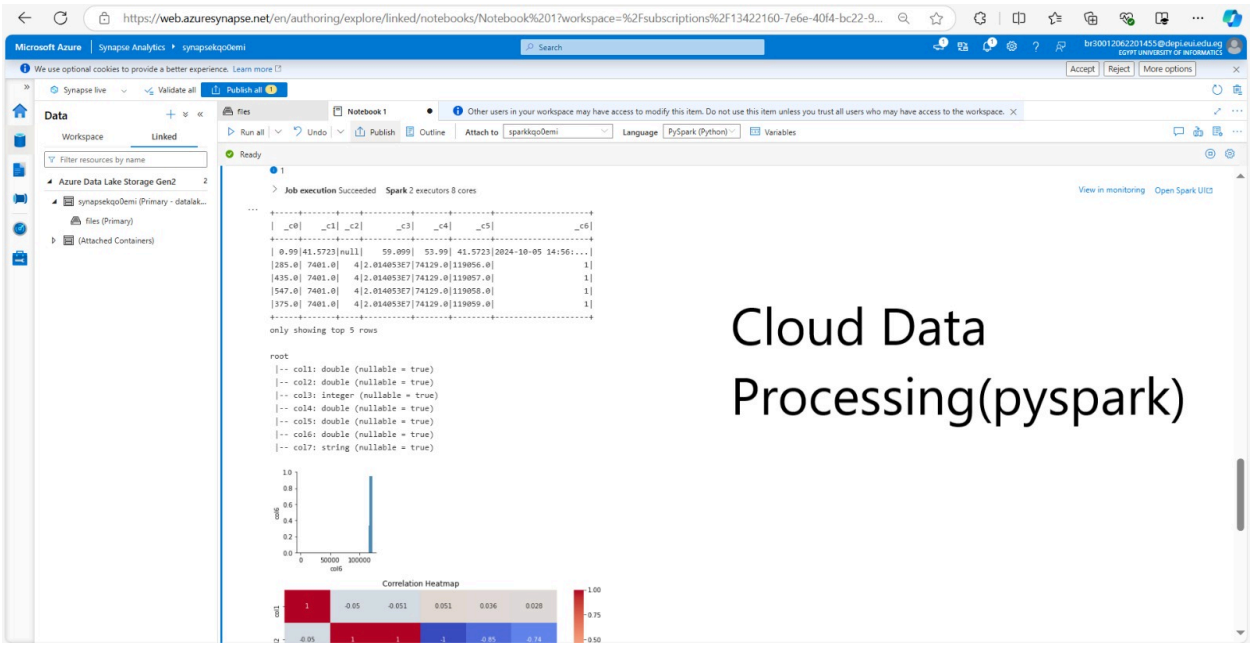
- **Azure Synapse Analytics**
- **Azure Synapse Link**

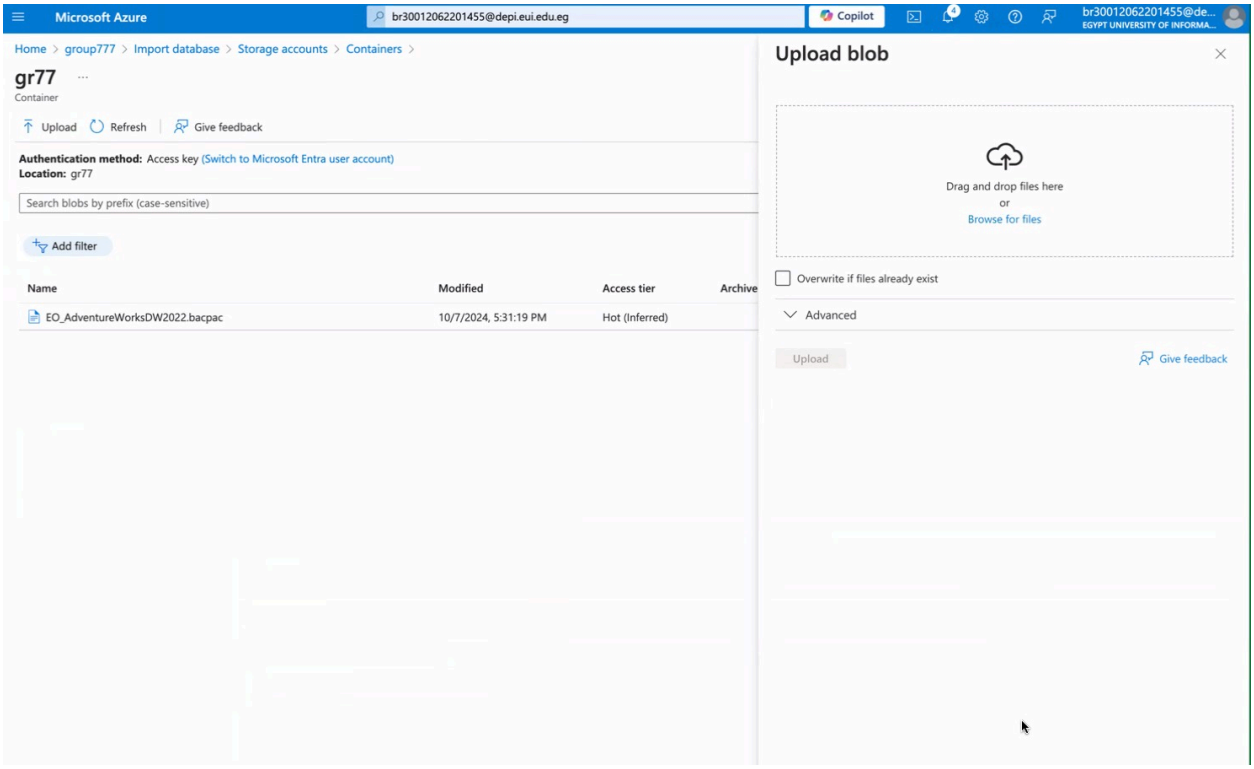
steps followed to perform data analysis using Azure Synapse Analytics and integrate data from Azure Cosmos DB through Azure Synapse Link. Additionally, Azure Data Factory was used to move data and set up visual analytics.

Using Azure Synapse Link to Connect to Azure Cosmos DB

1. **Setting Up Azure Synapse Link**:
    - Azure Synapse Link was enabled on Azure Cosmos DB to allow seamless integration between Cosmos DB and Azure Synapse Analytics.
    - I ensured the Hybrid Transactional and Analytical Processing (HTAP) feature was enabled for Cosmos DB, facilitating analytics without affecting the operational performance.
2. **Creating a Link Between Cosmos DB and Synapse**:
    - A direct connection between the Azure Synapse Workspace and the Cosmos DB was established, allowing real-time data import for analysis.
3. **Importing Data into SQL Pool**:
    - Using Synapse Link, I exported the required data from Cosmos DB into the Dedicated SQL Pool, where it became ready for analysis.


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
br30012062201455@de...
EGYPT UNIVERSITY OF INFORMA...
Create a resource
SQL databases
Azure Synapse Analytics
Education
Quickstart Center
Azure AI services
Kubernetes services
Virtual machines
App Services
More services
Resources
Recent
Favorite
Switch to Bash
Restart
Manage files
New session
Editor
Web preview
Settings
Help
The password must meet complexity requirements:
- Minimum 8 characters.
- At least one upper case English letter [A-Z]
- At least one lower case English letter [a-z]
- At least one digit [0-9]
- At least one special character (!,@,#,%,^,&,$)
: Karem89634@
Password Karem89634@ accepted. Make sure you remember this!
Registering resource providers...
Microsoft.Synapse : Registered
Microsoft.Sql : Registered
Microsoft.Storage : Registered
Microsoft.Compute : Registering
Microsoft.DocumentDB : Registering
Your randomly-generated suffix for Azure resources is kqo0emi
Finding an available region. This may take several minutes...
Trying australiaeast
Using australiaeast
Creating dp203-kqo0emi resource group in australiaeast ...
Creating synapsekqo0emi Synapse Analytics workspace in dp203-kqo0emi resource group...
(This may take some time!)
Cloud Data Processing(pyspark)


Cloud Data
Processing(pyspark)


Upload blob
Drag and drop files here
or
Browse for files
Overwrite if files already exist
Advanced
Upload
Give feedback
gr77
Container
Upload
Refresh
Give feedback
Add filter
Name
Modified
Access tier
EO_AdventureWorksDW2022.bacpac
10/7/2024, 5:31:19 PM
Hot (Inferred)

Setting Up the Azure Synapse Analytics Environment

1. **Creating an Azure Synapse Workspace**:
   - In this step, I created a Workspace in Azure Synapse Analytics, which serves as the central environment for running queries and performing data analysis.
   - I ensured the necessary resources such as SQL Pools and Spark Pools were activated to support various operations.
2. **Creating a Dedicated SQL Pool for Analytics**:
   - A Dedicated SQL Pool was created to serve as the database that holds the data for analysis.
   - The SQL Pool's capacity was determined based on the data size and performance requirements.

Importing Data Using Azure Data Factory

1. **Setting Up Azure Data Factory for Data Transfer**:
   - I used Azure Data Factory to transfer data from the source (e.g., a database or storage file like Blob Storage) to the SQL Pool created in Azure Synapse Analytics.
   - Activities such as **Copy Data** were used to execute data transfer operations.
2. **Configuring Integration Runtimes**:
   - The appropriate Integration Runtimes were configured to facilitate data movement between systems. I selected either a Self-hosted or Azure IR Integration Runtime based on the data's location.
3. **Running the Pipeline**:
   - A pipeline was created in Azure Data Factory to stream the data from the source (to the destination (SQL Pool.
   - I validated the transferred data and the performance of the process.

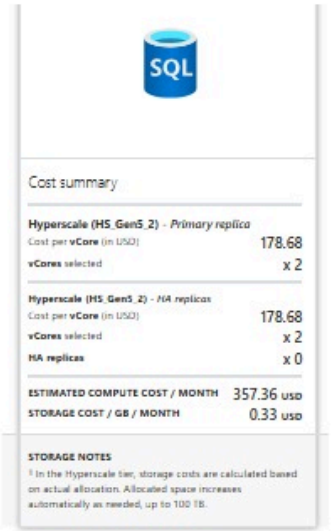
Create SQL Database

Microsoft

Product details

SQL database
by Microsoft
Terms of use | Privacy policy

Estimated cost per month
Compute cost 357.36 USD + Storage cost 0.33 USD / GB



Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.

Basics

Subscription	Azure for Students
Resource group	gr777
Region	South Africa North
Database name	group777
Server	(new) group777
Authentication method	SQL and Microsoft Entra authentication
Server admin login	baraka
Microsoft Entra Admin	br30012062201455@depi.eui.edu.eg
Compute + storage	Hyperscale: Standard-series (Gen5), 2 vCores, zone redundant disabled
Backup storage redundancy	Locally-redundant backup storage

Networking

Allow Azure services and resources to access this server	No
Private endpoint	None
Minimum TLS version	1.2
Connection Policy	Default

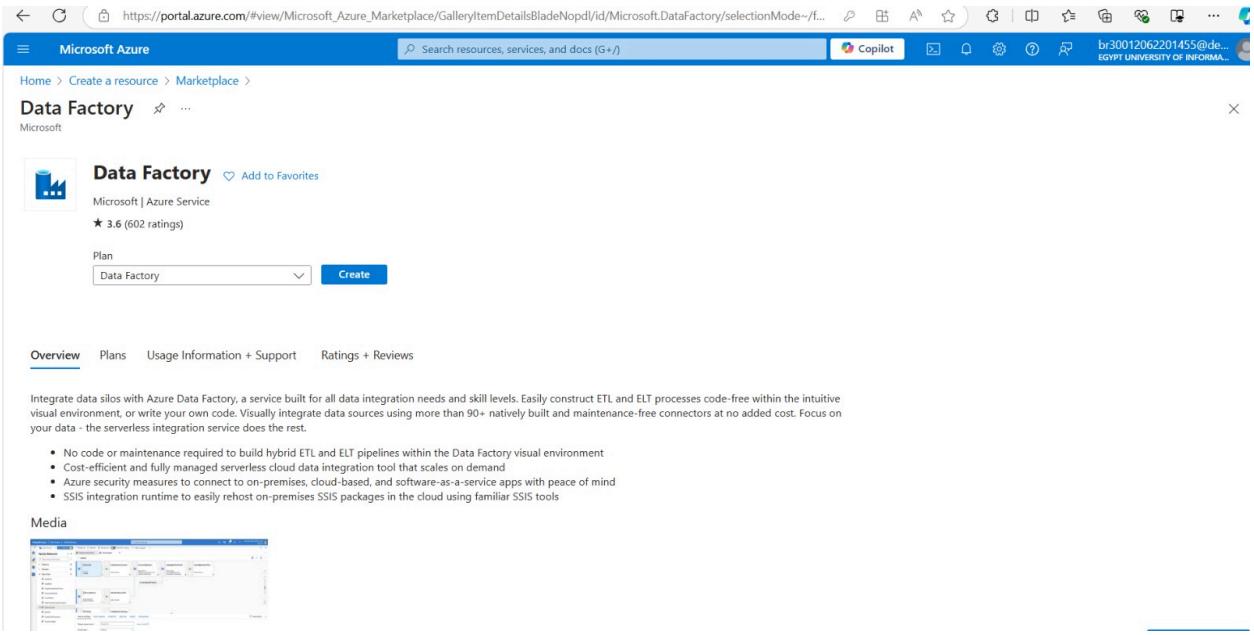
Security

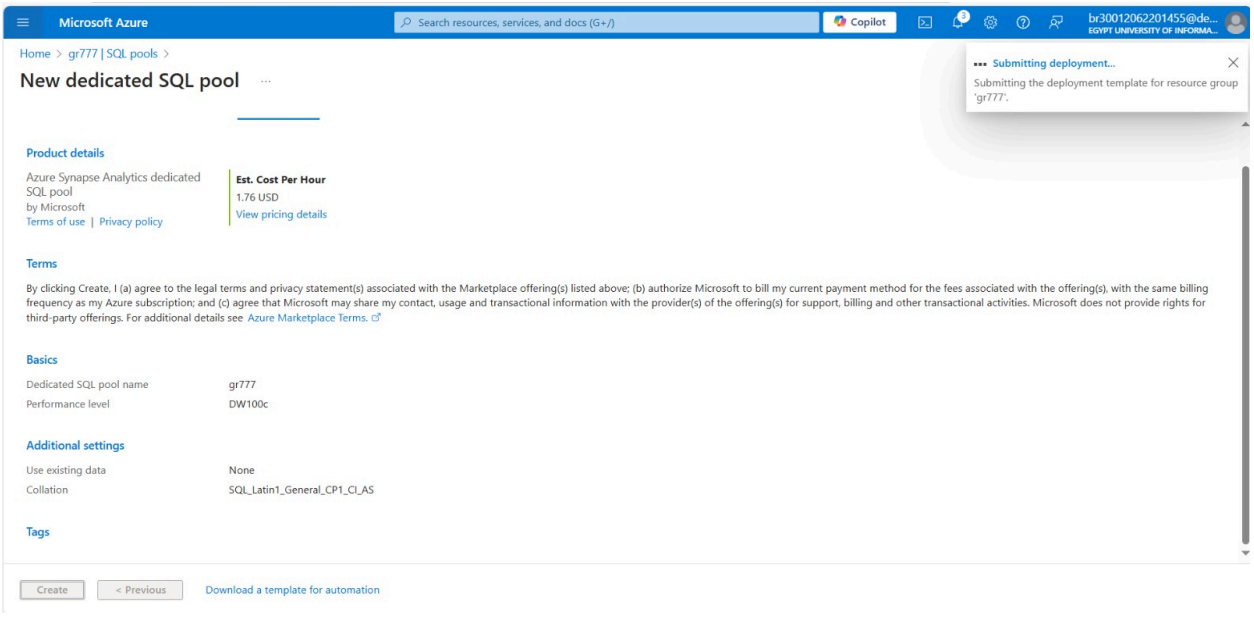
Identity	Not enabled
Transparent data encryption (Server level)	Service-managed key selected
Database level customer-managed key	Not configured
Database level user assigned managed identity	Not configured
Advanced data security	Start free trial
Always encrypted with secure enclaves	Not configured
Sql Ledger(Database)	Disabled
Digest Storage	Disabled

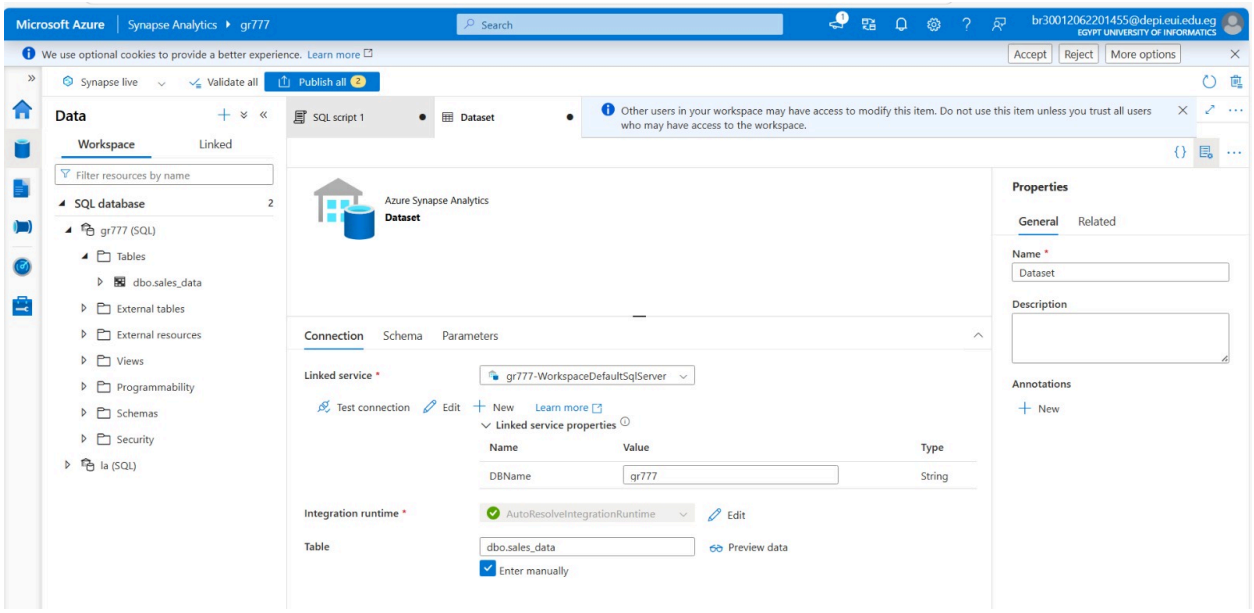
Additional settings

Use existing data	Blank
Collation	SQL_Latin1_General_CP1_CI_AS
Maintenance window	System default (5pm to 8am)

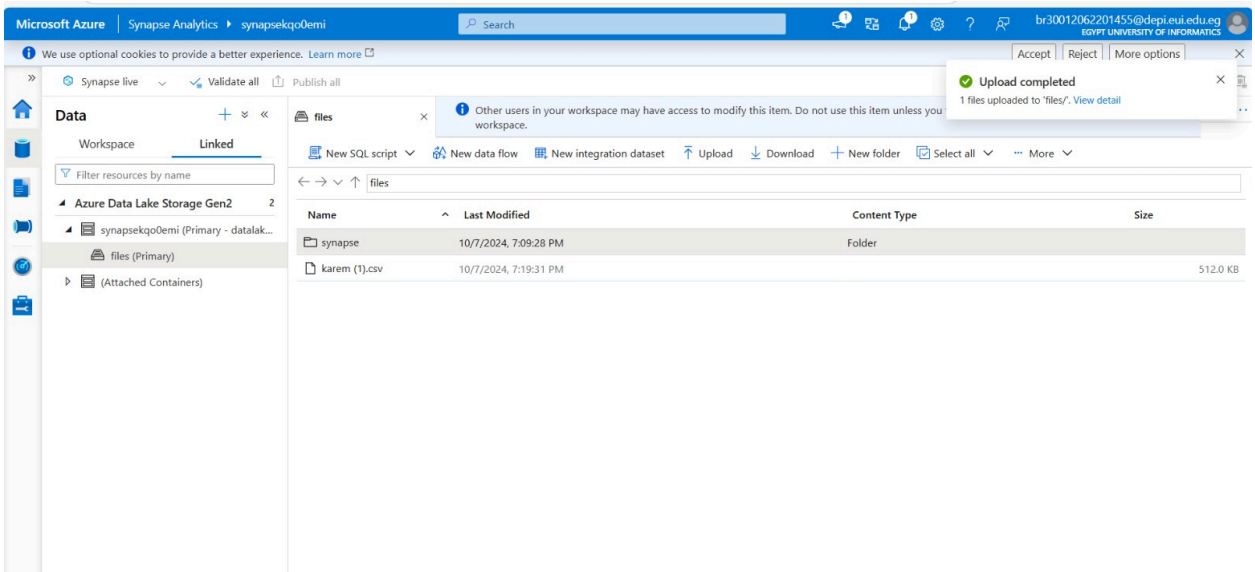
Tags

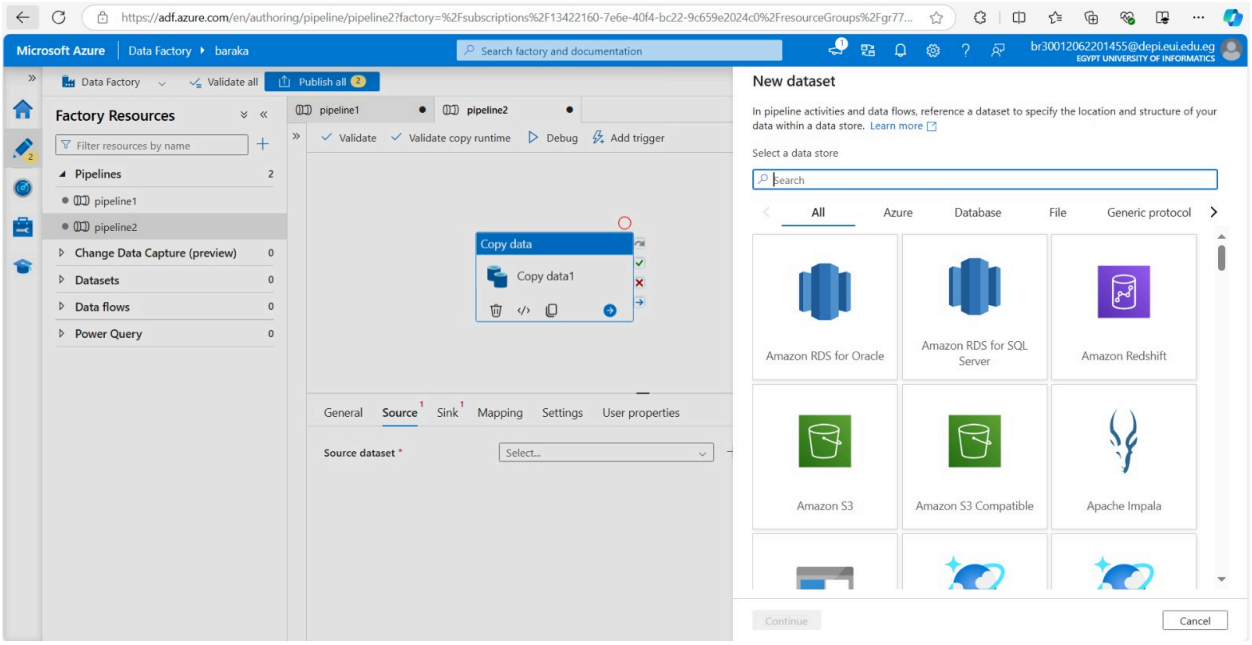

https://portal.azure.com/#view/Microsoft_Azure_Marketplace/GalleryItemDetailsBladeNopdl/id/Microsoft.DataFactory/selectionMode~/f...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
br30012062201455@de...
EGYPT UNIVERSITY OF INFORMA...
Home > Create a resource > Marketplace >
Data Factory
Microsoft
Data Factory
Add to Favorites
Microsoft | Azure Service
★ 3.6 (602 ratings)
Plan
Data Factory
Create
Overview
Plans
Usage Information + Support
Ratings + Reviews
Integrate data silos with Azure Data Factory, a service built for all data integration needs and skill levels. Easily construct ETL and ELT processes code-free within the intuitive visual environment, or write your own code. Visually integrate data sources using more than 90+ natively built and maintenance-free connectors at no added cost. Focus on your data - the serverless integration service does the rest.
No code or maintenance required to build hybrid ETL and ELT pipelines within the Data Factory visual environment
Cost-efficient and fully managed serverless cloud data integration tool that scales on demand
Azure security measures to connect to on-premises, cloud-based, and software-as-a-service apps with peace of mind
SSIS integration runtime to easily rehost on-premises SSIS packages in the cloud using familiar SSIS tools
Media

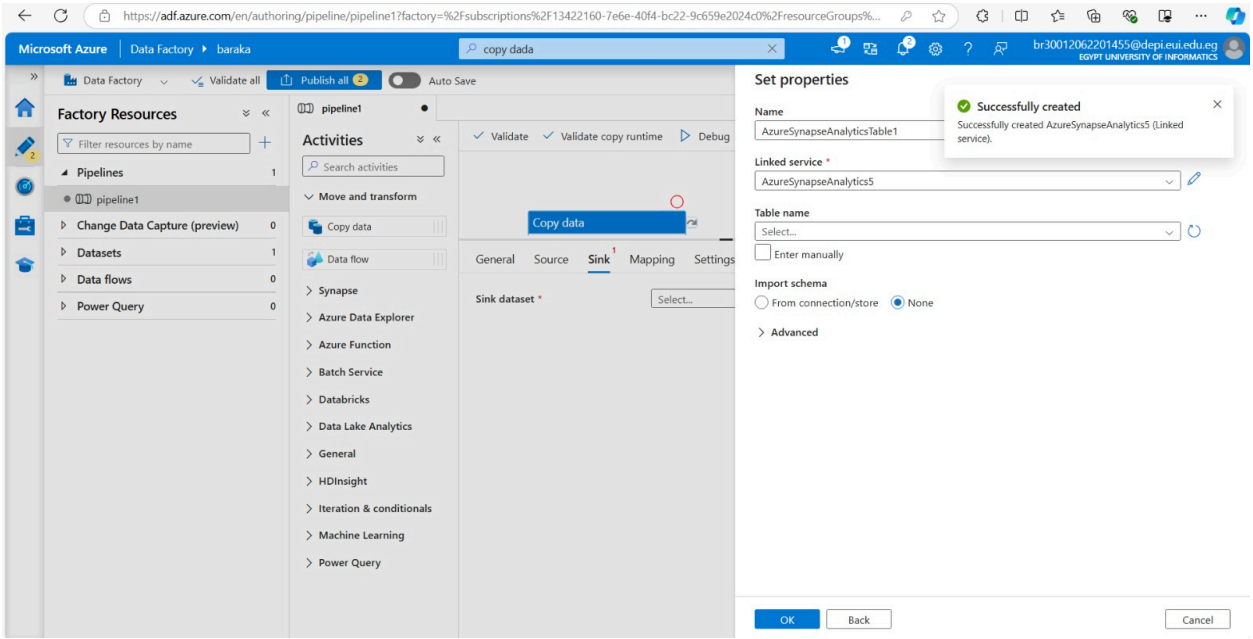

Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
br30012062201455@de...
EGYPT UNIVERSITY OF INFORMA...
Home > gr777 | SQL pools >
New dedicated SQL pool
Submitting deployment...
Submitting the deployment template for resource group 'gr777'.
Product details
Azure Synapse Analytics dedicated SQL pool
by Microsoft
Terms of use | Privacy policy
Est. Cost Per Hour
1.76 USD
View pricing details
Terms
By clicking Create, I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.
Basics
Dedicated SQL pool name
gr777
Performance level
DW100c
Additional settings
Use existing data
None
Collation
SQL_Latin1_General_CP1_CI_AS
Tags
Create
< Previous
Download a template for automation


Microsoft Azure | Synapse Analytics ▸ gr777
Search
br30012062201455@depi.eui.edu.eg
EGYPT UNIVERSITY OF INFORMATICS
We use optional cookies to provide a better experience. Learn more
Accept
Reject
More options
Synapse live
Validate all
Publish all 2
Data
Workspace
Linked
Filter resources by name
SQL database 2
gr777 (SQL)
Tables
dbo.sales_data
External tables
External resources
Views
Programmability
Schemas
Security
la (SQL)
SQL script 1
Dataset
Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who may have access to the workspace.
Azure Synapse Analytics
Dataset
Connection
Schema
Parameters
Linked service *
gr777-WorkspaceDefaultSqlServer
Test connection
Edit
New
Learn more
Linked service properties
Name
Value
Type
DBName
gr777
String
Integration runtime *
AutoResolveIntegrationRuntime
Edit
Table
dbo.sales_data
Preview data
Enter manually
Properties
General
Related
Name *
Dataset
Description
Annotations
New

.


Microsoft Azure | Synapse Analytics ▸ synapsekqo0emi
Search
br30012062201455@depi.eui.edu.eg
EGYPT UNIVERSITY OF INFORMATICS
We use optional cookies to provide a better experience. Learn more
Accept
Reject
More options
Synapse live
Validate all
Publish all
Upload completed
1 files uploaded to 'files/'. View detail
Data
Workspace
Linked
Filter resources by name
Azure Data Lake Storage Gen2 2
synapsekqo0emi (Primary - datalak...
files (Primary)
(Attached Containers)
files
Other users in your workspace may have access to modify this item. Do not use this item unless you
workspace.
New SQL script
New data flow
New integration dataset
Upload
Download
New folder
Select all
More
files
Name
Last Modified
Content Type
Size
synapse
10/7/2024, 7:09:28 PM
Folder
karem (1).csv
10/7/2024, 7:19:31 PM
512.0 KB


Microsoft Azure
Data Factory
baraka
Search factory and documentation
Factory Resources
Pipelines
pipeline1
pipeline2
Change Data Capture (preview)
Datasets
Data flows
Power Query
Validate all
Publish all
Validate
Validate copy runtime
Debug
Add trigger
Copy data
Copy data1
General
Source
Sink
Mapping
Settings
User properties
Source dataset *
Select...
New dataset
In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. Learn more
Select a data store
Search
All
Azure
Database
File
Generic protocol
Amazon RDS for Oracle
Amazon RDS for SQL Server
Amazon Redshift
Amazon S3
Amazon S3 Compatible
Apache Impala
Continue
Cancel


Microsoft Azure
Data Factory
baraka
copy dada
Factory Resources
Pipelines
pipeline1
Change Data Capture (preview)
Datasets
Data flows
Power Query
Validate all
Publish all
Auto Save
Activities
Search activities
Move and transform
Copy data
Data flow
Synapse
Azure Data Explorer
Azure Function
Batch Service
Databricks
Data Lake Analytics
General
HDInsight
Iteration & conditionals
Machine Learning
Power Query
Validate
Validate copy runtime
Debug
Copy data
General
Source
Sink
Mapping
Settings
Sink dataset *
Select...
Set properties
Name
AzureSynapseAnalyticsTable1
Linked service *
AzureSynapseAnalytics5
Table name
Select...
Enter manually
Import schema
From connection/store
None
Advanced
Successfully created
Successfully created AzureSynapseAnalytics5 (Linked service).
OK
Back
Cancel

CHAPTER 6

Approach 3: Python-Based

- Established Secure Database Connection
- Data Cleaning using Python
- Exploratory Data Analysis (EDA)
- Data Visualization

In this project, I took on the role of analyzing sales data retrieved from an Azure SQL Database to uncover valuable insights that could drive business decisions. My journey began by establishing a secure and efficient connection to the Azure SQL Database using

SQLAlchemy and `pyodbc`. I ensured that the connection parameters were correctly configured, specifying the appropriate ODBC driver, server name, database name, and authentication details. This was crucial for successfully retrieving data from the `fact_sales` table.

Once the connection was established, I executed a SQL query to fetch data from the `fact_sales` table and loaded it into a pandas DataFrame named `sales_data`. This provided a structured dataset that I could work with for further analysis.

To gain an initial understanding of the data, I performed exploratory data analysis (EDA). I used `sales_data.info()` to check the data structure, including the number of entries, column names, and data types. Additionally, `sales_data.describe()` gave me statistical summaries of the numerical columns, such as mean, standard deviation, and quartile values.

Recognizing the importance of data integrity, I checked for null values in each column using `sales_data.isnull().sum()`. Identifying any missing data early allowed me to address potential issues that could affect the analysis. I also examined the data types of each column to ensure they were appropriate for the analyses I planned to perform.

To handle missing values, I implemented a forward-fill imputation strategy using `sales_data.fillna(method='ffill', inplace=True)`. This method is suitable for time-series data where the previous value is a reasonable estimate for a missing one. I also removed any duplicate records with `sales_data.drop_duplicates(inplace=True)` to maintain data integrity and prevent skewed results.

Understanding that temporal features are crucial in sales data analysis, I converted the `date` column to a datetime object using `pd.to_datetime(sales_data['date'])`. This conversion allowed me to extract additional temporal features such as `year`, `month`, `day`, and `day_of_week` using pandas datetime attributes. Extracting these features enabled me to analyze trends and patterns over different time frames.

To visualize the distribution of sales amounts, I used seaborn's `histplot` function. Plotting a histogram with a kernel density estimate provided insights into the central tendency and variability of the sales data. This helped me understand the general behavior of sales amounts and identify any outliers or anomalies.

I then proceeded to analyze sales trends over time. By grouping the data by `year` and `month`, I calculated total sales for each month using `sales_data.groupby(['year', 'month'])['sales_amount'].sum().reset_index()`. Creating a pivot table with `sales_data.pivot_table(values='sales_amount', index='month', columns='year', aggfunc='sum')` allowed me to visualize sales across different months and years effectively.

To further explore temporal trends, I plotted the total sales over time using a line chart. Grouping the sales data by month and converting the index to timestamps ensured that the data was in the correct format for plotting. This visualization highlighted seasonal patterns and growth trends, providing a clear picture of how sales evolved over the analyzed period.

I also created a heatmap to visualize monthly sales using seaborn's `heatmap` function. The heatmap provided a color-coded representation of sales volumes, making it easy to identify months or years with exceptionally high or low sales. Adjusting the figure size and adding annotations enhanced the readability of the heatmap.

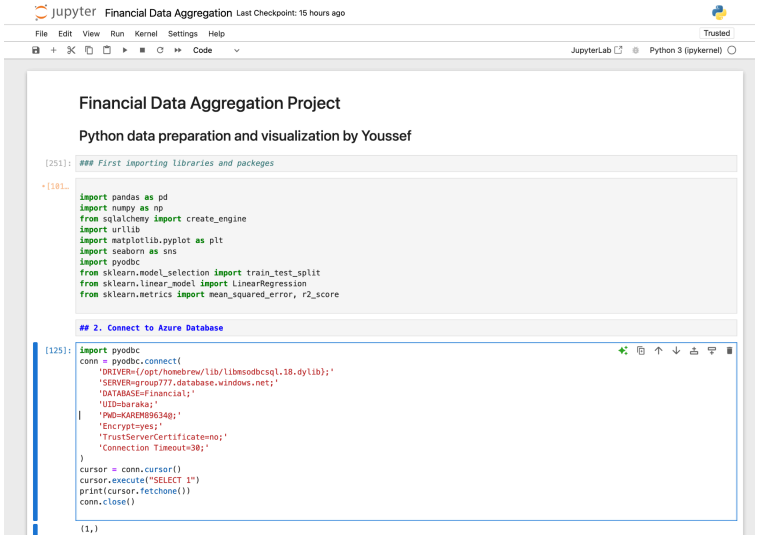
Understanding customer behavior on different days of the week is valuable for operational planning. I analyzed sales distributions by day of the week using a boxplot. By specifying the order of days, I ensured that the days were displayed from Monday to Sunday, providing an intuitive visualization. This analysis revealed which days had higher sales volumes and the variability in sales on each day, insights that could inform staffing and promotional strategies.

Finally, I developed a simple linear regression model to predict sales amounts based on temporal features. I selected `year` and `month` as predictors and the `sales_amount` as the target variable. After splitting the data into training and testing sets using scikit-learn's `train_test_split`, I trained a `LinearRegression` model. Evaluating the model with metrics like Mean Squared Error and R-squared provided a quantitative assessment of the model's performance.

Throughout the project, I ensured that all necessary libraries were imported and that the code was organized logically. I addressed potential issues such as missing imports, incorrect data types, and handling missing values to prevent runtime errors. I also paid attention to data visualization aesthetics, adjusting axis labels and tick rotations to enhance readability.

By meticulously processing the data and applying appropriate analytical methods, I was able to extract meaningful insights from the sales data. My work provided actionable findings that could inform strategic decisions, demonstrating my ability to handle end-to-end data analysis projects effectively. This project showcased my skills in data extraction, cleaning, visualization, and modeling, all of which are essential competencies in data science and analytics.

Figure 6.1: connecting to azure server and accessing the database



:

Figure 6.2: accessing data

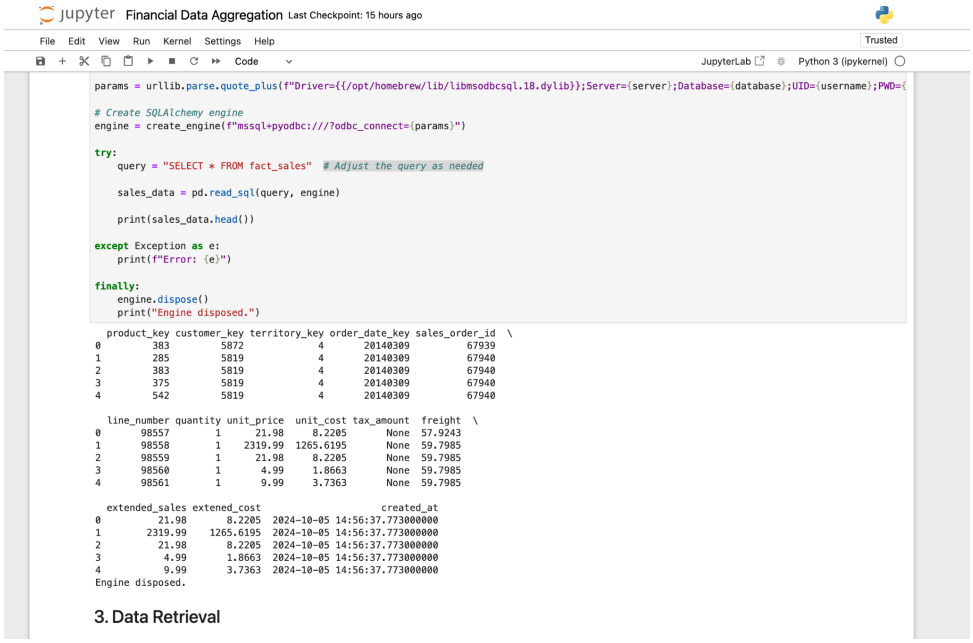
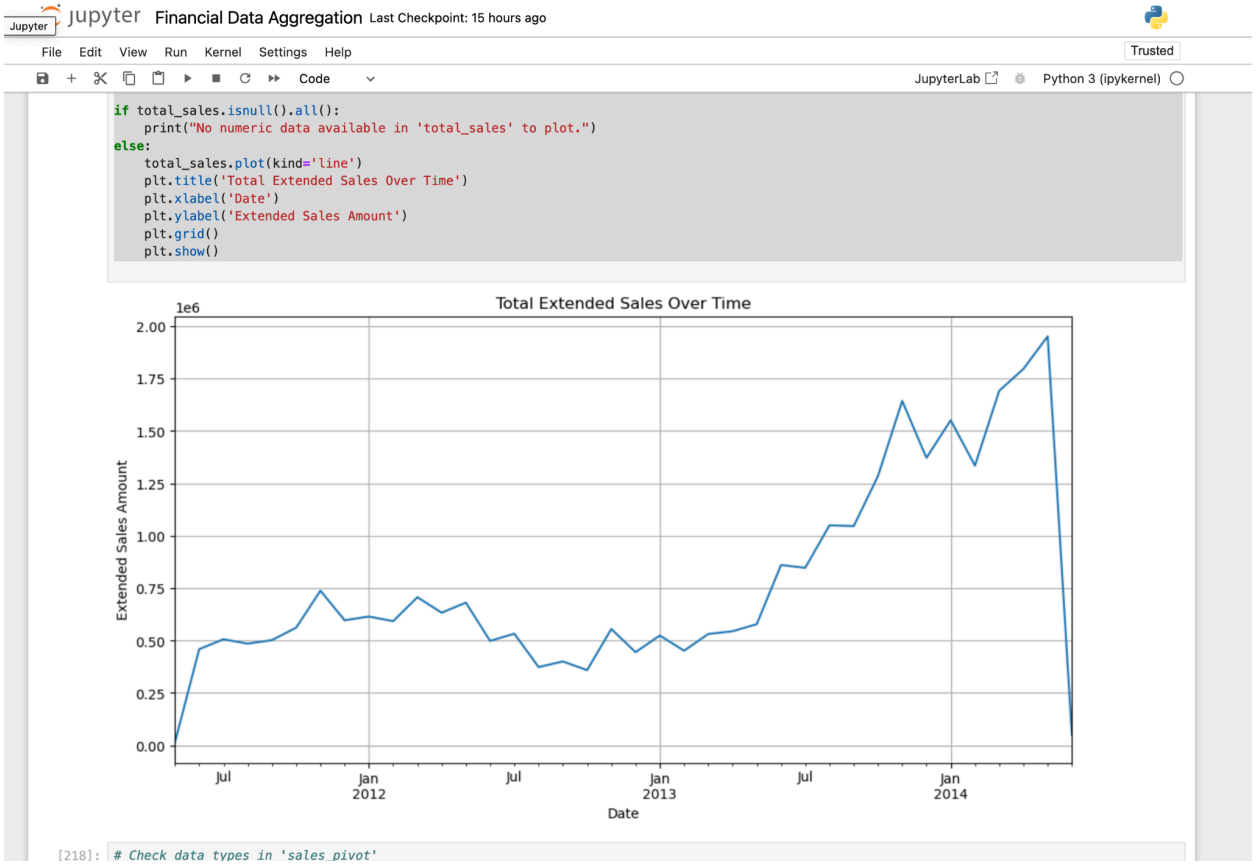
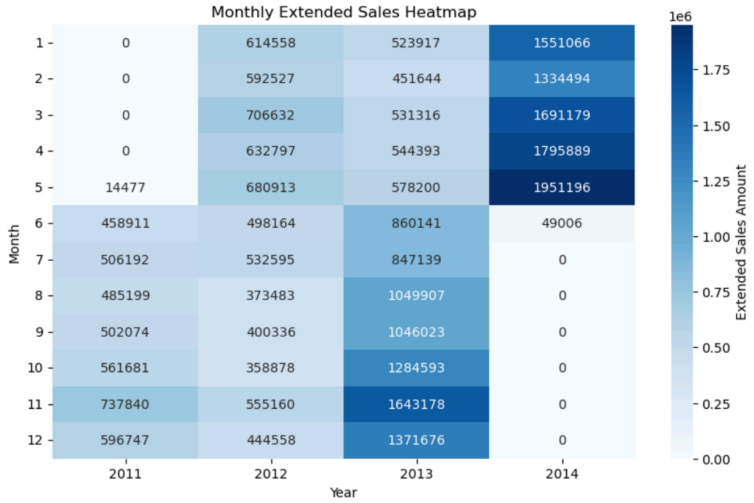


Figure 6.3: showing visualization and heatmaps



```
    cbar_kws={'label': 'Extended Sales Amount'}
)
plt.title('Monthly Extended Sales Heatmap')
plt.xlabel('Year')
plt.ylabel('Month')
plt.yticks(rotation=0)
plt.show()
```



[ ]:

[222]:
```
# Sales by day of the week
plt.figure(figsize=(10, 6))
```

Chapter 8

Analyzing Data

1. Analyzing Total Sales by Territory (`territory_key`):

This query calculates the total sales for each territory based on the `extended_sales` field.

```sql
SELECT
    territory_key,
    SUM(extended_sales) AS total_sales
FROM
    Warehouse_Analytics
GROUP BY
    territory_key
ORDER BY
    total_sales DESC;
```

Expected Result:

- **This query will display a list of territories (`territory_key`) along with the total sales for each territory (`total_sales`), sorted in descending order by total sales.**
- **You can use this information to identify which territories are generating the highest sales.**

Analyzing Total Sales by Product (`product_key`):

This query calculates the total sales for each product.

```sql
SELECT
    product_key,
    SUM(extended_sales) AS total_sales
FROM
    Warehouse_Analytics
GROUP BY
    product_key
ORDER BY
    total_sales DESC;
```

Expected Result:

- **This query will display a list of products (`product_key`) along with the total sales for each product (`total_sales`).**
- **The result will help identify the best-selling products.**

Analyzing the Number of Orders per Customer (`customer_key`):

This query counts the number of orders for each customer.

```sql
SELECT
    customer_key,
    COUNT(sales_order_id) AS total_orders
FROM
    Warehouse_Analytics
GROUP BY
    customer_key
ORDER BY
    total_orders DESC;
```

Expected Result:

- **This query will display the number of orders (`total_orders`) placed by each customer (`customer_key`).**
- **You can use this query to identify the most active customers.**

Calculating Total Profit by Product:

To calculate profit per product, you can use the difference between sales (`extended_sales`) and cost (`extened_cost`).

```sql
SELECT
    product_key,
    SUM(extended_sales - extened_cost) AS total_profit
FROM
    Warehouse_Analytics
GROUP BY
    product_key
ORDER BY
    total_profit DESC;
```

Expected Result:

- **This query will display a list of products with their total profits (`total_profit`).**
- **This information can be used to identify the most profitable products.**

Python Analysis

Analyzing Total Sales by Territory (`territory_key`):

- **I calculated the total sales (`extended_sales`) for each territory using the `groupby` function to analyze the data based on `territory_key`.**
- **I created a bar chart representing the total sales for each territory.**

```
import matplotlib.pyplot as plt

# Calculate total sales per territory
sales_per_territory = data.groupby('territory_key')['extended_sales'].sum().reset_index()

# Plot total sales per territory
plt.figure(figsize=(10,6))
plt.bar(sales_per_territory['territory_key'], sales_per_territory['extended_sales'], color
plt.title('Total Sales per Territory', fontsize=14)
plt.xlabel('Territory Key', fontsize=12)
plt.ylabel('Total Sales', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

Analyzing Total Sales by Product (`product_key`):

- **I calculated the total sales for each product (`product_key`) in a similar manner as I did for the territory analysis.**
- **I created a bar chart representing the total sales for each product.**

```
# Calculate total sales per product
sales_per_product = data.groupby('product_key')['extended_sales'].sum().reset_index()

# Plot total sales per product
plt.figure(figsize=(10,6))
plt.bar(sales_per_product['product_key'], sales_per_product['extended_sales'], color=
plt.title('Total Sales per Product', fontsize=14)
plt.xlabel('Product Key', fontsize=12)
plt.ylabel('Total Sales', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

Chapter 9

CONCLUSION AND FUTURE WORK

8.1 Summary of Results

In this section, summarize the key outcomes from the project:

- Successfully implemented a data warehouse using multiple approaches to handle the company's sales data.
- The ETL process was completed using SQL Server, SSIS, and Power BI for dashboard visualizations, providing valuable insights into sales performance and business operations.

- Alternative approaches, such as using Python for data cleaning and Microsoft Azure for cloud-based ETL, were explored and proved to be viable for scaling the project in the future.
- The dashboard enabled key decision-makers to track essential KPIs, including total sales, profit, sales trends, and regional performance.

8.2 Strengths and Limitations of Each Approach

Approach 1: SQL Server, SSIS, Power BI

- **Strengths**:
  - SQL Server and SSIS provided a stable environment for managing structured data and executing ETL processes efficiently.
  - Power BI dashboards allowed for easy visualization and interaction with the data.
  - This approach was effective for handling large datasets within an on-premises setup.
- **Limitations**:
  - Requires significant infrastructure, making it less flexible for scaling beyond a certain point.
  - Limited automation for data cleaning compared to more modern tools like Python.

Approach 2: Python, SSIS, Power BI

- **Strengths**:
  - Python enabled robust and flexible data cleaning, handling complex transformations more efficiently than SQL-based solutions.
  - SSIS and Power BI integration allowed for a smooth ETL and reporting process.
- **Limitations**:
  - Python-based solutions may require higher technical expertise.
  - Combining Python with SSIS can be less seamless than sticking with a fully integrated Microsoft suite.

Approach 3: Microsoft Azure

- **Strengths**:
  - Cloud-based architecture offers greater scalability and flexibility.
  - Easier to integrate various data sources and perform real-time analytics.
  - Lower infrastructure cost with high accessibility.
- **Limitations**:
  - Steeper learning curve for those unfamiliar with cloud technologies.
  - Requires stable internet connections, which could be a challenge in some regions.

8.3 Future Work and Recommendations

- **Automation**: Implement automated processes for data extraction, transformation, and loading, reducing manual intervention and improving efficiency.

- **Data Sources Expansion**: Integrate additional data sources such as customer behavior data, marketing campaigns, or competitor analysis for more comprehensive business insights.
- **Cloud Migration**: Explore full migration to cloud solutions like Microsoft Azure to enable real-time data processing and more scalable solutions.
- **Advanced Analytics**: Incorporate machine learning algorithms to predict sales trends, customer behaviors, and market demand, thus enhancing decision-making capabilities.
- **Security and Data Governance**: As data grows, implement advanced data governance and security frameworks to ensure compliance with data protection laws and to safeguard sensitive information.

APPENDICES

Appendix 1: SQL Queries Used

- https://github.com/Al-Moatasem/sales-data-mart

Appendix 2: Python Code Snippets

- This section contains Python scripts used for data cleaning, manipulation, and transformation.

Appendix 3: Power BI Screenshots

- https://drive.google.com/file/d/1_pWDMewCQvcBdqi7bGDr0fM88ShQnvuu/view

Appendix 4: List of Tools and Technologies

- Detailed list of software and tools used in the project, including SQL Server, SSMS, SSIS, Microsoft Azure, Python libraries (Pandas, NumPy), and Power BI.

- SSIS

Appendix 5: The data preparation and visualization notebook *note this is view only

Financial Data Aggregation file.ipynb

REFERENCES

The list of books, articles and other sources should be listed at the end of the report. All references must be used/cited in the text. The general format is as follows (use IEEE format):

Book

1. DEPI Materials of (Python, Data WH, ETL).