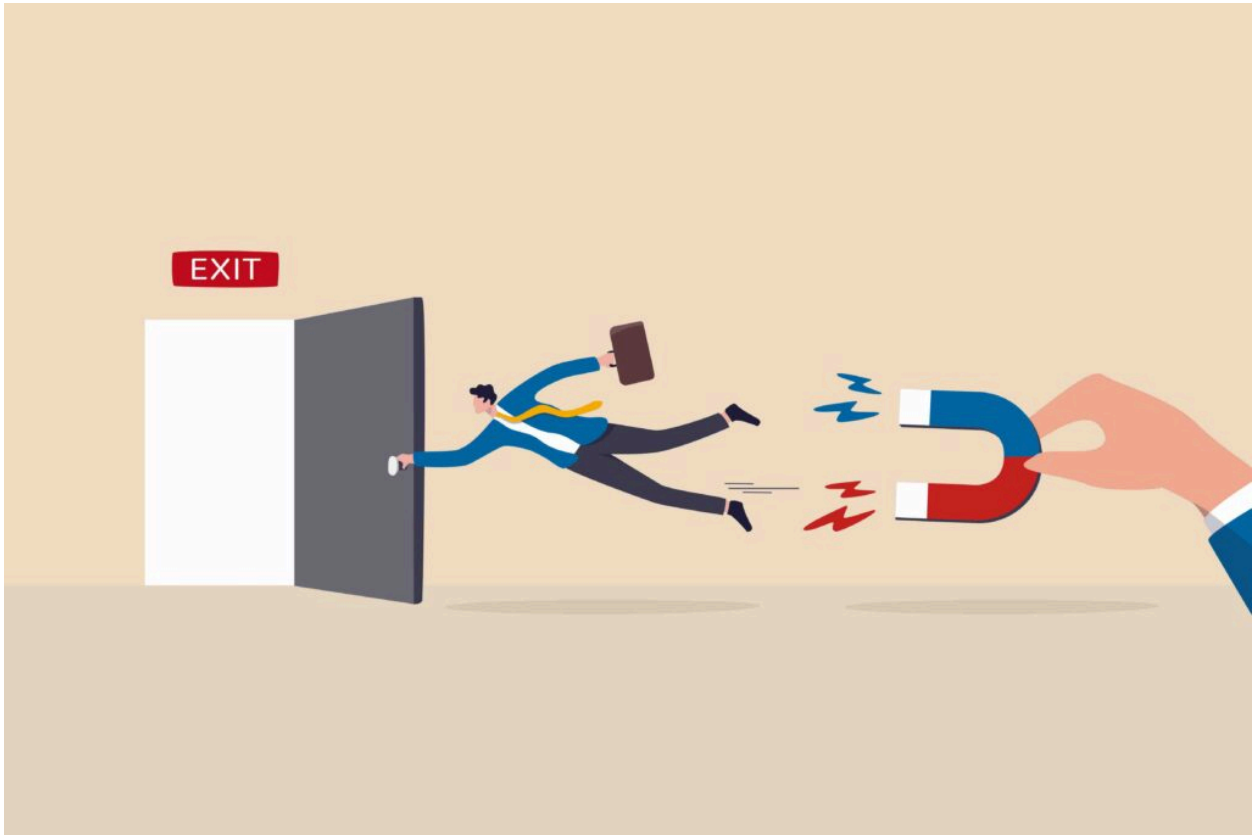
Microsoft Data Engineering

Project Documentation

DEPI



Customer Data Management and Analysis

WEEK 1

SQL Queries for Data Extraction, Update, and Analysis

1. Introduction

This document provides an overview and instructions for executing SQL queries aimed at extracting, updating, and analyzing customer data. The queries focus on customer information, revenue metrics, churn status, and customer behavior patterns. The task is divided into the following categories:

1- Data Extraction Queries

SQL queries used to retrieve important customer data from the database.

a. Extract all customer information:

This query pulls all relevant details for each customer.

b. Extract the top 10 customers with the highest total revenue:

This query identifies the customers who generate the most revenue.

c. Extract the average revenue per user (ARPU) for each region:

This query calculates the average revenue for customers in each geographic region.

```sql
SELECT * FROM telecom_customer_churn;

SELECT TOP 10 customer_id, total_revenue
FROM telecom_customer_churn
ORDER BY total_revenue DESC;

SELECT City, AVG(total_revenue) AS ARPU
FROM telecom_customer_churn
GROUP BY City;
```

2- Data Update Queries

SQL queries for updating the existing data to reflect changes in customer status or revenue.

a. Update the churn status of a customer:

This query updates the churn status for a specific customer based on their behavior.

b. Update the total revenue of a customer:

This query updates the revenue value for a customer based on new transactions.

```
UPDATE telecom_customer_churn
SET Married = '0'
WHERE customer_id = '0003-MKNFE';

UPDATE telecom_customer_churn
SET total_revenue = 1000
WHERE customer_id ='0002-ORFBO';
```

3- Data Analysis Queries

SQL queries that analyze customer behavior, revenue, and other important metrics.

a. Analyze the churn rate by region:

This query examines the percentage of churned customers in each region.

b. Analyze the average revenue per user (ARPU) by plan type:

This query calculates the ARPU based on the customers' subscription plans.

c. Analyze the correlation between total revenue and churn status:

This query helps understand if customers with higher or lower revenue are more likely to churn.

```sql
SELECT City, COUNT(CASE WHEN Customer_Status = 'Yes' THEN 1 END) AS Churned_Customers,
       COUNT(CASE WHEN Customer_Status = 'No' THEN 1 END) AS Retained_Customers,
       COUNT(*) AS Total_Customers
FROM telecom_customer_churn
GROUP BY City;

SELECT Phone_Service, AVG(total_revenue) AS ARPU
FROM telecom_customer_churn
GROUP BY Phone_Service;

SELECT Churn_Category, AVG(total_revenue) AS Average_Revenue
FROM telecom_customer_churn
GROUP BY Churn_Category;
```

4- Bake Analysis Queries

More complex queries that combine different data points for deeper insights.

a. Analyze the top 10 customers with the highest total revenue and churn status:

This query identifies the top revenue-generating customers and checks if they have churned.

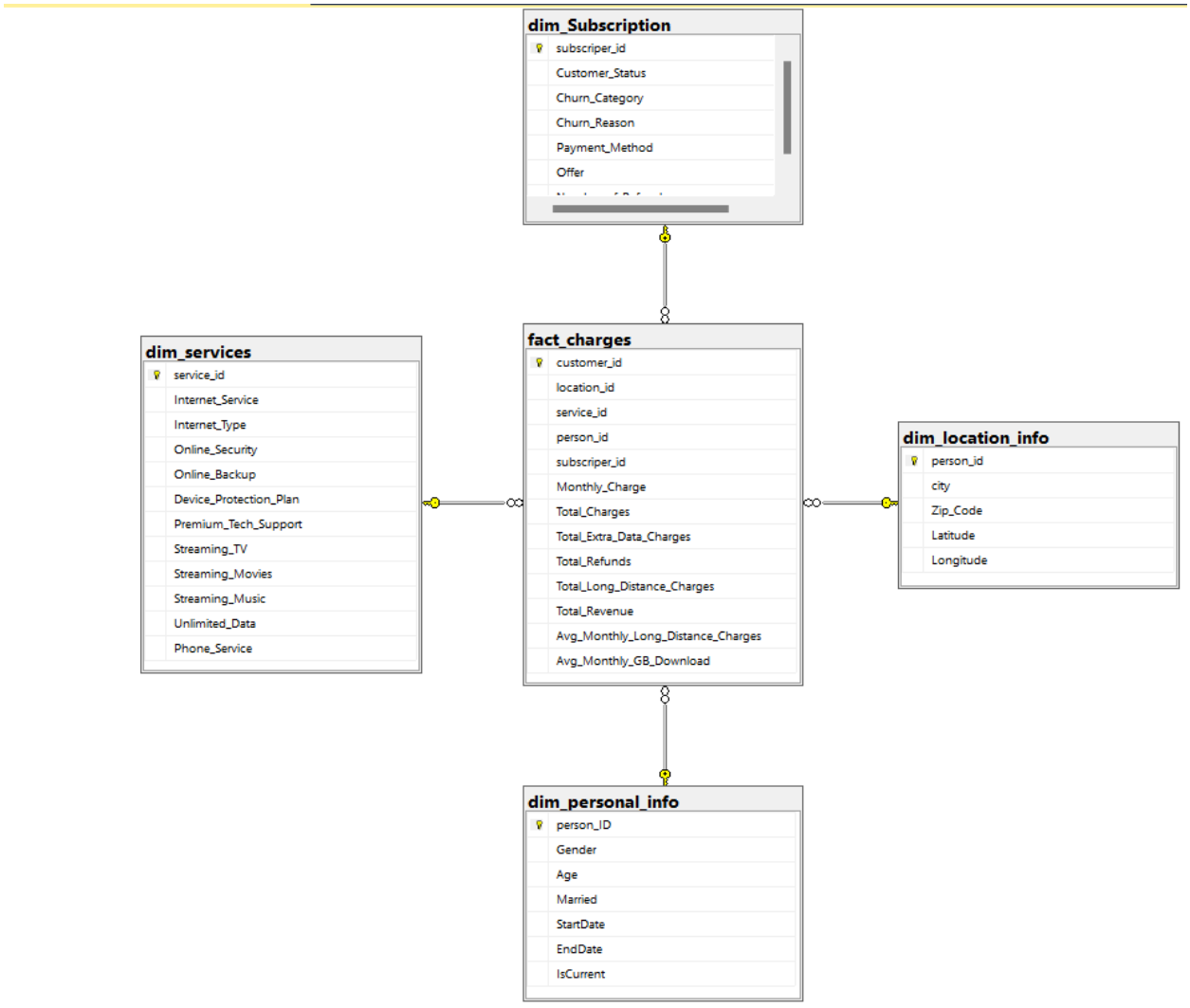
b. Analyze the average revenue per user (ARPU) by region and churn status:

This query provides ARPU details across regions, segmented by whether the customers have churned or not.

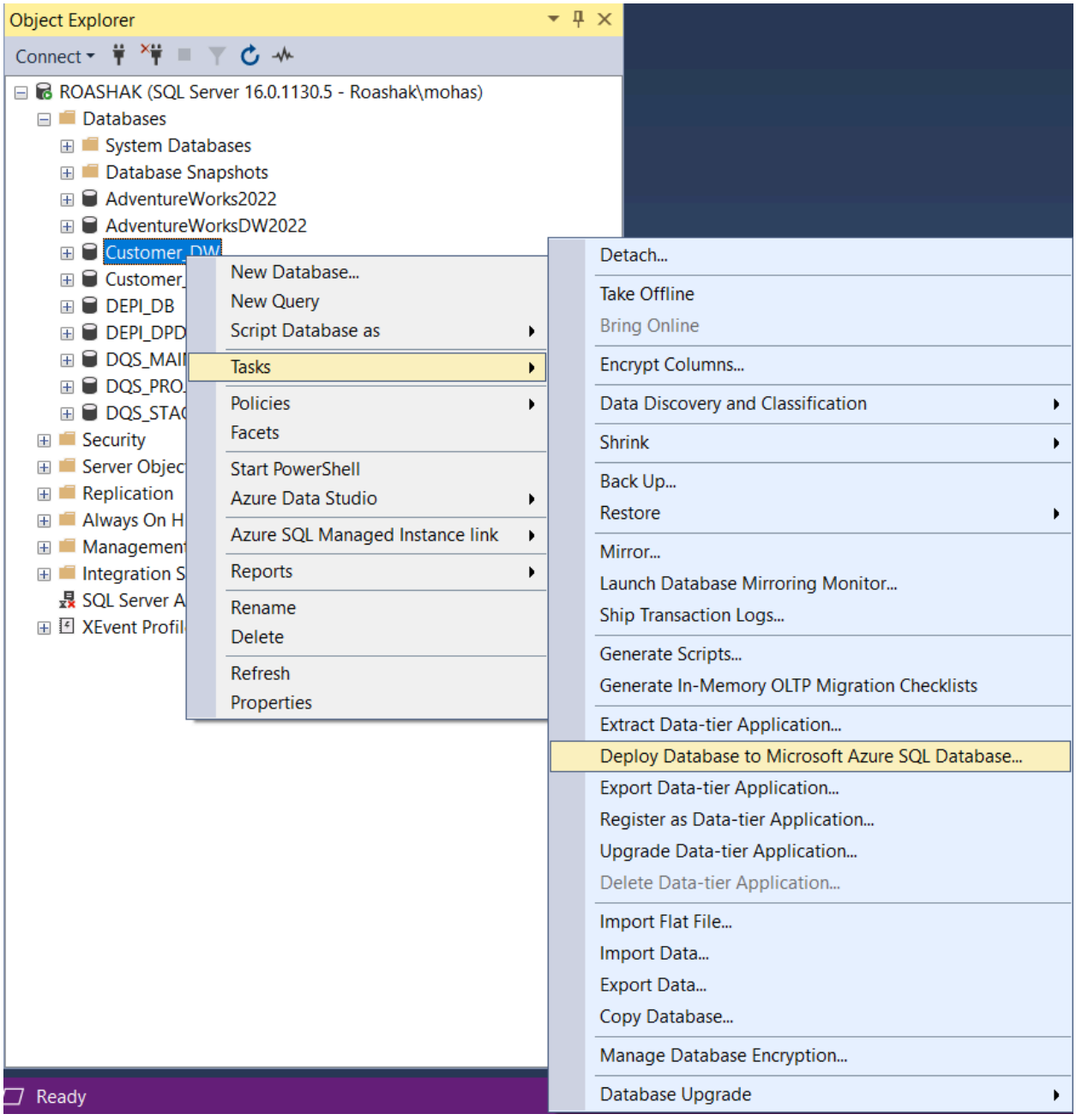
```
SELECT TOP 10 customer_id, total_revenue, Churn_Category
FROM telecom_customer_churn
ORDER BY total_revenue DESC;

SELECT City, Churn_Category, AVG(total_revenue) AS ARPU
FROM telecom_customer_churn
GROUP BY City, Churn_Category;
```

WEEK 2

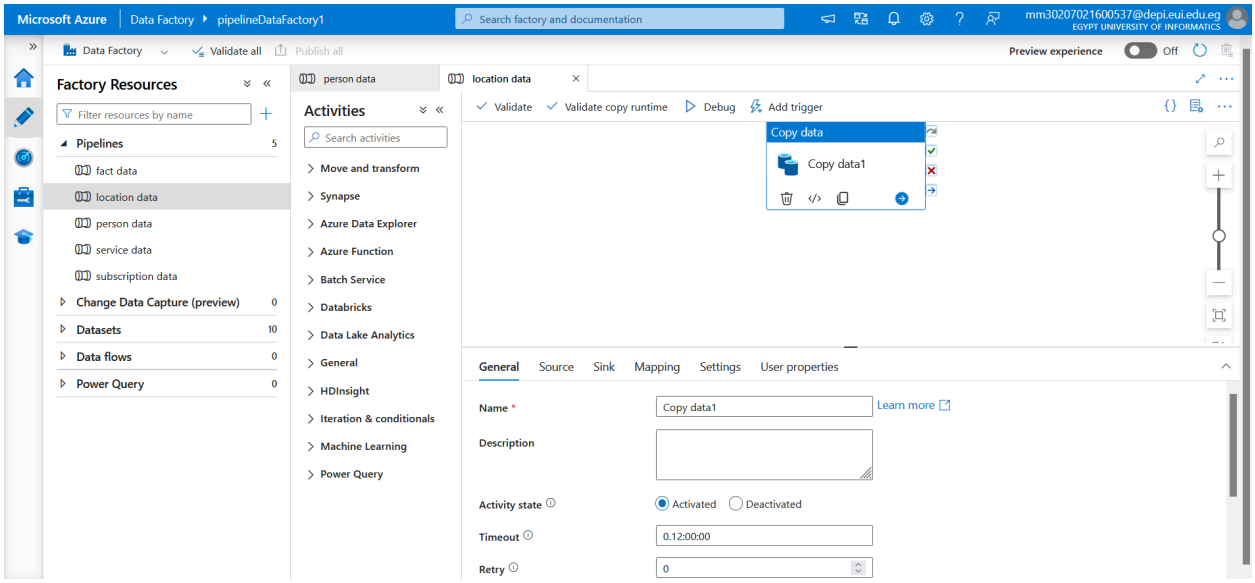
1- designing the schema of the data warehouse and make column store index for faster analysis



2- deploying the data source and the data warehouse on azure by using the moving task in SQL server management system:



3- using azure data factory to transfer data to the data warehouse



WEEK 3

Data Preprocessing

Before applying machine learning models, the dataset was cleaned and prepared for analysis. The following steps were implemented:

1. **Handling Missing Data**: Missing values were handled by either removing irrelevant records or imputing missing values using statistical methods like mean or median imputation.
2. **Categorical Data Encoding**: Categorical columns like `Gender` and `Churn_Reason` were converted into numeric formats using **one-hot encoding** to make them suitable for machine learning algorithms.
3. **Feature Scaling**: Numerical features such as `Monthly_Charge`, `Total_Charges`, and `Age` were standardized using **StandardScaler** to ensure that all features have a uniform scale.

Model Selection & Evaluation

Multiple machine learning algorithms were employed to predict customer churn. Models such as **Random Forest**, **Gradient Boosting** were explored. To deal with class imbalance in the dataset. followed by **train-test splitting** and **cross-validation** to ensure reliable performance estimation.

a. Random Forest

Random Forest is an ensemble method that reduces variance by averaging multiple decision trees. It is particularly effective when dealing with structured data like customer churn prediction.

b. Gradient Boosting

Gradient Boosting is a sequential ensemble method that focuses on correcting errors made by previous models. It generally offers strong predictive performance, especially for imbalanced datasets.

4. Performance Evaluation

Model performance was evaluated using metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score**.

WEEK 4

This project focuses on predicting customer churn using logistic regression, leveraging Azure Machine Learning Studio. The main goal is to integrate MLOps practices into the machine learning pipeline, enabling continuous integration, version control, and model management within the Azure ecosystem.

Architecture Overview

The workflow for the project consists of the following key steps:

1. **Data Loading**: Load data from an SQL database using PyODBC.
2. **Data Preprocessing**: Clean and preprocess the dataset for model training.
3. **Model Training**: Train a Logistic Regression model using scikit-learn.
4. **Model Evaluation**: Calculate the model's accuracy.
5. **MLOps Integration**: Log metrics and model to MLflow, and register the model in Azure Machine Learning Studio.

We begin by connecting to your Azure ML workspace, which acts as the environment for managing experiments, models, and data.

```
from azureml.core import Workspace, Experiment

# Connect to your workspace
ws = Workspace.from_config()

# Create an experiment in the workspace
experiment = Experiment(workspace=ws, name='customer-churn-prediction')
```

The customer churn data is loaded from an SQL Server using the PyODBC library.

```
import pandas as pd
import pyodbc

# Connection details for SQL Server
conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=group1server.database.windows.net;DATABASE=DEPI_DB;UID=group

# Load data into DataFrame
query = "SELECT * FROM [dbo].[telecom_customer_churn];"
df = pd.read_sql(query, conn)
conn.close()
```

The data is cleaned by handling missing values and transforming the `Customer_Status` column to a binary target variable for churn prediction. Additionally, categorical columns are one-hot encoded.

```
# Data preprocessing
df = df.dropna()
df['Churn'] = df['Customer_Status'].apply(lambda x: 1 if x == 'Churned' else 0)

# Include actual categorical columns
categorical_columns = ['Gender', 'City', 'Offer', 'Phone_Service', 'Multiple_Lines', 'Internet_Service', 'Internet_Type', 'Online

# One-hot encode categorical columns
df = pd.get_dummies(df, columns=categorical_columns, drop_first=True)
df = df.drop(columns=['Customer_ID', 'Customer_Status'])
```

A Logistic Regression model is trained using the processed dataset. The dataset is split into training and testing sets, and the model is fitted to the training data.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X = df.drop(columns=['Churn'])
y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy}")
```

MLflow is used to log model metrics and the trained logistic regression model. The model is also signed with its input and output signatures.

```
import mlflow
import mlflow.sklearn
from mlflow.models.signature import infer_signature

mlflow.start_run()
mlflow.log_metric("accuracy", accuracy)
signature = infer_signature(X_train, model.predict(X_train))
mlflow.sklearn.log_model(model, "logistic_regression_model", signature=signature, input_example=X_train[:5])
mlflow.end_run()
```

Once the model is trained and logged, it is saved and registered in Azure Machine Learning Studio. This ensures version control and easy deployment in production.

```
from azureml.core import Workspace, Model
import joblib

# Connect to your workspace
ws = Workspace.from_config()

# Save the model
joblib.dump(model, 'logistic_regression_model.pkl')

# Register the model
model = Model.register(
    workspace=ws,
    model_name="customer_churn_model",
    model_path="logistic_regression_model.pkl",
    description="Logistic Regression model for customer churn prediction"
)
```