Digital Egypt Pioneers

CUSTOMER DATA MANAGEMENT AND ANALYSIS

TEAM MEMBERS

- Mahmoud Sharabas
- Radwa Amr
- Youssif Qazamel
- Omar Ashraf
- Moahmmed Abdelhalim

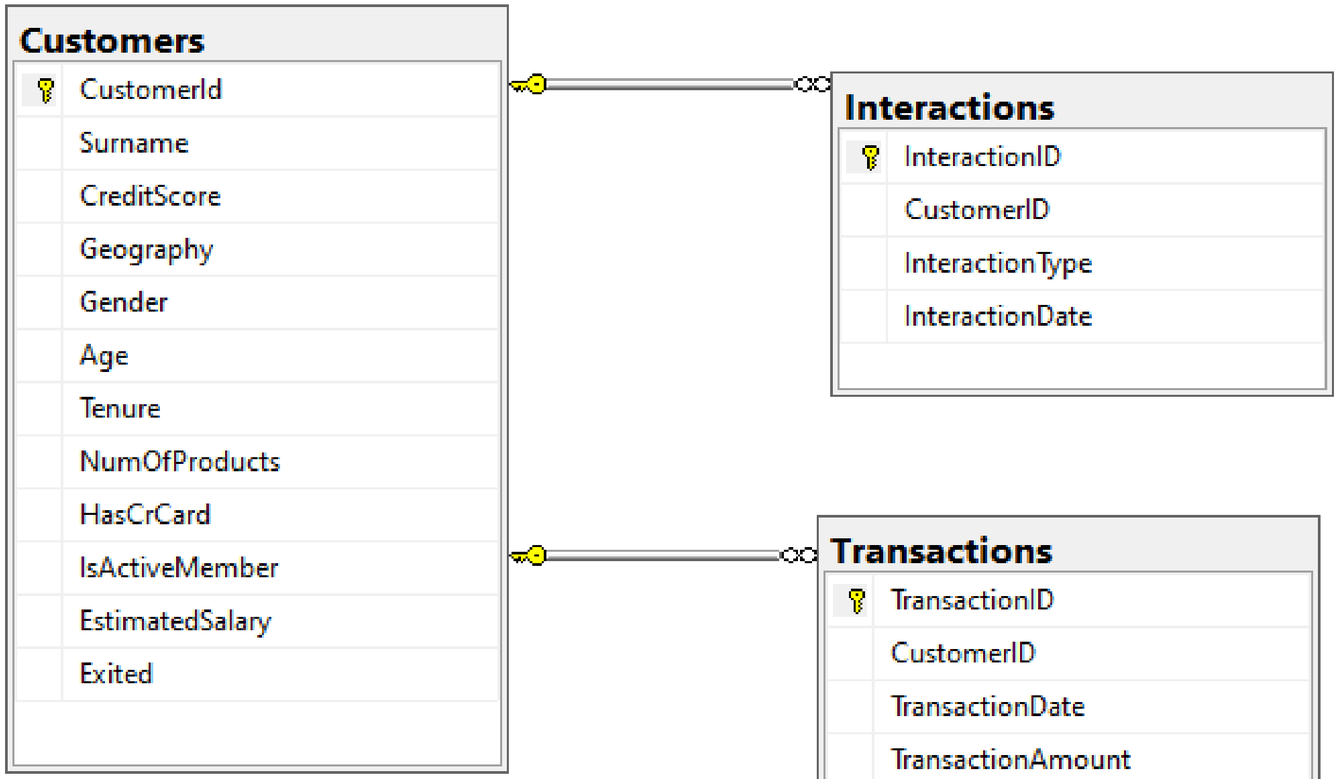
Supervision

- Dr Tamer Emara

Digital Egypt Pioneers

WEEK 1: DATA MANAGEMENT AND SQL DATABASE SETUP

1.Database Design



Design a SQL database schema to manage customer data, including tables for customer information, transactions, and interactions.

2.Implementation

Create and populate the SQL database using Microsoft SQL Server

```sql
CREATE DATABASE CustomerDB;
USE CustomerDB;


CREATE TABLE Customers (
    CustomerId INT PRIMARY KEY,
    Surname VARCHAR(50),
    CreditScore int,
    Geography VARCHAR(50),
    Gender VARCHAR(10),
    Age int,
    Tenure int,
    NumOfProducts int,
    HasCrCard int,
    IsActiveMember int,
    EstimatedSalary decimal(10,2),
    Exited int
);
```

```sql
ALTER TABLE Customers
ALTER COLUMN Age VARCHAR(10);

ALTER TABLE Customers
ALTER COLUMN HasCrCard VARCHAR(10);


ALTER TABLE Customers
ALTER COLUMN IsActiveMember VARCHAR(10);


CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
    TransactionDate DATE,
    TransactionAmount DECIMAL(10, 2)
    TransactionType VARCHAR(50),
    Balance DECIMAL(10, 2),
);
```

```sql
CREATE TABLE Interactions (
    InteractionID INT PRIMARY KEY,
    CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
    InteractionType VARCHAR(50),
    InteractionDate DATE,
);
```

2.Implementation

Create and populate the SQL database using Microsoft SQL Server

```
use CustomerDB
go

BULK INSERT Customers
FROM 'D:\Rewad\Project\dataset\Customer.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);


BULK INSERT transactions
FROM 'D:\Rewad\Project\dataset\Transcations.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);
```

```
BULK INSERT Interactions
FROM 'D:\Rewad\Project\dataset\Interactions.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);
```

3.SQL Queries

Write SQL queries to extract, update, and analyze customer data.

```sql
USE CustomerDB;
--SQL Queries for Data Extraction

--Retrieve all customer information
SELECT * FROM Customers;


--Find all transactions made by a specific customer
SELECT *
FROM Transactions
WHERE CustomerID= 15569120;


--Get all interactions of a specific customer
SELECT *
FROM Interactions
WHERE CustomerID= 15569120;
```

```sql
--Find the total amount spent by each customer
SELECT CustomerID, SUM(TransactionAmount) AS TotalAmount
FROM Transactions
GROUP BY CustomerID;


--List the top 5 customers who spent the most
SELECT TOP 5 C.Surname,SUM(T.TransactionAmount) AS TotalAmountSpent
FROM Customers C
JOIN Transactions T
ON C.CustomerId=T.CustomerID
GROUP BY C.Surname
ORDER BY TotalAmountSpent DESC;
```

3.SQL Queries

Write SQL queries to extract, update, and analyze customer data.

```sql
--Get customers who have had interactions but haven't made any transactions
SELECT C.Surname
FROM Customers C
LEFT JOIN Transactions T
ON C.CustomerID=T.CustomerID
LEFT JOIN Interactions I
ON C.CustomerId = I.CustomerID
WHERE T.TransactionID IS NULL AND I.CustomerID IS NOT NULL;
```

```sql
--SQL Queries for Basic Analysis

--Find the number of transactions per customer
SELECT CustomerID,COUNT(*) AS NumOfTransactions
FROM Transactions
GROUP BY CustomerID;

--Calculate the average transaction amount per customer
SELECT CustomerID , AVG(TransactionAmount) AS AvgTransactionAmount
FROM Transactions
GROUP BY CustomerID;

--Get the most popular transaction type by the number of transactions
SELECT TOP 1 TransactionType , COUNT(*) AS TransactionCount
FROM Transactions
GROUP BY TransactionType
ORDER BY TransactionCount DESC;
```

3.SQL Queries

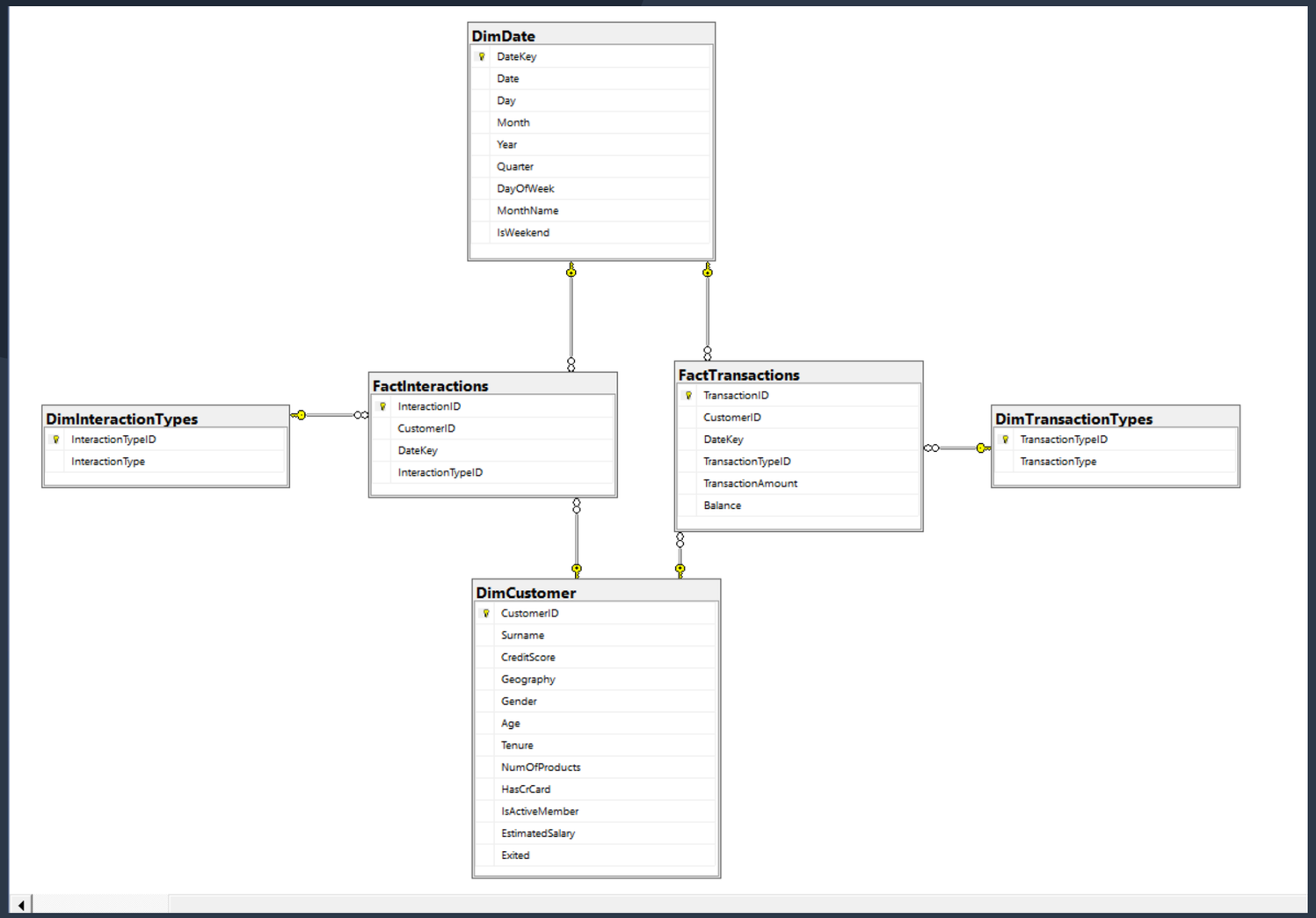
Write SQL queries to extract, update, and analyze customer data.

```sql
--Get the total number of interactions per customer
SELECT CustomerID,COUNT(*) AS InteractionCount
FROM Interactions
GROUP BY CustomerID;

--Find all transactions and corresponding customer information
SELECT C.CustomerId,C.Surname,T.TransactionType,T.TransactionAmount,T.TransactionDate
FROM Transactions T
JOIN Customers C
ON T.CustomerID=C.CustomerId;


--List all interactions along with customer details
SELECT C.CustomerId,C.Surname,I.InteractionType,I.InteractionDate
FROM Customers C
JOIN Interactions I
ON C.CustomerId=I.CustomerID;
```

```sql
--Get a summary report with the total amount spent and number of interactions for each customer
SELECT I.CustomerID, SUM(T.TransactionAmount) AS TotalAmount , COUNT(InteractionID) AS NumOfInteractions
FROM Interactions I
JOIN Transactions T
ON I.CustomerID=T.CustomerID
GROUP BY I.CustomerID;
```

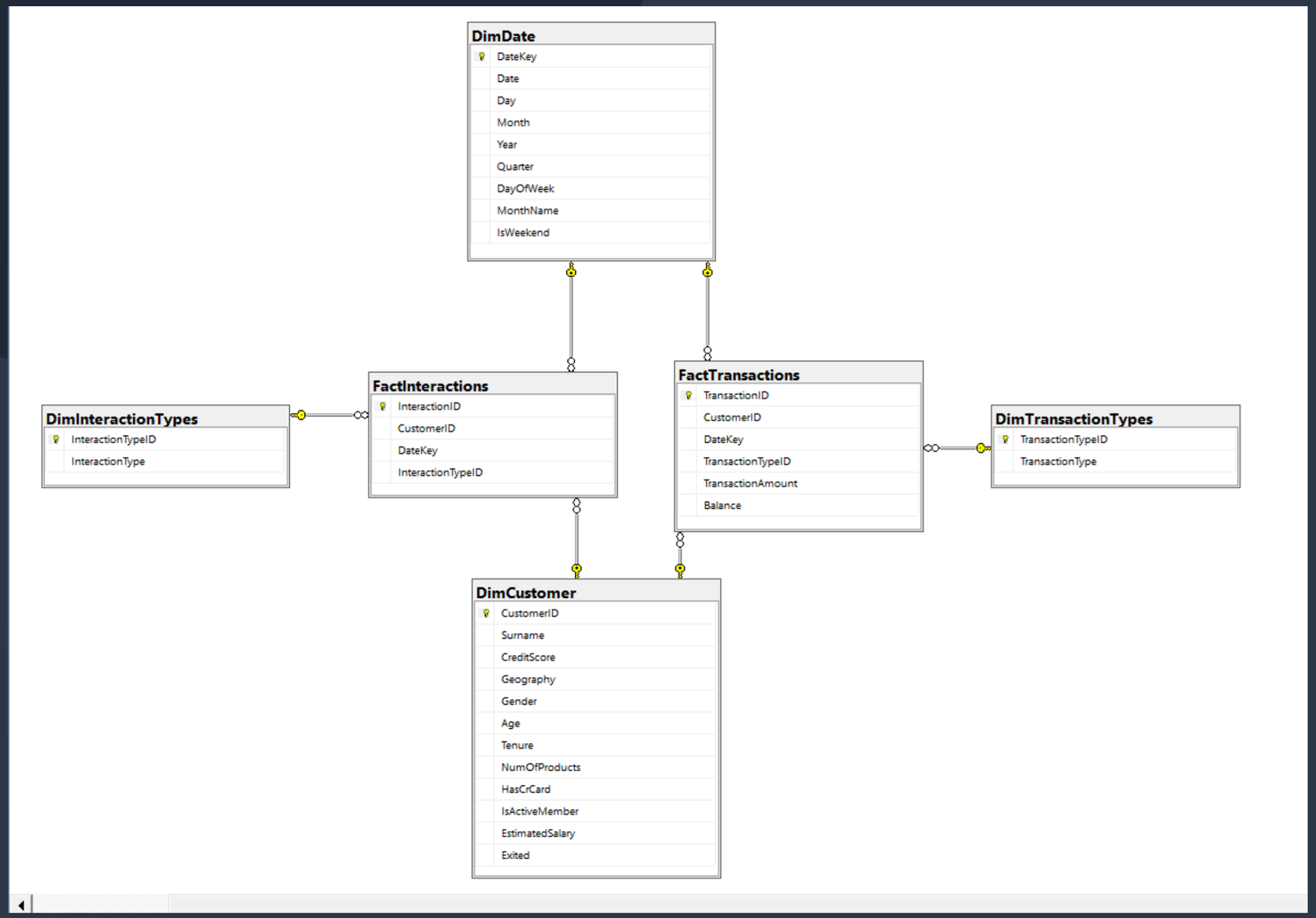
WEEK 2: DATA WAREHOUSING AND PYTHON PROGRAMMING

1.Data Warehouse Implementation



Implement a SQL Data Warehouse to aggregate and manage large volumes of customer data for analytical purposes

1.Data Warehouse Implementation



Implement a SQL Data Warehouse to aggregate and manage large volumes of customer data for analytical purposes

2.Data Integration

```
-- Insert unique transaction types with a unique ID
INSERT INTO CustomerDW.dbo.DimTransactionTypes (TransactionType)
SELECT DISTINCT TransactionType
FROM CustomerDB.dbo.Transactions;


-- Load unique interaction types
INSERT INTO CustomerDW.dbo.DimInteractionTypes (InteractionType)
SELECT DISTINCT InteractionType
FROM CustomerDB.dbo.Interactions;
```

Load data from various sources into the data warehouse

2.Data Integration

```sql
-- Insert into DimDate
INSERT INTO CustomerDW.dbo.DimDate(DateKey, Date, Day, Month, Year, Quarter, DayOfWeek, MonthName, IsWeekend)
SELECT
    CONVERT(INT, FORMAT(UniqueDate, 'yyyyMMdd')) AS DateKey, -- DateKey in YYYYMMDD format
    UniqueDate AS Date,                                      -- Actual date
    DAY(UniqueDate) AS Day,                                  -- Day of the month
    MONTH(UniqueDate) AS Month,                              -- Month number
    YEAR(UniqueDate) AS Year,                                -- Year
    DATEPART(QUARTER, UniqueDate) AS Quarter,                -- Quarter (1-4)
    DATENAME(WEEKDAY, UniqueDate) AS DayOfWeek,              -- Day name (e.g., Monday)
    DATENAME(MONTH, UniqueDate) AS MonthName,                -- Month name (e.g., January)
    CASE
        WHEN DATENAME(WEEKDAY, UniqueDate) IN ('Saturday', 'Sunday') THEN 1
        ELSE 0
    END AS IsWeekend                                         -- IsWeekend flag (1 = weekend, 0 = not weekend)
FROM
    (   SELECT DISTINCT CAST(TransactionDate AS DATE) AS UniqueDate
        FROM CustomerDB.dbo.Transactions

        UNION

        SELECT DISTINCT CAST(InteractionDate AS DATE) AS UniqueDate
        FROM CustomerDB.dbo.Interactions) AS DateList;       -- Ensure unique dates are inserted
```

Load data from various sources into the data warehouse

2.Data Integration

Load data from various sources into the data warehouse

```
-- Insert data into DimCustomer
INSERT INTO CustomerDW.dbo.DimCustomer (CustomerID, Surname, CreditScore, Geography, Gender, Age, Tenure,
NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited)
SELECT
    CustomerID,
    Surname,
    CreditScore,
    Geography,
    Gender,
    CAST(Age AS INT),
    Tenure,
    NumOfProducts,
    CAST(HasCrCard AS INT),
    CAST(IsActiveMember AS INT),
    EstimatedSalary,
    Exited
FROM
    CustomerDB.dbo.Customers;
```

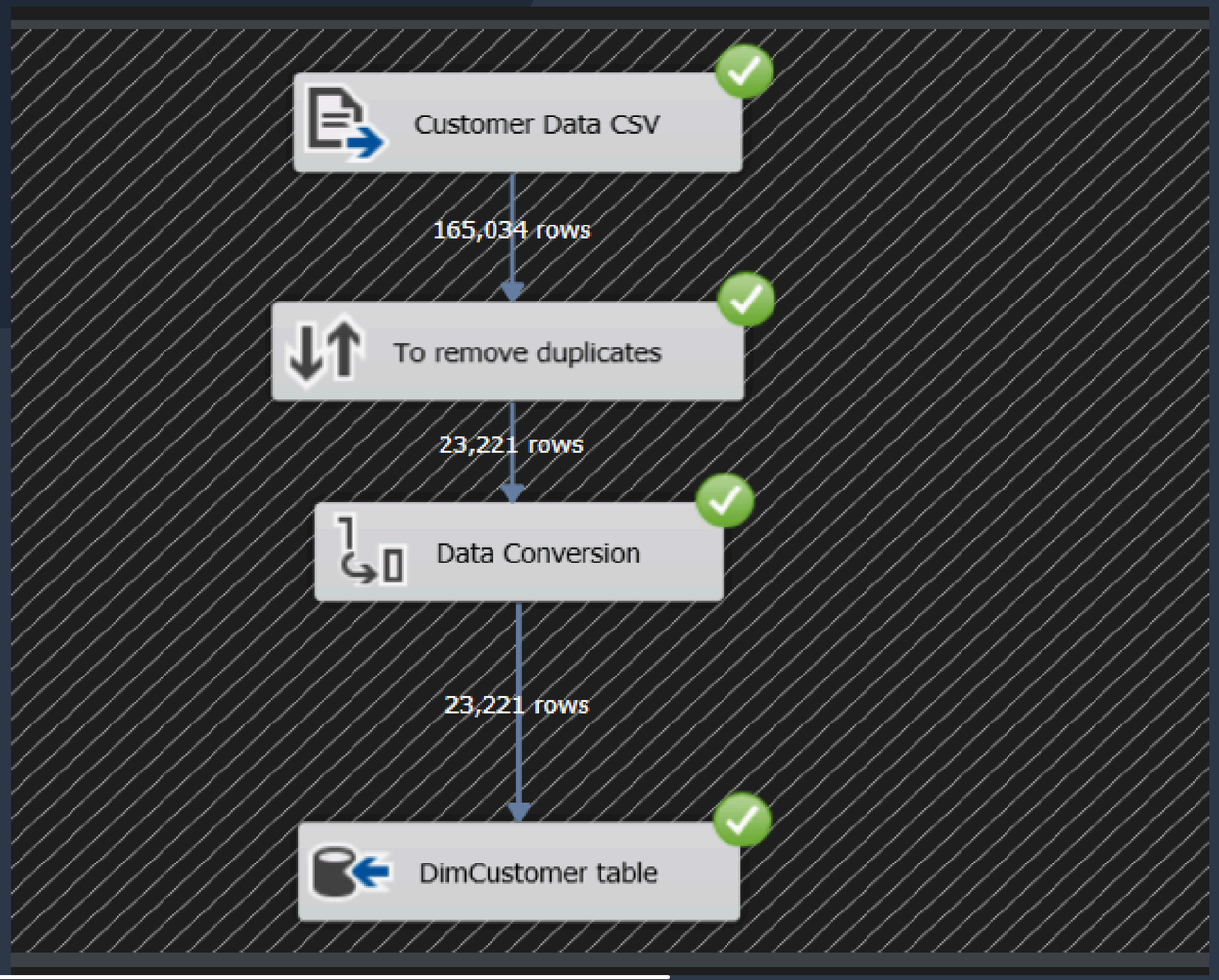
2.Data Integration

```sql
-- Insert data into FactTransactions
INSERT INTO CustomerDW.dbo.FactTransactions (TransactionID, CustomerID, DateKey, TransactionTypeID,
TransactionAmount, Balance)
SELECT t.TransactionID,
       t.CustomerID,
       CONVERT(INT, FORMAT(t.TransactionDate, 'yyyyMMdd')) AS DateKey,
       (SELECT dt.TransactionTypeID
        FROM CustomerDW.dbo.DimTransactionTypes dt
        WHERE dt.TransactionType = t.TransactionType) AS TransactionTypeID,
       t.TransactionAmount,
       t.Balance
FROM CustomerDB.dbo.Transactions t;



-- Insert data into FactInteractions
INSERT INTO CustomerDW.dbo.FactInteractions (InteractionID,CustomerID,DateKey,InteractionTypeID)
SELECT i.InteractionID,
       i.CustomerID,
       CONVERT(INT,FORMAT(i.InteractionDate, 'yyyyMMdd')) AS DateKey,
       (SELECT di.InteractionTypeID
       FROM CustomerDW.dbo.DimInteractionTypes di
       WHERE di.InteractionType=i.InteractionType) AS InteractionTypeID
FROM CustomerDB.dbo.Interactions i;
```

Load data from various sources into the data warehouse

2.Data Integration

SSIS

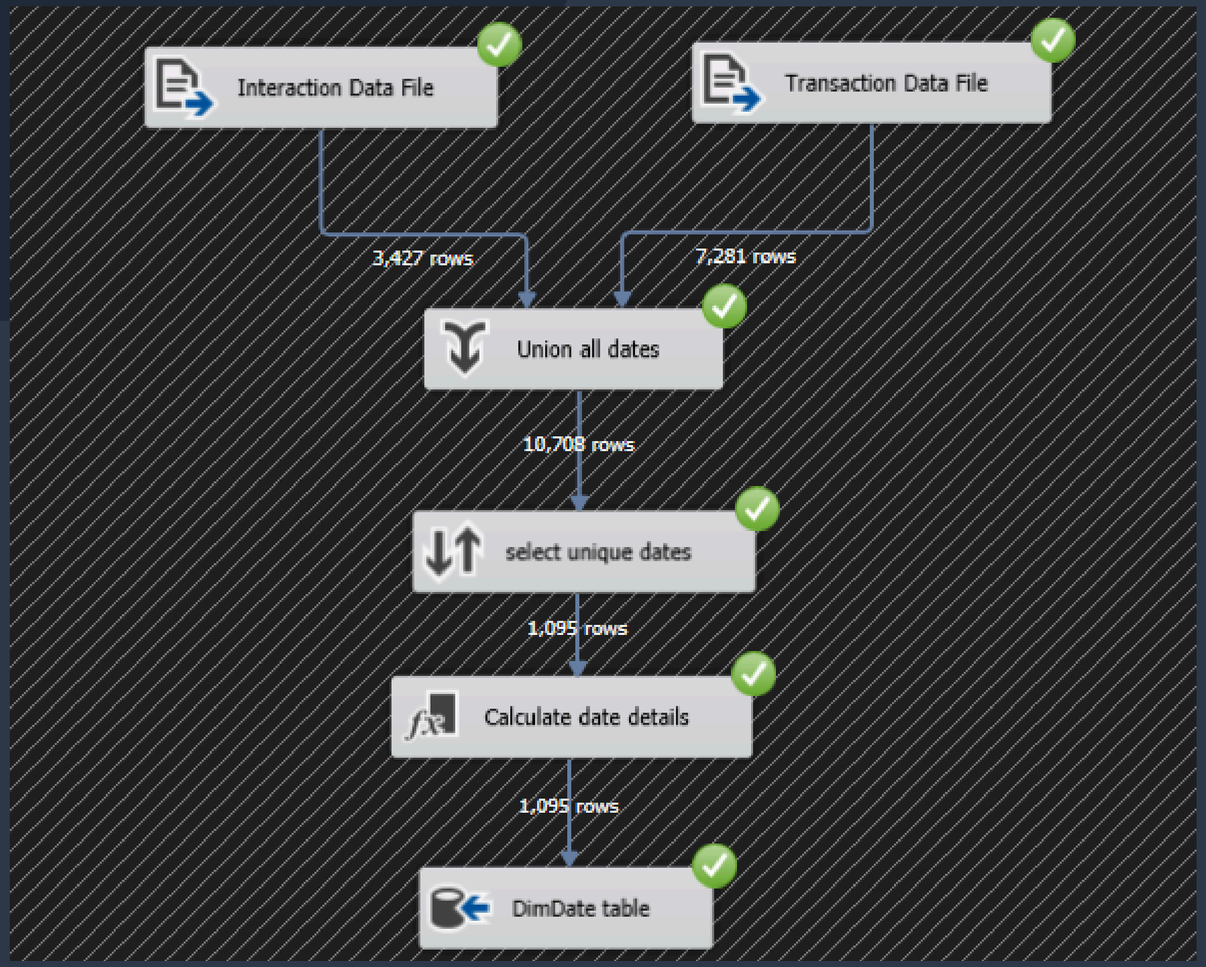


Load data from various sources into the data warehouse

DimCustRun

2.Data Integration

SSIS

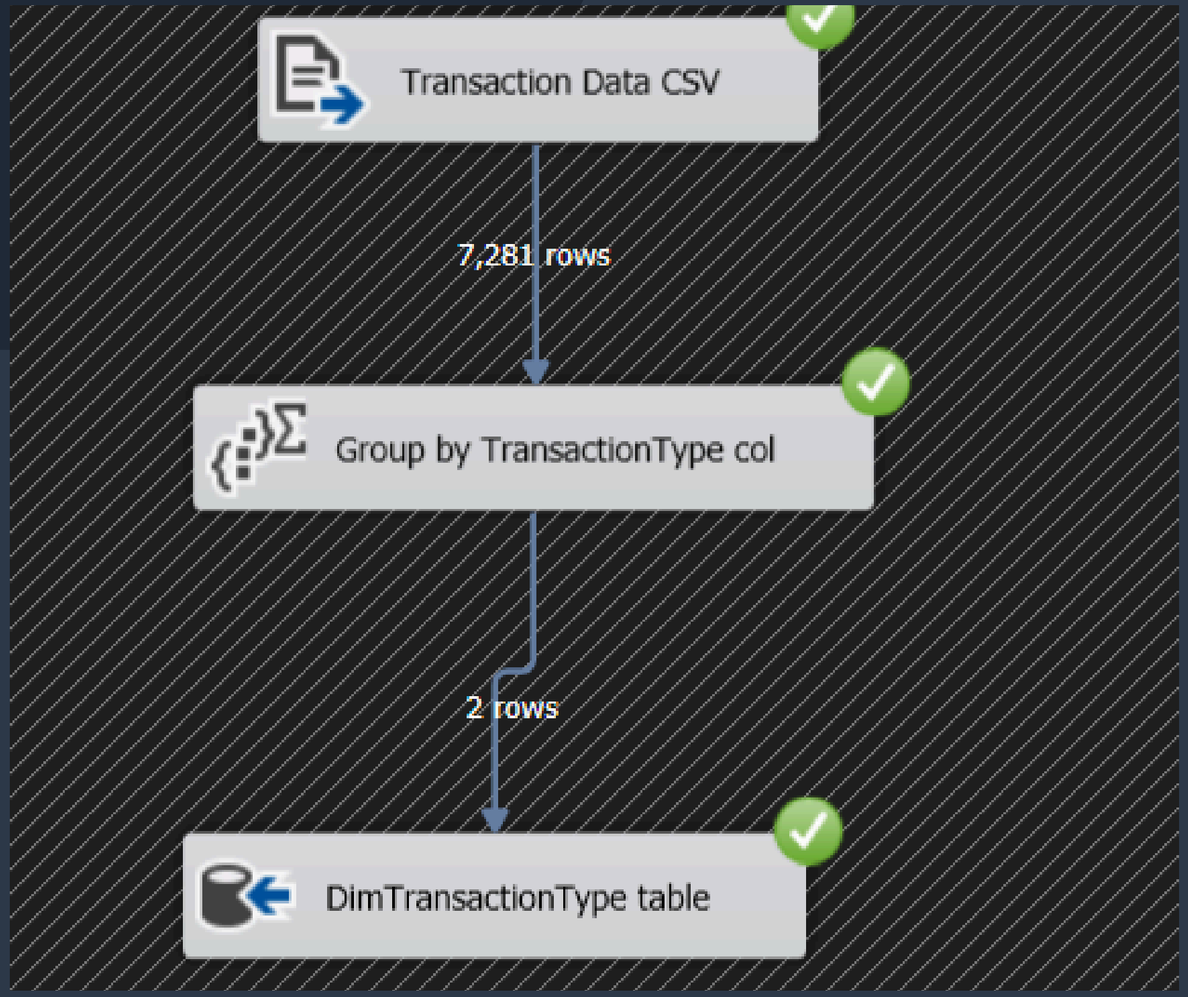


Load data from various sources into the data warehouse

DimDateRun

2.Data Integration

SSIS

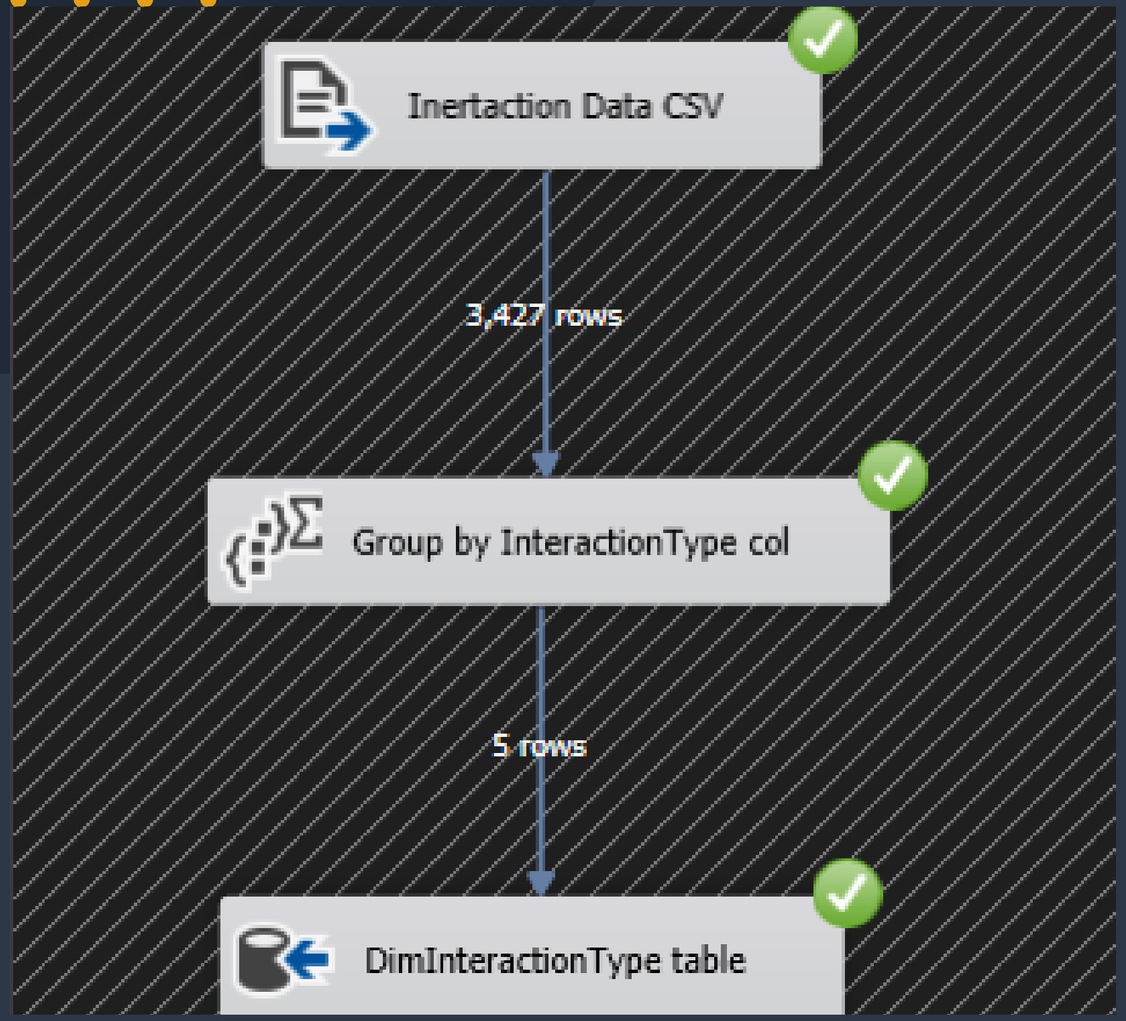


Load data from various sources into the data warehouse

DimTransRun

2.Data Integration

SSIS

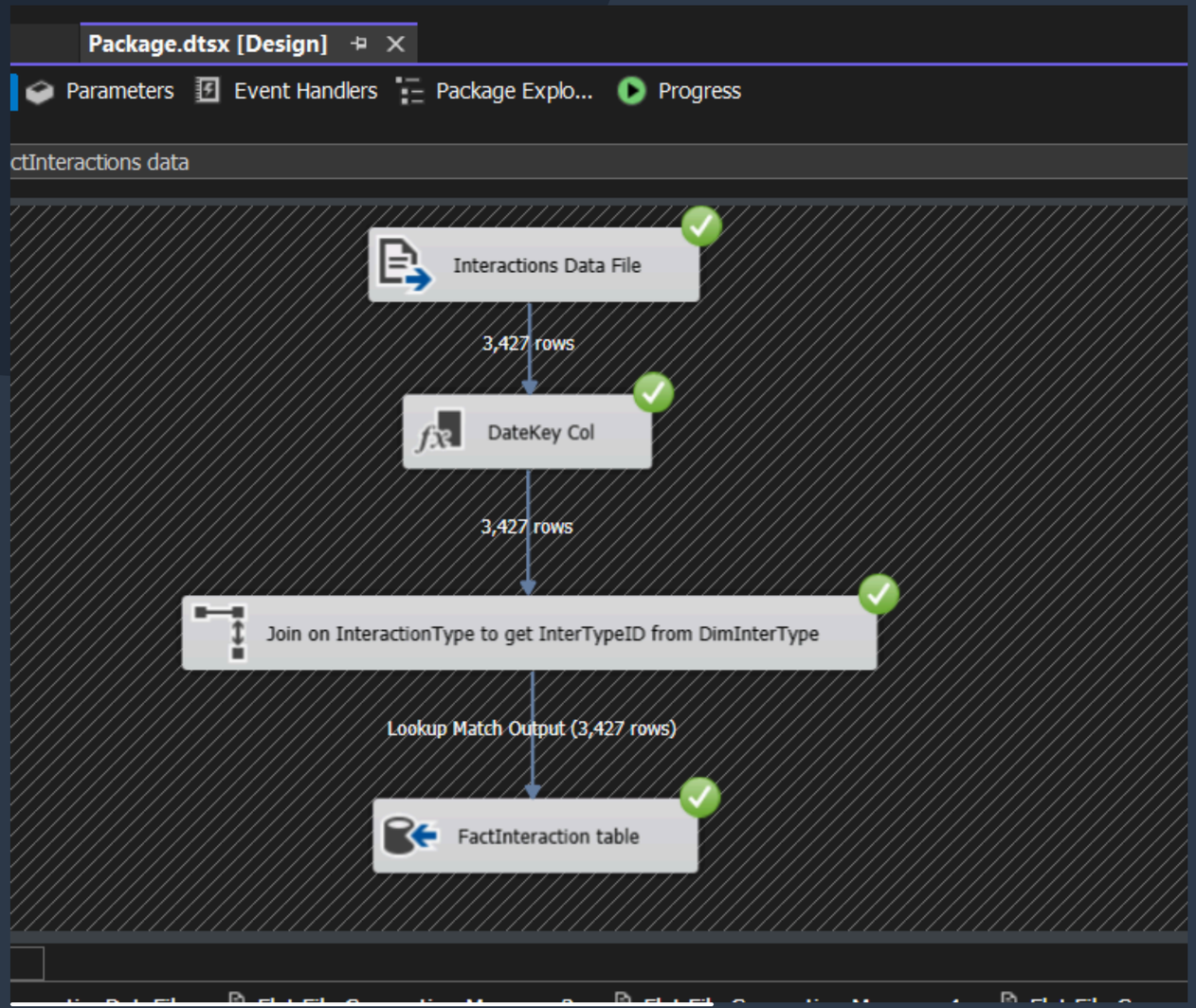


Load data from various sources into the data warehouse

DimInterRun

2.Data Integration

SSIS

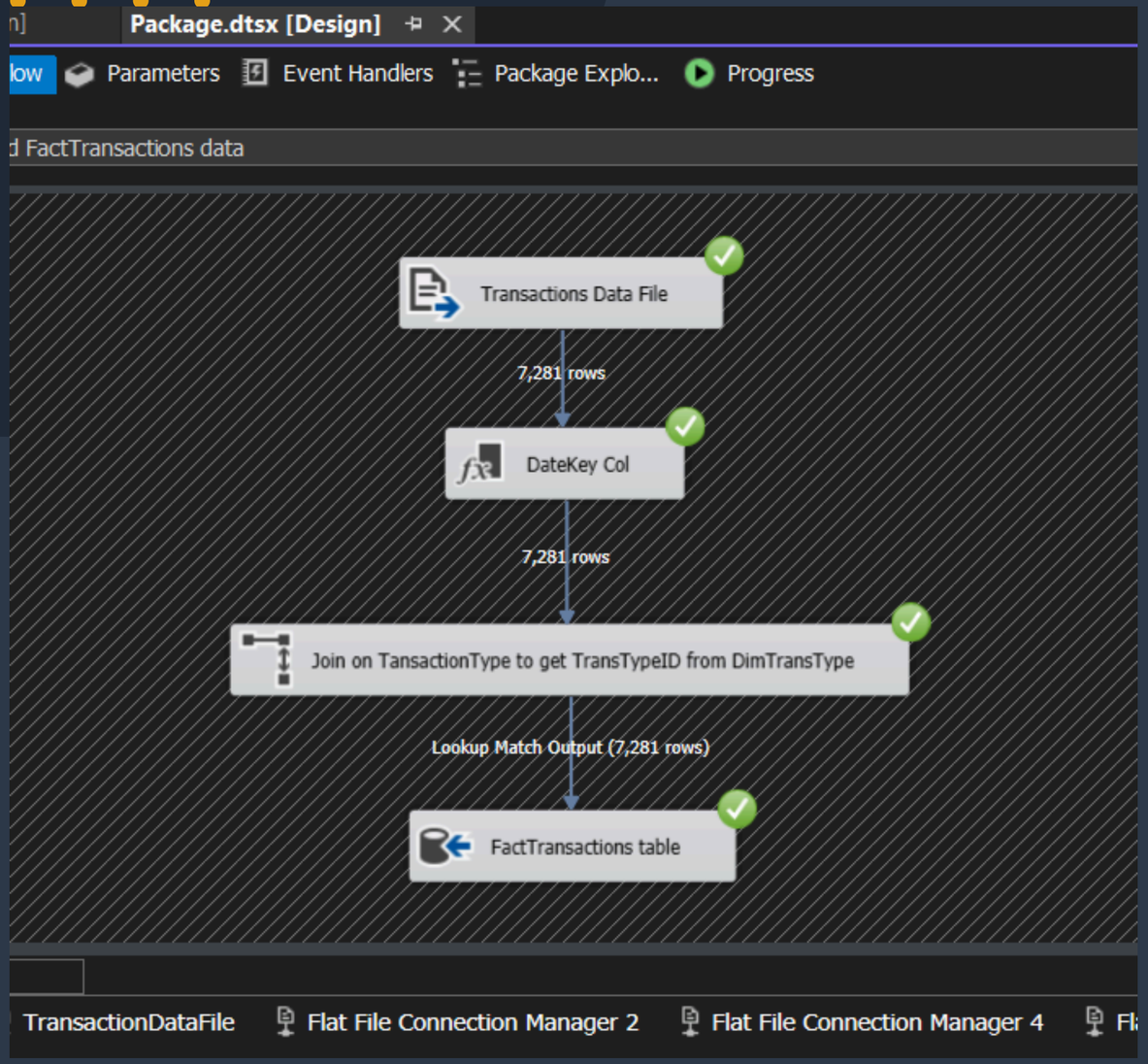


Load data from various sources into the data warehouse

FactInterRun

2.Data Integration

SSIS



Load data from various sources into the data warehouse

FactTransRun

3.Python Programming

```python
import pyodbc

conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};'
                      'SERVER=CAESAR\CAESAR;'
                      'DATABASE=CustomerDW;'
                      'UID=CAESAR\\shara')

cursor = conn.cursor()
cursor.execute('SELECT * FROM Customers')

for row in cursor:
    print(row)
```

Develop Python scripts to interact with the SQL database, perform data extraction, and prepare data for analysis

3.Python Programming

```
cursor = conn.cursor()
cursor.execute('SELECT * FROM Interactions')

for row in cursor:
    print(row)


cursor = conn.cursor()
cursor.execute('SELECT * FROM Transactions')

for row in cursor:
    print(row)

conn.close()
```

Develop Python scripts to interact with the SQL database, perform data extraction, and prepare data for analysis

WEEK 3: DATA SCIENCE AND AZURE INTEGRATION

Data Summary

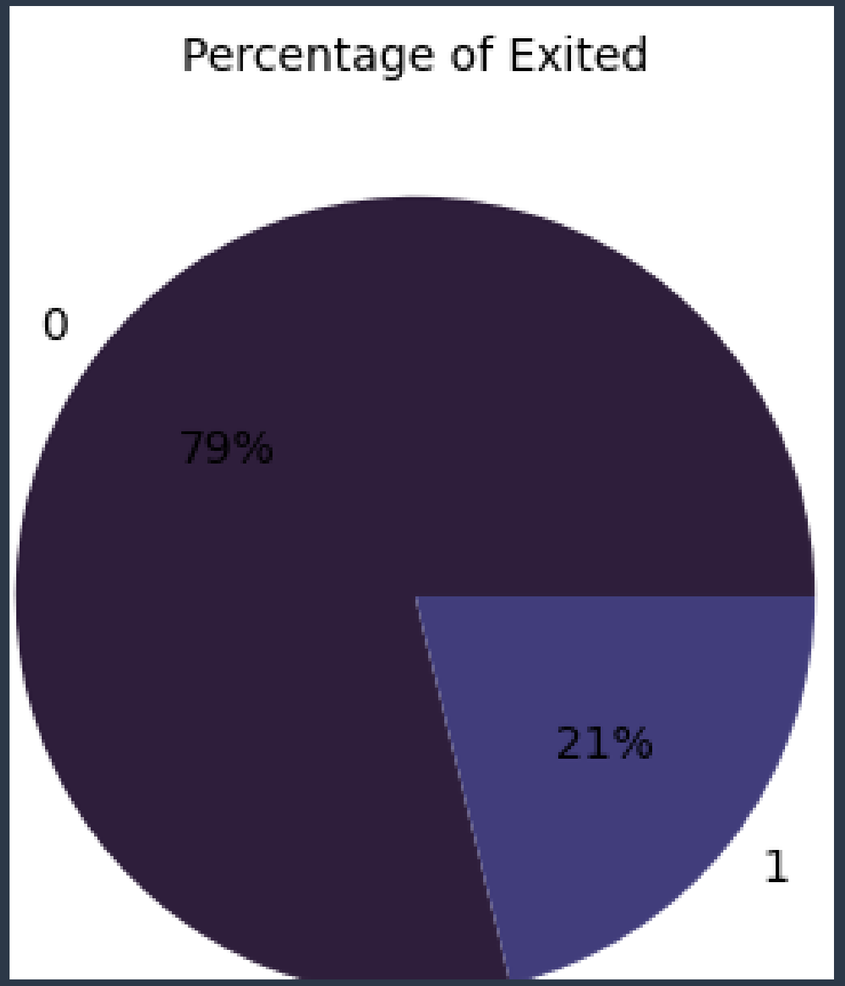
1.Data Science with Python

```
df_train.describe()
```

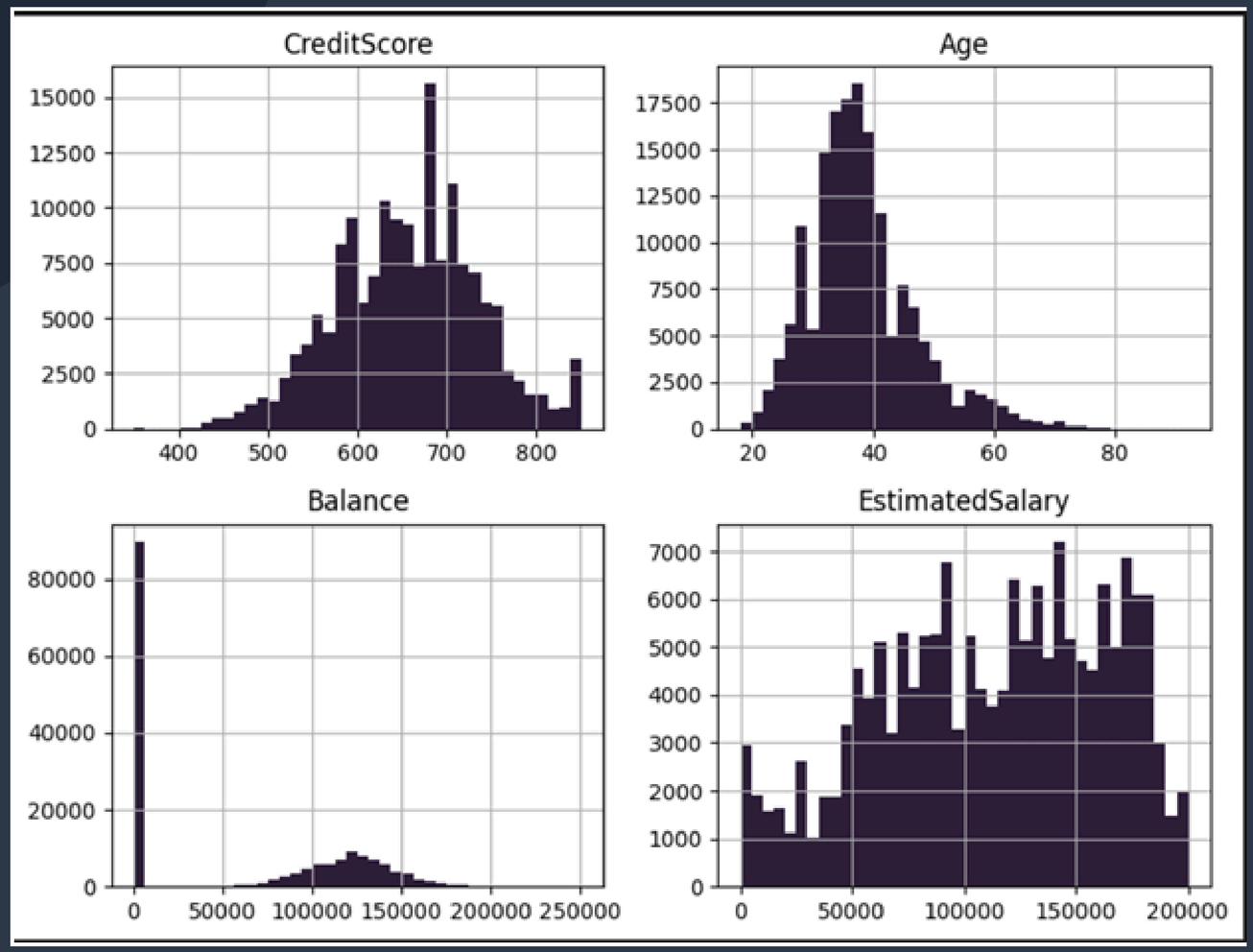
	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000
mean	656.454373	38.125888	5.020353	55478.086689	1.554455	0.753954	0.497770	112574.822734	0.211599
std	80.103340	8.867205	2.806159	62817.663278	0.547154	0.430707	0.499997	50292.865585	0.408443
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000	0.000000	11.580000	0.000000
25%	597.000000	32.000000	3.000000	0.000000	1.000000	1.000000	0.000000	74637.570000	0.000000
50%	659.000000	37.000000	5.000000	0.000000	2.000000	1.000000	0.000000	117948.000000	0.000000
75%	710.000000	42.000000	7.000000	119939.517500	2.000000	1.000000	1.000000	155152.467500	0.000000
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000	1.000000

1.Data Science with Python

Plot Target Column



Plot Numerical Columns

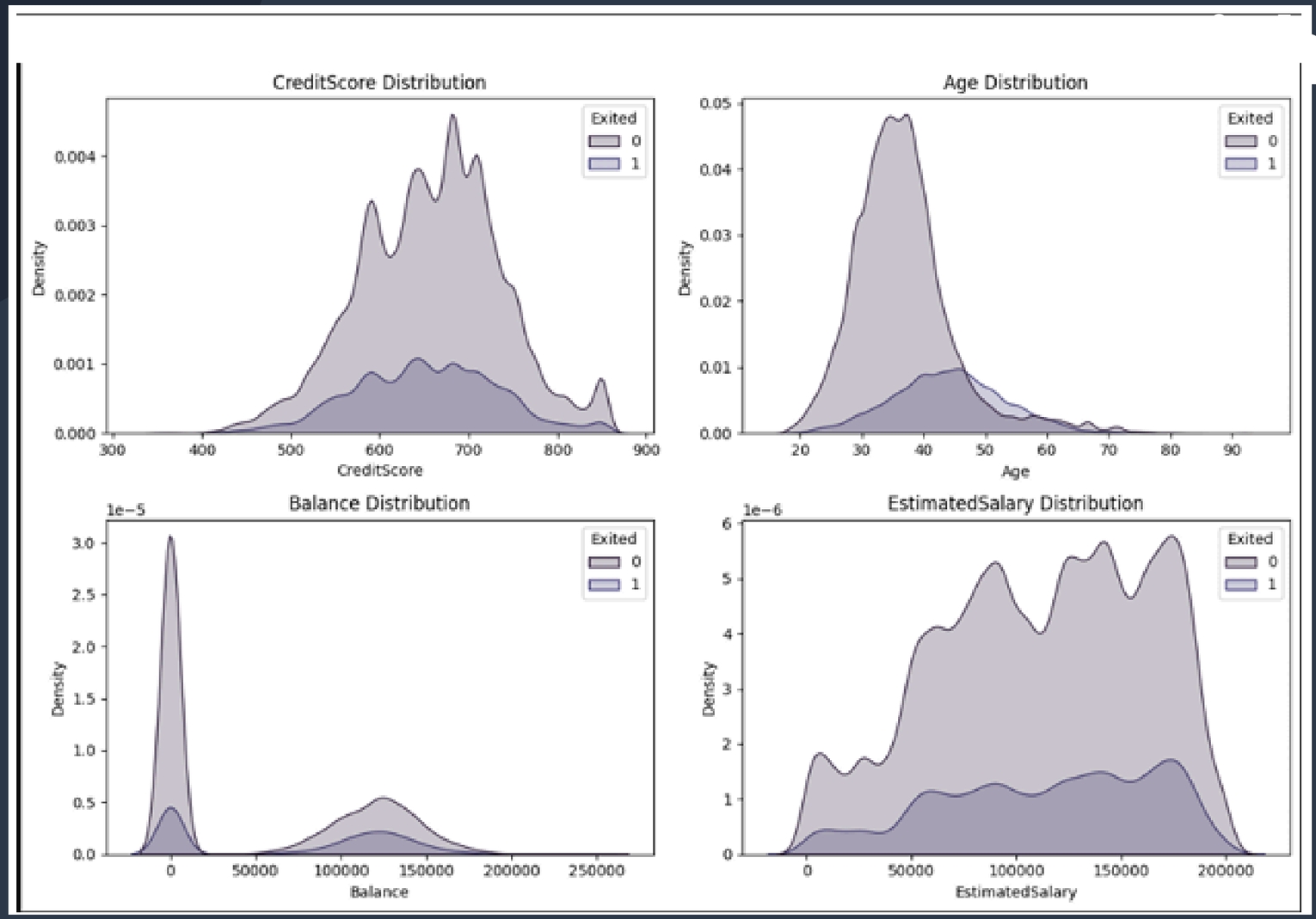


1.Data Science with Python

Plot Numerical Columns

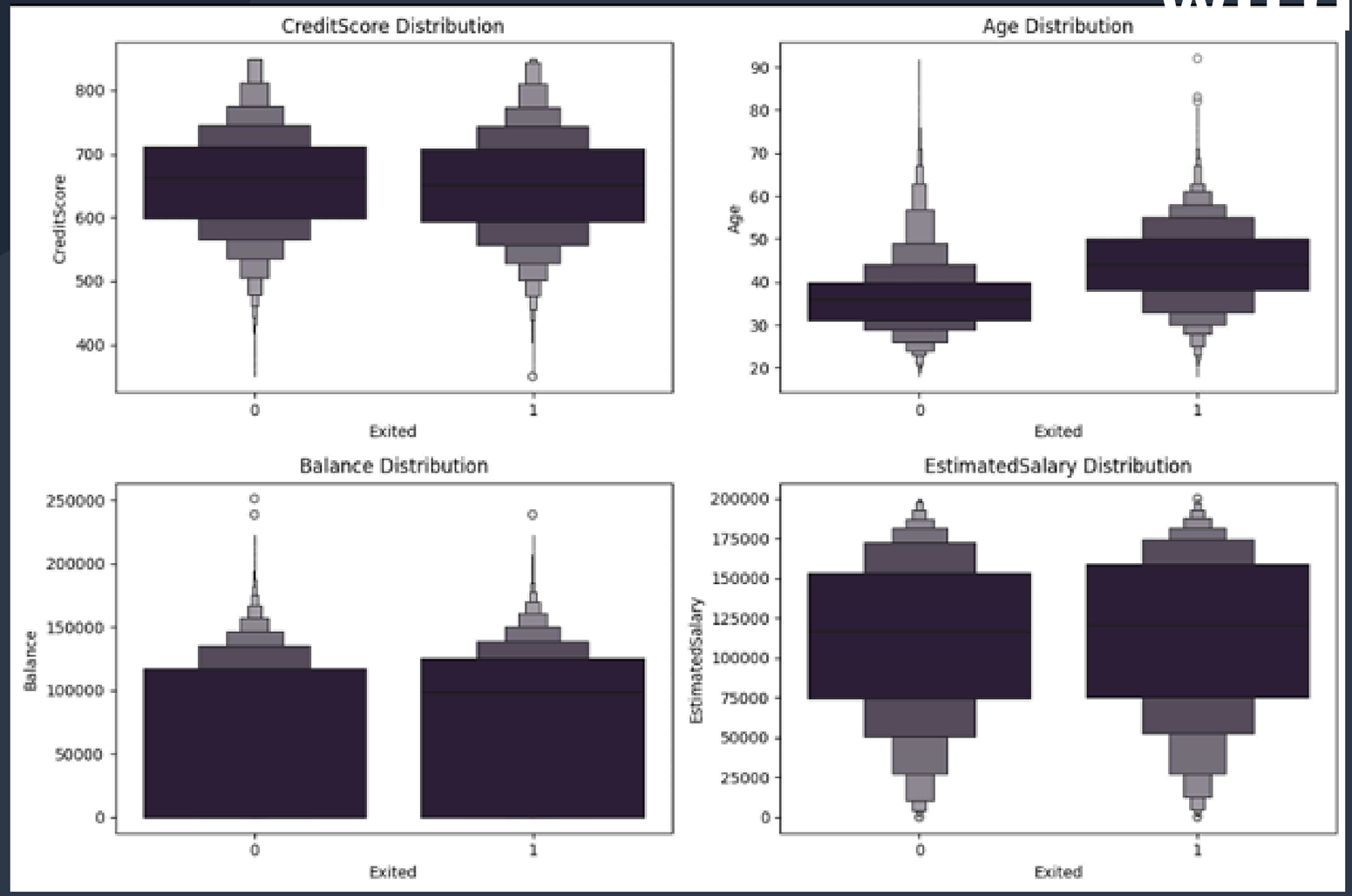
1.Data Science

Python



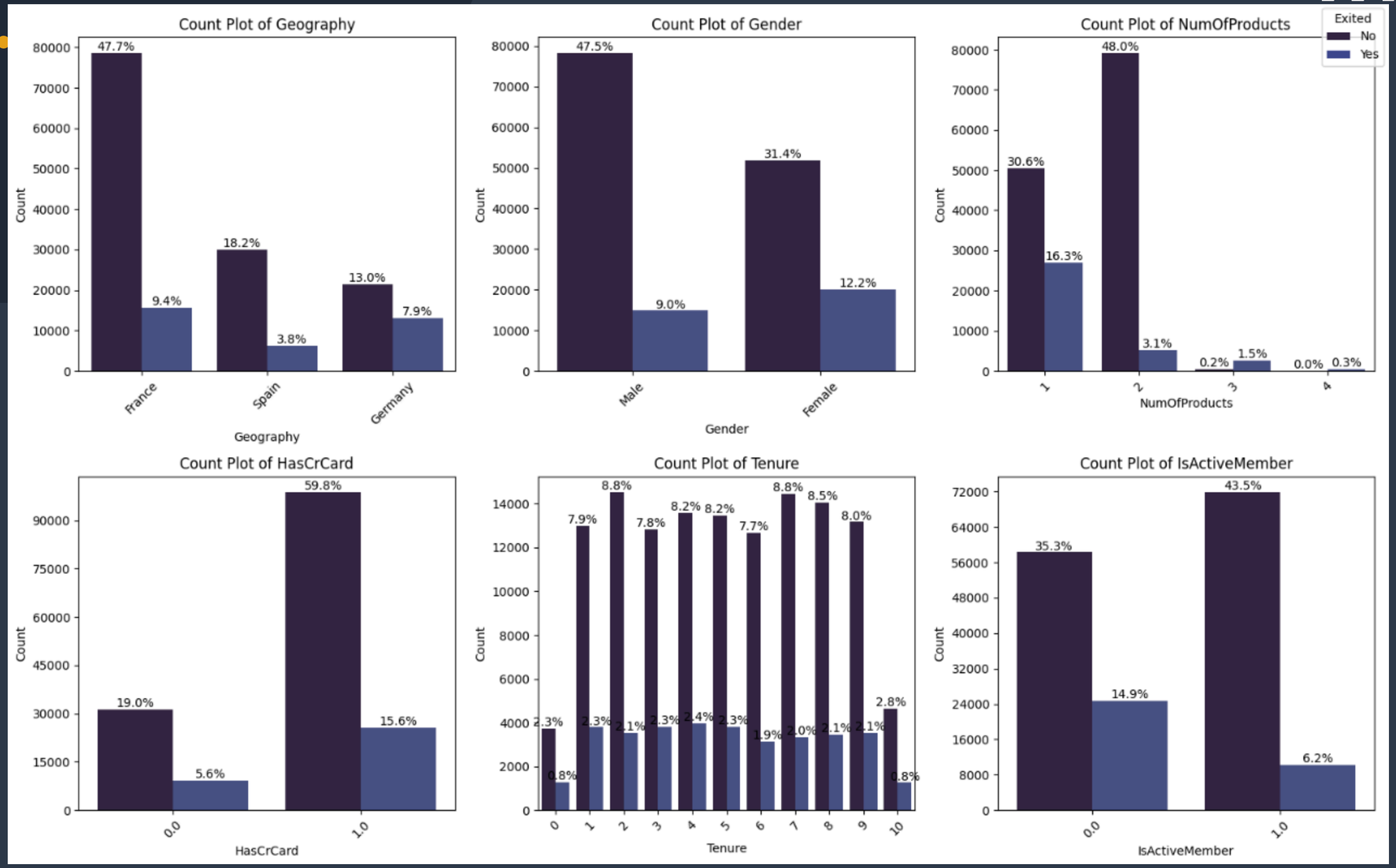
Plot Categorical Columns

1.Data Science with Python

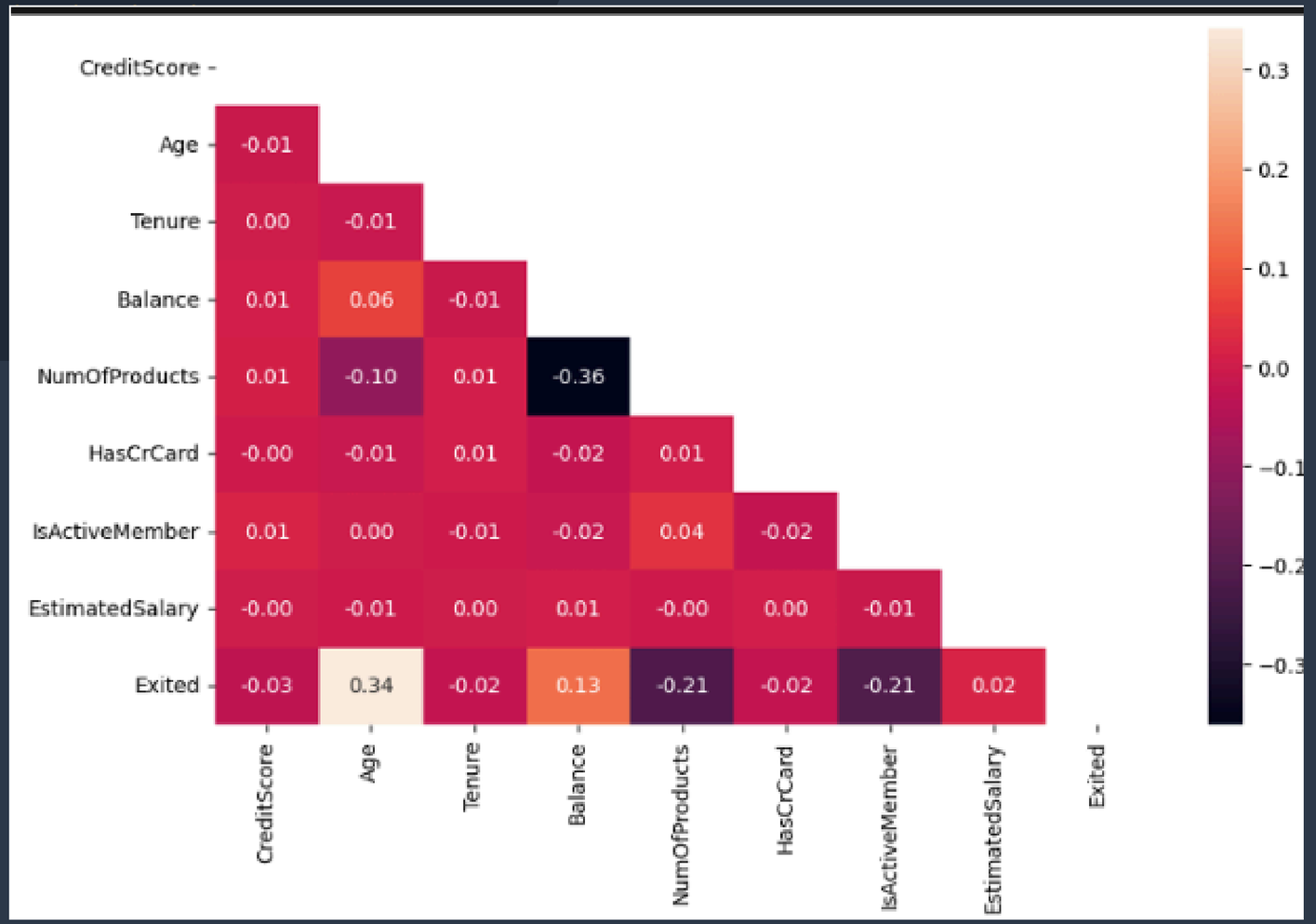


1.Data Science with Python

Categorical Columns EDA



1.Data Science with Python



Plotting Collinearity

1.Data Science with Python

ML Packages

```python
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.metrics import accuracy_score,classification_report,f1_score,mean_squared_error,roc_auc_score,precision_score,recall_score,roc_curve,ConfusionMatrixDisplay,confusion_matrix,auc
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler,LabelEncoder,OneHotEncoder,OrdinalEncoder,RobustScaler
from sklearn.linear_model import LogisticRegression,SGDClassifier, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,ExtraTreesClassifier,AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.base import BaseEstimator,TransformerMixin
from xgboost import XGBClassifier
from sklearn.impute import SimpleImputer,KNNImputer
from catboost import CatBoostClassifier
from sklearn.model_selection import RandomizedSearchCV
import category_encoders as ce
```

Python

ML Packages

1.Data Science with Python

```
class FullPipeline1:
    def __init__(self):
        self.numerical_cols = ['CreditScore', 'Age', 'Balance', 'EstimatedSalary']
        self.categorical_cols = ['Geography', 'Gender', 'NumOfProducts', 'HasCrCard', 'Tenure', 'IsActiveMember']
        self.OD_cols = ['Gender', 'NumOfProducts', 'HasCrCard', 'Tenure', 'IsActiveMember']
        self.OH_cols = ['Geography']
        self.TE_cols = ['Surname']
        self.full_pipeline = Pipeline([
            ('impute_num', DataFrameImputer(knn_cols=self.numerical_cols)),
            ('impute_cat', DataFrameImputer(freq_cols=self.categorical_cols)),
            ('scale', StandardScaleTransform(self.numerical_cols)),
            ('ordinal_encode', OrdinalEncodeColumns(self.OD_cols)),
            ("one_hot_encode", CustomOneHotEncoder(self.OH_cols)),
            ('target_encode', TargetEncoderTransformer(self.TE_cols))
        ])
    def fit(self, X_train, y_train):
        self.full_pipeline.fit(X_train, y_train)

    def transform(self, X_test):
        return self.full_pipeline.transform(X_test)
    def fit_transform(self, X_train, y_train):
        X_train = self.full_pipeline.fit_transform(X_train, y_train)
        return X_train, y_train

f1 = FullPipeline1()
```

Preprocessing Pipeline

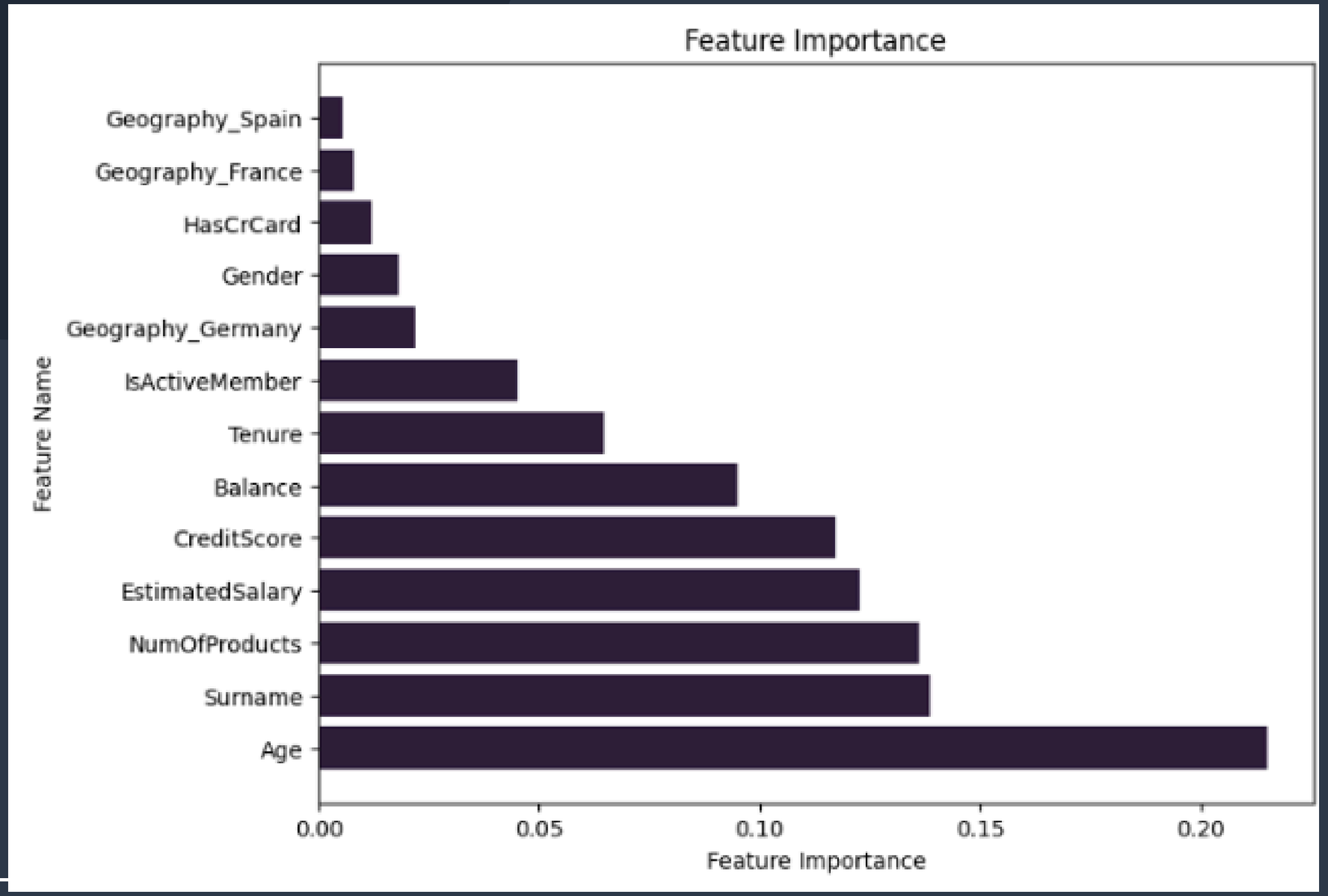
1.Data Science with Python

Step	Description
1	Numerical Imputation (KNN): CreditScore, Age, Balance, EstimatedSalary
2	Categorical Imputation (Frequency): Geography, Gender, NumOfProducts, HasCrCard, Tenure, IsActiveMember
3	Scaling Numerical Columns: CreditScore, Age, Balance, EstimatedSalary
4	Ordinal Encoding: Gender, NumOfProducts, HasCrCard, Tenure, IsActiveMember
5	One-Hot Encoding: Geography
6	Target Encoding: Surname

Preprocessing Pipeline

Feature Importance

1.Data Science with Python

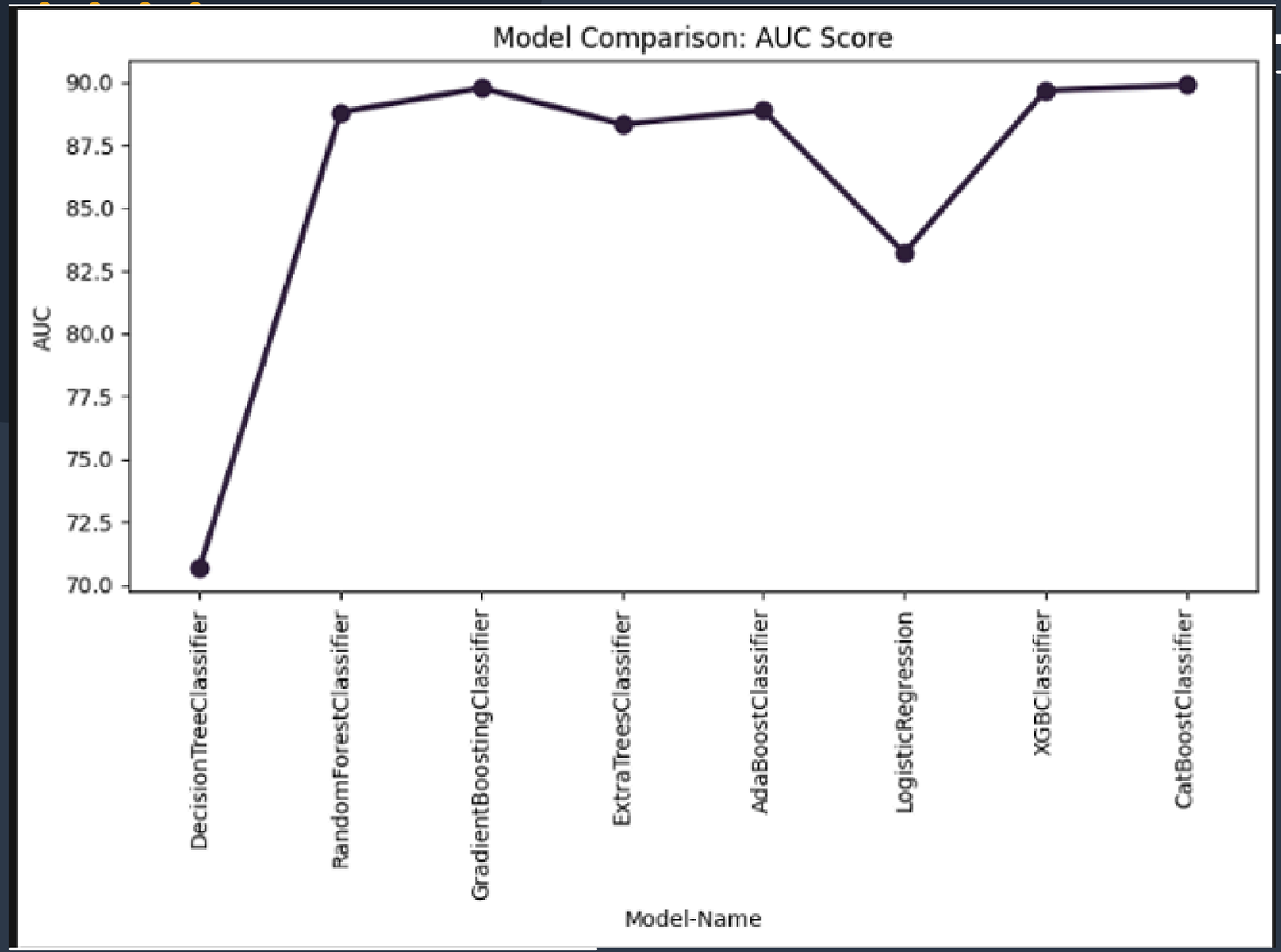


1.Data Science with Python

	Model-Name	Accuracy	AUC	F1-Score
7	CatBoostClassifier	87.030024	89.883171	65.135597
2	GradientBoostingClassifier	86.914897	89.766216	64.426324
6	XGBClassifier	86.987609	89.661380	65.504779
4	AdaBoostClassifier	86.705850	88.877745	63.699537
1	RandomForestClassifier	86.584664	88.803202	63.675144
3	ExtraTreesClassifier	86.239283	88.326611	62.450397
5	LogisticRegression	83.876147	83.196907	51.958837
0	DecisionTreeClassifier	79.849729	70.702893	53.440672

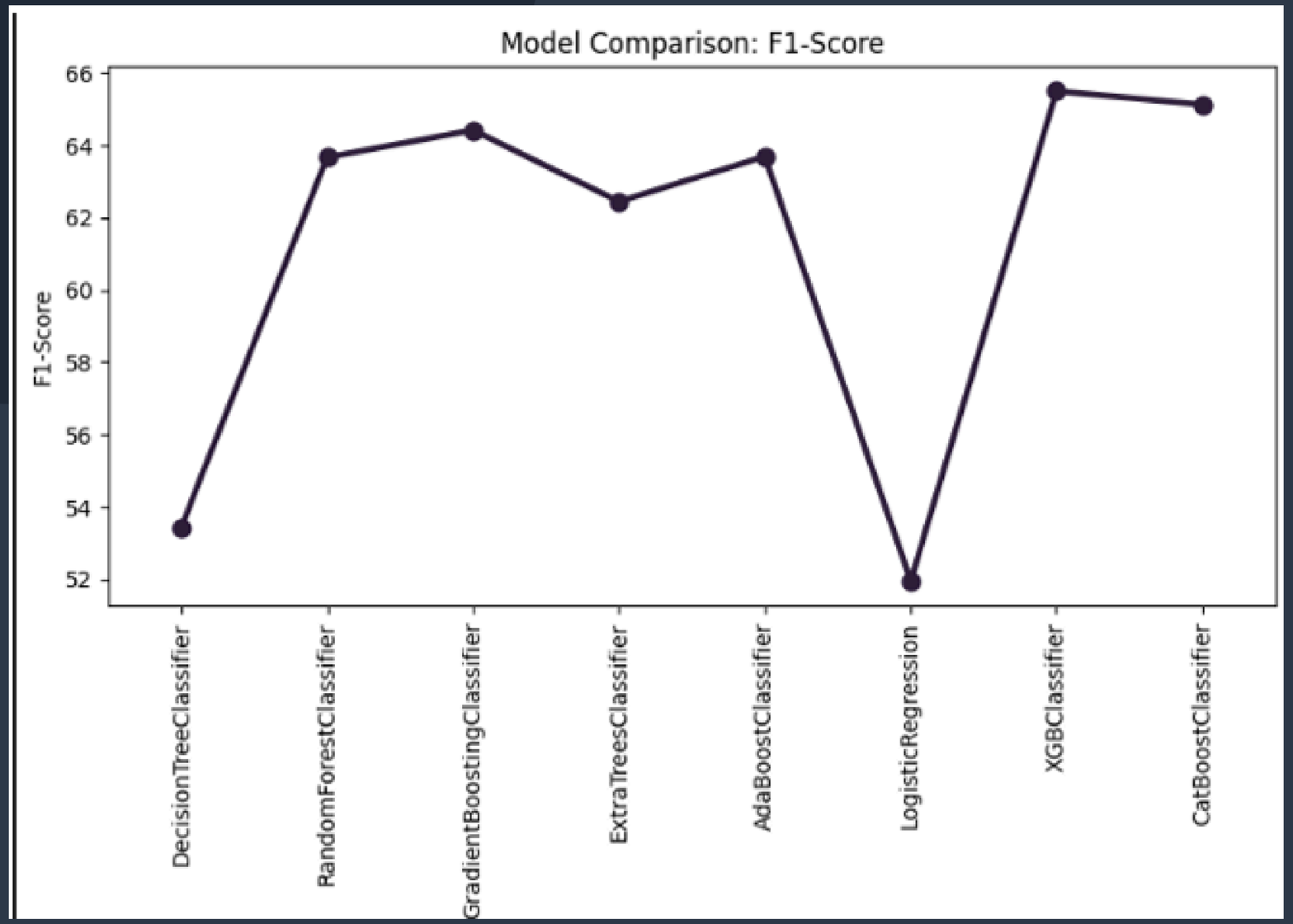
Model Evaluation

1.Data Science with Python



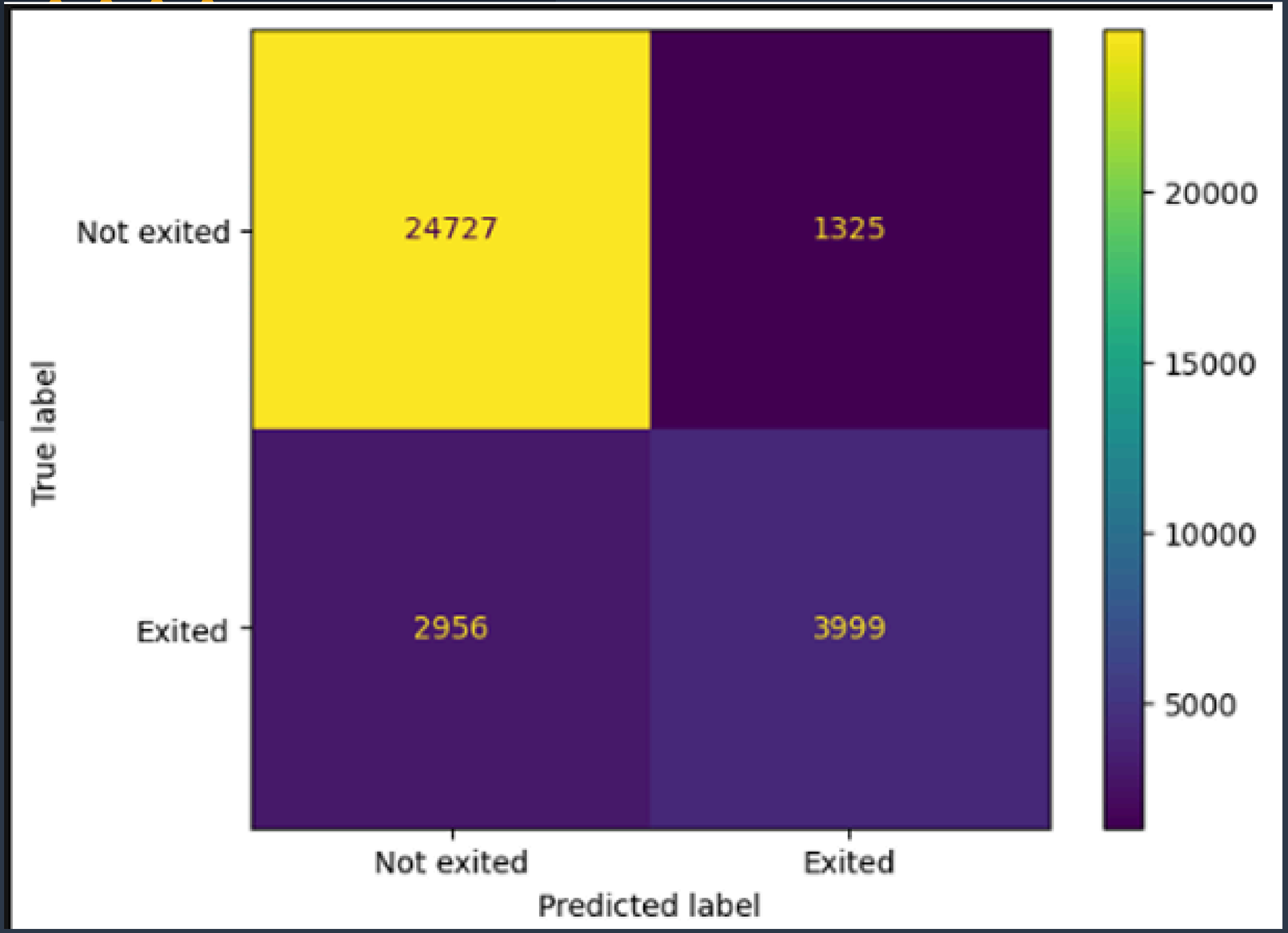
Model Evaluation

1.Data Science with Python

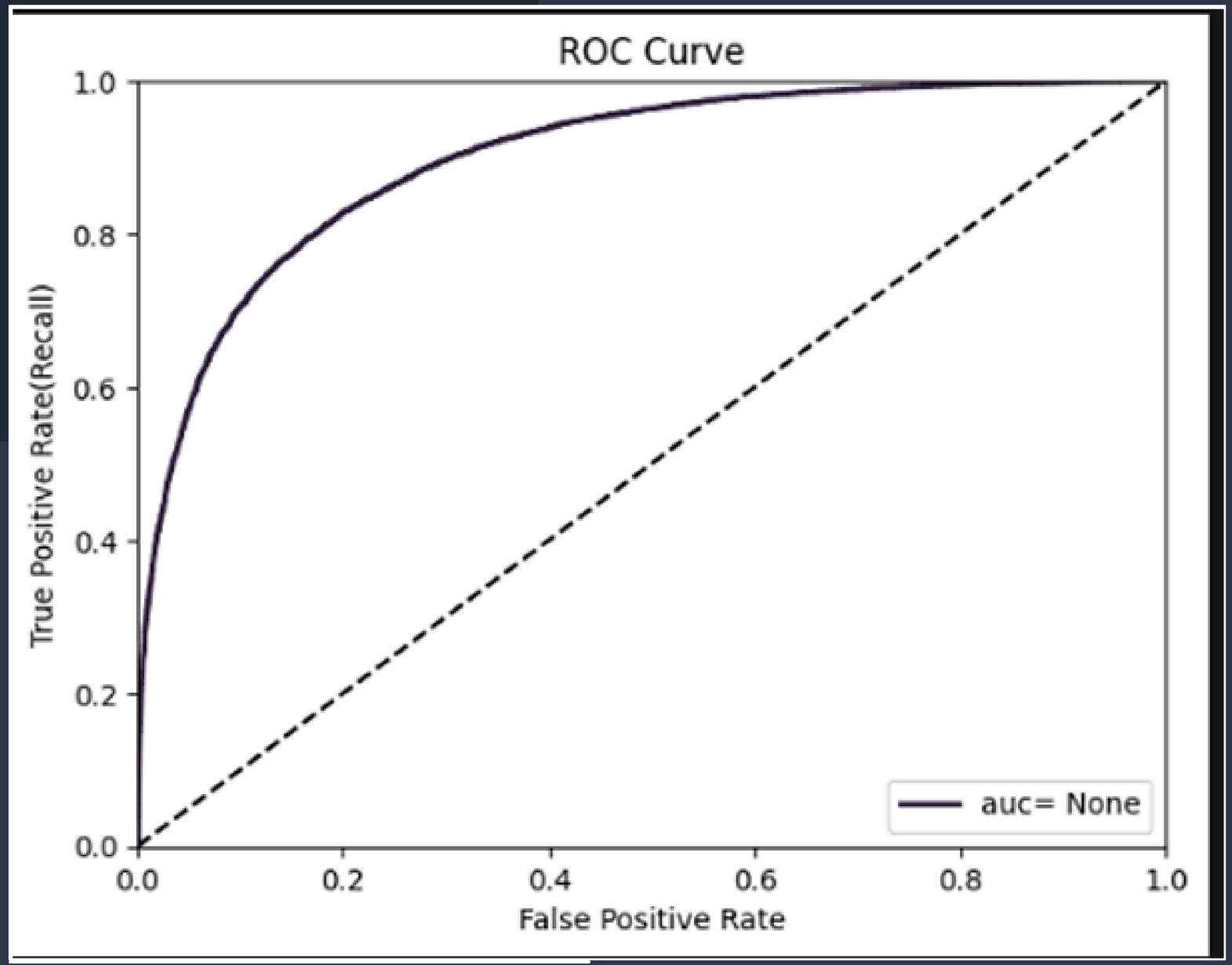


Model Evaluation

1.Data Science with Python



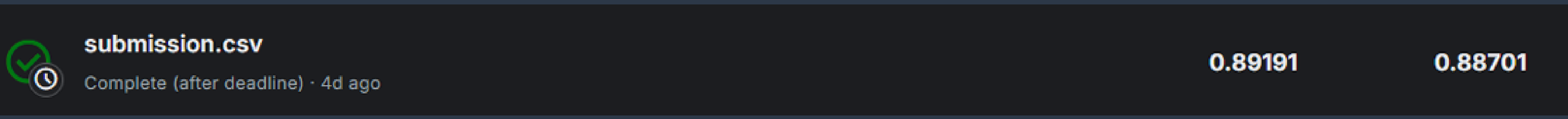
Confusion Matrix



1.Data Science with Python

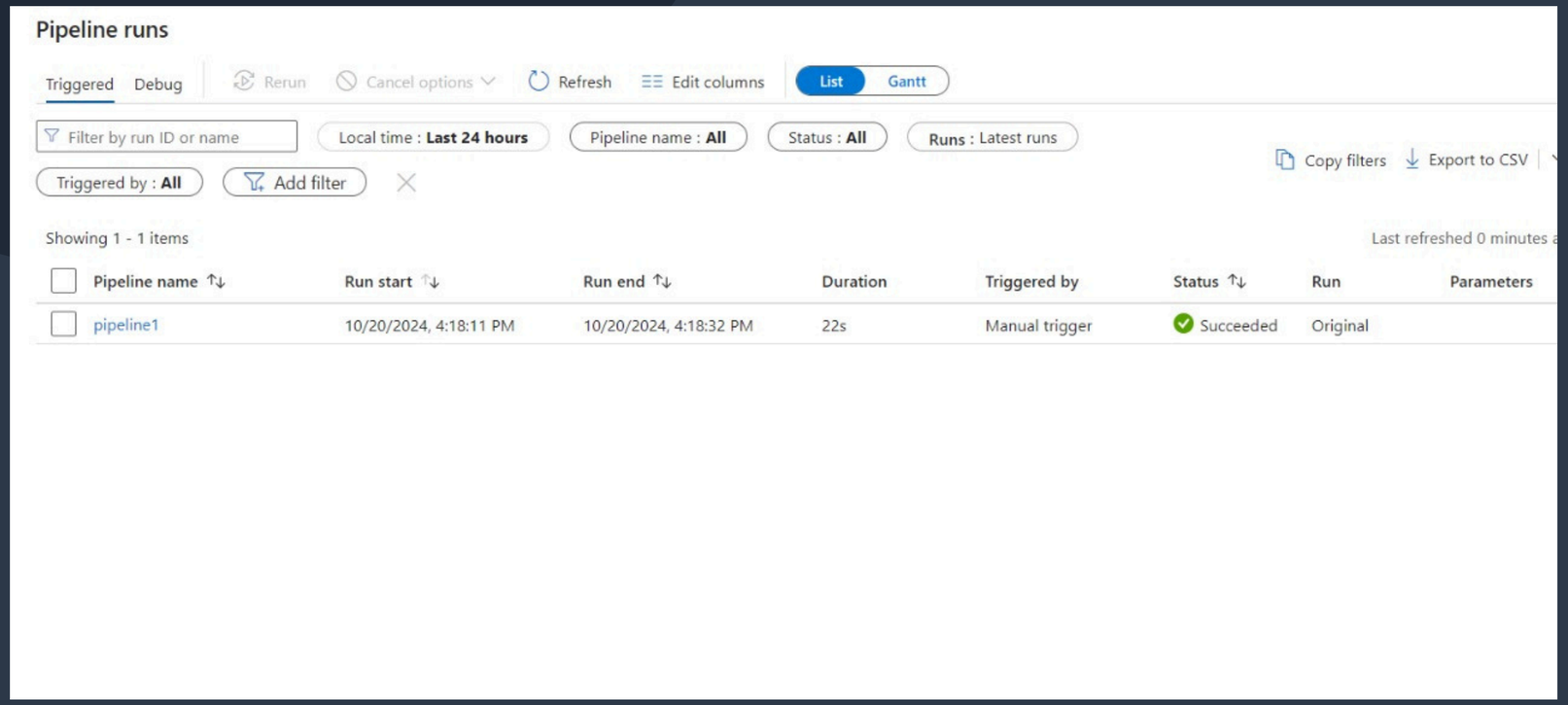
ROC Plot

1.Data Science with Python



Final Model

2.Azure Data Fundamentals



Utilize Azure Data services to manage and analyze customer data.

3.Model Development

Search

Algorithm name	Explained	Responsible AI	AUC weighted ↓	Sampling
VotingEnsemble	View explanation		0.89093	100.00 %
StackEnsemble			0.89048	100.00 %
MaxAbsScaler, LightGBM			0.88996	100.00 %
StandardScalerWrapper, XGBoostClassifier			0.88987	100.00 %
MaxAbsScaler, XGBoostClassifier			0.88901	100.00 %
SparseNormalizer, XGBoostClassifier			0.88846	100.00 %
SparseNormalizer, XGBoostClassifier			0.88833	100.00 %
SparseNormalizer, XGBoostClassifier			0.88721	100.00 %

Develop and evaluate machine learning models using Azure Machine Learning or similar services

3.Model Development

Develop and evaluate machine learning models using Azure Machine Learning or similar services

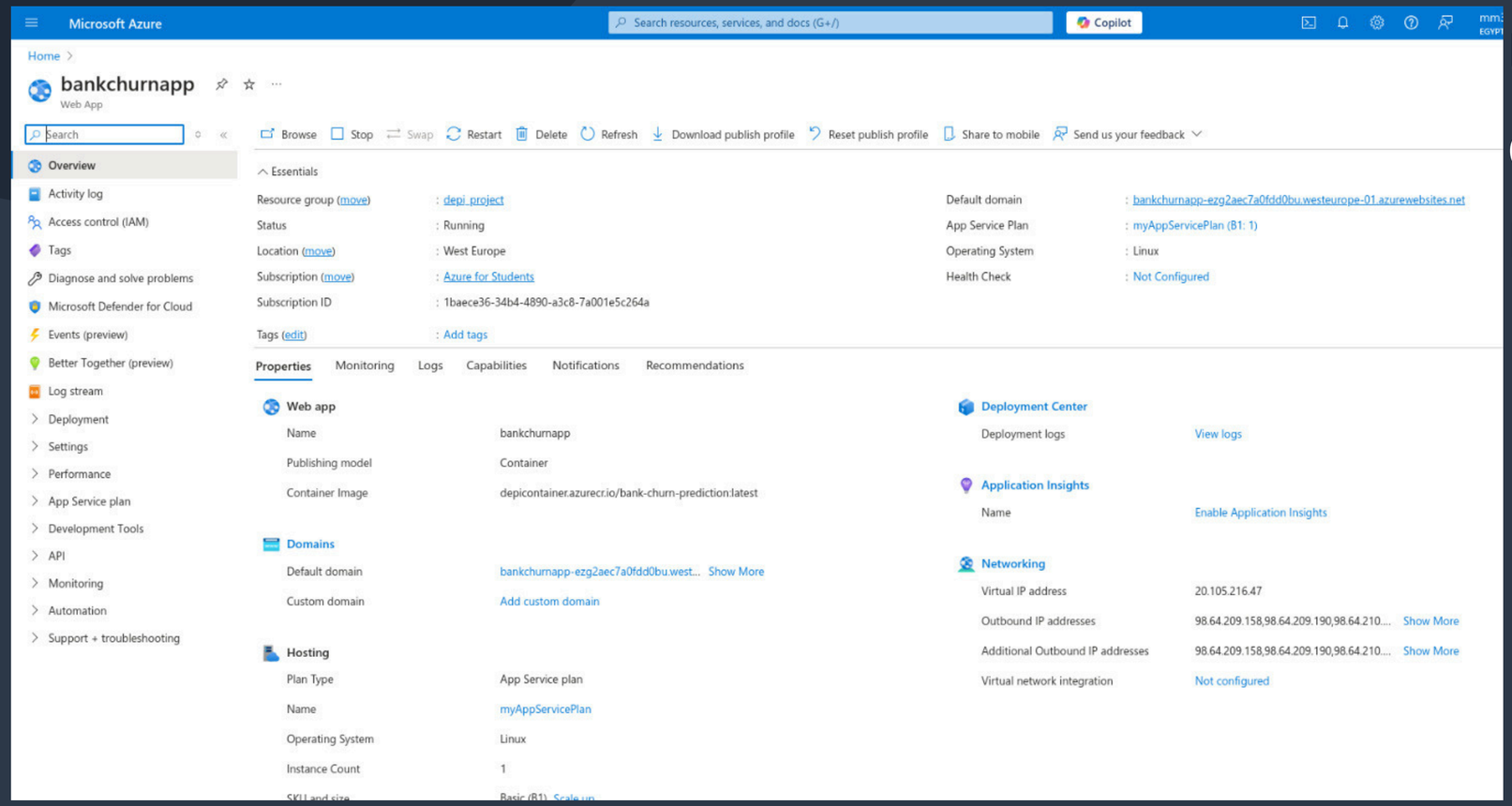
WEEK 4: MLOPS, DEPLOYMENT, AND FINAL PRESENTATION

1.MLOps Implementation

Use MLflow to track experiments and manage machine learning models.

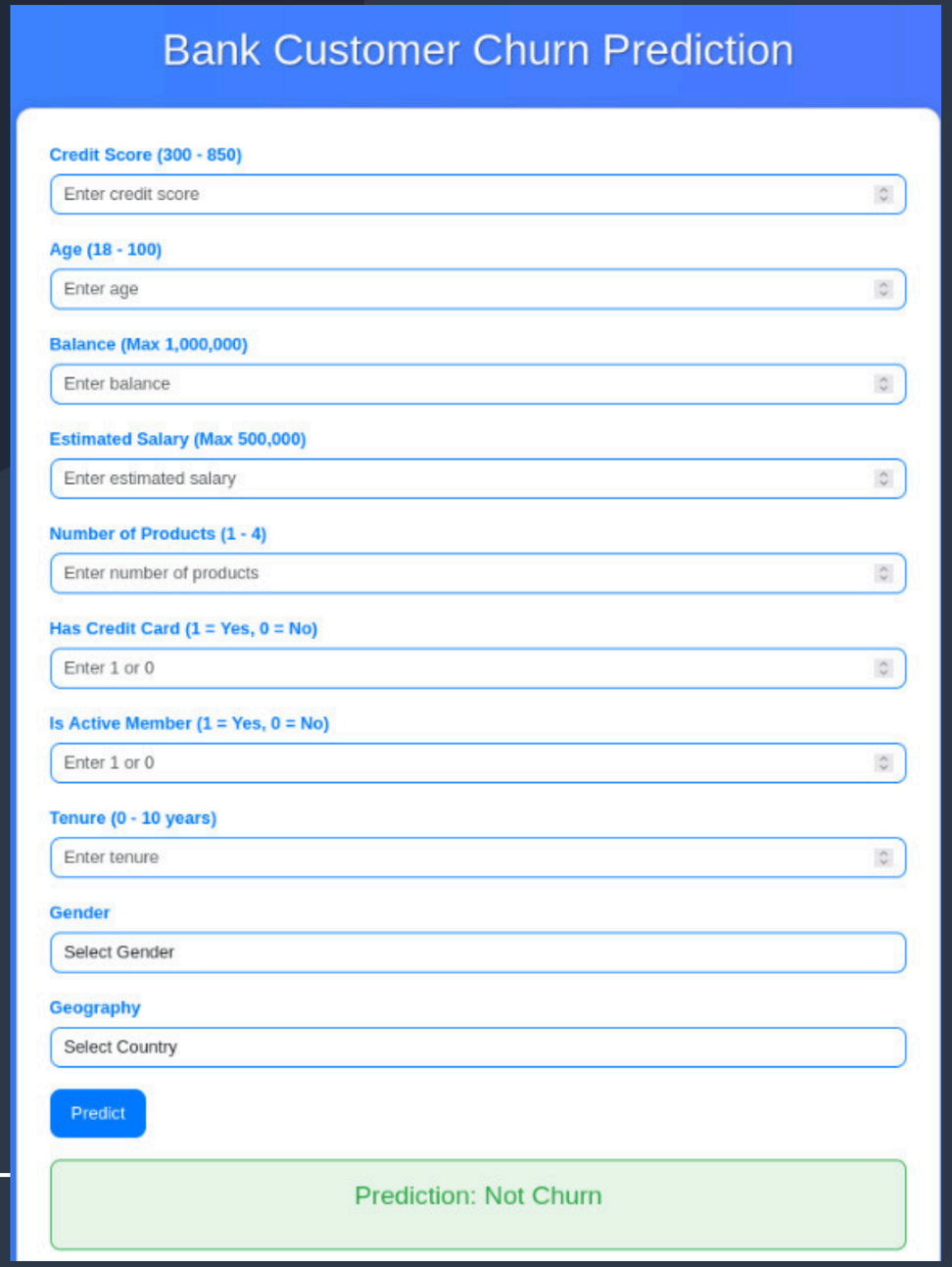
```
(base) abdelhalim@abdelhalim:~$ docker image ls
REPOSITORY                                  TAG       IMAGE ID       CREATED         SIZE
bank-churn-prediction                       latest    0f360f161579   3 days ago      1.62GB
```

1.MLOps Implementation



Use MLflow to track experiments and manage machine learning models.

2.Model Deployment



Deploy the machine learning model using Azure services or create a web application for model predictions.

Repo on GitHub

Final Report &Presentation

THANKS