•**Title:**
Amazon Sales Analytics & Forecasting – End-to-End Data Pipeline
•**Subtitle:**
SQL + Power BI + Python Forecasting + Cloud Simulation (Azurite)
•Name: Tanmay Sharma
•Role: Data Analyst | Specialized in BI, SQL, Forecasting & Cloud Integration
•**LinkedIn:** http://www.linkedin.com/in/

Project Overview

•**Goal:**

Build a complete sales analytics solution with:

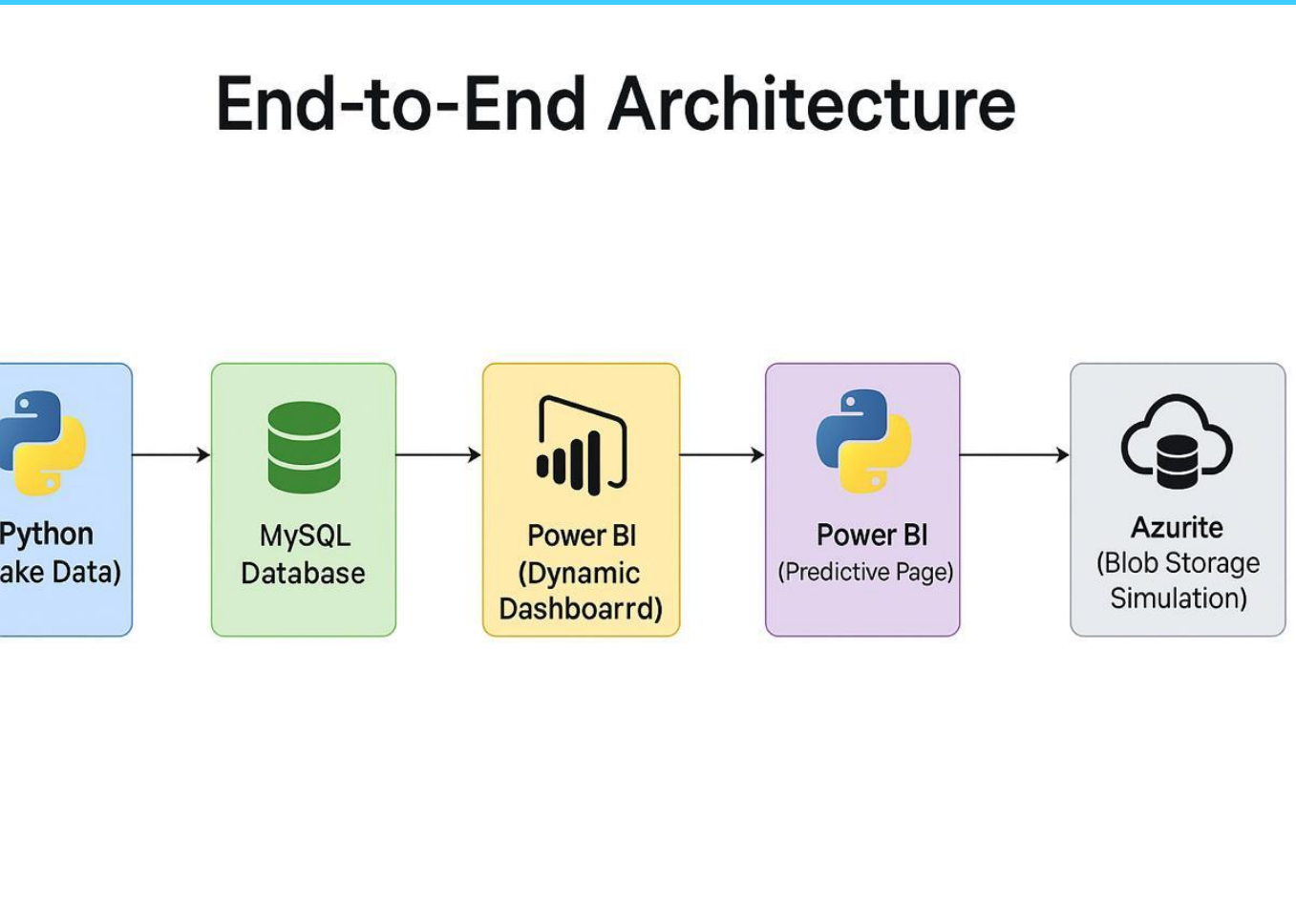
- Data generation (Python)
- Database (MySQL)
- Visualization (Power BI)
- Forecasting (Prophet)
- Cloud Storage Simulation (Azurite)

•**Tools Used:** MySQL, Python, Power BI, Azurite, Azure Storage Explorer.

End-to-End Architecture:

- **Flow:**

Python (Fake Data) → MySQL DB → Power BI → Python Forecast → Power BI Predictive Page → Azurite (Blob Simulation)



Data Generation:

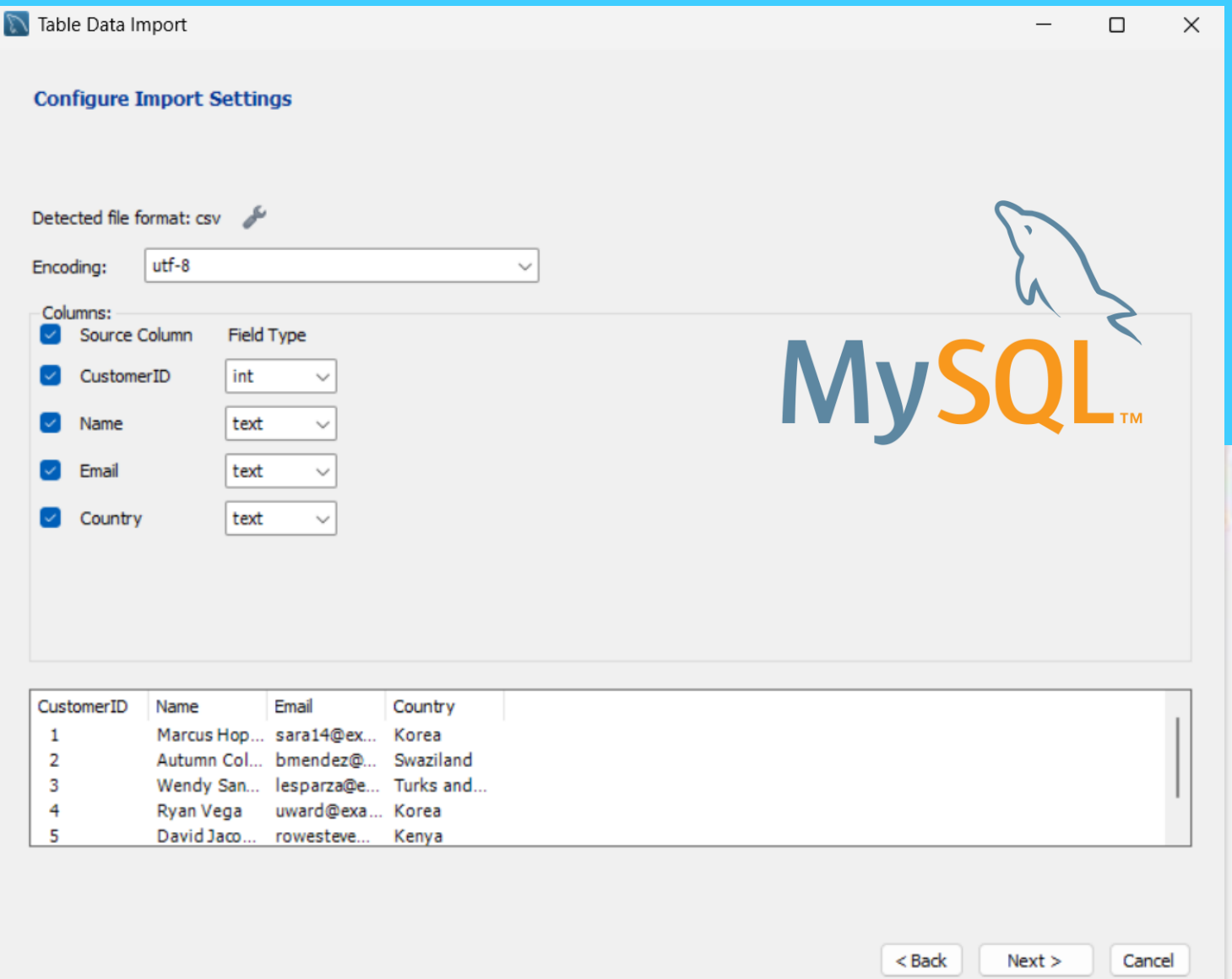
- **Description:**
  - Python script created **Customers, Products, Orders, Order_Details**.
  - Libraries: Faker, Pandas, Random.

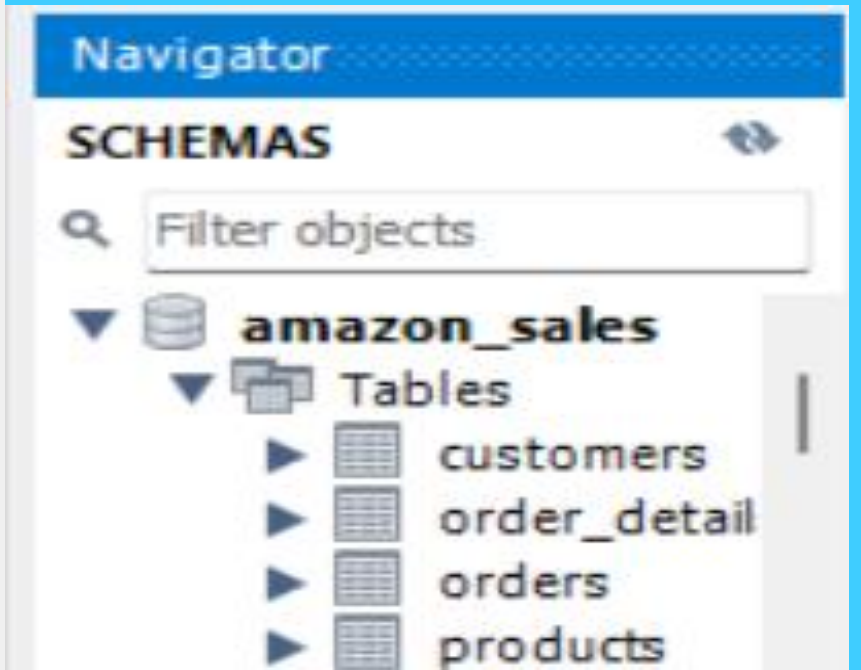
Amazon.py > ...

```python
import pandas as pd
import random
from faker import Faker
from datetime import datetime

fake = Faker()

# Configuration
NUM_CUSTOMERS = 2000
NUM_PRODUCTS = 500
NUM_ORDERS = 10000
categories = ["Electronics", "Home", "Fashion", "Beauty", "Sports", "Books"]
regions = ["North America", "Europe", "Asia", "South America"]
payment_methods = ["Credit Card", "Debit Card", "PayPal", "Net Banking", "UPI"]

# Generate Customers Table
customers = []
for i in range(1, NUM_CUSTOMERS + 1):
    customers.append([
        i,
        fake.name(),
        fake.email(),
        fake.country()
    ])
customers_df = pd.DataFrame(customers, columns=["CustomerID", "Name", "Email", "Country"])

# Generate Products Table
products = []
for i in range(1, NUM_PRODUCTS + 1):
    category = random.choice(categories)
    price = round(random.uniform(10, 500), 2)
    products.append([i, f"{category} Product {i}", category, price])
products_df = pd.DataFrame(products, columns=["ProductID", "ProductName", "Category", "Price"])

# Generate Orders and Order Details
orders = []
order_details = []
order_id = 1
for _ in range(NUM_ORDERS):
    customer_id = random.randint(1, NUM_CUSTOMERS)
    order_date = fake.date_between(start_date='-2y', end_date='today')
    region = random.choice(regions)
    payment_method = random.choice(payment_methods)

    orders.append([order_id, customer_id, order_date, region, payment_method])
```

Database Setup:

- **Description:**
  - Created amazon_sales schema in MySQL.
  - Imported generated CSVs via **Table Data Import Wizard**.





SQL Analysis:

•Queries Performed:

- Monthly Revenue & Profit
- Top Products by Revenue
- Regional Revenue
- Customer Lifetime Value

Result Grid | Filter Rows: | Export:

Year	Month	Monthly_Revenue	Monthly_Profit
2023	8	563201.93	112749.19
2023	9	616622.27	121715.19
2023	10	683420.26	136388.44
2023	11	638417.38	129880.25
2023	12	658462.39	132185.30
2024	1	684749.75	134721.19
2024	2	591255.02	117805.81
2024	3	626382.53	123657.00
2024	4	604027.79	121255.85

Customer	Total_Spent
Lonnie Smith	26192.62
Carl Ramirez	23853.04
Leslie Davis	22210.32
Marissa Mack	22146.91
Caitlin Rice	21817.42
Jeanne West	21581.38
Matthew Bell	21211.76
Daniel Case	20555.06
Diane Castro	20485.31
Haley Wagner	20437.21

ProductName	Total_Revenue
Beauty Product 374	80519.25
Fashion Product 435	74890.92
Electronics Product 174	73153.46
Electronics Product 159	72057.50
Sports Product 188	71303.82
Sports Product 190	70011.40
Home Product 216	69416.44
Fashion Product 227	68657.40
Electronics Product 104	67946.80
Electronics Product 249	66085.04

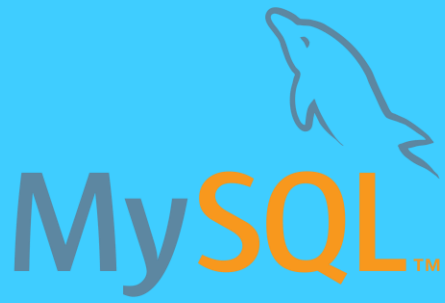
Result Grid | Filter Rows: | Export:

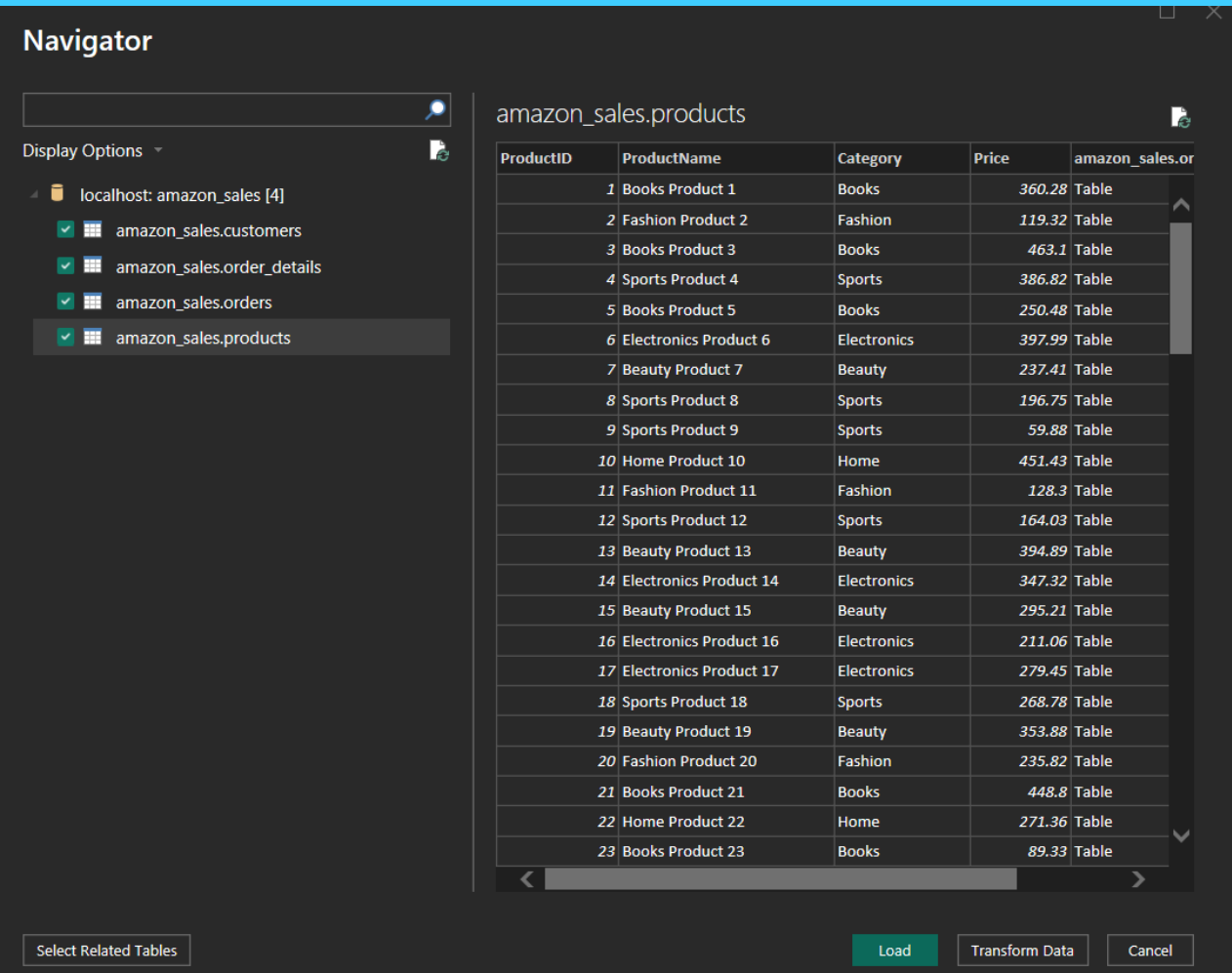
Region	Regional_Revenue	Regional_Profit
South America	3990996.71	804459.71
Asia	3889439.32	775746.32
Europe	3836225.18	765697.77
North America	3798811.63	763227.36

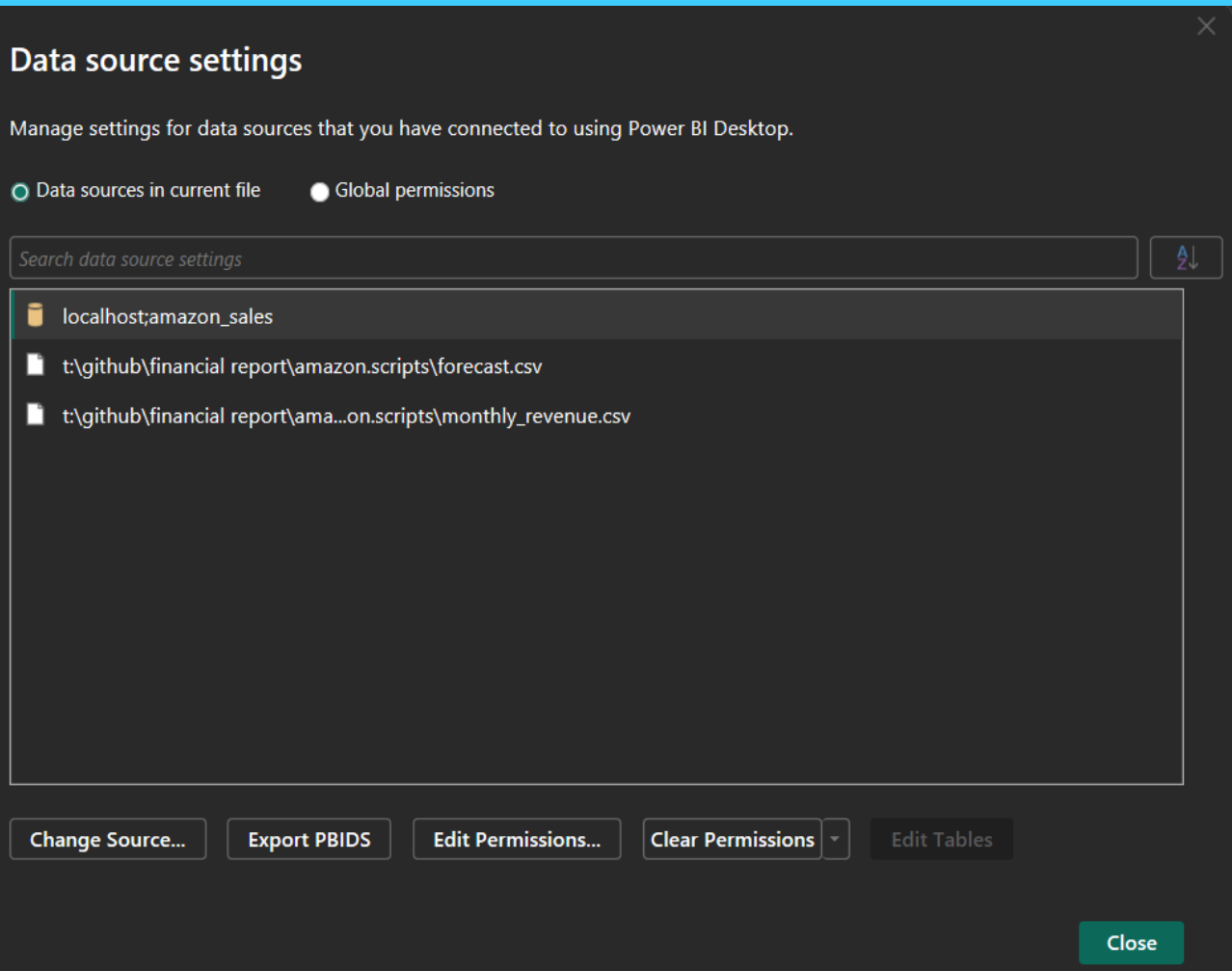
Power BI Integration:

•Description:

- Connected to MySQL for **dynamic dashboards**.
- Created relationships:
  - Customers → Orders → Order_Details
  - Products → Order_Details

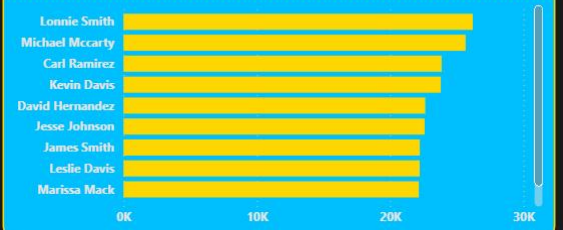






"Amazon Customer Insights Dashboard"

Top 10 Customers by Spend



Customers by Country

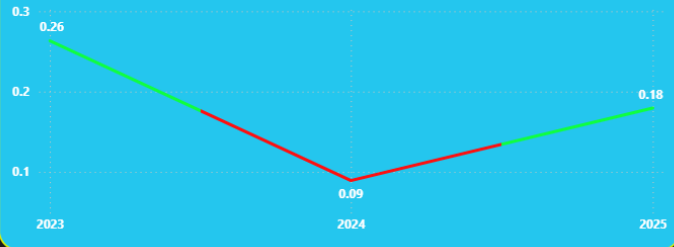
This page answers:

- Who are the top spending customers?
- Where are customers located?
- How many repeat vs new customers exist?

Repeat vs New Customers

"Amazon Growth & Profitability

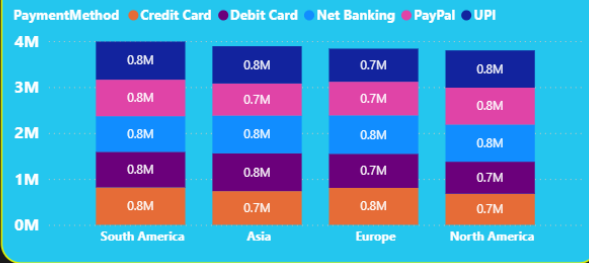
Revenue Growth % Over Time



Profit Margin by Region & Category

Region	Beauty	Books	Electronics	Fashion	Home	Sports	Total
Asia	0.20	0.20	0.20	0.20	0.20	0.20	0.20
Europe	0.20	0.20	0.20	0.20	0.20	0.20	0.20
North America	0.20	0.20	0.20	0.20	0.20	0.20	0.20
South America	0.20	0.20	0.20	0.20	0.20	0.20	0.20
Total	0.20	0.20	0.20	0.20	0.20	0.20	0.20

Revenue by Region & Payment Method

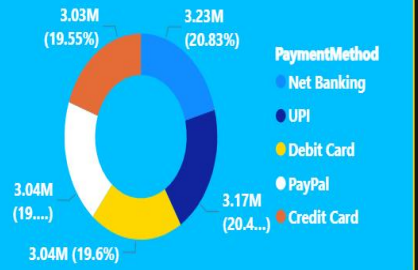


"Amazon Sales Performance Summary"

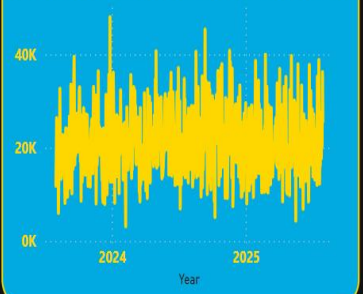
Total Revenue	Total Profits	Total Orders	Avg Order Value	Revenue Growth %
15.52M	3.11M	10K	1.55K	0.04

Revenue by Region

Revenue by Payment Method

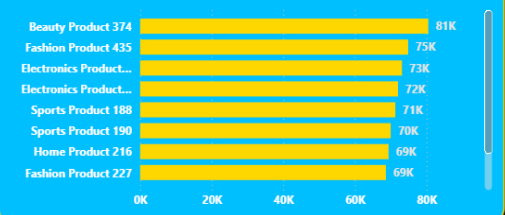


Monthly Revenue

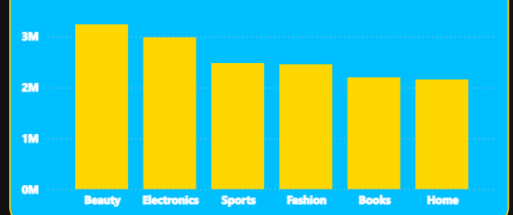


"Amazon Product Performance Analysis"

Top 10 Products by Revenue



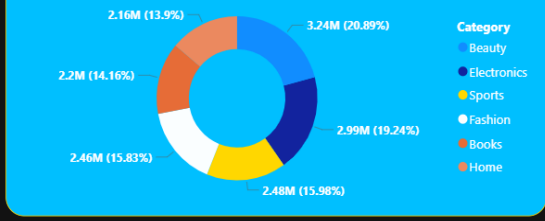
Revenue by Category



This page answers:

- Which products are generating the most revenue?
- Which categories contribute the most?

Category Contribution %

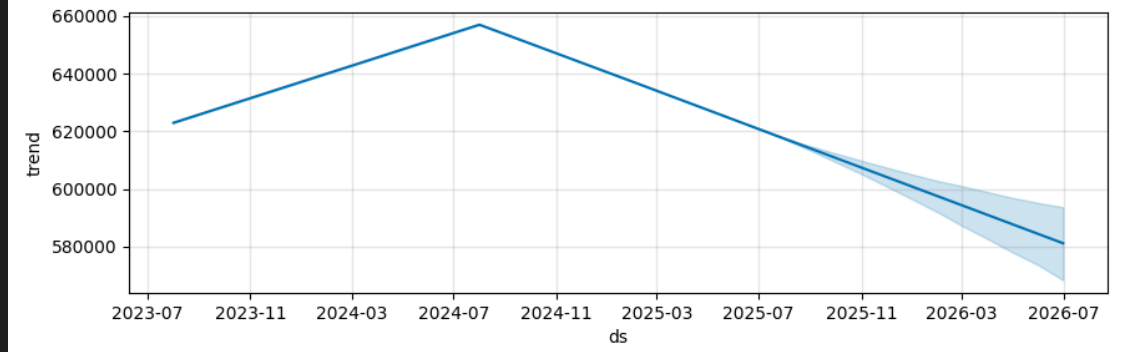


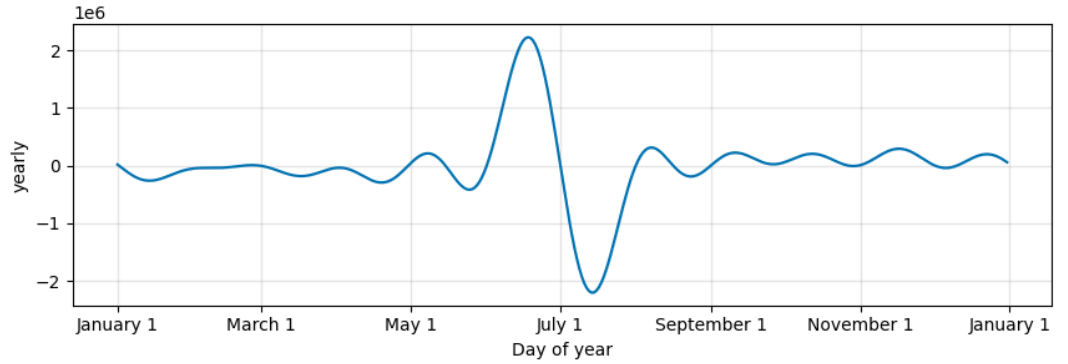
Predictive Analysis:

- **Description:**
  - Forecasted next 12 months using **Prophet**.
  - Imported forecast.csv into Power BI

forcast.py > ...

```python
# amazon_forecast.py

import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# Step 1: Load Historical Data
# Ensure your monthly_revenue.csv has columns: Month, Monthly_Revenue
df = pd.read_csv("monthly_revenue.csv")

# Rename columns as required by Prophet
df.rename(columns={"Month": "ds", "Monthly_Revenue": "y"}, inplace=True)

# Convert 'ds' to datetime
df['ds'] = pd.to_datetime(df['ds'])

# Step 2: Initialize Prophet Model
model = Prophet(yearly_seasonality=True, daily_seasonality=False)
model.fit(df)

# Step 3: Create Future DataFrame for 12 months
future = model.make_future_dataframe(periods=12, freq='M')

# Step 4: Forecast
forecast = model.predict(future)

# Step 5: Save Forecast to CSV
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].to_csv("forecast.csv", index=False)

# Step 6: Plot Forecast
fig1 = model.plot(forecast)
plt.title("Amazon Sales Forecast (Next 12 Months)", fontsize=16)
plt.xlabel("Date")
plt.ylabel("Revenue")
plt.show()

# Step 7: Plot Components (Trend, Seasonality)
fig2 = model.plot_components(forecast)
plt.show()

print("Forecast completed! File saved as forecast.csv")
```





"Predictive Analysis – Future Revenue Forecast"

Next Month "Predicted Revenue"

22.86M

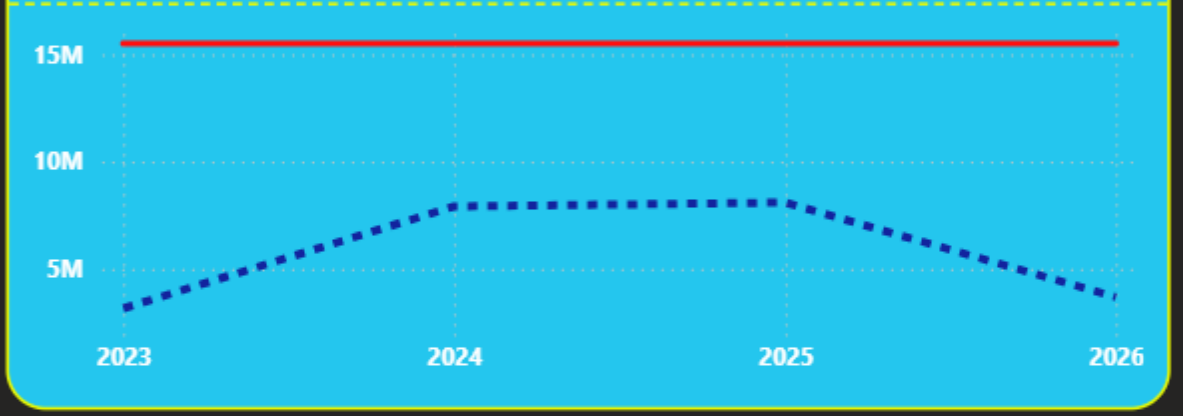
Confidence Range

₹963,446 - ₹990,326

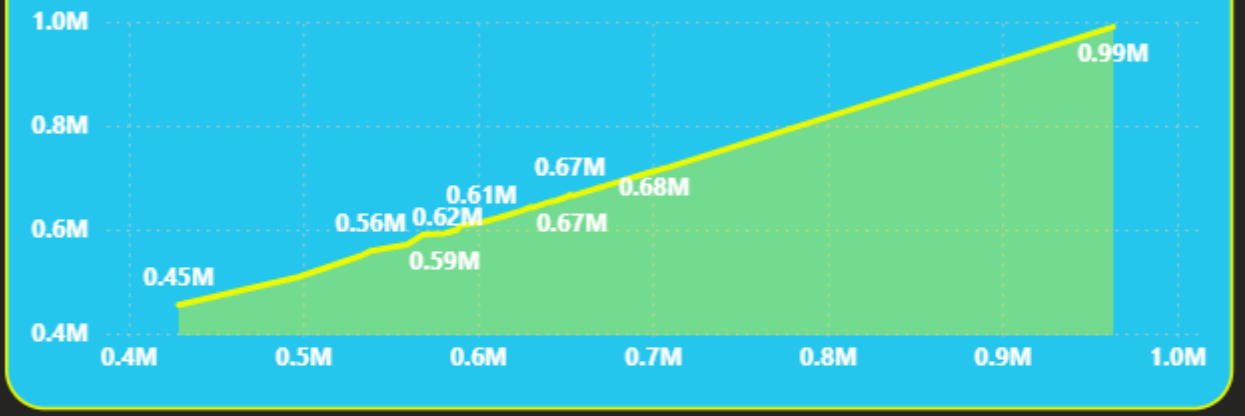
Forecast Growth %

0.37

(Historical + Forecast)



Confidence Interval



Forecast Table

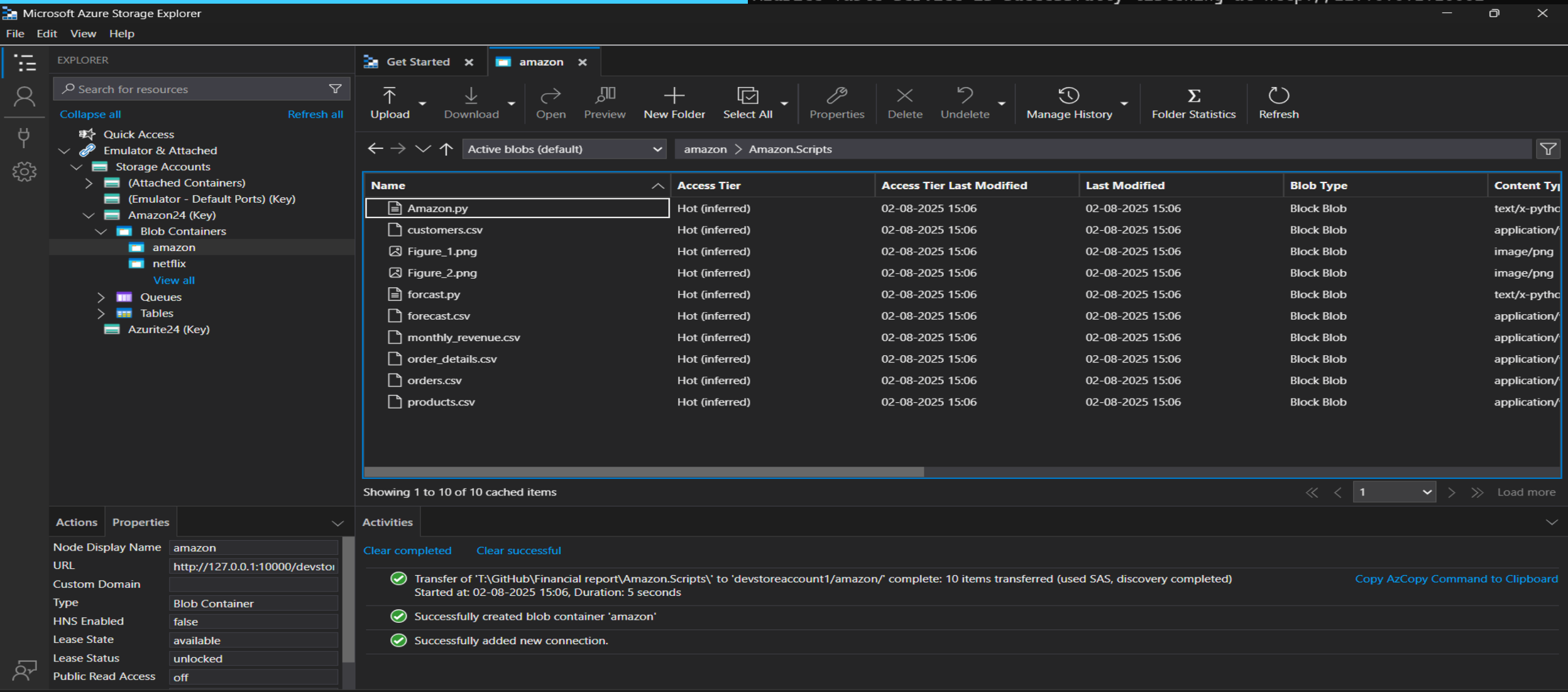
Year	Quarter	Month	Day	Sum of yhat	Sum of yhat_lower	Sum of yhat_upper
2023	Qtr 3	August	1	5,64,882.16	5,59,620.91	5,70,669.32
2023	Qtr 3	September	1	6,17,687.63	6,12,168.24	6,22,976.50
2023	Qtr 4	October	1	6,83,321.61	6,77,669.11	6,88,929.83
2023	Qtr 4	November	1	6,36,861.36	6,31,698.20	6,42,313.70
2023	Qtr 4	December	1	6,55,622.50	6,50,035.89	6,61,401.98
2024	Qtr 1	January	1	6,81,121.15	6,75,401.34	6,86,893.37
2024	Qtr 1	February	1	5,87,010.01	5,81,003.29	5,92,692.94
Total				**2,28,58,820.07**	**2,26,21,724.62**	**2,30,92,550.68**

Cloud Simulation

•**Description:**

- Used **Azurite** for Azure Blob Storage simulation.
- Uploaded all project files via Azure Storage Explorer.

```
Azurite Blob service is starting at http://127.0.0.1:10000
Azurite Blob service is successfully listening at http://127.0.0.1:10000
Azurite Queue service is starting at http://127.0.0.1:10001
Azurite Queue service is successfully listening at http://127.0.0.1:10001
Azurite Table service is starting at http://127.0.0.1:10002
Azurite Table service is successfully listening at http://127.0.0.1:10002
```



Key Learnings

- ETL Pipeline Design
- SQL & DAX for BI
- ML Forecasting (Prophet)
- Cloud Readiness with Azurite

Tagline:

End-to-End Sales Analytics Project using MAANG standards.

Link to Github repository: https://github.com/Tanu272004/Amazon_Analysis_Project.git