Amaron Sales Analysis Report

Overview

Welcome to an in-depth exploration of Amaron's sales data! In this report, we'll uncover trends, analyze key metrics, and share actionable insights in a clear and engaging manner. It includes steps for data preprocessing, revenue aggregation, and insights through visualizations. Additional functionalities such as file management using AWS S3 services have also been implemented to manage data efficiently.

Data Management and S3 Integration

Uploading Files to S3

Using the provided Python code, files are uploaded to an AWS S3 bucket, which serves as a centralized storage solution. This approach offers significant benefits, such as ensuring data accessibility from any location, maintaining a single source of truth for all data files, and enabling seamless version control. Additionally, the inclusion of timestamps in file paths simplifies tracking and management, making it ideal for collaborative and large-scale data operations. The uploaded file path includes a timestamp for easier tracking.

Code Implementation:

```
import boto3

import os

from datetime import datetime



def upload_to_s3(file_path, bucket_name, s3_key_prefix):

 s3_client = boto3.client('s3')

 timestamp = datetime.now().strftime('%Y_%m_%d')

 file_name = os.path.basename(file_path)

 s3_key = f"{s3_key_prefix}/{timestamp}_{file_name}"



 try:

 s3_client.upload_file(file_path, bucket_name, s3_key)

 print(f"File uploaded successfully to S3: {s3_key}")
 except Exception as e:
 print(f"Error uploading file: {e}")

file_path = r"C:\Users\viral\Desktop\project amaron - Copy\amaron_sales_sample_data.xlsx"
bucket_name = "amaron-client-bucket1"
s3_key_prefix = "amaron-fles1"

upload_to_s3(file_path, bucket_name, s3_key_prefix)
```

Downloading Files from S3

The latest file from the S3 bucket is downloaded to a specified local path for further analysis.

Code Implementation:

```
import boto3
import os

def get_latest_file_from_s3(bucket_name, s3_key_prefix, local_download_path):
 s3_client = boto3.client('s3')
 response = s3_client.list_objects_v2(Bucket=bucket_name, Prefix=s3_key_prefix)

 all_files = response.get('Contents', [])
 latest_file = max(all_files, key=lambda x: x['LastModified'])
 file_key = latest_file['Key']
 local_file_path = os.path.join(local_download_path, os.path.basename(file_key))

 try:
 s3_client.download_file(bucket_name, file_key, local_file_path)
 print(f"Downloaded file: {local_file_path}")
 except Exception as e:
```

```
 print(f"Error downloading file: {e}")



bucket_name = "amaron-client-bucket1"

s3_key_prefix = "amaron-fles1"

local_download_path = r"C:\Users\viral\Desktop\PROJECT A"



get_latest_file_from_s3(bucket_name, s3_key_prefix, local_download_path)
```

Sales Data Analysis

Data Overview

The sales data was loaded from an Excel file and thoroughly examined to gain a clear understanding of its structure and contents. This step was crucial for identifying key attributes such as column types, potential relationships between variables, and the overall quality of the data. Insights from this initial exploration laid the foundation for subsequent analyses by highlighting patterns, anomalies, and areas requiring preprocessing, such as handling missing or inconsistent values.

Code Implementation:

```
file_path = "2025_01_22_amaron_sales_sample_data.xlsx"

data = pd.read_excel(file_path)



print("Data Overview:")

print(data.head())



print("\nSummary Statistics:")

print(data.describe())



print("\nMissing Values:")

print(data.isnull().sum())
```

Key Findings:

· **Missing Values:** Rows with missing values were identified and addressed.

· **Revenue Calculation:** A new column, Total Revenue, was created by multiplying unit_price and units_sold.

Revenue Insights

Revenue by Country

Revenue by country was aggregated and visualized to identify key markets contributing to total revenue.

Code Implementation:

```
# Grouping and aggregations

# Total revenue by country

data["Total Revenue"] = data["unit_price"] * data["units_sold"]

revenue_by_country = data.groupby("country")["Total Revenue"].sum()

print("\nRevenue by Country:")

print(revenue_by_country)



# Visualization

plt.figure(figsize=(10, 6))

revenue_by_country.sort_values(ascending=False).plot(kind="bar", color="skyblue")

plt.title("Revenue by Country")

plt.ylabel("Total Revenue")

plt.xlabel("Country")

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```

Revenue by Product Category

Revenue was further broken down by product categories to determine high-performing products.

Code Implementation:

```
# Revenue by Product Category

revenue_by_category = data.groupby("product_category")["Total Revenue"].sum()

print("\nRevenue by Product Category:")

print(revenue_by_category)



# Visualization

plt.figure(figsize=(8, 5))

revenue_by_category.sort_values(ascending=False).plot(kind="bar", color="green")

plt.title("Revenue by Product Category")

plt.ylabel("Total Revenue")

plt.xlabel("Product Category")

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```

Sales Insights

Units Sold by Battery Type

Sales data was analyzed to determine the total units sold by battery type.

Code Implementation:

```
# Units Sold by Battery Type

units_by_battery = data.groupby("battery_type")["units_sold"].sum()

print("\nUnits Sold by Battery Type:")

print(units_by_battery)



# Visualization

plt.figure(figsize=(8, 5))
```

```
units_by_battery.sort_values(ascending=False).plot(kind="bar", color="orange")

plt.title("Units Sold by Battery Type")

plt.ylabel("Total Units Sold")

plt.xlabel("Battery Type")

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```

Summary

This report showcases the following key aspects of our analysis:

1. Leveraging AWS S3 for efficient and reliable file management, ensuring seamless data uploads and downloads.

2. Conducting a thorough exploration of sales data to extract actionable insights that can inform business strategies.

3. Presenting engaging visualizations to highlight trends in revenue and sales across countries, product categories, and battery types, making the insights both accessible and impactful.

The provided code and analysis form a robust foundation for driving data-driven decisions at Amaron.